

**INTEGRASI *FINE-TUNING* MODEL WHISPER DAN ALGORITMA
LEVENSHTTEIN DISTANCE SEBAGAI PENDEKATAN EVALUASI
PELAFALAN DOA HARIAN OLEH ANAK**



Disusun Oleh:

Nama : Umaymatun Az-Zauraa'

NIM : 22523283

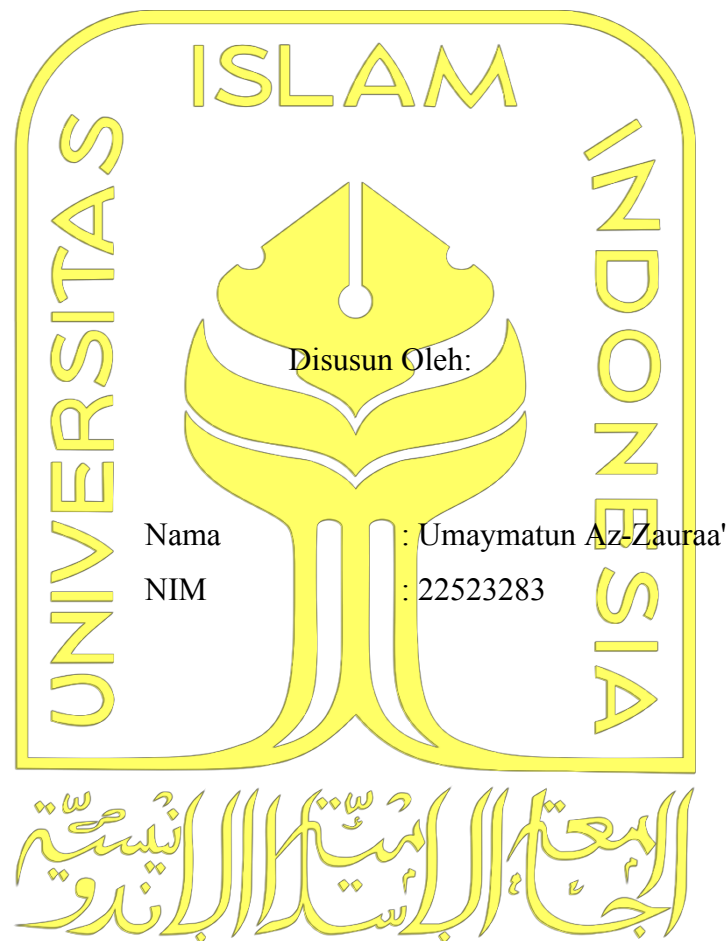
**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA**

2026

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**INTEGRASI *FINE-TUNING* MODEL WHISPER DAN ALGORITMA
LEVENSHTTEIN DISTANCE SEBAGAI PENDEKATAN EVALUASI
PELAFALAN DOA HARIAN OLEH ANAK**

TUGAS AKHIR



Yogyakarta, 26 April 2026

Pembimbing,

(Hari Setiaji, S.Kom., M.Eng.)

HALAMAN PENGESAHAN DOSEN PENGUJI

**INTEGRASI *FINE-TUNING* MODEL WHISPER DAN ALGORITMA
LEVENSHTEIN DISTANCE SEBAGAI PENDEKATAN EVALUASI
 PELAFALAN DOA HARIAN OLEH ANAK**

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 18 Mei 2026

Tim Penguji

Hari Setiaji, S.Kom., M.Eng.

Anggota 1

Kholid Haryono, S.T., M.Kom.

Anggota 2

Dr. Nur Wijyaning Rahayu, S.Kom., M.Cs.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia



(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Umaymatun Az-Zauraa'

NIM : 22523283

Tugas akhir dengan judul:

INTEGRASI *FINE-TUNING* MODEL WHISPER DAN ALGORITMA *LEVENSHTTEIN DISTANCE* SEBAGAI PENDEKATAN EVALUASI PELAFALAN DOA HARIAN OLEH ANAK

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 26 April 2026



(Umaymatun Az-Zauraa')

HALAMAN PERSEMBAHAN

Penulis mempersembahkan karya penelitian ini dengan penuh rasa syukur, cinta, dan bakti kepada keluarga tersayang. Untuk Abah dan Umi, terima kasih atas segala peluh, pengorbanan, serta lantunan doa yang tidak pernah putus mengiringi dan menguatkan setiap langkah penulis. Untuk Kakak dan Adik, terima kasih atas dukungan moral, keceriaan, dan semangat tanpa henti yang selalu diberikan selama proses penyelesaian tugas akhir ini.

Tidak lupa, persembahan ini juga penulis tujukan sebagai wujud apresiasi tertinggi untuk diri sendiri. Terima kasih karena telah memilih untuk tidak menyerah, menolak untuk berhenti melangkah, dan terus berjuang menyelesaikan penelitian ini di tengah berbagai tekanan, tantangan, dan hambatan. Terima kasih sudah bertahan hingga garis akhir.

HALAMAN MOTO

Allah tidak membebani seseorang, melainkan sesuai dengan kemampuannya.

(Q.S. Al-Baqarah: 286)

Sesungguhnya beserta kesulitan ada kemudahan.

(Q.S. Al-Insyirah: 5)

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas segala rahmat, karunia, dan hidayah-Nya, sehingga laporan tugas akhir yang berjudul “Integrasi *Fine-Tuning* Model Whisper dan Algoritma *Levenshtein Distance* sebagai Pendekatan Evaluasi Pelafalan Doa Harian Oleh Anak” ini dapat diselesaikan dengan baik. Shalawat serta salam semoga senantiasa tercurah kepada Nabi Muhammad SAW sebagai suri teladan dalam menuntut ilmu.

Laporan tugas akhir ini merupakan salah satu syarat dalam menyelesaikan pendidikan di Program Studi Informatika, Universitas Islam Indonesia. Dalam proses penyusunan tugas akhir ini, penulis menyadari bahwa terdapat berbagai kendala non-teknis yang dihadapi, seperti keterbatasan waktu, proses pengambilan data lapangan, serta tantangan dalam menjaga konsistensi pengerjaan. Namun demikian, berkat dukungan, bimbingan, dan bantuan dari berbagai pihak, laporan ini dapat diselesaikan dengan baik. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT, Tuhan Yang Maha Esa, atas segala nikmat, kasih, dan penyertaan-Nya dalam setiap langkah penulis.
2. Orang tua dan keluarga tercinta yang selalu memberikan doa, dukungan moral, dan materi selama penulis menjalani proses penyusunan laporan ini.
3. Bapak Hari Setiaji, S.Kom., M.Eng. yang telah memberikan arahan, bimbingan, dan motivasi selama proses penelitian dan penulisan laporan ini.
4. Seluruh dosen dan civitas akademika di Program Studi Informatika, yang telah membekali penulis dengan ilmu dan wawasan yang menjadi fondasi dalam menyelesaikan penelitian ini.
5. Sahabat-sahabat penulis, Novera, Fathya, Mifta, Ncip, Ahmad, Ihya, Fath, Hardy, dan Azza, atas kebersamaan, serta semangat yang telah membantu penulis menyelesaikan tugas akhir ini.
6. Teman seperjuangan penulis dalam masa penelitian dan penyusunan tugas akhir, Nasywa, yang selalu memberikan semangat, membantu proses penelitian, dan berprogres bersama.
7. Ibu Wiji Astuti, S.Hum. yang telah bersedia menjadi validator ahli dan membantu proses pengumpulan instrumen untuk kebutuhan penelitian.
8. Seluruh pengurus dan anak-anak dari TPA Ar-Rahmah Sawitsari, TPA Al-Baitul Makmur Wonorejo, dan TPA An-Nahrawi Plosokuning, yang telah berbaik hati memberikan izin dan kerja sama selama pelaksanaan penelitian.

Penulis menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk pengembangan penelitian selanjutnya. Penulis berharap penelitian ini dapat memberikan manfaat, baik sebagai kontribusi dalam pengembangan teknologi pembelajaran berbasis suara, maupun sebagai referensi bagi penelitian selanjutnya.

Yogyakarta, 26 April 2026

(Umaymatun Az-Zauraa')

SARI

Pendidikan agama Islam, khususnya dalam pembiasaan doa sehari-hari, menghadapi tantangan dalam hal pengawasan ketepatan pelafalan. Keterbatasan pendampingan dalam pembelajaran dalam pelafalan doa pada anak-anak, serta tidak tersedianya umpan balik saat latihan mandiri, menyebabkan proses evaluasi bacaan belum berjalan optimal. Kondisi ini terutama terlihat pada aspek *makharijul huruf* dan panjang pendek bacaan (*mad*), yang apabila tidak dikoreksi sejak dini dapat menimbulkan kesalahan pelafalan yang bersifat mendasar (*lahn jali*) dan berpotensi mengubah makna doa.

Untuk mengatasi permasalahan tersebut, penelitian ini menggunakan pendekatan *machine learning* berbasis *Automatic Speech Recognition* (ASR) dengan memanfaatkan model Whisper yang telah di-*fine-tuning* menggunakan *dataset* suara anak-anak, sehingga model mampu menyesuaikan variasi karakteristik akustik anak yang berbeda dengan suara orang dewasa. Model ini kemudian diintegrasikan dengan algoritma koreksi pelafalan berbasis *multi-layer text processing*, yang mencakup penyelarasan kata, deteksi kesalahan karakter menggunakan algoritma *two-pointer*, serta perhitungan tingkat kemiripan menggunakan *Levenshtein Distance*. Selain itu, sistem juga dilengkapi dengan validasi berbasis aturan tajwid untuk mengidentifikasi kesalahan *makhraj* dan *mad* sebagai bentuk deteksi *lahn jali*.

Hasil penelitian menunjukkan bahwa proses *fine-tuning* mampu meningkatkan performa model ASR secara signifikan dalam mengenali pelafalan anak, yang ditandai dengan penurunan nilai *Character Error Rate* (CER) dari 50,33% (*baseline*) menjadi 8,6%. Evaluasi terhadap algoritma koreksi pelafalan menghasilkan tingkat akurasi sebesar 92,83% dalam mendeteksi kesalahan pada level karakter. Namun demikian, tingkat kesesuaian dengan penilaian pakar berdasarkan koefisien *Cohen's Kappa* berada pada kategori sedang (0,4713), yang menunjukkan adanya perbedaan sensitivitas antara sistem dan manusia dalam menilai pelafalan anak. Sistem yang dikembangkan berhasil diimplementasikan secara luring (*offline*) pada perangkat Android melalui pendekatan *Edge AI* dengan integrasi *Java Native Interface* (JNI), sehingga dapat digunakan tanpa ketergantungan pada koneksi internet. Dengan demikian, sistem ini berpotensi menjadi media pembelajaran interaktif yang mampu memberikan umpan balik otomatis untuk mendukung latihan pelafalan doa secara mandiri pada anak-anak.

Kata kunci: *Automatic Speech Recognition*, Whisper, koreksi pelafalan, tajwid, *Levenshtein Distance*, *Edge AI*

GLOSARIUM

Automatic Speech Recognition	Teknologi yang digunakan untuk mengubah sinyal suara menjadi teks secara otomatis
Character Error Rate	Metrik evaluasi yang digunakan untuk mengukur tingkat kesalahan pengenalan karakter pada sistem pengenalan suara
Confusion Matrix	Tabel evaluasi yang digunakan untuk mengukur kinerja model klasifikasi berdasarkan perbandingan antara hasil prediksi dan kondisi aktual
Cohen's Kappa	Metrik statistik yang digunakan untuk mengukur tingkat kesepakatan antara dua penilai dengan mempertimbangkan kemungkinan kesepakatan secara kebetulan
Data Leakage	Kondisi di mana informasi dari data uji secara tidak sengaja digunakan dalam proses pelatihan model, sehingga menyebabkan hasil evaluasi menjadi bias
Edge AI	Pendekatan kecerdasan buatan yang dijalankan langsung pada perangkat lokal tanpa bergantung pada server atau koneksi internet
Fine-tuning	Proses pelatihan lanjutan pada model yang telah dilatih sebelumnya untuk menyesuaikan dengan <i>dataset</i> atau tugas tertentu
Java Native Interface	Antarmuka pemrograman yang memungkinkan kode Java atau Kotlin berinteraksi dengan kode <i>native</i> seperti C atau C++
Levenshtein Distance	Metode untuk mengukur perbedaan antara dua <i>string</i> berdasarkan jumlah operasi edit (penyisipan, penghapusan, atau penggantian karakter)
Makharijul Huruf	Ilmu dalam tajwid yang mempelajari tempat keluarnya huruf dalam pelafalan bahasa Arab
Two-Pointer Algorithm	Teknik algoritmik yang menggunakan dua indeks untuk menelusuri dan membandingkan dua data secara bersamaan
Whisper	Model <i>Automatic Speech Recognition</i> yang dikembangkan oleh OpenAI untuk melakukan transkripsi suara ke teks

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING.....	ii
HALAMAN PENGESAHAN DOSEN PENGUJI.....	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR	vii
SARI	ix
GLOSARIUM.....	x
DAFTAR ISI.....	xi
DAFTAR TABEL.....	xiv
DAFTAR GAMBAR.....	xv
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah dan Pertanyaan Penelitian.....	5
1.3 Batasan Masalah.....	5
1.4 Tujuan Penelitian.....	6
1.5 Manfaat Penelitian.....	6
1.6 Metodologi Penelitian	7
1.7 Sistematika Penulisan.....	8
BAB II KAJIAN PUSTAKA.....	10
2.1 Penelitian Terdahulu.....	10
2.2 Landasan Teori	12
2.2.1 Pengenalan Ucapan Otomatis (<i>Automatic Speech Recognition</i>).....	12
2.2.2 Karakteristik Akustik Suara Anak	15
2.2.3 Model Whisper OpenAI.....	15
2.2.4 <i>Fine-Tuning</i> pada Model ASR.....	20
2.2.5 Konsep <i>Lahn Jali</i> sebagai Parameter Evaluasi	21
2.2.6 Algoritma <i>Levenshtein Distance</i>	22
2.2.7 Metrik Evaluasi Sistem	23
BAB III METODOLOGI PENELITIAN	27
3.1 Pengumpulan Data.....	27

3.1.1	Pengumpulan Korpus Audio	28
3.1.2	Pengumpulan Data Label dan <i>Checkpoint</i>	31
3.2	Pra-Pemrosesan Data Audio	32
3.2.1	Pembersihan Sinyal Audio (<i>Audio Cleaning</i>)	32
3.2.2	Pemetaan dan Pembersihan Metadata	34
3.2.3	Pembagian <i>Dataset</i>	35
3.2.4	Augmentasi Data Latih	35
3.3	<i>Fine-Tuning</i> Model Whisper AI	38
3.3.1	Pemilihan Model <i>Baseline</i>	39
3.3.2	Ekstraksi Fitur dan Tokenisasi Data	39
3.3.3	Konfigurasi Hyperparameter	40
3.3.4	Metrik Evaluasi Kinerja Model	41
3.3.5	Skenario Evaluasi Model	42
3.4	Perancangan Algoritma Koreksi dan Sistem Umpan Balik	42
3.4.1	Kalkulasi Skor Kemiripan	43
3.4.2	Deteksi Kesalahan <i>Makhraj</i> dan <i>Mad</i>	46
3.5	Implementasi Model pada Lingkungan Android	50
3.5.1	Konversi dan Optimasi Model AI	50
3.5.2	Integrasi Model dan Algoritma via Java Native Interface (JNI)	51
3.5.3	Visualisasi Sistem sebagai Wadah Uji Coba	51
3.6	Pengujian dan Evaluasi Sistem	52
3.6.1	Skenario Pengujian Lapangan	52
3.6.2	Penetapan <i>Ground Truth</i> melalui Validasi Ahli (Expert Judgement)	52
3.6.3	Metrik Evaluasi Algoritma Koreksi	52
3.7	Lingkungan Eksperimen dan Implementasi Sistem	53
3.7.1	Lingkungan Pelatihan Model Kecerdasan Buatan	53
3.7.2	Lingkungan Pengembangan Aplikasi Bergerak	53
BAB IV HASIL DAN PEMBAHASAN		55
4.1	Pra-Pemrosesan Data Audio	55
4.1.1	Pembersihan Sinyal Suara (<i>Audio Cleaning</i>)	55
4.1.2	Pemetaan dan Pembersihan Metadata	58
4.1.3	Pembagian <i>Dataset</i>	59
4.1.4	Augmentasi Data	60
4.2	Proses <i>Fine-Tuning</i> dan Evaluasi Model Whisper AI	66

4.2.1 Konfigurasi <i>Hyperparameter</i>	66
4.2.2 Proses Pelatihan	67
4.2.3 Hasil Evaluasi Kinerja Model	70
4.3 Implementasi Algoritma Koreksi Pelafalan	75
4.3.1 Ekstraksi Teks Transkripsi	75
4.3.2 Penyelarasan Kata (<i>Word-Level Alignment</i>)	76
4.3.3 Penyelarasan Karakter dengan Algoritma <i>Two-Pointer</i>	77
4.3.4 Evaluasi Checkpoint Tajwid	78
4.3.5 Perhitungan Skor Kemiripan dengan <i>Levenshtein Distance</i>	79
4.4 Implementasi dan Integrasi Sistem	83
4.4.1 Konversi dan Optimasi Model	83
4.4.2 Integrasi Model dan Algoritma via Java Native Interface (JNI)	84
4.4.3 Perekaman dan Pemrosesan Audio di Latar Belakang	86
4.4.4 Visualisasi Sistem sebagai Wadah Uji Coba	88
4.5 Pengumpulan Data Uji Lapangan	90
4.6 Penetapan <i>Ground Truth</i> melalui Expert Judgement	92
4.6.1 Pelaksanaan <i>Blind Correction</i>	92
4.6.2 Hasil Evaluasi Pakar (<i>Ground Truth</i>)	93
4.7 Evaluasi Kinerja Algoritma Koreksi	95
4.7.1 <i>Confusion Matrix</i> Deteksi Kesalahan	96
4.7.2 Perhitungan Akurasi, Presisi, <i>Recall</i> , dan <i>F1-Score</i>	97
4.7.3 Perhitungan Koefisien <i>Cohen's Kappa</i>	98
BAB V KESIMPULAN DAN SARAN	100
5.1 Kesimpulan	100
5.2 Saran	101
DAFTAR PUSTAKA	102
LAMPIRAN	108

DAFTAR TABEL

Tabel 2.1 Perbandingan penelitian terdahulu.....	11
Tabel 2.2 Perbandingan arsitektur dan implementasi model ASR	13
Tabel 2.3 Spesifikasi varian model Whisper OpenAI	16
Tabel 2.4 Interpretasi nilai kappa oleh Landis dan Koch (1977).....	25
Tabel 3.1 Rangkuman distribusi <i>dataset</i>	29
Tabel 3.2 Cuplikan dokumentasi data primer mentah	30
Tabel 3.3 Konfigurasi <i>hyperparameter fine-tuning</i> model Whisper	41
Tabel 3.4 Contoh kasus perhitungan <i>Levenshtein Similarity</i>	45
Tabel 3.5 Pengelompokkan skor kemiripan untuk penilaian kualitas bacaan doa anak	46
Tabel 3.6 Pengelompokan umpan baik deteksi kesalahan tajwid pada <i>layer</i> koreksi	50
Tabel 4.1 Distribusi hasil akhir pembagian <i>dataset</i> audio doa	60
Tabel 4.2 Hasil evaluasi kinerja model <i>fine-tuned</i>	71
Tabel 4.3 Perbandingan metrik evaluasi model <i>baseline</i> vs model <i>fine-tuned</i>	72
Tabel 4.4 Rekapitulasi perbandingan tingkat kesalahan antara model <i>baseline</i> vs <i>fine-tuned</i>	73
Tabel 4.5 Contoh kasus pengujian perhitungan <i>Levenshtein Similarity</i>	81
Tabel 4.6 Perbandingan metrik beban penyimpanan model sebelum dan sesudah optimasi ..	84
Tabel 4.7 Daftar sampel audio pengujian lapangan.....	90
Tabel 4.8 Hasil evaluasi sistem terhadap data uji lapangan.....	91
Tabel 4.9 Rekapitulasi hasil evaluasi pakar (<i>ground truth</i>).....	93
Tabel 4.10 Hasil pemetaan <i>confusion matrix</i> algoritma koreksi.....	97

DAFTAR GAMBAR

Gambar 1.1 Perbandingan tingkat kesalahan ASR (WER) pada suara dewasa dan anak-anak	4
Gambar 2.1 Arsitektur Model Whisper	18
Gambar 3.1 Metodologi penelitian	27
Gambar 3.2 Contoh struktur JSON <i>checkpoint</i> evaluasi doa.....	32
Gambar 3.3 Alur proses augmentasi data untuk satu sampel audio masukan	36
Gambar 3.4 Alur pipa data dan skenario <i>fine-tuning</i> model Whisper	38
Gambar 3.5 Alur proses algoritma koreksi dan sistem pemberian umpan balik	43
Gambar 3.6 Ilustrasi mekanisme penguncian indeks jangkar absolut pada kasus kata terlewat	48
Gambar 4.1 Potongan kode pembersihan sinyal audio	56
Gambar 4.2 Perbandingan struktur sinyal audio mentah (atas) dan hasil pembersihan sinyal (bawah) pada sampel audio <i>doa_bangun_tidur_111</i>	57
Gambar 4.3 Proses pemuatan metadata	58
Gambar 4.4 Proses penghapusan fitur dan normalisasi ekstensi	59
Gambar 4.5 Proses pembagian data latih, data validasi, dan data uji	60
Gambar 4.6 Proses penambahan derau latar secara dinamis	61
Gambar 4.7 Perbandingan struktur sinyal audio asli (atas) dan sinyal audio dengan injeksi <i>gaussian noise</i> (bawah).....	62
Gambar 4.8 Proses pergeseran frekuensi nada secara dinamis.....	62
Gambar 4.9 Perbandingan spektrogram sinyal audio asli (atas) dan sinyal audio dengan pergeseran frekuensi nada ± 2 <i>semitones</i> (bawah).....	63
Gambar 4.10 Proses percepatan durasi secara dinamis	63
Gambar 4.11 Perbandingan struktur sinyal audio asli (atas) dan sinyal audio hasil percepatan durasi (bawah).....	64
Gambar 4.12 Proses perlambatan durasi secara dinamis	64
Gambar 4.13 Perbandingan struktur sinyal audio asli (atas) dan sinyal audio hasil perlambatan durasi (bawah).....	65
Gambar 4.14 Proses integrasi dan multiplikasi metadata audio baru hasil augmentasi	66
Gambar 4.15 Grafik <i>training loss</i> dan <i>validation loss</i> per epoch	68
Gambar 4.16 Grafik WER dan CER pada data validasi selama proses pelatihan	69
Gambar 4.17 Komparasi CER berdasarkan jenis doa.....	73
Gambar 4.18 Implementasi ekstraksi teks prediksi dan teks acuan.....	75

Gambar 4.19 Proses pemecahan kata dengan menggunakan ekspresi reguler	76
Gambar 4.20 Proses memasangkan kata dan menentukan status pasangan	77
Gambar 4.21 Implementasi algoritma <i>two-pointer lookahead</i> untuk penyelarasan karakter ..	78
Gambar 4.22 Proses validasi indeks kesalahan terhadap <i>hitbox checkpoint</i> tajwid	78
Gambar 4.23 Fungsi evaluasi mad berbasis rasio panjang karakter	79
Gambar 4.24 Implementasi fungsi <i>Levenshtein Distance</i> dan perhitungan <i>similarity</i> (skor kemiripan) pada sistem	80
Gambar 4.25 Hasil evaluasi <i>similarity</i> bacaan pada aplikasi <i>mobile</i>	82
Gambar 4.26 Kode eksekusi perintah konversi model ke dalam format GGML	84
Gambar 4.27 Deklarasi antarmuka JNI pada lapisan Kotlin	85
Gambar 4.28 Implementasi pengaturan parameter dan eksekusi pada lapisan <i>Native C++</i>	86
Gambar 4.29 Implementasi kelas perekam audio dan konversi <i>floating-point</i>	87
Gambar 4.30 Antarmuka tampilan doa terpilih (atas) dan proses tes pengujian sistem saat proses penangkapan audio (bawah)	88
Gambar 4.31 Implementasi pewarnaan karakter dan tampilan umpan balik	89
Gambar 4.32 Visualisasi hasil koreksi algoritma dan umpan balik kesalahan tajwid	89
Gambar 4.33 Visualisasi <i>heatmap confusion matrix</i> algoritma koreksi	96

BAB I PENDAHULUAN

1.1 Latar Belakang

Pendidikan agama Islam memegang peranan krusial dalam pembentukan karakter moral dan etika pada anak, terutama apabila ditanamkan sejak rentang usia sekolah dasar, yakni usia 6 hingga 12 tahun (Puteri, 2025). Pada usia tersebut, anak berada dalam fase perkembangan di mana integrasi nilai-nilai keagamaan tidak hanya berfungsi untuk membangun kedisiplinan spiritual, tetapi juga memberikan stimulasi positif pada perkembangan struktur kognitifnya (Masykuri *et al.*, 2024). Sehingga perhatian pendidikan keagamaan pada fase ini sangat menentukan tumbuh kembang mereka. Salah satu metode paling efektif yang dapat diterapkan pada usia ini yakni pembiasaan praktik ibadah, seperti pelafalan doa harian berbahasa Arab secara rutin. Pembiasaan pelafalan yang dilakukan melalui pengulangan atau *takrir* secara konsisten, terbukti mampu memindahkan informasi ke dalam memori jangka panjang anak secara efisien (Abouhalim, Afarfar dan Belahsen, 2023; Ridiawati *et al.*, 2025). Selain memperkuat daya ingat episodik, pembiasaan menghafal lafaz berbahasa Arab ini juga berperan secara signifikan dalam melatih dan meningkatkan kemampuan linguistik anak sejak dini (Hasim *et al.*, 2023).

Implementasi pembelajaran agama melalui pembiasaan doa ini memiliki urgensi yang sangat tinggi. Namun, dalam praktiknya, anak dengan rentang usia sekolah dasar sering kali menghadapi tantangan signifikan, khususnya tantangan fonologi pada saat melafalkan teks berbahasa Arab (Fitrianingrum dan Aminingsih, 2024; Wahyudin, Karim dan Solehah, 2024). Kesulitan ini dipengaruhi oleh interferensi fonetik dari bahasa ibu, di mana organ anak terbiasa dengan logat bahasa utamanya, sehingga anak kerap kali melakukan substitusi fonem dengan mengganti bunyi huruf Arab menggunakan konsonan yang paling dekat pada bahasa Indonesia (Nailurrahmi dan Marlina, 2025). Hal ini mengakibatkan ketidaktepatan pelafalan secara masif pada aspek *makhraj* dan *mad*, salah satunya kondisi di mana anak kesulitan dalam membedakan beberapa huruf *hijaiyah* yang tidak ada pada bahasa Indonesia, seperti ض, ص, ذ, خ, ث (Wahyudin, Karim dan Solehah, 2024) maupun tertukarnya huruf yang berdekatan, seperti ء dengan ع, ك dengan ق, ذ dengan ز, serta ث dengan س (Syari dan Mahendra, 2022; Kurniawati, Ni'mah dan Sofa, 2025).

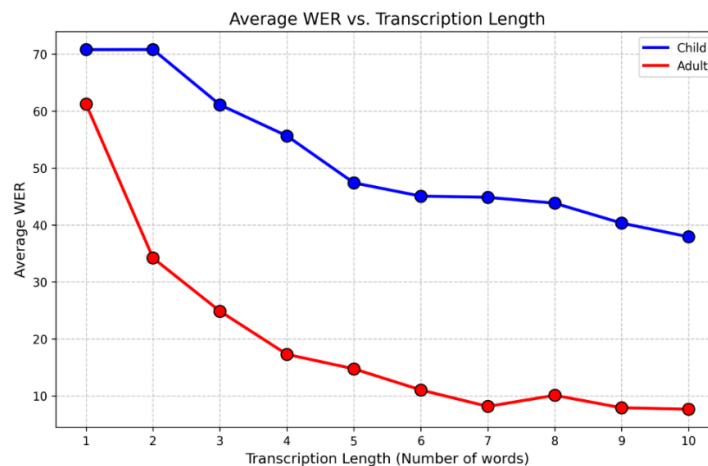
Dalam ilmu tajwid, kesalahan dalam pelafalan ayat Al-Qur'an, atau yang dikenal dengan istilah *al-lahn* diklasifikasikan ke dalam dua kategori, yakni *lahn jali* (jelas) dan *lahn khafi*

(tersembunyi) (Marzuki dan Ummah, 2020). *Al-lahnul jali* merupakan kesalahan pada pelafalan lafaz Al-Qur'an yang menyalahi kaidah tajwid, seperti mengganti huruf, mengubah harakat, maupun kekeliruan dalam panjang bacaan (*mad*), yang berpotensi mengubah makna kata. Sedangkan, *al-lahnul khafi* merupakan kesalahan pada pelafalan lafaz Al-Qur'an yang menyalahi sebagian kaidah tajwid, tetapi tidak menyalahi kaidah bahasa Arab sehingga tidak mengubah makna kata. Mengingat target pengguna sistem pada penelitian ini adalah anak dengan rentang usia sekolah dasar yang masih berada dalam tahap pengenalan dasar, evaluasi pelafalan doa harian difokuskan secara spesifik pada deteksi dan pencegahan *lahn jali*, yang berpotensi mengubah makna doa yang dilafalkan. Selain itu, kebiasaan melakukan kesalahan pelafalan yang tidak segera dikoreksi sejak dini berisiko menetap pada memori jangka panjang anak hingga ia tumbuh dewasa nanti. Maka dari itu, diperlukan adanya pendampingan serta mekanisme umpan balik dan koreksi yang konsisten untuk mencegah mengakarnya kesalahan pelafalan tersebut.

Saat ini, metode evaluasi yang digunakan untuk memperbaiki pelafalan bacaan anak masih sangat bergantung pada metode *talaqqi*. Metode di mana pembelajaran dan koreksi bacaan dilakukan secara langsung, dengan tatap muka bersama pengajar atau *ustadz*. Meskipun metode ini diakui memiliki tingkat akurasi yang mutlak dalam membenarkan *makhraj* dan *mad*, sayangnya *talaqqi* memiliki keterbatasan pada ruang dan waktu yang signifikan (Setiawan, Hartati dan Luthfi, 2025). Keterbatasan rasio jumlah pengajar berbanding lurus dengan minimnya waktu yang dimiliki *ustadz* dalam mengevaluasi bacaan masing-masing anak secara detail setiap saat. Akibatnya, ketika anak berlatih secara mandiri di rumah, ketiadaan umpan balik korektif yang objektif membuat proses evaluasi menjadi tidak terkontrol, sehingga anak seringkali tidak menyadari kesalahan pelafalan yang mereka lakukan (Sevara, 2024). Sebuah studi oleh (Khairani *et al.*, 2024) mengungkapkan bahwa belum ada aplikasi populer di pasaran yang menyematkan fitur koreksi bacaan otomatis untuk domain doa anak. Padahal, umpan balik korektif ini sangat penting untuk memastikan proses belajar anak berjalan dengan benar sejak awal, terutama untuk doa berbahasa Arab yang memiliki kaidah pelafalan yang ketat. Merespons keterbatasan tersebut, maka diperlukan adanya sebuah sistem yang mampu memberikan umpan balik instan, yang dapat mendampingi proses latihan mandiri anak dan meningkatkan keterlibatan serta pemahamannya (Ibadurrahman *et al.*, 2025). Sistem dirancang untuk membantu anak memvalidasi dan mengoreksi pelafalan mereka secara berulang sebelum akhirnya bacaan diajukan pada pengajar.

Untuk membangun sistem umpan balik mandiri, pemanfaatan teknologi kecerdasan buatan, khususnya sistem pengenalan dan analisis suara otomatis (*Automatic Speech Recognition*) menjadi salah satu pendekatan komputasi paling potensial dan menjanjikan. Hal ini dikarenakan anak dapat secara langsung berinteraksi dengan teknologi melalui ucapan dan mendapat umpan balik secara *real-time*. *Automatic Speech Recognition* (ASR) bekerja dengan cara memproses sinyal akustik dari ucapan manusia dan mengubahnya menjadi teks digital secara otomatis (Khairani *et al.*, 2024). Dalam konteks pembelajaran bahasa Arab, integrasi ASR memungkinkan sistem untuk menjadi “pendengar virtual” yang secara otomatis dapat mendeteksi setiap pelafalan huruf *hijaiyah* yang diucapkan oleh anak (Al-Fadhli, Al-Harbi dan Cherif, 2023). Melalui pendekatan ini, hasil transkripsi mesin dapat dievaluasi untuk mendeteksi letak kesalahan pelafalan anak secara objektif dan *real-time*, sehingga umpan balik dapat diberikan secara langsung melalui antarmuka sistem dalam bentuk perangkat *mobile* berbasis Android (Sujjada, Insany dan Nugraha, 2024).

Sayangnya, implementasi model ASR konvensional dihadapkan pada kendala teknis yang serius ketika memproses suara anak-anak. Secara akustik, suara anak-anak memiliki karakteristik yang sangat berbeda dengan orang dewasa. Rentang *pitch* mereka jauh lebih tinggi, karakteristik gelombang suara tidak stabil karena organ pita suaranya masih dalam tahap pertumbuhan, serta durasi jeda saat berbicara yang tidak beraturan (Yeung, Fan dan Alwan, 2021). Kondisi ini kemudian memicu terjadinya ketidakcocokan akustik pada tahap pemrosesan sinyal. Selain itu, model ASR standar yang mayoritas dilatih menggunakan korpus data suara orang dewasa, ketika diaplikasikan pada konteks pelafalan anak-anak, menimbulkan ketidaksesuaian domain data yang dapat memengaruhi akurasi model. Studi oleh (Ashvin *et al.*, 2025) membuktikan adanya kesenjangan performa yang signifikan pada model ASR ketika digunakan untuk mengenali suara anak, di mana tingkat *Word Error Rate* (WER) model melonjak lebih tinggi dibandingkan pemrosesan suara orang dewasa. Lonjakan WER dapat dilihat dalam Gambar 1.1.



Gambar 1.1 Perbandingan tingkat kesalahan ASR (WER) pada suara dewasa dan anak-anak

Sumber: (Ashvin *et al.*, 2025)

Merespons tingginya tingkat kesalahan komputasi tersebut, model ASR dengan arsitektur *deep learning* berbasis *transformer*, seperti Whisper yang dikembangkan oleh OpenAI menawarkan tingkat ketahanan yang luar biasa terhadap *noise* (Radford *et al.*, 2023). Melalui pelatihan skala besar menggunakan lebih dari 680.000 jam data dengan pengawasan lemah, Whisper memiliki kemampuan untuk mengenali suara multibahasa yang sangat tangguh, dalam kondisi suara yang bising maupun aksen bervariasi, dengan tingkat akurasi yang tinggi (Radford *et al.*, 2023). Dalam konteks pemrosesan suara dalam bahasa Arab, penelitian sebelumnya, telah membuktikan bahwa Whisper memperlihatkan performa dasar yang sangat baik dengan *Word Error Rate* (WER) di bawah 10% untuk domain formal dibandingkan arsitektur ASR lainnya (Talafta, Waheed dan Abdul-Mageed, 2023; Khairani *et al.*, 2024). Meski demikian, pengintegrasian ASR untuk mengenali karakteristik suara dalam konteks pembelajaran doa oleh anak, memerlukan proses *fine-tuning* untuk mengatasi masalah ketidakcocokan domain data akibat data yang belum pernah dipelajari sebelumnya (Talafta, Waheed dan Abdul-Mageed, 2023). Hal ini dilakukan agar model mampu mengenali karakteristik vokal anak-anak dengan lebih akurat.

Berdasarkan konteks inilah, penelitian ini dijalankan dengan tujuan mengembangkan sistem evaluasi bacaan doa anak terintegrasi, yang menyatukan teknologi ASR (Whisper) hasil *fine-tuning* dengan algoritma komputasi *Levenshtein Distance*. Sebagai wadah pengujian untuk memvalidasi performa algoritma tersebut secara praktis, sistem diimplementasikan ke dalam purwarupa antarmuka *mobile* berbasis Android. Melalui antarmuka ini, hasil komputasi

divisualisasikan ke dalam bentuk penilaian bacaan dan umpan balik warna pada teks Arab secara *real-time* untuk menunjukkan letak kesalahan pelafalan.

1.2 Rumusan Masalah dan Pertanyaan Penelitian

Rumusan masalah dari penelitian ini adalah:

- a. Karakteristik suara anak yang memiliki variasi *pitch*, artikulasi, dan tempo yang tinggi menjadi kendala utama bagi akurasi model *Automatic Speech Recognition* (ASR) standar.
- b. Model pengenalan suara yang sudah ada seperti Whisper dari OpenAI, belum optimal dalam mentranskripsikan pelafalan doa harian berbahasa Arab oleh subjek anak-anak.
- c. Evaluasi bacaan doa anak dengan *metode talaqqi* saat ini masih bersifat subjektif dan manual, sementara kebutuhan akan sarana latihan mandiri yang memberikan umpan balik objektif terus meningkat.
- d. Belum tersedianya sebuah sistem terintegrasi yang menggabungkan model ASR dan algoritma koreksi otomatis ke dalam aplikasi *mobile* (Android) sebagai *Proof of Concept* (PoC) untuk memberikan evaluasi visual di lingkungan nyata.
- e. Hasil evaluasi dari sistem koreksi memerlukan validasi ahli untuk memastikan kesesuaiannya dengan standar pembelajaran aktual.

Berdasarkan rumusan masalah yang telah disusun, secara spesifik penelitian ini dirancang untuk menjawab pertanyaan penelitian berikut:

1. Seberapa efektif metode *fine-tuning* pada model Whisper dalam meningkatkan akurasi pengenalan bacaan doa anak dibandingkan model *baseline*?
2. Bagaimana tingkat kesesuaian antara hasil deteksi kesalahan *lahn jali* yang dihasilkan oleh sistem dengan penilaian manual dari pakar?
3. Bagaimana performa algoritma koreksi yang diintegrasikan pada aplikasi Android dalam mendeteksi dan mengevaluasi kesalahan pelafalan doa secara otomatis?

1.3 Batasan Masalah

Untuk menjaga kedalaman analisis dan memastikan penelitian berjalan secara terarah, maka titik fokus yang diterapkan dalam penelitian ini, antara lain:

- a. Pengumpulan data suara difokuskan pada subjek anak-anak dengan rentang usia 6 hingga 12 tahun.

- b. Model ASR yang digunakan adalah Whisper (OpenAI) yang telah melalui proses *fine-tuning* untuk domain pelafalan doa berbahasa Arab oleh anak.
- c. Evaluasi bacaan menerapkan algoritma pencocokan *string* hasil transkripsi model terhadap transkrip referensi, dengan titik fokus pada deteksi kesalahan pelafalan *makhraj* dan *mad* (*lahn jali*) untuk 3 jenis doa harian.
- d. Pengujian performa model ASR dikhususkan pada penggunaan metrik *Word Error Rate* (WER) dan *Character Error Rate* (CER), serta validasi algoritma koreksi yang diselaraskan secara langsung dengan standar penilaian ahli.
- e. Aplikasi berbasis Android dikhususkan sebagai *Proof of Concept* (PoC) untuk menjalankan inferensi model secara lokal dan memvisualisasikan hasil koreksi, sehingga pengembangan murni berfokus pada pengujian fungsionalitas algoritma.

