



TUGAS AKHIR - EF234801

PENGENALAN FONIK MENGGUNAKAN METODE DEEP LEARNING

NAYYA KAMILA PUTRI YULIANTO

NRP 5025211183

Dosen Pembimbing I

Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.

NIP 198412102014042003

Dosen Pembimbing II

Dr. Dwi Sunaryono, S.Kom., M.Kom.

NIP 197205281997021001

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika

Cerdas Institut Teknologi Sepuluh Nopember

Surabaya



TUGAS AKHIR - EF234801

PENGENALAN FONIK MENGGUNAKAN METODE DEEP LEARNING

NAYYA KAMILA PUTRI YULIANTO

NRP. 5025211183

Dosen Pembimbing I

Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.

NIP 198412102014042003

Dosen Pembimbing II

Dr. Dwi Sunaryono, S.Kom., M.Kom.

NIP 19872020120041001

**Program Studi S1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2025**



FINAL PROJECT - EF234801

PHONICS RECOGNITION USING DEEP LEARNING METHODS

NAYYA KAMILA PUTRI YULIANTO

NRP. 5025211183

Advisor I

Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.

NIP 198412102014042003

Advisor II

Dr. Dwi Sunaryono, S.Kom., M.Kom.

NIP 19872020120041001

**Study Program Bachelor Informatics
Department of Informatics Engineering
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya
2025**

LEMBAR PENGESAHAN

PENGENALAN FONIK MENGGUNAKAN METODE DEEP LEARNING

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer pada
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : Nayya Kamila Putri Yulianto
NRP. 5025211183

Disetujui oleh Tim Pengaji Tugas Akhir :

1. Ratih Nur Esti Anggraini, S.Kom., M.Sc., P.hD.

Pembimbing

2. Dr. Dwi Sunaryono, S.Kom., M.Kom.

Ko-pembimbing

3. Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.

Pengaji

4. Dr. Wahyu Suadi, S.Kom, M.Kom.

Pengaji

SURABAYA
Juli, 2025

[Halaman ini sengaja dikosongkan]

APPROVAL SHEET

PHONICS RECOGNITION USING DEEP LEARNING METHODS

FINAL PROJECT

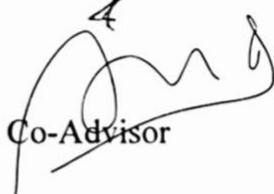
Submitted to fulfill one of the requirements
for obtaining a degree Bachelor of Informatics at
Undergraduate Study Program of Informatics Engineering
Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember

Oleh : Nayya Kamila Putri Yulianto
NRP. 5025211183

Approved by Final Project Examiner Team:

1. Ratih Nur Esti Anggraini, S.Kom., M.Sc., P.hD.

Advisor

2. Dr. Dwi Sunaryono, S.Kom., M.Kom.

Co-Advisor

3. Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.

Examiner

4. Dr. Wahyu Suadi, S.Kom, M.Kom.

Examiner



SURABAYA

July, 2025

[Halaman ini sengaja dikosongkan]

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP : Nayya Kamila Putri Yulianto
Program studi : S-1 Teknik Informatika
Dosen Pembimbing / NIP : Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D. /
198412102014042003
Dosen Ko-pembimbing / NIP : Dr. Dwi Sunaryono, S.Kom., M.Kom. /
197205281997021001

dengan ini menyatakan bahwa Tugas Akhir dengan judul “Pengenalan Fonik Menggunakan Metode Deep Learning” adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

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

Mengetahui
Dosen Pembimbing

Ratih Nur Esti Anggraini, S.Kom., M.Sc., P.hD.
NIP. 198412102014042003

Surabaya, 21 Juli 2025
Mahasiswa

Nayya Kamila Putri Yulianto
NRP. 5025211183

Dosen Ko-pembimbing

Dr. Dwi Sunaryono, S.Kom., M.Kom.
NIP. 197205281997021001

[Halaman ini sengaja dikosongkan]

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP : Nayya Kamila Putri Yulianto
Department : Informatics Engineering
Advisor I / NIP : Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D. /
198412102014042003
Advisor II / NIP : Dr. Dwi Sunaryono, S.Kom., M.Kom. /
197205281997021001

Hereby declare that Final Project with the title of “Phonics Recognition Using Deep Learning Methods” is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Acknowledge,
Advisor



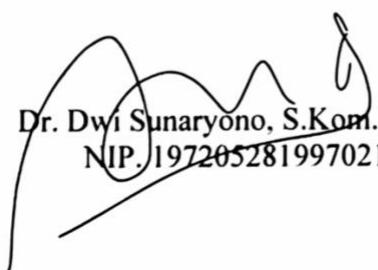
Ratih Nur Esti Anggraini, S.Kom., M.Sc., P.hD.
NIP. 198412102014042003

Surabaya, 21 July 2025
Student



Nayya Kamila Putri Yulianto
NRP. 5025211183

Co-Advisor



Dr. Dwi Sunaryono, S.Kom., M.Kom.
NIP. 197205281997021001

[Halaman ini sengaja dikosongkan]

PERNYATAAN KODE ETIK

PENGGUNAAN AI GENERATIF

Code of Conduct Statement: Generative AI or AI-Assisted Usage

Saya yang bertanda tangan di bawah ini:

I, the undersigned:

Nama Mahasiswa / NRP : Nayya Kamila Putri Yulianto / 5025211183
Full Name / Student ID

Program Studi : S-1 Teknik Informatika
Study Program

Judul Tugas Akhir : Pengenalan Fonik Menggunakan Metode Deep Learning
Final Project Title

dengan ini menyatakan bahwa pada Tugas Akhir dengan judul di atas tersebut:

hereby declare that in the Final Project with the above title:

No.	Pernyataan Statement	(<input checked="" type="checkbox"/>)
1	Saya hanya menggunakan AI generatif sebagai alat bantu untuk memperbaiki tata bahasa. AI generatif tidak digunakan untuk membuat isi Tugas Akhir. <i>I only used generative AI as a tool to improve the readability or language of the text in my Final Project. It was not used to generate a complete text of my work.</i>	<input checked="" type="checkbox"/>
2	Saya telah memeriksa dan/atau memperbaiki seluruh bagian dari Tugas Akhir saya yang dibantu oleh AI generatif agar sesuai dengan baku mutu penulisan karya ilmiah. <i>I have reviewed and refined all aspects of my work that generative AI assists with, ensuring it adheres to the standards of academic writing.</i>	<input checked="" type="checkbox"/>
3	Saya tidak menggunakan AI generatif untuk pembuatan data primer, grafik dan/atau tabel pada Tugas Akhir saya. <i>I did not use generative AI to generate primary data, figures, and/or tables in my work.</i>	<input checked="" type="checkbox"/>
4	Saya telah memberikan atribusi/pengakuan terhadap AI yang digunakan, secara rinci pada suatu bagian pada lampiran. <i>I have acknowledged the use of generative AI in any part of the work in the specific appendix page.</i>	<input checked="" type="checkbox"/>
5	Saya memastikan tidak ada plagiarisme, termasuk hal yang berasal dari penggunaan AI generatif. <i>I have ensured that there is no plagiarism issue in the work, including any parts generated by AI.</i>	<input checked="" type="checkbox"/>

Surabaya, 21 Juli 2025

Mahasiswa



Nayya Kamila Putri Yulianto
NRP. 5025211183

[Halaman ini sengaja dikosongkan]

ABSTRAK

PENGENALAN FONIK MENGGUNAKAN METODE DEEP LEARNING

Nama Mahasiswa / NRP : Nayya Kamila Putri Yulianto / 5025211183

Departemen : **Teknik Informatika FTEIC - ITS**

Dosen Pembimbing : Ratih Nur Esti Anggraini, S.Kom., M.Sc.,Ph.D.
Dr. Dwi Sunaryono, S.Kom., M.Kom.

Abstrak

Pendidikan fonik merupakan dasar penting dalam kemampuan membaca dan mengeja karena mengajarkan hubungan antara bunyi dan huruf. Fonik terbukti meningkatkan kemampuan anak dalam memahami dan memperlancar bacaan dengan membantu mereka mengidentifikasi pola bunyi huruf. Namun, di lingkungan multibahasa seperti Indonesia, penerapan fonik menjadi tantangan akibat keragaman dialek lokal yang memengaruhi pelafalan huruf, khususnya pada penutur *non-native* Bahasa Inggris.

Tugas Akhir ini menggunakan teknologi pengenalan suara berbasis *deep learning* untuk mengidentifikasi fonik. Sebanyak 986 audio dikumpulkan dari 38 penutur dan diproses melalui augmentasi. Fitur suara diekstraksi dengan *Mel Frequency Cepstrum Coefficients* (MFCC) dan *Power-Normalized Cepstral Coefficients* (PNCC), kemudian dilatih menggunakan *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Transformer*. Evaluasi dilakukan menggunakan nilai akurasi dan F1 Score. Hasil terbaik diperoleh dari model RNN-GRU dan ekstraksi fitur PNCC yang mencapai akurasi 94.59% dan F1 Score 0.946. Hasil ini melampaui penelitian sebelumnya yang menggunakan *Support Vector Machine* (SVM) dan ekstraksi fitur MFCC Librosa dengan nilai akurasi 91.95% dan F1 Score 0.9242.

Kata kunci: Pengenalan fonik, Deep learning, Dialek Indonesia

[Halaman ini sengaja dikosongkan]

ABSTRACT

PHONICS RECOGNITION USING DEEP LEARNING METHODS

Student Name / NRP	: Nayya Kamila Putri Yulianto / 5025211183
Department	: Teknik Informatika FTEIC - ITS
Advisor	: Ratih Nur Esti Anggraini, S.Kom., M.Sc.,Ph.D. Dr. Dwi Sunaryono, S.Kom., M.Kom.

Abstract

Phonics education is a fundamental foundation in developing reading and spelling skills by teaching the relationship between sounds and letters. It has been shown to enhance children's reading fluency and comprehension by helping them recognize sound patterns in words. However, in multilingual environments such as Indonesia, implementing phonics presents challenges due to diverse local dialects that affect pronunciation, particularly among non-native English speakers.

This Final Project employs speech recognition technology based on deep learning to identify phonics. A total of 986 audio recordings were collected from 38 speakers and processed through data augmentation methods. Audio features were extracted using Mel-Frequency Cepstral Coefficients (MFCC) and Power-Normalized Cepstral Coefficients (PNCC), then trained using Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Transformer. The models were evaluated using accuracy and F1 Score metrics. The best performance was achieved by the RNN-GRU model with PNCC features, reaching an accuracy of 94.59% and an F1 Score of 0.946. This result outperformed the previous study that employed Support Vector Machine (SVM) with MFCC Librosa features, which obtained an accuracy of 91.95% and an F1 Score of 0.9242.

Keywords: Phonics recognition, deep learning, Indonesian dialect

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Allah SWT. atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul “Pengenalan Fonik Menggunakan Metode Deep Learning”. Tugas Akhir ini tidak mungkin terselesaikan tanpa bantuan dan dukungan dari banyak pihak. Penulis mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT. yang memberikan petunjuk, kemudahan, dan hanya atas pertolongan-Nya penulis dapat menyelesaikan Tugas Akhir ini untuk mencapai gelar strata satu di Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember.
2. Ibu Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D., dan Bapak Dr. Dwi Sunaryono, S.Kom., M.Kom., sebagai pembimbing yang telah memberikan arahan dan masukan selama penyusunan Tugas Akhir ini.
3. Keluarga penulis tersayang, Papa, Mama, Kakak, Fasha, dan kucing-kucing, terima kasih atas dukungan baik secara fisik dan mental, doa, dan kasih sayang yang selalu menyertai di setiap langkah penulis.
4. Bapak/Ibu dosen di Departemen Teknik Informatika yang telah membekali penulis dengan ilmu dan pengetahuan selama masa perkuliahan.
5. Ayyi, Sasa, Aline, Jidan, Tegar, Meng, dan teman-teman lain yang tidak dapat disebutkan satu per satu atas dukungan, semangat, dan segala bentuk bantuan selama penulisan Tugas Akhir.

Penulis berharap Tugas Akhir ini dapat memberikan manfaat dan menjadi salah satu bentuk kontribusi dalam pengembangan ilmu pengetahuan dan teknologi. Penulis juga memohon maaf apabila terdapat kekurangan dan kesalahan dalam penulisan Tugas Akhir ini.

Surabaya, 21 Juli 2025

Nayya Kamila Putri Yulianto

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

LEMBAR PENGESAHAN	i
APPROVAL SHEET	iii
PERNYATAAN ORISINALITAS	v
STATEMENT OF ORIGINALITY	vii
PERNYATAAN KODE ETIK PENGGUNAAN AI GENERATIF	ix
ABSTRAK	xi
ABSTRACT	xiii
KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
DAFTAR TABEL	xxi
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Permasalahan	2
1.3 Batasan Masalah	2
1.4 Tujuan	2
1.5 Manfaat	2
BAB 2 TINJAUAN PUSTAKA	3
2.1 Penelitian Terkait	3
2.2 Fonik	4
2.3 <i>Speech Recognition</i>	5
2.4 <i>Pre-processing Data Audio</i>	6
2.5 Data Augmentasi	6
2.6 Ekstraksi Fitur	7
2.6.1 <i>Mel Frequency Cepstrum Coefficients (MFCC)</i>	7
2.6.2 <i>Power-Normalized Cepstral Coefficients (PNCC)</i>	9
2.7 Klasifikasi	11
2.8 <i>Deep Learning</i>	12
2.8.1 <i>Convolutional Neural Networks (CNN)</i>	13
2.8.2 <i>Recurrent Neural Networks (RNN)</i>	14
2.8.3 <i>Transformer</i>	16
2.9 Evaluasi Matrix	17
BAB 3 METODOLOGI	19
3.1 Metodologi Pengembangan	19

3.1.1	<i>Dataset</i>	20
3.1.2	<i>Pre-processing</i> Data	21
3.1.3	Pembuatan Model.....	23
3.2	Peralatan Pendukung	23
3.3	Implementasi	23
3.3.1	<i>Dataset</i>	24
3.3.2	<i>Data Pre-processing</i>	24
3.3.3	Implementasi <i>Convolutional Neural Network</i>	30
3.3.4	Implementasi <i>Recurrent Neural Network</i>	32
3.3.5	Implementasi <i>Transformer</i>	33
3.3.6	Evaluasi	35
3.3.7	Desain Antarmuka	36
BAB 4 HASIL DAN PEMBAHASAN		37
4.1	Pengumpulan <i>Dataset</i>	37
4.2	Hasil Eksperimen A	38
4.3	Hasil Eksperimen B	41
4.4	Hasil Eksperimen C	44
4.5	Hasil Eksperimen D	47
4.6	Hasil Eksperimen E	50
4.7	Diskusi	57
4.8	Hasil Antarmuka Website Pengujian.....	59
BAB 5 KESIMPULAN DAN SARAN		61
5.1.	Kesimpulan	61
5.2.	Saran	61
LAMPIRAN		63
DAFTAR PUSTAKA		91
BIODATA PENULIS		95

DAFTAR GAMBAR

Gambar 2.1 Struktur <i>Speech Recognition</i>	5
Gambar 2.2 Tahapan Metode MFCC	7
Gambar 2.3 Tahapan Metode PNCC	10
Gambar 2.4 Cakupan <i>Deep Learning</i>	12
Gambar 2.5 Contoh Struktur CNN	14
Gambar 2.6 Arsitektur <i>Vanilla RNN</i>	15
Gambar 2.7 Arsitektur <i>Transformer</i>	16
Gambar 2.8 <i>Confusion Matrix</i>	17
Gambar 3.1 Diagram Alur Pembuatan Model.....	19
Gambar 3.2 Diagram Alur <i>Website</i>	19
Gambar 3.3 Contoh Implementasi Arsitektur CNN	30
Gambar 3.4 Contoh Implementasi Arsitektur RNN	32
Gambar 3.5 Contoh Implementasi Arsitektur Transformer....	34
Gambar 3.6 Desain Antarmuka Pengguna	36
Gambar 4.1 Distribusi Kelas	37
Gambar 4.2 Distribusi <i>Speaker</i>	38
Gambar 4.3 Visualisasi Data Augmentasi Eksperimen A	39
Gambar 4.4 Visualisasi MFCC Eksperimen A.....	40
Gambar 4.5 Visualisasi Data Augmentasi Eksperimen B	42
Gambar 4.6 Visualisasi PNCC Eksperimen B.....	43
Gambar 4.7 Visualisasi Data Augmentasi Eksperimen C	45
Gambar 4.8 Visualisasi PNCC Eksperimen C.....	46
Gambar 4.9 Visualisasi Data Augmentasi Eksperimen D	48
Gambar 4.10 Visualisasi PNCC Eksperimen D	49
Gambar 4.11 Visualisasi Data Augmentasi Eksperimen E	51
Gambar 4.12 Visualisasi PNCC Eksperimen E.....	52
Gambar 4.13 Hasil Pengujian Huruf N	59
Gambar 4.14 Hasil Pengujian Huruf C.....	60

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 2.1 Penelitian Terkait.....	3
Tabel 3.1 <i>Dataset</i>	20
Tabel 3.2 Perangkat Keras	23
Tabel 3.3 Perangkat Lunak.....	23
Tabel 4.1 Skenario Eksperimen.....	37
Tabel 4.2 Hasil Eksperimen A.....	40
Tabel 4.3 Hasil Eksperimen B	43
Tabel 4.4 Hasil Eksperimen C	47
Tabel 4.5 Hasil Eksperimen D.....	49
Tabel 4.6 Parameter Terbaik CNN Eksperimen E	53
Tabel 4.7 Hasil CNN Eksperimen E.....	53
Tabel 4.8 Parameter Terbaik RNN Eksperimen E	54
Tabel 4.9 Hasil RNN Eksperimen E.....	55
Tabel 4.10 Parameter Terbaik <i>Transformer</i> Eksperimen E.....	56
Tabel 4.11 Hasil <i>Transformer</i> Eksperimen E	56

[Halaman ini sengaja dikosongkan]

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Fonik merupakan sistem pembelajaran literasi yang menekankan hubungan antara huruf dan bunyi bahasa berdasarkan prinsip alfabet. Metode ini mengajarkan bagaimana huruf-huruf alfabet mewakili bunyi lisan dalam Bahasa Inggris, sehingga anak dapat memahami pelafalan kata secara tepat (Westhisi, 2019). Penelitian menunjukkan bahwa fonik adalah kunci untuk mengembangkan kemampuan membaca sehingga anak dapat mengenali pola huruf dan bunyi serta meningkatkan kelancaran membaca dan pemahaman (Ehri, 2020). Namun, di lingkungan multibahasa seperti Indonesia, di mana anak-anak dihadapkan dengan berbagai macam bahasa dan dialek, pengenalan fonik menjadi rumit (Utami & Musthafa, 2023). Tugas Akhir ini bertujuan untuk mengkaji bagaimana perbedaan dialek mempengaruhi pengenalan fonik, terutama untuk orang bukan penutur asli Bahasa Inggris yang berbicara dengan dialek asli Indonesia, seperti Bali, Batam, Jawa, Madura dan Papua.

Penelitian sebelumnya oleh Razak (2024) merupakan Tugas Akhir yang telah mengeksplorasi pengenalan fonik, tetapi masih terdapat kekurangan, terutama dalam pengumpulan variasi *dataset* lokal Indonesia. Penelitian yang lalu menggunakan model pembelajaran mesin *Support Vector Machines* (SVM), *K-Nearest Neighbors* (KNN), dan *Random Forest*, serta *Mel-Frequency Cepstral Coefficients* (MFCC) sebagai ekstraksi fitur. Penelitian ini berfokus pada model pembelajaran mesin yang terbatas dan berpotensi melewatkannya metode yang lebih canggih, seperti *deep learning*, untuk meningkatkan pengenalan dan klasifikasi fonik pada berbagai dialek (Mehrish et al., 2023).

Dalam aplikasi terkait seperti *voice* dan *speech recognition*, metode *deep learning* seperti *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Transformer* telah menunjukkan performa yang baik. Contohnya, CNN unggul dalam menangkap pola lokal sehingga lebih tahan terhadap *noise* (Mehrish et al., 2023). Sementara itu, RNN terbukti efektif untuk mengenali pola temporal yang sangat penting dalam pemrosesan ucapan mengingat sifat audio yang bersifat *sequential* (Z. Zhang et al., 2020). Pham et al. (2019) memperkenalkan arsitektur *Transformer end-to-end* untuk pengenalan suara dan menunjukkan bahwa model ini mampu mengungguli model konvensional seperti TDNN-LSTM. Dengan arsitektur yang lebih dalam, *self-attention* terbukti efektif dalam menangkap pola kompleks dari sinyal audio dan meningkatkan performa sistem dalam *Automatic Speech Recognition* (ASR). Dengan beberapa hasil ini, metode *deep learning* memberikan alternatif yang menjanjikan untuk meningkatkan pengenalan fonik, terutama dalam konteks dialek yang beragam.

Tugas Akhir ini menerapkan metode *deep learning*, khususnya CNN, RNN, dan *Transformer* untuk mengatasi kekurangan yang ditemukan dalam studi pengenalan fonik sebelumnya. Dengan mengintegrasikan model-model ini, penelitian ini berupaya untuk memahami lebih dalam bagaimana dialek lokal memengaruhi pengenalan fonik serta memberikan wawasan yang dapat bermanfaat untuk pendidikan bahasa dan teknologi pengenalan suara. Hasil dari penelitian ini diharapkan dapat membantu pendidik dalam mengembangkan strategi pembelajaran literasi yang lebih baik dan meningkatkan teknologi pengenalan suara dengan mengakomodasi lebih banyak dialek.

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana proses pengembangan model *deep learning* untuk pengenalan fonik?
2. Bagaimana pengaruh metode *pre-processing*, khususnya perbandingan antara MFCC Librosa dan PNCC, terhadap hasil klasifikasi fonik?
3. Bagaimana perbedaan performa model *deep learning* dalam mengklasifikasi fonik dari penutur *native* dan *non-native*?
4. Bagaimana variasi dialek lokal Indonesia memengaruhi kinerja model *deep learning* pada pengenalan ejaan fonik?

1.3 Batasan Masalah

Topik yang dibahas dalam Tugas Akhir ini memiliki beberapa keterbatasan sebagai berikut:

1. Tugas Akhir ini menerapkan model *deep learning Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Transformer* yang kemudian digabungkan dengan *Mel-Frequency Cepstral Coefficient* (MFCC) dan *Power Normalized Cepstral Coefficient* (PNCC) untuk ekstraksi fitur.
2. *Dataset* terdiri dari sampel audio sejumlah dialek Indonesia yang terbatas (Bali, Batam, Jawa, Madura, dan Papua), sehingga dapat membatasi proses pelatihan dan memengaruhi kinerja model.
3. Penelitian ini akan fokus pada pengenalan fonik. Aspek lain dari *speech processing* seperti pengenalan kata atau *sentence-level comprehension*, tidak termasuk dalam cakupan penelitian ini.

1.4 Tujuan

Tujuan dari Tugas Akhir ini antara lain:

1. Mengeksplorasi bagaimana variasi dialek lokal Indonesia memengaruhi kinerja pengenalan fonik menggunakan model *deep learning*.
2. Menganalisis hasil evaluasi pengenalan fonik antara *native* dan *non-native speaker*.
3. Mengidentifikasi dampak *pre-processing* data terhadap kinerja model dalam pengenalan fonik.
4. Membandingkan akurasi berbagai model *deep learning* dalam mendekripsi huruf fonik, serta menentukan model mana yang menunjukkan performa terbaik untuk Tugas Akhir ini.

1.5 Manfaat

Tugas Akhir ini bertujuan untuk meningkatkan pengenalan fonik melalui penerapan metode *deep learning*, terutama dalam konteks menangani beragam dialek Indonesia. Studi ini membandingkan model CNN, RNN, dan *Transformer* untuk mengidentifikasi pendekatan terbaik dalam membedakan suara fonik Bahasa Indonesia. Hasil penelitian ini diharapkan dapat berkontribusi pada pengembangan media pembelajaran fonik yang lebih adaptif terhadap keragaman dialek, serta meningkatkan kualitas pengajaran dan literasi di lingkungan multibahasa.

BAB 2 TINJAUAN PUSTAKA

Pada bab ini, akan dibahas literatur dari penelitian yang terkait dengan topik yang dikerjakan, serta penjelasan mengenai dasar teori yang digunakan dalam pembuatan Tugas Akhir ini.

2.1 Penelitian Terkait

Tabel 2.1 berisi ringkasan singkat dari beberapa penelitian terkait. Studi pertama adalah “*Comparative Analysis of CNN and RNN for Voice Pathology Detection*” oleh Lenson & Airlangga, diikuti oleh “*Comparative Analysis of MLP, CNN, and RNN Models in Automatic Speech Recognition: Dissecting Performance Metric*” oleh Syed *et al.* Selanjutnya, terdapat studi “*Power-Normalized Cepstral Coefficients (PNCC) For Robust Speech Recognition*” yang ditulis oleh Kim & Stern, serta “*An Ensemble Learning Method for Dialect Classification*” oleh Ye *et al.* Studi berikutnya adalah “*A Review of Deep Learning Techniques for Speech Processing*” yang ditulis oleh Mehrish *et al.*, dan terakhir adalah “*Introducing Alphabet Phonics Spelling Using Machine Learning with MFCC Extraction*” oleh Razak.

Tabel 2.1 Penelitian Terkait

No	Penulis, Tahun	Deskripsi
1.	<i>Comparative Analysis of MLP, CNN, and RNN Models in Automatic Speech Recognition: Dissecting Performance Metric</i> (Lenson & Airlangga, 2023)	Studi ini membandingkan performa CNN dan RNN dalam mendekripsi patologi suara menggunakan <i>Saarbrücken Voice Database</i> yang berisi rekaman suara dari lebih dari 2000 individu dengan lebih dari 72 kondisi patologis. Penelitian ini menunjukkan CNN mencapai akurasi 87,11%, sedangkan RNN memiliki akurasi yang sebanding. Keterbatasan penelitian ini terletak pada ukuran <i>dataset</i> dan kualitas rekaman yang berdampak pada akurasi. Saran untuk penelitian selanjutnya adalah fokus pada peningkatan keragaman <i>dataset</i> dan penyempurnaan metode ekstraksi fitur untuk meningkatkan performa model.
2.	<i>Comparative Analysis of CNN and RNN for Voice Pathology Detection</i> (Syed <i>et al.</i> , 2021)	Penelitian ini membandingkan performa CNN dan RNN-LSTM dalam mendekripsi patologi suara menggunakan dataset SVD. Hasil validasi 10-fold menunjukkan bahwa CNN mencapai akurasi 87.11%, sementara RNN-LSTM sebesar 86.52%. Studi ini juga mengutip penelitian sebelumnya yang melaporkan akurasi MLP hingga 100% untuk suara perempuan dan 95% untuk laki-laki.
3.	<i>Power-Normalized Cepstral Coefficients (PNCC) For Robust Speech Recognition</i> (Kim & Stern, 2016)	Studi ini menyatakan PNCC sebagai ekstraksi fitur baru untuk pemrosesan suara. Beberapa fitur utama dari PNCC meliputi penggunaan <i>power-law-nonlinearity</i> yang menggantikan <i>log</i>

		<i>linearity</i> lama yang digunakan dalam koefisien MFCC. Hasil dari penelitian ini menunjukkan bahwa PNCC mengungguli MFCC dan PLP, terutama pada kondisi bising dan <i>reverberant</i> . Selain itu, hasil PNCC menunjukkan kompleksitas komputasinya sedikit lebih tinggi dibanding MFCC.
4.	<i>An Ensemble Learning Method for Dialect Classification</i>	(Ye et al., 2019) Penelitian ini menggunakan <i>dataset</i> yang terdiri dari 500 sampel fonetik dari enam dialek Tiongkok untuk mengevaluasi berbagai model <i>deep learning</i> , yaitu LSTM, GRU, CNN, DNN, dan model gabungan yang mengombinasikan GRU, CNN, dan DNN. Hasilnya, model gabungan menunjukkan performa terbaik dengan akurasi sebesar 85,29% dalam klasifikasi enam dialek, mengungguli model individu seperti LSTM (83,42%), GRU (78,30%), CNN (74,02%), dan DNN (72,08%).
5.	<i>A Review of Deep Learning Techniques for Speech Processing</i>	(Mehrish et al., 2023) Studi ini membahas representasi <i>self-supervised</i> untuk <i>speech processing</i> . Model <i>deep learning</i> yang digunakan seperti RNN, CNN, dan <i>Transformer</i> dieksplorasi dalam penelitian ini. Selain itu, studi ini menyoroti pentingnya penggunaan <i>dataset</i> yang besar untuk <i>Automatic Speech Recognition</i> (ASR). Teknik <i>speaker separation</i> dan <i>acoustic feature extraction</i> juga turut dibahas.
6	<i>Introducing Alphabet Phonics Spelling Using Machine Learning with MFCC Extraction</i>	(Razak, 2024) Penelitian ini mengevaluasi tiga model <i>machine learning</i> : <i>Support Vector Machine</i> (SVM), <i>K-Nearest Neighbor</i> (KNN), dan <i>Random Forest</i> . <i>Dataset</i> dikumpulkan dari sepuluh <i>native speaker</i> dan enam <i>non-native speaker</i> dari Jawa dan Batam. Model SVM dengan augmentasi data konstan dan ekstraksi fitur MFCC menggunakan <i>library</i> Librosa menghasilkan performa terbaik dengan akurasi tertinggi yang dicapai adalah 91,95%. Kinerja model dipengaruhi oleh jumlah data, metode pengumpulan data, dan pelafalan huruf. Augmentasi data dan pemilihan metode ekstraksi fitur yang tepat mempunyai peran penting dalam meningkatkan akurasi model.

2.2 Fonik

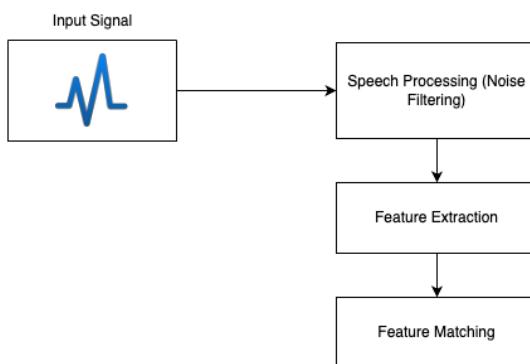
Fonik adalah metode membaca yang berfokus pada hubungan antara huruf dan suara. Meskipun huruf-huruf terlihat sama secara tulisan, bunyi yang dihasilkannya bisa berbeda tergantung bagaimana huruf itu muncul dalam kata. Dalam fonik, bunyi ini bisa muncul sebagai

bagian dari suku kata, bunyi awal (onset), bunyi akhir (rima), atau fonem. Jadi, hubungan antara huruf dan bunyi tidak selalu tetap, melainkan tergantung konteks kata dan struktur bunyinya. (Prayogo & Widyaningrum, 2017). Metode ini membantu anak memahami bagaimana huruf bergabung untuk membentuk suara dalam kata yang menjadi dasar untuk kemampuan membaca dan mengeja.

Instruksi fonik menggunakan beberapa teknik, seperti mengenali huruf, mengucapkan suara, dan menggabungkan suara untuk membentuk kata. Siswa belajar untuk mengenali pola bahasa yang membantu mereka dalam membaca dan mengeja. Metode ini sering melibatkan aktivitas menarik dan interaktif, seperti permainan kata dan latihan membaca sederhana, yang dirancang untuk menjaga minat siswa serta meningkatkan keterlibatan mereka dalam proses pembelajaran (Syarif et al., 2020).

2.3 *Speech Recognition*

Speech recognition adalah teknologi yang dapat membuat sistem komputer untuk mengidentifikasi dan menginterpretasikan ucapan manusia. Teknologi ini bekerja dengan menganalisis pola unik suara seseorang, mengubahnya ke dalam format digital, dan membandingkannya dengan basis data yang telah dipelajari oleh sistem. Kata-kata yang diucapkan diubah menjadi sinyal digital melalui proses konversi gelombang suara menjadi urutan numerik yang kemudian dicocokan dengan kode tertentu untuk mengenali kata-kata yang diucapkan (Tridarma & Endah, 2020).



Gambar 2.1 Struktur *Speech Recognition*

Menurut (Chavan & Gawande, 2015), terdapat empat tahapan komputasi dalam *speech recognition* seperti yang ditunjukkan pada Gambar 2.1, yaitu:

A. *Speech Signal*

Tahap pertama adalah menangkap sinyal suara, di mana suara yang diucapkan diubah menjadi sinyal digital yang dapat diproses oleh komputer. Pada tahap ini, gelombang suara direkam untuk diproses lebih lanjut.

B. *Speech Pre-processing*

Sinyal suara yang sudah diubah melewati tahap *pre-processing*, seperti penghilang *noise*, normalisasi sinyal, dan memotong bagian yang tidak relevan. Hal ini bertujuan untuk meningkatkan kualitas sinyal yang akan diekstraksi fiturnya.

C. Features Extraction

Tahap ini mencakup proses ekstraksi fitur penting dari sinyal suara, seperti *Linear Prediction Cepstrum Coefficient* (LPCC), *Mel-Frequency Cepstral Coefficient* (MFCC) dan *Power Normalized Cepstrum Coefficient* (PNCC). Fitur-fitur ini membantu sistem dalam mengenali pola-pola penting yang mewakili karakteristik suara yang diucapkan.

D. Feature Matching

Pada tahap ini, fitur-fitur yang sudah diekstraksi dibandingkan atau dicocokan dengan model yang ada untuk mengidentifikasi kata-kata yang diucapkan. Hasil dari tahap ini berupa output dalam bentuk kata atau frasa yang dikenali oleh sistem.

2.4 Pre-processing Data Audio

Pre-processing data audio melibatkan beberapa langkah penting untuk meningkatkan kualitas sinyal audio agar siap dianalisis. Langkah-langkah ini memastikan bahwa data yang digunakan bersih dan pantas untuk diproses lebih lanjut, sehingga hasil analisis bisa lebih akurat. Dengan *pre-processing* yang baik, performa model untuk pengenalan ucapan, deteksi emosi, hingga identifikasi fonik bisa meningkat secara signifikan.

Normalisasi audio berperan penting dalam *pre-processing* karena membantu menyeragamkan amplitudo sinyal dari berbagai rekaman audio. Teknik ini bermanfaat terutama ketika data berasal dari banyak sumber atau penutur yang berbeda. Normalisasi data dilakukan agar setiap fitur dalam data *train* memberikan kontribusi yang seimbang dalam proses prediksi. Tahap ini penting untuk mencegah dominasi fitur tertentu dan membantu peningkatan akurasi serta kemampuan generalisasi model (Yağanoğlu & Köse, 2018).

Selanjutnya, *data cleaning* juga menjadi tahap penting untuk memastikan kualitas *dataset*. *Cleaning* ini mencakup penanganan masalah seperti nilai yang hilang yang dapat memengaruhi hasil analisis. Selain itu, proses ini juga melibatkan penghapusan bagian audio yang tidak relevan seperti *silence* di awal atau akhir, serta *noise reduction* untuk meminimalkan suara yang tidak diinginkan. Dengan data yang bersih dan konsisten, model memiliki peluang lebih besar untuk belajar pola dengan akurasi yang lebih baik (Zhang & Sun, 2020).

2.5 Data Augmentasi

Data augmentasi adalah kumpulan metode yang merupakan versi terbaru dari *dataset* yang sudah ada. Metode ini menghasilkan berbagai variasi data asli dalam jumlah besar dan dapat berfungsi sebagai *regularizer* untuk mengurangi masalah *overfitting* (Fadillah et al., 2021). Terdapat beberapa metode audio *deformation* yang dapat digunakan untuk data augmentasi, yaitu *pitch shifting*, *dynamic range compression*, *time stretching*, dan *speed perturbation* (Esa et al., 2020). Dua metode augmentasi data yang digunakan dalam Tugas Akhir ini adalah *pitch shifting* dan *speed perturbation*. *Pitch shifting* adalah teknik yang mengubah nada atau ketinggian suara dalam sinyal audio untuk menciptakan variasi pada data. Tujuan dari *pitch shifting* adalah untuk meningkatkan kemampuan model dalam mengenali pola suara meskipun terjadi perubahan nada. Sementara itu, *speed perturbation* melibatkan perubahan kecepatan pemutaran sinyal audio, baik lebih cepat maupun lebih lambat, untuk menghasilkan variasi data yang membantu meningkatkan kemampuan generalisasi model.

Proses ini dilakukan dengan melakukan *resampling* sinyal audio dalam domain waktu (Geng et al., 2022).

2.6 Ekstraksi Fitur

Ekstraksi fitur merupakan prosedur dasar dalam berbagai bidang, terutama *speech processing* dan *recognition*. Proses ini melibatkan transformasi data mentah menjadi sekumpulan karakteristik yang dapat diterapkan secara efektif pada pengenalan, klasifikasi, atau analisis lanjutan lainnya. Tujuan dari ekstraksi fitur dalam pemrosesan suara adalah untuk mengurangi kompleksitas data dan tetap menangkap elemen-elemen utama dari sinyal *speech*. Langkah ini sangat penting untuk meningkatkan efisiensi dan akurasi sistem pengenalan ucapan, terutama saat dilatih dengan *dataset* besar atau aplikasi *real-time*.

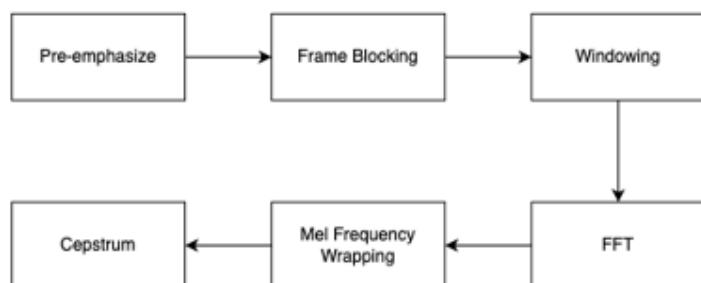
Dalam pengenalan suara, ekstraksi fitur digunakan untuk mengambil informasi penting dari sinyal suara sekaligus menghilangkan data yang tidak relevan yang dapat memengaruhi kinerja model. Beberapa metode yang sering digunakan antara lain *Linear Prediction Cepstral Coefficient* (LPCC), *Mel-Frequency Cepstral Coefficients* (MFCC), dan *Power Normalized Cepstral Coefficient* (PNCC). Setiap metode mempunyai kelebihan dan kekurangannya sehingga dapat disesuaikan penggunaannya untuk berbagai aplikasi *speech processing*. Studi oleh Kim & Stern membuktikan bahwa hasil akurasi PNCC lebih baik dibandingkan MFCC dan RASTA-PLP (C. Kim & Stern, 2016).

2.6.1 Mel Frequency Cepstrum Coefficients (MFCC)

Mel-Frequency Cepstral Coefficients (MFCC) adalah koefisien yang merepresentasikan data audio dan menjadi metode standar dalam pengenalan suara sejak diperkenalkan oleh Davis dan Mermelstein pada tahun 1980-an. Proses ekstraksi fitur ini mengubah data suara menjadi data berbentuk spektrum gelombang, mirip seperti gambar, sehingga sistem pengenalan suara dapat bekerja lebih akurat dalam berbagai kondisi (Chamidy, 2016). Tahapan MFCC dirancang menyerupai sistem pendengaran manusia saat menangkap sinyal suara, sehingga hasil ekstraksi fiturnya mendekati persepsi yang dihasilkan oleh indra pendengaran manusia. MFCC mampu menghasilkan data dengan ukuran minimal tanpa kehilangan informasi penting dari sinyal suara, menjadikannya lebih unggul dibandingkan metode lain (Sasilo et al., 2022).

Menurut Maurya et al., proses MFCC terdiri dari beberapa tahap, yaitu *pre-emphasis*, *frame blocking*, *windowing*, *fast fourier transform*, *mel frequency wrapping*, dan *cepstrum*. Gambar 2.3 menunjukkan langkah-langkah metode MFCC (Maurya et al., 2018).

Dari Gambar 2.2, terdapat enam tahapan pada MFCC dengan fungsinya masing-masing:



Gambar 2.2 Tahapan Metode MFCC

A. Pre-emphasize

Pre-emphasis adalah langkah awal dalam perhitungan MFCC, di mana sebuah filter diterapkan untuk meningkatkan energi dari frekuensi tinggi. Proses ini penting karena pada sinyal asli, biasanya energi frekuensi rendah cukup banyak dan energi frekuensi tingginya sedikit. Hal ini terjadi karena sinyal frekuensi rendah *di-sampling* dengan frekuensi *sampling* yang cukup tinggi sehingga menghasilkan nilai numerik sama. Pada Persamaan 2.1, $y'(n)$ merepresentasikan sinyal setelah diterapkan filter *pre-emphasis*, dengan $x(n)$ sebagai sinyal masukan dan α sebagai koefisien *pre-emphasis* yang menentukan tingkat amplifikasi frekuensi tinggi (biasanya antara 0,9 hingga 0,98). Filter ini umumnya berupa *high-pass filter* orde pertama.

$$y'(n) = x(n) - \alpha * x(n - 1) \quad (2.1)$$

Langkah ini membantu memperjelas sinyal dan meningkatkan efektivitas proses di tahap-tahap berikutnya.

B. Frame Blocking

Frame blocking adalah proses membagi sinyal audio menjadi blok-blok kecil (frame) yang antara frame satu dengan yang lain terjadi tumpang tindih atau *overlapping*. Sinyal suara dilakukan segmentasi menjadi beberapa frame dengan *overlap* agar tidak ada sinyal yang hilang (Helmiyah et al., 2018). Setiap frame biasanya berdurasi antara 20 ms hingga 40 ms. Tahap ini bertujuan membagi sinyal suara menjadi frame kecil dengan sampel yang cukup untuk mendapatkan informasi yang cukup. Apabila ukuran frame terlalu kecil, informasi yang didapat tidak cukup akurat. Sedangkan apabila ukuran frame terlalu besar, informasi dalam frame sering berubah-ubah (Prayogi, 2015).

C. Windowing

Proses *windowing* dilakukan untuk tiap frame yang bertujuan untuk meminimalkan ketidaksinambungan pada awal dan akhir frame. Fungsi *window* menghasilkan sinyal yang dikonsinyu. Salah satu cara menghindari dikontinyu pada ujung *window* yaitu sinyal diruncingkan menjadi nol atau dekat dengan nol sehingga dapat mengurangi kesalahan. Pada MFCC, metode *window* yang sering digunakan adalah *hamming window* (Prayogi, 2015).

D. Fast Fourier Transform (FFT)

Setiap frame yang sudah melalui *windowing* diproses menggunakan *Fast Fourier Transform* (FFT) untuk mengubah sinyal dari *time domain* ke *frequency domain*. Dengan menghitung *Discrete Fourier Transform* (DFT) secara efisien, FFT memungkinkan analisis komponen frekuensi dalam sinyal audio. Langkah ini mengubah tiap N sampel dari domain waktu ke domain frekuensi. Dalam pemrosesan suara, *Fast Fourier Transform* berguna untuk mengubah konvolusi getaran celah suara dan respon gelombang saluran suara dalam domain waktu (Chamidy, 2016).

E. Mel Frequency Wrapping

Setelah audio melewati tahap FFT, proses selanjutnya adalah tahap *Mel filterbank*. Tahap ini bertujuan meniru sistem pendengaran manusia terhadap frekuensi yang berbeda, membantu menangkap informasi akustik dari sinyal audio dan dibuat agar lebih mudah dipahami manusia.

Rumus *mel-filterbank* dapat dilihat pada Persamaan 2.2 dan 2.3.

$$M(f) = 1.125 \ln\left(1 + \frac{f}{700}\right) \quad (2.2)$$

$$f = 700 * \left(\exp\left(\frac{m}{1.125}\right) - 1\right) \quad (2.3)$$

Tahapan ini dilakukan untuk mementukan batas atas dan bawah dari filter. Proses dimulai dengan mendefinisikan rentang frekuensi minimum dan maksimum dalam Hertz yang kemudian dikonversi ke *Mel scale* menggunakan Persamaan 2.2, di mana $M(f)$ adalah frekuensi dalam skala Mel dan (f) adalah frekuensi dalam Hertz pada Persamaan 2.3. Lalu, *Mel scale* dibagi menjadi titik-titik yang terdistribusi merata yang berfungsi sebagai frekuensi pusat untuk triangular filters. *Mel scale* kemudian dikonversi kembali ke skala linier. *Triangular scale* dibuat berdasarkan titik-titik yang diperoleh dan bentuknya meningkat secara linear seiring dengan magnitudonya. Selanjutnya, tiap *triangular scale* dinormalisasi dengan membagi magnitudonya dengan jumlah semua *magnitude* dalam filter. Proses ini menghasilkan filter bank yang terdiri dari sekumpulan *triangular filters*, di mana tiap filter mewakili rentang frekuensi tertentu pada *Mel scale* (Razak, 2024).

F. Discrete Cosine Transform (DCT)

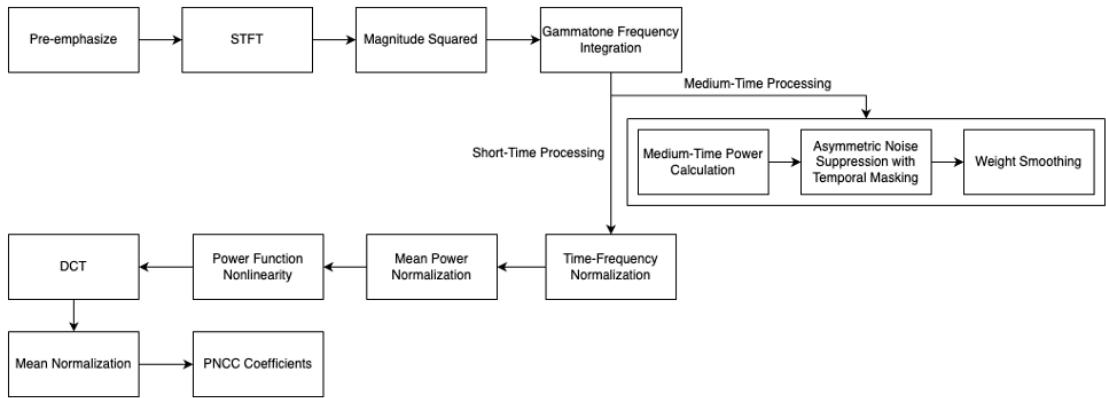
Langkah terakhir dalam perhitungan MFCC adalah *Discrete Cosine Transform* (DCT), yaitu proses perubahan konversi dari domain frekuensi ke domain waktu menggunakan DCT. Proses ini dilakukan untuk mendapatkan nilai koefisien dari hasil perkalian *mel-filterbank* yang telah dikonversi ke domain waktu. Proses ini akan menghasilkan log dari perkalian DCT yang telah diubah ke domain waktu. Hasil log perkalian domain waktu ini menghasilkan *mel-frequency cepstrum coefficient* (MFCC). Persamaan yang digunakan dapat dilihat pada Persamaan 2.4, di mana $j = 1, 2, 3, \dots, K$ adalah koefisien dan M adalah jumlah filter (Helmiyah et al., 2018).

$$C_j = \sum_{i=1}^M X_i \cos\left(j(i-2)/2 \frac{\pi}{M}\right) \quad (2.4)$$

2.6.2. Power-Normalized Cepstral Coefficients (PNCC)

PNCC pertama kali dikenalkan oleh Kim dan Stern pada penelitian berjudul *Power-Normalized Cepstral Coefficients (PNCC) for Robust Speech Recognition* dengan hasil yang menunjukkan hasil akurasi PNCC lebih tinggi dibandingkan dengan MFCC (Kim & Stern, 2016). PNCC menggunakan *filter bank* yang disebut *gammatone filter-bank* dan *power-law nonlinearity* sebagai pengganti *Mel filter-bank log nonlinearity* yang digunakan dalam framework MFCC. Selain itu, PNCC menggunakan pendekatan *medium-duration power bias subtraction* untuk mengurangi *noise* dengan didasarkan pada perbandingan antara *arithmetic mean* dan *geometric mean* dari energi sinyal.

Menurut Kim dan Stern, tahapan PNCC terdiri dari beberapa langkah, yaitu *pre-emphasis*, *Short-time Fourier Transform* (STFT), *magnitude squared*, *gammatone frequency integration*, *medium-time* atau *short-time processing*, *mean power normalization*, *power function nonlinearity*, DCT, *mean normalization*, dan PNCC *coefficient*. Gambar 2.3 menunjukkan langkah-langkah metode PNCC dengan penjelasan sebagai berikut:



Gambar 2.3 Tahapan Metode PNCC

A. *Pre-emphasize*

Tahap pertama dari pemrosesan sinyal suara adalah *pre-emphasize*. Pada tahapan ini, sinyal suara di-filter agar memperkuat komponen frekuensi tinggi dengan persamaan berikut.

$$H(z) = 1 - 0.97z^{-1} \quad (2.2)$$

Pre-emphasize dilakukan dengan tujuan meningkatkan rasio *signal-to-noise* pada frekuensi tinggi dan mengurangi efek *noise background*.

B. *Short-Time Fourier Transform (STFT)*

Tahapan ini bertujuan untuk mengubah sinyal suara dari domain waktu menjadi domain frekuensi. Pada tahap ini, sinyal dibagi menjadi beberapa *frame* pendek menggunakan *windowing* dengan durasi sekitar 20-30 ms dan *overlap* sekitar 50%. Kemudian, setiap frame dianalisis menggunakan domain frekuensi menggunakan *Discrete Fourier Transform* (DFT) yang menghasilkan spektrum frekuensi dari setiap frame sinyal.

C. *Magnitude Squared*

Setelah memperoleh hasil spektrum frekuensi dari STFT, langkah berikutnya adalah menghitung *magnitude squared* dari spektrum. *Magnitude squared* merupakan hasil kuadrat dari *magnitude* spektrum frekuensi (tanpa fase) yang menggambarkan energi pada setiap frekuensi dalam *frame*. Sinyal suara di-filter untuk memperkuat komponen frekuensi tinggi dengan persamaan 2.5.

$$|X(f)|^2 \quad (2.5)$$

Dengan $X(f)$ adalah spektrum frekuensi pada *frame*. Tahapan ini merupakan representasi energi dari sinyal yang lebih relevan untuk pemrosesan selanjutnya.

D. *Gammatone Frequency Integration*

Setelah mendapatkan nilai *magnitude squared*, sinyal diproses dengan *gammatone filter-bank* yang meniru pendengaran manusia dalam mendeteksi suara. *Filter gammatone* terdiri dari beberapa kanal frekuensi yang mendistribusikan energi sinyal di sepanjang spektrum frekuensi. *Filter-bank* ini memungkinkan pemrosesan yang lebih alami sesuai dengan pendengaran manusia.

E. Medium-Time or Short-Time Processing

Pada tahap ini, analisis dilakukan pada jangka waktu yang lebih panjang untuk menangkap perubahan dalam sinyal yang lebih lambat. Proses ini melibatkan analisis dengan kerangka waktu yang lebih panjang (*medium-time frames*) yang dapat berlangsung sekitar 70-120 ms. Tujuan dari analisis waktu medium adalah mengurangi efek dari *noise background* yang stabil dan mendeteksi perubahan dalam sinyal yang lebih signifikan.

F. Mean Power Normalization

Setelah sinyal diproses dengan *filter gammatone*, langkah selanjutnya adalah normalisasi daya (*mean power normalization*). Pada tahap ini, daya sinyal dinormalisasi berdasarkan rata-rata energi dalam *frame* waktu tertentu. Tujuan dari normalisasi ini adalah mengurangi ketergantungan pada *level* energi sinyal dan memastikan bahwa fitur yang dihasilkan lebih tahan terhadap fluktuasi energi yang tidak relevan dalam sinyal.

G. Power Function Nonlinearity

Setelah normalisasi daya, sinyal melewati fungsi nonlinier berbasis hukum pangkat (*power-law nonlinearity*). Fungsi ini menggantikan penggunaan fungsi logaritmik yang biasa digunakan dalam MFCC. Rumus untuk *power-law nonlinearity* dapat dilihat pada Persamaan 2.6.

$$y = x^{a_0} \quad (2.6)$$

Dengan a_0 adalah eksponen pangkat yang biasanya bernilai sekitar 0.1. Fungsi nonlinier ini dirancang untuk memberikan respons yang lebih kuat terhadap perubahan intensitas sinyal dan menekan variasi kecil yang tidak signifikan.

H. Discrete Cosine Transform (DCT)

Setelah penerapan fungsi nonlinier, fitur yang dihasilkan akan ditransformasi menggunakan *Discrete Cosine Transform* (DCT). DCT mengubah data yang telah diproses menjadi koefisien *cepstral* yang lebih mudah untuk dianalisis dan digunakan dalam pengenalan suara. Tahapan ini berfungsi untuk mengurangi dimensi fitur yang dihasilkan dan menyaring informasi yang tidak penting.

I. Mean Normalization

Setelah transformasi DCT, langkah selanjutnya adalah *mean normalization*, di mana koefisien *cepstral* dinormalisasi berdasarkan nilai rata-rata dari koefisien tersebut. Tujuan dari normalisasi ini adalah untuk menyeimbangkan koefisien dan mengurangi variasi yang disebabkan oleh perbedaan tingkat energi dalam sinyal.

2.7 Klasifikasi

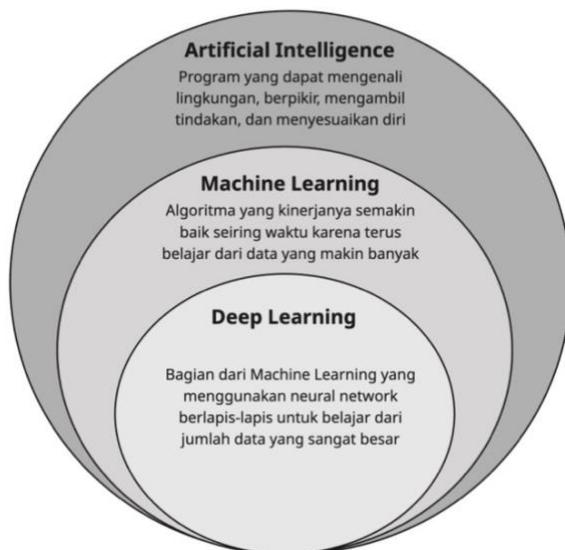
Klasifikasi adalah proses pemberian label pada data berdasarkan pola yang telah dipelajari dari contoh-contoh sebelumnya. Dalam pemrosesan suara, klasifikasi digunakan untuk mengelompokkan suara ke dalam kategori tertentu berdasarkan fitur-fiturnya. Proses ini sangat penting dalam pengenalan ucapan, deteksi emosi, atau identifikasi pembicara. Kemajuan *deep learning* telah membawa perubahan besar di bidang ini karena model-modelnya mampu

langsung mengekstraksi detail penting dari data audio mentah hingga menghasilkan klasifikasi yang lebih akurat dan efisien.

Salah satu metode yang sering digunakan untuk klasifikasi suara adalah *Convolutional Neural Network* (CNN). CNN dikenal unggul dalam mengenali pola dalam data sehingga cocok untuk mengenali fonik atau mendeteksi emosi dalam rekaman suara. Penelitian oleh Tumanyan (2022) menunjukkan bahwa CNN efektif untuk mengenali emosi dalam suara. Dengan memanfaatkan fitur MFCC, CNN mampu menangkap pola-pola penting dalam nada dan intensitas suara yang mencerminkan emosi seperti marah, bahagia, dan terkejut. Model ini mencapai akurasi 67,5%, mengungguli model LSTM pada data yang sama (Tumanyan, 2022). Selain CNN, *Recurrent Neural Networks* (RNN) juga berperan penting dalam klasifikasi suara, terutama untuk data sekuensial seperti *speech*. RNN dirancang khusus untuk menganalisis data yang berurutan, sehingga cocok untuk sinyal audio yang membutuhkan pemahaman konteks berdasarkan urutan suara.

2.8 Deep Learning

Deep learning adalah cabang dari *machine learning* yang menganalisis data menggunakan *deep neural networks* dan memproses data untuk mengenali dan memahami pola yang ditemukan. Teknologi ini terkenal sebagai alat revolusioner di berbagai bidang. Kata “*deep*” pada *deep learning* merujuk pada struktur *neural network* yang berlapis-lapis, di mana data diproses melalui beberapa tahap sehingga memungkinkan pengembangan model yang kompleks. *Deep learning* bekerja dengan memanfaatkan beberapa *layer* untuk memahami data secara bertahap. Proses pelatihan *deep learning* memang memakan waktu lebih lama karena kompleksitas model dan jumlah parameter yang besar, tetapi model lebih efisien dibanding *machine learning* umumnya saat tahap pengujian. Secara keseluruhan, *deep learning* belajar langsung dari data dengan mengandalkan *neural network* berlapis yang mampu menangkap pola secara mendalam (Sarker, 2021). Gambar 2.4 menunjukkan posisi *deep learning* dibandingkan dengan *machine learning* dan *artificial intelligence* (Alzubaidi et al., 2021).



Gambar 2.4 Cakupan Deep Learning

Deep learning membawa perubahan besar dalam penerapan pemrosesan suara, terutama di bidang seperti pemisahan suara, autentikasi pembicara, dan identifikasi patologi suara. Berdasarkan penelitian oleh Ali (2025), *Convolutional Neural Network* (CNN) menunjukkan performa akurasi tertinggi dalam klasifikasi *genre* musik dengan ekstraksi fitur MFCC, yaitu sekitar 75% setelah 50 epoch. Sebaliknya, meskipun *Transformer* membutuhkan waktu *training* yang lebih panjang, model ini menunjukkan peningkatan akurasi yang konsisten tanpa *plateau*. Hal ini menunjukkan bahwa *Transformer* konsisten meningkat selama pelatihan, cocok untuk data sekuensial. Sementara RNN-LSTM cenderung stagnan dan performanya di bawah CNN dan *Transformer* (Ali, 2025).

Untuk mendeteksi patologi suara melalui analisis pola vokal, penggunaan arsitektur *hybrid* yang menggabungkan CNN dan *Bidirectional Long Short-Term Memory* (BiLSTM) menunjukkan hasil yang menjanjikan. Metode ini dapat identifikasi kondisi seperti laringitis, nodul suara, bahkan mengindikasi tahap awal penyakit *Parkinson* (Amami et al., 2023; Lee, 2021). Pengembangan ini menunjukkan peran penting *deep learning* dalam meningkatkan akurasi dan efisiensi sistem pemrosesan suara.

2.8.1 Convolutional Neural Networks (CNN)

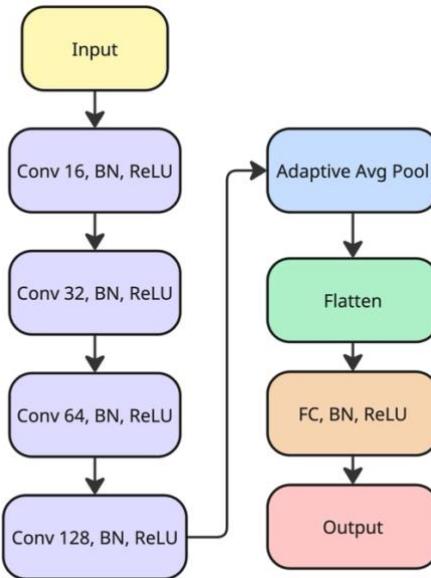
Convolutional Neural Networks (CNN) merupakan salah satu jenis model *deep learning* yang cocok untuk penerapan identifikasi fonik karena kemampuannya dalam mengenali pola lokal pada sinyal waktu. Dalam konteks fitur audio seperti PNCC atau MFCC, arsitektur CNN 1D memungkinkan model untuk secara otomatis mempelajari pola fonik secara temporal dari urutan fitur yang diekstraksi dari sinyal ucapan.

Arsitektur CNN biasanya terdiri dari beberapa komponen utama seperti *convolutional*, *pooling* dan *fully connected layers*. Lapisan *convolutional* menggunakan filter untuk memproses data yang di-input sehingga model dapat mendeteksi pola lokal seperti tepian dan tekstur. Selanjutnya, lapisan *pooling* melakukan *downsampling* pada peta fitur untuk menjaga informasi penting dan mengurangi kompleksitas data. Lapisan-lapisan ini memungkinkan CNN untuk menangkap fitur detail yang diperlukan dalam pengenalan fonik, terutama karena representasinya dapat sangat bervariasi (Mehrish et al., 2023).

Berdasarkan contoh arsitektur pada Gambar 2.5, lapisan input dari arsitektur CNN menerima fitur hasil ekstraksi dari *file* audio. Model CNN ini terdiri dari empat lapisan *convolutional* dengan jumlah filter sebesar 16, 32, 64, dan 128. Tiap *filter* berfungsi mengekstraksi pola yang penting dari fitur audio seperti intensitas dan transisi frekuensi. Setelah setiap lapisan *convolutional*, dilakukan *batch normalization* untuk mempercepat pelatihan dan membantu model belajar lebih stabil. Untuk memastikan bahwa *network* dapat mempelajari pola kompleks, setiap lapisan dilengkapi dengan fungsi aktivasi *Rectified Linear Unit* (ReLU) untuk membuat *neural network* dapat mempelajari dan mengenali pola yang lebih rumit dalam data. Selanjutnya, digunakan *layer adaptive average pooling* untuk mengecilkan dimensi fitur tanpa menghilangkan informasi pentingnya, sehingga hasil ekstraksi tetap relevan untuk proses klasifikasi selanjutnya (Ibrahim et al., 2022).

Tahap berikutnya adalah *data flattening*, yaitu proses mengubah *output multidimensional* dari lapisan *convolutional* menjadi *array* satu dimensi agar sesuai dengan lapisan

fully connected. Lapisan ini dilengkapi dengan *batch normalization* dan ReLU untuk memaksimalkan kemampuan model dalam membedakan pola sebelum menghasilkan prediksi akhir.



Gambar 2.5 Contoh Struktur CNN

Dalam pengenalan fonik, CNN 1D digunakan untuk mengekstraksi pola-pola audio yang berkaitan dengan pelafalan huruf tertentu. Model ini menerima input berupa sekuens fitur seperti PNCC, lalu mengekstraksi karakteristik suara huruf berdasarkan pola frekuensi dan waktunya. CNN 1D efektif dalam mengenali struktur lokal pada urutan fitur audio, sehingga mampu membedakan suara huruf yang mirip sekalipun. Penelitian menunjukkan bahwa CNN dapat mengungguli metode tradisional dalam *speech recognition* dengan kesalahan yang lebih rendah. Contohnya, studi oleh Abdel-Hamid *et al.* menunjukkan CNN dapat beradaptasi dengan variasi ucapan dan memberikan hasil yang lebih baik dibanding DNN standar (*Abdel-Hamid et al., 2014*).

2.8.2 Recurrent Neural Networks (RNN)

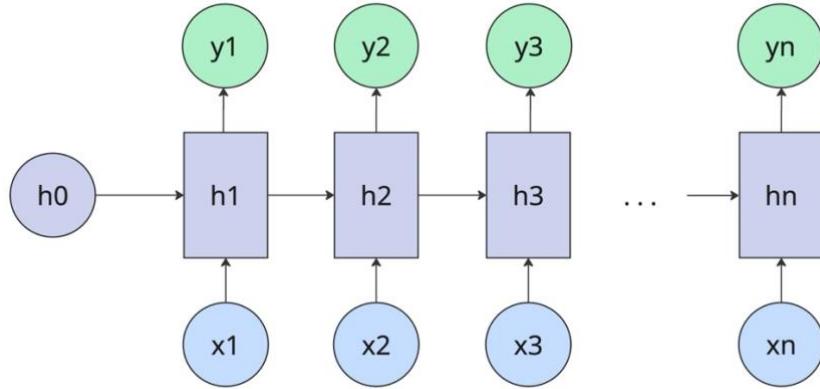
Recurrent Neural Networks (RNN) dirancang untuk menangani data berurutan dengan menyimpan informasi input sebelumnya dalam *hidden state*. Arsitektur RNN terdiri dari tiga lapisan utama: *input*, *hidden*, dan *output*. Berdasarkan Gambar 2.6, *recurrent connections* pada RNN berbeda dengan *neural network feedforward* sehingga memungkinkan informasi untuk berputar di dalam jaringan. Pada setiap waktu t , RNN memproses vektor input x_t dan memperbarui *hidden state* h_t menggunakan persamaan berikut:

$$h_t = \sigma_h (W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (2.7)$$

Di mana b_h , adalah vektor bias, σ_h adalah fungsi aktivasi (biasanya menggunakan ReLU) atau fungsi *hyperbolic tangent* (tanh)), W_{xh} adalah matriks bobot antara lapisan input dan lapisan tersembunyi (*hidden layer*), dan W_{hh} , adalah matriks bobot untuk *recurrent connection*. Persamaan berikut menunjukkan *output* pada setiap *step* waktu, t :

$$y_t = \sigma_y (W_{hy} h_t + b_y) \quad (2.8)$$

Di mana W_{hy} adalah matriks bobot antara *output* dan lapisan tersembunyi, b_y adalah vektor bias, dan σ_y adalah fungsi aktivasi pada lapisan *output* (Mienye et al., 2024).



Gambar 2.6 Arsitektur *Vanilla RNN*

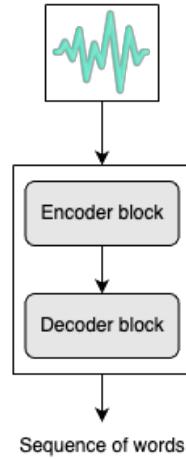
RNN adalah jenis *neural network* yang cocok untuk data berurutan karena efektif untuk *time series*, *natural language processing*, bahkan pengenalan fonik. RNN dapat mengingat informasi dari input sebelumnya karena mempunyai koneksi berulang. Fitur “memori” ini memungkinkan RNN untuk mengenali ketergantungan antar waktu (Mienye et al., 2024). Salah satu perkembangan besar dalam RNN adalah jaringan *Long Short-Term Memory* (LSTM) yang dirancang untuk mengatasi kesulitan dalam memahami hubungan jangka panjang pada data berurutan. Studi menunjukkan LSTM mengungguli RNN konvensional dalam *sequential tasks* seperti pengenalan suara karena kemampuannya memahami hubungan jangka panjang (Sun et al., 2020).

Selain LSTM, salah satu variasi RNN yang banyak digunakan adalah *Gated Recurrent Unit* (GRU) yang dirancang sebagai alternatif dengan struktur arsitektur lebih sederhana. Tidak seperti LSTM yang memiliki tiga *gate*, GRU hanya memiliki dua, yaitu *update* dan *reset gate*, serta menggabungkan *cell state* dan *hidden state* menjadi satu komponen. GRU menjadi lebih ringan dan cepat dilatih karena strukturnya yang lebih ringkas, tetapi tetap mampu menangkap hubungan temporal dalam data sekuensial. Berdasarkan studi oleh Shiri et al. (2024), GRU menawarkan efisiensi pelatihan yang lebih baik dibanding LSTM tanpa kehilangan performa secara signifikan dalam berbagai percobaan klasifikasi berbasis sekuens. Hal ini membuat GRU cocok digunakan pada skenario dengan keterbatasan sumber daya komputasi atau saat dibutuhkan *training* yang cepat.

Pada pengenalan fonik, RNN digunakan untuk menganalisis sinyal audio dan mengenali pola fonik. Sifat berurutan dari ucapan membuat RNN cocok untuk memproses audio *frames* karena memungkinkan model untuk mempelajari pola temporal yang melekat pada fonik. Kemampuan RNN untuk menangani urutan dengan panjang yang bervariasi menjadi sangat berguna karena durasi dan struktur fonik yang berbeda-beda.

2.8.3 Transformer

Transformer pertama kali diperkenalkan oleh Vaswani *et al.* pada 2017. Model ini merevolusi berbagai bidang *deep learning*, seperti NLP dan pengenalan suara. Keunikan arsitektur *Transformer* terletak pada mekanisme *self-attention* yang memungkinkan model untuk memperhatikan tingkat pentingnya setiap elemen dalam sebuah urutan, terlepas dari posisi elemen tersebut (Vaswani *et al.*, 2017).



Gambar 2.7 Arsitektur *Transformer*

Berdasarkan Gambar 2.7, *Transformer* terdiri dari blok-blok besar *encoder* dan *decoder*. Pada tahap *encoder*, model menerima vektor fitur dari sinyal audio yang direpresentasikan sebagai $X = (x_1, \dots, X_T)$. Selanjutnya, pada tahap *decoder* model mereplikasi urutan *output* $W = w_m = (w_1, \dots, w_M)$ berdasarkan representasi yang diterima dari *encoder*. Model ini bersifat *autoregressive* sehingga setiap langkah pada proses *decoding* memanfaatkan elemen sebelumnya untuk menghasilkan elemen berikutnya. Blok *encoder* dan *decoder* pada *Transformer* menggunakan beberapa lapisan *self-attention* yang saling terhubung. Setiap blok bekerja terpisah, tetapi saling melengkapi untuk membangun representasi data yang lebih baik (Orken *et al.*, 2022).

Arsitektur *end-to-end encoder/decoder* untuk pengenalan suara terdiri dari satu *encoder*, satu *decoder*, dan mekanisme *attention*. Mekanisme ini bekerja dengan menyoroti bagian penting dari data suara untuk memprediksi hasil akhir. *Encoder* mengubah data akustik, seperti hasil ekstraksi fitur, menjadi bentuk representasi alternatif. Lalu, digunakan *decoder* untuk menghasilkan label suara. Perbedaan *Transformer* dengan E2E adalah *Transformer* mempunyai banyak *encoder* dan *decoder*, di mana masing-masing mempunyai mekanisme *self-attention* yang bekerja internal. Umumnya, jumlah *encoder* dan *decoder* sama pada *Transformer*. Setiap *encoder* terdiri dari dua lapisan utama: *multi-head attention layer* dan *feed-forward neural network*. *Multi-head attention layer* memungkinkan *encoder* memahami koneksi kata-kata lain ketika memproses sebuah kata dalam sebuah kalimat. Hasilnya kemudian diteruskan ke *feed-forward neural network* yang memproses setiap kata secara terpisah (Orken *et al.*, 2022).

Transformer juga telah digunakan sebagai model *hybrid* dengan hasil yang menjanjikan. Li membandingkan model *Transformer* dan RNN dan hasilnya menunjukkan bahwa keunggulan dari masing-masing model dapat digabungkan untuk menangkap hubungan jarak

jauh serta pola berurutan dalam data suara dan bahasa alami. Metode ini memanfaatkan kemampuan kedua arsitektur secara maksimal, di mana RNN lebih ideal untuk menangani pola temporal dalam data berurutan, sementara *Transformer* lebih unggul dalam memproses data secara paralel dan memahami hubungan jarak jauh dengan lebih efektif (X. Li, 2024). Selain itu, studi oleh Pham *et al.* (2019) menyatakan *Transformer* bekerja baik untuk menangani *Automatic Speech Recognition* (ASR). Dengan metode ini, tidak diperlukan lagi langkah-langkah seperti ekstraksi fitur dan pemodelan terpisah sehingga membuat proses lebih sederhana (Pham *et al.*, 2019).

2.9 Evaluasi Matrix

Nilai akurasi dan F1 Score dari data pelatihan akan dievaluasi. Nilai akurasi menunjukkan seberapa akurat model memprediksi semua data, baik positif maupun negatif. Nilai F1 Score menunjukkan seberapa baik model bekerja berdasarkan kombinasi precision dan recall. Untuk menghitung akurasi, *precision*, *recall*, dan F1 Score, dapat digunakan metode analisis seperti *Confusion Matrix*.

Kinerja model *deep learning* dievaluasi menggunakan *confusion matrix*, yaitu sebuah matriks yang dapat menunjukkan hasil klasifikasi sesungguhnya dengan hasil klasifikasi yang diprediksi oleh model. Gambaran dari *confusion matrix* ditunjukkan pada Gambar 2.8. Berdasarkan nilai aktual dan prediksi, *confusion matrix* dibagi menjadi empat kategori: *True Negative* (TN), *True Positive* (TP), *False Negative* (FN), and *False Positive* (FP) dengan penjelasan sebagai berikut:

		a	b	...	z
Actual Labels	a				
	b				
...					
z					
Predicted Labels		a	b	...	z

Gambar 2.8 *Confusion Matrix*

- FP (*False Positive*): Jumlah sampel negatif yang salah diklasifikasi sebagai positif.
- FN (*False Negative*): Jumlah sampel positif yang salah diklasifikasi sebagai negatif.
- TP (*True Positive*): Jumlah sampel positif yang diklasifikasi dengan benar.
- TN (*True Negative*): Jumlah sampel negatif yang diklasifikasi dengan benar (Sary *et al.*, 2023).

Matriks TP, TN, FP, dan FN digunakan untuk menghitung metrik evaluasi model seperti akurasi, presisi, *recall*, dan F1 score.

- a. Akurasi adalah rasio prediksi yang benar (baik positif maupun negatif) berbasis data keseluruhan. Akurasi mengukur seberapa baik model klasifikasi dalam memprediksi seluruh data. Nilai akurasi dapat dihitung dengan Persamaan 2.9.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.9)$$

- b. Presisi adalah rasio antara jumlah prediksi positif yang benar dengan total hasil prediksi positif. Presisi menunjukkan tingkat akurasi data yang benar-benar positif terhadap hasil prediksi model. Nilai presisi dapat dihitung dengan Persamaan 2.10.

$$Presisi = \frac{TP}{TP + FP} \quad (2.10)$$

- c. *Recall* adalah rasio antara jumlah prediksi positif yang benar dengan total data positif yang sebenarnya. Recall menunjukkan seberapa baik model mampu menemukan data positif yang relevan. Nilai recall dapat dihitung menggunakan Persamaan 2.11.

$$Recall = \frac{TP}{TP + FN} \quad (2.11)$$

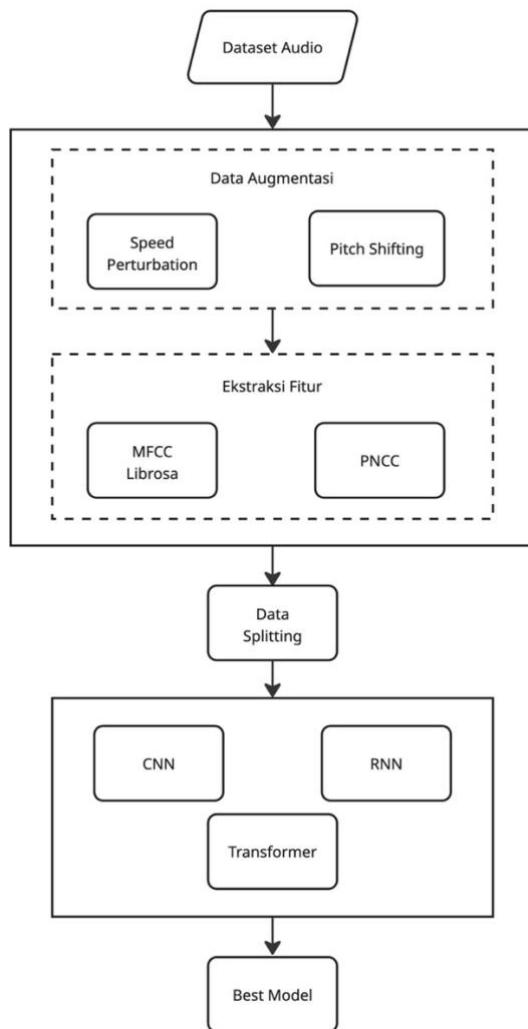
- d. F1 Score atau nilai pengukuran adalah hasil penggabungan nilai presisi dan *recall*. Nilai F1 Score dapat dihitung menggunakan Persamaan 2.12 (Clara et al., 2021).

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.12)$$

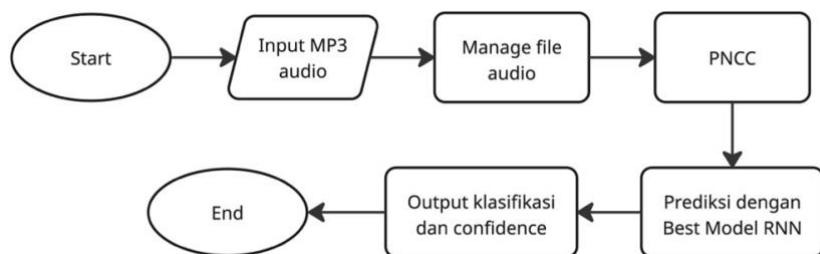
BAB 3 METODOLOGI

3.1 Metodologi Pengembangan

Tugas Akhir ini akan melalui beberapa tahapan, mulai dari *pre-processing*, pembuatan model, hingga evaluasi model. Pada Gambar 3.1 dijelaskan setiap komponen dan keterkaitannya dalam alur yang lebih jelas. Sedangkan Gambar 3.2 merupakan alur *website testing* menggunakan hasil model terbaik.



Gambar 3.1 Diagram Alur Pembuatan Model



Gambar 3.2 Diagram Alur Website

3.1.1 Dataset

Dataset yang digunakan dalam penelitian ini mengacu pada penelitian Razak (2024) yang sebelumnya berjumlah 10 *native speaker* dan 6 *non-native speaker*, sedangkan pada Tugas Akhir ini bertambah menjadi 19 *native speaker* dan 19 *non-native speaker*. Dari 19 *non-native speaker* ini, terdiri atas 6 penutur asal Jawa, 3 penutur asal Batam, 4 penutur asal Papua, 3 penutur asli Bali, dan 3 penutur asli Madura. Total audio keseluruhan yang digunakan adalah 986. *Dataset* ini dikumpulkan dengan cara merekam *voice note* menggunakan ponsel dalam format MP3, lalu disimpan dalam folder Google Drive. Rincian lengkap mengenai *dataset* yang digunakan dalam Tugas Akhir ini dijabarkan pada Tabel 3.1.

Tabel 3.1 Dataset

Sumber	No	Total
Native speaker 1	1	26 audio
Native speaker 2	2	26 audio
Native speaker 3	3	26 audio
Native speaker 4	4	25 audio
Native speaker 5	5	26 audio
Native speaker 6	6	26 audio
Native speaker 7	7	26 audio
Native speaker 8	8	26 audio
Native speaker 9	9	25 audio
Native speaker 10	10	26 audio
Native speaker 11	11	26 audio
Native speaker 12	12	26 audio
Native speaker 13	13	26 audio
Native speaker 14	14	26 audio
Native speaker 15	15	26 audio
Native speaker 16	16	26 audio
Native speaker 17	17	26 audio
Native speaker 18	18	26 audio
Native speaker 19	19	26 audio
Non-native speaker j_1 (Jawa)	20	26 audio
Non-native speaker j_2 (Jawa)	21	26 audio
Non-native speaker j_3 (Jawa)	22	26 audio
Non-native speaker j_4 (Jawa)	23	26 audio
Non-native speaker j_5 (Jawa)	24	26 audio
Non-native speaker j_6 (Jawa)	25	26 audio

<i>Non-native speaker bt_1</i> (Batam)	26	26 audio
<i>Non-native speaker bt_2</i> (Batam)	27	26 audio
<i>Non-native speaker bt_3</i> (Batam)	28	26 audio
<i>Non-native speaker b_1</i> (Bali)	29	26 audio
<i>Non-native speaker b_2</i> (Bali)	30	26 audio
<i>Non-native speaker b_3</i> (Bali)	31	26 audio
<i>Non-native speaker m_1</i> (Madura)	32	26 audio
<i>Non-native speaker m_2</i> (Madura)	33	26 audio
<i>Non-native speaker m_3</i> (Madura)	34	26 audio
<i>Non-native speaker p_1</i> (Papua)	35	26 audio
<i>Non-native speaker p_2</i> (Papua)	36	26 audio
<i>Non-native speaker p_3</i> (Papua)	37	26 audio
<i>Non-native speaker p_4</i> (Papua)	38	26 audio

Baris 1–19 mewakili *native speaker*, sementara baris 20–23 dan 26–27 merujuk pada *non-native speaker* dari dataset Razak. Baris 24, 25 dan 28–38 adalah *non-native speaker* tambahan yang ditambahkan oleh penulis.

3.1.2 *Pre-processing Data*

Pre-processing dalam penelitian ini melibatkan beberapa tahapan, di antaranya augmentasi data dan ekstraksi fitur.

A. Data Augmentasi

Tahap pertama dalam *pre-processing* adalah *data augmentation*. Pada tahap ini, *dataset* diuji dengan dua teknik augmentasi untuk menghasilkan versi baru dari data yang ada. Dua metode yang digunakan antara lain:

- *Constant Pitch Shifting*: Perubahan kecil *pitch* dapat menghasilkan suara yang lebih alami dan tidak terdengar artifisial saat mensintesis suara.
- *Constant Speed Perturbation*: Mengubah kecepatan pemutaran sinyal audio secara konstan.

Penelitian sebelumnya oleh Razak (2024) menggunakan *noise addition* dan *time shifting* untuk meningkatkan keragaman data audio. Berikut adalah perbandingan antara metode augmentasi data yang digunakan dalam penelitian sebelumnya dan penelitian ini:

- *Noise Addition*: Metode ini menambahkan *noise* dalam sampel audio. Sementara dalam Tugas Akhir ini, metode seperti *pitch shifting* dan *speed perturbation* berfokus pada perubahan isi fonik itu sendiri, sedangkan *noise addition* lebih berfokus pada memperkuat ketahanan sinyal dengan menambahkan *noise*.
- *Time Shifting*: Teknik ini mengubah waktu pada sinyal audio. Meskipun mirip dengan *speed perturbation* karena sama-sama memengaruhi domain temporal, keduanya meningkatkan variabilitas dengan cara yang berbeda. *Time shifting* dilakukan dengan sedikit menggeser waktu mulai atau berakhirnya sinyal, sehingga memengaruhi ritme

dan ketepatan waktu pengucapan fonik. Sementara itu, *speed perturbation* lebih berfokus pada mengubah kecepatan audio fonik secara keseluruhan.

Tujuan utama dari augmentasi data adalah menambahkan variasi tambahan dari *dataset*. Hal ini sangat penting karena ukuran dataset yang tidak terlalu besar. Dengan menambahkan data yang lebih beragam, model dapat menangkap lebih banyak pola dan karakteristik sehingga dapat mengurangi potensi terjadinya *overfitting*.

B. Ekstraksi PNCC

PNCC adalah salah satu metode yang dirancang lebih tahan *noise* dibandingkan MFCC. Pendekatan ini terdiri dari beberapa langkah yang bertujuan untuk menyaring, menstabilkan, dan menormalkan spektrum audio agar lebih representatif terhadap sinyal utama. Langkah pertama adalah *pre-emphasize* yang berfungsi untuk menonjolkan komponen frekuensi tinggi. Hal ini bertujuan untuk menyeimbangkan spektrum dan mengurangi *spektral leakage*. Lalu, sinyal diubah ke domain frekuensi menggunakan *Short Time Fourier Transform* (STFT) dan magnitude-nya dikuadratkan untuk menghasilkan *power spectrogram*. Hasilnya diproyeksikan ke *domain mel* menggunakan *mel filterbank* agar representasi energi selaras dengan persepsi manusia terhadap frekuensi.