1.4 Tujuan Penelitian

Sejalan dengan rumusan masalah dan pertanyaan penelitian yang telah ditetapkan, maka tujuan dari penelitian ini adalah:

- a. Mengevaluasi tingkat efektivitas metode *fine-tuning* pada arsitektur model ASR Whisper dalam meningkatkan akurasi transkripsi dalam mengenali bacaan doa anak dibandingkan dengan model *baseline*.
- b. Mengetahui tingkat kesesuaian antara evaluasi kesalahan pelafalan yang dihasilkan oleh sistem dengan penilaian manual secara langsung dari validator ahli.
- c. Menganalisis performa algoritma koreksi yang diintegrasikan pada aplikasi Android sebagai *Proof of Concept*, dalam mendeteksi dan mengevaluasi kesalahan pelafalan doa anak secara otomatis.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat dan kontribusi yang berarti bagi pengembangan keilmuan maupun penerapannya di masyarakat:

- a. Memberikan wawasan baru mengenai proses pengoptimalan model ASR dengan metode *fine-tuning* dalam mengenali suara anak-anak secara spesifik pada pelafalan bahasa Arab.
- b. Menjadi sarana latihan mandiri yang objektif bagi anak untuk memperbaiki pelafalan dasar sebelum proses *talaqqi* kepada pengajar.
- c. Memberikan referensi praktis terkait integrasi model ASR dan algoritma koreksi ke dalam bentuk purwarupa berupa aplikasi Android.

1.6 Metodologi Penelitian

Metodologi penelitian merupakan rangkaian tahapan yang dilalui oleh peneliti sepanjang melakukan penelitian, mulai dari merumuskan masalah hingga mencapai kesimpulan. Metodologi ini memiliki peran penting dalam proses penelitian oleh peneliti, yakni sebagai panduan utama yang memastikan bahwa proses yang dilakukan selama penelitian berlangsung akan menghasilkan luaran yang sesuai dengan tujuan yang diharapkan. Penelitian ini melalui serangkaian tahapan berikut:

a. Studi Literatur

Pada tahap ini, peneliti mengkaji literatur yang tersedia untuk menemukan informasi yang relevan dengan masalah penelitian. Tahap ini bertujuan untuk memahami permasalahan dan potensi solusi yang diangkat, yakni penggabungan integrasi model ASR dan algoritma *Levenshtein Distance* dalam membentuk sistem koreksi otomatis pelafalan doa harian oleh anak. Sumber informasi berasal dari buku, artikel riset, publikasi ilmiah, laman resmi penyedia model ASR, serta sumber elektronik lainnya.

b. Pengumpulan Data

Tahap ini digunakan untuk mempersiapkan segala kebutuhan yang dapat menunjang proses pelatihan atau *fine-tuning* model ASR. Proses pengumpulan data dibagi menjadi dua jenis. Pertama, pengumpulan korpus audio yang dilakukan dengan menggabungkan pencarian dari sumber daring dan sumber lapangan. Kedua, pengumpulan data label dan *checkpoint* evaluasi bacaan untuk setiap doa yang ada.

c. Pra-pemrosesan Data Audio

Implementasi pra-pemrosesan data audio dalam penelitian ini terbagi ke dalam empat skenario utama: pembersihan sinyal audio, pemetaan metadata, pembagian *dataset*, dan augmentasi data. Tahap ini bertujuan untuk memastikan bahwa data audio masukan memiliki sinyal yang bersih, sesuai format optimal yang dibutuhkan oleh *pipeline* pelatihan model ASR, dan kualitas data yang baik.

d. *Fine-Tuning* Model ASR

Pada tahap ini, peneliti melakukan pelatihan ulang (*fine-tuning*) model ASR dasar menggunakan data audio anak-anak yang telah dikumpulkan agar model dapat menyesuaikan hasil transkrip dengan karakteristik suara anak-anak yang variatif dan cenderung berbeda dengan pelafalan orang dewasa. Pelatihan dilakukan dengan memantau

metrik *training-validation loss* serta *Word Error Rate* (WER) dan *Character Error Rate* (CER) untuk hasil model yang optimal.

e. Perancangan Algoritma Koreksi dan Sistem Umpan Balik

Pada tahap ini, algoritma koreksi sebagai inti sistem dirancang sedemikian rupa untuk menjalankan fungsinya sebagai pendeteksi letak kesalahan pada pelafalan doa oleh anak. Proses ini dilakukan untuk mendukung fungsi sistem dalam memberikan penilaian dan umpan balik otomatis terhadap bacaan anak.

f. Implementasi Model pada Lingkungan Android

Pada tahap ini, model ASR Whisper yang telah melalui proses *fine-tuning* dioptimasi arsitektur modelnya dengan menyesuaikan karakteristik lingkungan Android. Proses implementasi memanfaatkan kombinasi bahasa antara kotlin sebagai antarmuka sistem dengan C++ melalui *Java Native Interface* (JNI) yang dapat membantu model beroperasi di dalam lingkungan Android.

g. Pengujian dan Evaluasi Sistem

Setelah rangkaian model dan algoritma diintegrasikan dalam perangkat *mobile*, tahap pengujian dilakukan untuk menilai performa sistem koreksi bacaan otomatis yang diangkat sebagai solusi. Pengujian dilakukan dengan mengunjungi salah satu TPA di kawasan Yogyakarta, lalu sampel dan hasil uji akan disimpan untuk tahap evaluasi sistem. Tingkat kesesuaian sistem dalam mendeteksi letak kesalahan dibandingkan dengan penilaian pakar menggunakan metode *Cohen's Kappa*. Sementara, performa sistem koreksi akan dievaluasi dengan menggunakan *confusion matrix*.

h. Penyusunan Laporan

Seluruh hasil dari tahap pengkajian literatur, persiapan, perancangan, pengembangan, implementasi model, hingga pengujian dan evaluasi sistem didokumentasikan pada tahap ini dengan penyusunan laporan yang sistematis. Analisis hasil dilakukan untuk menjawab rumusan masalah dan mengukur kinerja dan kesesuaian sistem berdasarkan validasi pakar. Laporan ini juga dapat menjadi acuan bagi penelitian maupun pengembangan selanjutnya.

1.7 Sistematika Penulisan

a. BAB I – PENDAHULUAN

Bab ini berisi pemaparan latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta sistematika penulisan laporan.

b. BAB II – KAJIAN PUSTAKA

Bab ini membahas *gap* atau celah penelitian terdahulu yang diangkat dalam penelitian, perbandingan beberapa arsitektur model *automatic speech recognition*, serta landasan teori seputar hal-hal yang berkaitan dalam pengintegrasian model ASR dan algoritma deteksi dalam sistem koreksi bacaan anak.

c. BAB III – METODOLOGI PENELITIAN

Bab ini merinci tahapan dalam metodologi penelitian yang digunakan untuk mencapai tujuan penelitian yang telah disusun, mulai dari pengumpulan data, tahap pra-pemrosesan data audio, proses *fine-tuning* model ASR, perancangan algoritma koreksi, implementasi model, hingga pengujian dan evaluasi.

d. BAB IV – HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil penelitian yang dilakukan, dilanjutkan dengan analisis hasil evaluasi kinerja model, tingkat akurasi, proses integrasi model dan algoritma koreksi pada lingkungan Android, serta pengujian dan validasi sistem koreksi yang dikembangkan.

e. BAB V – KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari hasil penelitian dan pembahasan yang telah dipaparkan. Selain itu, saran untuk penelitian selanjutnya juga disampaikan dalam bab ini.

BAB II

KAJIAN PUSTAKA

2.1 Penelitian Terdahulu

Pengembangan sistem ASR untuk mengevaluasi pelafalan bahasa Arab telah menjadi fokus penelitian yang ekstensif. Namun, dalam konteks pembelajaran agama Islam, sebagian besar literatur yang tersedia saat ini, seperti penelitian oleh (Al-Fadhli, Al-Harbi dan Cherif, 2023) sangat terfokus pada evaluasi bacaan Al-Qur'an yang menuntut kepatuhan mutlak terhadap kaidah tajwid yang kompleks. Sedangkan, dalam konteks pembelajaran doa harian, paradigma evaluasinya berbeda. Fokus utama pelafalan doa tidak terletak pada kesempurnaan hukum bacaan seperti *ikhfa* atau *idgham*, melainkan pada pencegahan *lahn jali* yang dapat mengubah makna doa itu sendiri, seperti kesalahan pada pengucapan huruf (*makhraj*), harakat, dan panjang bacaan (*mad*).

Penelitian terdahulu oleh (Sujjada, Insany dan Nugraha, 2024) telah berhasil mengimplementasikan Google Speech API sebagai instrumen evaluasi hafalan Al-Qur'an pada sistem operasi Android. Pendekatan ini terbukti efisien secara praktis karena kemudahannya dalam pengintegrasian model secara *real-time*. Namun, kelemahan penggunaan API komersial tersebut terletak pada sifatnya yang *closed-source*, sehingga pengembang tidak memiliki akses untuk memodifikasi dan melatih ulang model secara mandiri.

Keterbatasan pada model bersifat *closed-source* mendorong penelitian lain untuk membangun arsitektur secara mandiri. Pendekatan menggunakan arsitektur *deep learning* seperti penggunaan arsitektur *End-to-End CNN-BiGRU* yang dikombinasikan dengan *Connectionist Temporal Classification* (CTC) oleh (Al Harere dan Al Jallad, 2023) menunjukkan tingkat akurasi yang memuaskan dengan WER sebesar 8,34%, tetapi model tersebut harus dibangun dari awal dengan beban komputasi yang berat dan belum dioptimalkan untuk karakteristik suara anak. Di sisi lain, penelitian oleh (Khairani *et al.*, 2024) memanfaatkan model *transformer* terkini, dengan mengimplementasikan model dasar (*zero-shot*) dari Whisper OpenAI untuk kebutuhan edukasi kosakata bahasa Arab. Penelitian ini membuktikan efektivitas model Whisper dalam mengenali pelafalan bahasa Arab untuk penutur dewasa. Namun, ketiadaan proses *fine-tuning* membuatnya rentan akan bias *auto-correct* ketika dihadapkan pada pelafalan anak.

Beberapa pendekatan komputasi lain lebih berfokus pada klasifikasi bacaan tanpa proses transkripsi. Dalam penelitiannya, (Ar Rosyid, Abdullah dan Alqahtani, 2024) menggunakan

ekstraksi MFCC dan algoritma AlexNet (CNN) untuk mengklasifikasikan hafalan Surah Al-Ikhlas dengan akurasi 93,89%. Serupa dengan hal tersebut, (Daoud *et al.*, 2025) menggunakan arsitektur *Ensemble* (CNN, LSTM, GRU) dengan akurasi 88,57% untuk mendeteksi pelanggaran hukum Tajwid (*ikhfa* dan *ghunnah*). Namun, kedua penelitian tersebut memiliki keluaran sistem berupa label mutu bacaan saja tanpa menghasilkan sekuens teks utuh dan umpan balik berupa detail letak kesalahan pelafalan.

Sebuah tinjauan literatur komprehensif oleh (Al-Kharusi *et al.*, 2025) memberikan catatan kritis pada sistem komputasi ASR Al-Qur'an. Tinjauan tersebut mengungkap adanya bias demografi yang signifikan, di mana sistem ASR saat ini sering kali gagal memproses suara anak-anak. Selain itu, minimnya pengujian validitas algoritma yang melibatkan kesepakatan pakar secara langsung memberi tanda tanya pada keandalan kinerja sistem ASR itu sendiri.

Tabel 2.1 Perbandingan penelitian terdahulu

Peneliti (Tahun)	Objek Penelitian	Model/Algoritma	Hasil/Kelebihan	Celah Penelitian
(Al Harere dan Al Jallad, 2023)	Transkripsi bacaan Al-Qur'an	<i>End-to-End</i> CNN-BiGRU & CTC	Mencapai WER 8,34% dan CER 2,42%	Model dibangun dari nol (<i>from scratch</i>) dengan arsitektur yang berat, serta belum difokuskan untuk karakteristik suara anak
(Sujjada, Insany dan Nugraha, 2024)	Hafalan Al-Qur'an	Google Speech API	Memberikan umpan balik hafalan <i>real-time</i> di Android	Menggunakan model <i>cloud closed-source</i> yang tidak dapat diadaptasi khusus untuk rentang frekuensi suara anak
(Khairani <i>et al.</i> , 2024)	Edukasi kosakata bahasa Arab	Whisper (<i>Zero-Shot</i>)	Sangat efektif mengenali pelafalan Arab dengan dialek standar dewasa	Belum di- <i>fine-tune</i> untuk frekuensi suara anak-anak, sehingga rentan mengalami bias <i>auto-correct</i>
(Ar Rosyid, Abdullah dan Alqahtani, 2024)	Hafalan Surah Al-Ikhlas	AlexNet (CNN)	Akurasi klasifikasi mencapai 93,89%	Hanya mengklasifikasikan kualitas bacaan tanpa memberikan <i>feedback</i> letak kesalahan secara spesifik
(Daoud <i>et al.</i> , 2025)	Evaluasi tajwid (<i>lahn khafi</i>)	<i>Ensemble</i> (CNN, LSTM, GRU)	Akurasi klasifikasi 88,57% untuk mendeteksi <i>ikhfa</i> dan <i>ghunnah</i>	Fokus pada klasifikasi hukum tajwid murni tanpa melakukan transkripsi teks
(Al-Kharusi <i>et al.</i> , 2025)	Kritik evaluasi terhadap AI Al-Qur'an	<i>Systematic Literature Review</i>	Mengungkap kelemahan model ASR terkait bias <i>gender</i> /usia dan tidak adanya validasi ahli	Menjadi rujukan urgensi validasi pakar menggunakan uji kesepakatan (<i>Cohen's Kappa</i>) pada sistem ASR

Berdasarkan Tabel 2.1 di atas, terlihat jelas bahwa upaya untuk membangun sistem koreksi pelafalan bahasa Arab telah banyak dikembangkan, tetapi masih terdapat celah penelitian yang belum terselesaikan. Maka dari itu, untuk mengisi keterbatasan dari penelitian-penelitian terdahulu, peneliti mengusulkan untuk membuat sistem koreksi pelafalan doa dengan menggabungkan model Whisper yang secara khusus di-*fine-tune* menggunakan korpus suara anak usia sekolah dasar untuk mengatasi bias suara anak dengan algoritma *Levenshtein Distance*. Model ASR ini akan diposisikan sebagai ‘pendengar virtual’ yang menghasilkan transkripsi otomatis dari pelafalan doa oleh anak, sementara algoritma *Levenshtein Distance* akan dimodifikasi untuk dapat mendeteksi letak kesalahan pelafalan pada aspek *lahn jali*. Guna memastikan keandalan dan integritas sistem, kinerja algoritma akan dievaluasi dan divalidasi keakuratannya secara langsung melalui uji kesepakatan pakar oleh pengajar TPA atau *ustadz*. Untuk menjembatani kesenjangan dalam aspek penerapan, peneliti juga mengimplementasikan gabungan dari model ASR dan algoritma tersebut ke dalam purwarupa aplikasi *mobile* sebagai instrumen pra-evaluasi mandiri yang belum pernah diusulkan pada literatur sebelumnya, sehingga anak dapat belajar, memahami, dan memperbaiki letak kesalahannya sebelum menguji bacaannya kepada pengajar.

2.2 Landasan Teori

Bagian ini akan menguraikan berbagai konsep dasar dan landasan teoritis yang menjadi pondasi utama dalam perancangan sistem dalam penelitian ini. Landasan teori berperan penting untuk memperkaya pemahaman terhadap konsep-konsep yang terlibat dalam penelitian. Pembahasan mencakup prinsip kerja pengenalan ucapan otomatis, karakteristik suara anak, arsitektur model Whisper, konsep *fine-tuning* model, konsep *lahn jali* sebagai parameter evaluasi, algoritma yang digunakan dalam penelitian, hingga metrik evaluasi sistem.

2.2.1 Pengenalan Ucapan Otomatis (*Automatic Speech Recognition*)

Automatic Speech Recognition (ASR) atau pengenalan ucapan otomatis merupakan salah satu cabang kecerdasan buatan dan pemrosesan bahasa alami yang memungkinkan sistem komputer untuk menerima, mengenali, dan menerjemahkan sinyal suara dari ucapan manusia menjadi representasi teks secara otomatis melalui pencocokan pola sinyal dan representasi numerik (Isa, 2019; Al-Fadhli, Al-Harbi dan Cherif, 2023). Teknologi ini menjadi jembatan komunikasi yang efisien antara manusia dan mesin. Secara arsitektural, sistem ASR modern umumnya terdiri dari beberapa komponen penting dan krusial: modul pra-pemrosesan untuk

membersihkan *noise*, modul ekstraksi fitur untuk mengambil karakteristik sinyal, model klasifikasi dan model bahasa untuk memprediksi teks keluaran sesuai konteks (Malik *et al.*, 2021).

Di sisi lain, secara fundamental, sistem ASR bekerja dengan mengenali pola suara yang masuk, ekstraksi fitur, dan mencocokkannya dengan model bahasa untuk menghasilkan representasi teks (Graves, Mohamed dan Hinton, 2012). Sinyal suara analog yang masuk melalui mikrofon akan didigitalisasi melalui proses *sampling* dan kuantisasi. Setelah itu, sistem akan memecah sinyal ke dalam bingkai-bingkai waktu yang sangat pendek untuk mengekstraksi fitur spektralnya, seperti *Mel-Frequency Cepstral Coefficients* (MFCC) maupun *Log-Mel Spectrogram*, yang merepresentasikan pola unik dari setiap fonem yang diucapkan. Fitur-fitur tersebut kemudian dipetakan ke dalam model akustik dan bahasa untuk memberikan keluaran berupa rangkaian kata dalam bentuk teks (Daoud *et al.*, 2025).

Dalam konteks pengenalan bahasa Arab, komputasi ASR menghadapi tantangan tersendiri yang lebih kompleks dibandingkan bahasa Latin. Hal ini disebabkan oleh karakteristik bahasa Arab yang kaya, hal-hal krusial seperti panjang pendeknya harakat, kompleksitas struktur bahasa, serta fonem yang unik sangat memengaruhi makna kata (Talafha, Waheed dan Abdul-Mageed, 2023). Selain itu, suara anak-anak ikut memberikan tantangan tambahan karena karakteristik vokal yang dimiliki belum stabil, variasi kecepatan bicara, dan kemungkinan *noise* pada latar suara. Oleh karena itu, dalam perkembangannya, arsitektur ASR mengalami evolusi yang signifikan. Pendekatan ASR modern berbasis *deep learning* perlahan menggantikan metode konvensional berbasis *Hidden Markov Model* (HMM). Hal ini disebabkan oleh keunggulannya dalam menangani variasi suara dan kompleksitas bahasa. Secara ringkas, perbandingan komparatif antar arsitektur model ASR yang sering digunakan dalam literatur pemrosesan bahasa alami dapat dilihat pada Tabel 2.2.

Tabel 2.2 Perbandingan arsitektur dan implementasi model ASR

Model ASR	Karakteristik Arsitektur Model	Kelebihan	Keterbatasan
Google Speech API	Model <i>cloud-based/black-box</i>	1. Mudah diintegrasikan ke <i>mobile</i> (Android) (Akbar, Husodo dan Zubaidi, 2019); 2. Komputasi ringan di sisi pengguna; 3. Mendukung bahasa Arab (Assisi <i>et al.</i>, 2022)	1. Sifatnya <i>online</i> dan <i>closed-source</i> (Ar Rosyid, Abdullah dan Alqahtani, 2024); 2. Tidak mendukung <i>fine-tuning</i> secara mandiri (Akbar, Husodo dan Zubaidi, 2019; Assisi <i>et al.</i>, 2022)

Model ASR	Karakteristik Arsitektur Model	Kelebihan	Keterbatasan
CNN	Pemodelan akustik konvensional berbasis segmentasi pola audio	1. Ringan untuk segmentasi fonem (Hadiyansah dan Andamira, 2023); 2. Cocok untuk sistem berskala kecil (<i>edge devices</i>) (Ar Rosyid, Abdullah dan Alqahtani, 2024)	1. Kurang optimal dalam menangkap konteks kalimat panjang (Hadiyansah dan Andamira, 2023); 2. Rentan <i>error</i> pada transkripsi kontinu
Wav2Vec 2.0	Model <i>self-supervised learning</i> berbasis representasi audio langsung	1. Akurasi sangat tinggi (Baevski <i>et al.</i>, 2020; Mohamed dan Aly, 2021); 2. Representasi fonetik kuat; 3. Mendukung <i>fine-tuning</i> (Mohamed dan Aly, 2021); 4. Dapat diimplementasikan pada <i>edge devices</i> secara <i>offline</i> (Mohamed dan Aly, 2021)	1. Beban komputasi sangat berat (Baevski <i>et al.</i>, 2020); 2. Butuh <i>resource</i> memori besar pada <i>edge devices</i>
End-to-End CNN-BiGRU	<i>Encoder-decoder</i> berbasis karakter dengan fungsi <i>loss Connectionist Temporal Classification</i>	1. Meniadakan kebutuhan penyesuaian manual antara audio dan teks (Al Harere dan Al Jallad, 2023)	1. Membutuhkan pelatihan dari nol dengan arsitektur yang kompleks (Al Harere dan Al Jallad, 2023)
Whisper (OpenAI)	Arsitektur <i>transformer encoder-decoder</i> berbasis <i>weakly supervised learning</i>	1. Model <i>open-source</i> 2. Mendukung <i>fine-tuning</i> (Khairani <i>et al.</i>, 2024; Timmel <i>et al.</i>, 2025); 3. Dapat diimplementasikan pada <i>edge devices</i> secara <i>offline</i> (Timmel <i>et al.</i>, 2025); 4. Sangat tangguh terhadap <i>noise</i> (Radford <i>et al.</i>, 2023); 5. Multibahasa dan mendukung bahasa arab (Talafha, Waheed dan Abdul-Mageed, 2023; Khairani <i>et al.</i>, 2024); 6. Keluaran berupa sekuens teks utuh	1. Performa <i>zero-shot</i> (bawaan) bisa menurun pada dialek spesifik atau distribusi frekuensi di luar data latih tanpa <i>fine-tuning</i> (Talafha, Waheed dan Abdul-Mageed, 2023; Timmel <i>et al.</i>, 2025)

Berdasarkan komparasi arsitektur model pada Tabel 2.2, model Whisper menjadi pilihan utama sebagai model dasar sistem ASR pada penelitian ini. Pemilihan ini didasarkan pada karakteristik struktural Whisper yang bersifat *open-source*, fleksibilitasnya dalam mendukung proses *fine-tuning*, serta kemampuannya beroperasi pada *edge devices* secara *offline*. Selain itu, keluaran Whisper yang berupa sekuens teks sangat sesuai dengan kebutuhan sistem evaluasi

pelafalan dalam proses analisis menggunakan algoritma *Levenshtein Distance*. Model lain seperti Google Speech API meskipun mudah diintegrasikan dan ringan di sisi pengguna, model ini tidak mendukung fleksibilitas pelatihan ulang dan kemandirian sistem yang diharapkan dapat berjalan secara *offline* pada aplikasi. CNN meski ringan, model ini terbatas pada transkripsi pendek dan tidak memberi dukungan pada transkripsi panjang. Model klasifikasi audio murni seperti AlexNet (CNN) pun tidak relevan untuk studi kasus deteksi letak kesalahan *lahn jali* karena model hanya menghasilkan label klasifikasi tanpa transkripsi teks (Ar Rosyid, Abdullah dan Alqahtani, 2024). Terakhir, model Wav2Vec 2.0 walau memiliki *pretraining* yang kuat, model ini masih terlalu berat untuk implementasi pada perangkat yang akan dirancang saat ini. Sehingga penelitian ini menetapkan Whisper sebagai model dasar dalam pengembangan sistem koreksi. Namun, mengingat model Whisper yang memiliki kerentanan performa saat dihadapkan pada frekuensi suara anak, modifikasi dan penyesuaian melalui *fine-tuning* spesifik tetap harus dilakukan.

2.2.2 Karakteristik Akustik Suara Anak

Secara biologis dan kognitif, karakteristik suara anak memiliki perbedaan mendasar dengan orang dewasa. Dari segi fisik, pita suara pada anak-anak memiliki ukuran yang lebih kecil dan tipis, begitu pula dengan saluran vokalnya yang lebih pendek. Perbedaan anatomi ini menyebabkan frekuensi dasar (F_0) atau *pitch* suara yang dihasilkan oleh anak jauh lebih tinggi dan melengking ketika mereka bersuara (Yeung, Fan dan Alwan, 2021). Lonjakan drastis pada nilai f_0 ini turut mendistorsi frekuensi suara, sehingga pola gelombang spektral yang diciptakan sangat fluktuatif (Bhardwaj *et al.*, 2022). Selain faktor fisik, anak-anak juga memiliki kemampuan motorik dalam berartikulasi yang belum matang. Hal ini menyebabkan tingginya variasi pengucapan, di mana seorang anak bisa mengucapkan kata yang sama dengan nada dan durasi yang berbeda pada waktu yang berlainan. Ketidakmatangan artikulasi ini juga menyebabkan tingginya tingkat kesalahan fonologis dalam pelafalan kata, seperti pengucapan yang terbata-bata, penghilangan huruf atau suku kata tertentu, serta ketidaktepatan pada titik pengucapan (Horii *et al.*, 2025).

2.2.3 Model Whisper OpenAI

Definisi dan Konsep Dasar Whisper

Whisper merupakan model ASR *open-source* dari OpenAI yang dibangun di atas arsitektur *transformer encoder-decoder* (Radford *et al.*, 2023). Model ini dilatih menggunakan

pendekatan *weak supervision* pada *dataset* masif berdurasi lebih dari 680.000 jam yang mencakup data audio multibahasa yang dikumpulkan secara acak dari internet, di mana 117.000 jam di antaranya merupakan data non-Inggris, sehingga mampu mendukung pengenalan ucapan dari berbagai bahasa dengan cukup akurat (Radford *et al.*, 2023; Talafha, Waheed dan Abdul-Mageed, 2023). Skala pelatihan yang masif inilah yang memungkinkan model secara alami dapat mengenali berbagai anomali akustik, aksen, dan *noise* pada latar belakang suara tanpa memerlukan bantuan anotasi manusia. Dengan konsep ini, kapabilitas model dasar Whisper (*zero-shot*) memiliki kemampuan superior dalam menggeneralisasi pasangan bahasa yang belum pernah dilihat sebelumnya tanpa perlu penyetelan tambahan (Saraf *et al.*, 2025).

Dalam ekosistem peluncurannya, Whisper menyediakan beberapa varian ukuran model dengan enam varian skala utama. Variasi ukuran ini dibedakan berdasarkan kapasitas parameter komputasi yang dapat digunakan dalam setiap modelnya. Keenam varian tersebut dapat dilihat detail spesifikasinya pada Tabel 2.3.

Tabel 2.3 Spesifikasi varian model Whisper OpenAI

Varian Model	Jumlah Parameter	Model Khusus Bahasa Inggris	Model Multibahasa	Kebutuhan VRAM	Kecepatan Relatif
<i>Tiny</i>	39 M	<i>tiny.en</i>	<i>tiny</i>	~1 GB	~10x
<i>Base</i>	74 M	<i>base.en</i>	<i>base</i>	~1 GB	~7x
<i>Small</i>	244 M	<i>small.en</i>	<i>small</i>	~2 GB	~4x
<i>Medium</i>	769 M	<i>medium.en</i>	<i>medium</i>	~5 GB	~2x
<i>Large</i>	1550 M	-	<i>large</i>	~10 GB	1x
<i>Turbo</i>	809 M	-	<i>turbo</i>	~6 GB	~8x

Berdasarkan Tabel 2.3, setiap varian yang dimiliki oleh model Whisper OpenAI menawarkan metrik *trade-off* komputasional yang berbeda. Semakin besar ukuran model, maka akurasi dasar yang dihasilkan juga akan semakin tinggi. Namun, hal ini berdampak langsung pada peningkatan kebutuhan memori dan penurunan kecepatan inferensi yang dimiliki oleh setiap model.

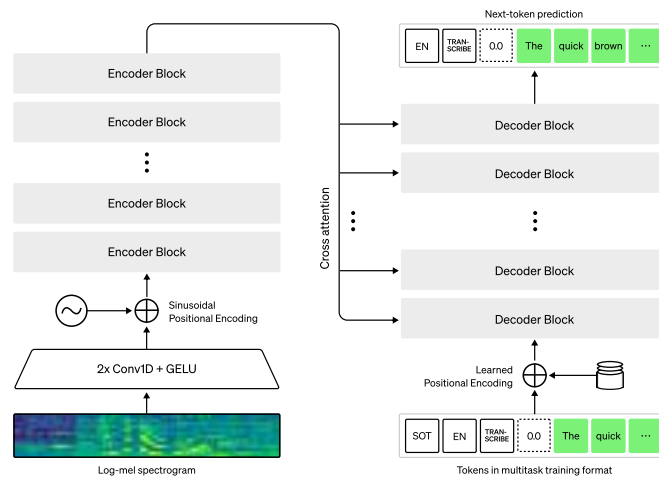
Dalam perancangan sistem ASR interaktif, pemilihan skala model ASR ini bergantung pada batasan komputasi dan target latensi dari sistem yang akan dibangun. Sebagaimana komparasi yang telah disajikan dalam Tabel 2.3, model dengan skala besar (seperti varian model *large*) memberikan akurasi dasar yang superior, tetapi model ini menuntut kapasitas VRAM yang sangat tinggi dan rentan memicu *bottleneck* karena waktu inferensinya yang terbilang paling rendah dibandingkan model lainnya. Sebaliknya, model dengan skala kecil

(seperti varian *tiny* dan *base*) menawarkan tingkat efisiensi memori dan kecepatan inferensi yang sangat tinggi, meskipun memiliki keterbatasan dalam memetakan kompleksitas fonetik secara bawaan. Meski demikian, berbagai literatur menunjukkan bahwa kelemahan akurasi pada arsitektur kecil ini dapat dikompensasi secara optimal melalui teknik *fine-tuning* menggunakan domain data yang spesifik. Pemahaman mengenai *trade-off* antara kecepatan komputasi dan akurasi bawaan ini menjadi landasan dasar yang sangat penting dalam menentukan spesifikasi model yang paling sesuai untuk dikembangkan dalam sistem yang dibangun.

Tahap Pra-Pemrosesan Audio pada Whisper

Mekanisme kerja Whisper diawali dengan proses segmentasi audio dengan mengekstraksi fitur akustik pada suara atau audio yang masuk. Pada tahap ini, Whisper memproses sinyal audio yang masuk sebagai *input* dengan *me-resample* frekuensinya menjadi 16.000 Hz. Sistem kemudian memecah aliran audio tersebut menjadi bagian berdurasi 30 detik. Dari potongan audio inilah, sistem kemudian mengonversikannya menjadi visual *log-Mel spectrogram*, yaitu representasi waktu frekuensi dua dimensi yang umum digunakan dalam pemrosesan ucapan (Saraf *et al.*, 2025). Representasi sinyal tersebut selanjutnya dikalkulasikan dengan 80 kanal (*channels*) dengan menggunakan pergeseran waktu (*stride*) sebesar 10 milidetik dan lebar jendela 25 milidetik (Radford *et al.*, 2023). Matriks *spectrogram* kemudian dinormalisasi dengan penskalaan mendekati mean nol (Salehi *et al.*, 2025; Saraf *et al.*, 2025). Hasil akhir dari tahap inilah yang akan menjadi *input* data murni yang akan dimasukkan dalam *encoder* pada proses lebih lanjut dalam arsitektur inti model Whisper.

Arsitektur Model Whisper



Gambar 2.1 Arsitektur Model Whisper

Sumber: (Open AI, 2022)

Setelah sinyal audio ditransformasikan menjadi *log-Mel spectrogram* ternormalisasi, matriks data tersebut dimasukkan ke dalam jaringan inti pemrosesan Whisper. Seperti yang disajikan dalam Gambar 2.1, arsitektur yang dimiliki oleh Whisper menggunakan pendekatan *end-to-end* yang diimplementasikan sebagai *transformer* standar yang terdiri dari blok-blok *encoder* dan *decoder* (Open AI, 2022). Dalam arsitektur Whisper, blok *encoder* bertugas untuk memproses seluruh representasi *spectrogram* secara paralel menggunakan mekanisme *multi-head self-attention*, yang memungkinkannya untuk menangkap fitur akustik dan *linguistik* dari awal hingga akhir jendela audio (Saraf *et al.*, 2025). Representasi laten dari sinyal ucapan yang berisi informasi temporal (urutan waktu pengucapan) dan kontekstual yang dihasilkan oleh *encoder* ini kemudian diteruskan ke blok *decoder*. Blok *decoder* pada Whisper beroperasi secara autoregresif untuk menghasilkan keluaran tekstual *token* demi *token*. Dengan menggunakan mekanisme *cross-attention*, *decoder* memproses keluaran yang didapat dari *encoder* dengan merujuk secara bersamaan pada representasi audio dan riwayat *token* yang telah dihasilkan sebelumnya (Radford *et al.*, 2023; Saraf *et al.*, 2025). Dengan demikian, representasi fitur *log-Mel spectrogram* yang masuk diproses oleh blok *encoder* untuk memahami konten akustik, lalu ditranskripsikan menjadi teks secara utuh oleh blok *decoder*.

Mekanisme Keluaran Whisper

Seperti yang telah dijelaskan sebelumnya, model Whisper yang mendapatkan masukan berupa representasi *log-Mel spectrogram*, memulai prosesnya dengan memetakan fitur akustik,

mengubahnya dalam bentuk representasi laten, kemudian mentranskripsikannya ke dalam bentuk *token* teks. *Token* ini kemudian di-*decode* ke dalam bentuk *string* menggunakan *tokenizer* khusus multilingual (Saraf *et al.*, 2025). Selain dapat mentranskripsikan suara ke dalam bentuk teks, blok *decoder* pada Whisper dirancang untuk dapat melakukan berbagai tugas spesifik lainnya dengan memanfaatkan serangkaian *token* khusus. *Token* khusus ini berfungsi sebagai tombol kendali yang mengarahkan model untuk mengeksekusi berbagai tugas dari satu masukan audio yang sama. Token-token ini dimasukkan di sisi *decoder* sebagai bagian dari urutan *input* awal, lalu model merespons dengan perilaku yang berbeda sesuai kombinasi token tersebut (Saraf *et al.*, 2025). Beberapa tugas spesifik yang dikendalikan oleh token khusus Whisper diantaranya transkripsi multibahasa, penerjemahan, deteksi aktivitas suara, identifikasi bahasa, segmentasi audio, pembuatan penanda waktu, hingga menentukan mode operasi antara transkripsi murni atau penerjemahan (Radford *et al.*, 2023; Salehi *et al.*, 2025).

Dalam konteks sistem evaluasi pelafalan doa harian, mekanisme keluaran Whisper akan dikonfigurasi secara spesifik menggunakan token kendali untuk tugas transkripsi teks berbahasa Arab tanpa melalui proses penerjemahan. Sekuens teks karakter Arab yang dihasilkan oleh *decoder* inilah yang nantinya akan diekstrak menjadi untaian data teks. Data tersebut selanjutnya menjadi parameter masukan utama bagi algoritma *levenshtein distance* dalam mengidentifikasi letak kesalahan pelafalan huruf (*lahn jali*) secara presisi.

Kelebihan dan Keterbatasan Whisper

Berdasarkan tinjauan dari berbagai literatur terkini, implementasi arsitektur Whisper memiliki sejumlah keunggulan komputasional maupun keterbatasan operasional. Beberapa keunggulan komputasional yang dimiliki Whisper dapat dijabarkan sebagai berikut:

- a. Ketahanan terhadap *noise* atau gangguan, Whisper dilatih pada 680.000 jam data audio yang sangat beragam, sehingga memiliki ketahanan terhadap suara latar, variasi aksen, dan kualitas audio yang buruk (Radford *et al.*, 2023).
- b. Kemampuan *zero-shot learning*, arsitektur model Whisper mampu mengenali dan mentranskripsikan bahasa dan dialek yang belum pernah dilihat sebelumnya dalam pelatihan tanpa memerlukan penyesuaian tambahan (Saraf *et al.*, 2025).
- c. Adaptabilitas melalui *fine-tuning* spesifik, Whisper memungkinkan dilakukannya proses *fine-tuning* untuk menyesuaikan model dengan domain data tertentu atau karakteristik

demografi khusus yang tidak terwakili dengan baik pada model dasar (Timmel *et al.*, 2025).

- d. Efisiensi latensi yang tinggi, dibandingkan dengan model ASR konvensional seperti Wav2Vec 2.0 dan DeepSpeech, arsitektur *transformer encoder-decoder* yang dimiliki Whisper terbukti mampu mereduksi latensi pemrosesan hingga 80-100% (Salehi *et al.*, 2025).
- e. Pemrosesan *multi task*, Whisper dapat melakukan identifikasi bahasa, transkripsi, hingga penerjemahan melalui penggunaan token khusus (Radford *et al.*, 2023).

Meskipun memiliki keunggulan komputasional yang beragam, Whisper juga dihadapkan pada beberapa keterbatasan dalam hal operasional yang perlu diperhatikan oleh pengembang. Hal ini dikarenakan Whisper masih berada dalam tahap pengembangan. Beberapa keterbatasan tersebut, dapat dijabarkan sebagai berikut:

- a. Komputasi berat untuk model besar, Whisper merupakan model skala besar yang memerlukan sumber daya komputasi yang tinggi, sehingga implementasi pada perangkat dengan sumber daya terbatas perlu memperhatikan ukuran model yang digunakan (Saraf *et al.*, 2025).
- b. Risiko degradasi performa pada bahasa *low-resource*, pada kasus bahasa dengan sumber data yang sangat minim dalam set pelatihan, kualitas transkripsi yang dihasilkan oleh Whisper dapat menurun (Radford *et al.*, 2023).
- c. Masalah halusinasi dan *auto-correct*, model bahasa bawaan Whisper cenderung akan melakukan koreksi otomatis pada pola kata atau kalimat yang umum serta menghasilkan teks kata atau kalimat yang tidak ada dalam audio. Hal ini dapat disebabkan oleh beberapa faktor seperti *overfitting* pada fase pelatihan maupun kompleksitas audio yang diterima (Graave *et al.*, 2024; Koenecke *et al.*, 2024).

2.2.4 Fine-Tuning pada Model ASR

Model ASR yang dilatih pada *dataset* umum sering kali mengalami penurunan performa/kinerja ketika diuji pada domain yang berbeda, kondisi ini dikenal sebagai *domain mismatch* (Mensah *et al.*, 2025; Raissi, Rossenbach dan Schlüter, 2025). Selain itu, fenomena *language model bias* atau kecenderungan sistem untuk melakukan koreksi secara otomatis terhadap input suara yang salah (Radford *et al.*, 2023), turut menjadi salah satu tantangan terbesar dalam evaluasi pelafalan. Dalam konteks pengenalan ucapan anak, ketidaksesuaian ini

terjadi karena model yang general biasanya dilatih menggunakan suara orang dewasa dengan karakteristik suaranya yang stabil. Sedangkan suara anak-anak memiliki rentang frekuensi yang sangat beragam dan cenderung lebih tinggi dengan variasi kejelasan artikulasi dan pelafalan (Yeung, Fan dan Alwan, 2021; Horii *et al.*, 2025). Karakteristik akustik dan fonologis ini menyulitkan model ASR standar untuk melakukan transkripsi secara akurat (Graave *et al.*, 2024; Singh, Sahidullah dan Kinnunen, 2025).

Untuk mengatasi bias tersebut sekaligus menanggulangi masalah *acoustic mismatch* pada suara anak-anak, teknik adaptasi domain melalui proses pelatihan ulang model atau *fine-tuning* berbasis *transfer learning* pada model menjadi solusi yang efektif (Timmel *et al.*, 2025). Secara konseptual, *fine-tuning* merupakan proses melatih ulang sebuah model yang sudah memiliki pengetahuan awal (*pre-trained*) menggunakan set data spesifik yang lebih kecil dan berfokus pada domain target tertentu. Dalam proses *fine-tuning* model ASR, tidak semua lapisan jaringan saraf dilatih kembali. Pendekatan dilakukan dengan membekukan bobot komputasi pada beberapa lapisan ekstraksi fitur di blok *encoder*, lalu pada lapisan *decoder* bobot diperbarui. Studi oleh (Khairani *et al.*, 2024) menunjukkan bahwa penerapan *transfer learning* dari model yang telah mempelajari bahasa terkait seperti Bahasa Arab ke data anak-anak akan meningkatkan akurasi pengenalan kata dibandingkan melakukan pelatihan dari awal.