Tahap selanjutnya adalah *medium time power* menggunakan *moving average* yang berfungsi untuk meratakan fluktuasi energi antar frame dengan menghitung rata-rata energi dari frame-frame sekitarnya. Proses ini bertujuan agar perubahan energi tidak terlalu drastis antar waktu. Untuk memisahkan komponen sinyal dari *noise*, digunakan *asymmetric lowpass filter* untuk memperkirakan *low envelope* sinyal. *Envelope* dikurangi dari *medium time power* agar komponen *noise background* dapat tereliminasi, lalu hasilnya dikenakan *rectification* untuk menghapus nilai negatif. Selanjutnya sinyal diproses menggunakan *temporal masking* yang menekan energi frame apabila tidak cukup kuat dibanding puncak sebelumnya. Lalu, dilakukan seleksi antara hasil *masking* atau *flooding* berdasarkan *threshold* energi. Proses selanjutnya adalah *weight smoothing*, yaitu perataan horizontal di domain frekuensi untuk mengurangi fluktuasi spektral yang tajam. Kemudian dilakukan *mean power normalization*, yaitu normalisasi berdasarkan rata-rata daya *running* untuk tiap frame agar perbedaan energi antar frame tidak mendominasi. Selanjutnya dilakukan *non-linear compression* berbasis *power law* (pangkat 1/15) sebagai pengganti *log compression* yang lebih tahan terhadap *noise impulsif*. Tahap terakhir, matriks hasil distandarisasi melalui *mean-std normalization* sehingga fitur siap digunakan untuk klasifikasi. *Output* akhirnya berupa matriks dua dimensi PNCC.

C. Ekstraksi MFCC

Pre-emphasis merupakan tahap awal yang bertujuan untuk memperkuat komponen frekuensi tinggi dalam sinyal audio, biasanya dengan menerapkan *filter high-pass*. Setelah itu, sinyal dibagi menjadi potongan-potongan pendek (frame) yang saling tumpang tindih. Ukuran frame dan jaraknya (*hop size*) menentukan seberapa rapat sinyal dibagi. Setiap frame kemudian diberi *windowing*, umumnya dengan *Hamming window*, untuk mengurangi distorsi di tepi dan mencegah kebocoran spektrum saat dilakukan *Fast Fourier Transform* (FFT). Proses ini mengubah sinyal dari bentuk waktu menjadi bentuk frekuensi. Selanjutnya, hasil transformasi frekuensi ini diproses menggunakan *Mel filterbank* untuk memetakan frekuensi ke skala Mel

yang mendekati cara manusia mendengar suara. Setelah itu, dilakukan transformasi DCT (*Discrete Cosine Transform*) untuk menyederhanakan data dan menghasilkan koefisien MFCC. Beberapa koefisien utama dipilih untuk merepresentasikan ciri penting dari suara. Proses ini bisa disesuaikan dengan kebutuhan, misalnya dengan mengatur jumlah koefisien, ukuran frame, atau *hop size*, agar hasil ekstraksi fitur benar-benar optimal.

3.1.3 Pembuatan Model

Tahap setelah *data pre-processing* adalah pembuatan model. Penelitian ini menggunakan model *Convolutional Neural Networks* (CNN), *Recurrent Neural Networks* (RNN), dan *Transformer*. Untuk mengoptimalkan performa model, digunakan *hyperparameter tuning* Optuna untuk mencari kombinasi terbaik dari parameter model sehingga dapat meningkatkan hasil akhirnya. Optuna melakukan banyak *trial* dalam sebuah *study* untuk mencari kombinasi *hyperparameter* terbaik. Ia menggunakan metode *sampling* untuk memilih nilai *hyperparameter* dan *pruning* untuk menghentikan *trial* yang tidak menjanjikan di awal. Setiap model akan dilatih menggunakan *dataset* yang sudah dikumpulkan untuk mengevaluasi performa masing-masing dalam pengenalan fonik.

3.2 Peralatan Pendukung

Untuk mendukung Tugas Akhir ini, sejumlah peralatan dan bahan telah disiapkan yang dijabarkan pada Tabel 3.2 untuk perangkat keras, serta Tabel 3.3 untuk perangkat lunak yang digunakan.

Tabel 3.2 Perangkat Keras

Atribut	Spesifikasi
Jenis perangkat	Laptop
Prosesor	Apple M1 (8-core CPU, 4x performance, 4x efficiency cores)
Memori	8.00 GB
Sistem operasi	MacOS Big Sur 11.0.1
Arsitektur sistem	ARM-based (Apple Silicon, 64-bit)

Tabel 3.3 Perangkat Lunak

Type	Perangkat Lunak
Bahasa pemrograman	Python 3.10.12
Integrated Development Environment (IDE)	Google Colab
Browser	Google Chrome
Library	Library pendukung di Python

3.3 Implementasi

Bahasa pemrograman Python dan Google Colaborator digunakan dalam proses pembuatan Tugas Akhir ini. Rencana implementasi untuk mengevaluasi model yang akan digunakan dalam studi dijelaskan dalam subbab ini.

3.3.1 Dataset

Dataset yang digunakan dalam Tugas Akhir ini mencakup 986 file audio yang berasal dari 19 *native speakers* dan 19 *non-native speakers*. Langkah pertama dalam pemrosesan data adalah inisialisasi daftar direktori huruf "a" hingga "z" yang mewakili setiap kelas atau fonik yang direkam, serta menentukan jalur direktori yang berisi *dataset* audio. Untuk *native*, file langsung disimpan dalam subfolder berdasarkan fonik. Sedangkan untuk *non-native*, file dikelompokkan dahulu berdasarkan wilayah penutur, kemudian dikelompokkan lagi ke dalam folder fonik. Tiap file diberi label berdasarkan abjad dan kelompok *speaker*, lalu seluruhnya dikompilasi ke dalam *DataFrame*. Proses ini ditunjukkan pada *Pseudocode* 3.1.

```
1. Function process_full_audio_dataset(base_path, folders, phonics_list)
2.   Input:
3.     base_path: root directory path
4.     folders: ['native', 'nonnative']
5.     phonics_list: ['a' to 'z']
6.   Output: DataFrame with columns: 'Phonics', 'Speaker_Group', 'Path'
7.
8.   Initialize empty lists: phonics_labels, file_paths, speaker_groups
9.
10.  For each folder in folders:
11.    Set data_path = base_path + folder
12.
13.    If folder is 'native':
14.      For each phonics in phonics_list:
15.        Set phonics_path = data_path + phonics
16.        If phonics_path exists:
17.          For each file in phonics_path:
18.            If file ends with '.mp3':
19.              Append phonics to phonics_labels
20.              Append full file path to file_paths
21.              Append 'native' to speaker_groups
22.
23.    Else: # nonnative
24.      For each region in data_path:
25.        Set region_path = data_path + region
26.        If region_path is a directory:
27.          For each phonics in phonics_list:
28.            Set phonics_path = region_path + phonics
29.            If phonics_path exists:
30.              For each file in phonics_path:
31.                If file ends with '.mp3':
32.                  Append phonics to phonics_labels
33.                  Append full file path to file_paths
34.                  Append region name to speaker_groups
35.
36.  Create DataFrame from phonics_labels, speaker_groups, and file_paths
37.
38.  Return the DataFrame
```

Pseudocode 3.1 *File Dataset Audio*

3.3.2 Data Pre-processing

Data preparation adalah langkah berikutnya setelah pengumpulan data selesai. Tahap ini mencakup augmentasi data dan ekstraksi fitur dengan MFCC dan PNCC.

A. Augmentasi Data

Augmentasi data adalah tahap awal dalam pra-pemrosesan. Metode yang diterapkan adalah *pitch shifting* dan *speed perturbation*.

Metode pertama adalah *pitch shifting*, yaitu teknik augmentasi audio yang mengubah tinggi nada (*pitch*) suara tanpa memengaruhi durasinya. Proses ini dilakukan dengan menggeser *pitch* sebesar beberapa *semitone* menggunakan parameter *n_steps*. Nilai *n_steps* positif akan menaikkan *pitch*, sedangkan nilai negatif akan menurunkannya. Hasilnya adalah versi audio baru dengan karakteristik nada yang berbeda, namun dengan durasi tetap sama. Teknik ini membantu memperkaya keragaman data dalam pengenalan fonik. Metode ini dijabarkan pada *Pseudocode 3.2*.

```
1. function pitch_shift(data, sample_rate, n_steps)
2.     Convert data to float32
3.     Apply pitch shifting using sample_rate dan n_steps
4.     Save to shifted_data
5.     return shifted_data
6. end function
```

Pseudocode 3.2 Pitch Shifting

Selanjutnya adalah *speed perturbation*, yaitu teknik augmentasi yang mengubah tempo atau kecepatan sinyal audio tanpa mengubah *pitch*-nya. Proses ini dijelaskan pada *Pseudocode 3.3*. Perubahan dilakukan dengan menerapkan faktor kecepatan (*speed factor*) yang sedikit lebih besar atau lebih kecil dari 1, misalnya antara 0.9 hingga 1.1. Jika nilainya lebih besar dari 1, audio dipercepat; jika lebih kecil dari 1, audio diperlambat. Teknik ini menghasilkan variasi tempo pelafalan yang dapat membantu model lebih adaptif terhadap kecepatan bicara yang berbeda.

```
1. function speed_perturbation(data, speed_factor)
2.     Convert data to float32
3.     Apply time stretching with speed_factor
4.     Save to stretched_data
5.     return stretched_data
6. end function
```

Pseudocode 3.3 Speed Perturbation

B. Ekstraksi MFCC

Langkah berikutnya adalah ekstraksi MFCC menggunakan *library* Librosa. Pada *Pseudocode 3.4*, dilakukan proses ekstraksi untuk masing-masing model yang digunakan, yaitu CNN, RNN, dan *Transformer*. CNN dan RNN memiliki proses yang sama, yaitu dengan menghitung MFCC menggunakan fungsi *librosa.feature.mfcc* dan melakukan *transpose* agar format hasilnya [T, D] di mana T adalah jumlah frame waktu dan D adalah koefisien fitur. Hal ini dikarenakan model memerlukan input dalam bentuk sekuens waktu. Pada *Transformer*, panjang sekuens MFCC dibatasi menjadi jumlah *frame* tertentu (semisal 100). Apabila *frame* melebihi batas, maka bagian tengah sekuens akan diambil agar konteks temporal terjaga. Apabila *frame* kurang dari batas, maka ditambahkan *padding* menggunakan *reflection*.

```
1. Function extract_mfcc_for_cnn(y, sr, n_mfcc)
2.     Compute MFCC with librosa using n_mfcc coefficients
3.     Transpose MFCC and return result
```

```

4.
5. Function extract_mfcc_for_rnn(y, sr, n_mfcc)
6.   Compute MFCC with librosa using n_mfcc coefficients
7.   Transpose MFCC and return result
8.
9. Function extract_mfcc_for_transformer(y, sr, n_mfcc, max_frames)
10.  Compute MFCC with librosa and transpose
11.  If number of frames > max_frames:
12.    Crop center portion to max_frames
13.  Else:
14.    Pad MFCC with reflection mode to match max_frames
15. Return MFCC

```

Pseudocode 3.4 MFCC Librosa

Selanjutnya, dilakukan ekstraksi fitur audio yang mencakup normalisasi, augmentasi, dan ekstraksi MFCC pada semua model pada *Pseudocode 3.5*. Langkah pertama adalah memuat berkas audio yang kemudian dilakukan normalisasi agar tidak terpengaruh oleh perbedaan amplitudo antar file. Lalu, dilakukan augmentasi data menggunakan *speed perturbation* dan *pitch shifting* untuk memperkaya *dataset*. Tiap variasi sinyal (termasuk sinyal asli sebelum augmentasi) diproses menggunakan fungsi MFCC yang sudah didefinisikan untuk tiap model sebelumnya. Hasil dari proses ekstraksi disimpan ke dalam daftar yang terpisah berdasarkan jenis model, begitu pula dengan label fonik dari setiap sampel yang dicatat juga.

```

1. Function extract_features(df, sr, dur, n_mfcc, augment)
2.
3. Input:
4.   df: DataFrame with audio paths and labels
5.   sr: sampling rate
6.   dur: fixed audio duration
7.   n_mfcc: number of MFCC coefficients
8.   augment: flag to enable augmentation
9.
10. Initialize empty lists: cnn_f, rnn_f, tsf_f, labels
11.
12. For each row in df:
13.   Load audio from file path with given sr and duration
14.   Fix signal length to match exact duration
15.   Normalize audio signal
16.
17.   Initialize list of versions with original signal
18.   If augment is True:
19.     Add speed-perturbed version
20.     Add pitch-shifted version
21.
22.   For each version in versions:
23.     Extract MFCC for CNN and append to cnn_f
24.     Extract MFCC for RNN and append to rnn_f
25.     Extract MFCC for Transformer and append to tsf_f
26.     Append label from row to labels
27.
28. Return dictionary with keys: cnn, rnn, tsf, labels
29.
30. End Function

```

Pseudocode 3.5 Ekstraksi Fitur

Karena tidak semua audio menghasilkan jumlah *frame* MFCC yang sama, perlu dilakukan penyamaan panjang agar dapat diproses lebih lanjut dalam *batch*. *Pseudocode* 3.6 berguna untuk menerima daftar sekuens MFCC dan menyesuaikan panjangnya sesuai dengan batas minimum dan maksimum yang telah ditentukan. Apabila jumlah *frame* melebihi batas, maka bagian akhir dibuang. Begitu pula dengan sebaliknya, bila *frame* kurang panjang maka nilai nol akan ditambahkan di belakang untuk mengisi kekurangan. Dengan proses *padding*, seluruh fitur MFCC memiliki bentuk yang konsisten sehingga model dapat memproses data dengan baik.

```

1. Function pad_or_truncate(sequences, max_len)
2. Initialize result list
3. For each sequence:
4.   If sequence longer than max_len:
5.     Truncate it
6.   Else:
7.     Pad it with zeros (constant mode) to match max_len
8. Return numpy array of fixed-length sequences

```

Pseudocode 3.6 Pad or Truncate

Pada tahap ini, langkah pertama adalah mengubah label fonik menjadi numerik menggunakan *LabelEncoder* yang dijabarkan pada *Pseudocode* 3.7. Untuk CNN, data di-*pad* ke panjang yang sama, kemudian dibagi menjadi data *train* dan *test* secara *stratify*. Selanjutnya, dilakukan normalisasi pada tiap *frame* menggunakan *StandardScaler*. Langkah yang sama diterapkan pada RNN dan *Transformer* dengan penyesuaian pada saat *padding*. RNN menormalisasi sekuens satu per satu setelah *fitting scaler* pada seluruh *frame* data *train*. Sedangkan *Transformer*, data dinormalisasi kemudian di-*pad* ke panjang tetap agar serata. Terakhir, seluruh data (fitur dan label) dikembalikan dalam bentuk *dictionary* yang siap di-*train* oleh model.

```

1. Function prepare_datasets(features)
2.   Input: features: dictionary with cnn, rnn, tsf features and labels
3.
4.   Encode labels using LabelEncoder -> y
5.
6.   Pad or truncate CNN features to fixed length (130)
7.   Split cnn and y into training and testing sets (stratified)
8.   Normalize CNN features with StandardScaler
9.
10.  Split RNN features and y into training and testing sets
11.  Fit StandardScaler on stacked training RNN data
12.  Normalize each RNN sequence individually
13.
14.  Split Transformer features and y into training and testing sets
15.  Fit StandardScaler on stacked training Transformer data
16.  Normalize and pad Transformer sequences to fixed length (100)
17.
18.  Return dictionary with keys:
19.    cnn: (X_train_cnn, X_test_cnn, y_train_cnn, y_test_cnn)
20.    rnn: (X_train_rnn, X_test_rnn, y_train_rnn, y_test_rnn)
21.    tsf: (X_train_tsf, X_test_tsf, y_train_tsf, y_test_tsf)
22.    label_encoder
23.
24. End Function

```

Pseudocode 3.7 Dataset Preparation

C. Ekstraksi PNCC

Ekstraksi fitur kedua yang digunakan dalam Tugas Akhir ini adalah PNCC (*Power Normalized Cepstral Coefficients*). Pada *Pseudocode* 3.8, dilakukan proses tahapan transformasi sinyal untuk menghasilkan fitur PNCC yang lebih *robust* terhadap *noise* dibandingkan dengan MFCC. Ekstraksi ini terdiri dari beberapa fungsi utama yang masing-masing mempunyai peran dalam memproses spektrum audio ke fitur PNCC.

Tahap pertama dimulai dengan *medium time power* yang berfungsi untuk meratakan fluktuasi energi antar *frame* dengan menghitung rata-rata energi dari *frame-frame* sekitarnya. Proses ini menggunakan metode *moving average* agar nilai energi menjadi lebih stabil dan tidak terlalu berubah drastis antar *frame*. Selanjutnya, fungsi *asym_lowpass* digunakan untuk memodelkan pelacakan *low envelope* dari sinyal menggunakan *asymmetric smoothing* (dengan parameter *alpha* untuk kenaikan lambat dan *beta* untuk penurunan cepat). Hal ini penting untuk memisahkan komponen utama sinyal dari *noise background*.

Lalu, pada fungsi *temporal_masking*, disimulasikan efek *masking temporal* seperti pada pendengaran manusia. Pada proses ini, nilai energi dalam *frame* akan ditekan apabila tidak cukup kuat dibandingkan *peak* sebelumnya (dengan pertimbangan *decay* menggunakan faktor *lambda* dan *mu*). Hasilnya, sinyal lebih fokus pada komponen yang dominan dalam waktu dekat. Proses selanjutnya adalah *weight smoothing*, yaitu *smoothing horizontal* pada domain frekuensi. Pada tahap ini, tiap nilai dalam sumbu frekuensi akan dirata-ratakan dengan nilai-nilai frekuensi di sekitarnya dalam suatu *window* ($\pm N$). Hal ini bertujuan untuk mengurangi perubahan tajam antarfrekvensi dan menjaga pola spektral tetap halus dan konsisten, sehingga fitur menjadi lebih tahan *noise*.

Setelah seluruh proses *masking* dan *smoothing* selesai, dilakukan normalisasi temporal menggunakan rata-rata eksponensial dari energi tiap frame (*mean_power*). Hal ini bertujuan untuk menstabilkan skala energi dari *frame* ke *frame* secara keseluruhan. Lalu, hasilnya dikompresi secara *nonlinier* menggunakan eksponen 1/15 untuk memperkecil rentang nilai yang terlalu lebar. Langkah terakhir, dilakukan normalisasi dengan *Z-score* agar distribusi nilainya seragam dan siap digunakan sebagai *input* model.

```
1. def medium_time_power(signal, M=2):
2.     Pad signal with zeros on both ends
3.     For each frame:
4.         Compute mean over (2M+1)-frame window
5.     Return smoothed signal
6.
7. def asym_lowpass(x, alpha=0.999, beta=0.5):
8.     Initialize output y
9.     For each timestep:
10.        If input >= previous output → apply slow smoothing (alpha)
11.        Else → apply fast smoothing (beta)
12.    Return filtered output
13.
14. def temporal_masking(x, lam=0.85, mu=0.2):
15.     For each timestep:
16.         Track peak value using lambda
17.         Apply conditional masking using mu factor
18.     Return masked signal
19.
```

```

20. def weight_smoothing(x, ref, N=4):
21.     For each time and frequency bin:
22.         Compute mean ratio between x and ref over a window of ±N
23.     Return smoothed feature
24.
25. def extract_pncc(data, sr):
26.     Trim silence and apply pre-emphasis
27.     Compute power spectrogram with STFT
28.     Apply mel filterbank
29.
30.     Compute medium-time power
31.     Subtract low envelope using asym_lowpass
32.     Rectify, mask, and switch based on power
33.     Apply weight smoothing with respect to reference power
34.     Normalize spectrum temporally with running average
35.
36.     Apply nonlinear compression (1/15)
37.     Normalize final matrix (mean-std normalization)
38.     Return PNCC features

```

Pseudocode 3.8 Fungsi PNCC

Setelah PNCC diekstrak dalam bentuk matriks dua dimensi, tahap selanjutnya adalah mengubah ke format yang sesuai dengan *pipeline* model. Fungsi *extract_tensor1d_pncc* berguna untuk mengonversi hasil PNCC ke dalam bentuk *tensor PyTorch*. Sebelum dikembalikan, fungsi ini memastikan bahwa tidak ada nilai ekstrem (seperti NaN atau inf) dalam hasilnya karena nilai-nilai dapat menyebabkan eror saat model dilatih. Langkah ini dapat dilihat pada *Pseudocode 3.9*.

```

1. def extract_tensor1d_pncc(data, sr):
2.     Call extract_pncc
3.     If result valid:
4.         Convert to FloatTensor
5.     Else:
6.         Return None

```

Pseudocode 3.9 Feature Wrapper PNCC

Fungsi pada *Pseudocode 3.10* menyatukan seluruh proses ekstraksi fitur PNCC dan membentuk *dataset* akhir untuk dilatih menggunakan CNN, RNN, dan *Transformer*. Untuk tiap file dalam *dataset_df*, dilakukan augmentasi menjadi tiga versi (asli, *speed perturbation*, dan *pitch shifting*). Setiap versi diproses menggunakan *extract_tensor1d_pncc* dan hasil akhirnya disimpan bersama dengan label fonik yang asli. Setelah seluruh data dikumpulkan, label dikodekan ke bentuk numerik menggunakan *LabelEncoder*. Lalu, data untuk masing-masing model CNN, RNN, dan *Transformer* dipisahkan dan dibagi menjadi data *train* dan *test* secara *stratified*. Karena panjang sekuens antar audio bisa bervariasi, tiap set di-*pad* menggunakan *pad_sequence* agar kompatibel dengan *batch processing*. *Output* akhirnya adalah tiga set *dataset* yang siap dipakai untuk *training*, masing-masing untuk CNN, RNN, dan *Transformer*. Seluruhnya sudah dalam format *tensor NumPy* dan dilengkapi dengan label yang telah *di-encode*.

```

1. def extract_all_features(dataset_df):
2.     Initialize lists for CNN, RNN, Transformer
3.

```

```

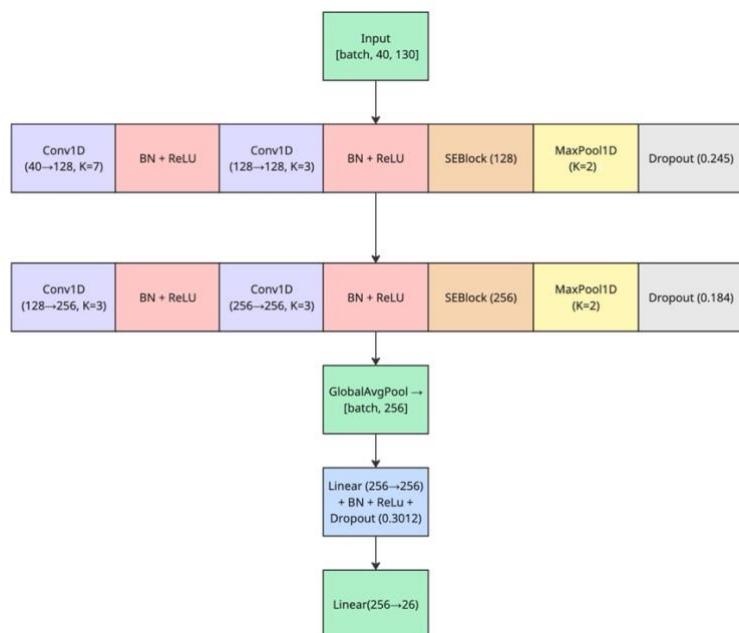
4.     For each audio path in dataset_df:
5.         Load original and augmented audio
6.         For each version:
7.             Extract sequential PNCC tensor
8.             If valid:
9.                 Append (tensor, label) to cnn_data, rnn_data, tsf_data
10.
11.    If no valid samples -> return {}
12.    Fit LabelEncoder to all labels
13.
14.    def prepare_seq_data(data):
15.        If empty -> return empty arrays
16.        Split features and labels → stratified train-test split
17.        Pad sequences using pad_sequence
18.        Return padded X_train, X_test, y_train, y_test
19.
20.    Prepare CNN, RNN, Transformer datasets
21.    Return dictionary with all datasets and label encoder
22.
23. End Function

```

Pseudocode 3.10 Dataset Preparation

3.3.3 Implementasi *Convolutional Neural Network*

Convolutional Neural Network (CNN) dilatih dan dievaluasi untuk mengklasifikasikan data audio fonik ke dalam label huruf A hingga Z dijelaskan pada *Pseudocode 3.11*. Contoh implementasi arsitektur CNN digambarkan pada Gambar 3.3. Model CNN dimulai dengan mendefinisikan dua kelas penting, yaitu *SEBlock* (*Squeeze and Excitation*) dan *CNN_Classifier*. SEB bertugas sebagai mekanisme *channel-wise attention* agar fitur-fitur penting dapat ditekankan secara adaptif. Sementara itu, *CNN_Classifier* merupakan arsitektur CNN yang dibuild secara modular yang terdiri dari beberapa *convolutional blocks* dengan konfigurasi yang dapat disesuaikan (termasuk *BatchNorm*, *activation function*, *pooling*, dan *dropout* pada tiap blok).



Gambar 3.3 Contoh Implementasi Arsitektur CNN

Setelah model dasar disusun, selanjutnya dilakukan *tuning hyperparameter* menggunakan *Optuna*. Berbagai parameter seperti jumlah blok (*n_blocks*), ukuran kernel, jumlah filter (*out_channels*), dan parameter lain seperti *dropout rate*, penggunaan *batch normalization*, dan ukuran *fully-connected layer* (*fc_units*) disampel secara acak dalam rentang tertentu. Selama tuning *Optuna*, digunakan *pruning* untuk menghindari eksperimen yang tidak menjanjikan sejak awal. Setelah ditemukan konfigurasi terbaik, model akhir dibuat ulang berdasarkan hasil parameter yang ditemukan dan dilakukan *training* ulang. Model yang selesai dilatih akhirnya dievaluasi dengan *test set* menggunakan *classification report* untuk mengukur performanya.

```

1. Function train_cnn(training_data, training_labels, test_data, test_labels,
label_encoder):
2.   Input: training_data, training_labels, test_data, test_labels, label_encoder
3.   Output: trained_model, classification_report
4.
5.   # Define Modules
6.   SEBlock with adaptive pooling, FC layers, sigmoid activation
7.   CNN_Classifier with:
8.     Configurable blocks (Conv1d, BatchNorm, ReLU, SEBlock, MaxPool,Dropout)
9.     Global average pooling
10.    Classifier:
11.      Linear -> BatchNorm -> ReLU -> Dropout -> Linear
12.
13.   # Define Optuna objective function
14.   Tune the following hyperparameters:
15.     - n_blocks: int from 2 to 4
16.     - dropout: float in range 0.2 to 0.5
17.     - fc_units: {128, 256, 512}
18.     - use_nb: True or False
19.
20.   For each convolutional block i:
21.     - channels_i: {64, 128, 256}
22.     - kernel_i: {3, 5, 7}
23.     - pool_i: 1 or 2
24.     - block_dropout_i: float in range 0.1 to 0.4
25.
26.   Instantiate CNN_Classifier using these parameters
27.   Optimizer: AdamW (learning rate = 0.001)
28.   Scheduler: CosineAnnealingLR
29.   Callbacks: AdaptiveEarlyStopping, EpochScoring (accuracy)
30.
31.   At each epoch:
32.     - Evaluate val_acc
33.     - Report val_acc to Optuna trial
34.     - Prune trial if it underperforms
35.   Return best validation accuracy from trials
36.
37.   # Main Training Flow
38.   Split training data -> train + val (stratified, test size = 0.2)
39.   Create Optuna study (maximize objective, use MedianPruner)
40.   Run Optuna trials with objective() function
41.   Extract best parameters from study.best_trial
42.   Build final CNN_Classifier using best configuration
43.   Create NeuralNetClassifier with:
44.     - AdaptiveEarlyStopping
45.     - LRScheduler (CosineAnnealingLR)
```

- ```

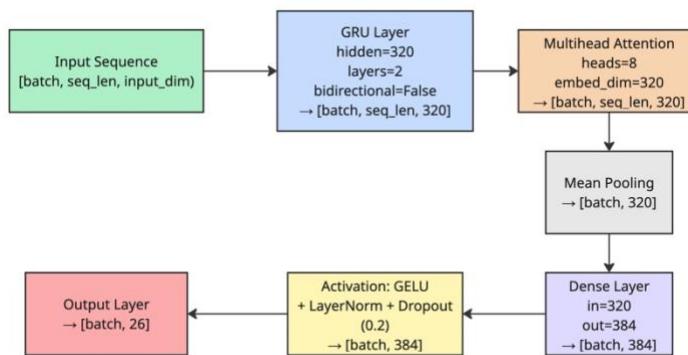
46. - EpochScoring for validation accuracy
47. - Predefined validation split
48. Fit model on training_data
49. Evaluate on test_data
50. Return trained model and classification report

```

*Pseudocode 3.11 Model Convolutional Neural Network*

### 3.3.4 Implementasi Recurrent Neural Network

*Recurrent Neural Network* (RNN) dilatih dan dievaluasi dijelaskan pada *Pseudocode 3.12*. Contoh implementasi arsitektur RNN digambarkan pada Gambar 3.4. Proses pelatihan model ini dimulai dengan membangun arsitektur utama *RNN\_GRU* yang di dalamnya terdapat *layer GRU* (*Gated Recurrent Unit*) yang dapat dikonfigurasi jumlah *layer*, ukuran *hidden state* (*gru\_hidden\_size*), dan apakah bersifat *bidirectional* atau tidak. Setelah itu, ada *layer MultiheadAttention* untuk memperkuat konteks temporal dari *sequence output*. Hasil atensinya diringkas melalui *mean pooling* dan dilanjutkan ke *classifier* yang terdiri dari lapisan *Linear*, *Activation* (ReLU/GELU/Mish), *LayerNorm*, *Dropout*, dan *Linear*.



*Gambar 3.4 Contoh Implementasi Arsitektur RNN*

Sebelum model dilatih, panjang *sequence* data di-*padding* agar panjangnya seragam dan dapat diproses dalam bentuk *batch*. Lalu, *dataset* dibagi menjadi *training*, *validation*, dan *test set* secara *stratified*. Selanjutnya, dilakukan *hyperparameter tuning* dengan *Optuna*. *Trial* dilakukan untuk menemukan kombinasi optimal dari jumlah *layer*, *hidden state GRU*, *dropout*, dan lain-lain. Apabila performa tidak meningkat, *Optuna* melakukan *pruning* untuk menghemat waktu. Pada proses *training*, digunakan *loss CrossEntropyLoss* dengan *label smoothing* untuk mengurangi *overconfidence* dan *scheduler CosineAnnealingLR* untuk mengatur *learning rate* secara dinamis. Setelah didapatkan konfigurasi terbaik, model akhir dibangun ulang menggunakan parameter tersebut dan dilatih lagi menggunakan seluruh *training set*. Lalu, model diuji di *test set* dan dievaluasi menggunakan *classification report*.

- ```

1. Function gru_optuna(training_data, training_labels, test_data, test_labels,
   label_encoder):
2.   Input: training_data, training_labels, test_data, test_labels, label_encoder
3.   Output: trained_model, classification_report
4.
5.   # Define Modules
6.   RNN_GRU:
7.     - GRU layer with configurable hidden_size, num_layers, bidirectional, DO
8.     - Multihead Attention layer (attention_heads)

```

```

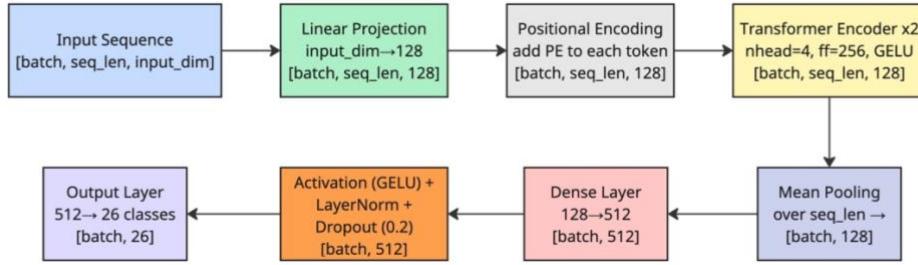
9.      - Classifier: Linear -> Activation (relu, gelu or mish) -> LayerNorm ->
    Dropout -> Linear
10.
11.     Split X_train, y_train -> train + val (test_size = 0.2 stratified)
12.     Pad sequences for train, val, test using pad_sequence
13.
14.     # Define Optuna objective function
15.     Tune the following hyperparameters:
16.         - gru_hidden_size: {256, 320, 384}
17.         - num_layers: from 1 to 3
18.         - attention_heads: {2, 4, 8}
19.         - dense_size: {384, 512, 640}
20.         - dropout: float in range 0.2 to 0.4
21.         - bidirectional: True or False
22.         - activation_fn: {relu, gelu, mish}
23.
24.     Instantiate RNN_GRU using these parameters
25.     Optimizer: AdamW (lr = 0.001)
26.     Loss: CrossEntropyLoss with label smoothing (0.1)
27.     Scheduler: CosineAnnealingLR
28.     Callbacks: AdaptiveEarlyStopping, EpochScoring (accuracy)
29.     At each epoch:
30.         - Report val_acc to Optuna trial
31.         - Prune if underperforming
32.     Return best val_acc
33.
34.     # Final training
35.     Instantiate final model using best trial parameters
36.     Create NeuralNetClassifier with:
37.         - AdamW optimizer
38.         - CosineAnnealingLR
39.         - AdaptiveEarlyStopping
40.         - EpochScoring for validation accuracy
41.         - Predefined validation split
42.     Fit model on full training set
43.     Predict on test set
44.     Return trained model and classification report

```

Pseudocode 3.12 Model Recurrent Neural Network

3.3.5 Implementasi *Transformer*

Model *Transformer* dilatih dan dievaluasi untuk mengklasifikasikan data audio fonik ke dalam label huruf A hingga Z dijelaskan pada Pseudocode 3.13. Contoh implementasi arsitektur *Transformer* digambarkan pada Gambar 3.5. Langkah pertama dalam arsitektur ini adalah mendefinisikan dua komponen utama, yaitu *PositionalEncoding* dan *TransformerClassifier*. *PositionalEncoding* berfungsi untuk menyisipkan informasi urutan posisi ke dalam *embedding input*, sedangkan *TransformerClassifier* merupakan arsitektur utama yang terdiri dari tahap *input projection*, *Transformer encoder*, dan *classifier*. Tahap pertama adalah mengubah dimensi input menjadi dimensi model (d_{model}) menggunakan *layer Linear*. Setelah itu, sinyal diberikan *positional encoding* dan diproses oleh *encoder Transformer* yang dapat dikonfigurasi banyaknya *layer*, jumlah *attention head* ($nhead$), ukuran *hidden layer* ($dim_{feedforward}$), *dropout*, dan fungsi aktivasi (ReLU/GELU). *Output* dari *encoder* diringkas menggunakan *mean pooling* dan diteruskan ke *classifier* yang terdiri dari *Linear*, *ReLU*, *LayerNorm*, *Dropout*, *Linear*.



Gambar 3.5 Contoh Implementasi Arsitektur Transformer

Sebelum model dilatih, *dataset* dibagi menjadi *training*, *validation*, dan *test set* secara *stratified*. Panjang *sequence* maksimum disimpan dan digunakan sebagai batas input. Selanjutnya, dilakukan *hyperparameter tuning* menggunakan *Optuna*. Beberapa parameter seperti *d_model*, *num_layers*, *nhead*, *dropout*, dan lain-lain disampel untuk menemukan konfigurasi terbaik berdasarkan nilai *val_acc*. *Optuna* akan melakukan *pruning* apabila performa *trial* kurang menjanjikan. Selama proses *tuning*, model dilatih menggunakan *loss function CrossEntropyLoss* dan dioptimasi dengan *AdamW*. *Scheduler CosineAnnealingLR* digunakan untuk mengatur penurunan *learning rate* secara bertahap, dan *callback AdaptiveEarlyStopping* akan mengurangi *learning rate* apabila akurasi validasi stagnan. Setelah ditemukan parameter terbaik, model akhir di-build ulang menggunakan konfigurasi tersebut. Dilakukan *training* ulang dengan *NeuralNetClassifier* dan hasil akhir dievaluasi pada *test set* menggunakan *classification report*.

```

1. Function train_transformer(X_train, y_train, X_test, y_test, label_encoder):
2.     Input: training_data, training_labels, test_data, test_labels, label_encoder
3.     Output: trained_model, classification_report
4.
5.     # Define Modules
6.     PositionalEncoding: encodes position information into input embeddings
7.     TransformerClassifier:
8.         - Input projection (Linear(input_dim -> d_model))
9.         - PositionalEncoding
10.        - TransformerEncoder:
11.            num_layers, nhead, dim_feedforward, dropout, activation
12.        - Classifier:
13.            Linear -> ReLU -> LayerNorm -> Dropout -> Linear
14.
15. Split X_train, y_train -> train + val (test_size = 0.2 stratified)
16. Save sequence length as max_seq_len
17.
18. # Define Optuna objective function
19. Tune the following hyperparameters:
20. - d_model: {64, 128, 256}
21. - num_layers: from 2 to 4
22. - nhead: {2, 4, 8} that divide d_model evenly
23. - dim_feedforward: {256, 512, 1024}
24. - dropout: between 0.2 to 0.4
25. - dense: {128, 256, 512}
26. - activation: {relu, gelu}
27.
28. Instantiate TransformerClassifier with above parameters
29. Optimizer: AdamW (lr = 0.001)

```

```

30.    Loss: CrossEntropyLoss
31.    Scheduler: CosineAnnealingLR
32.    Callbacks: AdaptiveEarlyStopping, EpochScoring
33.    At each epoch:
34.        - Report val_acc to trial
35.        - Prune trial if underperforming
36.    Return best val_acc
37.
38.    # Final Training
39.    Build final TransformerClassifier with best trial parameters
40.    Create NeuralNetClassifier with:
41.        - AdamW optimizer
42.        - CosineAnnealingLR scheduler
43.        - Predefined validation split
44.        - Callbacks: AdaptiveEarlyStopping, EpochScoring
45.    Fit model on full training set
46.    Predict on test set
47.    Return trained model and classification report

```

Pseudocode 3.13 Model Transformer

3.3.6 Evaluasi

Setelah mengumpulkan hasil dari hiperparameter, model dievaluasi menggunakan berbagai evaluasi model untuk mengetahui sejauh mana model dapat mengklasifikasikan data dengan benar. Beberapa metrik yang digunakan dalam evaluasi model meliputi F1-score, *recall*, akurasi, dan *precision*. Akurasi adalah metrik yang paling sederhana dan mudah dihitung. Metrik ini menunjukkan proporsi prediksi yang benar di antara semua prediksi yang dibuat oleh model. Meskipun akurasi memberikan gambaran umum tentang kinerja model, ia dapat terdistorsi, terutama ketika data yang digunakan tidak seimbang. Oleh karena itu, hanya mengandalkan akurasi tidak cukup. *Precision* digunakan untuk mengurangi *false positive*, yakni mengukur proporsi prediksi positif yang benar-benar positif. Sedangkan *recall* sangat penting untuk mengurangi *false negative* dan mengukur persentase data positif yang benar-benar berhasil diprediksi dengan benar oleh model. F1-score yang merupakan rata-rata dari *precision* dan *recall*, memberikan keseimbangan antara keduanya sehingga sangat berguna ketika data tidak seimbang. Metrik tambahan seperti F1-score, *precision*, dan *recall* sangat bermanfaat karena memberikan informasi lebih mendalam mengenai kelebihan dan kelemahan model dalam berbagai kondisi dan tipe data. Metrik-metrik ini menjadikan penilaian yang lebih lengkap terhadap kinerja model. Langkah ini dijelaskan pada *Pseudocode 3.14*.

Representasi visual dari kinerja model juga dapat diperoleh melalui *confusion matrix* yang menunjukkan proporsi prediksi benar dan salah untuk setiap kelas. *Confusion matrix* mengevaluasi hasil model dengan menggunakan data uji, serta membantu dalam mengidentifikasi pola kesalahan pada model.

```

1.  Function evaluate_model(model, X_test, y_test, label_encoder):
2.      Input:
3.          model: trained classifier model (CNN, RNN, or Transformer)
4.          X_test, y_test: test features and labels
5.          Label_encoder: label encoder instance
6.      Output: evaluation metrics and classification report
7.
8.      Use model to predict labels on X_test
9.

```

```

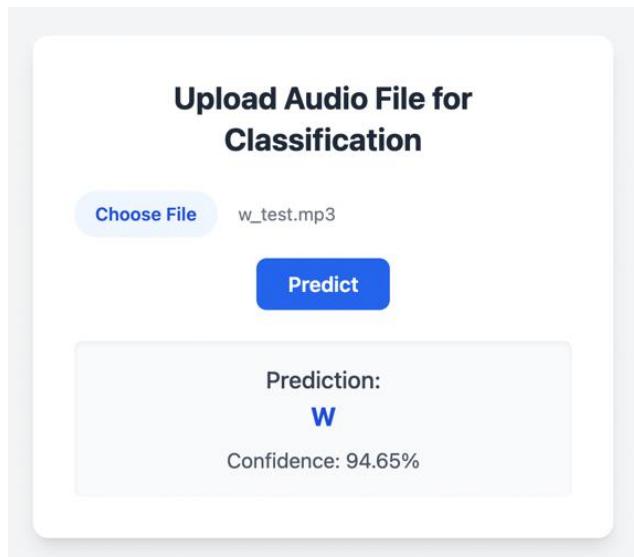
10.    # Compute metrics
11.    Accuracy = accuracy_score(y_test, y_pred)
12.    Precision precision_score(y_test, y_pred, average='macro')
13.    Recall recall_score(y_test, y_pred, average='macro')
14.    F1 Score f1_score(y_test, y_pred, average='macro')
15.
16.    Print accuracy, precision, recall, f1 score
17.    Print classification_report
18.
19.    Compute confusion_matrix(y_test, y_pred)
20.    Plot using seaborn heatmap with class labels

```

Pseudocode 3.14 Fungsi untuk Mengevaluasi Model

3.3.7 Desain Antarmuka

Untuk memprediksi bagaimana huruf fonik diucapkan, akan dikembangkan antarmuka pengguna sederhana. Antarmuka pengguna ini dirancang seperti yang terlihat pada Gambar 3.6. Terdapat tombol *choose file* untuk memilih *file* audio. Lalu, setelah audio ter-input, tekan tombol *predict* untuk memulai klasifikasi. *Output* klasifikasi akan muncul di bagian bawah tombol *predict* sesuai dengan hasil huruf yang berhasil diprediksi.