2.2.5 Konsep *Lahn Jali* sebagai Parameter Evaluasi

Dalam ilmu tajwid, *lahn jali* didefinisikan sebagai kesalahan nyata yang terjadi pada pelafalan yang dapat mengubah makna dan merusak kaidah bahasa Arab (Marzuki dan Ummah, 2020). Penelitian ini, menerapkan standar *lahn jali* sebagai parameter utama dalam proses evaluasi bacaan anak dengan membandingkan teks transkripsi dengan teks bacaan doa aslinya. Bentuk kesalahan yang menjadi cakupan *lahn jali*, yang kemudian diimplementasikan dan menjadi fokus deteksi dalam sistem koreksi pelafalan ini sebagai parameter evaluasi, meliputi:

- a. Perubahan huruf, yakni mengganti suatu huruf dengan huruf lainnya yang memiliki titik ucap (*makhraj*) yang berdekatan atau cara pelafalan yang mirip, tetapi identitas fonetiknya berbeda. Misal mengganti huruf *Syin* dengan *Sin* atau *Qaf* dengan *Kaf*.
- b. Perubahan harakat, yakni mengganti bunyi vokal huruf (*fathah*, *kasrah*, dan *dhammah*) yang dapat mengubah fungsi kata dalam kalimat.

- c. Kesalahan pada panjang pendek bacaan (*mad*), yakni menghilangkan atau menambah durasi pada pelafalan huruf *mad* yang seharusnya dibaca panjang atau pendek secara harfiah.

Dalam konteks komputasi, konsep lahn jali ini ditransformasikan sebagai bobot kesalahan yang akan dideteksi oleh algoritma *levenshtein distance* sebagai bentuk ketidaksesuaian karakter yang harus dikoreksi dan diberi umpan balik berupa peringatan agar anak dapat memperbaiki bacaan menjadi lebih sempurna.

2.2.6 Algoritma *Levenshtein Distance*

Levenshtein distance atau yang sering disebut sebagai edit distance, merupakan algoritma matematis yang digunakan untuk mengukur perbedaan antara dua sekuens teks (Sadiq dan Yousaf, 2020). Algoritma ini menghitung jumlah minimum operasi pengeditan yang diperlukan untuk mengubah teks referensi menjadi teks hipotesis (teks transkripsi yang dihasilkan oleh Whisper). Sehingga, algoritma ini bekerja dengan cara membandingkan kedua *string* tersebut.

Dalam proses komputasi, algoritma ini mendeteksi tiga jenis operasi pengeditan, yakni penyisipan, penghapusan, serta penggantian (Sadiq dan Yousaf, 2020). Penyisipan atau *insertion* didefinisikan sebagai penambahan karakter pada teks hipotesis yang tidak ada dalam teks referensi. Sementara penghapusan atau *deletion* dimaksudkan pada kondisi di mana karakter pada teks referensi hilang atau tidak muncul pada teks hipotesis. Sedangkan penggantian atau (*substitution*) merupakan bentuk penggantian suatu karakter pada teks referensi dengan karakter lain yang berbeda pada teks hipotesis.

Dalam konteks evaluasi pelafalan doa harian, ketiga operasi ini merepresentasikan bentuk identifikasi kesalahan pada *lahn jali*. Operasi penggantian akan digunakan untuk mengindikasikan ketidaktepatan *makharijul* huruf atau salah pelafalan huruf. Sementara operasi penghapusan dan penyisipan akan digunakan untuk mengindikasikan adanya huruf yang terlewat atau yang bertambah dari pelafalan doa oleh anak.

Untuk mengoptimalkan efisiensi komputasi, algoritma ini menggunakan metode pemrograman dinamis. Di mana sebuah matriks dua dimensi dibangun dalam sistem untuk menyimpan rute pengeditan terpendek dari awalan teks, sehingga mencegah penghitungan ulang pada pola karakter yang sama. Nilai jarak pada setiap sel matriks didapatkan melalui fungsi rekursif yang secara matematis dituliskan dalam persamaan (2.1) (Jurafsky dan Martin, 2026).

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} \end{cases} \quad (2.1)$$

2.2.7 Metrik Evaluasi Sistem

Metrik Evaluasi Model ASR

Dalam evaluasi model ASR, untuk mengukur keandalan dan kualitas model yang telah melalui proses *fine-tuning* dengan set data suara anak, algoritma *levenshtein distance* juga turut diterapkan dalam tahap ini. Metrik evaluasi standar yang digunakan yakni *Word Error Rate* (WER) dan *Character Error Rate* (CER) (Mensah *et al.*, 2025). Dalam penelitian ini, WER digunakan untuk mengukur tingkat kesalahan model pada level kata, sedangkan CER digunakan untuk mengukur tingkat kesalahan model pada level karakter.

Metrik evaluasi utama yang digunakan dalam penelitian ini yakni CER. Hal ini dikarenakan, CER akan lebih sensitif terhadap susunan atau fonem bahasa Arab yang memiliki harakat, huruf, serta variasi kecil lainnya dalam doa. Secara matematis, formulasi WER dan CER didefinisikan pada persamaan (2.2).

$$Error\ Rate = \frac{S + D + I}{N} \quad (2.2)$$

Keterangan:

- S = Jumlah substitusi (penggantian kata/karakter yang salah)
- D = Jumlah delesi (kata/karakter yang hilang atau tidak terdeteksi)
- I = Jumlah insersi (penambahan kata/karakter yang tidak seharusnya ada)
- N = Total jumlah kata/karakter pada teks referensi asli

Hasil dari kedua metrik ini menjadi indikator utama yang menentukan kualitas model dalam mengenali bacaan doa anak. Selain itu, kedua metrik ini juga digunakan untuk membandingkan performa antar percobaan, serta menjadi referensi kualitas model sebelum diintegrasikan ke aplikasi/sistem koreksi bacaan. Semakin kecil nilai persentase CER, maka model dianggap semakin presisi dalam memberikan transkripsi pelafalan doa oleh anak.

Validasi Ahli dan Reliabilitas Kesepakatan

Validasi kepakaran merupakan prosedur esensial dalam mengevaluasi sistem deteksi kesalahan pelafalan untuk memastikan akurasi komputasi sistem sejalan dengan kaidah tajwid yang benar. Untuk mengukur tingkat reliabilitas antar pengamat (*inter-rater reliability*) antara hasil deteksi otomatis sistem dengan hasil penilaian manual oleh ahli, penelitian ini menggunakan metrik statistik *Cohen's Kappa* (κ). Keunggulan utama metrik ini dibandingkan pengukuran akurasi sederhana adalah kemampuannya dalam mengeliminasi probabilitas kesepakatan yang mungkin terjadi akibat faktor kebetulan (*chance agreement*), sehingga memberikan estimasi kesesuaian yang lebih konservatif dan objektif (Gisev *et al.*, 2013; Warrens, 2015). Penggunaan *Cohen's Kappa* sebagai alat validasi terhadap keluaran model berbasis kecerdasan buatan (AI) juga telah terbukti efektif dalam standarisasi penilaian kualitas informasi yang dihasilkan oleh mesin (Sallam, Barakat dan Sallam, 2023).

Secara matematis, nilai *Cohen's Kappa* dihitung menggunakan persamaan koefisien (Cohen, 1960) yang dapat dilihat pada persamaan (2.3).

$$\kappa = \frac{p_o - p_e}{1 - p_e} \quad (2.3)$$

Keterangan:

- p_o (*Observed Agreement*) = rasio kesepakatan yang diamati
- p_e (*Expected Agreement*) = rasio kesepakatan yang diharapkan secara kebetulan berdasarkan distribusi data

Nilai κ yang dihasilkan berada pada rentang -1 hingga 1, di mana nilai 1 menunjukkan kesepakatan sempurna. Untuk menginterpretasikan tingkat keandalan sistem dalam mengevaluasi pelafalan doa anak, penelitian ini merujuk pada standar kategorisasi Landis dan Koch (1977) yang juga didukung dalam tinjauan metodologi reliabilitas modern (Gisev *et al.*, 2013) dapat dilihat dalam Tabel 2.4.

Tabel 2.4 Interpretasi nilai kappa oleh Landis dan Koch (1977)

Nilai κ	Kekuatan Kesepakatan
< 0.00	Sangat Buruk (<i>Poor</i>)
$0.00 - 0.20$	Sangat Rendah (<i>Slight</i>)
$0.21 - 0.40$	Rendah (<i>Fair</i>)
$0.41 - 0.60$	Sedang (<i>Moderate</i>)
$0.61 - 0.80$	Kuat/Baik (<i>Substantial</i>)
$0.81 - 1.00$	Hampir Sempurna (<i>Almost Perfect</i>)

Metrik Evaluasi Algoritma Koreksi

Untuk mengevaluasi kinerja dan performa algoritma deteksi dalam mengklasifikasikan bentuk kesalahan dalam pelafalan anak, hasil deteksi sistem dan penilaian pakar akan dibandingkan. Hasil perbandingan ini kemudian dipetakan ke dalam bentuk *confusion matrix*. *Confusion matrix* merupakan representasi visual yang berupa matriks dua dimensi, yang merangkum jumlah prediksi benar/salah (Algabri *et al.*, 2022). Pengklasifikasian dikategorikan ke dalam empat nilai:

- *True Positive* (TP): jumlah data yang dideteksi salah oleh sistem atau model, dan memang salah menurut referensi atau ahli.
- *True Negative* (TN): jumlah data yang dideteksi benar oleh sistem atau model, dan memang benar menurut referensi atau ahli.
- *False Positive* (FP): jumlah data yang dideteksi salah oleh sistem atau model, padahal benar menurut referensi atau ahli.
- *False Negative* (FN): jumlah data yang dideteksi benar oleh sistem atau model, padahal salah menurut referensi atau ahli.

Berdasarkan matriks tersebut, keandalan algoritma sistem akan dievaluasi menggunakan empat persamaan metrik standar, yakni *accuracy*, *precision*, *recall*, dan *F1-score*.

1. Akurasi (*Accuracy*) merupakan metrik yang merepresentasikan tingkat kesesuaian antara kelas hasil prediksi model dengan kondisi faktual yang sebenarnya (Sammud dan Webb, 2017). Secara matematis, metrik ini menghitung proporsi total prediksi yang berhasil ditebak dengan benar oleh sistem terhadap keseluruhan jumlah data yang dievaluasi. Nilai akurasi dihitung menggunakan persamaan (2.4).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.4)$$

2. Presisi (*Precision*) merupakan rasio perbandingan antara jumlah deteksi *True Positive* (TP) terhadap total keseluruhan label positif yang diprediksi oleh sistem (Sammut dan Webb, 2017). Metrik ini digunakan untuk mengevaluasi keandalan dan tingkat ketepatan model setiap kali memberikan klaim prediksi positif. Persamaan matematis untuk menghitung presisi ditunjukkan pada persamaan (2.5).

$$Precision = \frac{TP}{TP + FP} \quad (2.5)$$

3. *Recall* (*Sensitivity*) adalah rasio perbandingan antara jumlah deteksi *True Positive* (TP) terhadap keseluruhan data yang secara faktual benar-benar memiliki label kelas positif (Sammut dan Webb, 2017). Metrik ini mengukur seberapa tinggi daya model dalam mendeteksi dan menemukan seluruh target yang positif secara aktual di dalam *dataset* tanpa ada yang terlewat. Nilai *recall* didapat dari persamaan (2.6).

$$Recall = \frac{TP}{TP + FN} \quad (2.6)$$

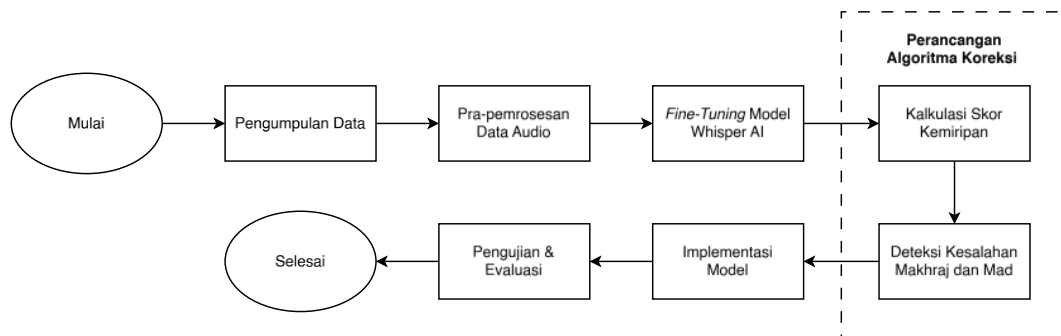
4. *F1-Score* merupakan metrik yang menghitung nilai rata-rata harmonis antara nilai presisi dan *recall*. Pendekatan ini digunakan untuk menilai titik keseimbangan komputasi antara kedua metrik tersebut (Sammut dan Webb, 2017). *F1-score* memberi gambaran mengenai seberapa tangguh model dalam menghasilkan prediksi positif yang presisi, sekaligus meminimalisir jumlah kasus positif yang gagal terdeteksi. Nilai *F1-score* dihitung dengan menggunakan persamaan (2.7).

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.7)$$

BAB III

METODOLOGI PENELITIAN

Penelitian ini dilakukan secara sistematis dengan menerapkan metode penelitian eksperimental yang bertujuan untuk mengembangkan dan mengevaluasi sistem koreksi bacaan doa anak berbasis *speech recognition*. Sistem yang dibangun memanfaatkan model *Automatic Speech Recognition* (ASR) Whisper untuk mentranskripsikan suara bacaan doa anak, yang kemudian dikolaborasikan dengan logika sistem sebagai mekanisme koreksi bacaan dalam aplikasi pembelajaran interaktif untuk membantu anak dalam menghafal doa sehari-hari. Fokus penelitian terletak pada pengembangan dan evaluasi kinerja sistem berbasis model, khususnya meningkatkan kinerja model ASR dalam mengenali bacaan doa anak melalui proses *fine-tuning* model dengan *dataset* spesifik yang dikurasi melalui beberapa sumber dan implementasi *logic layer* yang berfokus pada evaluasi bacaan pada level *makhraj* huruf dan *mad* (panjang pendeknya bacaan). Berdasarkan metode ini, proses pengembangan akan dibagi ke dalam enam tahapan proses yang berkesinambungan (lihat Gambar 3.1), mulai dari pengumpulan data, pra-pemrosesan data audio, proses *fine-tuning* model Whisper AI, perancangan algoritma koreksi, implementasi model, hingga pengujian dan evaluasi.



Gambar 3.1 Metodologi penelitian

3.1 Pengumpulan Data

Dalam penelitian pengembangan model *Automatic Speech Recognition* (ASR) untuk sistem koreksi bacaan doa anak ini, data yang dikumpulkan terbagi menjadi dua jenis. Data pertama yakni data primer berupa korpus audio atau rekaman suara bacaan doa oleh anak-anak. Kemudian, data kedua yakni data sekunder berupa label transkripsi berhuruf hijaiyah (teks Arab) dari setiap audio serta parameter *checkpoint* evaluasi bacaan untuk setiap doa yang ada.

3.1.1 Pengumpulan Korpus Audio

Data primer yang digunakan dalam penelitian ini merupakan *dataset* audio atau suara dari pelafalan beberapa doa harian yang diucapkan oleh anak-anak, yang dihimpun melalui pendekatan hibrida melalui dua sumber utama.

1. Perekaman data audio yang diambil secara langsung oleh peneliti dari tiga Taman Pendidikan Al-Qur'an (TPA) yang ada di Kabupaten Sleman (TPA Ar-Rahmah Sawitsari, TPA Al-Baitul Makmur Wonorejo, dan TPA An-Nahrawi Plosokuning) dengan subjek anak-anak pada rentang usia 6-12 tahun. Pengambilan data dilakukan dalam kondisi lingkungan semi-terkontrol menggunakan media ponsel Vivo Y12s yang terhubung dengan *microphone wireless* untuk mengurangi *noise*, dengan format hasil audio yakni M4A. Skenario ini bertujuan untuk menangkap artikulasi suara anak secara natural dengan derau suara latar ruangan yang wajar. Untuk meningkatkan jumlah sampel dan memberi variasi pengucapan, setiap anak diminta menghafal satu doa sebanyak satu hingga tiga kali.
2. Ekstraksi audio daring dari data suara publik yang tersedia pada *platform* YouTube dan *search engine* Google. Pencarian difokuskan pada konten anak dengan isi pelafalan doa harian yang telah ditentukan. Metode ini penting untuk memperkaya variasi *pitch* pita suara anak dan karakteristik kualitas mikrofon yang beragam. Data daring yang dihimpun, kemudian diseleksi dengan kriteria utama yakni konten yang disajikan memiliki suara pelafalan yang cukup jelas dan tidak didominasi oleh musik latar yang keras agar tidak merusak spektrum suara ketika pelatihan model dilakukan nantinya. Audio yang didapat dari video, di ekstrak ke format standar .wav lalu dipotong sesuai kebutuhan pelatihan.

Penggabungan dua sumber ini dilakukan untuk memberikan variasi pada data audio suara, memperkaya gaya pelafalan, serta meningkatkan *robustness* model terhadap *noise*, perbedaan karakteristik, maupun artikulasi.

Karakteristik *dataset* primer yang digunakan dalam proses pelatihan dan validasi model ASR untuk kebutuhan sistem koreksi bacaan doa anak, adalah sebagai berikut:

- *Dataset* terdiri dari 3 jenis doa, yakni doa sebelum tidur, doa bangun tidur, dan doa sebelum makan yang masing-masing terdiri dari 120 data audio dengan format M4A/WAV.
- *Dataset* lapangan diambil secara langsung di tiga TPA yang telah disebutkan dengan melibatkan total 45 anak (20 laki-laki dan 25 perempuan).

- *Dataset* memiliki *sampling rate* 48 kHz, *mono* dan *stereo*.
- Durasi rata-rata audio 7,95 detik.
- Total durasi *dataset* 47 menit 40,73 detik.

Secara keseluruhan, pendekatan hibrida ini berhasil menghimpun total 360 sampel rekaman audio mentah. Agar model ASR tidak mengalami bias saat proses pelatihan berlangsung, jumlah sampel dipastikan terdistribusi secara merata pada ketiga doa harian yang menjadi batasan penelitian. Rincian distribusi jumlah *dataset* berdasarkan jenis doa dan sumber pengambilannya terangkum dalam Tabel 3.1 .

Tabel 3.1 Rangkuman distribusi *dataset*

Jenis Doa	Jumlah Audio	Total Durasi	Rata-Rata Durasi	Jumlah Anak/Sumber
Doa sebelum tidur	120	11 menit 43,94 detik	5,87 detik	33 anak + 53 data daring
Doa bangun tidur	120	21 menit 19,72 detik	10,66 detik	7 anak + 109 data daring
Doa sebelum makan	120	14 menit 37,07 detik	7,31 detik	38 anak + 37 data daring

Seluruh korpus audio yang terangkum dalam tabel distribusi tersebut, kemudian dicatat secara terstruktur ke dalam log dokumentasi data primer audio yang juga difungsikan sebagai pangkalan data utama (*Master Dataset*). Pendokumentasian ini dilakukan untuk mempertahankan rekam jejak data lapangan maupun sumber laman data daring dari setiap data yang dikumpulkan. Informasi yang dicatat meliputi identitas data, nama *file* audio, jenis doa, transkripsi awal, durasi, sumber data, tanggal pengambilan data, identitas subjek atau penutur (ID, nama, dan usia) untuk data lapangan, hingga catatan yang berupa lokasi perekaman untuk data lapangan dan judul konten untuk data daring. Untuk menjaga privasi subjek dan mematuhi etika penelitian, atribut ‘Nama’ pada seluruh tabel yang dilampirkan dalam penelitian ini disamarkan menjadi bentuk inisial.

Tabel 3.2 menampilkan cuplikan sampel representatif dari log dokumentasi data primer mentah yang diambil secara acak untuk mewakili keragaman data pada proses penghimpunan dengan metode yang telah disebutkan.

Tabel 3.2 Cuplikan dokumentasi data primer mentah

ID Data	Nama File Audio	Jenis Doa	Teks Transkripsi	Durasi	Sumber	Tanggal Pengambilan	ID Penutur	Inisial Nama	Usia	Catatan
DST-001	doa_sebelum_tidur_001.m4a	Doa Sebelum Tidur	بِسْمِكَ اللَّهُمَّ أَحْيَا وَ بِسْمِكَ أَمُوتُ	00:00:04	Rekam Manual	13/11/2025	Anak-001	M.	11 tahun	TPA Ar-Rahmah Sawitsari
DST-066	doa_sebelum_tidur_066.wav	Doa Sebelum Tidur	بِسْمِكَ اللَّهُمَّ أَحْيَا وَ بِسْمِكَ أَمُوتُ	00:00:06	youtube.com/watch?v=AkolsTkV73c	29/04/2025	Web-001	Anak		Doa Sebelum Tidur bersama Diva Kastari Animation Official - Kastari Sentra
DBT-032	doa_bangun_tidur_032.wav	Doa Bangun Tidur	الْحَمْدُ لِلَّهِ الَّذِي أَحْيَانَا بَعْدَ مَا أَمَاتَنَا وَالْإِلَهَ الشُّعُورُ	00:00:11	youtube.com/shorts/rwyd8Pj9haQ	03/05/2025	Web-040	Anak		doa sebelum tidur dan bangun tidur - Ninda Nurfalalah
DBT-101	doa_bangun_tidur_101.wav	Doa Bangun Tidur	الْحَمْدُ لِلَّهِ الَّذِي أَحْيَانَا بَعْدَ مَا أَمَاتَنَا وَالْإِلَهَ الشُّعُورُ	00:00:08	Rekam Manual	07/12/2025	Anak-041	A.	10 tahun	TPA An-Nahrowi Plosokuning
DSM-040	doa_sebelum_makan_040.m4a	Doa Sebelum Makan	اللَّهُمَّ بَارِكْ لَنَا فِيهِ رَزَقْنَا وَكَفِّرْ أَدْبَارَنَا	00:00:04	Rekam Manual	15/11/2025	Anak-024	K.	6 tahun	TPA Al-Baitul Makmur Wonorejo
DSM-083	doa_sebelum_makan_083.wav	Doa Sebelum Makan	اللَّهُمَّ بَارِكْ لَنَا فِيهِ وَلَا تَدِينْ	00:00:11	youtube.com/shorts/de47V769Yf0	13/07/2025	Web-135	Anak		Doa sebelum makan. pinter banget sih sudah bisa sendiri tanpa di bantu - Ana Fauziah Channel

3.1.2 Pengumpulan Data Label dan *Checkpoint*

Untuk mendukung proses *fine-tuning* pada model ASR, setiap file korpus audio akan didampingi oleh data label (*ground truth*) berupa teks berbahasa Arab. Hal ini dilakukan agar model dapat menghitung nilai *loss* dan persentase kesalahan kata (WER) maupun huruf (CER) selama proses *training* dijalankan. Pengumpulan data pada tahap ini dibagi menjadi dua tahapan penting:

1. Transkripsi teks Arab atau anotasi secara manual oleh peneliti. Setiap sampel rekaman audio pada *dataset* yang telah dikumpulkan, disalin ke dalam bentuk teks berbahasa Arab menggunakan huruf *hijaiyyah* sesuai dengan kaidah ortografi dan tajwid. Sebagaimana yang telah didokumentasikan dalam kolom *Transkrip Teks Manual* pada Tabel 3.2 sebelumnya, teks ini menjadi target acuan bagi model untuk belajar dan mencocokkan pola *Log-Mel Spectrogram* suara dengan ejaan huruf yang benar pada proses *fine-tuning*.
2. Penyusunan metadata dalam format JSON untuk *checkpoint* evaluasi bacaan. Selain label pelatihan untuk model, penelitian ini juga mengumpulkan data parameter evaluasi tajwid yang sering terjadi pada anak. Setiap *checkpoint* dalam file JSON menyimpan informasi strategis untuk pengembangan *logic layer* Kotlin, yang meliputi: indeks target huruf, tipe evaluasi (*Makhrasj/Mad*), huruf penanda, serta kalimat umpan balik khusus yang akan ditampilkan pada antarmuka aplikasi sebagai media implementasi visual. Gambar 3.2 memperlihatkan contoh struktur penulisan data *checkpoint* yang digunakan sebagai parameter evaluasi pada aplikasi, yang disimpan dalam file JSON.

```
{
  "id": "doa_01",
  "imgSource": "sebelum_tidur",
  "title": "Doa Sebelum Tidur",
  "arabicText": "بِسْمِ اللَّهِ أَحْيَا وَ بِاسْمِكَ أَمُوتُ",
  "indoText": "Bismika allaahumma ahyaa wa bismika amuut(u)",
  "translate": "Dengan nama-Mu ya Allah, aku hidup dan dengan nama-Mu aku mati",
  "audioSource": "doa_sebelum_tidur",
  "checkpoints": [
    {
      "index": 3,
      "huruf": "س",
      "tipe": "MAKHRAJ",
      "feedback": "Suara desis huruf Sin (س) harus terdengar jelas ya."
    },
    {
      "index": 14,
      "huruf": "ا",
      "tipe": "MAD",

```

```

    "feedback": "Jangan lupa, bacaan 'Laa' di kata Allahumma harus
panjang 2 harakat."
  },
  {
    "index": 23,
    "huruf": "ح",
    "tipe": "MAKHRAJ",
    "feedback": "Huruf Ha (ح) harus bersih dari tengah tenggorokan."
  },
  {
    "index": 27,
    "huruf": "ا",
    "tipe": "MAD",
    "feedback": "Bacaan 'Yaa' di akhir kata Ahya harus panjang 2 harakat."
  },
  {
    "index": 46,
    "huruf": "و",
    "tipe": "MAD",
    "feedback": "Bacaan 'Muu' pada kata Amuut harus panjang 2 harakat."
  }
]
},

```

Gambar 3.2 Contoh struktur JSON *checkpoint* evaluasi doa

3.2 Pra-Pemrosesan Data Audio

Dalam ekosistem *deep learning*, khususnya pada arsitektur ASR, tahap pra-pemrosesan data diakui sebagai tahapan fundamental dan sangat krusial. Hal ini dikarenakan pada tahap pra-pemrosesan, kualitas data yang dihasilkan secara signifikan akan sangat memengaruhi tingkat akurasi akhir, ketangguhan (*robustness*) terhadap variabilitas, serta kapasitas generalisasi model saat menghadapi data baru (Chaugule dan Sankaye, 2025). Maka dari itu, sebelum korpus audio primer masuk ke dalam arsitektur model Whisper untuk tahap pelatihan, data tersebut harus melalui tahapan ini untuk memastikan bahwa seluruh data audio memiliki sinyal yang bersih dan berada dalam format yang sama dan optimal dengan *pipeline* pelatihan model ASR. Implementasi pra-pemrosesan data audio dalam penelitian ini terbagi ke dalam tiga skenario utama: pembersihan sinyal audio, pemetaan metadata, dan augmentasi data.

3.2.1 Pembersihan Sinyal Audio (*Audio Cleaning*)

Data primer asli yang dihimpun pada tahap pengumpulan data, umumnya memiliki karakteristik yang berbeda, mengingat korpus dikumpulkan melalui perekaman menggunakan perangkat yang bervariasi, format *file* audio, tingkat volume, dan kualitas yang beragam. Maka dari itu, langkah pertama yang perlu dilakukan pada fase pra-pemrosesan ini adalah melakukan pembersihan sinyal audio secara menyeluruh pada setiap rekaman mentah yang telah dikumpulkan. Tahapan ini mencakup tiga proses:

1. Penyeragaman format dan *resampling* audio. Korpus audio yang dihimpun sebelumnya memiliki format file yang berbeda yakni .m4a untuk data lapangan dan .wav untuk data daring. Oleh karenanya, pada fase ini, seluruh file audio perlu diseragamkan formatnya dengan cara mengonversikannya ke dalam format standar .wav (*Waveform Audio File Format*). Pemilihan format .wav bersifat mutlak karena format bawaan *smartphone* seperti .m4a menggunakan skema kompresi *lossy* yang rentan menghilangkan komponen frekuensi akustik minor yang terbukti dapat memicu kesalahan pengenalan fonetik pada sistem ASR. Selanjutnya, audio diseragamkan ke *sample rate* 16.000 Hz (16 kHz), diubah menjadi saluran tunggal (mono), dan dikonfigurasi ke kedalaman 16-bit agar memenuhi standar arsitektur masukan model Whisper, serta memitigasi beban komputasi dekompresi secara *on-the-fly* selama siklus pelatihan berlangsung.
2. Normalisasi audio dengan menyamakan intensitas volume dari seluruh sampel audio. Proses ini bertujuan untuk memitigasi variasi suara yang tidak relevan seperti perbedaan sensitivitas *microphone* saat perekaman.
3. Penghapusan jeda diam (*silence trimming*). Korpus audio yang dikumpulkan melalui pengambilan data lapangan, umumnya memiliki jeda diam yang panjang pada awal dan akhir pelafalan doa. Penghapusan jeda ini berfungsi untuk membuang segmen non-wicara (*non-speech segments*) yang tidak berisi informasi linguistik yang relevan. Arsitektur model Whisper memproses input audio dalam bentuk *Log-Mel Spectrogram* dengan durasi tetap, sehingga keberadaan jeda diam yang berlebih dalam audio akan membuang memori komputasi dan memicu model untuk mengalami halusinasi (Chaugule dan Sankaye, 2025). Untuk menghindari kondisi tersebut, proses penghapusan jeda ini dilakukan dengan mengeliminasi segmen sinyal akustik menggunakan algoritma deteksi batas amplitudo wicara. Fungsi akan mengekstraksi segmen audio yang berada di atas ambang batas (*threshold*) -45 desibel (dB), dengan toleransi penolakan derau untuk suara berdurasi dibawah 100 milidetik. Untuk mempertahankan integritas fonem, algoritma ini juga dikonfigurasi dengan pengaplikasian penambahan rentang waktu (*padding*) sebesar 150 milidetik pada kedua ujung segmen. Hal ini dilakukan untuk menyelamatkan sinyal akustik pada konsonan akhir audio yang perlahan menurun.

3.2.2 Pemetaan dan Pembersihan Metadata

Setelah sinyal audio dibersihkan dan distandarisasi ke dalam direktori keluaran, langkah selanjutnya yakni pemetaan dan pembersihan metadata. *Master dataset* yang telah disusun sebelumnya pada tahap pengumpulan data harus melalui proses penyaringan atribut terlebih dahulu sebelum dieksekusi nantinya oleh algoritma pembelajaran mesin. Proses ini berfungsi untuk mentransformasikan catatan log dokumentasi pengumpulan korpus audio menjadi format data terstruktur dan efisien untuk proses komputasi.

Pada tahap pembersihan data (*data cleaning*), beberapa kolom yang tidak relevan dengan kebutuhan pelatihan model dieliminasi. Atribut yang dihapus ini mencakup ID Data, Durasi, Sumber Data, Tanggal Pengambilan, Nama, Usia, dan Catatan. Hal ini dilakukan agar model berfokus pada variabel-variabel yang memiliki komputasional langsung dan tidak terbebani dengan informasi yang kurang relevan.

Setelah proses pembersihan data selesai dilakukan, metadata dipetakan kembali dengan empat atribut informasi utama, yaitu:

1. Nama *File* Audio, yakni nama unik dari berkas audio berformat `.wav` yang telah diproses sebelumnya. Nama berkas menggunakan format jenis doa dan penomoran dipisahkan dengan *underscore* (`_`) agar merepresentasikan sinyal masukan. Atribut ini menjadi rujukan jalur bagi model untuk memuat dan mengekstraksi fitur akustik data audio.
2. ID Penutur, yakni kode unik untuk setiap anak. Atribut ini penting karena identitas penutur ini digunakan sebagai dasar acuan pada algoritma pembagian *dataset* nantinya. Pemetaan identitas ini akan digunakan untuk mencegah terjadinya tumpang tindih subjek yang dapat memicu kebocoran data pada tahap *fine-tuning* model.
3. Jenis Doa, yakni jenis doa harian yang dibaca oleh penutur. Atribut ini digunakan untuk mengklasifikasikan berkas audio agar *dataset* yang dibagi ke dalam beberapa bagian memiliki proporsi yang seimbang dan representatif untuk seluruh jenis doa.
4. Teks Transkripsi, yakni data label yang merepresentasikan pelafalan anak yang telah disusun sebelumnya pada tahap pengumpulan data, yang berfungsi sebagai target acuan bagi model untuk belajar. Atribut ini juga digunakan sebagai acuan untuk metrik evaluasi model.

Selain melakukan eliminasi atribut, pada tahap ini, juga akan dilakukan normalisasi ekstensi pada atribut Nama *File* Audio. Hal ini dikarenakan seluruh sinyal audio yang memiliki dua format (`.m4a` dan `.wav`) telah dikonversi secara seragam ke dalam format `.wav` pada tahap

pembersihan sinyal audio. Maka dari itu, seluruh *string* ekstensi mentah (.m4a) pada metadata dikonversi menjadi .wav untuk memastikan sinkronisasi pemanggilan direktori data. Hasil akhir dari proses ini adalah sebuah berkas metadata terstruktur dan bersih, yang memuat relasi valid antara sinyal akustik, identitas penutur, dan teks transkripsi. Korpus terstruktur inilah yang selanjutnya akan diproses pada tahap pembagian data dan augmentasi.

3.2.3 Pembagian Dataset

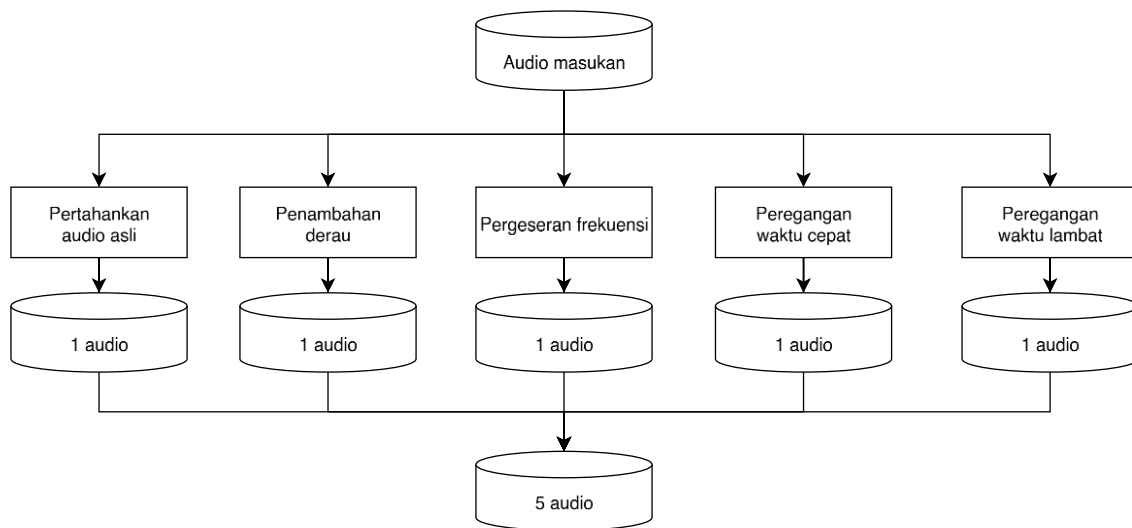
Sebelum memasuki tahap manipulasi sinyal lanjutan dan pelatihan model, *dataset* primer yang telah dibersihkan dan dipetakan metadatanya dibagi ke dalam tiga bagian yakni data latih, data validasi, dan data uji. Pembagian dilakukan dengan pendekatan *stratified sampling* berdasarkan identitas subjek penutur (*speaker-independent*), sehingga tidak terdapat data dari penutur yang sama pada lebih dari satu subset. Rasio distribusi data yang digunakan dalam penelitian ini adalah 80:10:10, merujuk pada praktik umum dalam pengembangan model *machine learning* (Draelos, 2019). Rasio ini dipilih untuk memberikan porsi data latih yang dominan agar model dapat mempelajari pola akustik secara optimal, sementara sebagian kecil data lainnya dialokasikan untuk validasi dan pengujian guna mengukur kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya. Dengan demikian, keseimbangan antara proses pembelajaran dan evaluasi performa model tetap terjaga secara objektif.

Dari total sebanyak 360 data audio, sebanyak 288 data digunakan sebagai data latih, 36 data sebagai data validasi, dan 36 data lainnya sebagai data uji. Pemisahan data pada tahap awal pra-pemrosesan ini disesuaikan dengan prinsip fundamental dalam *machine learning* sebelum menerapkan proses augmentasi (*augmentation isolation*). Langkah ini wajib dilakukan untuk mencegah terjadinya dua bentuk fenomena kebocoran data (*data leakage*). Pertama, mencegah model ‘menghafal’ karakteristik vokal unik dari anak yang sama karena sampelnya tumpang tindih antara data latih dan data uji. Kedua, memastikan model tidak mempelajari pola derau sintetik dari audio masukan yang sama akibat proses augmentasi. Dengan demikian, integritas pengujian tetap terjaga, data validasi dan data uji yang digunakan tetap merepresentasikan distribusi akustik dunia nyata tanpa manipulasi komputasional.

3.2.4 Augmentasi Data Latih

Keterbatasan jumlah sampel data primer merupakan tantangan utama dalam proses pelatihan model *deep learning*. Maka, untuk mengatasi tantangan tersebut, penelitian ini

menerapkan teknik augmentasi data untuk memperbanyak variabilitas data latih secara artifisial (Kathania *et al.*, 2021; Bouchelligua, Al-dayil dan Algaith, 2025). Hal ini perlu dilakukan untuk mencegah terjadinya *overfitting* sekaligus untuk meningkatkan ketangguhan model. Sesuai dengan prinsip isolasi yang telah disebutkan pada tahap sebelumnya, proses augmentasi ini hanya diaplikasikan pada 288 sampel data latih. Korpus audio primer pada data latih diproses menjadi data sintetis melalui empat skenario manipulasi sinyal digital: penambahan derau (*add background noise*), pergeseran frekuensi nada (*pitch shifting*), peregangan waktu cepat (*time stretching – fast*), dan peregangan waktu lambat (*time stretching – slow*).



Gambar 3.3 Alur proses augmentasi data untuk satu sampel audio masukan

Secara visual, mekanisme rekayasa audio pada proses augmentasi data untuk setiap sampel audio masukan diilustrasikan pada Gambar 3.3. Berdasarkan diagram alir tersebut, proses manipulasi sinyal digital dieksekusi secara paralel. Satu sampel audio masukan diduplikasi melalui lima jalur pemrosesan independen. Jalur pertama berfungsi untuk mempertahankan bentuk audio asli tanpa modifikasi, sedangkan empat jalur lainnya memproses audio menggunakan fungsi transformasi yang telah disebutkan sebelumnya, yakni *gaussian noise*, *pitch shifting*, *time stretching – fast*, dan *time stretching – slow*. Penentuan keempat skenario tersebut merujuk pada penelitian oleh (Kathania *et al.*, 2021) yang membuktikan bahwa integrasi antara augmentasi derau dan modifikasi skala waktu, yang mencakup penyesuaian nada dan tempo, merupakan strategi paling efektif dalam

mengkompensasi karakteristik akustik unik dan tingginya frekuensi dasar (F_0) pada suara anak-anak.

Penambahan Derau (*Add Background Noise*)

Tahap pertama dalam proses augmentasi ini yakni penambahan derau atau *background noise* pada sampel audio asli. Teknik augmentasi ini dipilih dengan tujuan untuk menyimulasikan kondisi lingkungan dunia nyata yang berisik/bising. Masing-masing sampel audio akan dimanipulasi dengan cara menyuntikkan derau *Gaussian* ke dalam sinyal audio asli dengan rentang amplitudo dinamis yang telah ditentukan, yakni antara 0,001 hingga 0,015. Modifikasi ini berfungsi untuk melatih ketahanan model agar tetap mampu mengekstrak fitur akustik pelafalan anak meskipun terdapat gangguan sinyal pada latar belakang audio.

Pergeseran Frekuensi Nada (*Pitch Shifting*)

Pada tahap *pitch shifting*, setiap sampel audio masukan akan dimodifikasi frekuensi nada dasarnya tanpa mengubah durasi waktu aslinya. Nada digeser secara acak dalam rentang -2 hingga +2 *semitones* (*semitone* turun dan naik). Teknik ini menyimulasikan keragaman karakteristik vokal pita suara anak-anak yang dinamis (dari suara berat hingga melengking), sehingga mencegah model mengalami *overfitting* pada satu jenis *pitch* tertentu.

Peregangan Waktu Cepat (*Time Stretching – Fast*)

Pada tahap ini, setiap sampel audio masukan akan diubah kecepatan tempo audionya menjadi lebih cepat tanpa memengaruhi nada dasarnya. Sinyal audio dipercepat dengan rasio acak pada rentang antara 1,05x hingga 1,15x dari kecepatan normalnya. Teknik manipulasi ini merepresentasikan variasi pelafalan doa oleh anak yang tergesa-gesa.

Peregangan Waktu Lambat (*Time Stretching – Slow*)

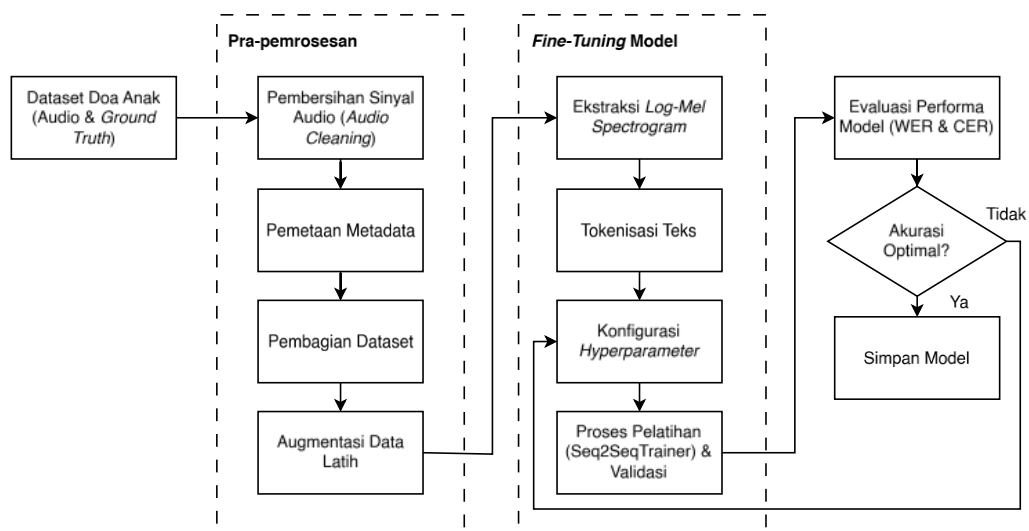
Berkebalikan dengan tahap sebelumnya, pada tahap ini, setiap sampel audio masukan akan diperlambat tempo audionya. Sinyal audio diperlambat dengan rasio acak pada rentang 0,85x hingga 0,95x. Teknik augmentasi ini diimplementasikan untuk merepresentasikan kondisi pelafalan doa oleh anak dengan kondisi mengeja secara perlahan, berhati-hati, maupun memiliki jeda pelafalan antar huruf yang dibaca panjang.

Ekspansi Dataset

Melalui tahap augmentasi ini, satu sampel data latih primer diekspansi secara paralel menjadi lima variasi file (rasio 1:5). Dengan menggunakan parameter nilai yang dinamis, seperti minimum dan maksimum (rentang acak), *dataset* yang dihasilkan untuk kebutuhan data latih menjadi jauh lebih kaya dibanding menggunakan nilai statis. Hal ini dikarenakan data audio yang dihasilkan memiliki karakteristik yang lebih bervariasi. Secara kumulatif, keseluruhan korpus latih yang berjumlah 288 sampel audio berhasil diekspansi menjadi 1.440 sampel data sintetis yang tangguh dan siap dipetakan ke dalam model. Setiap sampel audio tersebut selanjutnya dipetakan ke dalam struktur metadata baru dengan menambahkan baris data lengkap berisi ID Data, Teks Transkrip, dan ID Penutur untuk setiap hasil audio yang diaugmentasi. Dengan demikian, apabila data digabungkan dengan 72 sampel data validasi dan data uji yang dibiarkan murni, keseluruhan jumlah data audio yang siap diumpankan ke dalam siklus *fine-tuning* model Whisper berjumlah 1.512 sampel audio.

3.3 Fine-Tuning Model Whisper AI

Fase *fine-tuning* atau fase pelatihan model merupakan fase inti dalam penelitian ini. Pada tahapan ini, model *pre-trained* Whisper disesuaikan kembali bobot komputasinya menggunakan domain *dataset* spesifik yang telah dikumpulkan sebelumnya, yakni data audio pelafalan doa berbahasa Arab oleh anak-anak. Secara arsitektural, keseluruhan alur pipa data mulai dari pra-pemrosesan awal hingga siklus pelatihan dan evaluasi model diilustrasikan pada Gambar 3.4.



Gambar 3.4 Alur pipa data dan skenario *fine-tuning* model Whisper

Berdasarkan alur pada Gambar 3.4, *dataset* yang telah melalui tahap pra-pemrosesan dan isolasi augmentasi akan diproses menuju tahap ekstraksi fitur dan konfigurasi *hyperparameter* sebagai prasyarat utama pelatihan sebelum proses *fine-tuning* ini kemudian dijalankan.

3.3.1 Pemilihan Model *Baseline*

Sebelum siklus *fine-tuning* dieksekusi, penentuan arsitektur model dasar merupakan keputusan krusial yang mendasari keberhasilan penelitian. Alih-alih menggunakan model *pre-trained* Whisper standar dari OpenAI yang dilatih pada bahasa umum, penelitian ini secara spesifik menggunakan model Whisper dari *tarteel-ai/whisper-tiny-ar-quran* yang dapat diakses secara *open-source* melalui repositori HuggingFace.

Pemilihan model ini didasari oleh alasan adaptasi domain, yang mana model keluaran yang disediakan oleh Tarteel AI ini telah melewati fase pre-training yang dilatih khusus menggunakan korpus data berbahasa Arab dari pelafalan Al-Qur'an. Penggunaan model yang telah memahami dasar akustik bahasa Arab ini, akan mempercepat proses konvergensi model saat di-*fine-tuning* menggunakan *dataset* yang lebih spesifik, yakni pelafalan doa oleh anak. Sehingga, penelitian ini tidak melatih model dari representasi bahasa yang netral atau general, melainkan menerapkan *transfer learning* dari domain sebelumnya yang telah mengenal karakteristik bahasa Arab Al-Quran.

Selanjutnya, dari berbagai varian ukuran yang tersedia, penelitian ini memilih arsitektur model dengan varian *Tiny*, yang memiliki sekitar 39 juta parameter komputasi. Pemilihan model ini didasarkan pada target akhir penelitian, yakni integrasi model Whisper hasil *fine-tuning* ke dalam aplikasi bergerak Android secara *offline*. Selain itu, varian *Tiny* yang memiliki jejak memori yang kecil, menjadikannya memiliki ukuran yang ringkas sehingga memudahkan proses kuantisasi model agar dapat dimuat ke dalam RAM *smartphone* standar tanpa memicu kendala *Out of Memory* (OOM). Arsitekturnya yang ringan juga menghasilkan latensi inferensi yang rendah, sehingga perangkat mampu memproses transkripsi audio dengan cepat dalam hitungan detik secara langsung. Pendekatan *on-device* ini mendukung aplikasi untuk dapat beroperasi sepenuhnya secara *offline* tanpa memerlukan koneksi internet.

3.3.2 Ekstraksi Fitur dan Tokenisasi Data

Whisper merupakan model ASR *end-to-end* yang dibangun diatas arsitektur *transformer encoder-decoder*. Model ini tidak memiliki kemampuan untuk memproses sinyal audio mentah maupun karakter teks secara langsung. Maka, untuk dapat memproses kedua masukan tersebut,

sebelum proses pelatihan dimulai, model membutuhkan konversi data masukan ke dalam format numerik dan matriks yang menjadi representasi data mutlak yang cocok dengan bahasa mesin.

Untuk pemrosesan audio, data suara berformat `.wav` (16kHz) yang telah melewati tahap augmentasi, kemudian diekstraksi oleh modul `WhisperFeatureExtractor`. Proses ini menghasilkan representasi fitur standar Whisper berupa matriks *Log-Mel Spectrogram* beresolusi 80 kanal. Matriks ini memetakan seberapa kuat frekuensi suara anak dari awal hingga akhir rekaman. Hasil matriks ini disimpan sebagai *input_features*, yang kemudian dimasukkan ke dalam bagian *encoder* model Whisper sebagai nilai masukan.

Sementara itu, pemrosesan label teks doa berbahasa Arab yang telah disiapkan sebelumnya, dilakukan dengan memanfaatkan `Whisper Tokenizer`. Proses ini menggunakan algoritma *Byte-Pair Encoding* (BPE) untuk memotong kata-kata menjadi bagian yang lebih kecil (*sub-word tokens*), lalu diubah menjadi deretan vektor angka (*input IDs*) yang kemudian disimpan dalam variabel bernama *labels*. Deretan angka inilah yang kemudian menjadi target untuk diproses dan diurai oleh komponen *decoder* Whisper. Kedua tahapan ekstraksi ini menjadi syarat komputasional yang wajib dilakukan sebelum siklus *fine-tuning* dimulai.

3.3.3 Konfigurasi Hyperparameter

Setelah melalui tahap ekstraksi fitur dan tokenisasi teks, langkah selanjutnya adalah melakukan konfigurasi *hyperparameter*. Proses konfigurasi ini berfungsi untuk mengatur perilaku algoritma selama proses pembaruan bobot, serta mengoptimalkan performa model. Proses ini sangat menentukan keberhasilan fase *fine-tuning* model pada arsitektur *deep learning*. Pengaturan nilai *hyperparameter* yang tidak seimbang dapat mengakibatkan model mengalami *overfitting* (terlalu menghafal data latih) atau *underfitting* (gagal mengenali pola dasar).

Penelitian ini menerapkan proses optimasi bobot jaringan dengan menggunakan algoritma AdamW (*Adaptive Moment Estimation with Weight Decay*). Algoritma ini dipilih karena kemampuannya dalam mempercepat konvergensi sekaligus mencegah parameter model tumbuh terlalu besar melalui mekanisme *weight decay*. Selain itu, pelatihan juga mengimplementasikan format komputasi presisi ganda (FP16) untuk memaksimalkan kecepatan akselerator grafis NVIDIA L4 tanpa membebani memori (VRAM).

Secara lebih rinci, detail konfigurasi *hyperparameter* yang diimplementasikan pada skenario *fine-tuning* ini dirangkum pada Tabel 3.3.

Tabel 3.3 Konfigurasi *hyperparameter fine-tuning* model Whisper

Hyperparameter	Nilai
Learning Rate	1×10^{-5}
Train Batch Size	4
Gradient Accumulation	2
Max Steps	800
Warmup Steps	50
Weight Decay	0.01
Evaluation Strategy	Steps
Metric for Best Model	CER

Berdasarkan Tabel 3.3 diatas, penelitian ini tidak melakukan perbandingan skenario iterasi secara manual. Sebagai gantinya diimplementasikan mekanisme *dynamic checkpointing* (`load_best_model_at_end`). Model akan secara otomatis melakukan komparasi internal setiap 100 langkah iterasi, kemudian memuat dan menyimpan bobot model dari langkah yang menghasilkan nilai metrik CER terendah (paling optimal) sebagai hasil akhir pelatihan.

3.3.4 Metrik Evaluasi Kinerja Model

Kinerja atau performa model *Automatic Speech Recognition* (ASR) seperti Whisper dievaluasi berdasarkan tingkat kesalahan transkripsinya. Pada fase pelatihan ini, performa model diukur menggunakan dua metrik standar pengenalan suara, yakni *Word Error Rate* (WER) dan *Character Error Rate* (CER). *Word Error Rate* (WER) digunakan untuk mengukur persentase kesalahan model pada level kata, sedangkan *Character Error Rate* (CER) digunakan untuk mengukur persentase kesalahan model pada level karakter. WER dan CER didapatkan melalui penghitungan jumlah kata yang diganti (*substitution*), dihapus (*deletions*), dan disisipkan (*insertions*) dibandingkan dengan total kata pada label acuan.

Pada penelitian ini, CER ditetapkan sebagai parameter utama yang menjadi penentu model terbaik (`metric_for_best_model`). Hal ini dikarenakan, CER akan lebih sensitif terhadap karakteristik maupun susunan atau fonem bahasa Arab yang sangat sensitif terhadap perubahan satu karakter karena komponen penulisannya tersusun atas harakat, huruf, serta variasi kecil lainnya dalam doa. Kesalahan pelafalan satu huruf (misalnya '*Ain* menjadi *Alif*, atau *Sin* menjadi *Syin*) dalam WER dihitung sebagai satu kesalahan kata, sedangkan metrik CER menghitungnya sebagai satu kesalahan karakter saja. Sehingga, metrik CER memberikan evaluasi yang lebih presisi untuk mendeteksi kesalahan makhraj dan panjang pendeknya

harakat pada tingkat huruf (sub-kata) yang dilafalkan oleh anak. Semakin kecil nilai CER yang dihasilkan, maka semakin akurat model dalam mentranskripsikan pelafalan doa oleh anak.

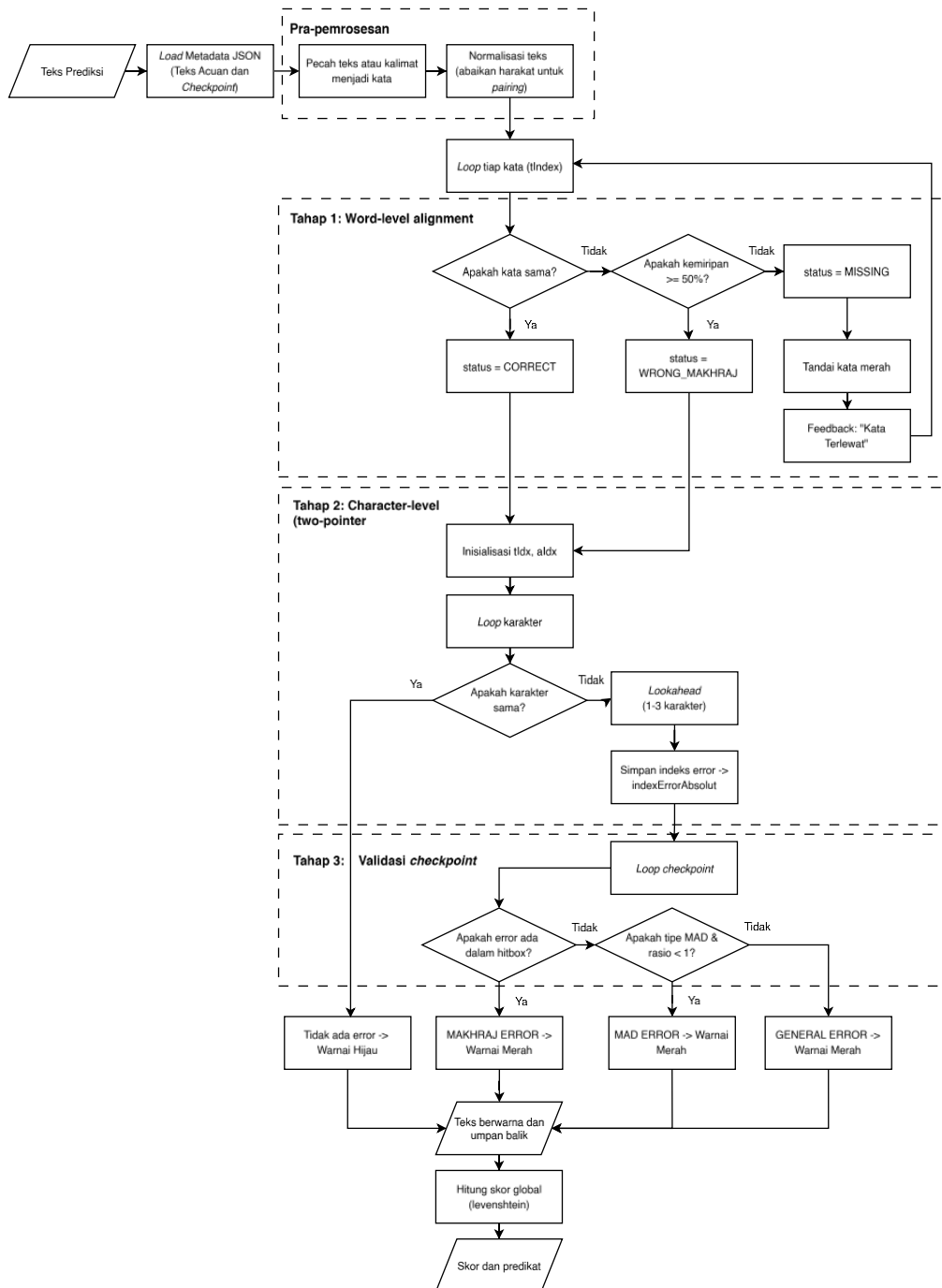
3.3.5 Skenario Evaluasi Model

Untuk mengukur efektivitas metode *fine-tuning* dan augmentasi data yang diusulkan, penelitian ini merancang skenario pengujian komparatif. Evaluasi dilakukan dengan membandingkan nilai *Word Error Rate* (WER) dan *Character Error Rate* (CER) antara dua model pada data uji yang sama. Dua model yang dievaluasi yakni (1) Model Whisper dari *tarteel-ai/whisper-tiny-ar-quran* yang merupakan model *baseline* pada penelitian ini, sebagai garis batas bawah performa, dan (2) Model Whisper yang telah melalui proses *fine-tuning* dengan data augmentasi dari korpus audio pelafalan doa oleh anak.

3.4 Perancangan Algoritma Koreksi dan Sistem Umpan Balik

Setelah teks transkripsi berhasil di-*generate* oleh model ASR Whisper, sistem akan memulai proses koreksi bacaan melalui serangkaian algoritma komputasi. Proses ini dilakukan untuk memberikan penilaian dan umpan balik otomatis terhadap pelafalan doa oleh anak.

Perancangan algoritma ini dibagi menjadi dua tahapan, yakni kalkulasi skor kemiripan teks secara global dan deteksi kesalahan bacaan secara spesifik pada aspek makhraj dan mad secara terperinci. Kedua tahapan ini dirancang untuk saling melengkapi, di mana skor kemiripan memberikan gambaran umum kualitas bacaan. Sedangkan mekanisme deteksi kesalahan akan memberikan informasi detail terkait letak kesalahan pelafalannya. Alur proses algoritma koreksi dan sistem pemberian umpan balik yang akan dibangun diilustrasikan pada Gambar 3.5.



Gambar 3.5 Alur proses algoritma koreksi dan sistem pemberian umpan balik

3.4.1 Kalkulasi Skor Kemiripan

Tahap ini bertujuan untuk mengukur tingkat kemiripan pelafalan doa harian oleh anak secara garis besar. Proses kalkulasi skor kemiripan bacaan dilakukan dengan membandingkan hasil transkripsi model Whisper terhadap teks acuan (teks asli doa) menggunakan salah satu metode dalam *Levenshtein Distance*, yakni *Levenshtein Similarity*, yang secara matematis

dapat didefinisikan dalam persamaan (3.1) dengan pembilang yang merujuk pada persamaan (2.1).

$$Similarity = 1 - \frac{LD(pred, ref)}{\max(len(pred), len(ref))} \quad (3.1)$$

dengan *pred* adalah hasil transkripsi yang dibuat oleh model ASR dan *ref* adalah referensi teks doa asli dengan penulisan yang benar. Nilai asli *similarity* berada pada rentang 0-1, di mana nilai mendekati 1 menunjukkan bahwa kedua teks tersebut memiliki kesamaan yang hampir identik. Penggunaan metrik dengan pembagi nilai panjang maksimum pada persamaan (3.1) secara matematis akan memberikan penalti pemotongan skor yang adil, baik ketika anak tidak membaca doa sampai selesai maupun ketika terdapat penambahan huruf yang tidak semestinya. Metode ini sesuai dengan fakta di lapangan bahwa bentuk kesalahan yang umumnya muncul pada pelafalan doa oleh anak diantaranya kekeliruan dalam pelafalan huruf atau huruf tertukar, hilangnya huruf, serta penambahan huruf pada kasus panjang pendeknya bacaan.

Dalam penelitian ini, evaluasi kemiripan bacaan berbasis *Levenshtein Similarity* tidak hanya mempertimbangkan apakah suatu frasa berhasil dikenali pada hasil transkripsi yang dihasilkan oleh model ASR Whisper, tetapi juga memperhatikan kesesuaian urutan dan kelengkapan bacaan atau pelafalan anak terhadap teks acuan (*referensi/ground truth*) secara utuh. Oleh karena itu, setiap operasi *insertion* (penambahan karakter), *deletion* (penghapusan karakter), dan *substitution* (penggantian karakter) akan mengurangi nilai *similarity* secara proporsional. Sebagai contoh, apabila teks referensi adalah “بِسْمِكَ”، sedangkan hasil transkripsi yang diucapkan anak terdeteksi sebagai “بِسْمِكَ اللَّهُمَّ أَحْيَا”، maka sistem tidak memberikan nilai *similarity* sebesar 100%, meskipun frasa utama berhasil dibaca atau dikenali. Hal ini disebabkan oleh tambahan karakter yang dihitung sebagai *insertion error* sehingga struktur bacaan tidak lagi identik dengan teks acuan. Pendekatan ini dipilih karena tujuan sistem bukan hanya sekadar untuk mendeteksi keberadaan kata tertentu, melainkan untuk mengevaluasi ketepatan dan kesesuaian keseluruhan bacaan doa terhadap teks referensi yang ditampilkan kepada pengguna sebelum proses perekaman dilakukan. Dengan demikian, penurunan nilai *similarity* pada penelitian ini merepresentasikan tingkat deviasi pelafalan terhadap teks referensi secara menyeluruh, bukan hanya keberhasilan mendeteksi sebagian kata target.

Untuk memperjelas mekanisme perhitungan nilai *similarity*, berikut disajikan contoh proses evaluasi antara teks referensi dan hasil transkripsi model ASR menggunakan metode *Levenshtein Similarity*.

Tabel 3.4 Contoh kasus perhitungan *Levenshtein Similarity*

Komponen	Nilai
Teks referensi (ref)	بِسْمِكَ
Hasil transkripsi (pred)	بِسْمِكَ اللَّهُمَّ أَحْيَا
Jenis perbedaan	Penambahan karakter
Jumlah operasi edit	17 karakter
Panjang maksimum <i>String</i>	23 karakter
Nilai <i>similarity</i>	26,09%

Sebagaimana yang ditunjukkan dalam Tabel 3.4, frasa utama “بِسْمِكَ” berhasil dikenali dengan benar oleh sistem. Namun, hasil transkripsi yang merupakan bacaan anak, masih mengandung tambahan frasa “اللَّهُمَّ أَحْيَا” yang tidak terdapat pada teks referensi. Tambahan karakter tersebut meningkatkan nilai *edit distance* karena menyebabkan struktur bacaan tidak lagi identik dengan teks referensi yang menjadi acuan.

Secara matematis, perhitungan nilai *similarity* ini dikalkulasikan menggunakan persamaan (3.1). Perhitungan nilai *similarity* untuk contoh kasus pada Tabel 3.4 disimulasikan dengan perhitungan berikut:

$$Similarity = 1 - \frac{17}{23} = 0,2609 = 26,09\%$$

Hasil perhitungan tersebut menunjukkan bahwa sistem tidak hanya mengevaluasi keberhasilan pendeteksian sebagian kata target, tetapi juga mengevaluasi kesesuaian keseluruhan struktur bacaan terhadap teks referensi. Oleh karena itu, penambahan karakter yang tidak sesuai tetap menghasilkan penalti *similarity* meskipun frasa utama berhasil dikenali dengan benar. Dengan demikian, nilai *similarity* pada penelitian ini merepresentasikan tingkat kesesuaian pelafalan secara menyeluruh terhadap teks referensi, bukan hanya keberhasilan mendeteksi keberadaan sebagian kata tertentu.

Hasil akhir dari kalkulasi *Levenshtein Similarity* yang dimunculkan dalam aplikasi, dikemas dalam bentuk nilai persentase berskala 0% hingga 100%. Kemudian, untuk memberikan umpan balik yang mudah dipahami bagi pengguna anak-anak, nilai persentase ini selanjutnya diklasifikasikan ke dalam beberapa kategori predikat penilaian kualitas bacaan.

Penentuan rentang persentase klasifikasi ini merujuk pada kriteria penilaian standar evaluasi pendidikan menurut (Purwanto, 2010). Predikat tersebut kemudian disesuaikan dengan terminologi apresiasi Islami agar selaras dengan konteks aplikasi evaluasi bacaan doa harian. Kategori nilai dibawah 65% digabungkan menjadi satu predikat bimbingan untuk menjaga motivasi belajar anak. Adapun kriteria dan bentuk pengelompokkan skor kemiripan pada sistem ini ditunjukkan pada Tabel 3.5.

Tabel 3.5 Pengelompokkan skor kemiripan untuk penilaian kualitas bacaan doa anak

Rentang Skor	Predikat (Purwanto, 2010)	Predikat Aplikasi	Interpretasi	Bentuk Umpan Balik
90% - 100%	Sangat Baik (A)	<i>Mumtaz</i>	Sangat Lancar	MasyaAllaah, hebat! Bacaan doamu sudah sangat jelas dan tepat. Bintang lima buat kamu!
80% - 89%	Baik (B)	<i>Jayyid</i>	Lancar	Alhamdulillah, sudah bagus! Yuk perbaiki sedikit lagi pelafalannya supaya makin sempurna.
65% - 79%	Cukup (C)	<i>Maqbul</i>	Cukup	Semangatmu luar biasa! Coba kita baca lagi pelan-pelan supaya doanya makin pas ya!
< 65%	Kurang / Tidak Lulus (D / TL)	Perlu Bimbingan	Perlu Latihan/Di bimbing	Ups, suaramu masih kurang jelas nih. Jangan menyerah, yuk tarik napas dan rekam sekali lagi!

Rentang skor ini digunakan sebagai ambang batas (*threshold*) dalam sistem untuk mengklasifikasikan kualitas pelafalan secara otomatis berdasarkan nilai kemiripan yang dihasilkan dari perhitungan *levenshtein similarity*.

3.4.2 Deteksi Kesalahan *Makhraj* dan *Mad*

Setelah melalui tahap kalkulasi skor kemiripan, teks hasil transkripsi dari model ASR Whisper kemudian diproses ke dalam arsitektur pemrosesan teks berlapis (*multi-layer text processing*) untuk mengekstraksi koordinat letak kesalahan pelafalan secara spesifik, yang nantinya akan ditampilkan sebagai umpan balik visual terkait letak kesalahan bacaan oleh anak. Lapisan logika koreksi ini disusun untuk membedah teks kata demi kata maupun karakter demi karakter untuk memberikan koreksi pada kesalahan *makhraj* huruf dan *mad* (panjang pendeknya bacaan). Logika koreksi tersusun atas tiga lapisan komputasi internal yang berkesinambungan, di mana keluaran dari satu lapisan akan menjadi bahan baku komputasi bagi lapisan berikutnya. Sistem dirancang demikian, untuk mencapai akurasi deteksi yang tinggi dan menghindari terjadinya pergeseran indeks yang dapat mengakibatkan kesalahan koreksi. Tiga lapisan tersebut

terdiri atas penjajaran tingkat kata, ekstraksi kesalahan karakter dan penguncian indeks, serta validasi *checkpoint* tajwid dan umpan balik visual.

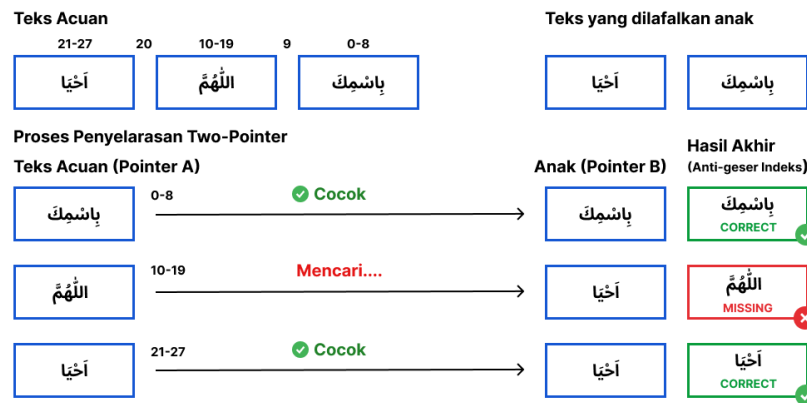
Penjajaran Tingkat Kata (*Word-Level Alignment*)

Pada lapisan pertama, sistem akan berfokus pada isolasi ruang lingkup evaluasi. Sistem memecah teks acuan dan teks prediksi hasil transkripsi model ASR menjadi larik kata menggunakan pemisah spasi. Kemudian, untuk meningkatkan fleksibilitas pencocokan, sistem akan menggunakan ekspresi reguler (Regex) untuk menormalisasi kata dan mengabaikan tanda baca harakat sementara. Hal ini dilakukan untuk memudahkan pencarian pasangan kata yang bersesuaian dari kedua teks, dengan mengomparasi konsonan dasar huruf Arab. Normalisasi ini hanya digunakan pada tahap penyelarasan kata, sementara pada tahap evaluasi karakter, sistem tetap menggunakan teks asli berharakat untuk menjaga presisi posisi indeks.

Selanjutnya sistem melakukan pencocokan kata secara berurutan untuk menentukan pasangan kata yang bersesuaian. Proses ini bertujuan untuk mengidentifikasi apakah terdapat kata yang terlewat dari bacaan anak (*missing word*). Pasangan kata yang memiliki kemiripan diatas 50% namun tidak menyentuh angka sempurna 100%, akan dianggap sebagai kata yang bersesuaian tetapi memiliki ketidaktepatan pelafalan. Sedangkan, untuk pasangan kata yang memiliki rasio nilai dibawah 50%, sistem akan memvalidasi kata acuan selanjutnya dan mengidentifikasi kemungkinan terjadinya penghilangan kata atau penambahan kata. Lalu, berdasarkan hasil pencocokan ini, setiap kata diklasifikasikan ke dalam tiga kondisi: benar, salah *makhraj/mad*, dan missing.

Proses yang terjadi pada lapisan pertama ini, menghasilkan kotak pembatas berbasis indeks jangkar absolut yang krusial untuk tahap selanjutnya. Saat memecah teks acuan, sistem akan mencatat indeks titik mulai (*start index*) dan panjang asli dari setiap kata atau indeks akhirnya (*end index*). Misalnya, kata pertama menempati indeks 0 hingga 8, lalu kata kedua menempati indeks 10 hingga 19. Pemetaan koordinat spasial ini sangat penting untuk memastikan bahwa setiap kata prediksi dari pelafalan pengguna dievaluasi murni di dalam rentang indeks kata acuannya masing-masing, sehingga komputasi di lapisan selanjutnya tidak perlu menelusuri seluruh kalimat. Hal ini dilakukan untuk mencegah hancurnya urutan indeks pada setiap kata apabila pengguna melewati suatu kata di tengah pelafalan doa.

Untuk memberikan gambaran yang lebih komprehensif mengenai mekanisme penguncian indeks jangkar ini, sistem pemetaan koordinat spasial pada lapisan pertama ini, diilustrasikan pada Gambar 3.6.



Gambar 3.6 Ilustrasi mekanisme penguncian indeks jangkar absolut pada kasus kata terlewat

Gambar 3.6 diatas, membuktikan bahwa kesalahan lokal pada suatu kata, seperti pengurangan huruf atau hilangnya sebuah kata dari pelafalan pengguna, tidak akan merusak tatanan indeks pada kata setelahnya. Kata ketiga akan tetap dievaluasi pada koordinat spasial aslinya, sehingga akurasi sistem penjajaran tingkat kata tetap terjaga.

Ekstraksi Kesalahan Karakter dan Penguncian Indeks (*Two-Pointer Lookahead*)

Pasangan kata serta indeks jangkar absolut yang telah diisolasi dari lapisan sebelumnya, selanjutnya akan dibedah secara berurutan huruf demi huruf pada lapisan kedua. Pada tahap ini, sistem akan mengabaikan teks yang sudah dinormalisasi sementara oleh lapisan pertama dan memanggil kembali referensi variabel teks aslinya secara utuh (teks lengkap berharakat). Pemanggilan teks asli ini menjadi syarat utama dalam lapisan ini agar posisi indeks karakter yang sedang diekstraksi tetap presisi dan tidak bergeser dari koordinat absolutnya saat dicocokkan dengan *database* aturan makhraj dan mad nantinya.

Selanjutnya, pada pasangan kata utuh tersebut, sistem menerapkan algoritma pencarian titik ganda (*Two-Pointer*) untuk mendeteksi letak kesalahan secara spesifik pada kata prediksi. Dengan adanya indeks jangkar dari setiap kata acuan yang disimpan pada lapisan pertama, terjadinya penambahan huruf, pengurangan huruf, maupun hilangnya kata dari pelafalan pengguna, tidak akan memicu efek domino yang dapat merusak urutan indeks keseluruhan kalimat. Kesalahan akan sepenuhnya dikurung secara lokal.

Apabila ditemukan ketidakcocokan karakter antara kedua pasangan kata, dalam kata prediksi, algoritma tidak akan langsung menyalahkan sisa huruf di belakangnya. Sistem akan melakukan pengecekan sejauh 1 hingga 3 indeks ke depan untuk mencari menemukan kembali kecocokan karakter. Mekanisme ‘penguncian indeks anti-geser’ ini berfungsi untuk mengunci

indeks huruf yang benar-benar salah saja, sehingga urutan evaluasi huruf pada kata-kata berikutnya tetap akurat karena indeks acuan akan selalu di-*reset* dengan merujuk pada indeks awal kata acuan masing-masing pasangan kata. Titik-titik kesalahan yang ditemukan kemudian dicatat dalam bentuk angka indeks untuk diteruskan ke lapisan selanjutnya.

Validasi Checkpoint Tajwid dan Umpan Balik Visual

Pada lapisan terakhir, sistem akan melakukan pencocokan antara kumpulan angka indeks karakter salah yang telah disimpan pada tahap sebelumnya, dengan data aturan tajwid (*hitbox*) yang tersimpan dalam *database* doa berformat JSON. Karena rentang indeks per kata sudah dijaga agar tidak bergeser pada lapisan-lapisan sebelumnya, pencocokan indeks ini dapat dijalankan dengan presisi. Tahapan pemrosesan kata selanjutnya diteruskan pada evaluasi akhir oleh sistem pada dua ranah tajwid yang menjadi fokus pada penelitian ini, yakni evaluasi *makhraj* dan evaluasi *mad*.

1. Evaluasi *Makhraj*: Sistem mengecek apakah angka indeks karakter yang divalidasi sebagai bentuk kesalahan berada di dalam rentang *hitbox* aturan *makharijul huruf* pada file JSON yang tersimpan dalam variabel *checkpoint*. Apabila indeks tersebut jatuh tepat pada rentang *hitbox*, maka kesalahan tersebut dikenali sebagai pelanggaran tajwid spesifik bertipe 'MAKHRAJ'.
2. Evaluasi *Mad* (Ortografis): Khusus untuk *checkpoint* dengan tipe 'MAD', sistem akan menghitung perbandingan rasio panjang karakter antara kata prediksi dan kata acuan, untuk mengevaluasi aturan panjang bacaan. Pada proses evaluasi ini, sistem mengandalkan kembali ruang lingkup kata yang telah diisolasi pada lapisan pertama. Sistem mengomparasi rasio panjang karakter yang telah dinormalisasi tanpa harakat antara teks prediksi dan teks acuan dengan ketentuan, apabila rasio panjang karakter hasil pelafalan berada dibawah nilai 1.0 dibandingkan teks acuan, sistem akan mencatatnya sebagai durasi bacaan yang kurang panjang.
3. Pada kasus khusus untuk kesalahan pelafalan anak yang terdeteksi oleh sistem, tetapi posisi indeks hurufnya diluar zona *hitbox/checkpoint* JSON, sistem tidak akan mengabaikannya. Karakter tersebut akan tetap dicatat sebagai kesalahan pelafalan umum dan diberikan umpan balik berupa pesan *template* bawaan (*default*).

Pada lapisan terakhir, sistem akan melakukan pencocokan antara kumpulan angka indeks karakter salah yang telah disimpan pada tahap sebelumnya, dengan data aturan tajwid (*hitbox*) yang tersimpan dalam *database* doa berformat JSON. Karena rentang indeks per kata sudah

dijaga agar tidak bergeser pada lapisan-lapisan sebelumnya, pencocokan indeks ini dapat dijalankan dengan presisi. Tahapan pemrosesan kata selanjutnya diteruskan pada evaluasi akhir oleh sistem pada dua ranah tajwid yang menjadi fokus pada penelitian ini, yakni evaluasi makhraj dan evaluasi mad.

Berdasarkan hasil pencocokan tersebut, sistem akan memberikan umpan balik visual pada antarmuka aplikasi menggunakan *Spannable String*. Dengan adanya penyimpanan indeks jangkar absolut yang tidak bergeser sejak awal komputasi, sistem dapat mewarnai teks secara tepat. Warna merah digunakan untuk menunjukkan huruf yang pelafalannya salah atau terlewat, sedangkan warna hijau digunakan untuk menunjukkan pelafalan huruf-huruf dengan benar. Pengelompokan jenis kesalahan beserta umpan balik pesan (*pop-up feedback*) yang ditampilkan dalam aplikasi, dirangkum dalam Tabel 3.6.

Tabel 3.6 Pengelompokan umpan balik deteksi kesalahan tajwid pada *layer* koreksi

Status Evaluasi	Kondisi Pemicu	Tindakan Visual	Kategori Umpan Balik
CORRECT	Kata terucap dengan tepat dan lengkap	Seluruh huruf diwarnai hijau	Hebat!
MISSING	Kata tidak terdeteksi atau terlewat	Seluruh huruf diwarnai Merah	Kata Terlewat
WRONG_MAKHRAJ / MAD	Terdapat kesalahan karakter yang menabrak <i>hitbox/checkpoint</i> JSON	Huruf yang salah diwarnai merah, sisanya hijau	Koreksi Makhraj/Mad
	Terdapat kesalahan karakter, tetapi tidak menabrak <i>hitbox/checkpoint</i> JSON		Koreksi Pelafalan

3.5 Implementasi Model pada Lingkungan Android

Model ASR Whisper yang telah melalui proses *fine-tuning* pada server, tidak dapat secara langsung dijalankan pada perangkat seluler. Hal ini disebabkan oleh adanya keterbatasan komputasi, memori, dan daya baterai pada *smartphone*. Sehingga, sebelum diimplementasikan ke dalam lingkungan Android, perlu adanya optimasi arsitektur model yang kompatibel dan sesuai dengan karakteristik lingkungan Android. Oleh karena itu, sistem ini menerapkan pendekatan *Edge AI*, yakni menjalankan pemrosesan *machine learning* secara lokal (*offline*) di dalam perangkat Android pengguna tanpa bergantung pada koneksi internet.