Gambar 3.6 Desain Antarmuka Pengguna

BAB 4 HASIL DAN PEMBAHASAN

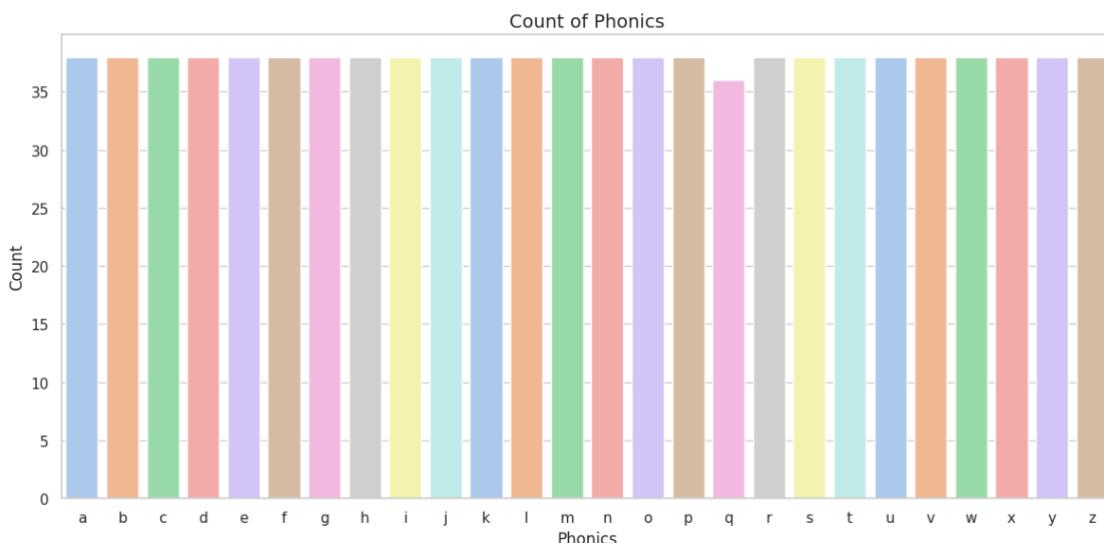
Hasil dan penjelasan dari setiap percobaan akan dijelaskan dalam bab ini. Penelitian ini melakukan lima kali percobaan untuk mendapatkan hasil yang terbaik dalam mengenali pelafalan fonik. Percobaan-percobaan yang dilakukan ditunjukkan pada Tabel 4.1.

Tabel 4.1 Skenario Eksperimen

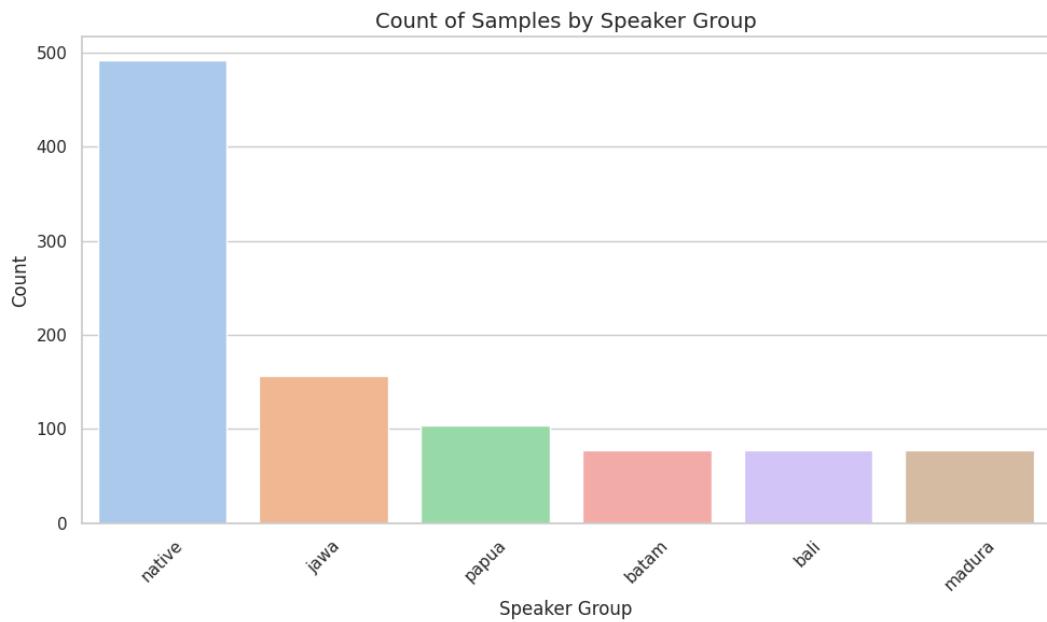
Eksperimen	Deskripsi
A	MFCC Librosa Extraction
B	PNCC Extraction
C	Native PNCC Extraction
D	Nonnative PNCC Extraction
E	Nonnative per Region PNCC Extraction

4.1 Pengumpulan Dataset

Seperi yang sudah dijelaskan pada Subbab 3.1.1. bahwa ada data yang tidak lengkap, yaitu pada *speaker Native 4* dan *Native 9* tidak ada data huruf ‘q’. Jumlah data keseluruhan dari *native* dan *non-native* adalah 986 audio. Setelah dilakukan augmentasi dengan *speed perturbation* dan *pitch shift*, jumlah dataset total adalah 2.958 dengan pembagian masing-masing 986 data asli, 986 data setelah *speed perturbation*, dan 986 data setelah *pitch shifting*. Untuk *dataset native speaker*, jumlah data asli adalah 492 dan jumlah data setelah diaugmentasi adalah 1476. Sedangkan untuk *dataset non-native speaker*, jumlah data asli adalah 494 dan jumlah data setelah diaugmentasi adalah 1482. Visualisasi dari kelas *dataset* keseluruhan dapat dilihat pada Gambar 4.1, sedangkan visualisasi persebaran dari data tiap penutur dapat dilihat pada Gambar 4.2.



Gambar 4.1 Distribusi Kelas

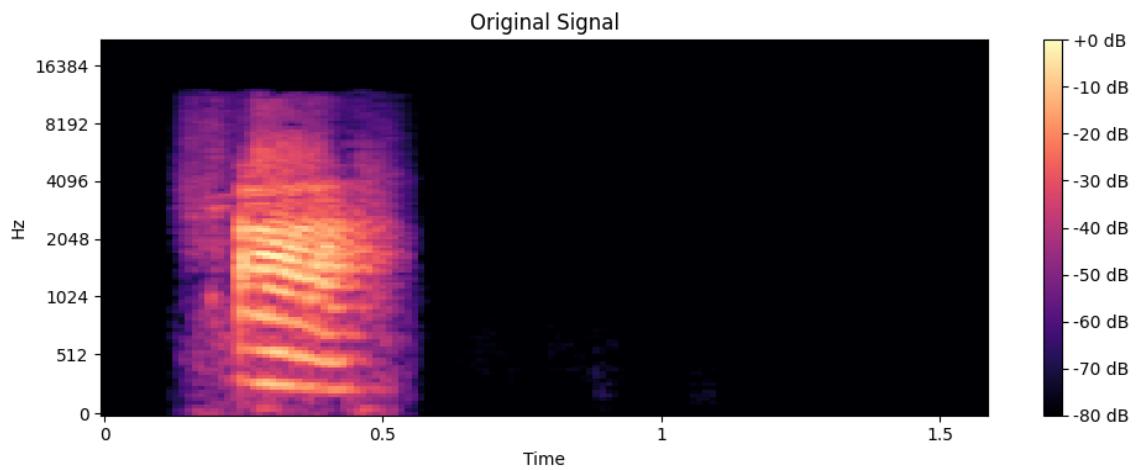


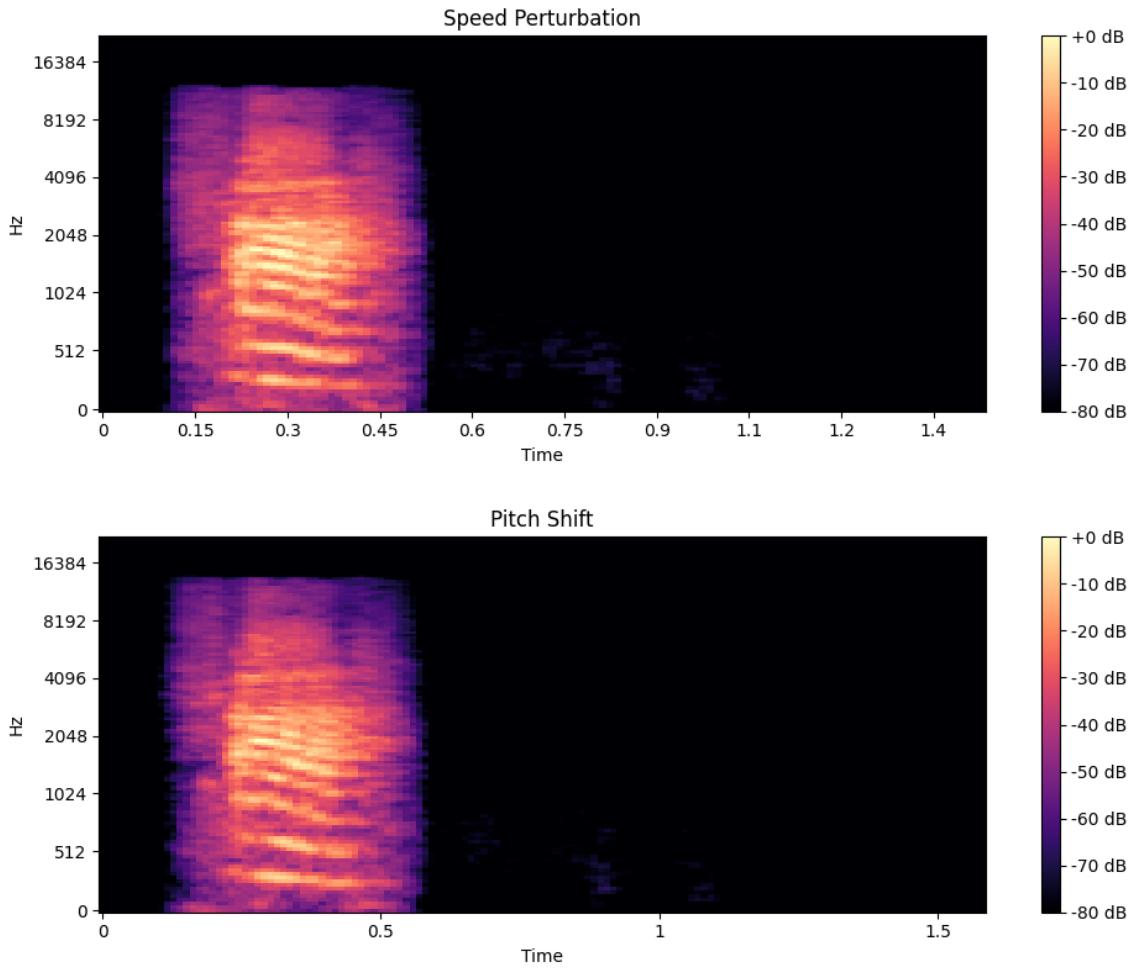
Gambar 4.2 Distribusi *Speaker*

4.2 Hasil Eksperimen A

Pada eksperimen A, metode *pre-processing* yang digunakan adalah MFCC dengan *library* Librosa dengan data augmentasi *speed perturbation* dan *pitch shifting*. Tiga model yang digunakan adalah *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Transformers*.

Augmentasi data dilakukan pada setiap *dataset* audio. Sebagai contoh, salah satu audio yang divisualisasi adalah huruf ‘a’. Gambar 4.3 menunjukkan hasil *spectrogram* sebelum dan sesudah audio diaugmentasi sehingga dapat dibandingkan perubahan yang terjadi. *Spectrogram* pertama menunjukkan audio asli, kemudian *spectrogram* kedua menunjukkan audio setelah diaugmentasi dengan *speed perturbation* yang hasilnya durasi audio menjadi lebih cepat. Terakhir, *spectrogram* ketiga adalah audio setelah dilakukan *pitch shifting*.

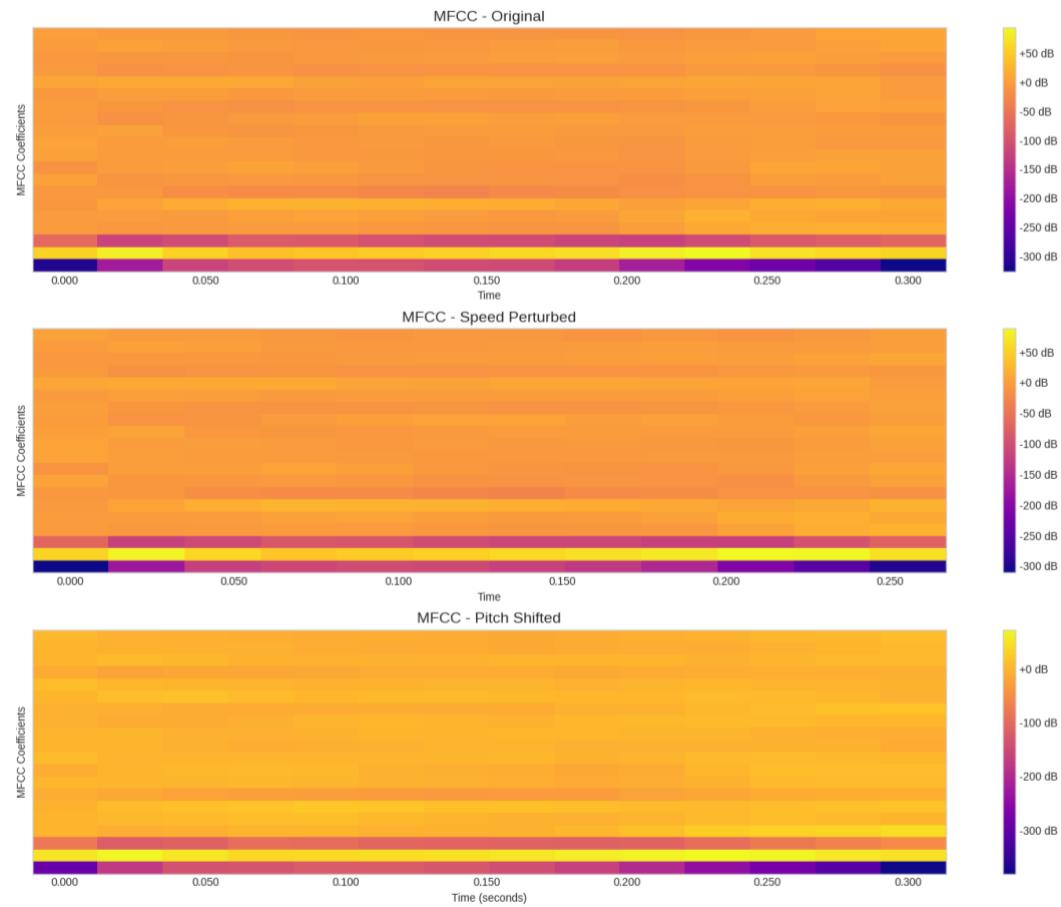




Gambar 4.3 Visualisasi Data Augmentasi Eksperimen A

Setelah dilakukan data augmentasi, langkah selanjutnya adalah ekstraksi fitur menggunakan MFCC Librosa. Dengan menggunakan *library* Librosa, ekstraksi MFCC menjadi lebih cepat dan lebih *automated*. Gambar 4.4 menunjukkan hasil ekstraksi dari MFCC Librosa. Pada data asli, hanya *amplitude*. Selanjutnya, pada data yang sudah *di-speed perturbation*, hasilnya durasi audio menjadi lebih singkat. Lalu, pada visualisasi terakhir terlihat bahwa terjadi pergeseran spektral ke frekuensi yang tinggi tanpa adanya perubahan durasi sebagai hasil data yang *di-pitch shift*.

Langkah selanjutnya adalah *modeling*. Pada tahap ini, digunakan Optuna untuk menemukan parameter-parameter terbaik untuk setiap model. Untuk eksperimen A, parameter terbaik pada CNN adalah [n_blocks: 2, dropout: 0.3310, fc_units: 128, use_bn: True, channels_0: 64, kernel_0: 5, pool_0: 2, block_dropout_0: 0.2961, channels_1: 256, kernel_1: 7, pool_1: 1, block_dropout_1: 0.1378]. Pada RNN, parameter terbaik adalah [gru_hidden_size: 320, num_layers: 2, attention_heads: 4, dense_size: 640, dropout: 0.35, bidirectional: False, activation_fn: mish]. Sedangkan pada *Transformer*, parameter terbaiknya adalah [d_model: 128, num_layers: 3, nhead: 2, dim_feedforward: 256, dropout: 0.2, dense: 256, activation: relu]. Setelah model selesai dilatih, hasilnya akan dievaluasi menggunakan *Accuracy*, *Precision*, *Recall*, dan *F1 Score*.



Gambar 4.4 Visualisasi MFCC Eksperimen A

Ketiga model dievaluasi menggunakan *evaluation metrics Accuracy, Precision, Recall*, dan *F1 Score*. Rincian nilai metrik dari masing-masing model ditampilkan pada Tabel 4.2, sedangkan visualisasi *confusion matrix* untuk setiap model dapat dilihat pada Lampiran 1.

Tabel 4.2 Hasil Eksperimen A

Eksperimen	Model	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
A	CNN	0.8919	0.9028	0.8918	0.891
	RNN GRU	0.9291	0.9313	0.929	0.9289
	Transformer	0.8125	0.8223	0.8124	0.8126

Berdasarkan hasil evaluasi, model RNN menunjukkan performa terbaik di antara ketiga model. Model ini memperoleh akurasi sebesar 0.9291, yang berarti model mampu mengklasifikasikan 92.91% data dengan benar. Nilai *Precision* sebesar 0.9313 menunjukkan bahwa dari seluruh prediksi positif yang dibuat oleh model, sekitar 93.13% benar-benar sesuai dengan label sebenarnya (*true positive*). Sementara itu, nilai *Recall* mencapai 0.929 yang berarti model berhasil mendekripsi sekitar 92.9% dari semua data yang seharusnya diklasifikasikan sebagai positif. Nilai *F1 Score* sebesar 0.9289 menandakan keseimbangan yang baik antara *Precision* dan *Recall*.

Model CNN berada di urutan kedua dengan akurasi sebesar 0.8919 dan *F1 Score* sebesar 0.891. Nilai *Precision* mencapai 0.9028 yang berarti 90.28% dari prediksi positif yang dihasilkan oleh model benar adanya. Sementara itu, nilai *Recall* sebesar 0.8918 menunjukkan kemampuan model dalam mengenali sebagian besar *true positive*, meskipun tidak sebanyak RNN. Model Transformer memperoleh hasil evaluasi yang paling rendah dibanding dua model lainnya, dengan akurasi sebesar 0.8125 dan *F1 Score* sebesar 0.8126. Nilai *Precision* sebesar 0.8223 untuk mengenali prediksi positif dan nilai *Recall* yang hanya mencapai 0.8124 menunjukkan bahwa model masih kurang optimal dalam mendeteksi seluruh kasus yang seharusnya diklasifikasikan sebagai positif.

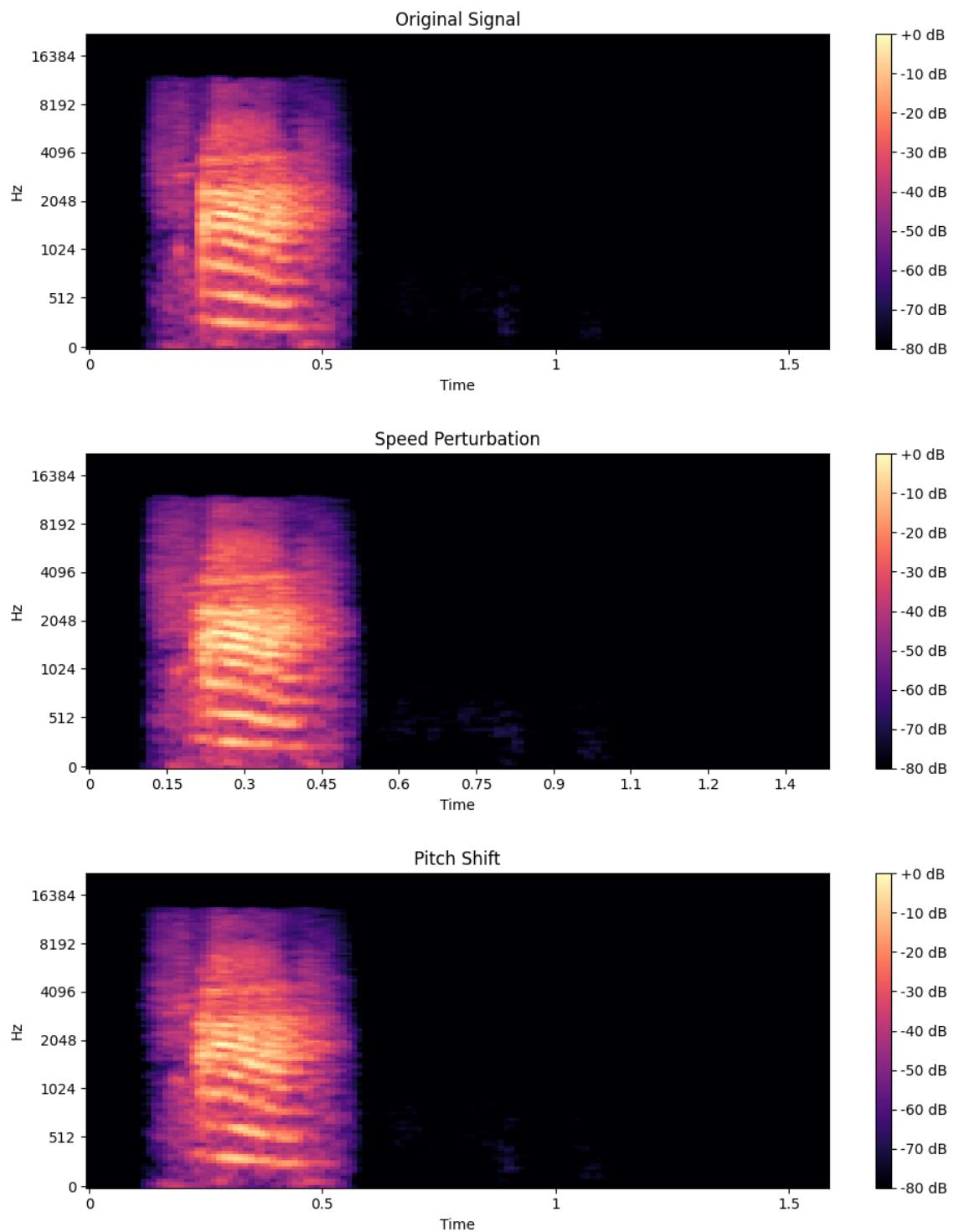
Secara keseluruhan, hasil Eksperimen A menunjukkan bahwa model RNN dengan arsitektur GRU paling efektif dalam menangani data suara fonik keseluruhan yang meliputi *native* dan *non-native speaker*. Model ini mampu mempertahankan performa yang konsisten dalam mengenali variasi suara hasil augmentasi dengan fitur audio yang diekstraksi menggunakan metode MFCC melalui library Librosa. Hasil ini membuktikan bahwa RNN lebih unggul dalam memahami pola sekuensial pada data audio dibanding CNN maupun Transformer.

4.3 Hasil Eksperimen B

Pada eksperimen ini, digunakan metode augmentasi data berupa *speed perturbation* dan *pitch shifting* yang bertujuan untuk meningkatkan variasi data suara dan memperkuat generalisasi model. Setelah augmentasi dilakukan, dilakukan ekstraksi fitur menggunakan metode PNCC (*Power-Normalized Cepstral Coefficients*). Metode ini dipilih karena mampu menghasilkan representasi fitur yang lebih tahan terhadap gangguan noise dan lebih stabil dibandingkan metode MFCC.

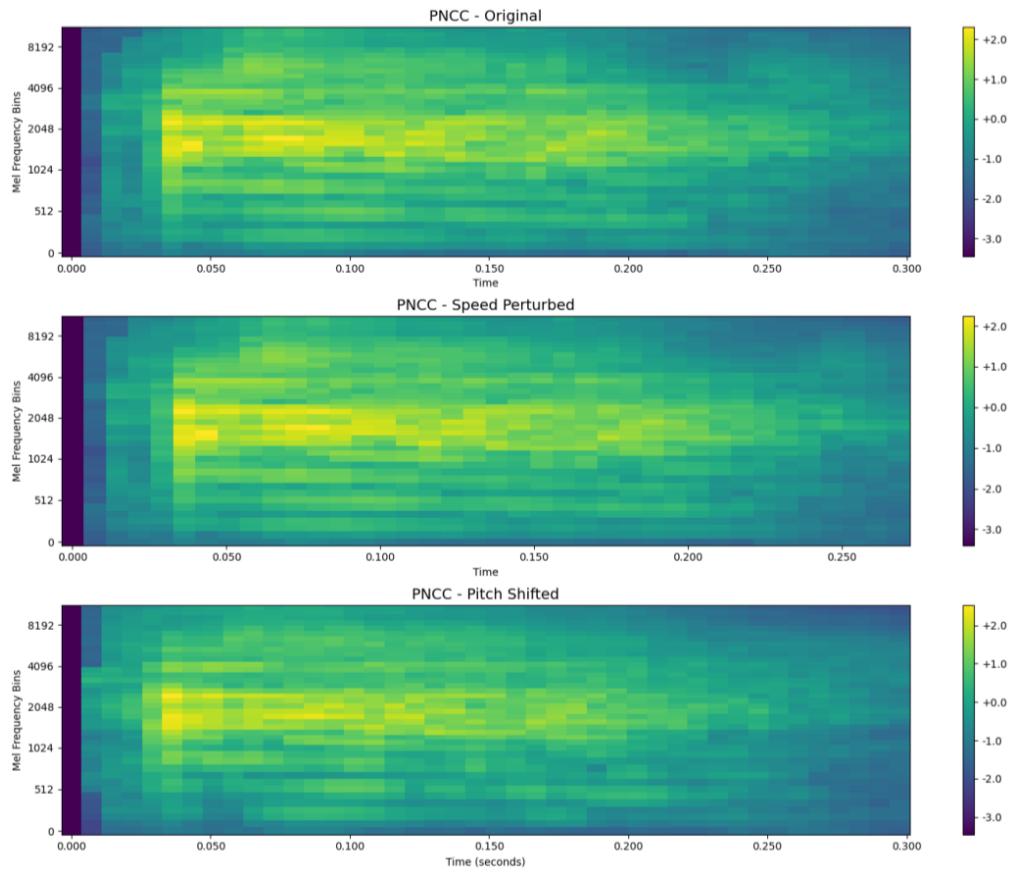
Eksperimen ini menggunakan tiga arsitektur model sebagai *classifier*, yaitu *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Transformer*. Sama seperti pada eksperimen sebelumnya, augmentasi diterapkan pada setiap file audio dalam *dataset*. Salah satu contoh hasil augmentasi divisualisasikan pada huruf ‘a’, sebagaimana ditunjukkan pada Gambar 4.5. Gambar tersebut menggambarkan perubahan bentuk spektrum audio sebelum dan sesudah dilakukan augmentasi. Pada *spectrogram* pertama ditampilkan audio asli, kemudian *spectrogram* kedua menunjukkan hasil augmentasi dengan *speed perturbation*, di mana durasi audio menjadi lebih pendek. Terakhir, *spectrogram* ketiga menampilkan hasil *pitch shifting*, yang menggeser frekuensi suara ke arah yang lebih tinggi tanpa mengubah durasi secara signifikan.

Setelah augmentasi, tahap berikutnya adalah ekstraksi fitur menggunakan PNCC. Gambar 4.6 menampilkan hasil visualisasi fitur PNCC dari ketiga data tersebut. Dari visualisasi tersebut, terlihat bahwa pada audio asli, energi akustik dominan berada pada rentang frekuensi menengah. Setelah dilakukan *speed perturbation*, terlihat bahwa panjang audio berkurang, menandakan percepatan dalam waktu. Sementara itu, hasil *pitch shifting* menunjukkan adanya pergeseran spektrum ke frekuensi yang lebih tinggi, sesuai dengan tujuan dari augmentasi tersebut.



Gambar 4.5 Visualisasi Data Augmentasi Eksperimen B

Tahap selanjutnya adalah proses pelatihan model. Untuk meningkatkan performa masing-masing model, dilakukan pencarian kombinasi hyperparameter terbaik menggunakan *Optuna*, yaitu sebuah *framework* otomatisasi tuning yang efisien. *Optuna* melakukan proses pencarian *hyperparameter* melalui metode *trial* dan menghentikan percobaan yang hasilnya tidak menjanjikan sejak awal (*pruning*).



Gambar 4.6 Visualisasi PNCC Eksperimen B

Untuk Eksperimen B, parameter terbaik pada CNN adalah [n_blocks: 2, dropout: 0.3012, fc_units: 256, use_bn: True, channels_0: 128, kernel_0: 7, pool_0: 2, block_dropout_0: 0.245, channels_1: 256, kernel_1: 3, pool_1: 2, block_dropout_1: 0.1843]. Pada RNN, parameter terbaik adalah [gru_hidden_size: 320, num_layers: 1, attention_heads: 8, dense_size: 384, dropout: 0.25, bidirectional: False, activation_fn: relu]. Sedangkan pada Transformer, parameter terbaiknya adalah [d_model: 128, num_layers: 2, nhead: 4, dim_feedforward: 256, dropout: 0.2, dense: 512, activation: gelu]. Setelah model selesai dilatih menggunakan parameter terbaik, performa dievaluasi menggunakan empat metrik utama: *Accuracy*, *Precision*, *Recall*, dan *F1 Score*. Ringkasan hasil evaluasi ditampilkan pada Tabel 4.3, sedangkan *confusion matrix* untuk masing-masing model dapat dilihat pada Lampiran 2.

Tabel 4.3 Hasil Eksperimen B

Eksperimen	Model	Accuracy	Precision	Recall	F1 Score
B	CNN	0.9071	0.9132	0.9074	0.9067
	RNN GRU	0.9459	0.9499	0.9461	0.946
	Transformer	0.897	0.9014	0.8971	0.8966

Berdasarkan hasil evaluasi, model RNN menunjukkan performa tertinggi, dengan *F1 Score* sebesar 0.946. Angka ini mengindikasikan keseimbangan yang sangat baik antara *Precision* dan *Recall*. Akurasi sebesar 0.9459 menunjukkan bahwa 94.59% dari seluruh data

berhasil diklasifikasikan dengan benar oleh model. *Precision* sebesar 0.9499 menandakan bahwa dari semua prediksi positif yang dihasilkan, sebanyak 94.99% merupakan prediksi yang benar. Sementara itu, *Recall* sebesar 0.9461 berarti model mampu mendeteksi hampir semua instance yang seharusnya diklasifikasikan sebagai positif.

Model CNN menempati posisi kedua terbaik dengan Akurasi sebesar 0.9071 dan *F1 Score* sebesar 0.9067. Nilai *Precision* mencapai 0.9132 dan *Recall* sebesar 0.9074, yang menunjukkan performa cukup baik dan stabil. Model Transformer menunjukkan performa terendah di antara ketiga model, dengan Akurasi sebesar 0.897 dan *F1 Score* sebesar 0.8966. Nilai *Precision* dan *Recall* masing-masing adalah 0.9014 dan 0.8971, yang meskipun masih tergolong baik, namun tidak sekuat dua model lainnya.

Jika dibandingkan dengan hasil pada Eksperimen A, terlihat adanya peningkatan performa di seluruh model, khususnya pada RNN. RNN meningkat dari Akurasi 0.9291 menjadi 0.9459, dan F1 Score dari 0.9289 menjadi 0.946. Demikian pula, CNN dan Transformer juga menunjukkan peningkatan metrik. Dari hasil ini dapat disimpulkan bahwa penggunaan metode ekstraksi fitur PNCC lebih efektif dibandingkan MFCC dalam meningkatkan kinerja model klasifikasi fonik. PNCC mampu menangkap karakteristik akustik dengan lebih baik, terutama setelah data dikenai augmentasi.

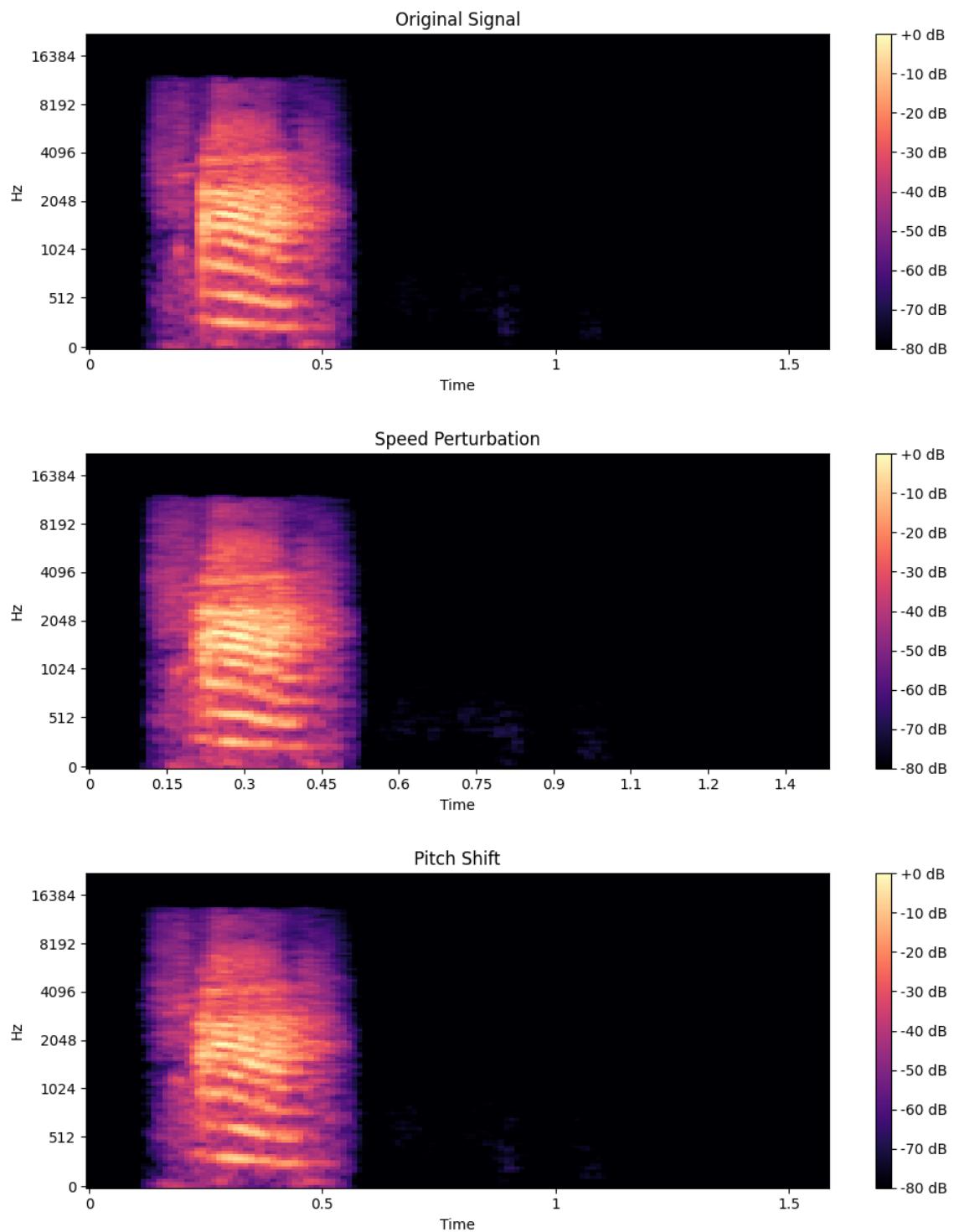
4.4 Hasil Eksperimen C

Eksperimen C fokus pada data yang berasal dari penutur asli Bahasa Inggris (*native speaker*) dengan total sebanyak 494 audio. Tujuan dari eksperimen ini adalah untuk mengetahui sejauh mana model mampu mengenali pelafalan fonik dari penutur yang memiliki artikulasi lebih stabil dan akurat dibandingkan penutur *non-native*. Seperti pada eksperimen sebelumnya, metode augmentasi yang digunakan adalah *speed perturbation* dan *pitch shifting* untuk meningkatkan keragaman data dan ketahanan model terhadap variasi pelafalan.

Tiga model digunakan sebagai classifier dalam eksperimen ini, yaitu *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Transformer*. Proses augmentasi diterapkan pada setiap *file* audio dalam *dataset* sehingga tiap model dilatih dengan data asli dan data yang telah diaugmentasi. Salah satu contoh hasil augmentasi dapat dilihat pada huruf ‘a’, yang divisualisasikan dalam Gambar 4.7. Gambar ini memperlihatkan tiga *spectrogram* yang merepresentasikan proses perubahan audio.

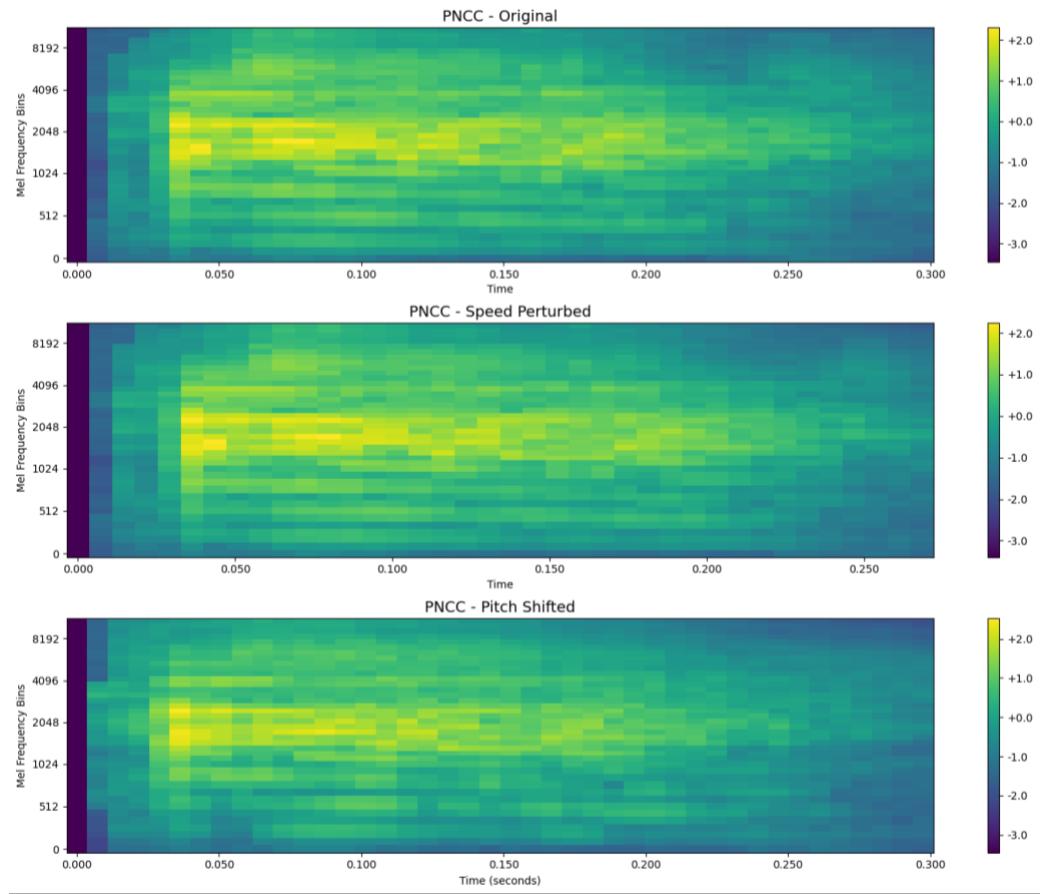
Spectrogram pertama menampilkan bentuk visual dari audio asli sebelum augmentasi. *Spectrogram* kedua menunjukkan hasil setelah diterapkan teknik *speed perturbation*, di mana durasi audio menjadi lebih pendek akibat percepatan kecepatan. Sedangkan *spectrogram* ketiga menunjukkan hasil *pitch shifting*, yang menggeser frekuensi suara ke arah lebih tinggi tanpa mengubah durasi secara signifikan. Visualisasi ini bertujuan untuk memberikan gambaran perbedaan karakteristik sinyal sebelum dan sesudah proses augmentasi dilakukan.

Setelah proses augmentasi selesai, tahap berikutnya adalah ekstraksi fitur menggunakan PNCC. Pemilihan metode PNCC didasarkan pada hasil dari eksperimen sebelumnya yang menunjukkan bahwa PNCC memberikan performa lebih baik dibandingkan MFCC dalam mengklasifikasi fonik.



Gambar 4.7 Visualisasi Data Augmentasi Eksperimen C

Visualisasi hasil ekstraksi fitur PNCC ditunjukkan pada Gambar 4.8 yang mencakup data asli, hasil *speed perturbation*, dan hasil *pitch shifting*. Pada audio asli, distribusi energi tampak dominan di frekuensi menengah, menunjukkan pola pelafalan yang stabil. Setelah diterapkan *speed perturbation*, durasi audio menjadi lebih pendek. Sedangkan pada hasil *pitch shifting*, tampak pergeseran energi spektral ke arah frekuensi yang lebih tinggi tanpa perubahan durasi yang signifikan.