3.5.1 Konversi dan Optimasi Model AI

Untuk menjalankan model ASR Whisper AI di dalam perangkat seluler, model bawaan yang sebelumnya berformat standar PyTorch perlu diekspor dan dikonversi ke dalam format

yang lebih ringan. Proses konversi ini menggunakan pendekatan arsitektur C/C++ dari framework `whisper.cpp` yang dirancang khusus untuk inferensi model pada perangkat dengan sumber daya terbatas. Selain konversi format, sistem juga melakukan optimasi melalui teknik kuantisasi. Teknik ini memadatkan atau mengkompresi ukuran bobot parameter model yang awalnya berukuran sangat besar menjadi jauh lebih kecil tanpa mengorbankan tingkat akurasi secara signifikan. Hasil akhir dari proses konversi dan optimasi ini adalah sebuah *file* model tereksekusi berekstensi `.bin`. File inilah yang menjadi ‘otak’ pengenalan suara mandiri yang siap ditanam dalam aplikasi.

3.5.2 Integrasi Model dan Algoritma via Java Native Interface (JNI)

Setelah file model berformat `.bin` berhasil dibuat, file tersebut kemudian dimasukkan ke dalam direktori `assets` pada proyek Android Studio agar terpaket bersama aplikasi saat proses instalasi dijalankan. Namun, karena antarmuka dan logika utama aplikasi dibangun menggunakan bahasa pemrograman Kotlin, sedangkan model AI dijalankan menggunakan bahasa C/C++, maka diperlukan sebuah jembatan penghubung antar-bahasa pemrograman. Oleh karenanya, sistem menggunakan *Java Native Interface* (JNI) sebagai lapisan perantara (*bridge*). Melalui JNI, kode Kotlin dapat memanggil fungsi-fungsi spesifik di dalam modul C++ untuk mengaktifkan mikrofon, memuat model `.bin` ke dalam memori perangkat, dan menjalankan proses transkripsi suara menjadi teks berbahasa Arab secara real-time.

Teks transkripsi mentah yang dihasilkan dari modul C++ ini kemudian dikembalikan ke sisi Kotlin. Teks inilah yang selanjutnya akan langsung diumpankan ke dalam algoritma koreksi dan sistem umpan balik yang telah dijabarkan pada Subbab 3.4 sebelumnya. Teks dikalkulasi skor kemiripannya, lalu dievaluasi letak kesalahan tajwidnya, dan ditampilkan bentuk koreksi dan umpan baliknya pada layar pengguna.

3.5.3 Visualisasi Sistem sebagai Wadah Uji Coba

Sebagaimana yang telah dijelaskan sebelumnya, penelitian ini memerlukan wadah pengujian yang representatif untuk membuktikan bahwa seluruh algoritma yang dikembangkan dapat beroperasi secara integratif dan *real-time*. Oleh karena itu, sebagai *proof of concept*, sistem diimplementasikan ke dalam antarmuka Android. Antarmuka pada lingkungan uji ini, dirancang dan difokuskan untuk memfasilitasi proses *input* dalam bentuk perekaman suara audio dan memvisualisasikan output hasil komputasi dari algoritma *layer* koreksi.

3.6 Pengujian dan Evaluasi Sistem

Tahap akhir dari metodologi ini adalah merencanakan skenario pengujian untuk mengukur akurasi dan reliabilitas sistem. Pengujian dilakukan melalui tiga tahapan yakni, pengujian lapangan, evaluasi algoritma koreksi, serta validasi sistem koreksi bacaan doa oleh ahli.

3.6.1 Skenario Pengujian Lapangan

Proses pengujian aplikasi sistem koreksi bacaan doa harian dilakukan secara langsung untuk mendapatkan data primer dari target pengguna. Pengujian melibatkan responden anak-anak dengan rentang usia 6-12 tahun. Dalam skenario ini, setiap anak atau responden diminta untuk memilih satu doa yang ingin dibaca atau dihafalkan, kemudian suara direkam melalui bantuan mikrofon *smartphone* yang terhubung ke antarmuka aplikasi ‘Teman Ibadah Si Kecil’. Seluruh hasil pengujian dicatat dalam tabel rekapitulasi yang memuat identitas responden, jenis doa, hasil transkripsi sistem, skor kemiripan, dan jenis kesalahan yang muncul. Pendekatan ini memastikan bahwa proses penilaian berlangsung konsisten dan objektif dengan menggambarkan performa aplikasi dalam konteks penggunaan di dunia nyata oleh anak-anak.

3.6.2 Penetapan *Ground Truth* melalui Validasi Ahli (*Expert Judgement*)

Sebagai sistem yang mengoreksi kaidah keagamaan (tajwid), performa algoritma koreksi yang dibangun dalam sistem harus diukur berdasarkan standar kebenaran (*ground truth*). *Ground truth* ini diperoleh melalui metode *blind correction*, di mana seorang Ahli Tajwid mengevaluasi sampel audio pengujian secara mandiri tanpa melihat hasil deteksi sistem. Hasil penilaian pakar ini kemudian akan ditetapkan sebagai tolok ukur atau kunci jawaban letak kesalahan bacaan anak. Hasil deteksi otomatis yang dihasilkan dari algoritma koreksi akan dibandingkan dengan penilaian manual dari seorang ahli tajwid, untuk memvalidasi apakah logika algoritma dan sistem skor yang ada dalam aplikasi telah berjalan akurat dan selaras dengan kaidah ilmu tajwid. Tingkat kesepakatan antara mesin dan pakar, selanjutnya akan diukur menggunakan koefisien *Cohen’s Kappa*.

3.6.3 Metrik Evaluasi Algoritma Koreksi

Kinerja algoritma pemrosesan teks dalam mengkalkulasikan skor kemiripan dan mendeteksi kesalahan *makhraj* dan *mad* diukur menggunakan *confusion matrix*. Kinerja algoritma dihitung berdasarkan kemampuannya mengklasifikasikan huruf ke dalam kelas *True*

Positive, *True Negative*, *False Positive*, dan *False Negative*. Dari matriks tersebut, akan dihitung juga nilai *Precision*, *Recall*, dan *F1-Score* untuk melihat keseimbangan akurasi logika sistem.

3.7 Lingkungan Eksperimen dan Implementasi Sistem

Seluruh rangkaian eksperimen dalam penelitian ini, mulai dari tahap pra-pemrosesan data, *fine-tuning* model, hingga implementasinya ke dalam aplikasi bergerak, membutuhkan spesifikasi perangkat yang memadai baik pada perangkat keras (*hardware*) dan perangkat lunak (*software*). Pemenuhan spesifikasi minimal dilakukan untuk menjamin konsistensi hasil dan mencegah terjadinya *bottleneck* komputasi. Lingkungan eksperimen dalam penelitian ini dipisahkan ke dalam dua kelompok berdasarkan fungsinya, yakni lingkungan pelatihan model dan lingkungan pengembangan aplikasi.

3.7.1 Lingkungan Pelatihan Model Kecerdasan Buatan

Siklus pelatihan dan pengujian model Whisper dieksekusi menggunakan bantuan layanan komputasi awan (*cloud computing*) yang disediakan oleh *platform* Google Colaboratory. Hal ini dilakukan untuk mendapatkan akselerasi Unit Pemroses Grafis (GPU) yang dedikatif. Spesifikasi lingkungan pelatihan yang digunakan dalam penelitian ini adalah sebagai berikut:

- GPU: GPU NVIDIA Ada Lovace L4 dengan VRAM 22,5 GB dan System RAM 53,0 GB
- Bahasa Pemrograman: Python versi 3.12.12
- Mode Eksekusi: Notebook .ipynb

3.7.2 Lingkungan Pengembangan Aplikasi Bergerak

Setelah model ASR Whisper diekspor ke dalam format .bin yang dioptimasi, proses integrasi dan pembangunan antarmuka sistem dilakukan pada lingkungan pengembangan lokal menggunakan perangkat pribadi dengan spesifikasi sebagai berikut:

- Perangkat Keras (Pengembang): Laptop Acer Aspire 5 A514-54
 - Prosesor: 11th Gen Intel® Core™ i5-1135G7 @ 2.40GHz (2.42 GHz)
 - RAM: 24,0 GB
 - Sistem Operasi: Windows 11 Home Single Language 64-bit
- Perangkat Keras (Pengujian): *Smartphone* Android Vivo Y12s
 - Prosesor: 1,95 GHz Snapdragon 439 Octa-core

- RAM: 3,00 GB
- Versi Android: 11.0
- Perangkat Lunak (IDE): Android Studio Koala Feature Drop | 2024.1.2 Patch 1
- Bahasa Pemrograman: Kotlin untuk logika antarmuka dan C++ (melalui JNI) untuk menjembatani inferensi model AI dalam Aplikasi

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pra-Pemrosesan Data Audio

Tahap pra-pemrosesan data audio merupakan fondasi utama dalam penelitian ini, karena kualitas sinyal yang dihasilkan secara langsung memengaruhi performa model ASR dalam mengenali pelafalan doa anak. Proses ini bertujuan untuk memastikan bahwa setiap rekaman yang diumpankan ke dalam arsitektur model telah berada dalam kondisi yang optimal, baik dari segi kebersihan sinyal, konsistensi format, maupun representasi metadata. Seluruh proses ini dieksekusi menggunakan lingkungan komputasi *cloud* Google Colab dengan bahasa pemrograman Python. Tahapan yang dilakukan meliputi pembersihan sinyal suara, pemetaan dan pembersihan metadata, pembagian *dataset*, serta augmentasi data untuk meningkatkan ketahanan model terhadap variasi pelafalan anak di dunia nyata.

4.1.1 Pembersihan Sinyal Suara (*Audio Cleaning*)

Proses *audio cleaning* dilakukan secara menyeluruh terhadap data sampel yang dihimpun. Tahapan ini berupa serangkaian proses untuk penyeragaman format dan *resampling* audio ke laju 16 kHz, normalisasi amplitudo, serta penghapusan bagian hening yang tidak mengandung informasi fonetik di awal dan di akhir rekaman doa. Berdasarkan batasan parameter yang telah ditetapkan pada metodologi, proses pembersihan sinyal audio ini dieksekusi dengan memanfaatkan pustaka `pydub` (modul `AudioSegment`). Tahap pembersihan sinyal audio ini sangat penting untuk dilakukan karena model ASR seperti Whisper sangat sensitif terhadap *noise* dan bagian hening tanpa suara. Keberadaan jeda yang terlalu lama dan tidak relevan dapat menyebabkan model salah mengenali awal atau akhir kata, yang dapat berdampak pada akurasi transkripsi yang akan digunakan sebagai masukan utama sistem evaluasi pelafalan anak.

Implementasi teknis penghapusan jeda dibungkus ke dalam sebuah fungsi yang diberi nama `trim_silence()`. Fungsi ini bertugas memindai objek audio dan mendeteksi segmen wicara yang memiliki intensitas di atas ambang batas -45 dBFS, sekaligus mengabaikan derau pendek berdurasi di bawah 100 milidetik. Setelah batas wicara terdeteksi, sistem mengaplikasikan penambahan rentang waktu (*padding*) sebesar 150 milidetik pada titik awal dan akhir pemotongan, untuk mencegah terjadinya pemotongan kasar pada ujung pelafalan

yang melemah. Potongan kode yang digunakan dalam rangkaian proses pembersihan sinyal audio tersebut diilustrasikan pada Gambar 4.1 .

```
def trim_silence(audio, silence_thresh=-45.0, min_silence_len=100, padding_ms=150):
    # Suara yang panjangnya kurang dari 100ms akan diabaikan
    nonsilent_ranges = detect_nonsilent(audio,
                                        min_silence_len=min_silence_len,
                                        silence_thresh=silence_thresh)

    if nonsilent_ranges:
        start_trim = nonsilent_ranges[0][0]
        end_trim = nonsilent_ranges[-1][1]

        start_trim = max(0, start_trim - padding_ms)
        end_trim = min(len(audio), end_trim + padding_ms)

        return audio[start_trim:end_trim]
    else:
        return audio

# Memuat audio mentah dan penyeragaman format (16kHz, Mono, 16-bit)
raw_path = os.path.join(audio_input, filename)
audio = AudioSegment.from_file(raw_path)
audio = audio.set_frame_rate(16000).set_channels(1).set_sample_width(2)

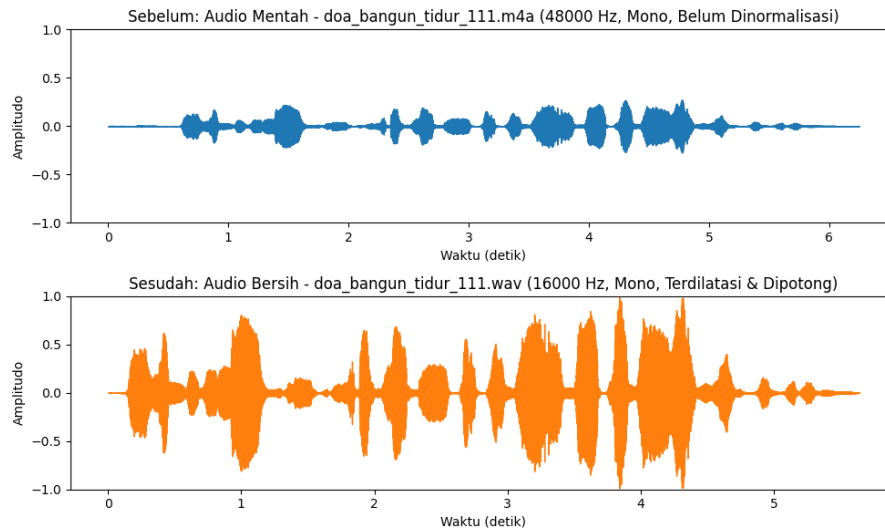
# Normalisasi intensitas suara
audio = audio.normalize()

# Proses trimming: menghapus silence dengan threshold -40.0 dB
clean_audio = trim_silence(audio)

# Menyimpan audio bersih ke format .wav
clean_filename = os.path.splitext(filename)[0] + ".wav"
clean_audio.export(os.path.join(audio_output, clean_filename), format="wav")
```

Gambar 4.1 Potongan kode pembersihan sinyal audio

Tahap pembersihan sinyal audio ini diterapkan pada seluruh berkas audio dalam *dataset* secara berulang dan konsisten. Hasil akhir pada tahap ini yakni sampel audio baru berformat .wav yang lebih padat. Perubahan struktur sinyal audio, pada sampel audio bersih ini secara visual dapat direpresentasikan melalui grafik gelombang suara (*waveform*). Struktur sinyal audio sebelum dan sesudah proses *audio cleaning* dilakukan dapat dilihat pada Gambar 4.2.



Gambar 4.2 Perbandingan struktur sinyal audio mentah (atas) dan hasil pembersihan sinyal (bawah) pada sampel audio `doa_bangun_tidur_111`

Berdasarkan visualisasi pada Gambar 4.2, dapat dilihat bahwa hasil dari ketiga proses tersebut terbukti secara nyata. Pertama, seluruh berkas telah terstandarisasi format dan laju sampelnya agar sesuai dengan arsitektur Whisper, yakni .wav dengan *sampling rate* 16.000 Hz. Kedua, efek normalisasi audio terlihat jelas pada plot bawah (oranye). Sinyal suara anak yang sebelumnya bervolume rendah, ditingkatkan ke skala optimal (-1.0 hingga 1.0) tanpa merusak kualitasnya. Terakhir, efektivitas proses *trimming* dibuktikan dengan terpotongnya bagian dengan amplitudo datar pada awal rekaman dan akhir rekaman yang merepresentasikan suara hening.

Pada kasus sampel diatas, jeda hening sepanjang 0,61 detik telah berhasil dieliminasi, sehingga durasi sampel audio yang awalnya 6,25 detik menyusut menjadi 5,64 detik. Proses ini menghasilkan keluaran berupa berkas audio yang lebih padat, di mana seluruh durasi rekaman kini sepenuhnya berisi pelafalan doa dari anak. Penerapan algoritma pemotongan jeda ini berhasil mereduksi ruang kosong pada *dataset* secara signifikan. Dari total 360 sampel data yang berdurasi 47 menit 40,73 detik, sistem berhasil menyusutkan data menjadi 46 menit 44,18 detik yang siap untuk diproses ke tahap berikutnya. Proses pembersihan ini tidak hanya meningkatkan kualitas sinyal audio, tetapi juga memastikan bahwa *input* yang diterima model benar-benar merepresentasikan pelafalan doa, sehingga hasil transkripsi yang diberikan menjadi lebih akurat dan dapat diandalkan untuk proses evaluasi bacaan. Dengan demikian, kualitas tahap ini secara langsung menentukan ketepatan sistem dalam menilai kesalahan pelafalan anak.

4.1.2 Pemetaan dan Pembersihan Metadata

Setelah proses pembersihan sinyal audio selesai dilakukan, tahap selanjutnya yang dilakukan yakni memetakan dan membersihkan *master dataset* yang akan digunakan sebagai metadata untuk menghindari beban komputasi pada atribut yang kurang relevan, dalam proses pelatihan atau *fine-tuning*. Langkah pertama dalam tahap ini yakni memuat pangkalan data utama atau *master dataset* ke dalam *dataframe*. Pemuatan ini menggunakan pustaka *Pandas* untuk membaca berkas `log_dokumentasi_audio.csv` yang berisi seluruh log dokumentasi pengumpulan data. Proses ini dilakukan dengan menjalankan serangkaian kode perintah sebagaimana yang termuat dalam Gambar 4.3.

```
import pandas as pd

# Memuat master data log_dokumentasi_audio.csv dari Google Drive
master_path = '/content/drive/MyDrive/Proyek_ASR2/log_dokumentasi_audio.csv'
df = pd.read_csv(master_path, sep=';')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 360 entries, 0 to 359
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID Data                360 non-null   object
1   Nama File Audio        360 non-null   object
2   Jenis Doa              360 non-null   object
3   Teks Transkripsi       360 non-null   object
4   Durasi                 360 non-null   object
5   Sumber Data            360 non-null   object
6   Tanggal Pengambilan    360 non-null   object
7   ID Penutur             360 non-null   object
8   Nama                   360 non-null   object
9   Usia                   161 non-null   object
10  Catatan                360 non-null   object
dtypes: object(11)
memory usage: 31.1+ KB
```

Gambar 4.3 Proses pemuatan metadata

Selanjutnya, sesuai dengan rancangan pada Bab 3, metadata yang telah dimuat diproses ke tahap pembersihan atribut dan normalisasi ekstensi. Atribut atau fitur yang tidak relevan, seperti ID Data, Durasi, Sumber Data, Tanggal Pengambilan, Nama, Usia, dan Catatan dieliminasi menggunakan perintah `df.drop()`. Sedangkan, normalisasi ekstensi pada atribut Nama *File* Audio dilakukan dengan perintah `replace()` untuk mengganti string `.m4a` pada log dokumentasi ke `.wav` sebagaimana format baru dari hasil proses pembersihan sinyal audio. Proses penghapusan fitur dan normalisasi ekstensi dapat dilihat pada Gambar 4.4.

```
# Eliminasi atribut yang tidak relevan
df= df.drop(['ID Data', 'Nama', 'Usia', 'Tanggal', 'Durasi', 'Sumber', 'Catatan'], axis=1)

# Normalisasi ekstensi berkas menjadi .wav
df['Nama File Audio'] = df['Nama File Audio'].str.replace(r'\.[a-zA-Z0-9]+$', '.wav',
regex=True)

df.head()
```

	Nama File Audio	Jenis Doa	Teks Transkripsi	ID Penutur
0	doa_sebelum_tidur_001.wav	Doa Sebelum Tidur	يَسْتَوِيكَ اللَّهُمَّ أَخِيَا وَ يَسْتَوِيكَ أُمُّ	Anak-001
1	doa_sebelum_tidur_002.wav	Doa Sebelum Tidur	يَسْتَوِيكَ اللَّهُمَّ أَخِيَا وَ يَسْتَوِيكَ أُمُّ	Anak-001
2	doa_sebelum_tidur_003.wav	Doa Sebelum Tidur	يَسْتَوِيكَ اللَّهُمَّ أَخِيَا وَ يَسْتَوِيكَ أُمُّ	Anak-001
3	doa_sebelum_tidur_004.wav	Doa Sebelum Tidur	يَسْتَوِيكَ اللَّهُمَّ أَخِيَا وَ يَسْتَوِيكَ أُمُّ	Anak-002
4	doa_sebelum_tidur_005.wav	Doa Sebelum Tidur	يَسْتَوِيكَ اللَّهُمَّ أَخِيَا وَ يَسْتَوِيكَ أُمُّ	Anak-002

Gambar 4.4 Proses penghapusan fitur dan normalisasi ekstensi

Pembersihan metadata ini berperan penting dalam mengurangi kompleksitas fitur yang tidak relevan, sehingga proses pelatihan model menjadi lebih efisien. Selain itu, konsistensi format nama *file* audio memastikan keterhubungan yang valid antara data audio dan teks transkripsi, yang menjadi dasar utama dalam proses pembelajaran model ASR. Kesalahan pada tahap ini berpotensi menyebabkan ketidaksesuaian antara audio dan teks, yang dapat menurunkan kualitas pembelajaran model.

4.1.3 Pembagian Dataset

Selanjutnya, set data kemudian dibagi menjadi tiga bagian, yakni data latih, data validasi, dan data uji dengan rasio 80:10:10. Pemilihan rasio 80:10:10 ini didasarkan pada praktik umum dalam pelatihan model machine learning, di mana sebagian besar data digunakan untuk pembelajaran pola, sementara sebagian kecil disisihkan untuk validasi dan pengujian guna mengukur kemampuan generalisasi model. Selain itu, penggunaan pendekatan *speaker-independent split* berbasis variabel ID Penutur menjadi sangat penting dalam penelitian ini karena sistem dirancang untuk mengevaluasi pelafalan anak secara umum, bukan hanya menghafal karakteristik suara individu tertentu. Dengan demikian, model diuji pada penutur atau subjek yang belum pernah ‘dilihat’ sebelumnya yang menjamin validitas hasil pengujian. Hal ini memastikan bahwa performa model yang dihasilkan benar-benar mencerminkan kemampuan sistem dalam mengenali pelafalan baru, bukan sekadar mengingat data yang telah dilatih.

Proses ini diimplementasikan menggunakan fungsi bawaan `GroupShuffleSplit` dari pustaka `scikit-learn`. Proses ini penting untuk memastikan tidak ada identitas penutur yang saling tumpang tindih antara data latih, data validasi, serta data uji untuk menghindari

kebocoran data dan *overfitting*. Proses komputasi pembagian data menjadi tiga himpunan data ini dapat dilihat pada Gambar 4.5.

```
from sklearn.model_selection import GroupShuffleSplit

# Pemisahan 80% data latih, 20% sisa data sementara
gss_1 = GroupShuffleSplit(n_splits=1, train_size=0.8, random_state=19)
train_idx, temp_idx = next(gss_1.split(df, groups=df['ID Penutur']))

df_train = df.iloc[train_idx]
df_temp = df.iloc[temp_idx] # Himpunan sisa 20%

# Pemisahan sisa data 20% menjadi 10% data validasi dan 10% data uji
gss_2 = GroupShuffleSplit(n_splits=1, test_size=0.5, random_state=38)
val_idx, test_idx = next(gss_2.split(df_temp, groups=df_temp['ID Penutur']))

df_val = df_temp.iloc[val_idx]
df_test = df_temp.iloc[test_idx]
```

Gambar 4.5 Proses pembagian data latih, data validasi, dan data uji

Berdasarkan hasil eksekusi proses pada Gambar 4.5, dari total keseluruhan data bersih sebanyak 360 sampel audio, sistem telah membaginya secara presisi sesuai dengan rasio yang ditetapkan. Rincian jumlah akhir sampel pada setiap himpunan data dari distribusi pembagian *dataset* tersebut disajikan pada Tabel 4.1.

Tabel 4.1 Distribusi hasil akhir pembagian *dataset* audio doa

Kategori Data	Proporsi	Jumlah Sampel Akhir
Data Latih	80%	288 sampel
Data Validasi	10%	36 sampel
Data Uji	10%	36 sampel
Total Keseluruhan	100%	360 sampel

4.1.4 Augmentasi Data

Tahap augmentasi data dilakukan untuk mensimulasikan variasi kondisi pelafalan anak di dunia nyata, seperti perbedaan intonasi, kecepatan bicara, serta gangguan lingkungan yang berisik/bising. Hal ini penting karena dalam praktiknya, anak-anak tidak selalu membaca doa dalam kondisi yang ideal, sehingga model perlu dilatih untuk tetap *robust* terhadap variasi tersebut. Selain itu, untuk mengatasi keterbatasan jumlah dan varian akustik pada data audio yang dimiliki, tahapan augmentasi data dieksekusi secara komputasional menggunakan pustaka *audiomentations*. Merujuk pada skenario yang telah disusun pada metodologi, tahap ini diimplementasikan khusus untuk himpunan data latih, di mana setiap berkas audio asli

diekstraksi menjadi empat varian modifikasi. Augmentasi dengan pendekatan isolasi pada data latih ini bermaksud untuk mencegah terjadinya kebocoran data pada fase evaluasi model.

Tahap augmentasi data ini berhasil meningkatkan jumlah data latih secara signifikan. Dari 288 sampel audio asli, *dataset* berkembang menjadi 1.440 sampel data latih yang tangguh dan siap dipetakan ke dalam model. Peningkatan ini berasal dari empat variasi augmentasi yang diterapkan pada setiap data. Rincian implementasi algoritma untuk masing-masing varian dijabarkan pada sebagai berikut.

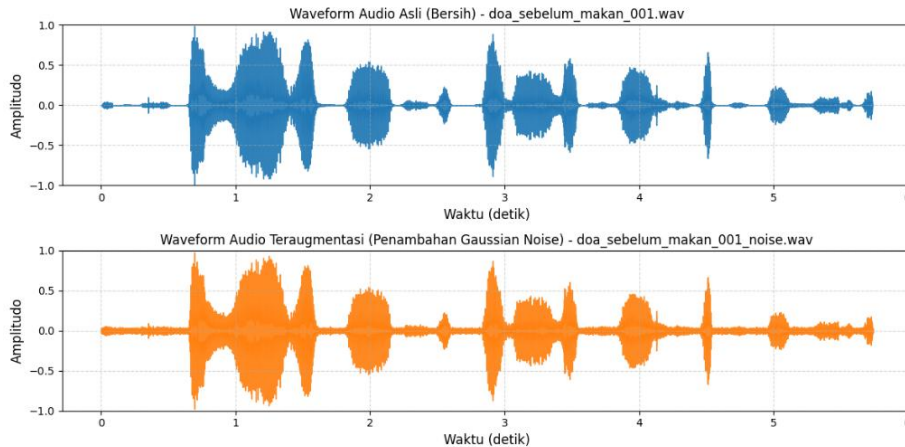
Penambahan Derau (*Add Background Noise*)

Penambahan derau dilakukan untuk mensimulasikan kondisi lingkungan yang tidak ideal dari lingkungan anak ketika melakukan perekaman suara, seperti suara lingkungan rumah atau kelas, yang umum terjadi pada penggunaan nyata aplikasi. Proses ini membantu model untuk tetap mampu mengenali pelafalan meskipun terdapat gangguan. Derau latar ini disuntikkan secara acak menggunakan rentang amplitudo 0.001 hingga 0.015. Injeksi derau sintetik ini dikonfigurasi melalui pustaka `AddGaussianNoise` dengan menjalankan perintah yang dapat dilihat pada Gambar 4.6.

```
# Penambahan gaussian noise dengan rentang amplitudo 0.001-0.015
aug_noise = AddGaussianNoise(min_amplitude=0.001, max_amplitude=0.015, p=1.0)
```

Gambar 4.6 Proses penambahan derau latar secara dinamis

Komputasi algoritma ini menyumbang 288 sampel audio berderau ke dalam data latih. Distorsi akustik yang ditimbulkan oleh derau latar yang ditambahkan dari proses ini, terlihat pada penebalan pita sinyal audio, di mana garis datar yang semula bersih (mendekati nol) menjadi bergerigi secara konstan. Perubahan struktur sinyal tersebut dapat dilihat pada Gambar 4.7.



Gambar 4.7 Perbandingan struktur sinyal audio asli (atas) dan sinyal audio dengan injeksi *gaussian noise* (bawah)

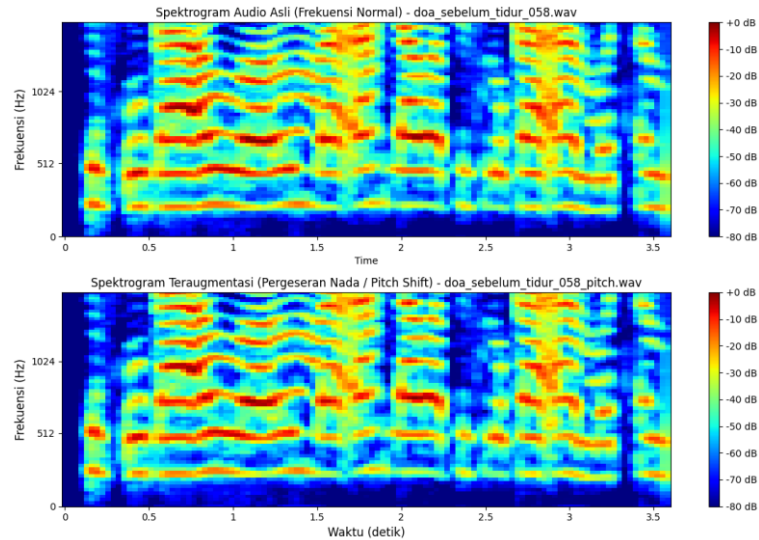
Pergeseran Frekuensi Nada (*Pitch Shifting*)

Untuk melatih model menghadapi karakteristik biologis pita suara anak-anak yang menghasilkan frekuensi dasar lebih tinggi dibandingkan orang dewasa, pada tahap ini, dilakukan augmentasi pergeseran frekuensi nada. Variasi ini merepresentasikan perbedaan karakter suara anak, yang secara alami memiliki frekuensi lebih tinggi dan bervariasi. Nada digeser secara acak dalam rentang dinamis -2 hingga +2 *semitones* (*semitone* turun dan naik). Pemilihan batas pergeseran nada ini penting untuk mencegah *distorsi formant* agar suara anak tidak kehilangan struktur keaslian vokalnya. Konfigurasi parameter batas dinamis ± 2 *semitones* ini diimplementasikan melalui pustaka `PitchShift` dengan menjalankan perintah yang dapat dilihat pada Gambar 4.8.

```
# Inisialisasi parameter pergeseran frekuensi nada dalam rentang  $\pm 2$  semitones
aug_pitch = PitchShift(min_semitones=-2, max_semitones=2, p=1.0)
```

Gambar 4.8 Proses pergeseran frekuensi nada secara dinamis

Eksekusi komputasi pada tahap ini menyumbang 288 sampel audio dengan variasi nada ke dalam data latih. Pergeseran frekuensi ini ditunjukkan melalui representasi spektrogram pada Gambar 4.9, di mana distribusi energi harmonik mengalami transisi secara vertikal tanpa merusak struktur waktu pelafalan.



Gambar 4.9 Perbandingan spektrogram sinyal audio asli (atas) dan sinyal audio dengan pergeseran frekuensi nada ± 2 semitones (bawah)

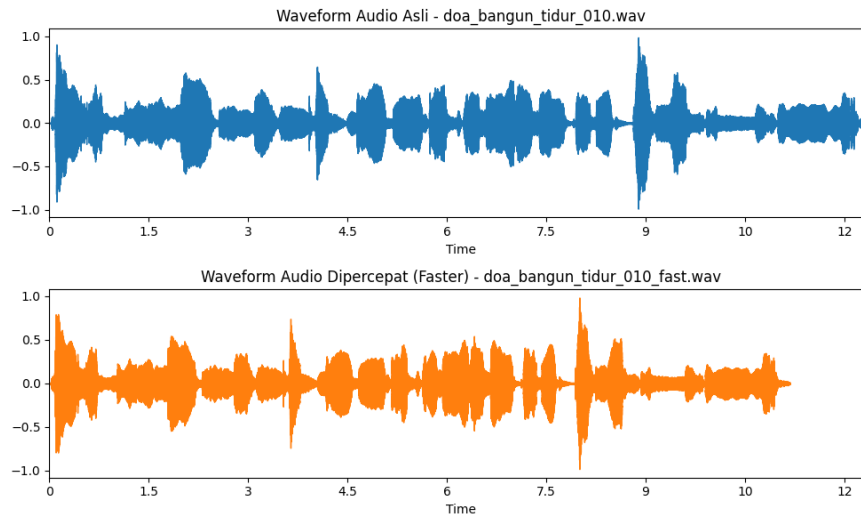
Peregangan Waktu Cepat (*Time Stretching – Fast*)

Augmentasi dengan peregangan waktu cepat atau percepatan durasi ini dirancang untuk merepresentasikan kondisi di mana anak melafalkan doa dengan tempo yang terburu-buru. Audio dipercepat secara acak dengan rentang percepatan 1.05x sampai 1.15x. Eksekusi algoritma ini dikonfigurasi melalui pustaka `TimeStretch` dengan menjalankan perintah yang dapat dilihat pada Gambar 4.10.

```
# Inisialisasi parameter percepatan durasi dalam rentang 1.05-1.15
aug_fast = TimeStretch(min_rate=1.05, max_rate=1.15, leave_length_unchanged=False, p=1.0)
```

Gambar 4.10 Proses percepatan durasi secara dinamis

Tahap ini menyumbang 288 sampel audio cepat ke dalam data latih. Modifikasi durasi ini dapat direpresentasikan melalui pemampatan gelombang amplitudo secara horizontal, di mana jarak antar puncak gelombang menjadi lebih rapat dibandingkan sinyal aslinya. Perbandingan struktur sinyal audio ini dapat dilihat pada Gambar 4.11.



Gambar 4.11 Perbandingan struktur sinyal audio asli (atas) dan sinyal audio hasil percepatan durasi (bawah)

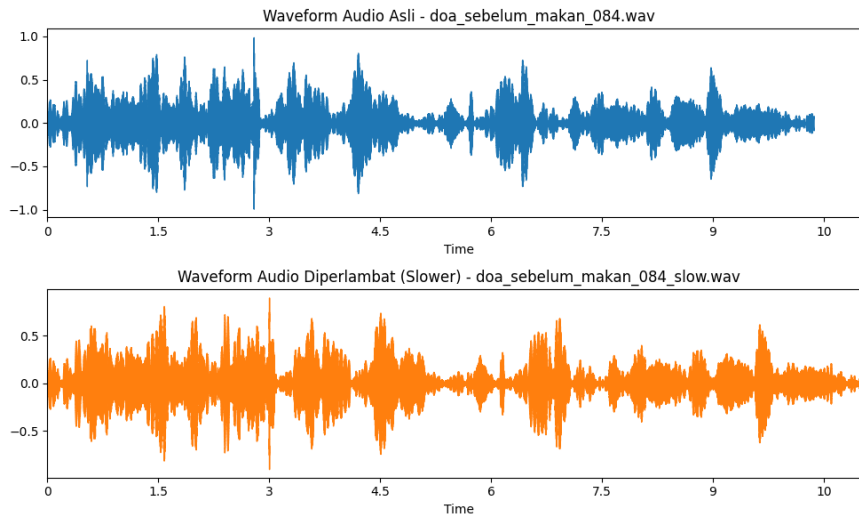
Peregangan Waktu Lambat (*Time Stretching – Slow*)

Berbanding terbalik dengan tahap sebelumnya, peregangan waktu lambat atau perlambatan laju bicara diterapkan untuk menyimulasikan pelafalan anak-anak di kondisi nyata, di mana anak membaca dengan tempo lambat, terbata-bata, ataupun mengeja. Audio diperlambat secara acak dengan rentang perlambatan 0.85x sampai 0.95x. Eksekusi algoritma ini diimplementasikan melalui pustaka `TimeStretch` dengan menjalankan perintah yang dapat dilihat pada Gambar 4.12.

```
# Inisialisasi parameter percepatan durasi dalam rentang 1.05-1.15
aug_slow = TimeStretch(min_rate=0.85, max_rate=0.95, leave_length_unchanged=False, p=1.0)
```

Gambar 4.12 Proses perlambatan durasi secara dinamis

Eksekusi kode pada Gambar 4.12 berhasil menyumbang 288 sampel audio baru yang diperlambat ke dalam data latih. Modifikasi durasi ini dapat direpresentasikan melalui peregangan gelombang amplitudo yang memanjang secara horizontal. Perbandingan struktur sinyal audio ini dapat dilihat pada Gambar 4.13.



Gambar 4.13 Perbandingan struktur sinyal audio asli (atas) dan sinyal audio hasil perlambatan durasi (bawah)

Integrasi dan Ekspansi *Dataset*

Setelah seluruh parameter augmentasi diinisialisasi, langkah selanjutnya adalah mengeksekusi *pipeline* komputasi terhadap himpunan data latih orisinal. Sistem melakukan iterasi pada setiap baris metadata train.csv. Pada setiap iterasi, sampel audio asli diekstraksi dan dilewatkan pada keempat fungsi augmentasi secara paralel.

Setiap keluaran dari fungsi augmentasi tersebut disimpan sebagai sampel audio berformat .wav baru. Sementara metadatanya, sampel audio baru hasil augmentasi dipetakan dengan metadata yang disalin persis dari berkas asli, dengan pembaruan pada bagian atribut Nama File Audio. Implementasi integrasi *dataset* ini diterapkan pada seluruh berkas audio dalam data latih, sehingga setiap sampel menghasilkan beberapa baris metadata dengan varian barunya. Logika inti dari proses tersebut ditunjukkan dengan potongan kode pada Gambar 4.14.

```
# Eksekusi pipeline augmentasi dan penyimpanan metadata
# Load audio asli
audio, sr = librosa.load(path_asli, sr=16000)

# Eksekusi cabang augmentasi dan simpan audio baru (contoh cabang penambahan derau)
audio_noise = aug_noise(samples=audio, sample_rate=sr)
nama_noise = nama_file_asli.replace('.wav', '_noise.wav')
sf.write(os.path.join(folder_audio_aug, nama_noise), audio_noise, sr)

row_noise = row.copy()
row_noise['Nama File Audio'] = nama_noise
list_baru.append(row_noise)

# Simpan Metadata hasil augmentasi
df_train_aug = pd.DataFrame(list_baru)
df_train_aug.to_csv('train_augmented.csv', index=False, encoding='utf-8-sig')
```

Gambar 4.14 Proses integrasi dan multiplikasi metadata audio baru hasil augmentasi

Proses komputasi di atas menghasilkan berkas metadata baru yang memuat seluruh data hasil augmentasi dari sampel data latih yang lebih siap diproses pada tahap berikutnya. File bernama `train_augmented.csv` ini memuat data Nama File Audio, Jenis Doa, Teks Transkripsi, dan ID Penutur, yang menggabungkan integrasi dari 288 data asli dan 1.152 data termodifikasi. Berkas metadata baru ini yang akan digunakan dalam fase pelatihan dan diumpankan ke arsitektur model Whisper. Secara keseluruhan, rangkaian pra-pemrosesan ini tidak hanya menghasilkan *dataset* yang bersih dan terstandarisasi, tetapi juga memperkaya variasi akustik yang merepresentasikan kondisi pelafalan anak di dunia nyata. Hal ini menjadi fondasi penting dalam meningkatkan performa model ASR sebagai ‘pendengar virtual’, yang pada akhirnya akan berkontribusi langsung dalam menentukan akurasi dan kualitas sistem dalam mengevaluasi pelafalan doa anak secara otomatis.

4.2 Proses *Fine-Tuning* dan Evaluasi Model Whisper AI

Tahap pelatihan atau *fine-tuning* model Whisper AI dilakukan untuk menyesuaikan bobot model *pre-trained* agar lebih sensitif dan akurat dalam mengenali karakteristik suara anak-anak, khususnya dalam pelafalan doa harian berbahasa Arab. Model dasar atau *baseline* yang digunakan dalam penelitian ini adalah model Whisper varian dasar *Tiny* dari Tarteel AI yang telah dilatih menggunakan korpus bahasa Arab Al-Qur’an. Model diakses dari *platform* HuggingFace dengan id ‘*tarteel-ai/whisper-tiny-ar-quran*’. Proses *fine-tuning* ini menjadi langkah penting agar model tidak hanya mampu melakukan transkripsi, tetapi juga dapat digunakan sebagai dasar dalam sistem evaluasi pelafalan doa anak secara otomatis.

4.2.1 Konfigurasi *Hyperparameter*

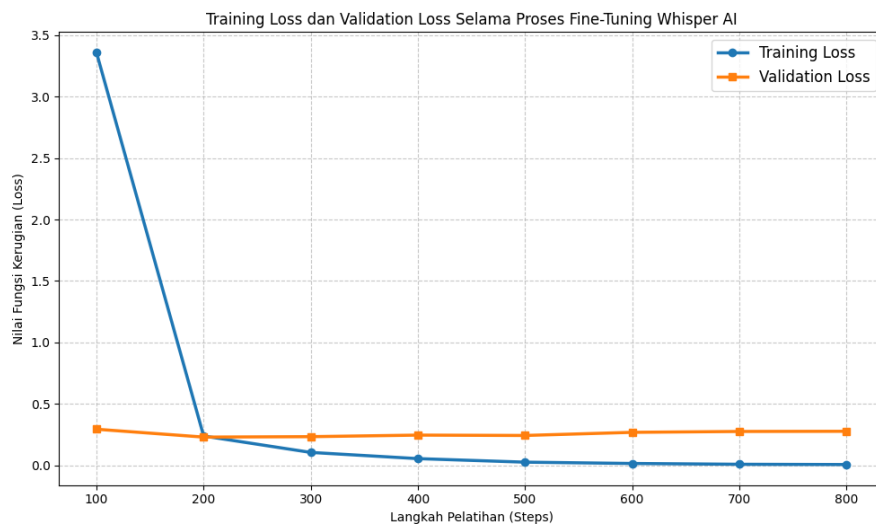
Sebagaimana yang telah dirancang pada metodologi, proses *fine-tuning* model dilakukan dengan mengatur sejumlah *hyperparameter* yang disesuaikan dengan karakteristik *dataset* dan tujuan penelitian. Penentuan konfigurasi ini tidak hanya mempertimbangkan aspek teknis pelatihan, tetapi juga menyesuaikan dengan kondisi data berupa rekaman suara anak yang cenderung memiliki variasi pelafalan, intonasi, dan kestabilan yang beragam. Oleh karena itu, konfigurasi dirancang agar model dapat belajar secara stabil tanpa kehilangan kemampuan generalisasinya terhadap data baru. Adapun konfigurasi *hyperparameter* yang ditetapkan pada penelitian ini adalah sebagai berikut:

- a. *Learning Rate* sebesar $1e-5$, nilai yang kecil ini dipilih agar proses pembelajaran berlangsung secara bertahap. Hal ini penting karena model perlu menyesuaikan diri dengan pola pelafalan anak yang tidak terlalu konsisten, sehingga perubahan bobot yang terlalu besar justru berisiko menurunkan kualitas pembelajaran.
- b. *Batch Size* (4) dan *Gradient Accumulation* (2), ukuran *batch* yang kecil digunakan untuk menyesuaikan keterbatasan memori GPU. Untuk tetap menjaga kestabilan pembelajaran, digunakan teknik *gradient accumulation* sehingga model seolah-olah memproses 8 sampel dalam satu siklus pembaruan. Pendekatan ini memungkinkan proses pelatihan tetap efisien tanpa mengorbankan kualitas hasil.
- c. *Max Steps* (800) dan *Warmup Steps* (50), total langkah pelatihan dibatasi hingga 800 iterasi, untuk menjaga efisiensi waktu pelatihan. Pada 50 langkah awal pelatihan, diterapkan fase pemanasan (*warmup*) di mana model mulai belajar secara perlahan sebelum mencapai kecepatan belajar yang optimal. Tahap ini membantu model beradaptasi dengan data tanpa mengalami ketidakstabilan di awal pelatihan.
- d. *Weight Decay* sebesar 0.01, digunakan sebagai bentuk regularisasi untuk mencegah terjadinya *overfitting*, yaitu kondisi di mana model hanya "menghafal" data latih namun kesulitan saat memproses suara anak yang baru. Hal ini penting karena sistem diharapkan mampu mengenali pelafalan dari anak lain yang tidak terdapat dalam data pelatihan.
- e. Presisi Komputasi (FP16), pelatihan dipercepat dengan mengaktifkan format komputasi presisi setengah (*Half-Precision* / FP16). Fitur ini mengoptimalkan kinerja GPU agar waktu pelatihan berjalan lebih singkat tanpa mengorbankan akurasi hasil akhir.
- f. *Generation Max Length* (225), panjang maksimal teks yang dapat dihasilkan model saat evaluasi dibatasi hingga 225 token. Penentuan batas ini bertujuan untuk memastikan transkripsi dari bacaan doa yang cukup panjang tidak akan terpotong sebelum selesai diproses. Hal ini penting karena hasil transkripsi akan digunakan sebagai dasar dalam proses evaluasi pelafalan.

4.2.2 Proses Pelatihan

Proses pelatihan model Whisper dari *tarteel-ai/whisper-tiny-ar-quran* dieksekusi pada lingkungan komputasi Google Colab dengan memanfaatkan akselerasi GPU NVIDIA L4 (Ada Lovelace). Model Whisper dari Tarteel AI ini merupakan model *pre-trained* yang sudah dilatih

pada *dataset* bahasa Arab Quran. Dalam proses pelatihan ini, model dilatih kembali dengan *domain* khusus yakni *dataset* bacaan doa anak yang telah dikumpulkan agar dapat mengenali karakteristik dan pola bacaan doa pada anak-anak. Pelatihan dilakukan dengan mengimplementasikan arsitektur HuggingFace Transformers, yang menyediakan kelas Seq2SeqTrainer sebagai *training loop* utama. Grafik *training* dan *validation loss* pada Gambar 4.15 menunjukkan pola pembelajaran model yang berjalan saat proses pelatihan.



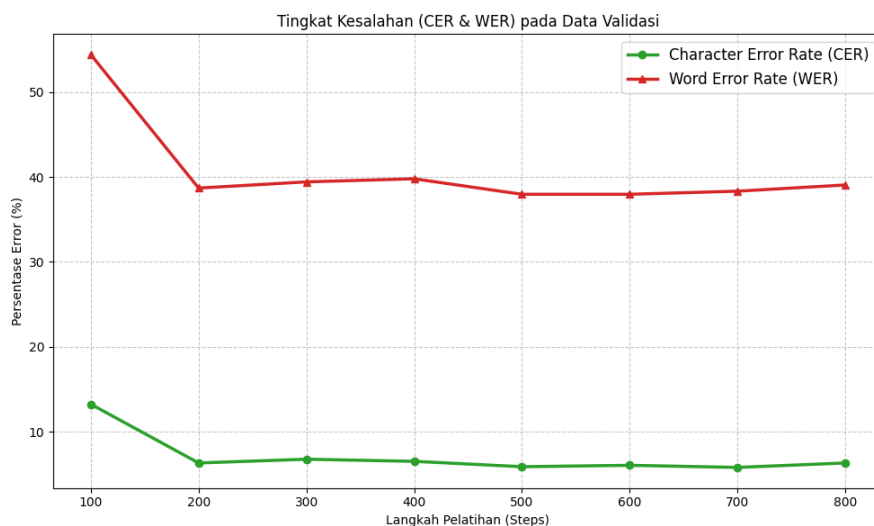
Gambar 4.15 Grafik *training loss* dan *validation loss* per epoch

Berdasarkan grafik pergerakan nilai kerugian (*loss*) pada Gambar 4.15, dapat diamati bahwa *training loss* (kurva biru) mengalami penurunan yang sangat tajam pada 200 langkah pertama. Penurunan tajam ini menunjukkan bahwa di awal pelatihan, model dengan cepat mulai mengenali pola dasar pelafalan doa anak, seperti variasi pelafalan, intonasi, serta struktur fonetik yang ada pada himpunan data latih. Sehingga proses adaptasi berlangsung cukup cepat dan efektif.

Sementara itu, titik konvergensi paling optimal tercapai di sekitar langkah pelatihan ke-200, di mana nilai *validation loss* (kurva oranye) menyentuh titik terendahnya dan bersinggungan dengan kurva *training loss*. Namun, saat memasuki rentang iterasi ke-300 hingga batas akhir pelatihan, yakni 800 langkah, mulai terlihat adanya kecenderungan *overfitting* dalam skala yang masih tergolong ringan. Hal ini ditandai dengan nilai *training loss* yang terus mengecil mendekati nol, sedangkan nilai *validation loss* justru melakukan sebaliknya, parameter ini perlahan mengalami tren kenaikan secara bertahap menuju angka 0.28.

Terjadinya pelebaran jarak antara kedua kurva tersebut menunjukkan bahwa pada tahap ini, model mulai terlalu ‘nyaman’ dengan data latihnya sendiri. Model mulai cenderung menghafal pola pada data latih yang sudah dikenal, tetapi mulai kehilangan kepekaan saat memahami variasi pelafalan baru dari data validasi. Kondisi ini pada akhirnya sedikit menurunkan tingkat fleksibilitas model ketika ditugaskan untuk mengenali data validasi yang belum pernah dipelajari sebelumnya. Meskipun demikian, karena sistem pelatihan telah dikonfigurasi untuk secara otomatis menyimpan *checkpoint* model terbaik berdasarkan metrik *Character Error Rate* (CER) terendah di akhir siklus, kenaikan *loss* di ujung iterasi ini dipastikan tidak akan mendegradasi kualitas model final yang diimplementasikan pada tahap selanjutnya. Stabilitas ini menjadi penting karena hasil transkripsi dari model akan digunakan sebagai *input* utama dalam algoritma evaluasi pelafalan berbasis perbandingan teks.

Selain evaluasi berdasarkan fungsi kerugian (*loss*), kinerja model juga diukur secara langsung menggunakan metrik evaluasi akhir dari *speech recognition*, yaitu *Character Error Rate* (CER) dan *Word Error Rate* (WER). Berdasarkan grafik pada Gambar 4.16, terlihat bahwa model berhasil menekan tingkat kesalahan karakter (CER) secara signifikan dari kisaran 13% pada awal pelatihan turun menjadi stabil di angka ~6% setelah melewati langkah ke-200. Penurunan yang linier ini juga diikuti oleh metrik WER yang stabil di kisaran 38% hingga 40%.



Gambar 4.16 Grafik WER dan CER pada data validasi selama proses pelatihan

Stabilitas kurva CER pada rentang iterasi 200 hingga 800 ini memberikan temuan yang sangat krusial terhadap kualitas model. Meskipun pada grafik sebelumnya (*Validation Loss*)

sempat terjadi indikasi pergeseran distribusi, grafik metrik kesalahan ini membuktikan bahwa fenomena tersebut tidak memberikan dampak signifikan terhadap kemampuan model dalam mengenali pelafalan anak. Dengan kata lain, model tetap mampu mengenali pola pelafalan anak dengan cukup konsisten pada tingkat *makharijul huruf* anak-anak.

Menariknya, meskipun kesalahan pada level kata (WER) masih cukup tinggi, kesalahan pada level karakter (CER) justru sudah rendah. Artinya, model sebenarnya sudah ‘mendengar dengan cukup tepat’, hanya saja masih ada pergeseran kecil pada huruf atau harakat yang membuat satu kata dianggap salah secara utuh. Hal ini disebabkan oleh sifat kalkulasi WER, di mana melesetnya deteksi pada satu karakter *makhraj* tunggal dalam sebuah kata akan secara otomatis menggugurkan kebenaran kata tersebut secara keseluruhan. Oleh karena itu, dalam konteks penelitian ini, CER menjadi metrik yang lebih relevan karena evaluasi pelafalan dilakukan pada tingkat karakter atau huruf hijaiyah, bukan hanya pada tingkat kata. Selain itu, dalam konteks penelitian ini, performa sistem koreksi yang akan dirancang tidak hanya diukur sebagai keberhasilan transkripsi dari model ASR semata, tetapi performa model menjadi fondasi utama dalam proses evaluasi pelafalan doa anak. Hasil transkripsi yang dihasilkan oleh model akan dibandingkan dengan teks referensi menggunakan algoritma *levenshtein similarity* untuk menghitung tingkat kesesuaian bacaan.

Penurunan nilai *Character Error Rate* (CER) dan *Word Error Rate* (WER) ini memiliki implikasi langsung terhadap kualitas evaluasi pelafalan. Semakin kecil kesalahan transkripsi yang dihasilkan model, maka semakin akurat pula perhitungan jarak *Levenshtein* dalam merepresentasikan kesalahan pelafalan yang sebenarnya dilakukan oleh anak. Sebaliknya, jika model menghasilkan transkripsi yang keliru secara signifikan, maka sistem evaluasi berpotensi memberikan skor yang tidak sesuai dengan kondisi pelafalan asli. Dengan demikian, peningkatan performa model ASR melalui proses *fine-tuning* menjadi langkah krusial untuk memastikan bahwa sistem tidak hanya ‘mendengar’, tetapi juga ‘memahami’ pelafalan anak secara tepat sebagai dasar evaluasi yang adil dan akurat.

4.2.3 Hasil Evaluasi Kinerja Model

Setelah proses pelatihan selesai, proses evaluasi kinerja model dilakukan secara objektif dengan memanfaatkan 36 data uji yang telah disiapkan dan belum dilibatkan sama sekali dalam proses pelatihan. Hal ini dilakukan untuk menguji kemampuan generalisasi model dalam mengenali variasi suara baru di luar data latih maupun data validasi. Untuk mengukur efektivitas metode *fine-tuning*, evaluasi dilakukan dalam dua tahap: (1) Evaluasi model hasil

fine-tuning, dan (2) Analisis komparatif terhadap model *baseline* (*whisper-tiny-ar-quran* versi asli yang tidak di *fine-tuning*).

Analisis Kinerja Model Setelah *Fine-Tuning*

Tabel 4.2 Hasil evaluasi kinerja model *fine-tuned*

Metrik	Nilai (%)
<i>Word Error Rate</i> (WER)	33.21
<i>Character Error Rate</i> (CER)	8.60

Berdasarkan Tabel 4.2, hasil pengujian menunjukkan bahwa model sudah cukup mampu mengenali pelafalan doa anak. Meskipun sekitar sepertiga kata masih belum tepat (WER 33.21%). Dalam konteks ASR untuk anak-anak, angka ini tergolong kompetitif dan dapat diterima. Namun, di sisi lain tingkat kesalahan pada level karakter (CER 8.6%) justru memperlihatkan bahwa sebagian besar kesalahan yang terjadi, ternyata berada pada level perbedaan yang kecil, seperti perbedaan huruf atau harakat, bukan kesalahan pada keseluruhan kata atau kalimat. Dengan kata lain, model sudah cukup ‘dekat’ dalam menangkap apa yang diucapkan oleh anak, meskipun masih terdapat beberapa perbedaan minor. Nilai CER yang rendah ini menjadi indikator bahwa hasil transkripsi model memiliki kedekatan yang tinggi dengan teks referensi. Hal ini sangat berpengaruh untuk memastikan transkripsi yang dihasilkan model yang akan bertugas menjadi ‘pendengar virtual’ dalam sistem sesuai dengan pelafalan anak, sehingga transkripsi ini dapat digunakan secara andal dalam proses perhitungan kesalahan pelafalan menggunakan algoritma pembandingan *levenshtein similarity* nantinya.

Analisis Komparatif Kinerja Model *Baseline* vs *Fine-Tuned*

Untuk memvalidasi urgensi *fine-tuning*, evaluasi kinerja model dilakukan dengan membandingkan performa model *whisper-tiny-ar-quran* sebelum dan sesudah proses *fine-tuning* menggunakan *dataset* bacaan doa anak. Perbandingan merujuk pada dua metrik utama evaluasi, yakni WER dan CER. Hasil lengkap perbandingan dapat dilihat pada Tabel 4.3.

Tabel 4.3 Perbandingan metrik evaluasi model *baseline* vs model *fine-tuned*

Model ASR	Karakteristik	WER (%)	CER (%)
Model <i>Baseline</i> (tarteel-ai/whisper-tiny-ar-quran)	Belum dilatih dengan suara anak	98.91	50.33
Model Usulan (<i>fine-tuned</i>)	Telah dilatih dengan suara anak	33.21	8.60
Peningkatan (Penurunan Kesalahan)		Turun 65.7%	Turun 41.73%

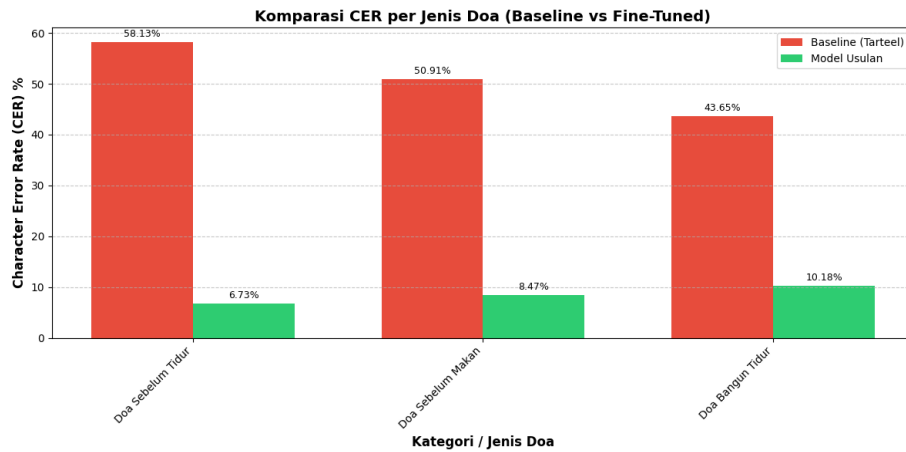
Hasil eksperimen tersebut menunjukkan perbedaan performa yang sangat signifikan. Model *baseline* yang belum dilatih menggunakan *dataset* doa anak, menunjukkan performa yang belum mampu mengenali pola pelafalan anak dengan baik, dengan WER yang mendekati 100%. Yang secara praktis menunjukkan bahwa model gagal dalam mengenali doa anak, ditandai oleh ketidaksesuaian hampir seluruh kata yang ditranskripsikan oleh model ASR dengan target atau *ground truth* dari transkrip asli audio. Hal ini dapat terjadi karena model awal tidak dibangun secara spesifik untuk domain doa anak yang sangat berbeda dengan bacaan Al-Qur'an yang menjadi fokus pelatihan model aslinya.

Sebaliknya, setelah dilakukan proses *fine-tuning*, peningkatan performa terjadi secara signifikan pada setiap metrik yang ada. WER turun dari 98.91% menjadi 33.21%, yang menandakan adanya peningkatan akurasi kata sebesar 65.7%. Penurunan CER dari 50.33% menjadi 8.60% juga menunjukkan bahwa kesalahan yang tersisa sebagian besar berupa perbedaan kecil pada huruf atau harakat, bukan kesalahan struktur kalimat secara keseluruhan. Temuan ini menunjukkan bahwa pendekatan *fine-tuning* merupakan langkah krusial dan efektif dalam mengadaptasi model ASR untuk kebutuhan domain spesifik seperti bacaan doa anak dan model yang telah di *fine-tune* sudah layak digunakan sebagai 'pendengar virtual' dalam aplikasi atau sistem koreksi bacaan doa anak.

Analisis Kinerja Berdasarkan Jenis Doa

Selain pengujian secara komprehensif, untuk memahami lebih dalam bagaimana model bekerja pada berbagai jenis bacaan, analisis performa lebih dalam dilakukan untuk melihat korelasi antara kompleksitas fonetik doa dengan tingkat kesalahan model berdasarkan jenis doa yang dilafalkan. Hal ini penting untuk ditinjau karena setiap doa memiliki karakteristik fonetik yang berbeda, baik dari segi panjang kalimat maupun kompleksitas *makhraj* huruf. Dengan analisis ini, dapat diidentifikasi apakah terdapat doa tertentu yang lebih sulit dikenali oleh model, yang pada akhirnya juga akan berdampak pada akurasi evaluasi pelafalan dalam sistem.

Perbandingan tingkat kesalahan pada level karakter atau *Character Error Rate* antara jenis doa, secara visual diilustrasikan dengan Gambar 4.17.



Gambar 4.17 Komparasi CER berdasarkan jenis doa

Berdasarkan Gambar 4.17, terlihat jelas bahwa model usulan secara konsisten berhasil menekan angka kesalahan pada seluruh jenis doa dibandingkan dengan model *baseline*. Penurunan paling mencolok terlihat pada jenis doa yang memiliki durasi pendek, khususnya doa sebelum tidur. Selanjutnya, untuk membedah lebih dalam terkait performa komputasional tiap kategori, rekapitulasi perbandingan tingkat kesalahan, baik CER maupun WER, antara model *baseline* dan model *fine-tuned* yang menjadi model usulan pada setiap jenis doa ditunjukkan dalam Tabel 4.4.

Tabel 4.4 Rekapitulasi perbandingan tingkat kesalahan antara model *baseline* vs *fine-tuned*

Jenis Doa	CER (%)		WER (%)	
	<i>Baseline</i>	<i>Fine-Tuned</i>	<i>Baseline</i>	<i>Fine-Tuned</i>
Doa Sebelum Tidur	58,13	6,73	106,06	22,73
Doa Bangun Tidur	43,65	10,18	97,53	29,63
Doa Sebelum Makan	50,91	8,47	96,06	40,94

Berdasarkan hasil pemetaan tersebut, model usulan yang merupakan model yang telah melalui proses *fine-tuning* terbukti mampu memberikan perbaikan akurasi yang masif dan konsisten di seluruh kategori uji. Kinerja paling optimal terjadi pada kategori doa sebelum tidur, yang menghasilkan tingkat kesalahan pada level kata atau WER terendah di angka 22,73%. Hal ini cukup masuk akal karena doa sebelum tidur memiliki struktur doa yang pendek dan berulang, khususnya pada pengulangan frasa '*bismika*' (بِسْمِكَ). Sehingga doa ini

menjadi lebih mudah ‘dipelajari’ oleh model dibandingkan doa dengan susunan kalimat yang lebih kompleks. Selain itu, komposisi fonetik pada doa ini minim huruf dengan makhraj dari tenggorokan bagian dalam, sehingga model lebih mudah melakukan prediksi yang akurat.

Pada kasus doa bangun tidur, terjadi lonjakan metrik WER menjadi 29,63%. Analisis terhadap hasil prediksi menunjukkan bahwa peningkatan WER ini disebabkan oleh kekeliruan model dalam mentranskripsikan huruf-huruf dengan *makhraj* lisan (lidah). Sebagai contoh, karakter Dzal (ذ) pada kata ‘*alladzii* (الَّذِي)’ anak-anak kerap kali melafalkan kata ini menjadi ‘*alladii*’ ataupun ‘*allazii*’ menggunakan huruf Dal (د) atau Zai (ز). Namun, model terjebak dalam dilema, di satu sisi ia mencoba untuk mengekstrak suara anak sesuai dengan apa yang di dapat, di sisi lain probabilitas *language* model bawaannya memaksa teks menjadi bahasa baku, sehingga ia mentranskripsikan karakter Zai atau Dal tadi ke karakter Dzal.

Sementara itu, WER tertinggi terlihat pada jenis doa sebelum makan. WER yang dihasilkan menyentuh angka 40,94%. Kompleksitas pada kategori ini dipengaruhi oleh tingginya variasi presisi artikulasi anak pada frasa ‘*razaqtanaa* (رَزَقْتَنَا)’ dan ‘*adzaaban-naar* (عَذَابُ النَّارِ)’. Pengucapan huruf Qaf (ق) sering kali tereduksi menjadi Kaf (ك) dalam rangkaian kata ‘*razaqtanaa* (رَزَقْتَنَا)’. Sedangkan pada rangkaian kata ‘*adzaaban-naar* (عَذَابُ النَّارِ)’, huruf Jim (ج) dan Dzal (ذ) bergeser menjadi Zai (ز). Kesalahan ini terjadi karena adanya ketidakstabilan artikulasi anak serta kesulitan mereka dalam melafalkan huruf-huruf sulit ini dan mendorong terjadinya variasi yang beragam pada pelafalan kata tersebut. Akhirnya, banyak tebakan kata yang meleset dari referensi asli oleh model dan berujung pada peningkatan kesalahan kata.

Implikasi dari temuan ini sangat penting dalam konteks sistem evaluasi pelafalan. Ketika model mampu menghasilkan transkripsi yang lebih mendekati ucapan asli anak, maka perhitungan kesalahan menggunakan algoritma *levenshtein* juga menjadi lebih merepresentasikan kesalahan pelafalan yang sebenarnya. Dengan kata lain, peningkatan performa model tidak hanya berdampak pada kualitas transkripsi, tetapi juga memengaruhi keadilan dan ketepatan sistem dalam menilai bacaan anak. Namun, secara keseluruhan, hasil ini menegaskan bahwa metode *fine-tuning* yang diusulkan valid dan tangguh dalam menangani berbagai jenis doa. Temuan ini menunjukkan bahwa kompleksitas fonetik dan variasi pelafalan anak memiliki pengaruh langsung terhadap performa model, yang pada akhirnya juga akan memengaruhi hasil evaluasi kesalahan bacaan dalam sistem yang dikembangkan.

4.3 Implementasi Algoritma Koreksi Pelafalan

Pada tahap implementasi, algoritma koreksi pelafalan dan sistem umpan balik yang telah dirancang, diintegrasikan ke dalam aplikasi berbasis Android untuk melakukan evaluasi bacaan doa anak secara otomatis. Dalam sistem ini, proses evaluasi tidak hanya berhenti pada pemberian skor akhir, tetapi dirancang untuk mampu mengidentifikasi letak kesalahan secara spesifik serta memberikan umpan balik yang bersifat edukatif.

Untuk mencapai tujuan tersebut, algoritma koreksi dirancang menggunakan pendekatan *multi-layer text processing*. Artinya, proses evaluasi tidak hanya dilakukan secara langsung pada tingkat kalimat, tetapi dipecah menjadi beberapa lapisan analisis, mulai dari tingkat kata hingga karakter. Pendekatan ini dipilih agar sistem tidak hanya mampu memberikan skor akhir, tetapi juga dapat menunjukkan bagian mana dari bacaan yang salah dan mengapa kesalahan tersebut terjadi. Secara umum, sistem bekerja dengan membandingkan teks hasil transkripsi dari model ASR dengan teks acuan doa menggunakan kombinasi metode penyelarasan kata, analisis karakter, validasi tajwid, serta perhitungan kemiripan berbasis *Levenshtein similarity*.

4.3.1 Ekstraksi Teks Transkripsi

Tahap pertama dalam implementasi algoritma koreksi yakni menyiapkan dua komponen utama dalam proses evaluasi, yakni teks hasil transkripsi dari model ASR dan teks acuan doa yang tersimpan dalam basis data. Teks hasil transkripsi merepresentasikan apa yang ‘didengar’ oleh model ASR dari pelafalan anak, sedangkan teks acuan menjadi standar bacaan yang benar. Proses implementasi pengambilan data transkripsi ini dapat dilihat pada Gambar 4.18.

```
// Menangkap hasil prediksi AI (Whisper)
val hasilWhisper = intent.getStringExtra("HASIL_REKAMAN_AI") ?: ""

// Menangkap seluruh data referensi Doa (Database JSON)
val dataDoaUtuh = if (Build.VERSION.SDK_INT >= 33) {
    intent.getParcelableExtra("DATA_DOA_UTUH", DoaModel::class.java)
} else {
    @Suppress("DEPRECATION")
    intent.getParcelableExtra("DATA_DOA_UTUH")
}
```

Gambar 4.18 Implementasi ekstraksi teks prediksi dan teks acuan

Pada implementasinya, hasil transkripsi dari model Whisper dikirimkan ke modul evaluasi dalam bentuk *string*, bersamaan dengan data doa yang berisi teks referensi serta metadata aturan tajwid. Kedua data ini kemudian menjadi fondasi utama dalam proses evaluasi pada tahap berikutnya.

4.3.2 Penyelarasan Kata (*Word-Level Alignment*)

Setelah teks diperoleh, tahap selanjutnya, merupakan implementasi dari lapisan pertama dalam arsitektur *multi-layer*, yakni *word-level alignment* atau penyelarasan kata. Pada tahap ini, teks hasil transkripsi dan teks acuan dipecah menjadi kumpulan kata untuk dibandingkan secara berurutan. Langkah ini menjadi penting karena kesalahan pelafalan doa oleh anak tidak selalu terjadi pada seluruh kalimat, melainkan seringkali hanya pada kata-kata tertentu. Dengan melakukan penyelarasan pada tingkat kata, sistem dapat mengisolasi kesalahan secara lebih presisi tanpa memengaruhi bagian bacaan lain yang sudah benar.

Dalam tahap ini, sistem memecah teks acuan dan teks hasil transkripsi menjadi larik kata untuk menentukan pasangan kata mana yang bersesuaian. Proses pemecahan kata dilakukan dengan menggunakan ekspresi reguler sebagaimana yang dapat dilihat dalam Gambar 4.19.

```
val targetWords = targetText.split("\\s+").toRegex().filter { it.isNotBlank() }
val actualWords = teksWhisperUtuh.split("\\s+").toRegex().filter { it.isNotBlank() }
```

Gambar 4.19 Proses pemecahan kata dengan menggunakan ekspresi reguler

Setelah proses tokenisasi, sistem melakukan pencocokan kata satu per satu menggunakan pendekatan sekuensial dengan indeks `aIndex` sebagai penunjuk posisi pada teks hasil transkripsi. Dalam proses ini, setiap kata diklasifikasikan ke dalam tiga kategori:

- CORRECT, jika kata sesuai dengan teks acuan
- WRONG_MAKHRAJ, jika kata mirip tetapi tidak identik (tingkat kemiripan >50%)
- MISSING, jika kata tidak ditemukan atau terlewat

Klasifikasi ini membantu sistem dalam membedakan apakah kesalahan terjadi karena pelafalan yang kurang tepat, atau karena ada bagian bacaan yang terlewat atau tidak diucapkan sama sekali oleh anak. Proses dalam menentukan status pasangan diimplementasikan dengan logika kode sebagaimana yang ditunjukkan pada Gambar 4.20.

```
if(aIndex < actualWords.size) {
    val aWord = actualWords[aIndex]
    if (tWord == aWord) {
        status = "CORRECT"
        pairedActualWord = aWord
        aIndex++
    } else {
        val isMirip = isKataBerpasangan(tWord, aWord)
        if (isMirip) {
```

```

        status = "WRONG_MAKHRAJ"
        pairedActualWord = aWord
        aIndex++
    } else {
        var isNextTargetMatched = false
        if (tIndex + 1 < targetWords.size) {
            if (isKataBerpasangan(targetWords[tIndex + 1], aWord))
isNextTargetMatched = true
        }
        if (isNextTargetMatched) {
            status = "MISSING"
        } else {
            status = "WRONG_MAKHRAJ"
            pairedActualWord = aWord
            aIndex++
        }
    }
}
}
}

```

Gambar 4.20 Proses memasangkan kata dan menentukan status pasangan

Selain itu, pada tahap ini, sistem juga menyimpan posisi setiap kata dalam bentuk indeks absolut untuk menjaga konsistensi posisi karakter pada tahap berikutnya. Sehingga, kesalahan pada satu kata tidak akan memengaruhi evaluasi kata lain.

4.3.3 Penyelarasan Karakter dengan Algoritma *Two-Pointer*

Setelah pasangan kata diperoleh, analisis dilanjutkan ke tingkat karakter untuk mendeteksi kesalahan secara lebih detail. Pada tahap ini digunakan pendekatan algoritma *two-pointer lookahead*, di mana sistem membandingkan karakter dari teks acuan dan teks hasil transkripsi secara berurutan. Berbeda dengan pendekatan sederhana yang langsung menandai setiap perbedaan sebagai kesalahan, metode ini memaksa sistem untuk mencoba ‘memahami’ kemungkinan adanya variasi pelafalan oleh anak, seperti penambahan atau pengurangan huruf. Oleh karena itu, digunakan pendekatan *lookahead*, di mana sistem melihat beberapa karakter ke depan untuk mencari kemungkinan kecocokan yang tertunda. Implementasi utama algoritma ini dilakukan dengan menggunakan rangkaian kode yang dapat dilihat pada Gambar 4.21.

```

while (tIdx < tWord.length && aIdx < pairedActualWord.length) {
    val tChar = tWord[tIdx]
    val aChar = pairedActualWord[aIdx]

    if (tChar == aChar) { // Jika karakter cocok, pointer maju bersama
        tIdx++
        aIdx++
    } else {
        // Lookahead pada kata acuan (mencari huruf yang hilang)
        var foundInTarget = false
        for (offset in 1..3) {
            if (tIdx + offset < tWord.length && tWord[tIdx + offset] == aChar) {
                for (i in 0 until offset) indexErrorAbsolut.add(wordStartIndex + tIdx + i)
            }
        }
    }
}

```

```

        tIdx += offset
        foundInTarget = true
        break
    }
}

// Lookahead pada kata prediksi (mengabaikan huruf sisipan tambahan)
if (!foundInTarget) {
    var foundInActual = false
    for (offset in 1..3) {
        if (aIdx + offset < pairedActualWord.length && tChar == pairedActualWord[aIdx
+ offset]) {
            aIdx += offset
            foundInActual = true
            break
        }
    }

    // Jika benar-benar salah, simpan indeks error
    if (!foundInActual) {
        indexErrorAbsolut.add(wordStartIndex + tIdx)
        tIdx++
        aIdx++
    }
}
}
}

```

Gambar 4.21 Implementasi algoritma *two-pointer lookahead* untuk penyelarasan karakter

Pendekatan ini penting untuk menghindari efek domino kesalahan, yaitu kondisi di mana satu kesalahan kecil menyebabkan seluruh sisa kata dianggap salah. Karakter yang terdeteksi sebagai kesalahan disimpan dalam bentuk indeks absolut yang kemudian digunakan dalam proses validasi aturan tajwid. Dengan strategi ini, sistem mampu menjaga evaluasi tetap stabil dan hanya menandai kesalahan yang benar-benar relevan.

4.3.4 Evaluasi Checkpoint Tajwid

Setelah lokasi kesalahan pada tingkat karakter diperoleh, tahap selanjutnya adalah memvalidasi kesalahan tersebut terhadap aturan tajwid. Dalam sistem ini, aturan tajwid direpresentasikan dalam bentuk *checkpoint* berbasis indeks, yang menunjukkan posisi huruf tertentu beserta jenis aturan dan umpan balik yang sesuai. Proses validasi ini dilakukan dengan memeriksa apakah indeks kesalahan berada dalam rentang *hitbox checkpoint* atau tidak (lihat Gambar 4.22).

```

val errorsInHitbox = indexErrorAbsolut.filter { it in hitBoxStart..hitBoxEnd }
val isHurufSalah = errorsInHitbox.isNotEmpty()

```

Gambar 4.22 Proses validasi indeks kesalahan terhadap *hitbox checkpoint* tajwid

Sistem akan memeriksa apakah kesalahan yang terdeteksi berada dalam rentang *checkpoint* tertentu. Jika ya, maka sistem akan mengklasifikasikan kesalahan tersebut sebagai pelanggaran tajwid spesifik (*makhraj* atau *mad*). Khusus untuk evaluasi *mad* (panjang pendek bacaan), sistem menggunakan pendekatan ortografis berbasis rasio panjang karakter antara teks acuan dan teks hasil transkripsi.

```

if (!WhisperHelper.isMadSesuaiEjaan(tWord, pairedActualWord)) {
    isDurasiKurang = true
}

//Fungsi isMadSesuaiEjaan() dalam class WhisperHelper
fun isMadSesuaiEjaan(kataTarget: String, kataAI: String): Boolean {
    // Hilangkan harakat agar kita menghitung murni huruf dasar (konsonan + mad)
    val bersihTarget = kataTarget.replace(Regex("[\\u064B-\\u065F]"), "")
    val bersihAI = kataAI.replace(Regex("[\\u064B-\\u065F]"), "")

    // Cegah pembagian dengan nol
    if (bersihTarget.isEmpty()) return true

    val rasioPanjang = bersihAI.length.toDouble() / bersihTarget.length.toDouble()
    return rasioPanjang >= 1.0
}

```

Gambar 4.23 Fungsi evaluasi mad berbasis rasio panjang karakter

Berdasarkan fungsi yang ditunjukkan oleh Gambar 4.23, jika panjang hasil bacaan lebih pendek dari yang seharusnya (rasio kurang dari 1.0), maka sistem akan mengindikasikan adanya kekurangan durasi dalam pelafalan yang dilakukan oleh pengguna. Selanjutnya, untuk semua kesalahan yang tidak berada dalam *checkpoint*, sistem tetap akan memberikan umpan balik khusus sebagai kesalahan pelafalan umum, sehingga tidak ada kesalahan yang diabaikan. Dengan pendekatan ini, sistem tidak hanya mendeteksi kesalahan secara umum, tetapi juga mampu mengaitkannya dengan aturan tajwid tertentu. Hal ini membuat umpan balik yang diberikan menjadi lebih edukatif, karena pengguna tidak hanya mengetahui bahwa bacaannya salah, tetapi juga memahami jenis kesalahan yang dilakukan.

4.3.5 Perhitungan Skor Kemiripan dengan *Levenshtein Distance*

Selain analisis detail pada tingkat kata dan karakter, sistem juga menghitung skor dan tingkat kemiripan bacaan secara keseluruhan menggunakan perhitungan *Levenshtein Similarity*. Metode ini digunakan untuk mengukur tingkat kesesuaian hasil transkripsi model ASR terhadap teks referensi doa sebagai dasar penentuan skor evaluasi dan predikat kualitas bacaan anak. Banyaknya perbedaan karakter tersebut dihitung dengan mengkalkulasi jumlah minimum perubahan yang diperlukan untuk mengubah teks hasil transkripsi menjadi teks

acuan. Pendekatan ini dipilih karena sistem tidak hanya dirancang untuk mendeteksi keberadaan kata tertentu, tetapi juga mengevaluasi ketepatan dan kesesuaian struktur bacaan secara utuh terhadap teks referensi yang seharusnya dilafalkan oleh anak.