Gambar 4.8 Visualisasi PNCC Eksperimen C

Proses berikutnya adalah pembangunan model dan *tuning hyperparameter* menggunakan *Optuna*. Framework ini melakukan pemilihan kombinasi parameter terbaik berdasarkan hasil evaluasi dari setiap percobaan (*trial*) *training*. Parameter terbaik yang ditemukan untuk CNN adalah [n_blocks: 2, dropout: 0.414, fc_units: 512, use_bn: True, channels_0: 256, kernel_0: 5, pool_0: 2, block_dropout_0: 0.294, channels_1: 256, kernel_1: 7, pool_1: 1, block_dropout_1: 0.136]. Sedangkan untuk RNN, parameter terbaiknya adalah [gru_hidden_size: 320, num_layers: 2, attention_heads: 8, dense_size: 384, dropout: 0.2, bidirectional: False, activation_fn: gelu]. Terakhir, parameter terbaik *Transformer* adalah [d_model: 128, num_layers: 2, nhead: 2, dim_feedforward: 256, dropout: 0.2, dense: 128, activation: gelu].

Setelah model selesai dilatih menggunakan parameter optimal, performa model dievaluasi menggunakan metrik *Accuracy*, *Precision*, *Recall*, dan *F1 Score*. Ringkasan hasil evaluasi ditampilkan pada Tabel 4.4, sementara *confusion matrix* dari masing-masing model dapat dilihat pada Lampiran 3.

Hasil evaluasi menunjukkan bahwa model RNN GRU memberikan performa terbaik dalam mengenali suara fonik dari penutur *native*. Model ini mencapai Akurasi sebesar 0.9293 dan *F1 Score* tertinggi sebesar 0.9385. Selain itu, nilai *Precision* sebesar 0.9454 dan Recall sebesar 0.9402 menunjukkan bahwa model mampu menjaga keseimbangan yang sangat baik antara prediksi positif yang tepat dan keberhasilan mendeteksi seluruh data positif. Hasil eksperimen ini konsisten dengan hasil pada Eksperimen A dan B yang menunjukkan bahwa arsitektur RNN GRU unggul dalam mengklasifikasi audio fonik.

Tabel 4.4 Hasil Eksperimen C

Eksperimen	Model	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
C	CNN	0.9324	0.944	0.9335	0.9329
	RNN GRU	0.9392	0.9454	0.9402	0.9385
	Transformer	0.9257	0.9309	0.926	0.9246

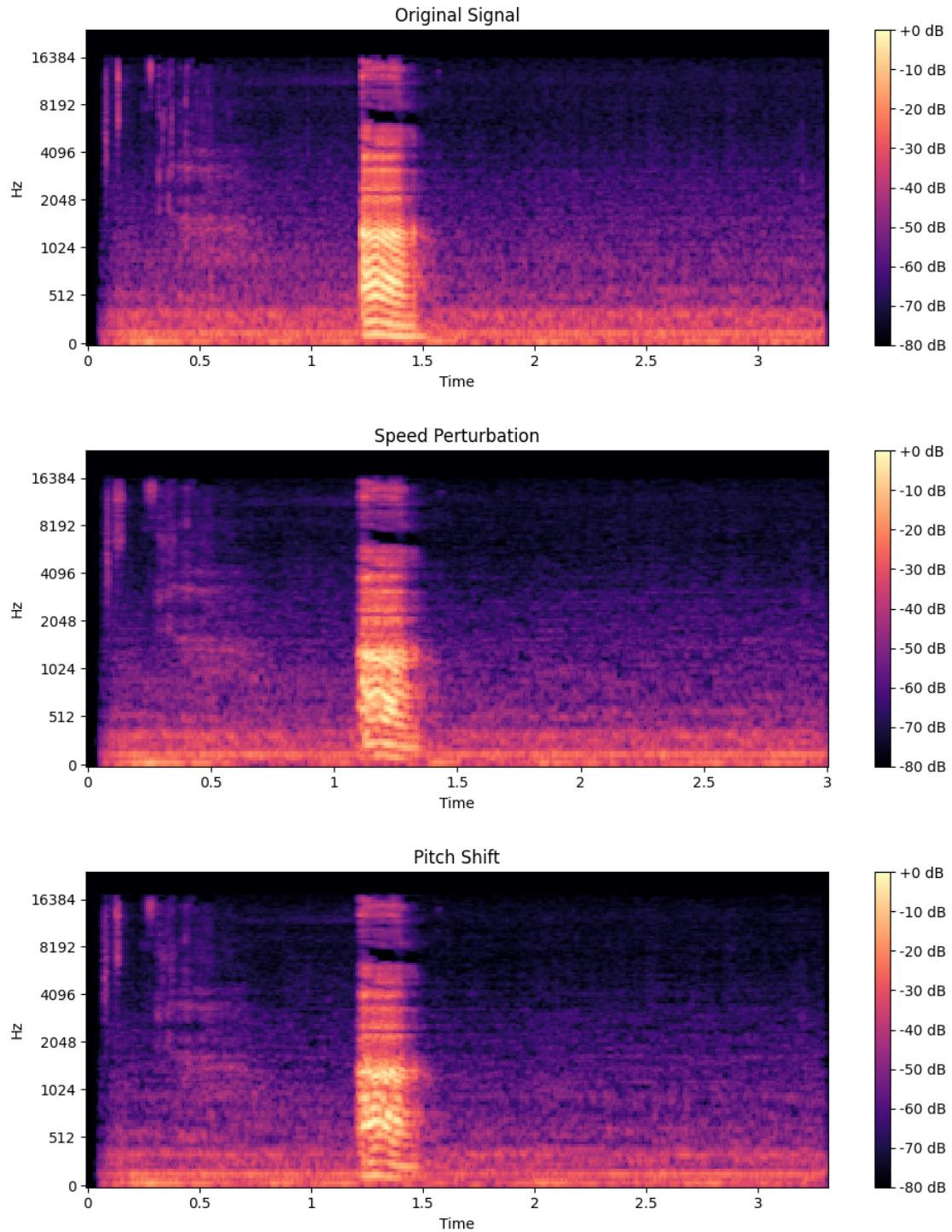
Model CNN berada di urutan kedua terbaik dengan hasil Akurasi sebesar 0.9324 dan *F1 Score* sebesar 0.9329. Nilai *Precision* sebesar 0.944 menunjukkan bahwa 94.4% dari prediksi positif yang dihasilkan oleh model benar, sedangkan *Recall* sebesar 0.9335 menandakan model berhasil mengidentifikasi sebagian besar data positif. Sementara itu, model *Transformer* menempati posisi ketiga terbaik dengan Akurasi sebesar 0.9257 dan *F1 Score* sebesar 0.9246. Nilai *Precision* sebesar 0.9309 dan *Recall* sebesar 0.926 menunjukkan performa yang sedikit lebih rendah dibanding dua model lainnya.

Berdasarkan hasil evaluasi pada Eksperimen C, dapat disimpulkan bahwa model RNN GRU merupakan pilihan terbaik untuk melakukan klasifikasi pelafalan fonik pada *dataset penutur native*. Model ini menunjukkan performa tertinggi secara keseluruhan, dengan nilai *F1 Score* sebesar 0.9385, *Precision* 0.9454, dan *Recall* 0.9402. Jika dibandingkan dengan hasil pada Eksperimen A dan B, performa RNN pada Eksperimen C mengalami peningkatan. Pada Eksperimen A, RNN menghasilkan F1 Score sebesar 0.9289, dan pada Eksperimen B meningkat menjadi 0.946, namun menggunakan data keseluruhan (*native* dan *non-native*). Dalam konteks *dataset native* saja, nilai evaluasi yang tinggi pada Eksperimen C memperkuat bahwa RNN lebih optimal dalam mengenali pola fonik dari penutur *native*.

4.5 Hasil Eksperimen D

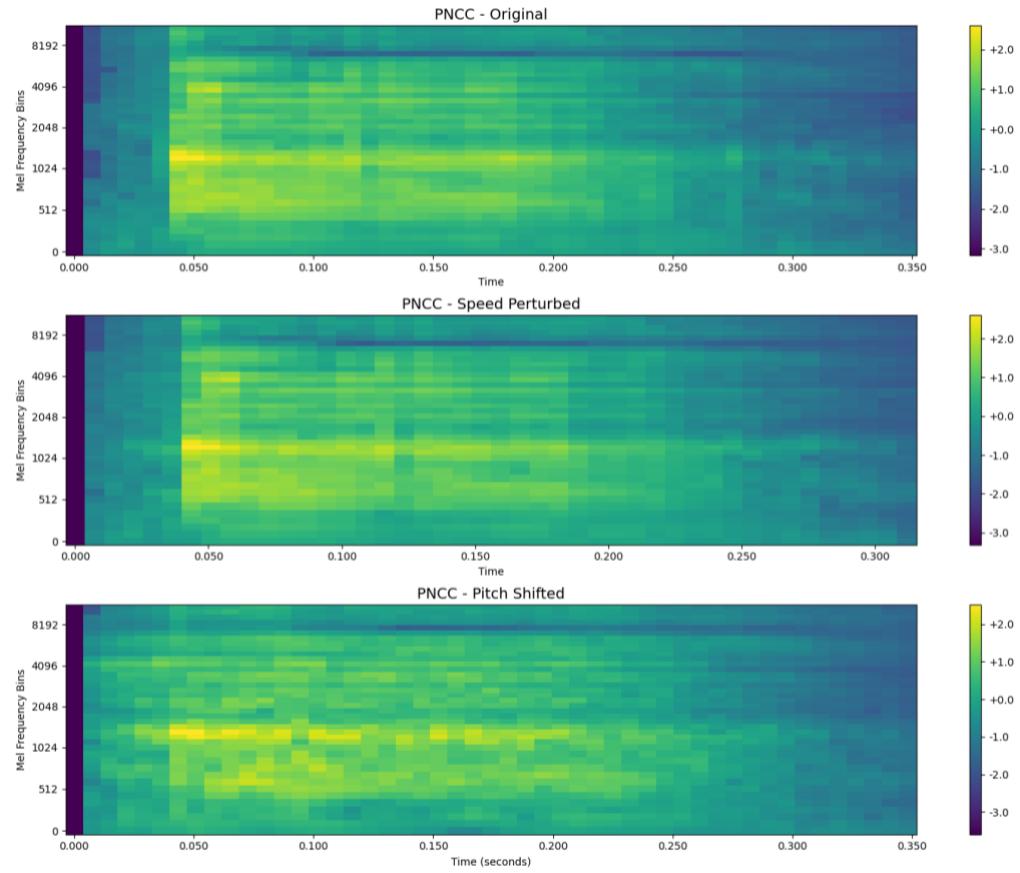
Eksperimen kali ini hanya akan menggunakan data dari *non-native speaker*. Jumlah audio yang digunakan adalah 494 audio yang terdiri dari enam *speaker* asal Jawa, empat *speaker* asal Papua, tiga *speaker* asal Bali, tiga *speaker* asal Batam, dan tiga *speaker* dari Madura. Metode data augmentasi yang digunakan adalah *speed perturbation* dan *pitch shifting* dengan ekstraksi fitur PNCC. Tiga model yang digunakan sebagai *classifier* adalah *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Transformers*.

Metode augmentasi data yang digunakan dalam eksperimen ini adalah *speed perturbation* dan *pitch shifting*, yang sebelumnya juga diterapkan pada eksperimen-eksperimen sebelumnya. Visualisasi *spectrogram* hasil augmentasi ditampilkan pada Gambar 4.9. Contoh yang digunakan adalah audio dari huruf ‘a’, yang divisualisasikan dalam tiga bentuk *spectrogram*. *Spectrogram* pertama menggambarkan audio sebelum dilakukan augmentasi atau dalam bentuk aslinya. *Spectrogram* kedua merupakan hasil dari augmentasi *speed perturbation* yang terlihat dari durasi sinyal audio yang menjadi lebih pendek akibat percepatan kecepatan *playback*. Sedangkan *spectrogram* ketiga memperlihatkan hasil dari *pitch shifting*, hasilnya menyebabkan adanya pergeseran spektral ke arah frekuensi yang lebih tinggi, tetapi tidak mengubah durasi sinyal secara signifikan. Visualisasi ini bertujuan untuk menunjukkan perubahan yang terjadi pada sinyal setelah melalui proses augmentasi.



Gambar 4.9 Visualisasi Data Augmentasi Eksperimen D

Langkah selanjutnya setelah augmentasi adalah ekstraksi PNCC. Metode ini dipilih karena terbukti memberikan hasil yang lebih baik dibandingkan MFCC dengan *library Librosa*. Visualisasi dari ekstraksi PNCC dapat dilihat pada Gambar 4.10. Pada visualisasi pertama, distribusi energi dominan di frekuensi menengah. Pada *spectrogram* kedua, terlihat bahwa durasi audio menjadi lebih cepat karena efek dari *speed perturbation*. Lalu, *spectrogram* ketiga menunjukkan perubahan *pitch* yang menuju frekuensi lebih tinggi akibat *pitch shifting*.



Gambar 4.10 Visualisasi PNCC Eksperimen D

Langkah selanjutnya setelah ekstraksi PNCC adalah *modeling*. Pada eksperimen ini, digunakan Optuna untuk mencari parameter terbaik dari CNN, RNN dan Transformer. Setelah model selesai dilatih, hasilnya akan dievaluasi menggunakan *Accuracy*, *Precision*, *Recall*, dan *F1 Score* yang hasilnya dijabarkan pada Tabel 4.5. Visualisasi *confusion matrix* dapat dilihat pada Lampiran 4.

Hasil parameter terbaik dari Optuna untuk CNN adalah [n_blocks: 3, dropout: 0.258, fc_units: 128, use_bn: True, channels_0: 128, kernel_0: 7, pool_0: 2, block_dropout_0: 0.226, channels_1: 128, kernel_1: 5, pool_1: 2, block_dropout_1: 0.306, channels_2: 256, kernel_2: 3, pool_2: 2, block_dropout_2: 0.105]. Parameter terbaik untuk RNN adalah [gru_hidden_size: 384, num_layers: 2, attention_heads: 4, dense_size: 512, dropout: 0.250, bidirectional: False, activation_fn: mish]. Selanjutnya, parameter terbaik dari Transformer adalah [d_model: 128, num_layers: 2, nhead: 2, dim_feedforward: 512, dropout: 0.350, dense: 512, activation: gelu].

Tabel 4.5 Hasil Eksperimen D

Eksperimen	Model	Accuracy	Precision	Recall	F1 Score
D	CNN	0.8923	0.8989	0.8939	0.8906
	RNN GRU	0.9293	0.9389	0.9301	0.9305
	Transformer	0.8519	0.8714	0.8537	0.8555

Hasil evaluasi dari *Convolutional Neural Network*, *Recurrent Neural Network*, dan *Transformer* yang dilatih menggunakan ekstraksi PNCC dan augmentasi *speed perturbation* dan *pitch shifting* ditampilkan pada Tabel 4.9. Hasil menunjukkan bahwa CNN menghasilkan Akurasi 0.8923 dan *F1 Score* 0.8906 yang berarti model cukup seimbang antara *Precision* dan *Recall* dan dapat memprediksi dengan benar 89.23% keseluruhan data. Nilai *Precision* 0.8989 menunjukkan bahwa seluruh prediksi positif yang dibuat oleh model, 89.89% merupakan *true positive*, sedangkan *Recall* bernilai 0.8939 berarti model berhasil mendeteksi 89.39% keseluruhan *true positive*. Model RNN menunjukkan nilai yang lebih baik dengan Akurasi 0.9293 dan *F1 Score* 0.9305. Didapatkan nilai *Precision* 0.9389 yang berarti model seluruh prediksi positif yang dibuat model, 93.89% merupakan *true positive* dan nilai *Recall* 0.9301 menunjukkan model berhasil mendeteksi 93.01% seluruh *true positive*. Lalu, Model *Transformer* menghasilkan Akurasi 0.8519 dan *F1 Score* 0.8555 dengan nilai *Precision* 0.8714 yang berarti model menghasilkan 87.14% prediksi positif dan *Recall* bernilai 0.8537 menunjukkan bahwa model berhasil memprediksi 85.37% dari keseluruhan *true positive*.

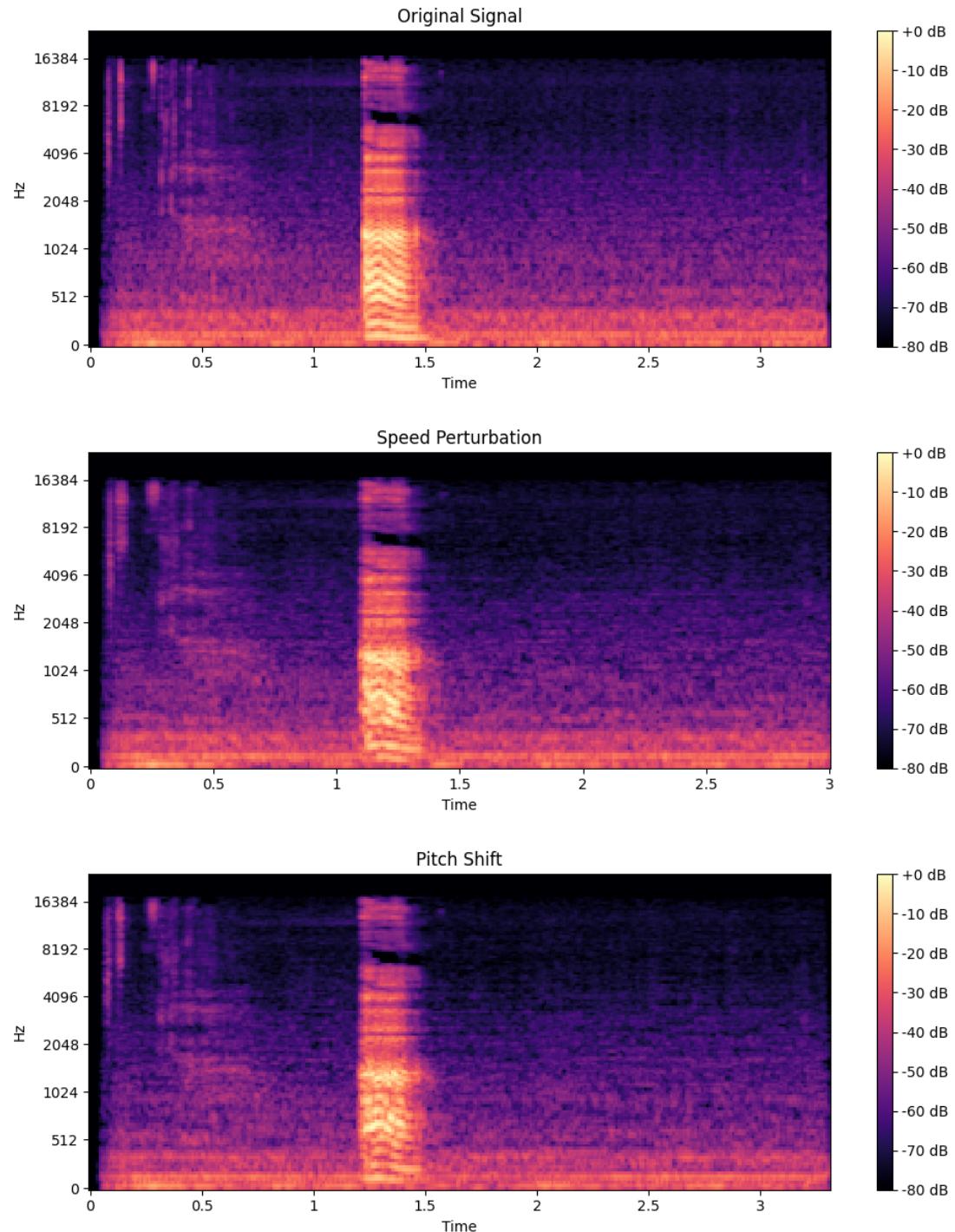
Dibandingkan dengan hasil pada Eksperimen C yang menggunakan data dari penutur *native*, maka dapat terlihat bahwa hasil evaluasi pada Eksperimen D, yang menggunakan data *non-native*, cenderung lebih rendah di semua model. Model CNN pada data *native* berhasil mencapai akurasi 0.9324 dan *F1 Score* 0.9329, sedangkan pada data *non-native* hanya mencapai akurasi 0.8923 dan *F1 Score* 0.8906. Begitu pula pada model RNN, walaupun tetap menjadi model dengan performa terbaik di antara ketiganya, hasil akurasi dan *F1 Score* pada data *native* lebih tinggi (0.9392 dan 0.9385) dibandingkan dengan hasil pada data *non-native* (0.9293 dan 0.9305). Model *Transformer* juga mengalami penurunan performa yang cukup signifikan. Pada data *native*, *Transformer* memperoleh akurasi 0.9257 dan *F1 Score* 0.9246, sedangkan pada data *non-native* hanya mencapai akurasi 0.8519 dan *F1 Score* 0.8555.

Penurunan hasil evaluasi menunjukkan bahwa data dari penutur *non-native* lebih sulit dikenali oleh model. Hal ini disebabkan oleh variasi pelafalan yang lebih tinggi, perbedaan aksen, intonasi, atau kejelasan pengucapan dari masing-masing penutur *non-native*, sehingga memengaruhi representasi fitur audio yang diekstraksi dan diproses oleh model. Secara keseluruhan, eksperimen ini memperkuat beberapa eksperimen sebelumnya bahwa model RNN GRU tetap menjadi model dengan performa terbaik dalam tugas klasifikasi fonik, baik pada data *native* maupun *non-native*.

4.6 Hasil Eksperimen E

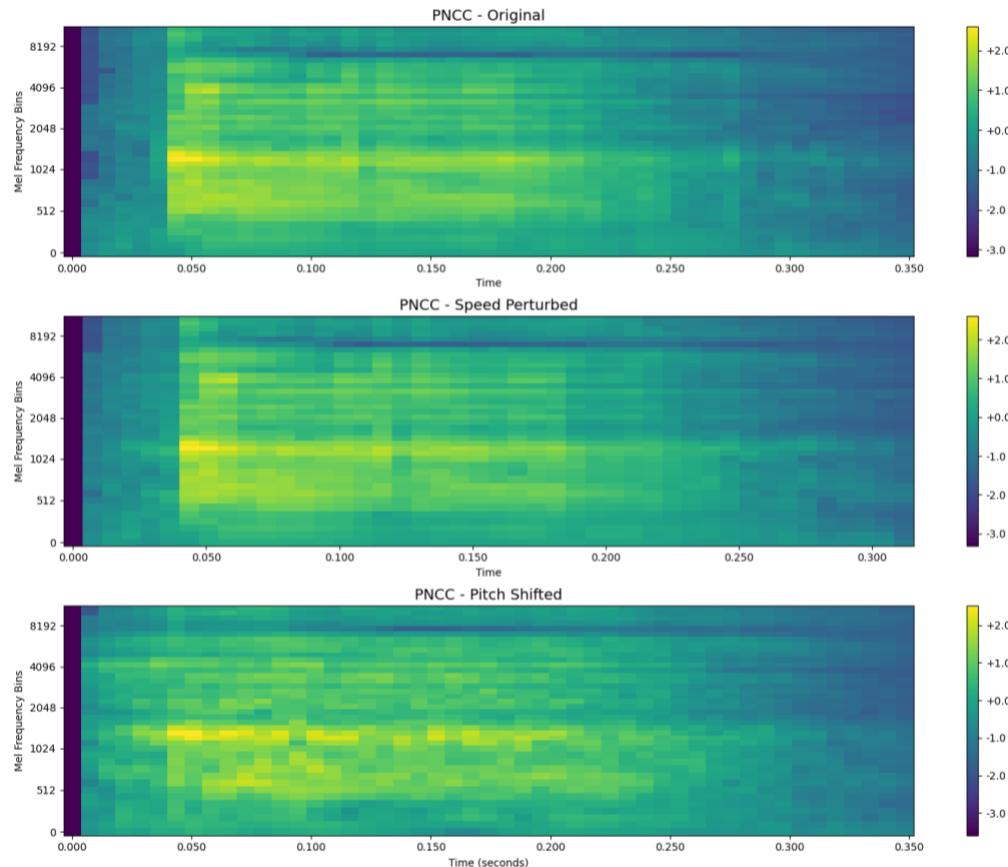
Eksperimen E menggunakan *dataset* dari *non-native speaker* dengan jumlah 494 audio yang terdiri dari enam *speaker* asal Jawa, empat *speaker* asal Papua, tiga *speaker* asal Bali, tiga *speaker* asal Batam, dan tiga *speaker* dari Madura. Perbedaan antara Eksperimen D dan Eksperimen E adalah Eksperimen E melatih audio dari setiap *speaker* untuk membandingkan apakah logat dari setiap daerah memengaruhi hasil prediksi dari model dan digunakan masing-masing dua parameter augmentasi untuk tiap metode yang diaplikasikan. Metode data augmentasi yang digunakan adalah *speed perturbation* dan *pitch shifting* dengan ekstraksi fitur PNCC. Tiga model yang digunakan sebagai *classifier* adalah *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Transformers*.

Sama seperti eksperimen sebelumnya, pertama dilakukan augmentasi pada *dataset* menggunakan *speed perturbation* dan *pitch shifting*. Hasil visualisasi dari augmentasi ditampilkan pada Gambar 4.11 yang bertujuan untuk melihat perbedaan antara data asli dan hasil dari augmentasi *speed perturbation* dan *pitch shifting*. *Spectrogram* pertama merupakan hasil dari audio asli, *spectrogram* kedua merupakan sinyal setelah dilakukan augmentasi *speed perturbation* yang hasilnya durasi sinyal menjadi lebih cepat, dan *spectrogram* terakhir merupakan hasil dari *pitch shifting* dengan adanya pergeseran ke spektral yang lebih tinggi.



Gambar 4.11 Visualisasi Data Augmentasi Eksperimen E

Langkah selanjutnya setelah augmentasi adalah ekstraksi PNCC. Visualisasi dari ekstraksi PNCC ditampilkan pada Gambar 4.12. Pada sinyal asli, pola energi suara terlihat tersebar rata dengan dominasi di frekuensi menengah. Setelah dilakukan *speed perturbation* (gambar kedua), durasi audio menjadi lebih pendek. Sedangkan pada gambar terakhir, pola spektralnya bergeser ke frekuensi lebih tinggi yang berarti *pitch* suara meningkat efek dari *pitch shifting*.



Gambar 4.12 Visualisasi PNCC Eksperimen E

Setelah dilakukan proses ekstraksi fitur menggunakan metode PNCC, tahap berikutnya adalah modeling, di mana digunakan Optuna untuk mencari kombinasi parameter terbaik dari masing-masing model. Optuna membantu proses tuning secara otomatis dan efisien, sehingga dapat mempercepat pencarian konfigurasi model yang optimal. Pada eksperimen ini, pencarian parameter dilakukan secara khusus untuk setiap subset data berdasarkan asal daerah speaker, yaitu Jawa, Papua, Bali, Batam, dan Madura.

4.6.1. Convolutional Neural Network

Pencarian parameter terbaik ini menghasilkan konfigurasi yang berbeda untuk masing-masing daerah, sebagaimana ditunjukkan pada Tabel 4.6. Setelah parameter optimal diperoleh, setiap model dilatih menggunakan data dari masing-masing daerah secara terpisah. Evaluasi dilakukan menggunakan empat metrik utama, yaitu *Accuracy*, *Precision*, *Recall*, dan *F1 Score*, dengan hasil akhir dijabarkan pada Tabel 4.7, sedangkan visualisasi *confusion matrix* untuk masing-masing model dan wilayah dapat dilihat pada Lampiran 5.

Tabel 4.6 Parameter Terbaik CNN Eksperimen E

Model	Daerah	Parameter Terbaik
CNN	Bali	n_blocks: 2, dropout: 0.297, fc_units: 512, use_bn: True, channels_0: 128, kernel_0: 3, pool_0: 1, block_dropout_0: 0.345, channels_1: 64, kernel_1: 3, pool_1: 2, block_dropout_1: 0.387
	Batam	n_blocks: 2, dropout: 0.403, fc_units: 256, use_bn: True, channels_0: 256, kernel_0: 7, pool_0: 1, block_dropout_0: 0.372, channels_1: 64, kernel_1: 3, pool_1: 1, block_dropout_1: 0.37
	Jawa	n_blocks: 4, dropout: 0.475, fc_units: 512, use_bn: True, channels_0: 64, kernel_0: 3, pool_0: 2, block_dropout_0: 0.358, channels_1: 64, kernel_1: 5, pool_1: 1, block_dropout_1: 0.219, channels_2: 128, kernel_2: 3, pool_2: 2, block_dropout_2: 0.298, channels_3: 64, kernel_3: 3, pool_3: 2, block_dropout_3: 0.161
	Madura	n_blocks: 4, dropout: 0.225, fc_units: 128, use_bn: True, channels_0: 256, kernel_0: 7, pool_0: 2, block_dropout_0: 0.376, channels_1: 128, kernel_1: 3, pool_1: 2, block_dropout_1: 0.397, channels_2: 256, kernel_2: 3, pool_2: 1, block_dropout_2: 0.392, channels_3: 128, kernel_3: 3, pool_3: 2, block_dropout_3: 0.332
	Papua	n_blocks: 2, dropout: 0.319, fc_units: 256, use_bn: False, channels_0: 256, kernel_0: 5, pool_0: 2, block_dropout_0: 0.232, channels_1: 128, kernel_1: 7, pool_1: 2, block_dropout_1: 0.356

Tabel 4.7 Hasil CNN Eksperimen E

Eksperimen	Model	Daerah	Accuracy	Precision	Recall	F1 Score
E	CNN	Bali	0.8205	0.8637	0.8205	0.8102
		Batam	0.9487	0.9231	0.9487	0.9341
		Jawa	0.9103	0.9448	0.9103	0.9135
		Madura	0.9487	0.9396	0.9487	0.9385
		Papua	0.9904	0.9923	0.9904	0.9902

Berdasarkan hasil evaluasi pada Tabel 4.7, ditemukan bahwa performa terbaik diperoleh oleh model CNN ketika digunakan untuk melatih *dataset* dari *speaker* Papua. Model tersebut mencapai Akurasi sebesar 0.9904, *F1 Score* sebesar 0.9902, *Precision* 0.9923, dan *Recall* 0.9904. Nilai *F1 Score* yang hampir sempurna menunjukkan bahwa model tidak hanya tepat dalam prediksi positif, tetapi juga tidak banyak melewatkannya data positif yang seharusnya terdeteksi. Hasil terbaik berikutnya diperoleh dari data *speaker* Madura, yang meskipun memiliki nilai akurasi yang sama dengan Batam, tetapi nilai *F1 Score*-nya lebih unggul

(0.9385), sehingga Madura menempati posisi kedua. Data dari Batam menempati posisi ketiga, diikuti oleh Jawa di posisi keempat, dengan *F1 Score* sebesar 0.9135. Data dari Bali menempati posisi terakhir, dengan *F1 Score* sebesar 0.8102, yang menunjukkan bahwa model memiliki kesulitan lebih besar dalam mengenali pelafalan dari *speaker* asal Bali. Hal ini dapat disebabkan oleh variasi pelafalan yang lebih beragam atau kualitas data audio yang tidak setara dengan daerah lain.

Secara keseluruhan, Eksperimen E pada model CNN menunjukkan bahwa asal daerah speaker dapat memengaruhi performa model CNN, baik dari segi akurasi maupun konsistensi prediksi. Model lebih mudah belajar dari pelafalan yang konsisten dan jelas, seperti yang ditemukan pada data Papua, sementara performa menurun pada data dengan pelafalan yang lebih bervariasi atau tidak konsisten.

4.6.2. Recurrent Neural Network

Selanjutnya, pada model RNN, proses pencarian parameter terbaik kembali dilakukan menggunakan *Optuna*, dengan metode yang sama seperti sebelumnya, yaitu menyesuaikan parameter untuk setiap *subset* data berdasarkan asal daerah *speaker*. Konfigurasi *hyperparameter* terbaik yang diperoleh untuk masing-masing wilayah dijabarkan pada Tabel 4.8. *Optuna* berperan penting dalam proses ini karena dapat mengevaluasi banyak kombinasi parameter dengan lebih efisien dan sistematis, serta menghentikan percobaan yang tidak menjanjikan di awal. Setelah parameter terbaik berhasil ditemukan, model RNN dilatih menggunakan konfigurasi tersebut untuk masing-masing daerah.

Tahap selanjutnya adalah proses evaluasi performa model dengan menggunakan empat metrik utama, yaitu *Accuracy*, *Precision*, *Recall*, dan *F1 Score*. Seluruh hasil evaluasi dari tiap daerah dirangkum dan disajikan pada Tabel 4.9, sedangkan visualisasi hasil klasifikasi dalam bentuk *confusion matrix* dapat dilihat pada Lampiran 5.

Tabel 4.8 Parameter Terbaik RNN Eksperimen E

Model	Daerah	Parameter Terbaik
RNN	Bali	gru_hidden_size: 320, num_layers: 1, attention_heads: 8, dense_size: 512, dropout: 0.350, bidirectional: False, activation_fn: gelu
	Batam	gru_hidden_size: 320, num_layers: 1, attention_heads: 8, dense_size: 640, dropout: 0.350, bidirectional: True, activation_fn: relu
	Jawa	gru_hidden_size: 256, num_layers: 3, attention_heads: 4, dense_size: 384, dropout: 0.250, bidirectional: False, activation_fn: mish
	Madura	gru_hidden_size: 256, num_layers: 1, attention_heads: 4, dense_size: 512, dropout: 0.350, bidirectional: False, activation_fn: mish
	Papua	gru_hidden_size: 384, num_layers: 3, attention_heads: 8, dense_size: 512, dropout: 0.350, bidirectional: False, activation_fn: gelu

Tabel 4.9 Hasil RNN Eksperimen E

Eksperimen	Model	Daerah	Accuracy	Precision	Recall	F1 Score
E	RNN	Bali	0.9615	0.9327	0.9615	0.9451
		Batam	0.9615	0.9365	0.9615	0.9464
		Jawa	0.9615	0.9327	0.9615	0.961
		Madura	0.9615	0.9423	0.9615	0.9487
		Papua	0.9904	0.9923	0.9904	0.9902

Hasil evaluasi model *Recurrent Neural Network* (RNN) menunjukkan performa yang secara umum lebih baik dibandingkan CNN di hampir seluruh wilayah asal *speaker non-native*. Performa tertinggi kembali diraih oleh data dari speaker asal Papua, dengan nilai *F1 Score* sebesar 0.9902 dan Akurasi 0.9904. Nilai ini identik dengan hasil terbaik CNN di wilayah yang sama pada eksperimen sebelumnya, dan memperkuat bahwa pelafalan dari *speaker Papua* memberikan representasi sinyal yang paling konsisten dan mudah dipelajari oleh model. Hal ini terlihat dari keseimbangan antara *Precision* dan *Recall* yang hampir sempurna, menunjukkan bahwa model tidak hanya tepat dalam melakukan klasifikasi positif, tapi juga tidak melewatkannya banyak *instance* yang seharusnya dikenali.

Meskipun akurasi model untuk wilayah lain juga tinggi dan relatif seragam, nilai *F1 Score*-nya bervariasi, seberapa seimbang model dalam memprediksi benar dan tidak melewatkannya. Setelah Papua, performa terbaik secara berurutan ditunjukkan oleh data Madura (*F1 Score* 0.9487), Batam (*F1 Score* 0.9464), dan Bali (*F1 Score* 0.9451). Meskipun nilai akurasinya hampir identik, perbedaan pada *F1 Score* ini menunjukkan bahwa model bisa memiliki performa yang berbeda tergantung bagaimana pelafalan masing-masing daerah memengaruhi pola fitur yang dihasilkan dan dipelajari.

Jika dibandingkan dengan hasil CNN, terlihat bahwa wilayah dengan performa terendah tidak selalu sama untuk setiap model. CNN sebelumnya mencatat performa terendah pada data dari Bali, sedangkan pada RNN, justru wilayah Jawa mencatat nilai *F1 Score* terendah. Hal ini menunjukkan bahwa karakteristik pelafalan antar daerah memengaruhi bagaimana model belajar dan mengenali pola suara, tergantung pada arsitektur model yang digunakan. Namun demikian, wilayah Papua secara konsisten mencatat hasil terbaik di kedua model, yang memperkuat pernyataan bahwa logat, kejelasan pelafalan, serta konsistensi artikulasi dari *speaker* di daerah tersebut berkontribusi besar terhadap keberhasilan model dalam mengenali fonik secara akurat. Dengan demikian, selain kualitas data dan arsitektur model, logat atau aksen dari penutur juga terbukti menjadi faktor penting dalam memengaruhi performa sistem klasifikasi fonik.

4.6.3. Transformer

Pada model *Transformer*, dilakukan proses yang sama seperti pada dua model sebelumnya, yaitu pencarian *hyperparameter* terbaik menggunakan Optuna. Tujuan dari proses ini adalah untuk menyesuaikan parameter secara spesifik terhadap karakteristik data dari masing-masing daerah, sehingga model dapat bekerja secara optimal. Parameter terbaik yang ditemukan untuk tiap daerah secara lengkap dijabarkan pada Tabel 4.10.

Tabel 4.10 Parameter Terbaik *Transformer* Eksperimen E

Model	Daerah	Parameter Terbaik
Transformer	Bali	d_model: 128, num_layers: 4, nhead: 4, dim_feedforward: 256, dropout: 0.400, dense: 512, activation: gelu
	Batam	d_model: 128, num_layers: 2, nhead: 8, dim_feedforward: 512, dropout: 0.250, dense: 512, activation: gelu
	Jawa	d_model: 256, num_layers: 2, nhead: 8, dim_feedforward: 512, dropout: 0.350, dense: 256, activation: relu
	Madura	d_model: 64, num_layers: 3, nhead: 4, dim_feedforward: 1024, dropout: 0.200, dense: 512, activation: relu
	Papua	d_model: 256, num_layers: 3, nhead: 8, dim_feedforward: 512, dropout: 0.250, dense: 512, activation: gelu

Dengan menggunakan parameter hasil *tuning* Optuna, model *Transformer* kemudian dilatih menggunakan data dari masing-masing daerah. Evaluasi dilakukan menggunakan empat metrik utama: *Accuracy*, *Precision*, *Recall*, dan *F1 Score*, dengan hasil akhir dijabarkan pada Tabel 4.11. Sementara itu, visualisasi *confusion matrix* untuk setiap daerah dapat dilihat pada Lampiran 5.

Tabel 4.11 Hasil *Transformer* Eksperimen E

Eksperimen	Model	Daerah	Accuracy	Precision	Recall	F1 Score
E	Transformer	Bali	0.9615	0.9423	0.9615	0.9487
		Batam	0.9615	0.9365	0.9615	0.9464
		Jawa	0.9359	0.9388	0.9359	0.935
		Madura	0.8974	0.9107	0.8974	0.8912
		Papua	0.9808	0.9827	0.9808	0.9806

Hasil evaluasi menunjukkan bahwa meskipun performa *Transformer* masih berada di bawah model RNN, model ini tetap menunjukkan konsistensi dalam mengklasifikasi fonik pada sebagian besar wilayah. Papua kembali menjadi wilayah dengan performa terbaik, dengan nilai Akurasi sebesar 0.9808 dan *F1 Score* sebesar 0.9806. *Precision* yang mencapai 0.9827 dan *Recall* 0.9808 mengindikasikan bahwa model mampu menghasilkan prediksi yang akurat dan

seimbang pada data dari penutur Papua. Untuk wilayah Bali dan Batam, *Transformer* menghasilkan nilai Akurasi yang identik, yaitu 0.9615. Namun, dilihat dari nilai F1 Score, Bali berada sedikit di atas Batam dengan skor 0.9487, sedangkan Batam mencatatkan 0.9464. Perbedaan ini menunjukkan bahwa meskipun tingkat klasifikasi keseluruhan serupa, distribusi kesalahan prediksi, terutama keseimbangan antara *Precision* dan *Recall*, masih bervariasi antar wilayah. Sedangkan pada wilayah Jawa dan Madura menunjukkan kecenderungan penurunan performa. Pada data Jawa, *Transformer* mencatatkan Akurasi sebesar 0.8974 dan *F1 Score* 0.935, sedangkan Madura memiliki *F1 Score* terendah yaitu 0.8912, dengan akurasi yang sama. Hal ini menunjukkan bahwa model mengalami lebih banyak kesalahan klasifikasi pada data dari *speaker* Madura, hal ini dikarenakan pelafalan yang tidak seragam atau lebih kompleks secara fonik.

Dibandingkan dengan hasil klasifikasi keseluruhan model, RNN tetap menempati posisi tertinggi secara konsisten dalam seluruh wilayah uji coba, dengan skor akurasi dan *F1 Score* yang tinggi dan merata. *Transformer* berada pada posisi kedua dengan kinerja yang kompetitif di wilayah tertentu seperti Papua, Bali, dan Batam, tetapi mengalami penurunan pada wilayah lain, terutama Madura. Sementara itu, CNN menunjukkan tingkat variabilitas yang lebih besar antar wilayah, yang mengindikasikan bahwa performanya lebih dipengaruhi oleh perbedaan karakteristik pelafalan dari tiap daerah.

Secara keseluruhan, hasil Eksperimen E memperkuat pernyataan bahwa RNN GRU merupakan model yang paling adaptif untuk tugas klasifikasi fonik, terutama pada *dataset non-native* dengan ragam aksen dan pelafalan. Selain itu, perbedaan hasil antar wilayah juga memperjelas bahwa logat, aksen, dan kejelasan pelafalan memiliki dampak langsung terhadap performa model. Hal ini diperkuat dengan hasil bahwa wilayah dengan hasil terendah berbeda-beda untuk setiap model, tetapi Papua konsisten mencatatkan hasil tertinggi, menandakan bahwa karakteristik pelafalan dari wilayah ini lebih mudah dikenali oleh ketiga model yang diuji.

4.7 Diskusi

Pada bagian ini, akan disimpulkan hasil dari serangkaian eksperimen yang dilakukan untuk mengevaluasi hasil dan dampak dari pengaruh teknik *pre-processing*, ekstraksi fitur, dan arsitektur model terhadap performa klasifikasi fonik.

Penelitian ini melakukan lima eksperimen untuk mengevaluasi berbagai metode ekstraksi fitur dan arsitektur model untuk mengklasifikasi fonik menggunakan data *native* dan *non-native speaker*. Tiga model yang digunakan adalah *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Transformer*.

Pada Eksperimen A, dilakukan ekstraksi fitur MFCC dengan *library* Librosa dengan data augmentasi *speed perturbation* dan *pitch shifting*. Hasil evaluasi terbaik diperoleh oleh RNN (*F1 Score*: 0.9289 dan Akurasi 0.9291), diikuti oleh CNN, kemudian *Transformer*. Hal ini menunjukkan bahwa RNN merupakan model paling efektif dalam menangani audio fonik yang diekstraksi menggunakan MFCC Librosa.

Selanjutnya, Eksperimen B menggunakan PNCC sebagai ekstraksi fitur dan hasilnya menunjukkan peningkatan pada ketiga model. Hasil paling unggul tetap dipegang RNN dengan

nilai Akurasi 0.9459 dan *F1 Score* 0.946. Model terbaik kedua merupakan CNN dan disusul oleh *Transformer*. Kesimpulannya adalah PNCC merupakan ekstraksi fitur yang lebih efektif daripada MFCC Librosa dalam klasifikasi fonik.

Eksperimen C dan D melakukan perbandingan performa antara *dataset native* (Eksperimen C) dan *non-native speaker* (Eksperimen D). Hasilnya menunjukkan bahwa model yang dilatih menggunakan data *native* menghasilkan performa yang lebih unggul dibandingkan data *non-native* dan RNN masih menjadi model yang paling unggul dalam kedua eksperimen ini. Pada RNN, *F1 Score* meningkat dari 0.9305 (*non-native*) menjadi 0.9385 (*native*) dan Akurasi meningkat dari 0.9293 (*non-native*) menjadi 0.9392 (*native*). Model terbaik kedua merupakan CNN dan diikuti oleh *Transformer*. Hal ini menunjukkan bahwa variasi dan kualitas data *native speaker* berkontribusi dalam peningkatan akurasi klasifikasi fonik.

Eksperimen E memperluas analisis dengan membandingkan hasil klasifikasi berdasarkan tiap daerah asal *non-native speaker*. Dilatih dengan tiga model yang sama, hasilnya menunjukkan bahwa Papua konsisten menjadi daerah yang memperoleh nilai tertinggi pada seluruh model dengan *F1 Score* mencapai 0.9902 dan nilai Akurasi sebesar 0.9904 pada CNN dan RNN, serta nilai 0.9806 untuk *F1 Score* dan 0.9808 untuk *Transformer*.

Hasil Eksperimen E memperlihatkan bahwa model memberikan performa lebih tinggi ketika dilatih menggunakan data *native* dibandingkan data *non-native*. Selain itu, ditemukan bahwa performa model juga dipengaruhi oleh logat atau aksen daerah dari penutur *non-native*. Hal ini terlihat dari perbedaan hasil antar wilayah, di mana wilayah dengan hasil terendah berbeda-beda untuk tiap model, sementara wilayah Papua konsisten memberikan performa tertinggi. Hasil evaluasi ini mengindikasikan bahwa pelafalan yang lebih konsisten, jelas, dan mudah dikenali, seperti pada data Papua, membuat model lebih mudah mempelajari pola suara, sedangkan variasi logat dan pelafalan huruf yang berbeda-beda mempunyai potensi menjadi penyebab penurunan akurasi model.

Secara keseluruhan, hasil dari lima eksperimen ini menunjukkan bahwa model *Recurrent Neural Network* (RNN-GRU) merupakan arsitektur paling unggul dan stabil dalam klasifikasi fonik. Selain itu, PNCC mengungguli MFCC Librosa dalam ekstraksi fitur karena mampu memberikan fitur audio yang mendukung performa model. Teknik data augmentasi yang digunakan, yaitu *speed perturbation* dan *pitch shifting*, berkontribusi dalam memperkaya variasi *dataset* yang membantu dalam mencapai hasil evaluasi yang baik. Hasil eksperimen juga membuktikan bahwa asal daerah *speaker* mempengaruhi hasil akurasi model, di mana Papua lebih mudah dikenali dibandingkan wilayah lainnya. Selain itu, hasil klasifikasi dari data *native speaker* mengungguli data *non-native speaker*. Hal ini menunjukkan bahwa aksen yang beragam, kejelasan dalam pengucapan, dan kualitas rekaman berperan dalam hasil klasifikasi model.

Perbandingan dengan penelitian terdahulu oleh Razak (2024), terdapat beberapa perbedaan signifikan baik dari sisi metodologi maupun hasil. Penelitian sebelumnya mendapatkan hasil tertinggi menggunakan SVM dengan ekstraksi fitur MFCC dengan *library* Librosa dan menerapkan teknik data augmentasi berupa penambahan *noise* dan *time shifting*. Meskipun berhasil mencapai performa yang baik (Akurasi: 0.9195 dan F1 Score: 0.9242), penelitian kali ini menunjukkan peningkatan performa secara konsisten menggunakan model

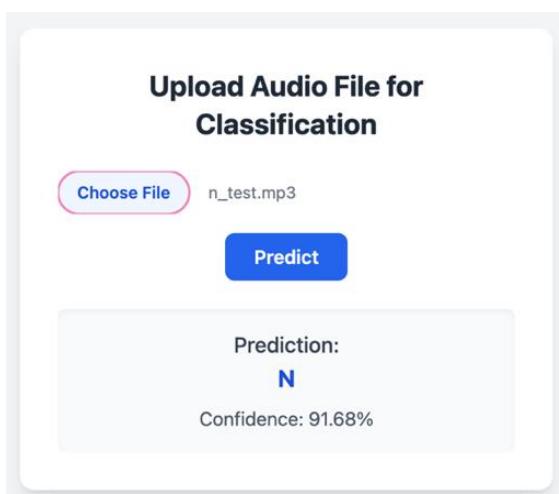
RNN-GRU dan ekstraksi fitur PNCC. Selain itu, penelitian ini memperluas cakupan analisis dengan mengevaluasi pengaruh logat daerah penutur *non-native* terhadap akurasi klasifikasi. Pendekatan ini belum dilakukan pada penelitian sebelumnya dan terbukti memberikan informasi tambahan mengenai seberapa besar aksen dan latar belakang *regional* memengaruhi hasil prediksi.

4.8 Hasil Antarmuka Website Pengujian

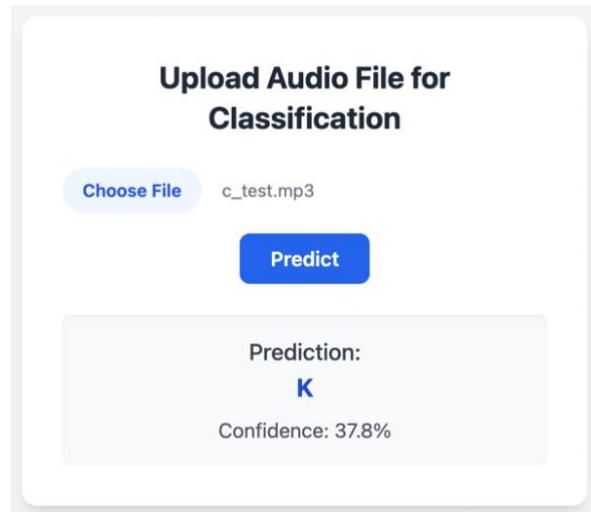
Subbab ini membahas hasil pengujian terhadap antarmuka yang dikembangkan sebagai media pengujian pelafalan huruf fonik secara langsung oleh pengguna. Antarmuka ini dibangun berdasarkan model terbaik yang diperoleh dari hasil eksperimen sebelumnya, yaitu model RNN-GRU dengan ekstraksi fitur PNCC. Model tersebut dipilih karena menunjukkan performa paling stabil dan akurat pada pengujian di seluruh eksperimen.

Antarmuka dikembangkan menggunakan arsitektur berbasis web. Bagian *backend* menggunakan *framework Flask (Python)* yang berperan dalam menangani proses *upload* audio, ekstraksi fitur, pemanggilan model, serta pengiriman hasil prediksi ke tampilan web. Sementara itu, bagian *frontend* menggunakan *HTML* dan *TailwindCSS*, yang dirancang agar antarmuka tetap sederhana, responsif, dan mudah digunakan. Pengguna dapat mengunggah file audio dalam format MP3 yang kemudian diproses secara otomatis oleh sistem. Proses meliputi pemangkasan suara (*trimming*), ekstraksi fitur PNCC, normalisasi, hingga prediksi huruf menggunakan model RNN yang telah dilatih sebelumnya. Lalu, hasil prediksi ditampilkan pada halaman web dalam bentuk huruf yang diprediksi dan tingkat kepercayaan model terhadap prediksi tersebut (*confidence score*).

Pengujian dilakukan terhadap dua huruf fonik, yaitu n dan c, yang masing-masing direkam secara terpisah dan diunggah melalui antarmuka. Berdasarkan hasil yang ditunjukkan pada Gambar 4.13, sistem berhasil memprediksi huruf n dengan benar. Namun, pada pelafalan huruf c yang ditunjukkan pada Gambar 4.14, sistem memberikan hasil prediksi sebagai huruf k. Kesalahan ini dapat terjadi karena kemiripan pola akustik antara huruf c dan k sehingga menghasilkan pola spektral yang mirip. Ketika ada huruf fonik yang memiliki karakteristik yang berdekatan, hal ini menjadi salah satu permasalahan dalam klasifikasi suara karena model dapat mengklasifikasi huruf dengan salah.



Gambar 4.13 Hasil Pengujian Huruf N



Gambar 4.14 Hasil Pengujian Huruf C

BAB 5 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran yang disusun berdasarkan hasil eksperimen yang telah dilakukan pada Tugas Akhir ini.

5.1. Kesimpulan

Bagian ini menjelaskan mengenai kesimpulan yang dapat ditarik dari hasil penelitian yang telah dilakukan. Beberapa poin penting dari rumusan masalah terjawab sebagai berikut:

1. Pengembangan model *deep learning* untuk pengenalan fonik dilakukan melalui tiga tahap utama, yaitu augmentasi data, ekstraksi fitur, dan klasifikasi. Augmentasi dilakukan menggunakan metode *pitch shifting* dan *speed perturbation* untuk meningkatkan variasi data. Ekstraksi fitur dilakukan dengan dua metode, yaitu MFCC dan PNCC, di mana PNCC menghasilkan performa yang lebih baik. Tahap klasifikasi menggunakan tiga arsitektur model, yaitu CNN, RNN-GRU, dan *Transformer*. Dari seluruh eksperimen, model RNN-GRU dengan fitur PNCC memberikan hasil terbaik dengan akurasi sebesar 94,59% dan *F1 Score* sebesar 0,946.
2. Metode ekstraksi fitur PNCC menghasilkan performa yang lebih unggul dibandingkan MFCC *Librosa*. Pada model RNN, *F1 Score* meningkat dari 0.9289 (MFCC, Eksperimen A) menjadi 0.946 (PNCC, Eksperimen B). Peningkatan juga terjadi pada CNN dan *Transformer*, menunjukkan bahwa PNCC memberikan representasi fitur yang lebih relevan untuk pengenalan fonik.
3. Model yang dilatih menggunakan data dari penutur *native* menunjukkan hasil evaluasi yang lebih tinggi dibandingkan *non-native*. Misalnya, pada model RNN, *F1 Score* meningkat dari 0.9305 menjadi 0.9385, dan akurasi dari 0.9293 menjadi 0.9392. Konsistensi pelafalan pada data *native* membantu model mengenali fonik dengan lebih akurat, sedangkan variasi logat pada data *non-native* menurunkan performa klasifikasi.
4. Variasi dialek lokal Indonesia terbukti memengaruhi akurasi model dalam mengenali fonik. Ketiga model menunjukkan hasil terbaik pada data dari penutur Papua, dengan *F1 Score* mencapai 0.9902. Sebaliknya, performa terendah berbeda-beda tergantung model. CNN menunjukkan nilai terendah pada data dari Bali (*F1 Score*: 0.8102), dan *Transformer* pada data dari Madura (*F1 Score*: 0.8912). Hal ini menunjukkan bahwa logat yang lebih konsisten seperti Papua lebih mudah dikenali oleh model, sedangkan pelafalan yang lebih bervariasi cenderung menurunkan akurasi klasifikasi.

5.2. Saran

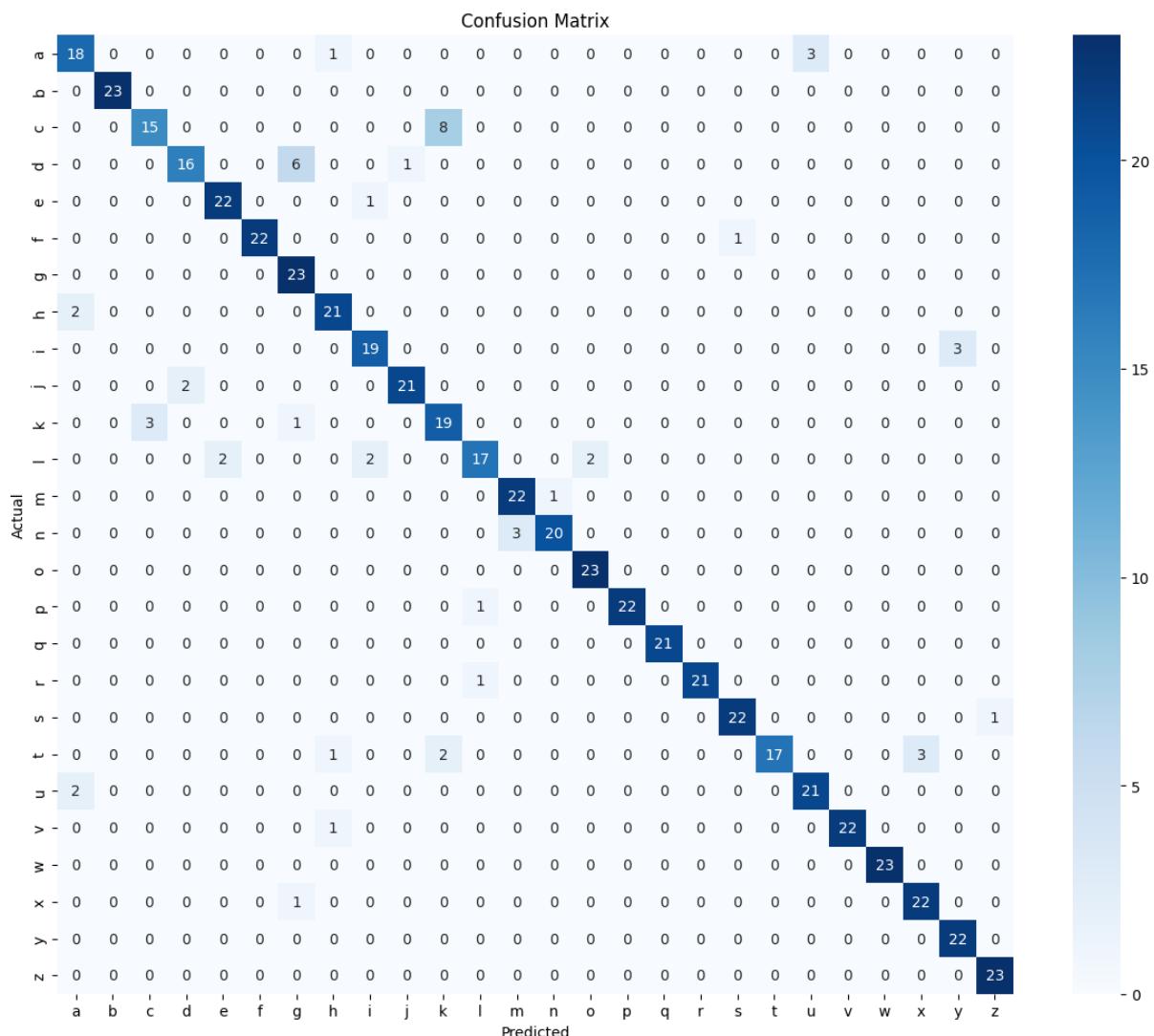
Beberapa saran yang dapat dapat diajukan sebagai pendukung pengembangan penelitian ke depannya, antara lain:

1. Perlu dilakukan perluasan pada variasi dialek lokal dalam *dataset*. Penelitian ini hanya mencakup lima daerah, yaitu Bali, Batam, Jawa, Madura, dan Papua. Disarankan untuk menambahkan dialek dari wilayah lain yang mempunyai ciri khas pelafalan berbeda.
2. Ditingkatkan kualitas rekaman untuk memastikan *dataset* yang digunakan bersih dari *noise* dan konsisten untuk tiap *speaker*.

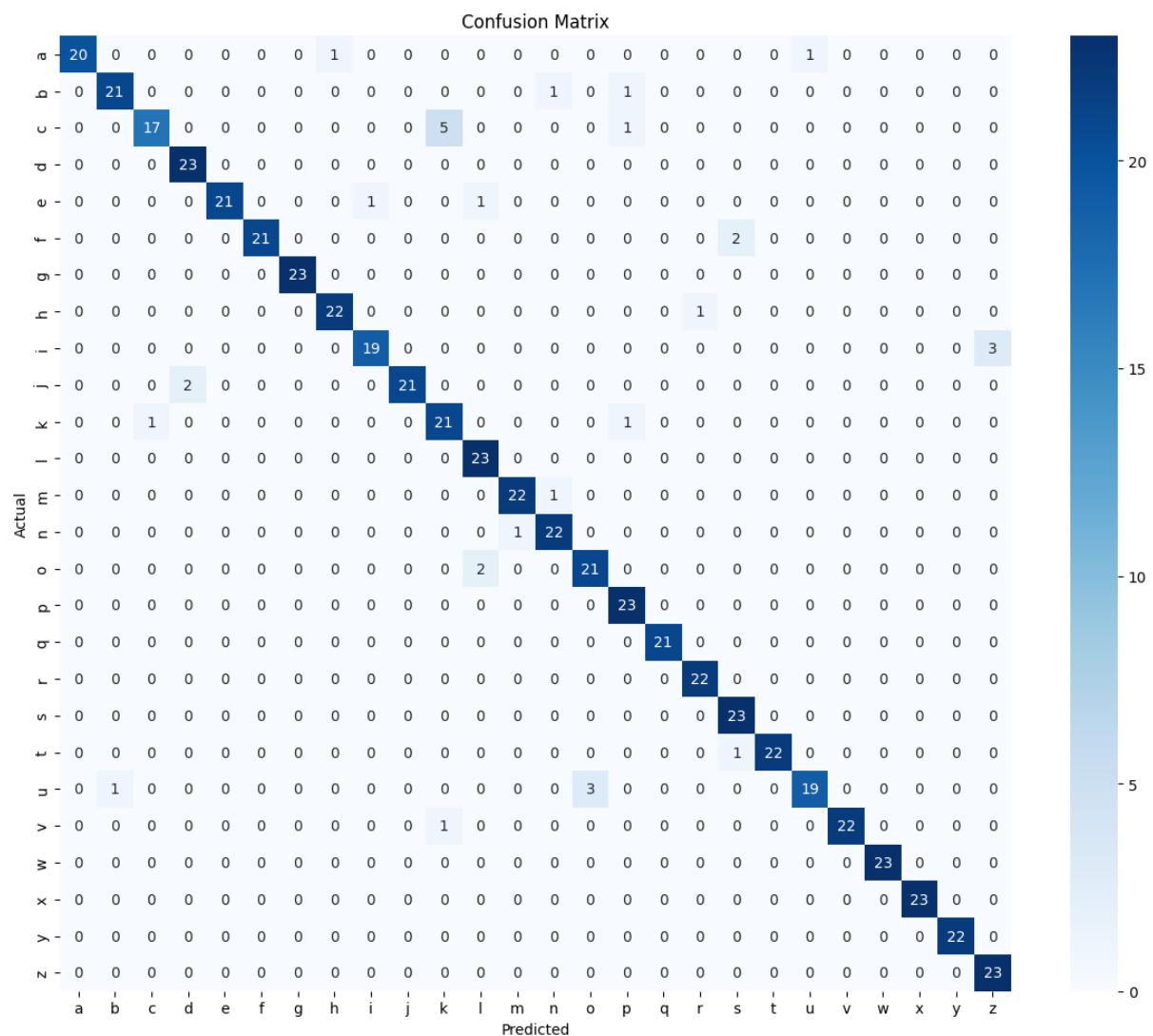
3. Penggunaan metode klasifikasi yang belum dieksplorasi dalam penelitian ini maupun penelitian sebelumnya.
4. Penerapan teknik augmentasi lain seperti *frequency masking*, *time warping*, atau kombinasi beberapa metode augmentasi secara bersamaan.

LAMPIRAN

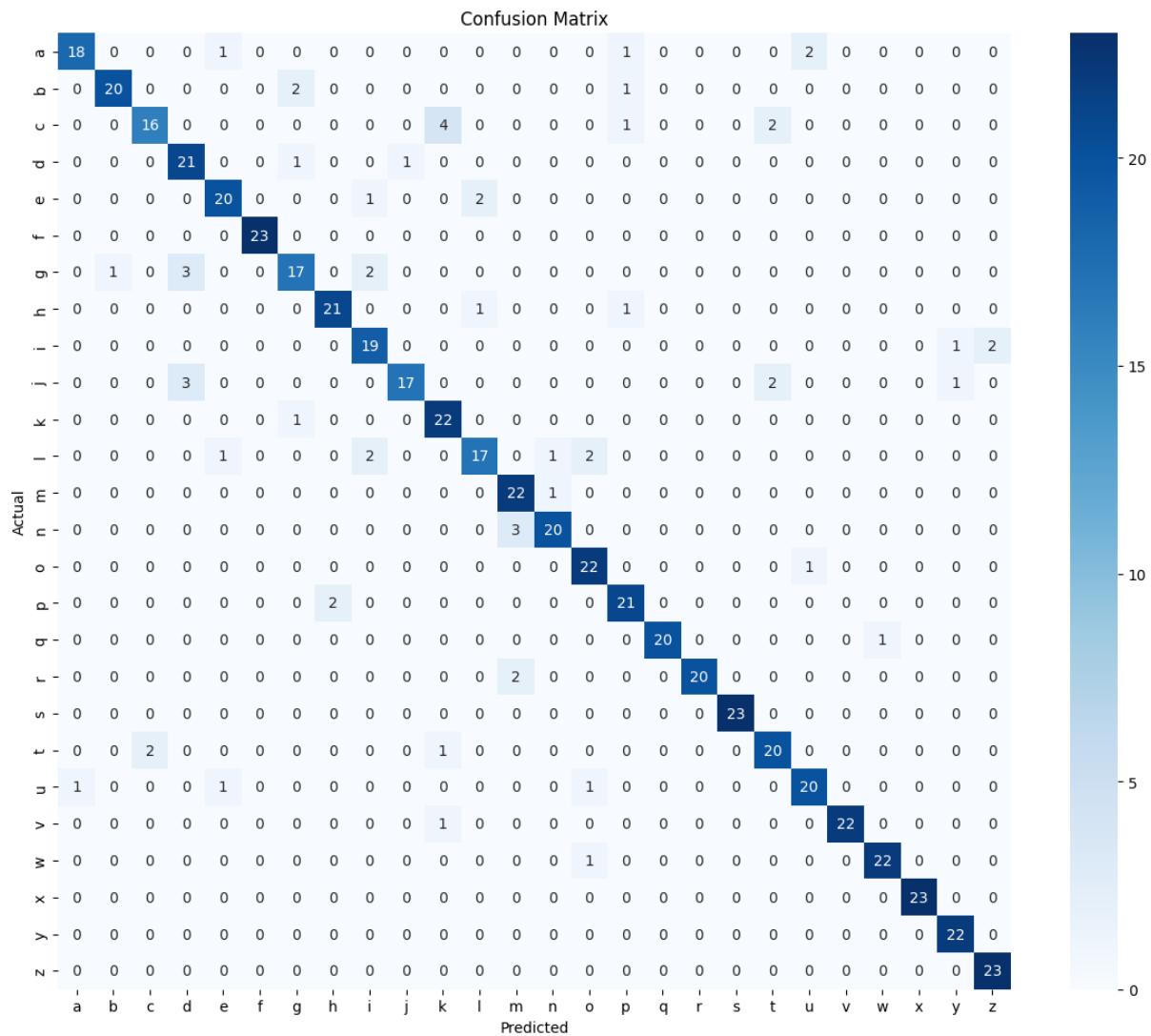
1. Visualisasi Confusion Matrix Eksperimen A
 - Convolutional Neural Network (CNN)



- Recurrent Neural Network (RNN)

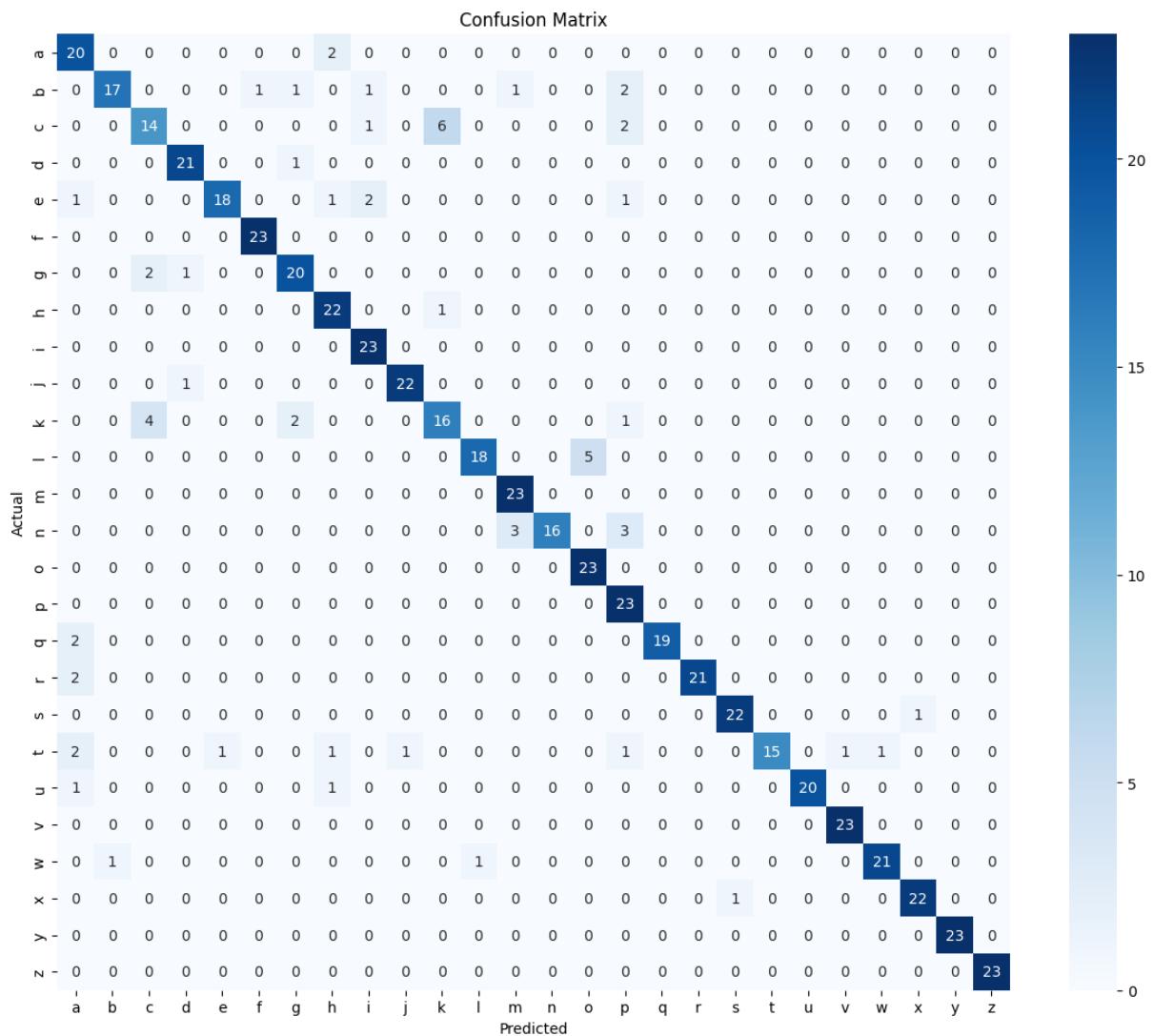


- Transformer

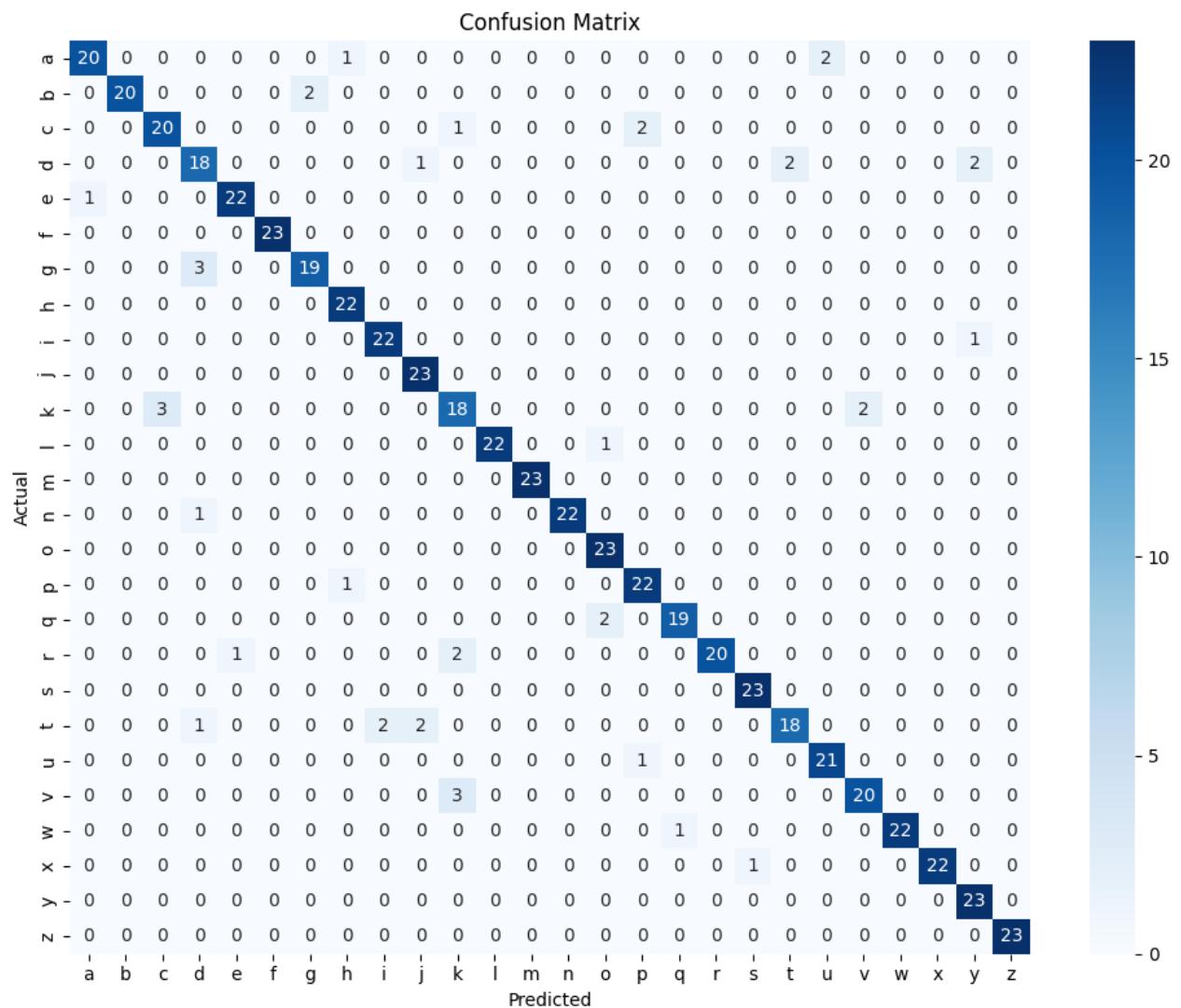


2. Visualisasi Confusion Matrix Eksperimen B

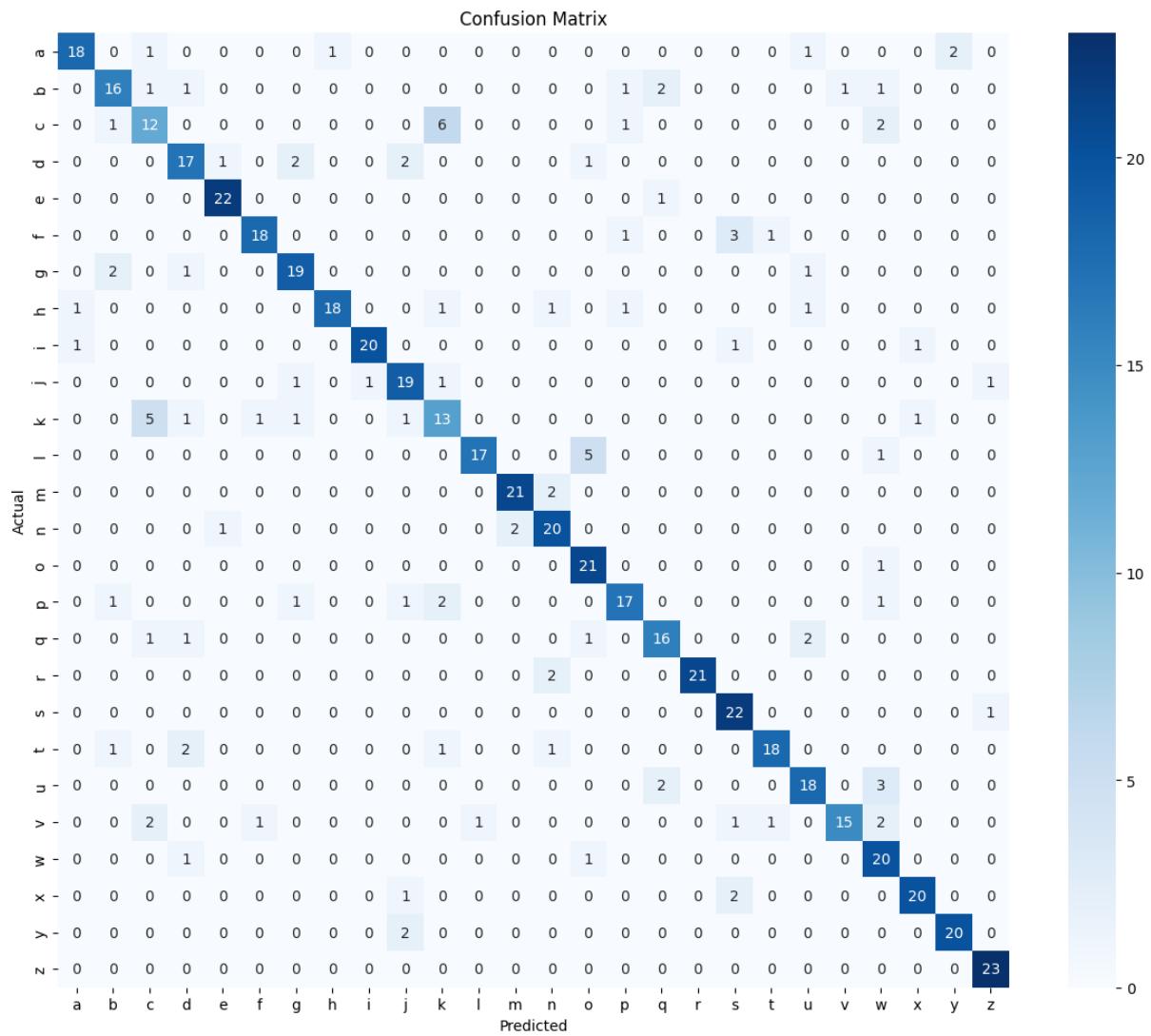
- Convolutional Neural Network (CNN)



- Recurrent Neural Network (RNN)

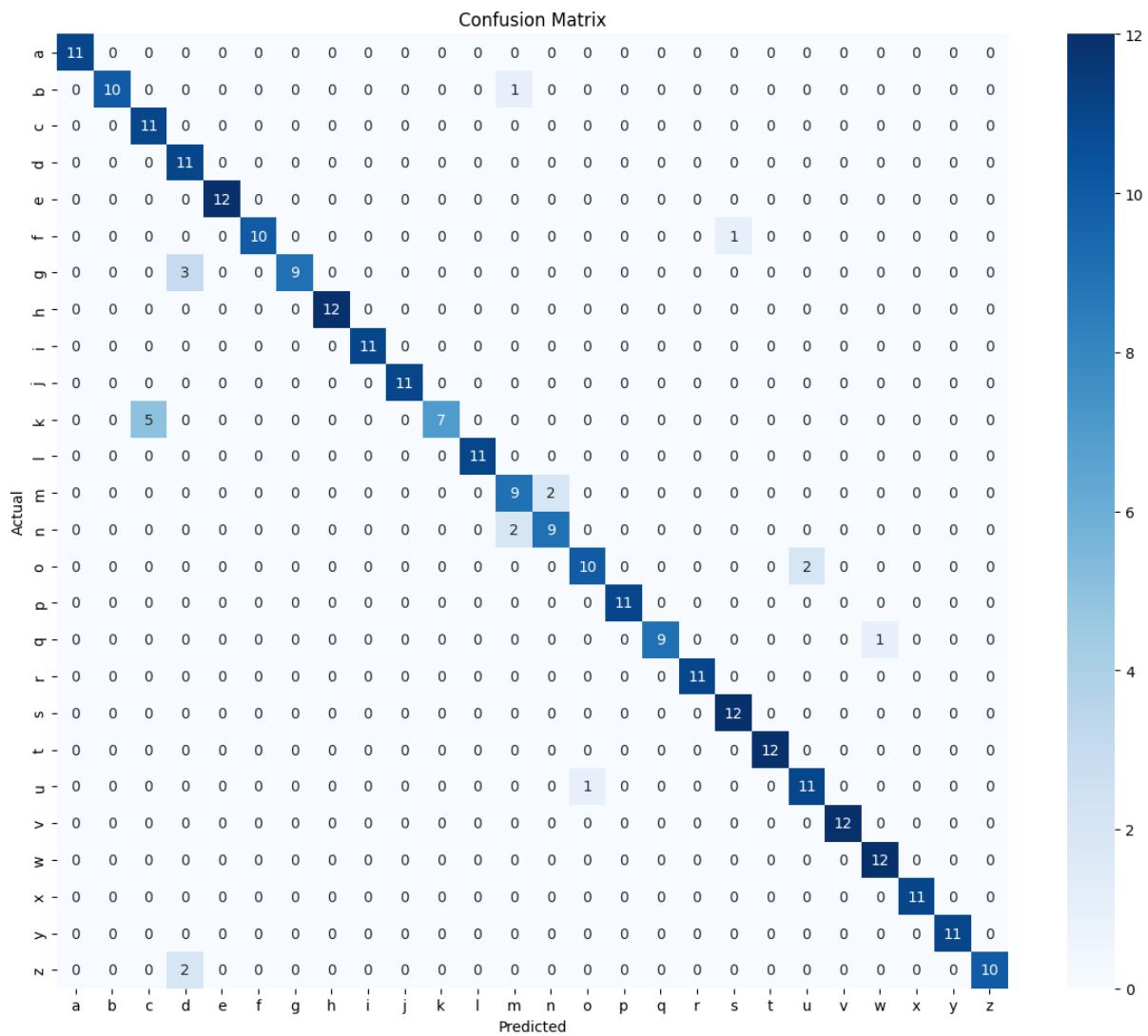


- Transformer

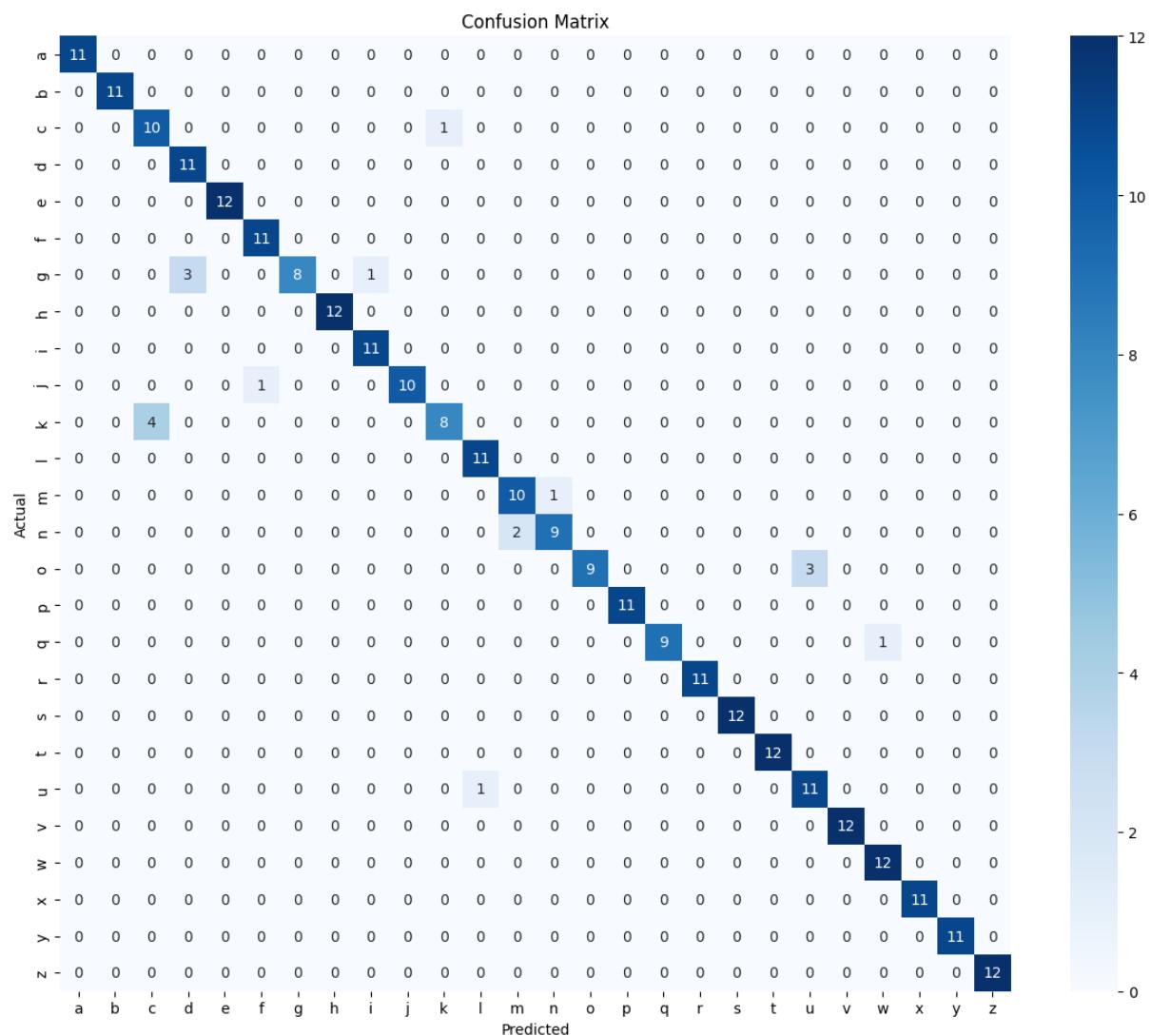


3. Visualisasi Confusion Matrix Eksperimen C

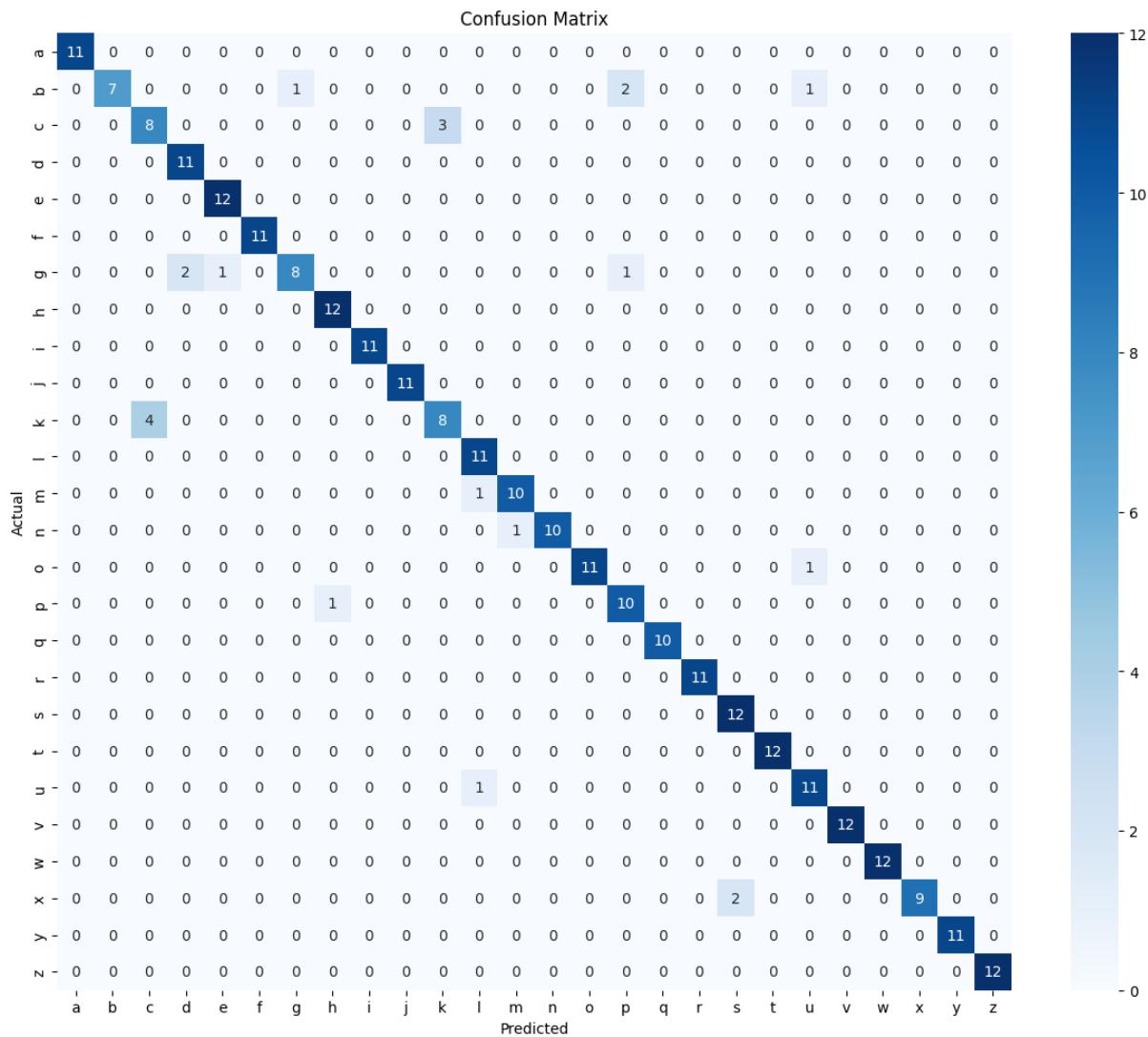
- Convolutional Neural Network (CNN)