Implementasi proses perhitungan jarak edit ini dilakukan menggunakan fungsi *Levenshtein Distance* berbasis *dynamic programming* yang dikembangkan pada aplikasi Android menggunakan bahasa pemrograman Kotlin. Fungsi ini bertugas menghitung jumlah minimum operasi edit yang diperlukan untuk mengubah teks hasil transkripsi menjadi identik dengan teks referensi. Operasi edit yang digunakan terdiri dari *insertion* (penambahan karakter), *deletion* (penghapusan karakter), dan *substitution* (penggantian karakter). Implementasi fungsi perhitungan jarak edit dan kalkulasi skor *similarity* pada sistem dijalankan dengan rangkaian fungsi dan kode yang dapat dilihat pada Gambar 4.24.

```
private fun hitungLevenshteinDistance(s1: String, s2: String): Int {
    val dp = Array(s1.length + 1) { IntArray(s2.length + 1) }
    for (i in 0..s1.length) dp[i][0] = i
    for (j in 0..s2.length) dp[0][j] = j

    for (i in 1..s1.length) {
        for (j in 1..s2.length) {
            val cost = if (s1[i - 1] == s2[j - 1]) 0 else 1
            dp[i][j] = minOf(
                dp[i - 1][j] + 1,          // Deletion (Penghapusan)
                dp[i][j - 1] + 1,          // Insertion (Penyisipan)
                dp[i - 1][j - 1] + cost    // Substitution (Penggantian)
            )
        }
    }
    return dp[s1.length][s2.length]
}

private fun hitungPersentaseSkor(target: String, aktual: String): Int {
    val tBersih = target.replace(Regex("\\s+"), " ").trim()
    val aBersih = aktual.replace(Regex("\\s+"), " ").trim()

    val maxLength = Math.max(tBersih.length, aBersih.length)
    if (maxLength == 0) return 0 // Mencegah error bagi nol

    val distance = hitungLevenshteinDistance(tBersih, aBersih)
    val similarity = (1.0 - (distance.toDouble() / maxLength)) * 100

    // Dibulatkan dan dipastikan nilainya 0 - 100
    return Math.round(similarity).toInt().coerceIn(0, 100)
}
```

Gambar 4.24 Implementasi fungsi *Levenshtein Distance* dan perhitungan *similarity* (skor kemiripan) pada sistem

Berdasarkan implementasi pada Gambar 4.24, sistem terlebih dahulu melakukan normalisasi teks dengan menghapus spasi berlebih serta karakter kosong di awal maupun di akhir kalimat menggunakan proses *trimming*. Namun, spasi antar kata tetap dipertahankan dan

dihitung sebagai bagian dari *string* karena merepresentasikan struktur dan urutan bacaan antar kata dalam doa. Setelah tahap normalisasi dilakukan, sistem menghitung nilai *Levenshtein Distance* antara teks referensi dan hasil transkripsi model ASR. Nilai jarak edit tersebut kemudian dinormalisasi menggunakan panjang *string* maksimum agar menghasilkan skor *similarity* dalam rentang 0% hingga 100%.

Untuk memperjelas mekanisme perhitungan *similarity*, berikut disajikan contoh proses evaluasi antara teks referensi dan hasil transkripsi model ASR menggunakan metode *Levenshtein Similarity*.

Tabel 4.5 Contoh kasus pengujian perhitungan *Levenshtein Similarity*

Komponen	Nilai
Teks referensi (ref)	اللَّهُمَّ بَارِكْ لَنَا فِيْمَا رَزَقْتَنَا وَقِنَا عَذَابَ النَّارِ
Hasil transkripsi (pred)	اللَّهُمَّ بَارِكْ لَنَا فِيْمَ رَزَقْتَنَا بَعْدَ مَنَابَا لَاللَّهُمَّ لَذِي وَهُمْ بَرِيْتْ وَقِنَا عَذَابَ النَّارِ
Jenis perbedaan	Penambahan, penggantian, dan penghapusan karakter
Panjang ref	61 karakter
Panjang pred	115 karakter
Jumlah operasi edit	54 karakter
Panjang maksimum <i>String</i>	115 karakter

Pada contoh tersebut, teks referensi doa yang seharusnya dibaca pengguna, sebagaimana pada Tabel 4.5, yakni “اللَّهُمَّ بَارِكْ لَنَا فِيْمَا رَزَقْتَنَا وَقِنَا عَذَابَ النَّارِ”. Namun, hasil transkripsi model ASR masih mengandung tambahan frasa “بَعْدَ مَنَابَا لَاللَّهُمَّ لَذِي وَهُمْ بَرِيْتْ” yang tidak terdapat pada teks referensi. Selain itu, terdapat juga penggantian dan penghapusan karakter akibat kesalahan pelafalan huruf dan pemendekan bacaan pada kata “فِيْمَا”. Meskipun sebagian besar frasa utama berhasil dikenali dengan benar oleh sistem, tambahan kata, penggantian, dan penghapusan karakter tersebut dihitung sebagai suatu bentuk kesalahan dalam proses perhitungan *Levenshtein Distance*. Secara matematis, perhitungan nilai *similarity* untuk contoh kasus pada Tabel 4.5 disimulasikan dengan perhitungan berikut:

$$Similarity = 1 - \frac{54}{115} = 0,5304 = 53,04\% \approx 53\%$$

Hasil perhitungan tersebut menunjukkan bahwa sistem tidak hanya mengevaluasi keberhasilan pendeteksian sebagian kata target, tetapi juga mengevaluasi kesesuaian keseluruhan struktur bacaan terhadap teks referensi. Oleh karena itu, penambahan karakter, penghapusan karakter, maupun penggantian karakter tetap menghasilkan penalti *similarity* meskipun sebagian frasa

utama berhasil dikenali dengan benar. Untuk memvalidasi implementasi algoritma pada aplikasi, hasil perhitungan manual tersebut kemudian dibandingkan dengan keluaran sistem pada media uji aplikasi *mobile*. Visualisasi hasil evaluasi *similarity* dan predikat kualitas bacaan ditunjukkan pada Gambar 4.25.



Gambar 4.25 Hasil evaluasi *similarity* bacaan pada aplikasi *mobile*

Berdasarkan hasil pada Gambar 4.25, nilai *similarity* yang dihasilkan sistem sebesar 53% konsisten dengan hasil perhitungan manual menggunakan persamaan *Levenshtein Similarity*. Kesesuaian hasil ini menunjukkan bahwa implementasi fungsi perhitungan *edit distance* dan normalisasi *similarity* pada aplikasi telah berjalan sesuai dengan rancangan metodologi yang disusun. Nilai *similarity* yang dihasilkan dari perhitungan *levenshtein distance* ini tidak hanya ditampilkan sebagai angka persentase, tetapi juga digunakan sebagai dasar dalam pengambilan keputusan evaluasi pelafalan. Untuk menghindari interpretasi yang bersifat subjektif, sistem menerapkan ambang batas (*threshold*) yang telah ditentukan pada tahap perancangan (Bab 3).

Berdasarkan rentang nilai tersebut, kualitas bacaan diklasifikasikan ke dalam empat kategori utama, yakni *Mumtaz* (90-100%), *Jayyid* (80-89%), *Maqbul* (65-79%), dan Perlu Bimbingan sesuai dengan tabel yang telah dirancang pada Bab 3. Agar hasil evaluasi lebih mudah dipahami oleh pengguna, khususnya anak-anak dan orang tua, setiap rentang skor tidak hanya ditampilkan dalam bentuk predikat seperti *Mumtaz*, *Jayyid*, atau *Maqbul*, tetapi juga diinterpretasikan ke dalam kategori tingkat kelancaran bacaan. Secara praktis, interpretasi tersebut dijelaskan secara berikut:

- Skor 90-100% (*Mumtaz*) menunjukkan bahwa anak sudah sangat lancar dalam melafalkan doa, dengan kesalahan yang sangat minimal atau hampir tidak ada.

- Skor 80-89% (*Jayyid*) menunjukkan bahwa bacaan sudah lancar, namun masih terdapat beberapa kesalahan kecil yang perlu diperbaiki.
- Skor 65-79% (*Maqbul*) menunjukkan bahwa bacaan berada pada kategori cukup, tetapi masih membutuhkan latihan agar pelafalan menjadi lebih tepat.
- Skor < 65% (Perlu Bimbingan) menunjukkan bahwa anak masih perlu belajar dan berlatih kembali karena terdapat banyak kesalahan dalam pelafalan.

Dengan interpretasi ini, sistem tidak hanya memberikan nilai dalam bentuk angka atau istilah predikat, tetapi juga memberikan pemahaman yang lebih intuitif mengenai tingkat kemampuan pelafalan anak. Algoritma koreksi pun tidak hanya mampu mengukur tingkat kemiripan bacaan secara umum, tetapi juga mampu mengidentifikasi letak kesalahan pelafalan secara spesifik pada tingkat kata dan karakter, sehingga mendukung penyediaan umpan balik yang lebih terarah dan edukatif.

4.4 Implementasi dan Integrasi Sistem

Tahap implementasi dan integrasi sistem merupakan proses penerjemahan seluruh rancangan yang telah dibangun pada tahap sebelumnya ke dalam bentuk aplikasi yang dapat digunakan secara langsung oleh pengguna. Pada penelitian ini, sistem koreksi pelafalan doa anak tidak hanya berhenti pada pengembangan model ASR dan algoritma evaluasi, tetapi juga diimplementasikan secara utuh dalam aplikasi berbasis Android sebagai media uji coba. Integrasi sistem ini mencakup beberapa komponen utama, yakni konversi model hasil *fine-tuning* agar dapat dijalankan pada perangkat *mobile*, penghubungan antara model dengan aplikasi melalui *native interface*, pengolahan audio secara *real-time*, serta visualisasi hasil evaluasi dalam bentuk yang mudah dipahami oleh anak-anak.

4.4.1 Konversi dan Optimasi Model

Model Whisper hasil *fine-tuning* pada tahap sebelumnya masih berada dalam format penyimpanan bawaan dari ekosistem HuggingFace berupa berkas *.safetensors* dengan tingkat presisi komputasi *floating-point* (FP16). Meskipun format ini optimal untuk pelatihan, ukuran yang dimilikinya relatif besar dan tidak efisien jika dijalankan langsung pada perangkat Android yang memiliki keterbatasan memori. Oleh karena itu, perlu dilakukan proses konversi dan optimasi model agar lebih ringan dan dapat digunakan secara luring (*offline*).

Pada penelitian ini, model dikonversi model ke dalam format GGML menggunakan skrip `convert-h5-to-ggml.py` sehingga dapat dijalankan menggunakan mesin inferensi `whisper.cpp`. Selain itu, untuk mengoptimalkan kinerja dan meminimalkan konsumsi RAM pada perangkat Android, proses ini juga menerapkan teknik kuantisasi 4-bit integer (Q4_0). Tahapan eksekusi konversi ini diimplementasikan menggunakan Google Colab dengan rangkaian kode yang ditunjukkan dalam Gambar 4.26.

```
# Mendefinisikan path direktori model fine-tuned
model_drive = "/content/drive/MyDrive/Proyek_ASR2/whisper-tiny-tarteel-augmented2"

# Mengeksekusi konversi dari format PyTorch/H5 menjadi GGML
!python3 whisper.cpp/models/convert-h5-to-ggml.py {model_drive} whisper.cpp .

# Memodifikasi penamaan berkas biner hasil konversi
!mv ggml-model.bin ggml-model-tiny-augmented2.bin
```

Gambar 4.26 Kode eksekusi perintah konversi model ke dalam format GGML

Hasil akhir dari proses ini adalah sebuah berkas model biner (.bin) yang jauh lebih ringkas dan siap digunakan pada perangkat Android. Secara praktis, optimasi ini memungkinkan aplikasi berjalan dengan lebih efisien, baik dari sisi penggunaan memori maupun kecepatan inferensi, sehingga pengalaman pengguna tetap responsif. Perbandingan efisiensi metrik beban komputasi sebelum dan sesudah proses optimasi dirangkum pada Tabel 4.6.

Tabel 4.6 Perbandingan metrik beban penyimpanan model sebelum dan sesudah optimasi

Parameter	Model Asal Hasil <i>Fine-Tuning</i>	Model Optimasi (GGML)
Format Bobot Utama	.safetensors (HuggingFace)	.bin (GGML Q4_0)
Jumlah File	Beragam (1 Folder)	1 file biner independen
Total Ukuran Penyimpanan	~147 MB	~74 MB
Kebutuhan Komputasi RAM	Presisi tinggi (<i>Float 16-bit</i>)	Presisi rendah (<i>Integer 4-bit</i>)

4.4.2 Integrasi Model dan Algoritma via Java Native Interface (JNI)

Salah satu tantangan utama dalam implementasi sistem ini adalah perbedaan lingkungan eksekusi antara aplikasi Android dan mesin inferensi model Whisper. Aplikasi Android secara umum dibangun menggunakan Kotlin yang berjalan di atas *Java Virtual Machine* (JVM). Di sisi lain, mesin inferensi model ASR (`whisper.cpp`) beroperasi secara independen menggunakan bahasa pemrograman C/C++. Untuk menjembatani perbedaan tersebut, digunakan *Java Native Interface* (JNI) yang memungkinkan aplikasi Kotlin memanggil fungsi-fungsi *native* dari C++. Melalui pendekatan ini, proses inferensi model dapat dijalankan secara langsung di perangkat

tanpa perlu koneksi internet. Implementasi pemuatan *library* dan deklarasi fungsi *native* pada antarmuka kotlin ditunjukkan pada Gambar 4.27.

```
class WhisperLib {
    companion object {
        init {
            // Memuat shared library "libwhisper.so" hasil kompilasi NDK
            System.loadLibrary("whisper")
        }
    }

    // Menginisialisasi model Whisper
    external fun initContext(modelPath: String): Long

    // Mengeksekusi proses transkripsi audio
    external fun fullTranscribe(context: Long, numThreads: Int, audioData: FloatArray):
    String

    // Membebaskan alokasi memori setelah selesai digunakan untuk mencegah memory leak
    external fun freeContext(context: Long)
}
```

Gambar 4.27 Deklarasi antarmuka JNI pada lapisan Kotlin

Pada sisi Kotlin, fungsi *native* dideklarasikan menggunakan kata kunci *external*, sedangkan implementasi logika komputasi sesungguhnya ditulis pada berkas C++ menggunakan makro pengikatan *JNIEXPORT* dan *JNICALL*. Alur kerjanya fungsi C++ bertugas menerima kiriman data audio dalam bentuk *array float* yang dikirim oleh Kotlin, lalu model Whisper mengeksekusi proses pengenalan suara (*speech recognition*). Setelah teks transkripsi berhasil diekstrak oleh mesin Whisper, hasil transkripsi dikembalikan ke Kotlin kembali dalam bentuk teks untuk dievaluasi lebih lanjut oleh algoritma koreksi. Potongan kode implementasi pengikatan pada lapisan *native* C++ ini ditunjukkan pada Gambar 4.28.

```
JNIEXPORT jstring JNICALL
Java_com_example_temanibadahsikecil_ui_adapter_WhisperLib_fullTranscribe(
    JNIEnv *env, jobject thiz, jlong context_ptr, jint num_threads, jfloatArray
    audio_data) {

    struct whisper_context *context = (struct whisper_context *) context_ptr;
    jfloat *audio_data_arr = (*env)->GetFloatArrayElements(env, audio_data, NULL);
    const jsize audio_data_length = (*env)->GetArrayLength(env, audio_data);

    // Konfigurasi parameter inferensi Whisper (Greedy Sampling)
    struct whisper_full_params params = whisper_full_default_params(WHISPER_SAMPLING_GREEDY);
    params.print_realtime = true;
    params.print_progress = false;
    params.print_timestamps = true;
    params.print_special = false;
    params.translate = false;
    params.language = "ar";
    params.n_threads = num_threads;
    params.offset_ms = 0;
```

```

params.no_context = true;
params.temperature = 0.0f;
params.beam_search.beam_size = 1;
params.suppress_blank = false;
params.single_segment = false;
params.no_speech_thold = 0.6f;
params.entropy_thold = 2.4f;

whisper_reset_timings(context);

LOGI("About to run whisper_full");
if (whisper_full(context, params, audio_data_arr, audio_data_length) != 0) {
    LOGI("Failed to run the model");
    (*env)->ReleaseFloatArrayElements(env, audio_data, audio_data_arr, JNI_ABORT);
    return (*env)->NewStringUTF(env, "Error: Gagal transkripsi");
} else {
    whisper_print_timings(context);
}

int n_segments = whisper_full_n_segments(context);

if (n_segments == 0) {
    return (*env)->NewStringUTF(env, "AI tidak mendengar suara");
}

char result_text[4096] = {0};
for (int i = 0; i < n_segments; ++i) {
    const char *text = whisper_full_get_segment_text(context, i);

    if (strlen(result_text) + strlen(text) < 4000) {
        strcat(result_text, text);
    }
}

(*env)->ReleaseFloatArrayElements(env, audio_data, audio_data_arr, JNI_ABORT);
return (*env)->NewStringUTF(env, result_text);
}

```

Gambar 4.28 Implementasi pengaturan parameter dan eksekusi pada lapisan *Native C++*

Integrasi ini menjadi komponen kunci dalam sistem karena memungkinkan model ASR berfungsi sebagai ‘pendengar virtual’ yang bekerja secara *real-time* di dalam aplikasi.

4.4.3 Perekaman dan Pemrosesan Audio di Latar Belakang

Agar model dapat bekerja secara optimal, audio yang direkam harus sesuai dengan spesifikasi yang dibutuhkan oleh Whisper. Sebagaimana yang telah disebutkan sebelumnya, model arsitektur Whisper membutuhkan masukan audio dengan laju sampel sebesar 16.000 Hz, saluran (*channel*) mono, serta format numerik presisi *floating-point* (nilai berkisar -1.0 hingga 1.0). Maka, untuk memenuhi kebutuhan tersebut, sistem memanfaatkan kelas `AudioRecord` bawaan dari Android. Proses perekaman dan ekstraksi data *waveform* dijalankan pada *background thread* terpisah dari antarmuka pengguna utama, agar tidak mengganggu tampilan aplikasi, sehingga pengguna tetap mendapatkan pengalaman yang lancar saat

melakukan perekaman. Gambar 4.29 menunjukkan implementasi logika perekaman dan normalisasi audio pada sistem aplikasi ‘Teman Ibadah Si Kecil’.

```
class AudioRecorderHelper(private val context: Context) {

    // Konfigurasi parameter audio wajib untuk model Whisper
    private val SAMPLE_RATE = 16000
    private val CHANNEL_CONFIG = AudioFormat.CHANNEL_IN_MONO
    private val AUDIO_FORMAT = AudioFormat.ENCODING_PCM_16BIT
    private val BUFFER_SIZE = AudioRecord.getMinBufferSize(SAMPLE_RATE, CHANNEL_CONFIG,
        AUDIO_FORMAT)

    private val audioDataList = mutableListOf<Float>()
    private var isRecording = false

    @SuppressWarnings("MissingPermission")
    fun startRecording() {
        recorder = AudioRecord(
            MediaRecorder.AudioSource.MIC, SAMPLE_RATE, CHANNEL_CONFIG, AUDIO_FORMAT, BUFFER_SIZE
        )
        recorder?.startRecording()
        isRecording = true

        // Menjalankan proses ekstraksi di dalam background thread
        Thread {
            val buffer = ShortArray(BUFFER_SIZE / 2)
            while (isRecording) {
                val read = recorder?.read(buffer, 0, buffer.size) ?: 0
                if (read > 0) {
                    // Konversi dan normalisasi format PCM 16-bit menjadi Float (-1.0f hingga 1.0f)
                    for (i in 0 until read) {
                        audioDataList.add(buffer[i] / 32768.0f)
                    }
                }
            }
        }.start()
    }

    fun stopRecording(): FloatArray {
        isRecording = false
        recorder?.stop()
        recorder?.release()

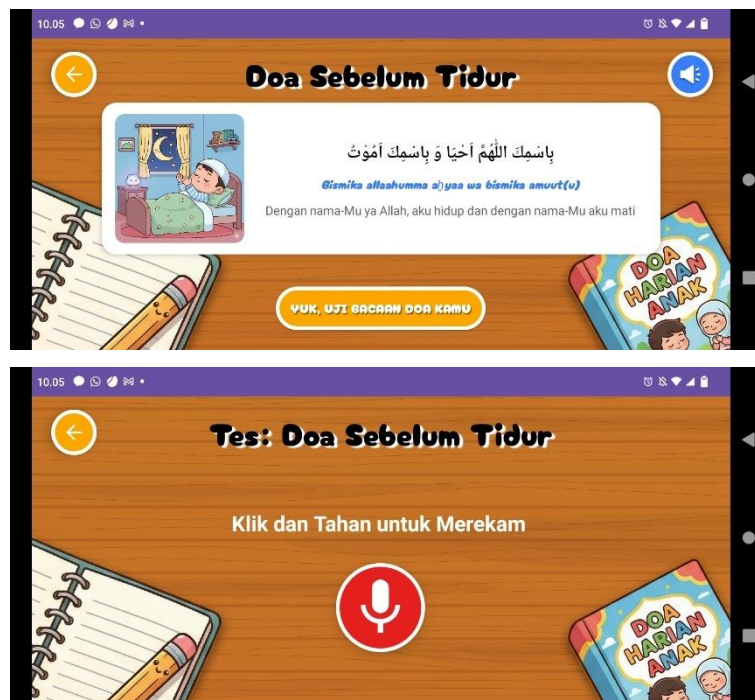
        // Mengembalikan larik data audio mentah yang siap diumpankan ke JNI C++
        return audioDataList.toFloatArray()
    }
}
```

Gambar 4.29 Implementasi kelas perekam audio dan konversi *floating-point*

Audio yang direkam dalam format PCM 16-bit kemudian dikonversi menjadi nilai *float* dalam rentang -1.0 hingga 1.0 sebelum dikirim ke model. Pendekatan ini dilakukan secara langsung di memori tanpa menyimpan file sementara, sehingga lebih efisien dan memiliki latensi yang rendah. Dengan mekanisme ini, sistem mampu menangkap suara anak secara *real-time* dan langsung memprosesnya menjadi teks yang siap dievaluasi.

4.4.4 Visualisasi Sistem sebagai Wadah Uji Coba

Untuk mendukung tahap pengujian sistem, tahap akhir dari integrasi sistem adalah menyajikan hasil evaluasi dalam bentuk yang mudah dipahami oleh pengguna, khususnya anak-anak. Oleh karena itu, antarmuka aplikasi dirancang untuk memfasilitasi interaksi perekaman audio dan visualisasi hasil komputasi algoritma evaluasi kepada pengguna. Tahap pengujian dimulai ketika pengguna memilih kategori doa dan menekan tombol rekam. Aksi ini memicu fungsi perekaman audio yang bekerja di latar belakang untuk mulai menangkap suara. Visualisasi antarmuka saat pengguna telah memilih kategori doa dan proses tes pelafalan, di mana sistem sedang menangkap dan memproses audio ditunjukkan pada Gambar 4.30.



Gambar 4.30 Antarmuka tampilan doa terpilih (atas) dan proses tes pengujian sistem saat proses penangkapan audio (bawah)

Setelah proses transkripsi dan evaluasi selesai, hasil pelafalan divisualisasikan dalam bentuk teks berwarna Huruf yang dilafalkan dengan benar ditandai dengan warna hijau, sedangkan kesalahan pelafalan atau ketidaktepatan durasi panjang pendek huruf ditandai dengan warna merah. Pendekatan visual ini membantu anak untuk langsung mengetahui bagian mana yang perlu diperbaiki.

Selain itu, setiap kata juga dilengkapi dengan fitur interaktif berupa *pop-up feedback* yang dapat diakses dengan menekan kata tersebut. Umpan balik ini berisi penjelasan sederhana

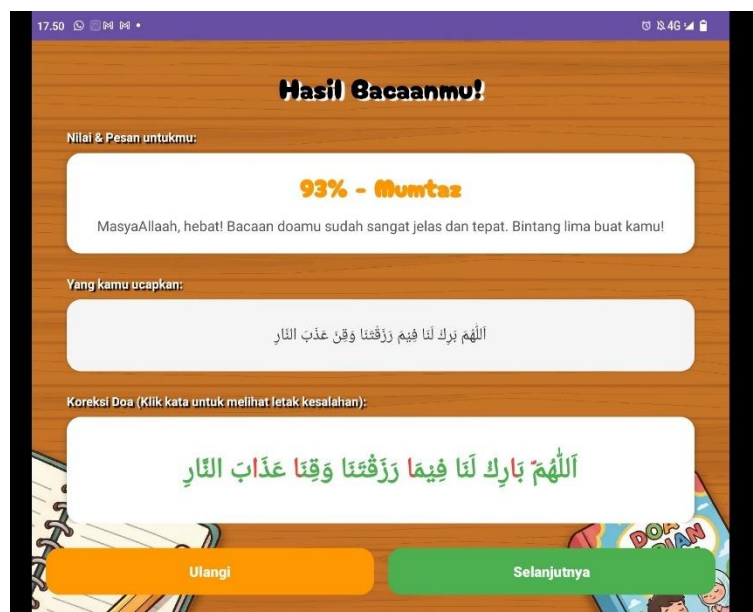
terkait kesalahan yang terjadi, sehingga anak tidak hanya tahu bahwa dirinya salah, tetapi juga memahami letak dan jenis kesalahannya. Implementasi pewarnaan dan penampilan umpan balik *pop up* dilakukan dengan rangkaian kode pada Gambar 4.31.

```
// Pewarnaan karakter menggunakan spannable string
spannable.setSpan(
    ForegroundColorSpan(Color.parseColor("#F44336")),
    idx,
    idx + 1,
    Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
)

// Implementasi clickable span untuk menampilkan umpan balik
val clickableSpan = object : ClickableSpan() {
    override fun onClick(widget: View) {
        tampilkanPopUpFeedback(feedbackTitle, feedbackPesan)
    }
}
```

Gambar 4.31 Implementasi pewarnaan karakter dan tampilan umpan balik

Di samping evaluasi detail, sistem juga menampilkan skor akhir dalam bentuk persentase beserta predikat pencapaiannya juga divisualisasikan. Penyajian ini bertujuan untuk mengemas hasil perhitungan teknis menjadi bentuk apresiasi yang mudah dipahami oleh anak-anak. Visualisasi luaran sistem koreksi ini diilustrasikan pada Gambar 4.32. Dengan integrasi ini, sistem tidak hanya berfungsi sebagai alat penguji, tetapi juga sebagai media pembelajaran yang memberikan umpan balik secara langsung, terarah, dan mudah dipahami oleh anak.



Gambar 4.32 Visualisasi hasil koreksi algoritma dan umpan balik kesalahan tajwid

4.5 Pengumpulan Data Uji Lapangan

Tahap pengumpulan data uji lapangan dilakukan untuk mendapatkan sampel suara dari kondisi nyata target pengguna sistem, yakni anak-anak. Proses ini dilaksanakan di TPA An-Nahrawi Plosokuning dengan melibatkan enam orang anak dalam rentang usia 6-12 tahun. Pada tahap ini, anak diminta menggunakan aplikasi secara langsung untuk membaca doa, sehingga data yang diperoleh mencerminkan kondisi pelafalan yang alami sekaligus memastikan sistem dapat berjalan sebagaimana mestinya. Setiap anak diminta untuk membaca lebih dari satu jenis doa di depan mikrofon, sehingga total diperoleh 12 sampel audio pelafalan doa harian. Doa yang digunakan dalam pengujian ini terdiri dari tiga kategori, yaitu doa sebelum tidur, doa bangun tidur, dan doa sebelum makan. Seluruh rekaman suara kemudian diproses langsung oleh sistem dalam aplikasi yang telah terintegrasi dengan model Whisper yang telah di *fine-tuning*.

Pengambilan data ini dilakukan dalam kondisi semi-terkontrol di lingkungan TPA dengan tingkat kebisingan rendah hingga sedang. Perekaman dilakukan menggunakan perangkat *smartphone* dengan jarak mikrofon sekitar 5-10 cm dari sumber suara untuk menjaga kualitas *input* audio tetap optimal. Seluruh sampel suara disimpan dalam format .wav dengan spesifikasi sample rate 16 kHz (mono) agar kompatibel dengan input arsitektur model Whisper. Rincian data dari 12 sampel audio yang berhasil dikumpulkan dalam pengujian ini, dirangkum dalam Tabel 4.7, dengan identitas anak disamarkan demi menjaga privasi partisipan.

Tabel 4.7 Daftar sampel audio pengujian lapangan

ID Data	Inisial Nama	Usia	Jenis Doa	Durasi
Audio-001	A.	6 tahun	Doa Sebelum Tidur	00:00:07
Audio-002	A.	10 tahun	Doa Sebelum Tidur	00:00:05
Audio-003	U.	10 tahun	Doa Sebelum Tidur	00:00:09
Audio-004	U.	10 tahun	Doa Sebelum Tidur	00:00:04
Audio-005	A.	8 tahun	Doa Sebelum Tidur	00:00:05
Audio-006	A.	6 tahun	Doa Bangun Tidur	00:00:12
Audio-007	A.	6 tahun	Doa Sebelum Makan	00:00:09
Audio-008	A.	10 tahun	Doa Sebelum Makan	00:00:08
Audio-009	U.	10 tahun	Doa Sebelum Makan	00:00:10
Audio-010	U.	10 tahun	Doa Sebelum Makan	00:00:05
Audio-011	A.	8 tahun	Doa Sebelum Makan	00:00:09
Audio-012	Q.	12 tahun	Doa Sebelum Makan	00:00:05

Selain digunakan sebagai data uji, setiap sampel suara juga diproses langsung oleh sistem untuk menghasilkan skor kemiripan berbasis *Levenshtein* serta predikat pelafalan. Hasil ini ditampilkan untuk menunjukkan bagaimana sistem memberikan evaluasi secara otomatis

terhadap bacaan anak. Selain itu, pada tabel disertakan kolom kategori sebagai interpretasi peneliti terhadap predikat yang dihasilkan sistem, guna mempermudah pembacaan hasil dalam konteks evaluasi pelafalan. Namun, perlu ditekankan bahwa penyajian hasil ini bertujuan untuk menggambarkan keluaran sistem, bukan untuk melakukan analisis perilaku pengguna. Hasil evaluasi sistem terhadap data uji lapangan disajikan dalam Tabel 4.8.

Tabel 4.8 Hasil evaluasi sistem terhadap data uji lapangan

ID Data	Jenis Doa	Skor Similarity (%)	Predikat	Kategori
Audio-001	Doa Sebelum Tidur	92%	<i>Mumtaz</i>	Lancar
Audio-002	Doa Sebelum Tidur	94%	<i>Mumtaz</i>	Lancar
Audio-003	Doa Sebelum Tidur	88%	<i>Jayyid</i>	Lancar
Audio-004	Doa Sebelum Tidur	86%	<i>Jayyid</i>	Lancar
Audio-005	Doa Sebelum Tidur	92%	<i>Mumtaz</i>	Lancar
Audio-006	Doa Bangun Tidur	74%	<i>Maqbul</i>	Cukup
Audio-007	Doa Sebelum Makan	91%	<i>Mumtaz</i>	Lancar
Audio-008	Doa Sebelum Makan	97%	<i>Mumtaz</i>	Lancar
Audio-009	Doa Sebelum Makan	91%	<i>Mumtaz</i>	Lancar
Audio-010	Doa Sebelum Makan	93%	<i>Mumtaz</i>	Lancar
Audio-011	Doa Sebelum Makan	93%	<i>Mumtaz</i>	Lancar
Audio-012	Doa Sebelum Makan	84%	<i>Jayyid</i>	Lancar

Berdasarkan hasil pada Tabel 4.8, sistem tidak hanya menghasilkan skor numerik, tetapi juga mengonversikannya ke dalam bentuk predikat seperti *Mumtaz*, *Jayyid*, *Maqbul*, dan Perlu Bimbingan yang ditampilkan langsung pada aplikasi. Untuk mempermudah interpretasi dalam konteks penelitian, predikat tersebut dapat dipahami sebagai representasi tingkat kemampuan pelafalan, yaitu *Mumtaz* menunjukkan kategori “sangat lancar”, *Jayyid* menunjukkan kategori “lancar”, *Maqbul* menunjukkan “cukup”, serta Perlu Bimbingan menunjukkan bahwa pelafalan masih perlu diperbaiki. Temuan ini menunjukkan bahwa sistem yang dikembangkan tidak hanya mampu melakukan transkripsi, tetapi juga dapat memberikan evaluasi pelafalan yang terstruktur dan mudah dipahami, sehingga sesuai dengan tujuan penelitian dalam membangun sistem koreksi bacaan doa anak secara otomatis.

Jumlah sampel dalam pengujian ini selanjutnya akan digunakan sebagai uji awal (*pilot testing*) untuk mengevaluasi performa sistem dalam kondisi penggunaan nyata (*real-world scenario*), khususnya dalam mengenali variasi pelafalan anak. Data ini tidak hanya digunakan untuk menguji kemampuan transkripsi model, tetapi juga untuk mengevaluasi konsistensi sistem dalam memberikan skor pelafalan dan umpan balik berbasis algoritma *Levenshtein* serta validasi aturan tajwid. Keseluruhan data audio uji ini kemudian digunakan sebagai instrumen utama dalam tahap validasi pakar untuk menetapkan *ground truth* atau kunci jawaban. Validasi

dilakukan oleh pihak yang memiliki kompetensi dalam bidang tajwid untuk memastikan bahwa hasil evaluasi sistem sesuai dengan penilaian manusia secara linguistik dan fonetik.

4.6 Penetapan *Ground Truth* melalui Expert Judgement

Tahap ini bertujuan untuk menetapkan *ground truth* atau kunci jawaban dari data audio uji. Dalam konteks evaluasi pelafalan doa berbahasa Arab, kebenaran mutlak tidak dapat ditentukan secara mandiri oleh mesin, melainkan memerlukan validasi dari manusia yang memiliki kapabilitas keilmuan yang relevan. Oleh karena itu, pada tahap ini, penilaian dari seorang ahli tajwid digunakan sebagai standar emas untuk mengukur tingkat reliabilitas algoritma koreksi yang diterapkan dalam aplikasi ‘Teman Ibadah Si Kecil’.

4.6.1 Pelaksanaan *Blind Correction*

Proses validasi ini melibatkan seorang pakar tajwid tersertifikasi, yakni Wiji Astuti, S.Hum.. Beliau dipilih sebagai validator ahli berdasarkan kredibilitas dan jam terbang keilmuannya, di mana beliau telah aktif mengajar *tahsin*, *tahfidz*, dan Al-Qur’an selama lebih dari 20 tahun. Saat ini, beliau aktif mengajar di halaqah *tahfidz* khusus bersama guru dari negeri Irak sebagai ketua *halaqah*, di SD Muhammadiyah Gendol 6 dan Pondok Pesantren Baitul Qowwam sebagai guru *tahfidz*, dan beberapa tempat privat sebagai guru *tahsin*. Metode pembelajaran Qur’an yang diterapkan yakni metode *iqro’* sehingga penilaiannya sangat memenuhi standar kelayakan *expert judgement* dalam penelitian ini.

Pada tahap pelaksanaannya, proses evaluasi pakar dilakukan dengan menggunakan pendekatan *blind correction*. Pada metode ini, pakar mendengarkan 12 sampel audio uji secara mandiri tanpa diberikan akses atau informasi mengenai hasil transkripsi, skor, maupun hasil deteksi oleh sistem koreksi yang dibangun. Pendekatan ini diimplementasikan untuk mencegah terjadinya bias kognitif, sehingga penilaian pakar murni didasarkan pada kualitas akustik *makharijul huruf* anak, bukan karena terpengaruh oleh sugesti dari keluaran yang didapat dari aplikasi.

Selama proses evaluasi, pakar dibekali dengan lembar validasi ahli yang berisi ID Data dan teks referensi doa, seperti doa sebelum tidur, doa bangun tidur, maupun doa sebelum makan. Pakar secara manual kemudian memberikan garis bawah pada titik kesalahan bacaan secara langsung pada teks referensi yang diberikan, mengklasifikasikan, jenis kesalahan apa saja yang terjadi, serta memberikan predikat akhir bacaan berupa *Mumtaz* (90-100%), *Jayyid*

(80-89%), *Maqbul* (65-79%), atau Perlu Bimbingan (<65%) sebagaimana yang telah ditentukan pada metodologi.

4.6.2 Hasil Evaluasi Pakar (*Ground Truth*)

Data berupa coretan letak kesalahan dan predikat yang diberikan oleh pakar pada lembar validasi, pada tahap ini diekstraksi dan direkapitulasi ke dalam bentuk tabel sebagaimana yang dapat dilihat pada Tabel 4.9. Data inilah yang secara resmi ditetapkan sebagai *ground truth*, untuk menjadi parameter pembandingan utama pada tahap evaluasi kinerja algoritma koreksi atau deteksi kesalahan bacaan.

Tabel 4.9 Rekapitulasi hasil evaluasi pakar (*ground truth*)

ID Data	Jenis Doa	Area Evaluasi	Kategori Kesalahan	Predikat Pakar
Audio-001	Doa Sebelum Tidur	اللَّهُمَّ	<i>Makharijul Huruf</i> (tasydid kurang dengung)	<i>Jayyid</i>
		أَحْيَا	<i>Mad</i> (terlalu pendek)	
Audio-002	Doa Sebelum Tidur	بِاسْمِكَ	<i>Mad</i> (terlalu panjang)	<i>Jayyid</i>
		اللَّهُمَّ	<i>Makharijul Huruf</i> (tasydid kurang dengung)	
		أَحْيَا	<i>Mad</i> (terlalu pendek)	
Audio-003	Doa Sebelum Tidur	بِاسْمِكَ	<i>Mad</i> (terlalu panjang)	<i>Mumtaz</i>
		أَمُوتُ	<i>Mad</i> (terlalu panjang)	
Audio-004	Doa Sebelum Tidur	اللَّهُمَّ	<i>Mad</i> (terlalu pendek)	<i>Maqbul</i>
		اللَّهُمَّ	<i>Makharijul Huruf</i> (tasydid kurang dengung)	
		أَحْيَا	<i>Mad</i> (terlalu pendek)	
		بِاسْمِكَ	<i>Mad</i> (terlalu panjang)	
		أَمُوتُ	<i>Mad</i> (terlalu panjang)	
Audio-005	Doa Sebelum Tidur	اللَّهُمَّ	<i>Makharijul Huruf</i> (tasydid kurang dengung)	<i>Jayyid</i>
		بِاسْمِكَ	<i>Mad</i> (terlalu panjang)	
		أَمُوتُ	<i>Mad</i> (terlalu panjang)	

ID Data	Jenis Doa	Area Evaluasi	Kategori Kesalahan	Predikat Pakar
Audio-006	Doa Bangun Tidur	الَّذِي	Mad (terlalu pendek)	Jayyid
		أَحْيَانًا	Mad (terlalu pendek)	
		بَعْدَ	Makharijul Huruf (seperti hamzah)	
		أَمَّا تَنَا	Mad (terlalu pendek)	
		النُّشُورَ	Makharijul Huruf (tasydid kurang dengung)	
Audio-007	Doa Sebelum Makan	رَزَقْنَا	Mad (terlalu panjang)	Maqbul
		رَزَقْنَا	Makharijul Huruf (seperti kaf, kurang memantul)	
		وَقِنَا	Makharijul Huruf (kurang sempurna)	
		عَذَابَ	Makharijul Huruf (kurang maksimal)	
		عَذَابَ	Makharijul Huruf (seperti dal)	
		النَّارَ	Makharijul Huruf (tasydid kurang dengung)	
Audio-008	Doa Sebelum Makan	اللَّهُمَّ	Makharijul Huruf (tasydid kurang dengung)	Jayyid
		رَزَقْنَا	Makharijul Huruf (seperti kaf, tidak memantul)	
		وَقِنَا	Makharijul Huruf (seperti kaf)	
Audio-009	Doa Sebelum Makan	اللَّهُمَّ	Mad (terlalu panjang)	Jayyid
		بَارِكْ	Mad (terlalu pendek)	
		رَزَقْنَا	Makharijul Huruf (seperti kaf, tidak memantul)	
Audio-010	Doa Sebelum Makan	اللَّهُمَّ	Mad (terlalu pendek)	Maqbul
		اللَّهُمَّ	Makharijul Huruf (tasydid kurang dengung)	
		بَارِكْ	Mad (terlalu pendek)	
		فِيمَا	Mad (terlalu pendek)	
		رَزَقْنَا	Makharijul Huruf (seperti kaf, tidak memantul)	
Audio-011	Doa Sebelum Makan	اللَّهُمَّ	Mad (terlalu panjang)	Mumtaz
		رَزَقْنَا	Makharijul Huruf (seperti kaf, tidak memantul)	

ID Data	Jenis Doa	Area Evaluasi	Kategori Kesalahan	Predikat Pakar
Audio-012	Doa Sebelum Makan	اللَّهُمَّ	<i>Makharijul Huruf</i> (tasydid kurang dengung)	<i>Maqbul</i>
		بَارِكْ	<i>Mad</i> (terlalu pendek)	
		لَنَا	<i>Mad</i> (terlalu pendek)	
		فِيْمَا	<i>Mad</i> (terlalu pendek)	
		فِيْمَا	<i>Mad</i> (terlalu pendek)	
		رَزَقْنَا	<i>Makharijul Huruf</i> (seperti kaf, tidak memantul)	
		رَزَقْنَا	<i>Mad</i> (terlalu pendek)	
		وَقْنَا	<i>Makharijul Huruf</i> (kurang maksimal)	

Berdasarkan hasil rekapitulasi pada Tabel 4.9, sebagian besar kesalahan pelafalan yang dilakukan oleh anak berada pada dua kategori utama, yaitu kesalahan *makharijul huruf* dan kesalahan panjang pendek (*mad*). Kesalahan *makharijul huruf* umumnya terjadi pada huruf-huruf dengan artikulasi kompleks seperti ق, ذ, dan ع sedangkan kesalahan *mad* didominasi oleh durasi pelafalan yang terlalu pendek. Penetapan predikat oleh pakar mengacu pada rentang penilaian yang telah ditetapkan pada Bab 3, sehingga konsisten dengan mekanisme klasifikasi yang digunakan dalam sistem berbasis skor kemiripan.