- Recurrent Neural Network (RNN)

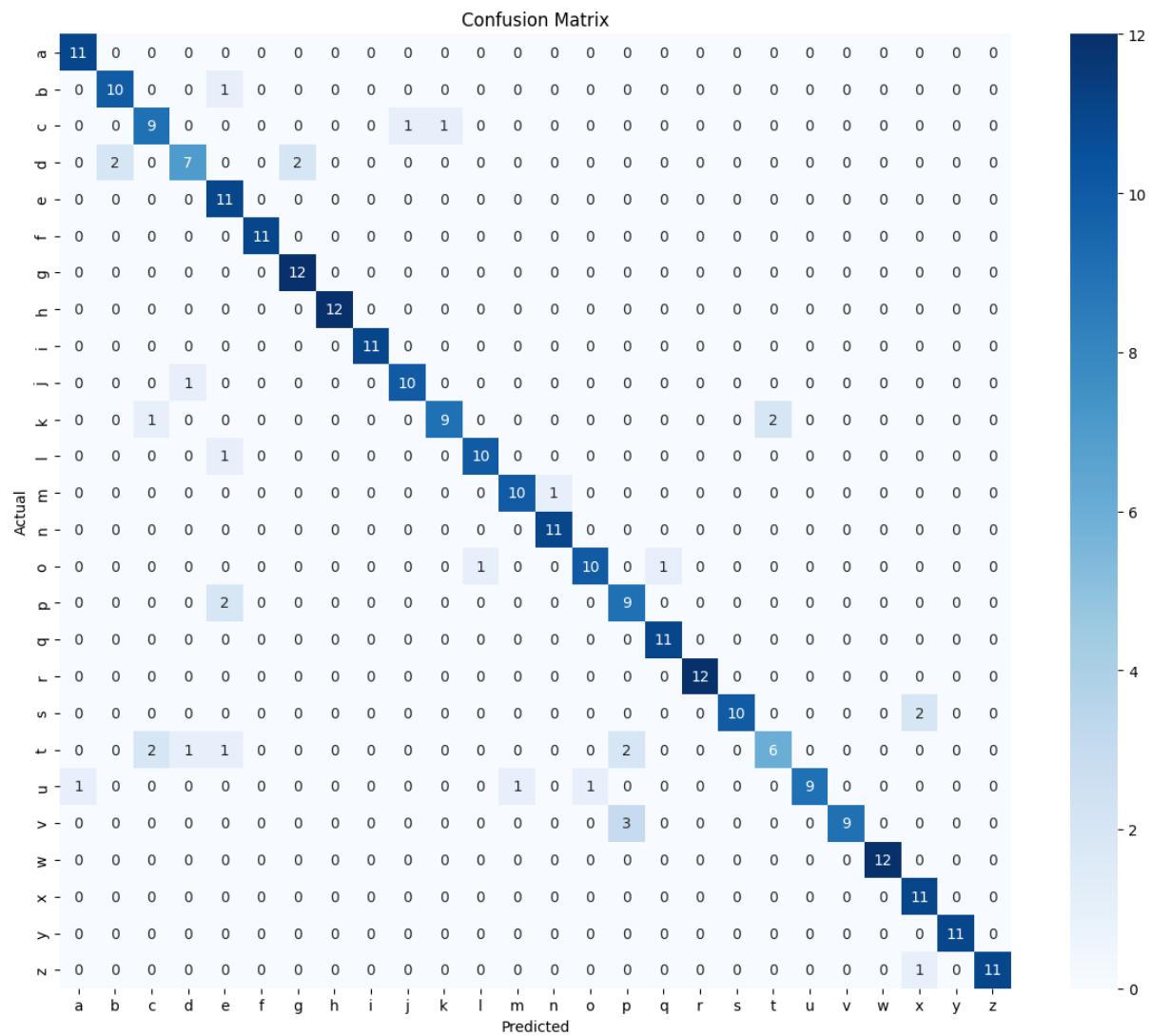


- Transformer

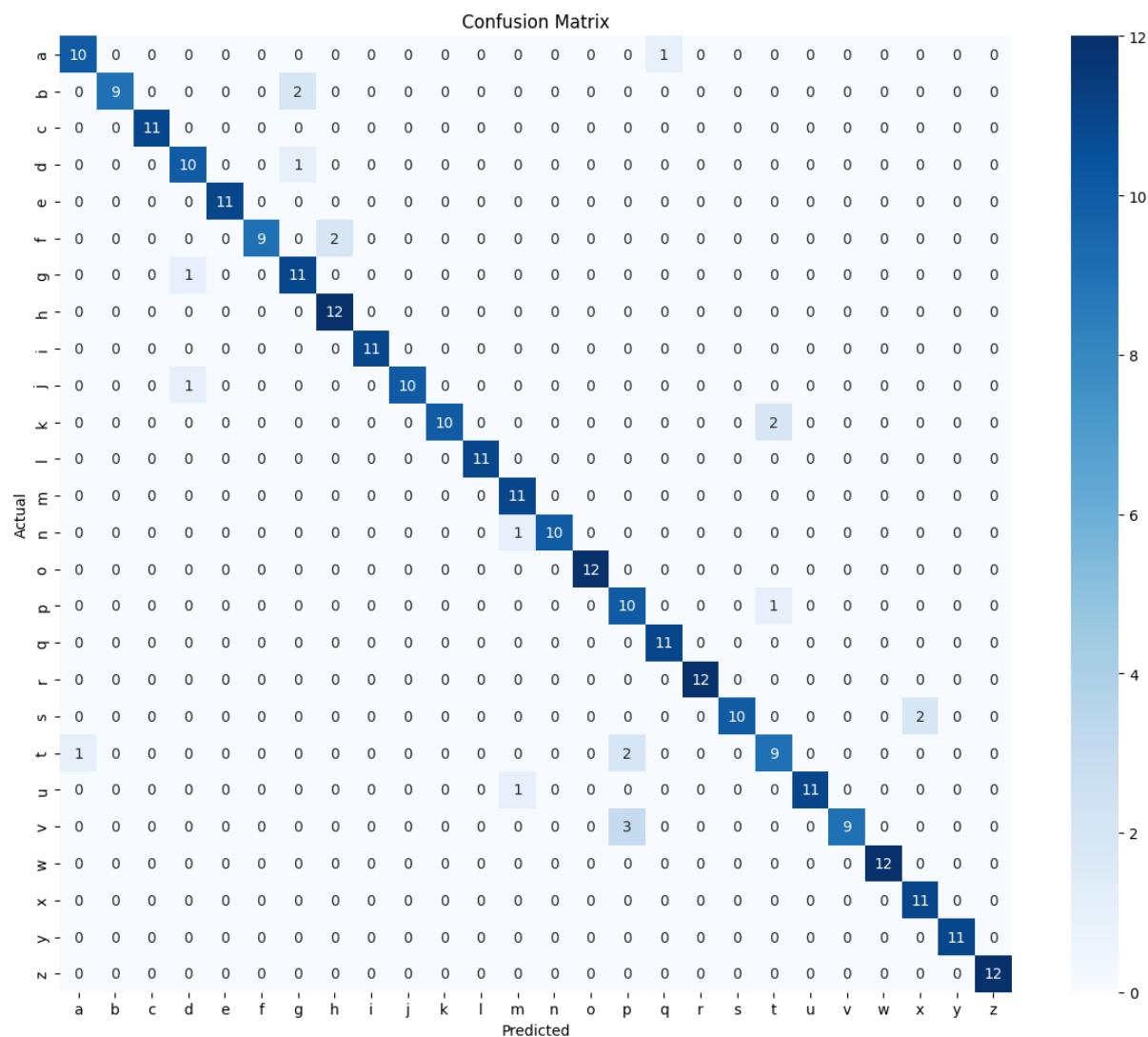


4. Visualisasi Confusion Matrix Eksperimen D

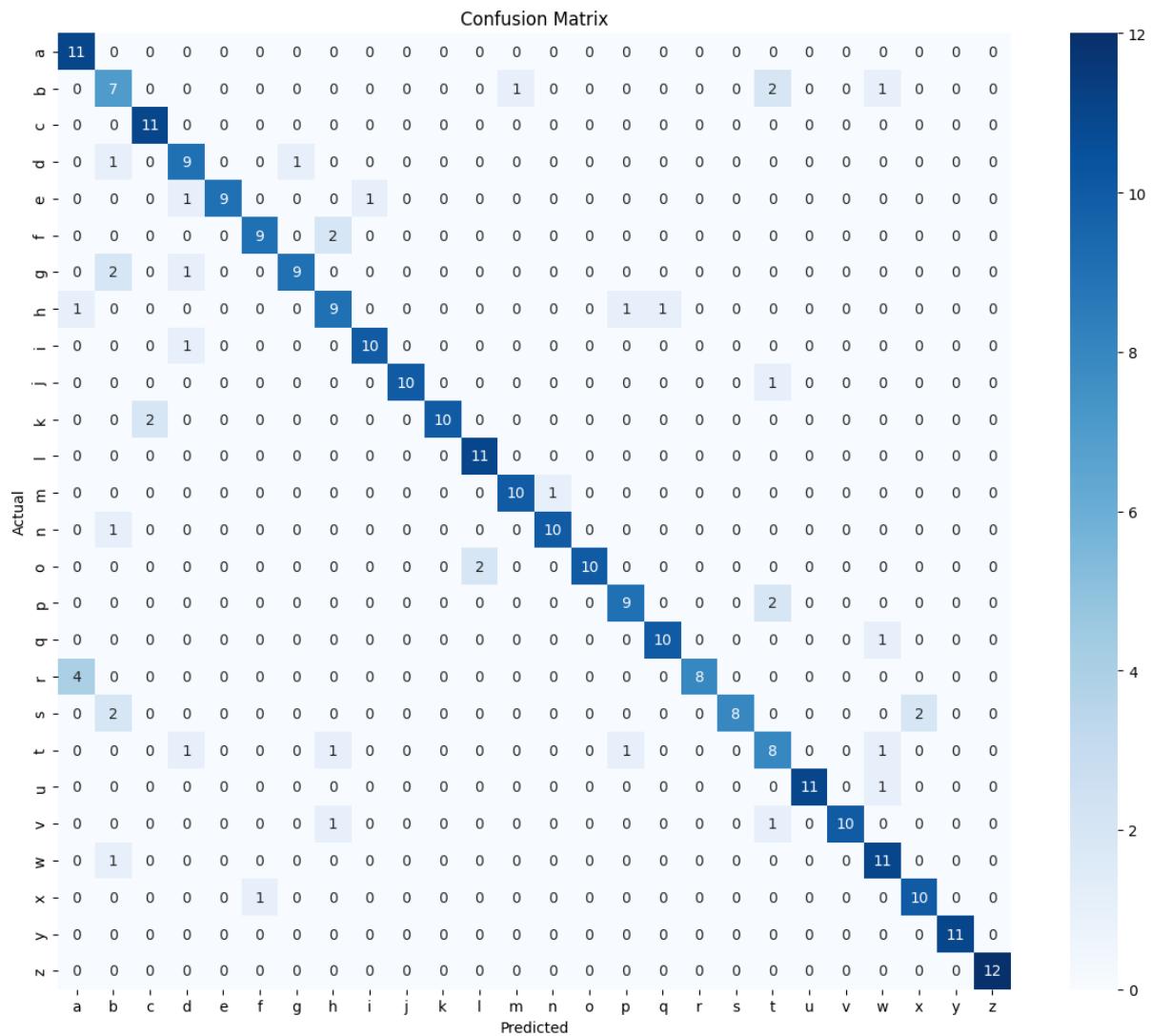
- Convolutional Neural Network (CNN)



- Recurrent Neural Network (RNN)

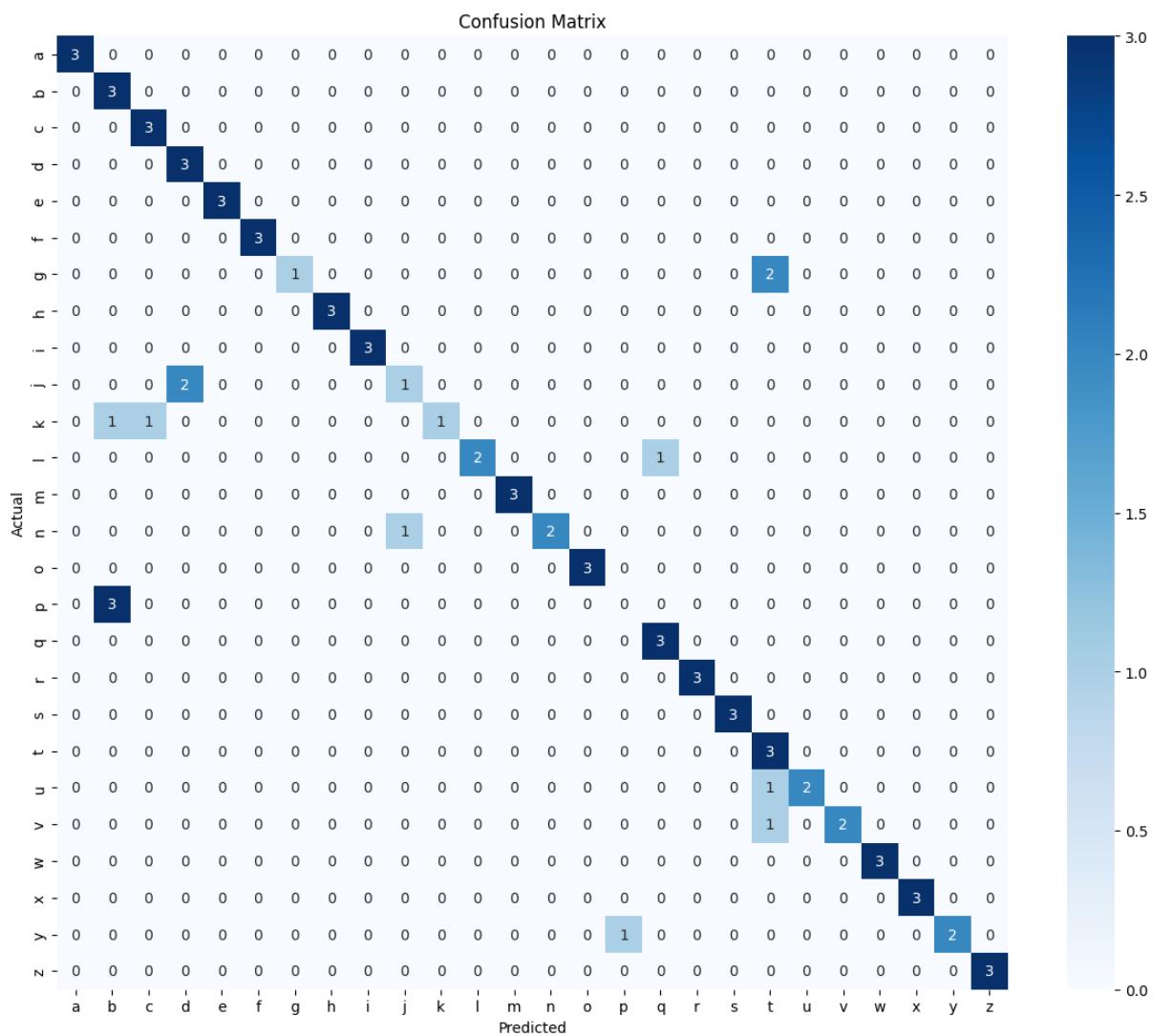


- Transformer

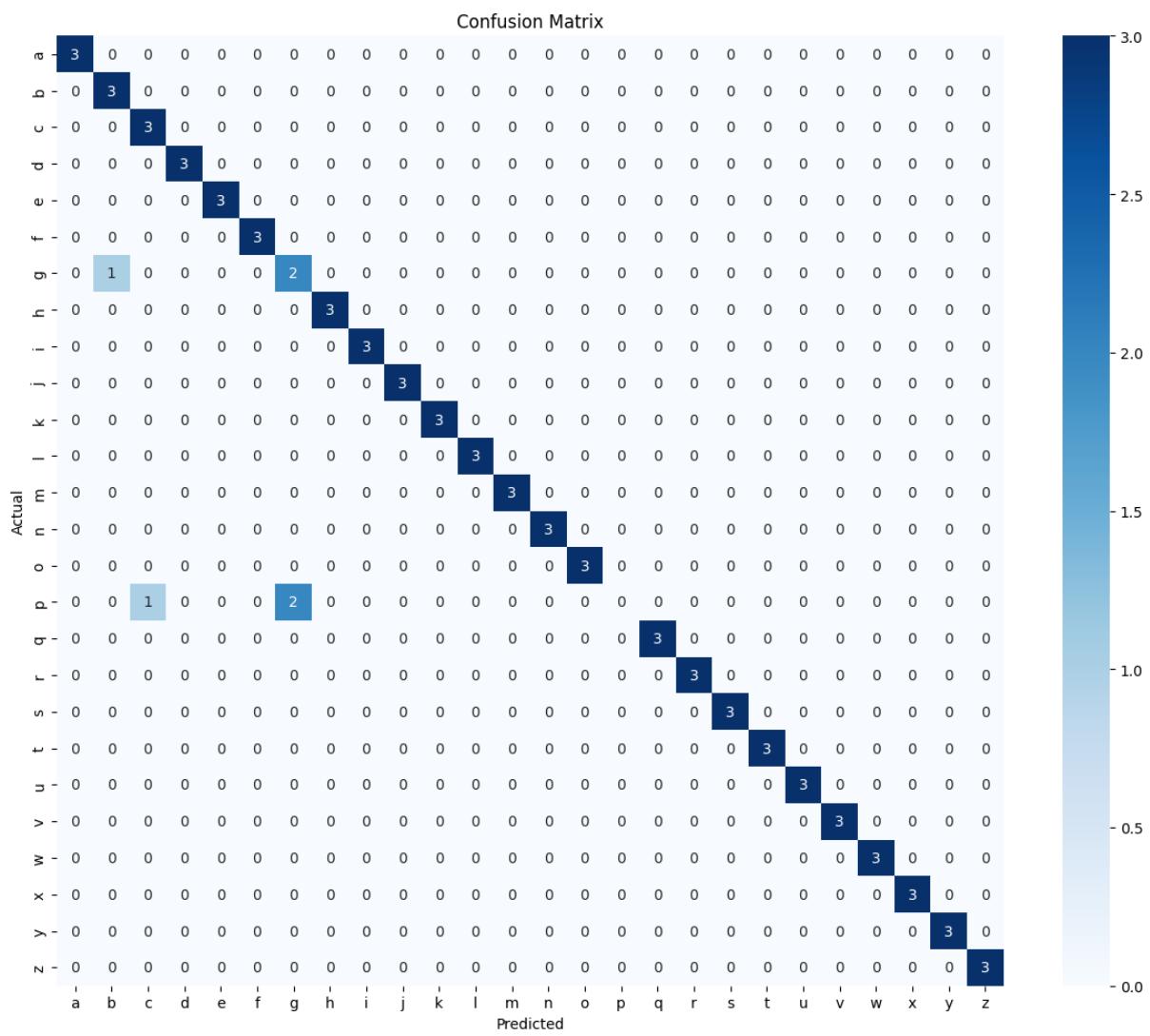


5. Visualisasi Confusion Matrix Eksperimen E

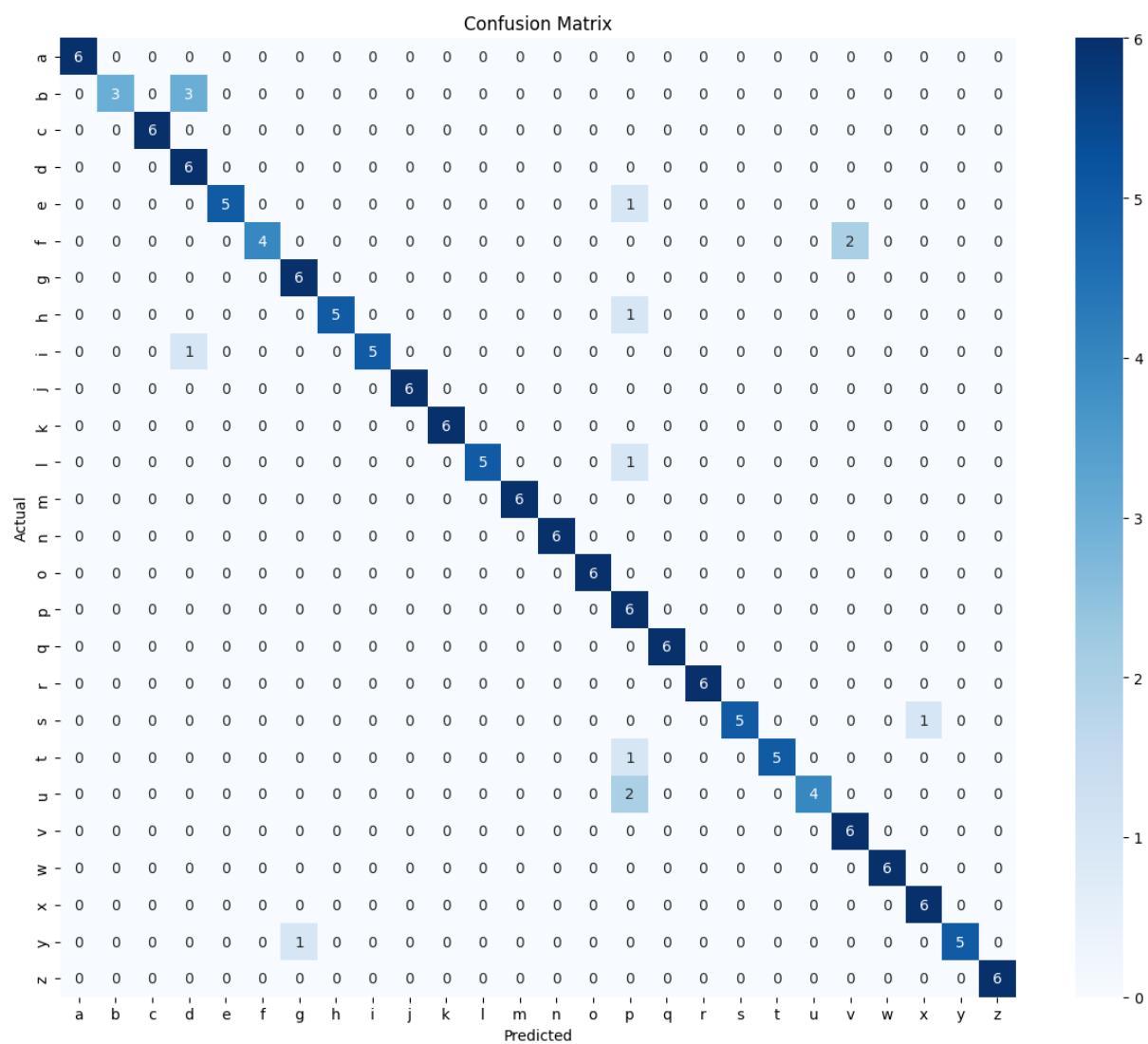
- Convolutional Neural Network (CNN)
- a. Bali



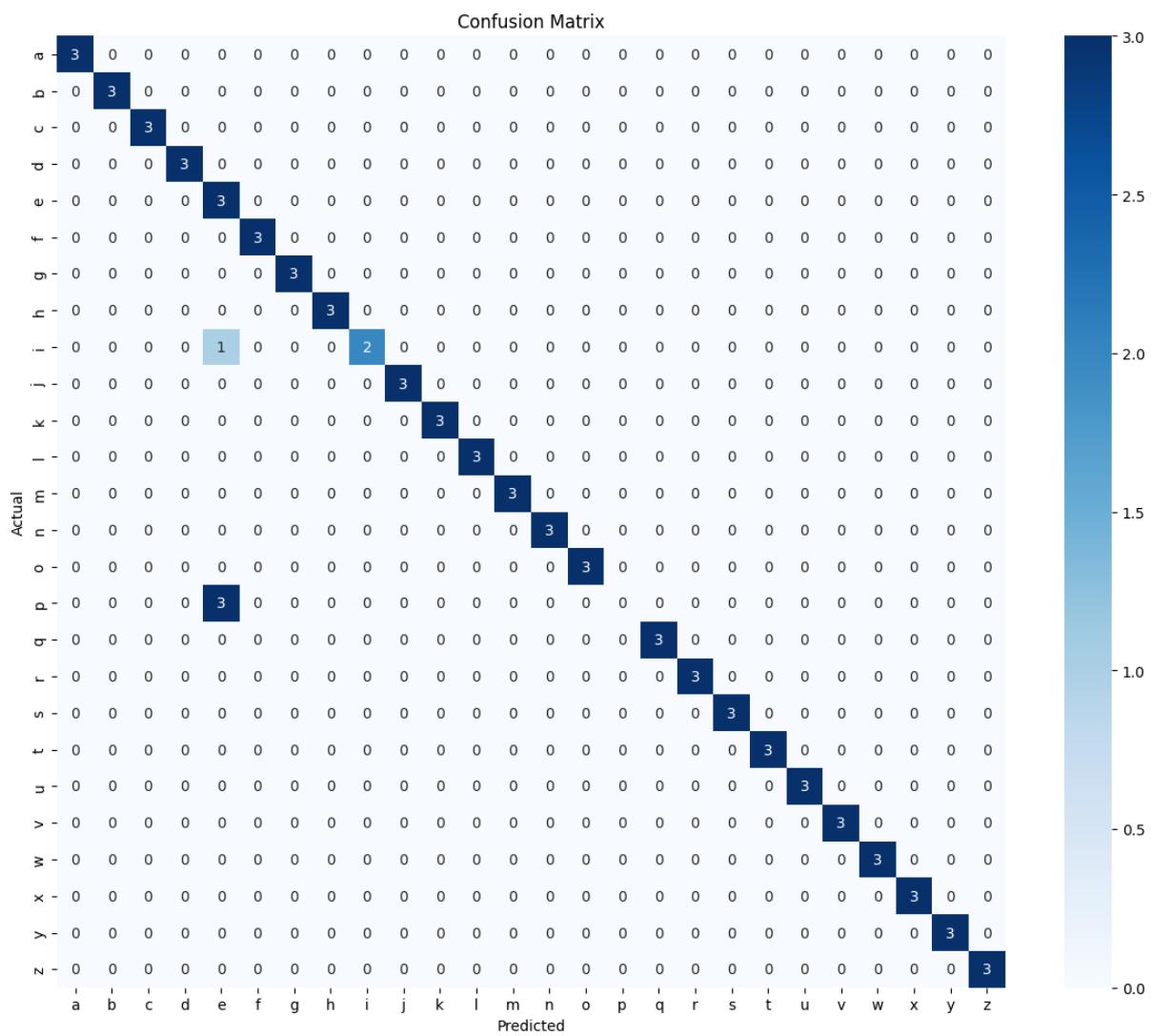
b. Batam



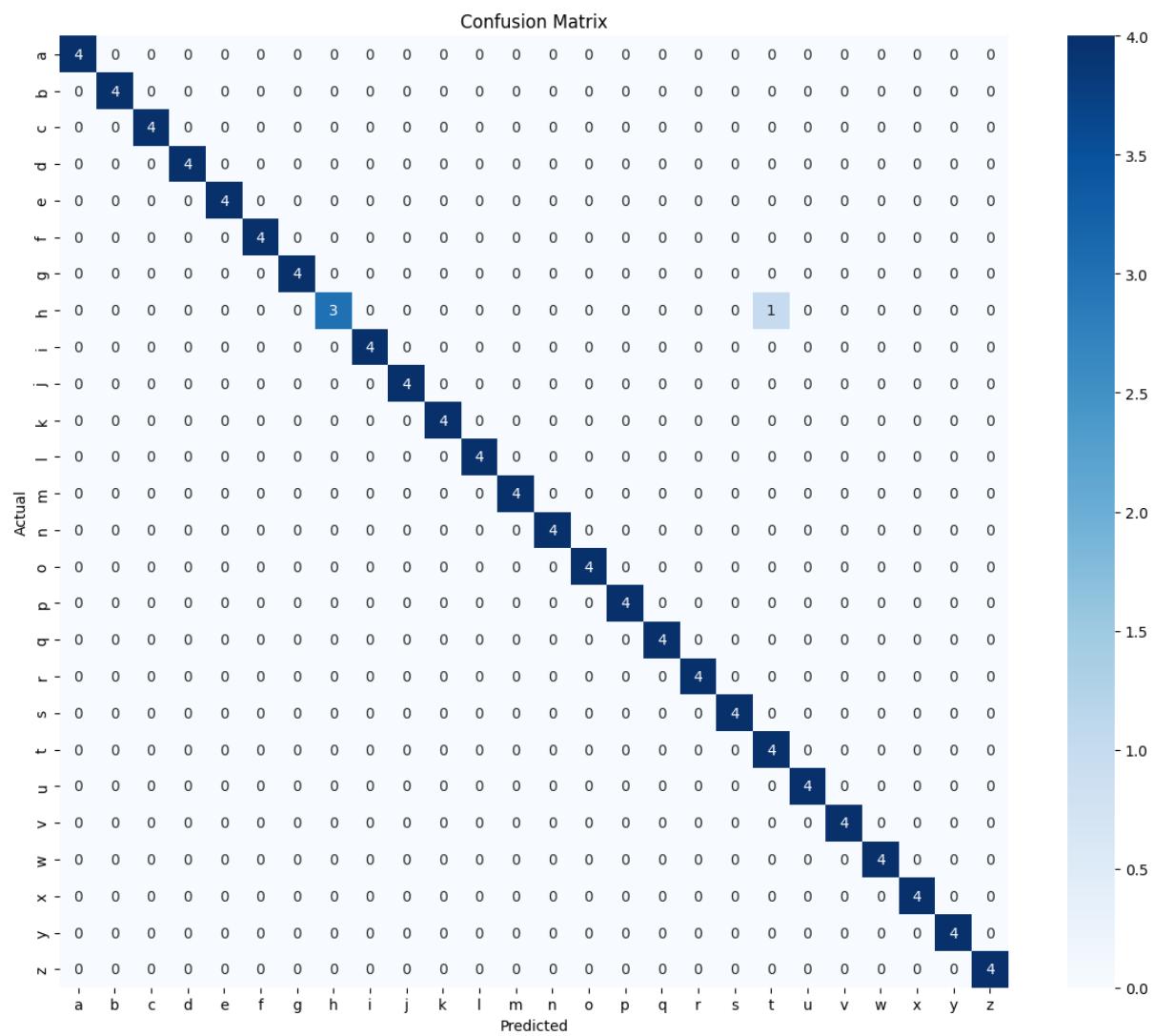
c. Jawa



d. Madura

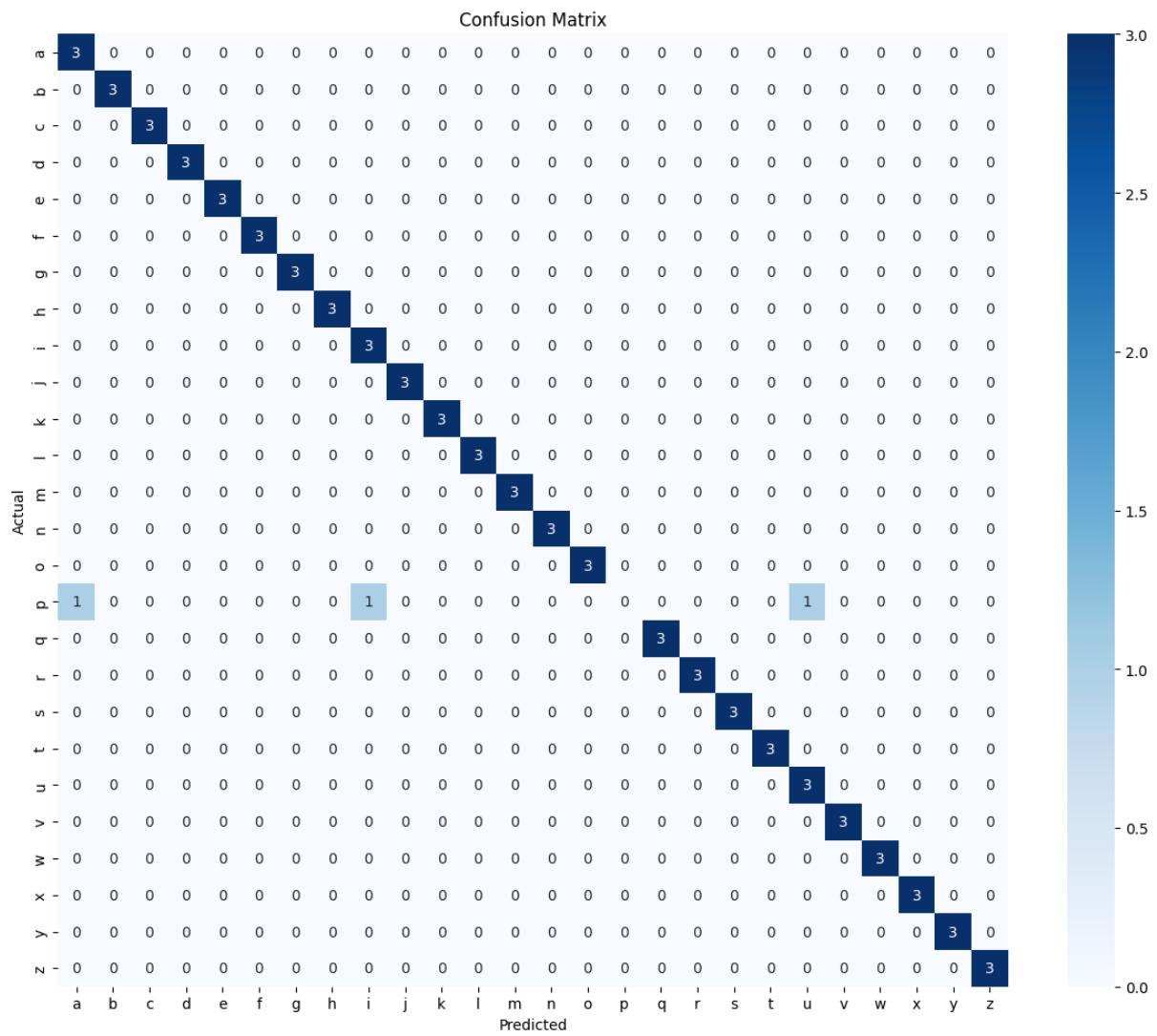


e. Papua

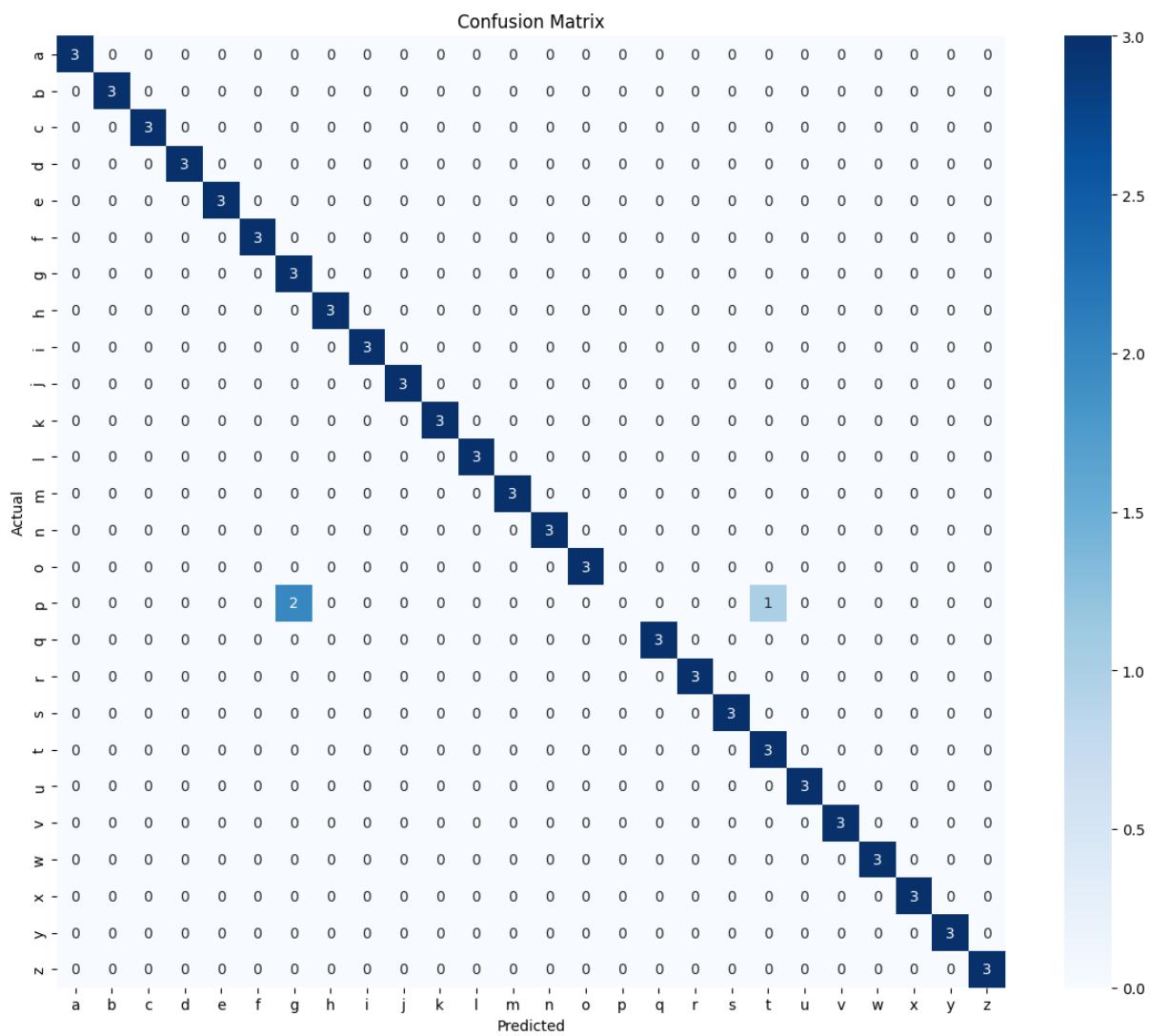


- Recurrent Neural Network (RNN)

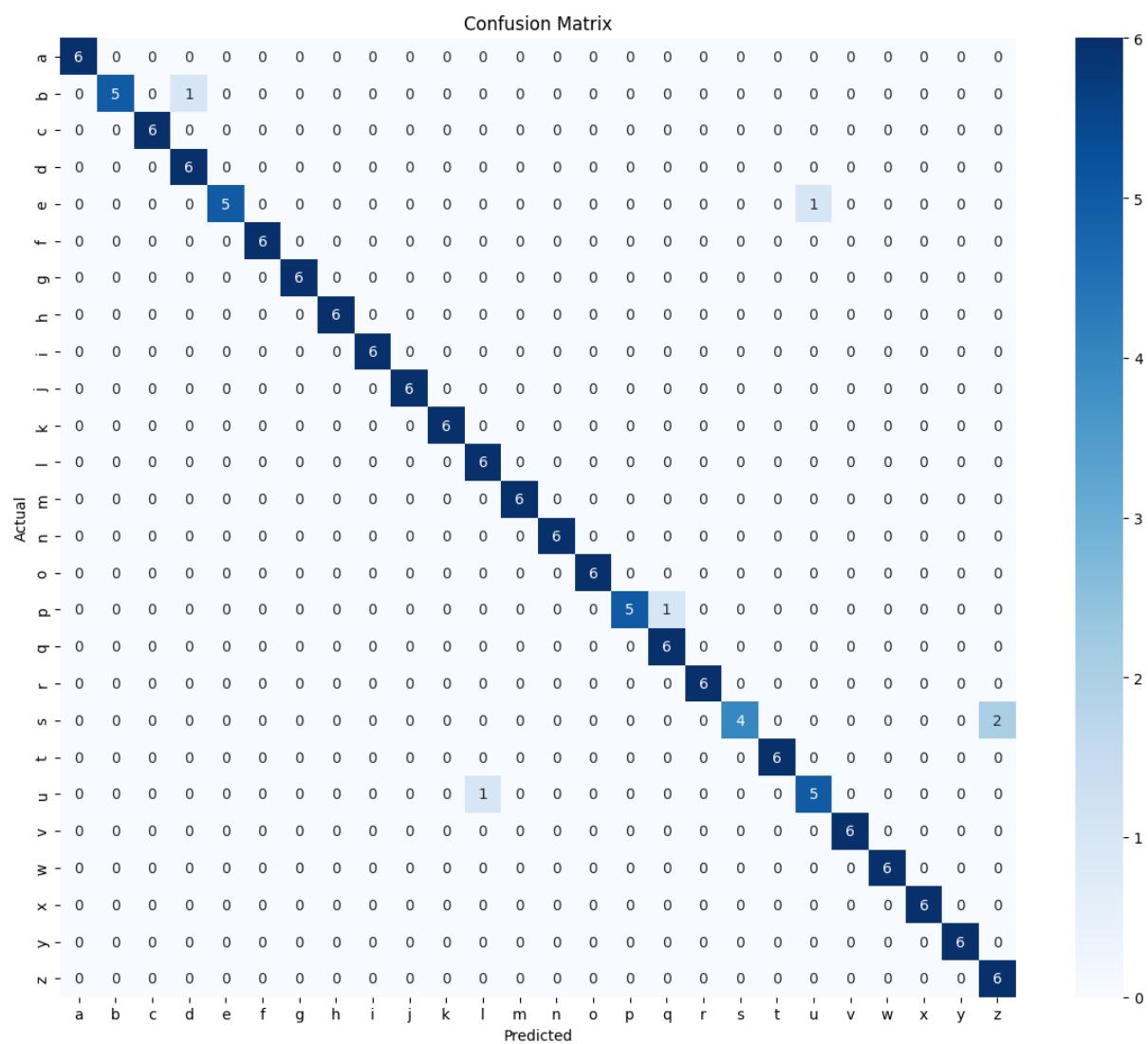
a. Bali



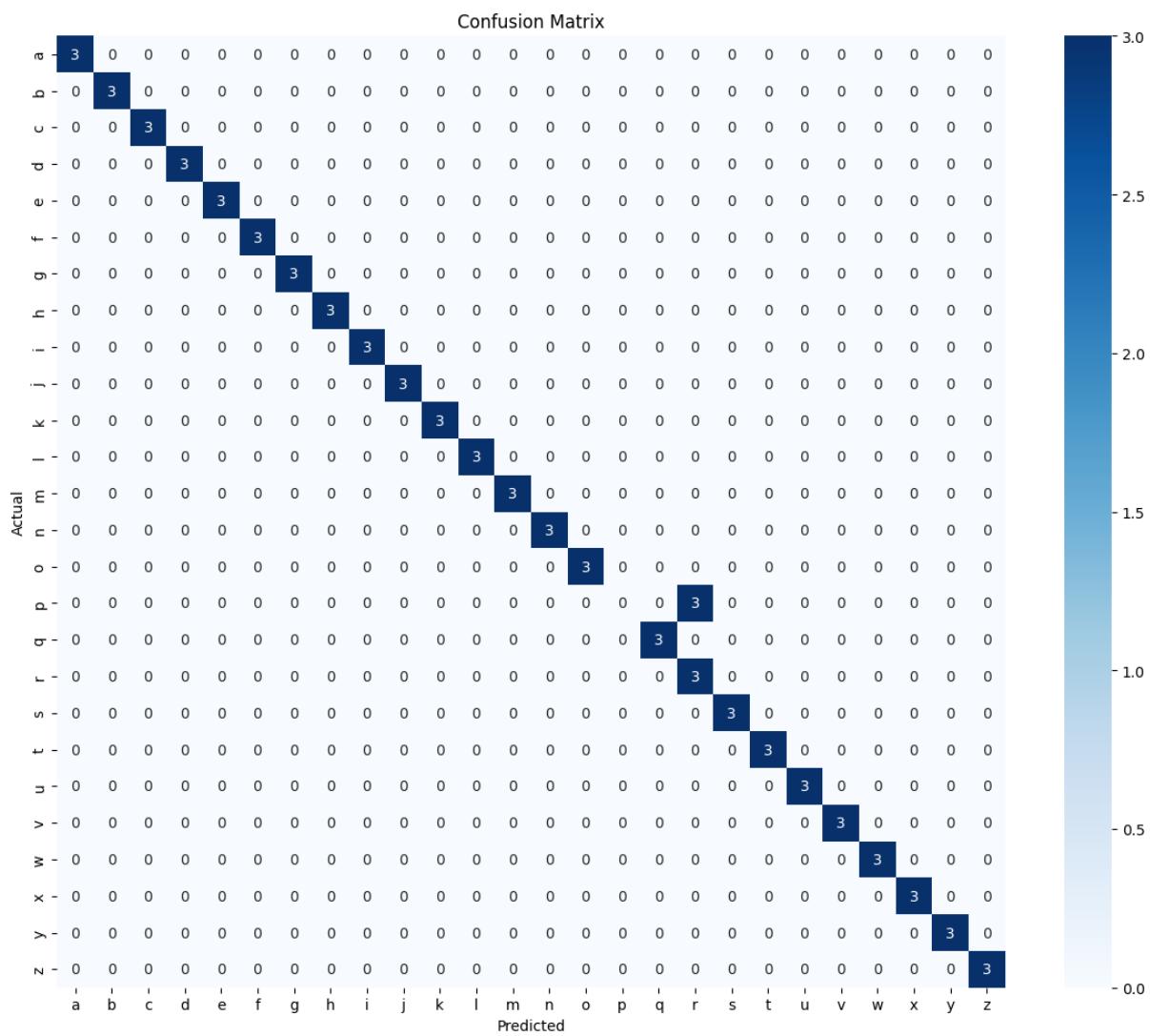
b. Batam



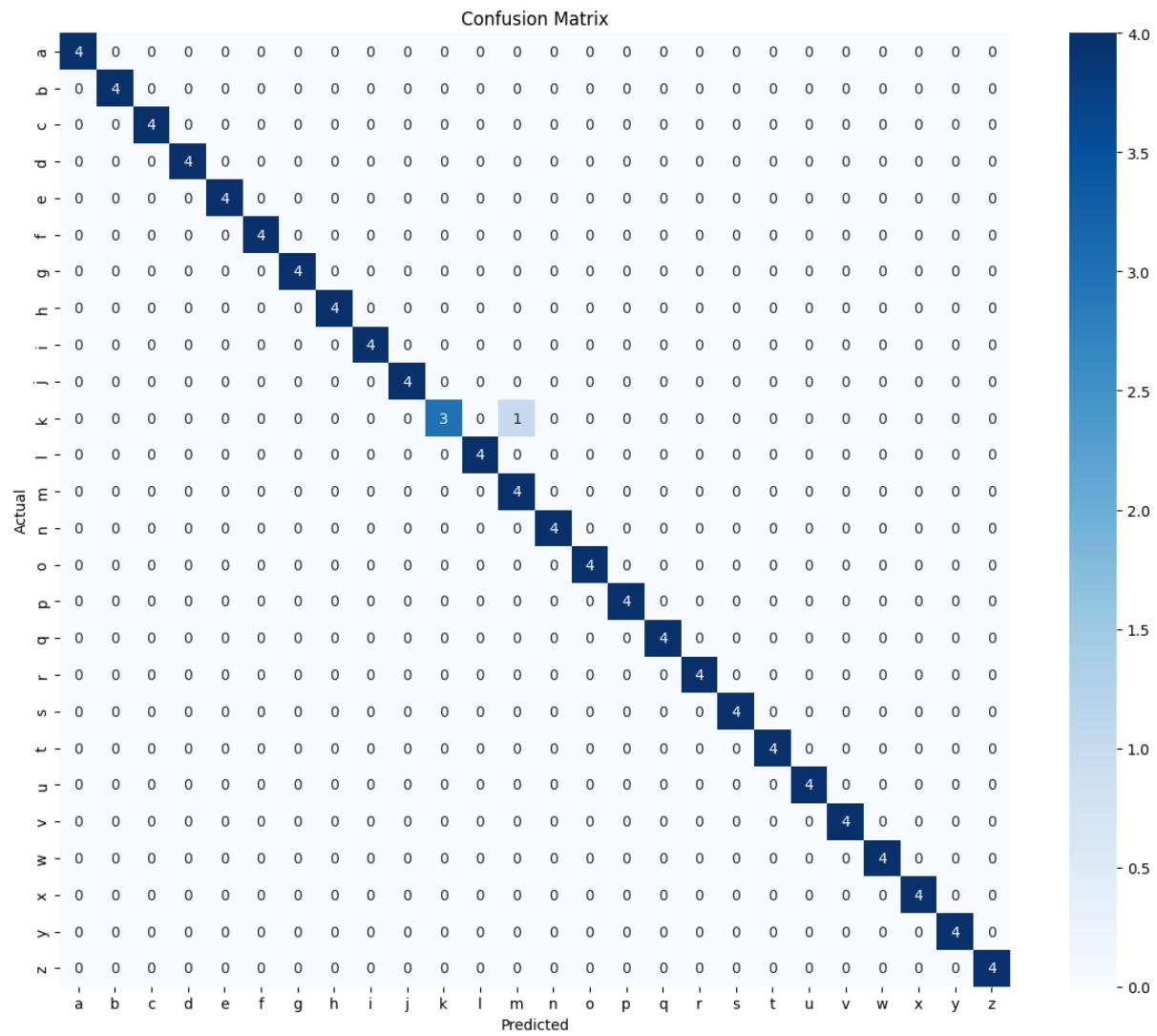
c. Jawa



d. Madura

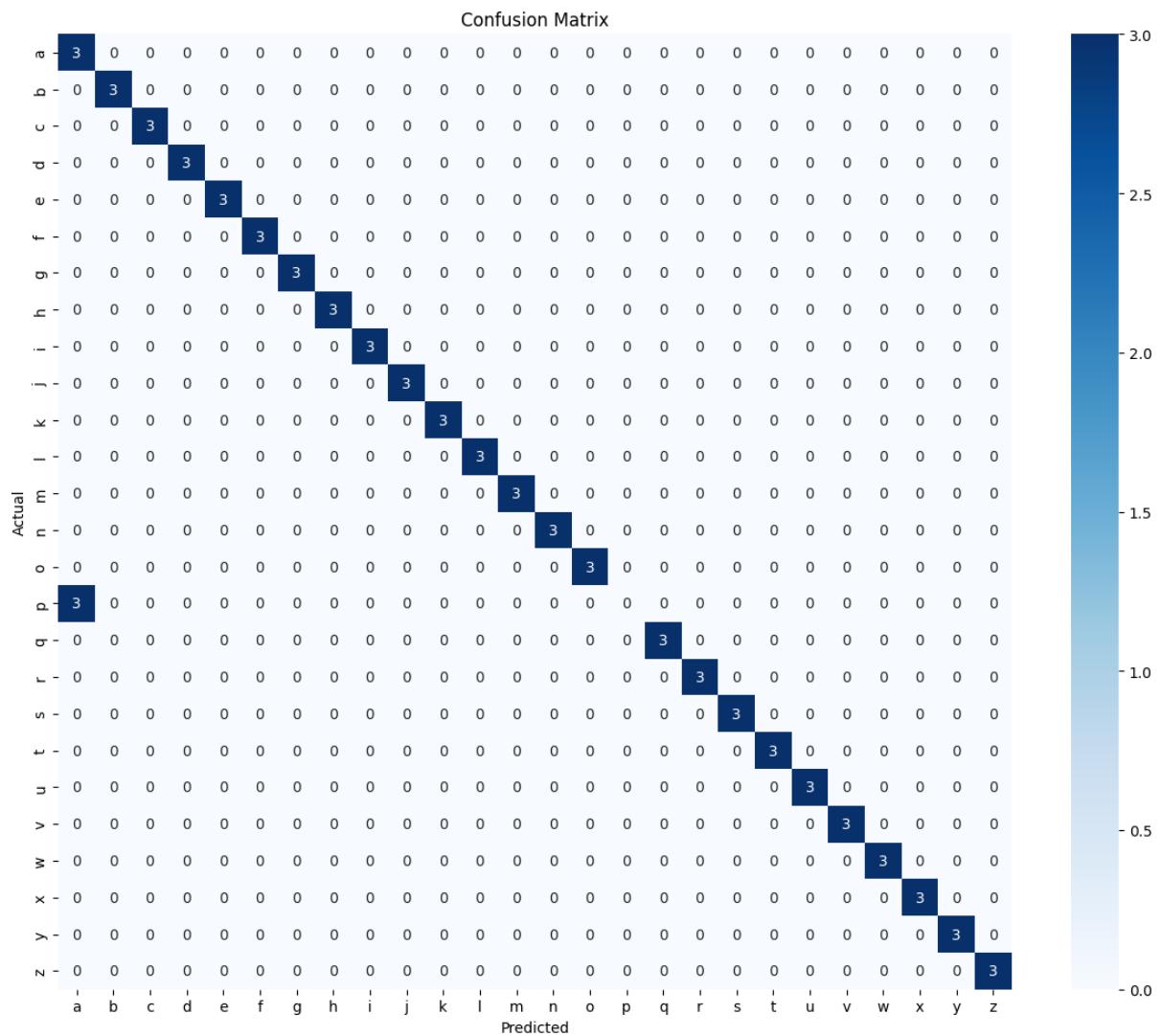


e. Papua

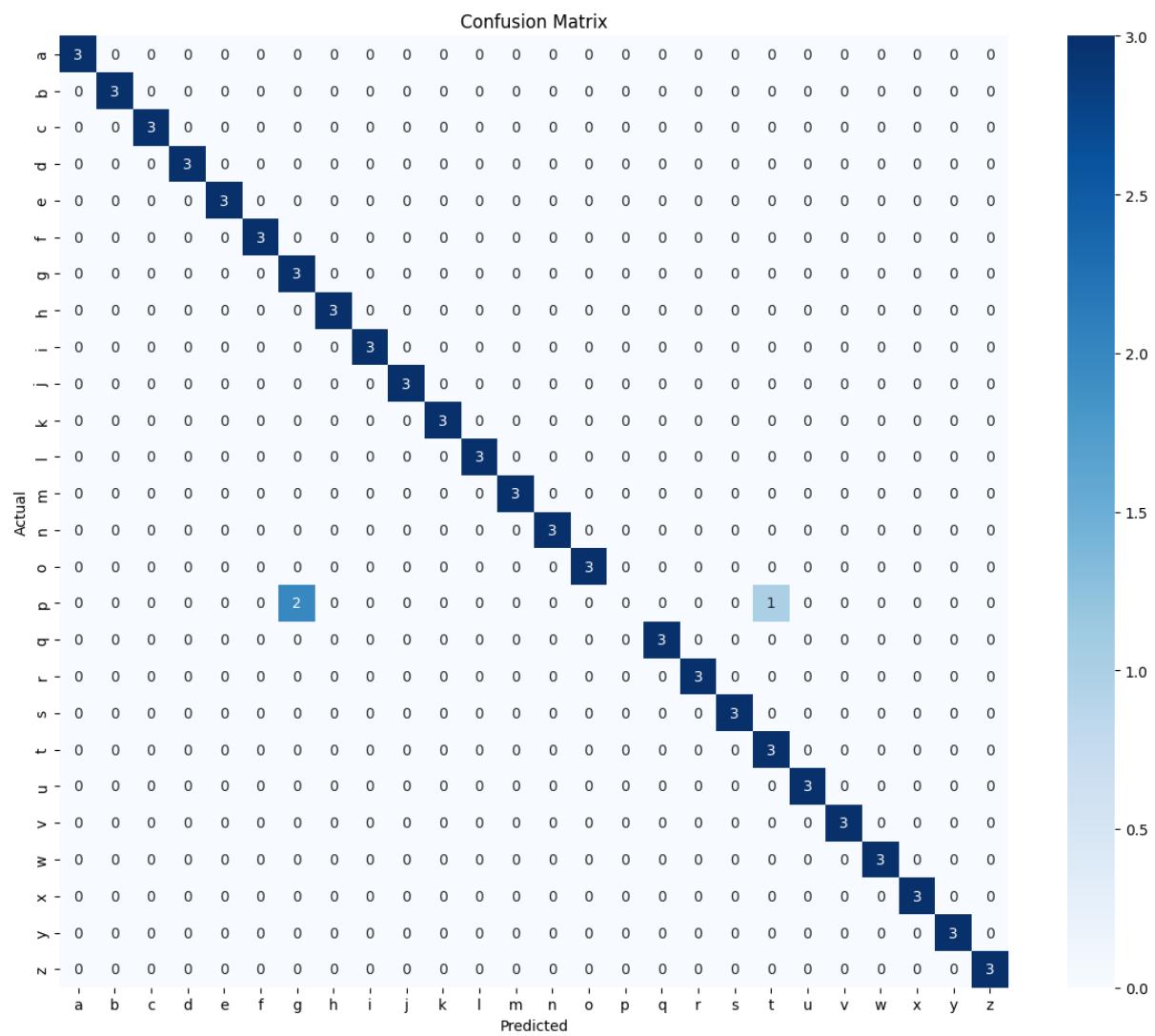


- Transformer

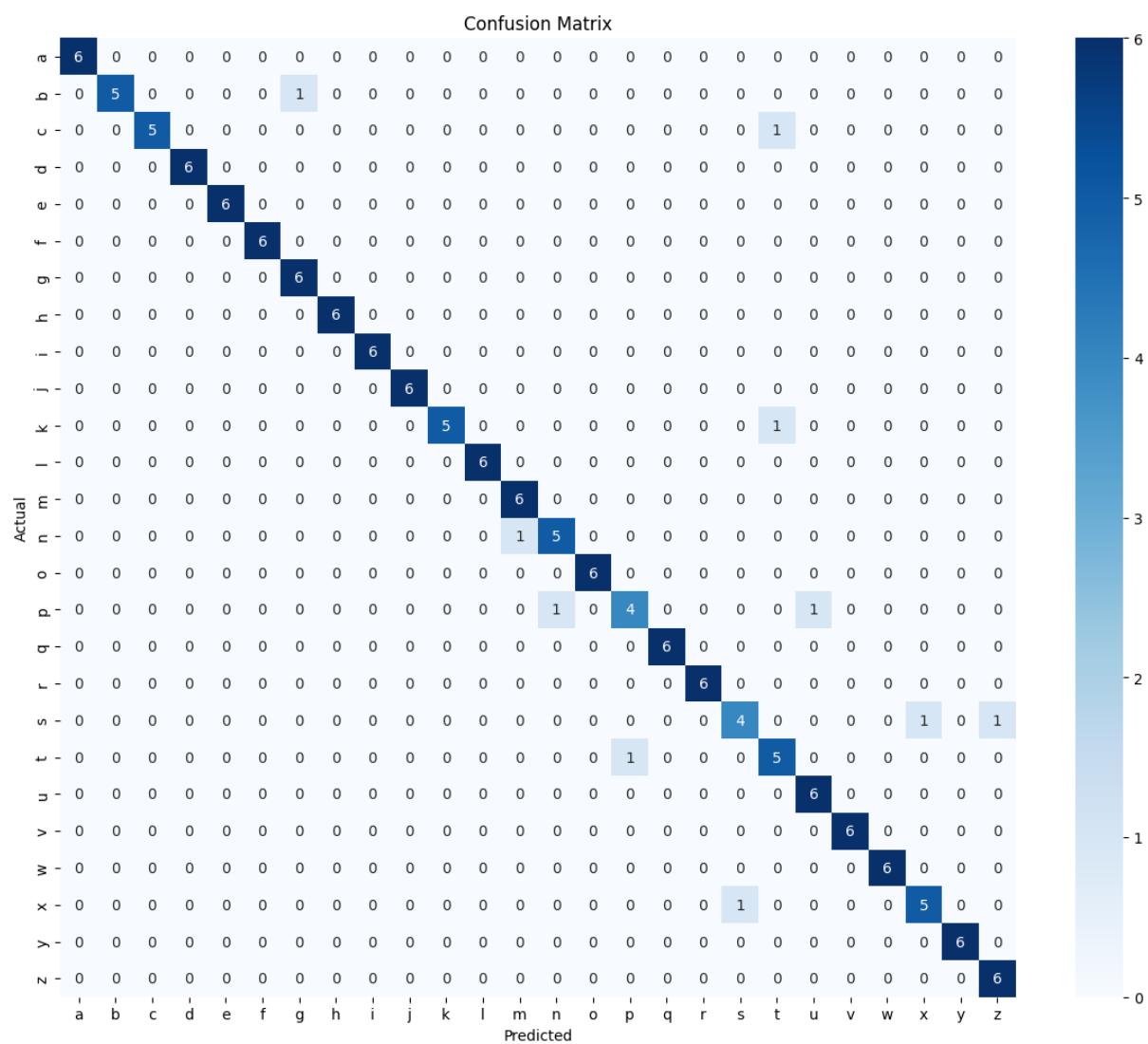
a. Bali



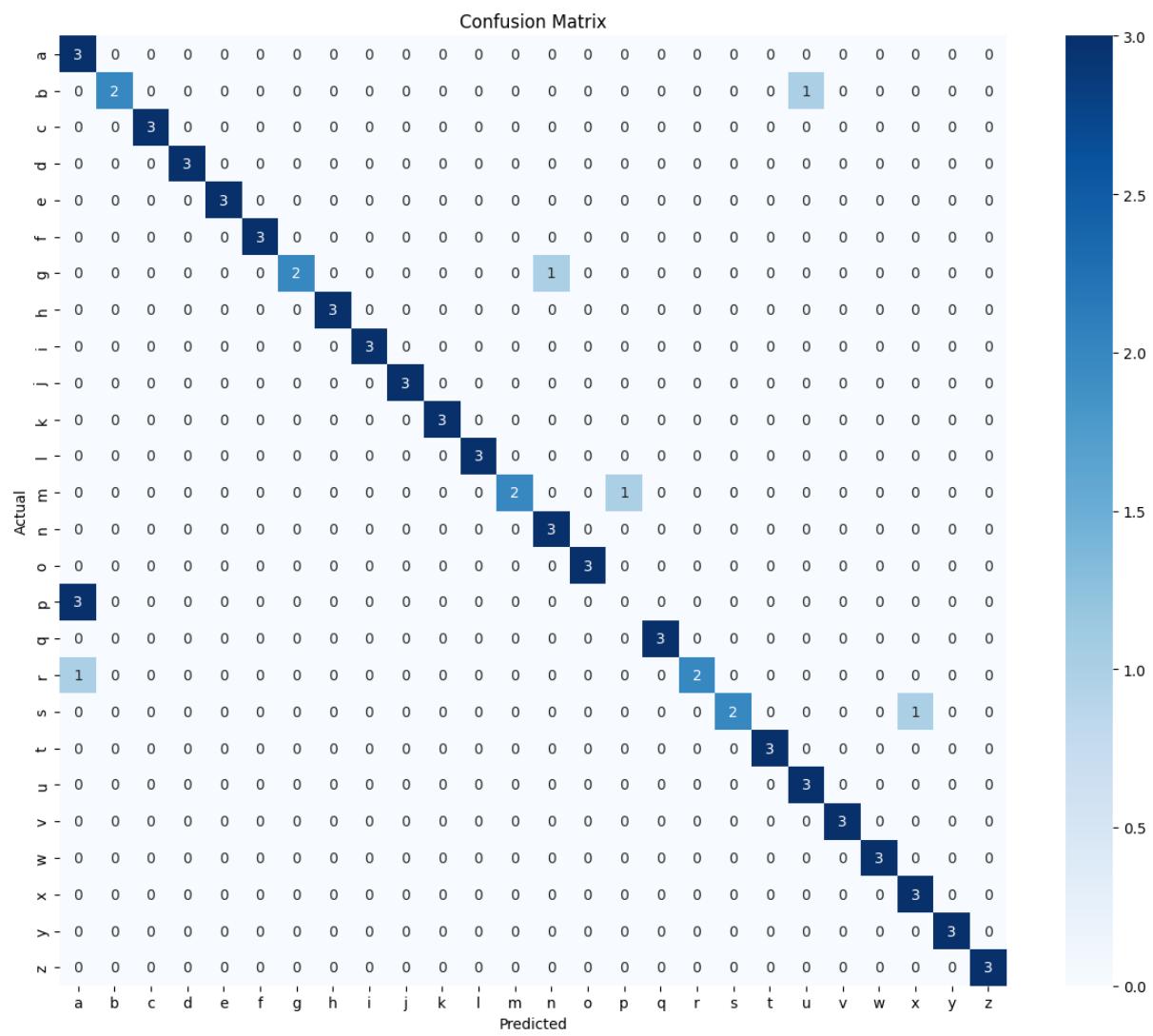
b. Batam



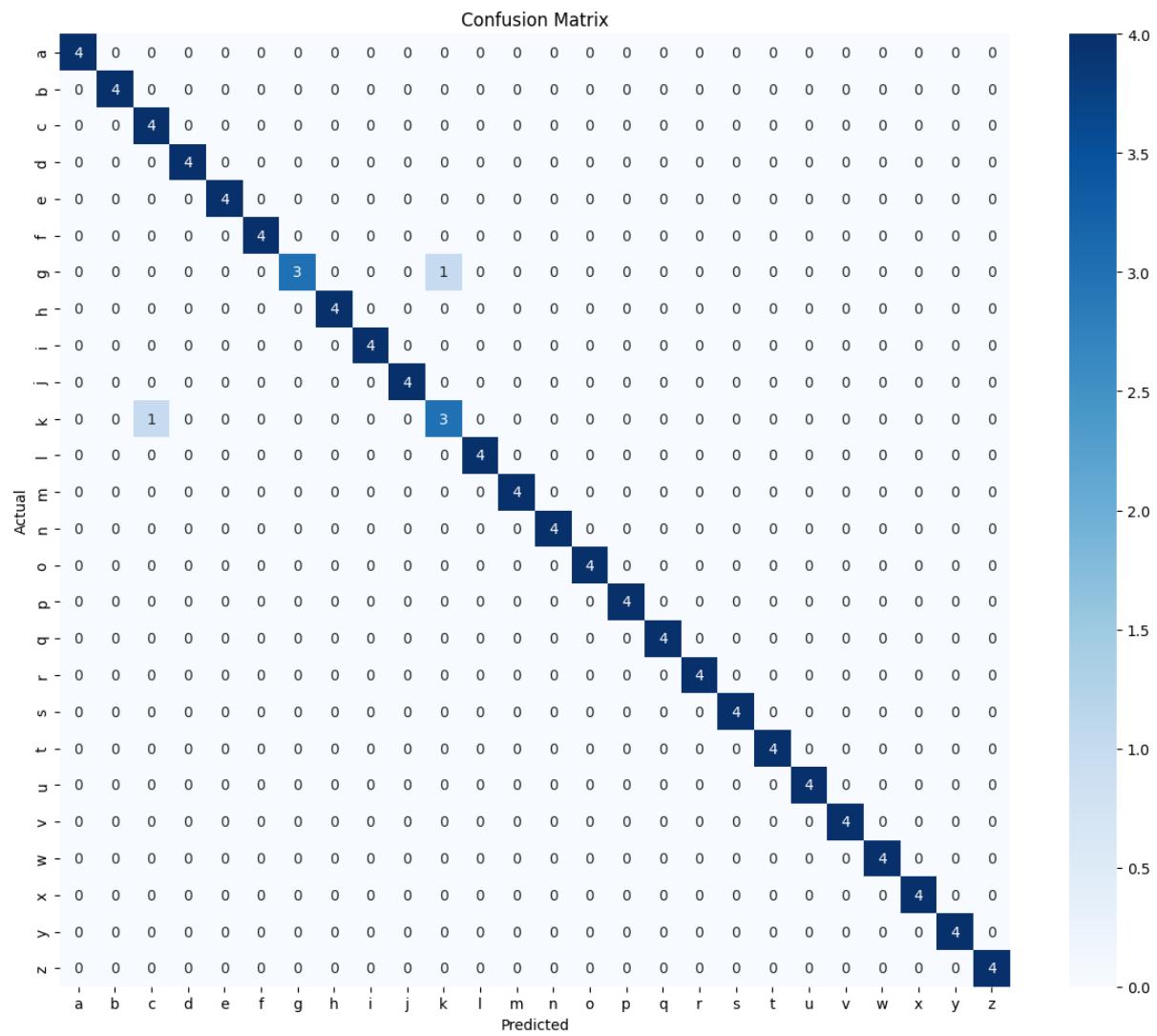
c. Jawa



d. Madura



e. Papua



[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

- Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 22(10), 1533–1545.
<https://doi.org/10.1109/TASLP.2014.2339736>
- Ali, Z. (2025). *A Comprehensive Overview and Comparative Analysis of CNN, RNN-LSTM and Transformer*. <https://doi.org/10.2139/ssrn.5175090>
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1). <https://doi.org/10.1186/s40537-021-00444-8>
- Alzu'Bi, D., & Duwairi, R. (2021). Detecting Regional Arabic Dialect based on Recurrent Neural Network. *2021 12th International Conference on Information and Communication Systems, ICICS 2021*, 90–93.
<https://doi.org/10.1109/ICICS52457.2021.9464605>
- Amami, R., Amami, R., Trabelsi, C., Mabrouk, S. H., & Khalil, H. A. (2023). A Robust Voice Pathology Detection System Based on the Combined BiLSTM-CNN Architecture. *MENDEL-Soft Computing Journal*, 29, 2571–3701.
<https://doi.org/10.13164/mendel.2023..202>
- Chamidy, T. (2016). Metode Mel Frequency Cepstral Coefficients (MFCC) Pada klasifikasi Hidden Markov Model (HMM) Untuk Kata Arabic pada Penutur Indonesia. *MATICS*, 8(1), 36. <https://doi.org/10.18860/mat.v8i1.3482>
- Chavan, K., & Gawande, U. (2015, October 5). Speech recognition in noisy environment, issues and challenges: A review. *Proceedings of the IEEE International Conference on Soft-Computing and Network Security, ICSNS 2015*.
<https://doi.org/10.1109/ICSNS.2015.7292420>
- Clara, S., Laksmi Prianto, D., Al Habsi, R., Friscila Lumbantobing, E., Chamidah, N., Kom, S., Kom, M., Informatika, J., Ilmu Komputer, F., Pembangunan Nasional Veteran Jakarta Jl Fatmawati Raya, U. R., Labu, P., Cilandak, K., & Depok, K. (2021). Implementasi Seleksi Fitur Pada Algoritma Klasifikasi Machine Learning Untuk Prediksi Penghasilan Pada Adult Income Dataset. In *Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA) Jakarta-Indonesia*.
- Ehri, L. C. (2020). The Science of Learning to Read Words: A Case for Systematic Phonics Instruction. *Reading Research Quarterly*, 55(S1), S45–S60.
<https://doi.org/10.1002/rrq.334>
- Esa, M. F. M., Mustaffa, N. H., Omar, H., M Radzi, N. H., & Sallehuddin, R. (2020). Learning Convolution Neural Network with Shift Pitching based Data Augmentation for Vibration Analysis. *IOP Conference Series: Materials Science and Engineering*, 864(1). <https://doi.org/10.1088/1757-899X/864/1/012086>
- Fadillah, R. Z., Irawan, A., Susanty, M., & Artikel, I. (2021). Data Augmentasi Untuk Mengatasi Keterbatasan Data Pada Model Penerjemah Bahasa Isyarat Indonesia (BISINDO). *JURNAL INFORMATIKA*, 8(2). <http://ejournal.bsi.ac.id/ejurnal/index.php/ji>
- Geng, M., Xie, X., Liu, S., Yu, J., Hu, S., Liu, X., & Meng, H. (2022). *Investigation of Data Augmentation Techniques for Disordered Speech Recognition*.
<https://doi.org/10.21437/Interspeech.2020-1161>
- Helmiyah, S., Fadlil, A., Yudhana, A., Dahlan, A., & Studi Teknik Elektro, P. (2018). Pengenalan Pola Emosi Manusia Berdasarkan Ucapan Menggunakan Ekstraksi Fitur Mel-Frequency Cepstral Coefficients (MFCC) Speech Based Emotion Pattern

- Recognition Using Mel-Frequency Cepstral Coefficients (MFCC) Feature Extraction. *Cogito Smart Journal* |, 4(2).
- Ibrahim, W., Candra, H., & Isyanto, H. (2022). Voice Recognition Security Reliability Analysis Using Deep Learning Convolutional Neural Network Algorithm. *Journal of Electrical Technology UMY (JET-UMY)*, 6(1).
- Kim, C., & Stern, R. M. (2016). Power-Normalized Cepstral Coefficients (PNCC) for Robust Speech Recognition. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 24(7), 1315–1329. <https://doi.org/10.1109/TASLP.2016.2545928>
- Lee, J. Y. (2021). Experimental evaluation of deep learning methods for an intelligent pathological voice detection system using the saarbruecken voice database. *Applied Sciences (Switzerland)*, 11(15). <https://doi.org/10.3390/app11157149>
- Lenson, A. K. S., & Airlangga, G. (2023). Comparative Analysis of MLP, CNN, and RNN Models in Automatic Speech Recognition: Dissecting Performance Metric. *Buletin Ilmiah Sarjana Teknik Elektro*, 5(4), 576–583. <https://doi.org/10.12928/biste.v5i4.9668>
- Li, X. (2024). Comparative analysis and prospect of RNN and Transformer. *Applied and Computational Engineering*, 75(1), 178–184. <https://doi.org/10.54254/2755-2721/75/20240535>
- Maurya, A., Kumar, D., & Agarwal, R. K. (2018). Speaker Recognition for Hindi Speech Signal using MFCC-GMM Approach. *Procedia Computer Science*, 125, 880–887. <https://doi.org/10.1016/j.procs.2017.12.112>
- Mehrish, A., Majumder, N., Bhardwaj, R., Mihalcea, R., & Poria, S. (2023). *A Review of Deep Learning Techniques for Speech Processing*. <http://arxiv.org/abs/2305.00359>
- Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. *Information*, 15(9), 517. <https://doi.org/10.3390/info15090517>
- Orken, M., Dina, O., Keylan, A., Tolganay, T., & Mohamed, O. (2022). A study of transformer-based end-to-end speech recognition system for Kazakh language. *Scientific Reports*, 12(1). <https://doi.org/10.1038/s41598-022-12260-y>
- Pham, N. Q., Nguyen, T. S., Niehues, J., Müller, M., & Waibel, A. (2019). Very deep self-attention networks for end-to-end speech recognition. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2019-September*, 66–70. <https://doi.org/10.21437/Interspeech.2019-2702>
- Prayogi, Yanuar Risah. (2015). Modifikasi Metode Ekstraksi Fitur Mel Frequency Cepstral Coefficient untuk Identifikasi Pembicara pada Lingkungan Berderau Menggunakan Residu Endpoint Detection.
- Prayogo, A., & Widyaningrum, L. (2017). Implementasi Metode Fonik dalam Pengenalan Bunyi Bahasa Inggris. In *DIMAS* (Vol. 17, Issue 1).
- Razak, Farzana Afifah. (2024). *Introducing Alphabet Phonics Spelling Using Machine Learning With MFCC Extraction*.
- Sarker, I. H. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. In *SN Computer Science* (Vol. 2, Issue 6). Springer. <https://doi.org/10.1007/s42979-021-00815-1>
- Sasilo, A. A., Saputra, R. A., & Ningrum, I. P. (2022). Sistem Pengenalan Suara Dengan Metode Mel Frequency Cepstral Coefficients Dan Gaussian Mixture Model. *Komputika : Jurnal Sistem Komputer*, 11(2), 203–210. <https://doi.org/10.34010/komputika.v11i2.6655>
- Syarif, A., Supriyati, Y., & Zulela, Z. (2020). Phonics Instruction with Storytelling Toward Learning to Read and Oral Language Development. *TARBIYA: Journal of Education in Muslim Society*, 6(2), 210–219. <https://doi.org/10.15408/tjems.v6i2.14322>

- Syed, S. A., Rashid, M., Hussain, S., & Zahid, H. (2021). Comparative Analysis of CNN and RNN for Voice Pathology Detection. *BioMed Research International*, 2021. <https://doi.org/10.1155/2021/6635964>
- Tridarma, P., & Endah, S. N. (2020). Pengenalan Ucapan Bahasa Indonesia Menggunakan MFCC dan Recurrent Neural Network. In *Jurnal Masyarakat Informatika* (Vol. 11, Issue 2).
- Tumanyan, N. (2022). Emotion Classification of Voice Recordings Using Deep Learning. *Mathematical Problems of Computer Science*, 57. <https://doi.org/10.51408/1963-0082>
- Utami, S., & Musthafa, B. (2023). The Utilization of Phonics Songs in Phonics Reading Classes in Indonesia: Teachers' Perspectives. *Jurnal Pendidikan Bahasa Dan Sastra*, 22(2), 201–208. https://doi.org/10.17509/bs_jpbsp.v22i2.55911
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. <http://arxiv.org/abs/1706.03762>
- Westhis, Sharina Munggaran. (2019). Metode Fonik dalam Pembelajaran Membaca Permulaan Bahasa Inggris Anak Usia Dini.
- Yağanoğlu, M., & Köse, C. (2018). Real-time Parental Voice Recognition System For Persons Having Impaired Hearing. *Bilge International Journal of Science and Technology Research*, 2(1), 40–46. <https://doi.org/10.30516/bilgesci.350016>
- Ye, S., Zhao, R., & Fang, X. (2019). An Ensemble Learning Method for Dialect Classification. *IOP Conference Series: Materials Science and Engineering*, 569(5). <https://doi.org/10.1088/1757-899X/569/5/052064>
- Zhang, L., & Sun, L.-G. (2020). Research on Data Preprocess in Data Mining. *DEStech Transactions on Computer Science and Engineering*, cmso. <https://doi.org/10.12783/dtcse/cmso2019/33630>
- Zhang, Z., He, C., & Yang, K. (2020). A novel surface electromyographic signal-based hand gesture prediction using a recurrent neural network. In *Sensors (Switzerland)* (Vol. 20, Issue 14, pp. 1–12). MDPI AG. <https://doi.org/10.3390/s20143994>

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS



Penulis lahir di Semarang pada 19 Juli 2003, merupakan anak kedua dari tiga bersaudara. Penulis telah menempuh pendidikan formal di TK Melati Kota Semarang, SDN Pedurungan Tengah 02 Semarang, SMPN 9 Semarang, dan SMAN 2 Semarang. Setelah lulus SMA pada 2021, penulis meneruskan studinya di Departemen Teknik Informatika FTEIC - ITS pada tahun 2021 dan terdaftar dengan NRP 5025211183.

Selama masa studi di Departemen Teknik Informatika, penulis sempat aktif di beberapa kegiatan akademik dan non akademik. Beberapa di antaranya adalah partisipasi dalam program Bangkit Academy, Staf Ahli Schematics 2023 dan Kerja Praktik di PT Telkom Indonesia Regional IV Kota Semarang yang diikuti sebagai upaya untuk memperluas wawasan dan mengembangkan kompetensi penulis.