Data pada Tabel 4.9 ini merupakan representasi kuantitatif dari kepakaran manusia. Pada tahap evaluasi kinerja, algoritma koreksi pelafalan yang dirancang akan diukur seberapa jauh kemampuan yang dimiliki dalam mereplikasi akurasi dan ketepatan letak kesalahan seperti yang ditunjukkan oleh pakar pada tabel ini. Perbandingan dilakukan dengan melihat kesesuaian antara hasil deteksi kesalahan oleh sistem dengan hasil identifikasi kesalahan oleh pakar, baik dari segi lokasi kesalahan maupun jenis kesalahan yang terjadi. Dengan adanya *ground truth* berbasis *expert judgement* ini, proses evaluasi sistem tidak hanya bergantung pada metrik komputasi, tetapi juga tervalidasi secara linguistik dan fonetik sesuai kaidah tajwid yang berlaku.

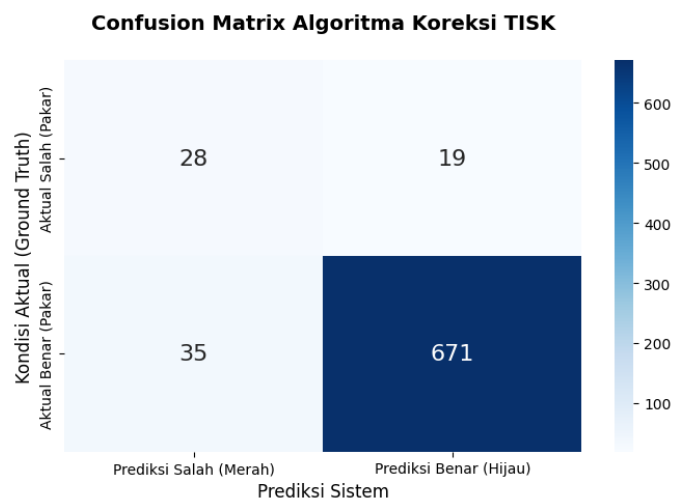
4.7 Evaluasi Kinerja Algoritma Koreksi

Tahap akhir dari penelitian ini adalah melakukan evaluasi kinerja algoritma koreksi pelafalan. Evaluasi dilakukan dengan membandingkan hasil deteksi kesalahan tajwid oleh sistem dengan hasil penilaian manual oleh validator/pakar ahli, untuk mengetahui sejauh mana

algoritma mampu mereplikasi penilaian manusia. Pengujian ini diekstraksi secara spesifik pada level karakter dengan total himpunan data evaluasi sebanyak 753 karakter yang dilafalkan oleh subjek uji coba anak-anak. Dalam proses evaluasi kinerja algoritma ini, predikat pelafalan seperti *Mumtaz*, *Jayyid*, *Maqbul*, dan Perlu Bimbingan tidak digunakan dalam perhitungan *confusion matrix*. Hal ini dikarenakan predikat merupakan bentuk evaluasi akhir yang bersifat agregatif terhadap keseluruhan bacaan. Sementara itu, evaluasi pada tahap ini difokuskan pada kemampuan sistem dalam mendeteksi kesalahan pelafalan secara spesifik pada level karakter dan aturan tajwid. Oleh karena itu, perbandingan dilakukan berdasarkan kesesuaian deteksi kesalahan antara sistem dan pakar dalam mengklasifikasikan setiap karakter sebagai benar atau salah, bukan berdasarkan hasil akhir agregat bacaan.

4.7.1 *Confusion Matrix* Deteksi Kesalahan

Berdasarkan pengujian lapangan, performa algoritma koreksi dalam mengklasifikasikan status pelafalan (benar atau salah) dipetakan ke dalam *confusion matrix*. Visualisasi *heatmap* dari pemetaan tersebut ditunjukkan pada Gambar 4.33.



Gambar 4.33 Visualisasi *heatmap confusion matrix* algoritma koreksi

Berdasarkan pada diatas, sebaran data divisualisasikan menggunakan gradasi warna berskala, di mana intensitas warna yang semakin gelap menunjukkan konsentrasi jumlah karakter yang lebih tinggi pada kuadran klasifikasi tertentu. Untuk menjabarkan rincian angka eksak dari visualisasi matriks tersebut, hasil pemetaan direkapitulasi ke dalam bentuk tabulasi silang seperti yang ditunjukkan pada berikut Tabel 4.10.

Tabel 4.10 Hasil pemetaan *confusion matrix* algoritma koreksi

Kondisi Aktual (Penilaian Pakar)	Prediksi Sistem Salah (Merah)	Prediksi Sistem Benar (Hijau)	Total Aktual
Aktual Salah	TP = 28	FN = 19	47
Aktual Benar	FP = 35	TN = 671	706
Total Prediksi	63	690	Total = 753

Dari Tabel 4.10, dapat diamati bahwa distribusi data sangat tidak seimbang, di mana mayoritas karakter (671 karakter) berhasil dilafalkan dengan benar oleh anak-anak dan divalidasi secara tepat oleh sistem (*True Negative*). Sistem juga berhasil menangkap 28 letak kesalahan dengan tepat (*True Positive*). Namun, terdapat 35 kasus *over-correction* (*False Positive*) di mana sistem menyalahkan pelafalan yang sebenarnya benar, dan 19 kasus *missed-detection* (*False Negative*) di mana sistem gagal menyalahkan pelafalan yang keliru.

4.7.2 Perhitungan Akurasi, Presisi, Recall, dan F1-Score

Berdasarkan nilai pemetaan pada perhitungan *confusion matrix* tersebut, selanjutnya dilakukan perhitungan matematis untuk mengukur reliabilitas algoritma koreksi yang dibangun.

- a. Akurasi (*Accuracy*)

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{28 + 671}{753} = 0,9283 = 92,83\%$$

- b. Presisi (*Precision*)

$$Precision = \frac{TP}{TP + FP} = \frac{28}{28 + 35} = \frac{28}{63} = 0,4444 = 44,44\%$$

- c. Recall (*Sensitivity*)

$$Recall = \frac{TP}{TP + FN} = \frac{28}{28 + 19} = \frac{28}{47} = 0,5957 = 59,57\%$$

- d. F1-Score

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = 2 \times \frac{0,4444 \times 0,5957}{0,4444 + 0,5957} = 50,91\%$$

Berdasarkan perhitungan di atas, sistem memperoleh Akurasi yang sangat tinggi sebesar 92,83%. Hal ini menunjukkan bahwa secara umum sistem cukup handal dalam mengonfirmasi pelafalan anak yang sudah benar. Namun, nilai akurasi yang tinggi ini sangat dipengaruhi oleh

dominasi data benar (*True Negative*), sehingga tidak sepenuhnya mencerminkan kemampuan sistem dalam mendeteksi kesalahan, mengingat distribusi data yang tidak seimbang.

Adapun nilai Presisi (44,44%) dan Recall (59,57%) yang berada pada kisaran menengah ke bawah disebabkan oleh tingginya nilai *False Positive* (35 kasus). Fenomena ini umum terjadi pada pengujian *Automatic Speech Recognition* (ASR) untuk anak-anak, di mana model Whisper cenderung bertindak terlalu sensitif terhadap variasi akustik, seperti suara napas, jeda, ragu-ragu, atau pelafalan cadel yang belum stabil. Sensitivitas ini menyebabkan sistem lebih sering menandainya sebagai bentuk kesalahan, meskipun secara penilaian manusia hal tersebut masih dapat ditoleransi sebagai bacaan yang sah. Selain itu, pendekatan algoritma *two-pointer* yang dirancang untuk menjaga sensitivitas deteksi kesalahan juga berkontribusi terhadap peningkatan *False Positive*, karena sistem lebih memilih untuk menandai potensi kesalahan daripada melewatkan kesalahan yang sebenarnya terjadi.

4.7.3 Perhitungan Koefisien *Cohen's Kappa*

Mengingat evaluasi ini melibatkan penilaian subjektif dari validator manusia dan kondisi *dataset* yang tidak seimbang (didominasi *True Negative*), metrik *Cohen's Kappa* (κ) digunakan untuk mengukur tingkat kesepakatan murni dengan mengeliminasi probabilitas kebetulan (*chance agreement*). Perhitungan metrik ini menggunakan probabilitas proporsi kesepakatan aktual (P_o) dan probabilitas ekspektasi kesepakatan kebetulan (P_e) yang diekstraksi dari probabilitas marginal pada *confusion matrix* (Tabel 4.10). Penjabaran nilai dan perhitungan koefisien *Kappa* dijabarkan sebagai berikut:

a. Perhitungan *Observed Agreement* (P_o)

(P_o) adalah probabilitas proporsi kesepakatan yang diobservasi secara aktual (ekuivalen dengan nilai Akurasi).

$$P_o = \frac{TP + TN}{Total\ Data} = \frac{28 + 671}{753} = 0,9283$$

b. Perhitungan *Expected Agreement* (P_e)

(P_e) adalah probabilitas ekspektasi kesepakatan yang mungkin terjadi secara kebetulan. Nilai ini didapatkan dengan menjumlahkan hasil kali peluang marginal (baris dan kolom) untuk setiap kategori.

$$P_e = \frac{(Aktual_{Salah} \times Prediksi_{Salah}) + (Aktual_{Benar} \times Prediksi_{Benar})}{Total\ Data^2}$$

$$P_e = \frac{(47 \times 63) + (706 \times 690)}{753^2} = \frac{490101}{567009} = 0,8644$$

c. Perhitungan Nilai Kappa

Setelah nilai diobservasi (P_o) dan ekspektasi (P_e) didapatkan, nilai koefisien Kappa dihitung untuk mengetahui tingkat reliabilitas algoritma.

$$\kappa = \frac{P_o - P_e}{1 - P_e} = \frac{0,9283 - 0,8644}{1 - 0,8644} = \frac{0,0639}{0,1356} = \mathbf{0,4713}$$

Berdasarkan hasil komputasi di atas, diperoleh koefisien *Cohen's Kappa* sebesar 0,4713. Merujuk pada standar interpretasi metrik kategorisasi Landis dan Koch (1977) yang juga didukung dalam tinjauan metodologi reliabilitas modern (Gisev *et al.*, 2013), nilai tersebut berada pada rentang 0,41-0,60, yang mengindikasikan tingkat Kesepakatan Sedang (*Moderate Agreement*). Pencapaian tingkat kesepakatan sedang ini menunjukkan bahwa algoritma koreksi pelafalan yang dikembangkan memiliki tingkat kesesuaian yang cukup dengan penilaian pakar dan berpotensi digunakan sebagai alat bantu evaluasi pelafalan secara otomatis, dengan catatan ruang pengembangan lanjutan pada optimasi ambang batas (*threshold*) akustik untuk menekan tingkat *over-correction* pada artikulasi anak-anak.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian performa algoritma deteksi pelafalan berbasis *Automatic Speech Recognition* (ASR) yang telah dilakukan, dapat disimpulkan bahwa metode *fine-tuning* pada model Whisper terbukti efektif dalam meningkatkan kemampuan pengenalan suara anak-anak. Hal ini ditunjukkan oleh penurunan nilai *Character Error Rate* (CER) yang signifikan, dari 50,33% pada model *baseline* menjadi 8,6% setelah proses *fine-tuning*. Kinerja model ASR yang telah dioptimalkan tersebut kemudian diintegrasikan dengan algoritma *two-pointer* dan perhitungan kemiripan berbasis *levenshtein distance* untuk membentuk sistem koreksi pelafalan yang dapat berjalan secara luring (*offline*). Hasil pengujian menunjukkan bahwa sistem mampu mencapai tingkat akurasi yang tinggi, yaitu sebesar 92,83% dalam mengidentifikasi pelafalan benar dan salah pada level karakter.

Namun demikian, meski sistem mencatatkan akurasi yang tinggi, ketika dibandingkan dengan penilaian pakar melalui metrik *Cohen's Kappa*, sistem menunjukkan tingkat Kesepakatan Sedang (*Moderate Agreement*) dengan nilai 0,4713. Perbedaan ini terutama disebabkan oleh kecenderungan sistem yang lebih sensitif dalam mendeteksi kesalahan (*over-correction*), khususnya terhadap variasi suara anak seperti jeda, napas, maupun pelafalan yang belum sempurna, yang dalam praktiknya masih dapat ditoleransi oleh manusia (pakar). Kendati demikian, sistem yang dikembangkan telah mampu menunjukkan kinerja yang konsisten dan rasional dalam mendeteksi kesalahan pelafalan. Selain itu, penelitian ini juga berhasil mengimplementasikan konsep *Edge AI* dengan menjalankan model *deep learning* secara langsung pada perangkat *mobile* melalui *Java Native Interface* (JNI), yang tereksekusi secara aman tanpa memicu kebocoran memori (*memory leak*). Sehingga sistem dapat digunakan tanpa ketergantungan pada koneksi internet. Melalui capaian tersebut, penelitian ini menghasilkan sebuah pendekatan yang layak sebagai dasar pengembangan sistem evaluasi pelafalan berbasis suara, khususnya untuk pembelajaran doa anak. Ke depan, peningkatan pada aspek sensitivitas deteksi dan penyesuaian ambang batas (*threshold*) masih diperlukan agar hasil evaluasi sistem dapat semakin mendekati penilaian manusia.

5.2 Saran

Berdasarkan analisis keterbatasan yang ditemukan selama proses penelitian dan pengujian lapangan, terdapat beberapa saran pengembangan yang direkomendasikan bagi peneliti selanjutnya, antara lain:

1. Penyesuaian ambang batas (*threshold*) toleransi, tingginya angka *False Positive* menunjukkan bahwa sistem ASR terlalu sensitif terhadap ketidaksempurnaan wajar pada artikulasi anak-anak. Peneliti selanjutnya disarankan untuk menerapkan fungsi *threshold* fleksibel pada perhitungan *Levenshtein Distance*, atau membangun kamus toleransi fonetik (misalnya memaklumi pertukaran konsonan spesifik akibat pelafalan cadel) agar sistem tidak terlalu kaku (*over-corrective*).
2. Perluasan dan variasi *dataset* pelatihan, untuk meningkatkan generalisasi model Whisper, disarankan untuk memperbanyak variasi *dataset* pelatihan (*fine-tuning*) yang mencakup lebih banyak rentang usia anak, berbagai dialek daerah, serta penambahan rekaman dengan berbagai tingkat derau latar belakang (*background noise*) agar model lebih tangguh (*robust*) saat digunakan di lingkungan nyata.
3. Penyempurnaan penanganan indeks paralel, pada pengembangan selanjutnya, pemetaan indeks warna untuk visualisasi *Spannable String* dapat dioptimalkan dengan menggunakan algoritma *Dynamic Time Warping* (DTW) agar sinkronisasi pemotongan kata dari output mesin C++ ke lingkungan UI Android (Kotlin) dapat berjalan lebih presisi, terutama pada kasus doa yang berdurasi sangat panjang.

DAFTAR PUSTAKA

Abouhalim, M., Afarfar, A. dan Belahsen, M.F. (2023) “Episodic Memory and Memory Training : An Example in Moroccan Subjects Who Recite the Koran,” *Psychology*, 14(10), hal. 1617–1633. doi:10.4236/psych.2023.1410093.

Akbar, A., Husodo, A.Y. dan Zubaidi, A. (2019) “Implementasi Google Speech API pada Aplikasi Koreksi Hafalan Al-Qur’an Berbasis Android | The Implementation of the Google Speech on Qur’an Recitation Correction Application Based on Android,” *Jurnal Teknologi Informasi, Komputer, dan Aplikasinya (JTika)*, 1(1), hal. 1–8. doi:https://doi.org/10.29303/jtika.v1i1.8.

Al-Fadhli, S., Al-Harbi, H. dan Cherif, A. (2023) “Speech Recognition Models for Holy Quran Recitation Based on Modern Approaches and Tajweed Rules : A Comprehensive Overview,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, 14(12). doi:10.14569/IJACSA.2023.0141297.

Al-Kharusi, M.H. *et al.* (2025) “A Critical Review of The Need for Knowledge–Centric Evaluation of Quranic Recitation,” hal. 1–32. doi:10.48550/arXiv.2510.12858.

Algabri, M. *et al.* (2022) “Mispronunciation Detection and Diagnosis with Articulatory-Level Feedback Generation for Non-Native Arabic Speech,” *Mathematics* [Preprint]. doi:10.3390/math10152727.

Ar Rosyid, H., Abdullah, D. dan Alqahtani, M.S. (2024) “Constructing Qur’an Recitation Classification using Alexnet Algorithm,” *Knowledge Engineering and Data Science (KEDS)*, 7(2), hal. 152–163. doi:10.17977/um018v7i22024p152-163.

Ashvin, A. *et al.* (2025) “Evaluation of Speech Foundation Models for ASR on Child-Adult Conversations in Autism Diagnostic Sessions,” hal. 3–7. doi:https://doi.org/10.48550/arXiv.2409.16135.

Assisi, M. *et al.* (2022) “Rancang Bangun Aplikasi Hafalan Al-Quran dengan Google Speech API Berbasis Android,” *Jurnal Rekayasa Teknologi Informasi (JURTI)*, 6(1), hal. 26–35. doi:10.30872/jurti.v6i1.8006.

Baevski, A. *et al.* (2020) “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,” hal. 1–19. doi:https://doi.org/10.48550/arXiv.2006.11477.

Bhardwaj, V. *et al.* (2022) “Automatic Speech Recognition (ASR) Systems for Children: A Systematic Literature Review,” *Applied Sciences*, 12(9). doi:10.3390/app12094419.

Bouchelligua, W., Al-dayil, R. dan Algaith, A. (2025) “Effective Data Augmentation

Techniques for Arabic Speech Emotion Recognition Using Convolutional Neural Networks,” *Applied Sciences*, 15(4), hal. 1–36. doi:10.3390/app15042114.

Chaugule, I. dan Sankaye, S.R. (2025) “The Critical Role and Systematic Evaluation of Data Preprocessing in Deep Learning for Speech Emotion Recognition,” *International Journal of Scientific Research in Science and Technology (IJSRST)*, 12(3), hal. 1191–1203. doi:10.32628/IJSRST25123135.

Cohen, Jacob (1960) “A Coefficient of Agreement for Nominal Scales,” *Educational and Psychological Measurement*, 20(1), hal. 37–46. doi:10.1177/001316446002000104.

Daoud, M.A. *et al.* (2025) “Enhancing Qur’anic recitation through machine learning : a predictive approach to Tajweed optimization,” *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, 39(3), hal. 1562–1570. doi:10.11591/ijeecs.v39.i3.pp1562-1570.

Draelos, R. (2019) *Best Use of Train/Val/Test Splits, with Tips for Medical Data*. Tersedia pada: <https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/> (Diakses: 16 Maret 2026).

Fitrianingrum, S.S. dan Aminingsih, E.F. (2024) “Analisis Kesalahan Pengucapan dalam Membaca Huruf Hijaiyah: Kajian Fonologi,” *DIAJAR: Jurnal Pendidikan Dan Pembelajaran*, 3(1), hal. 1–7. doi:10.54259/diajar.v3i1.2224.

Gisev, N. *et al.* (2013) “Interrater agreement and interrater reliability : Key concepts , approaches , and applications,” *Research in Social and Administrative Pharmacy*, 9(3), hal. 330–338. doi:10.1016/j.sapharm.2012.04.004.

Graave, T. *et al.* (2024) “Mixed Children/Adult/Childrenized Fine-Tuning for Children’s ASR: How to Reduce Age Mismatch and Speaking Style Mismatch,” *Interspeech 2024*, hal. 5188–5192. doi:10.21437/Interspeech.2024-499.

Graves, A., Mohamed, A. dan Hinton, G. (2012) “Speech Recognition with Deep Recurrent Neural Networks,” *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, (3), hal. 6645–6649. doi:10.1109/ICASSP.2013.6638947.

Hadiyansah, R. dan Andamira, R. (2023) “Convolutional Neural Network (CNN) for Detecting Al-Qur’an Reciting and Memorizing,” *Khazanah Journal of Religion and Technology (KJRT)*, 1(2), hal. 44–48. doi:<https://doi.org/10.15575/kjrt.v1i2.235>.

Al Harere, A. dan Al Jallad, K. (2023) “Quran Recitation Recognition using End-to-End Deep Learning,” hal. 1–22. doi:<https://doi.org/10.48550/arXiv.2305.07034>.

Hasim, R. *et al.* (2023) “Scoping Review on the Benefits of Reciting, Listening and

Memorising the Quran,” *Environment-Behaviour Proceedings Journal*, 8(25), hal. 37–43. doi:10.21834/e-bpj.v8i25.4826.

Horii, K. *et al.* (2025) “Why is children’s ASR so difficult? Analyzing children’s phonological error patterns using SSL-based phoneme recognizers,” *Interspeech 2025*, hal. 2870–2874. doi:10.21437/Interspeech.2025-2645.

Ibadurrahman, M. *et al.* (2025) “Mobile Learning Applications for Islamic Studies : A Systematic Review of Design Principles and Learning Outcomes,” *Journal of International Multidisciplinary Research*, 3(8), hal. 12–19. doi:10.62504/fx1bd279.

Isa, S.M. (2019) *Speech Recognition*. Tersedia pada: <https://mti.binus.ac.id/2019/05/08/speech-recognition/>.

Jurafsky, D. dan Martin, J.H. (2026) *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd ed.

Kathania, H.K. *et al.* (2021) “Using Data Augmentation and Time-Scale Modification to Improve ASR of Children’s Speech in Noisy Environments,” *Applied Sciences*, 11(18), hal. 1–18. doi:10.3390/app11188420.

Khairani, D. *et al.* (2024) “Enhancing Speech-to-Text and Translation Capabilities for Developing Arabic Learning Games : Integration of Whisper OpenAI Model and Google API Translate,” *Jurnal Teknik Informatika (JTI)*, 17(2), hal. 203–212. doi:<https://doi.org/10.15408/jti.v17i2.41240>.

Koenecke, A. *et al.* (2024) “Careless Whisper: Speech-to-Text Hallucination Harms,” *FAccT ’24: Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, hal. 1672–1681. doi:10.1145/3630106.3658996.

Kurniawati, E., Ni’mah, M. dan Sofa, A.R. (2025) “Tahlîlu al Akhta’ fî Mahârah al Kalâm bi al-Lughah al-Arabîyah li Tullâbi Ma’hadî Sirâj at-Thâlibîn al-Islâmiy | Analysis of Errors in Speaking Skill in Arabic for Students of SiraaJ Al-Talibeen Islamic Institute,” *Al Manar: English And Arabic Journal*, 16(1), hal. 78–93. doi:<http://dx.doi.org/10.24014/al-manar.v16i1.29076>.

Malik, Mishaim *et al.* (2021) “Automatic speech recognition: a survey,” *Multimedia Tools and Applications*, 80, hal. 1–47. doi:10.1007/s11042-020-10073-7.

Marzuki dan Ummah, S.C. (2020) *Dasar-Dasar Ilmu Tajwid*. cet. 1. Diedit oleh Y. Arifin. Yogyakarta: DIVA Press.

Masykuri, A. *et al.* (2024) “Korelasi antara Pembelajaran Nilai Moral dan Agama dengan

Perkembangan Kognitif pada Anak Usia Dini: Sebuah Review Sistematis,” *Nusantara Educational Review*, 2(1), hal. 52–58. doi:10.55732/ner.v2i1.1260.

Mensah, M.A. *et al.* (2025) “Benchmarking Akan ASR Models Across Domain- Specific Datasets: A Comparative Evaluation of Performance, Scalability, and Adaptability,” hal. 1–17. doi:10.48550/arXiv.2507.02407.

Mohamed, O. dan Aly, S.A. (2021) “ASER: Arabic Speech Emotion Recognition Employing Wav2vec2.0 and HuBERT Based on BAVED Dataset,” *Transactions on Engineering and Computing Sciences*, 9(6), hal. 1–8. doi:https://doi.org/10.14738/tmlai.96.11039.

Nailurrahmi, F. dan Marlina, L. (2025) “Analisis Interferensi Fonetik Bahasa Ibu Terhadap Pelafalan Fonem Bahasa Arab dalam Pembelajaran Kosakata,” *Moderasi : Journal of Islamic Studies*, 05(01), hal. 279–290. doi:10.54471/moderasi.v5i1.113.

Open AI (2022) *Introducing Whisper*. Tersedia pada: <https://openai.com/id-ID/index/whisper/>.

Purwanto, M.N. (2010) *Prinsip-Prinsip dan Teknik Evaluasi Pengajaran*. Cet. 16. Diedit oleh T. Surjaman. Bandung: PT REMAJA ROSDAKARYA.

Puteri, S.M. (2025) “Kontribusi Pendidikan Agama Islam dalam Edukasi Moral dan Etika Anak Usia Sekolah Dasar,” *Journal of Character and Elementary Education*, 4(3), hal. 24–35. doi:10.35508/jocee.v4i3.23051.

Radford, A. *et al.* (2023) “Robust Speech Recognition via Large-Scale Weak Supervision,” *Proceedings of the 40th International Conference on Machine Learning, in Proceedings of Machine Learning Research*, 202, hal. 28492–28518.

Raissi, T., Rossenbach, N. dan Schlüter, R. (2025) “Analysis of Domain Shift across ASR Architectures via TTS-Enabled Separation of Target Domain and Acoustic Conditions.” doi:10.48550/arXiv.2508.09868.

Ridiawati *et al.* (2025) “Implementasi Metode Takrir dalam Meningkatkan Hafalan Al-Qur’an,” *Kharismatik : Jurnal Ilmu Pendidikan*, 3(1), hal. 1–14. doi:10.70757/kharismatik.v3i1.22.

Sadiq, M.U. dan Yousaf, M.M. (2020) “Distributed Algorithm for Parallel Edit Distance Computation,” *Computing and Informatics*, 39(4), hal. 757–779. doi:10.31577/cai_2020_4_757.

Salehi, P. *et al.* (2025) “Comparative Analysis of Audio Feature Extraction for Real-Time Talking Portrait Synthesis,” *Big Data and Cognitive Computing*, 9(3), hal. 1–21.

doi:10.3390/bdcc9030059.

Sallam, Malik, Barakat, M. dan Sallam, Mohammed (2023) “Pilot Testing of a Tool to Standardize the Assessment of the Quality of Health Information Generated by Artificial Intelligence-Based Models,” *Cureus*, 15(11), hal. 1–9. doi:10.7759/cureus.49373.

Sammut, C. dan Webb, G.I. (2017) “Encyclopedia of Machine Learning and Data Mining .” Springer Science + Business Media . doi:10.1007978-1-4899-7687-1 .

Saraf, A. *et al.* (2025) “Multilingual Translation for Speech and Text using Whisper AI: A Deep Learning Approach,” *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 13(VII), hal. 1895–1901. doi:10.22214/ijraset.2025.73288.

Setiawan, A., Hartati, Z. dan Luthfi, S. (2025) “Implementasi Metode Talaqqi Pada Pembelajaran Al- Qur’an,” *Jurnal PAI Raden Fatah*, 7(3), hal. 303–310. doi:10.19109/pairf.v7i3.31139.

Singh, V.P., Sahidullah, M. dan Kinnunen, T.H. (2025) “Causal Analysis of ASR Errors for Children: Quantifying the Impact of Physiological, Cognitive, and Extrinsic Factors,” hal. 1–19. doi:https://doi.org/10.48550/arXiv.2502.08587.

Sujjada, A., Insany, G.P. dan Nugraha, M.F. (2024) “Implementasi Automatic Speech Recognition Pada Penilaian Hafalan Al-Quran Dengan Metode Muroja’ah Berbasis Android,” *CICES (Cyberpreneurship Innovative and Creative Exact and Social Science)*, 10(2), hal. 216–228. doi:10.33050/cices.v10i2.3247.

Syari, A.P. dan Mahendra, A. (2022) “Taḥlīlu al Akḥṭā’i fī Nuṭqi al Lughati al ‘Arabiyyati ‘inda al Talāmīzi fī Ma’hadī al Nūri Tangkit Sungai Gelam Muaro Jāmbī (Dirāsātun Taḥlīliyyatun fī ‘Ilmi al Aṣwāti),” *Al-Uslub: Journal of Arabic Linguistic and Literature*, 06(02), hal. 291–316. doi:https://doi.org/10.30631/al-uslub.v6i02.133.

Sevara, S. (2024) “The role of corrective feedback in second language acquisition,” *European Journal of Innovation in Nonformal Education (EJINE)*, 4(06), hal. 205–208. doi:https://doi.org/10.5281/zenodo.12599307.

Talafha, B., Waheed, A. dan Abdul-Mageed, M. (2023) “N-Shot Benchmarking of Whisper on Diverse Arabic Speech Recognition.” doi:10.48550/arXiv.2306.02902.

Timmel, V. *et al.* (2025) “Fine-tuning Whisper on Low-Resource Languages for Real-World Applications.” doi:10.48550/arXiv.2412.15726.

Wahyudin, D., Karim, A. dan Solehah, M. (2024) “Phonological Errors in Reciting the Quran among Madrasah Diniyyah Students: Identifying Sources and Solutions|Al-Akhthā Al-

Shauthiyyah fī Qirāah Al-Qurān Al-Karīm lada Thullāb Al-Madāris Al-Dīniyyah: Tahdīd Al-Mashādir Wa Al-Hulūl,” *Ta’lim al-‘Arabiyyah: Jurnal Pendidikan Bahasa Arab dan Kebahasaaraban*, 8(1), hal. 132–147. doi:10.15575/jpba.v8i1.34806.

Warrens, M.J. (2015) “Journal of Psychology & Psychotherapy Five Ways to Look at Cohen ’ s Kappa,” *Journal of Psychology & Psychotherapy*, 5(4), hal. 8–11. doi:10.4172/2161-0487.1000197.

Yeung, G., Fan, R. dan Alwan, A. (2021) “Fundamental Frequency Feature Normalization and Data Augmentation for Child Speech Recognition.” doi:10.48550/arXiv.2102.09106.

LAMPIRAN

Lampiran A - Surat Pernyataan Ahli (*Expert Judgement*)

SURAT PERNYATAAN AHLI (*EXPERT JUDGEMENT*)

Yang bertanda tangan di bawah ini:

Nama Validator : Wiji Astuti S.S
Pekerjaan/Instansi : IRT dan Guru Tahfidh dan bahasa Arab
Bidang Keahlian : Tahfidh, tahsin

Telah melakukan pencermatan, penelaahan, dan analisis instrumen penelitian dengan judul
"INTEGRASI *FINE-TUNING* MODEL WHISPER DAN ALGORITMA
LEVENSHTEIN DISTANCE SEBAGAI PENDEKATAN EVALUASI PELAFALAN
DOA HARIAN OLEH ANAK" yang dibuat oleh:

Nama Mahasiswa : Umaymatun Az-Zauraa'
NIM : 22523283
Program Studi : Informatika
Perguruan Tinggi : Universitas Islam Indonesia

Dengan ini menyatakan bahwa instrumen penelitian yang terdiri dari:

1. Dataset Audio Rekaman Pelafalan Doa Anak (12 Sampel Audio)
2. Lembar Kerja Evaluasi Tajwid dan Makhraj (*Ground Truth*)
3. Mekanisme Koreksi Buta (*Blind Correction*) untuk Penentuan Akurasi Sistem

dinyatakan **VALID** / ~~TIDAK VALID~~ dengan keterangan terlampir.

Selanjutnya, instrumen ini:

- ☒ Layak digunakan untuk penelitian
☐ Layak digunakan untuk penelitian dengan perbaikan
☐ Tidak layak digunakan untuk penelitian

Demikian pernyataan ini dibuat untuk dapat digunakan sebagaimana mestinya.

Yogyakarta, 23 April 2026

Validator,



(.....)

~~HP~~/NIK: 340407471081004

Lampiran B - Lembar Validasi Ahli (*Ground Truth*) Pengujian Pelafalan Doa Anak

LEMBAR VALIDASI AHLI (*GROUND TRUTH*)

PENGUJIAN PELAFALAN DOA ANAK

Nama Ahli : Wulji Astuti S.S

Tanggal : 23 April 2026

Instruksi: Mohon dengarkan rekaman audio, lalu lingkari atau coret huruf/kata pada teks di bawah ini jika terdapat kesalahan pelafalan menurut hukum tajwid. Gunakan kolom catatan jika terdapat detail kesalahan spesifik.

ID Audio	Area Evaluasi & Transkrip Aktual (<i>Ground Truth</i>)	Kategori Kesalahan	Predikat
Audio-001 (Doa Sebelum Tidur)	بِسْمِكَ اللَّهُمَّ أَحْيَا وَبِسْمِكَ أَمُوتُ Catatan Ahli: 1. bacaan tidak nyambung 2. Saung kurang sempurna 3. Panjang mad kurang sempurna	☐ Makhraj ☐ Mad Thabi'i ☐ Kelengkapan Kata	☐ Mumtaz ☒ Jayyid ☒ Maqbul ☐ Bimbingan

90-100 10-2
80-89 13-54
65-79 16-8
565 % / 9-82

ID Audio	Area Evaluasi & Transkrip Aktual (Ground Truth)	Kategori Kesalahan	Predikat
Audio-002 (Doa Sebelum Tidur)	بِسْمِكَ اللَّهُمَّ أَحْيَا وَبِسْمِكَ أَمُوتُ 3 2 1 Catatan Ahli: 1. harusnya pendek dibaca panjang 2. glunusah kurang 3. sempurna 3. mad thabi'i kurang. sempurna	☐ Makhraj ☐ Mad Thabi'i ☐ Kelengkapan Kata	☐ Mumtaz ☒ Jayyid ☒ Maqbul ☐ Bimbingan
Audio-003 (Doa Sebelum Tidur)	بِسْمِكَ اللَّهُمَّ أَحْيَا وَبِسْمِكَ أَمُوتُ 2 1 Catatan Ahli: 2 kesalahan harusnya pendek dibaca panjang	☐ Makhraj ☐ Mad Thabi'i ☐ Kelengkapan Kata	☒ Mumtaz ☒ Jayyid ☐ Maqbul ☐ Bimbingan

ID Audio	Area Evaluasi & Transkrip Aktual (Ground Truth)	Kategori Kesalahan	Predikat
Audio-004 (Doa Sebelum Tidur)	بِاسْمِكَ اللَّهُمَّ أَحْيَا وَبِاسْمِكَ أَمُوتُ _{1 5 4 3 2 1} Catatan Ahli: 1. harusnya nyambung dirotong 2. kurang panjang 3. ghunnah kurang 4. tidak panjang 5 & 6 harusnya pendek dibaca panjang	☐ Makhraj ☐ Mad Thabi'i ☐ Kelengkapan Kata	☐ Mumtaz ☐ Jayyid ☒ Maqbul ☐ Bimbingan
Audio-005 (Doa Sebelum Tidur)	بِاسْمِكَ اللَّهُمَّ أَحْيَا وَبِاسْمِكَ أَمُوتُ _{4 3 2 1} Catatan Ahli: 1. tidak nyambung 2. kurang sempurna 3 & 4 kepanjangan harusnya pendek	☐ Makhraj ☐ Mad Thabi'i ☐ Kelengkapan Kata	☐ Mumtaz ☒ Jayyid ☐ Maqbul ☐ Bimbingan

ID Audio	Area Evaluasi & Transkrip Aktual (Ground Truth)	Kategori Kesalahan	Predikat
Audio-006 02 (Doa Sebelum Tidur)	بِسْمِكَ اللَّهُمَّ أَخِيَا وَبِسْمِكَ أَمُوتُ اللَّهُمَّ بَارِكْ لَنَا فِيهِمَا رَزَقْنَا وَقِنَا عَذَابَ النَّارِ Catatan Ahli: 10 8 7 6 5 4 3 2 1 1. kurang sungguh, 2 kurang panjang, 6 kurang tebal 7. tidak wantul ق → د , 9 baca ق kurang maksimal 10. kurang tebal.	☐ Makhraj ☐ Mad Thabi'i ☐ Kelengkapan Kata	☐ Mumtaz ☐ Jayyid ☒ Maqbul ☐ Bimbingan
Audio-006 (Doa Bangun Tidur)	الْحَمْدُ لِلَّهِ الَّذِي أَخْيَانَا بَعْدَ مَا آمَنَّا وَإِلَيْهِ النُّشُورُ Catatan Ahli: 2, 4 4 3 2 1 1. kurang panjang 3. huruf ع kurang seperti hamzah 4 5 ghunnah kurang 6. huruf ج kurang tebal	☐ Makhraj ☐ Mad Thabi'i ☐ Kelengkapan Kata	☐ Mumtaz ☒ Jayyid ☐ Maqbul ☐ Bimbingan

ID Audio	Area Evaluasi & Transkrip Aktual (Ground Truth)	Kategori Kesalahan	Predikat
Audio-010 (Doa Sebelum Makan)	اللَّهُمَّ بَارِكْ لَنَا فِيْمَا رَزَقْتَنَا وَقِنَا عَذَابَ النَّارِ 4 3 2 1 Catatan Ahli: <u>1. kurang panjang, 2. kurang panjang 3. kurang tebal</u> <u>4. tidak mantul, 4 → 5. huruf r kurang tebal</u>	☐ Makhraj ☐ Mad Thabi'i ☐ Kelengkapan Kata	☐ Mumtaz ☒ Jayyid ☐ Maqbul ☐ Bimbingan
Audio-011 (Doa Sebelum Makan)	اللَّهُمَّ بَارِكْ لَنَا فِيْمَا رَزَقْتَنَا وَقِنَا عَذَابَ النَّارِ 4 3 2 1 Catatan Ahli: <u>1. kurang panjang, 2. kurang panjang 3. kurang tebal</u> <u>6. tidak mantul 6 → 5</u>	☐ Makhraj ☐ Mad Thabi'i ☐ Kelengkapan Kata	☐ Mumtaz ☐ Jayyid ☒ Maqbul ☐ Bimbingan

ID Audio	Area Evaluasi & Transkrip Aktual (Ground Truth)	Kategori Kesalahan	Predikat
Audio-012 011 (Doa Sebelum Makan)	اللَّهُمَّ بَارِكْ لَنَا فِيْمَا رَزَقْتَنَا وَقِنَا عَذَابَ النَّارِ Catatan Ahli: 1. kepanjangan, 2. tidak masuk, 3. kurang tebal	☐ Makhraj ☐ Mad Thabi'i ☐ Kelengkapan Kata	☒ Mumtaz ☐ Jayyid ☐ Maqbul ☐ Bimbingan

*Lembar ini digunakan sebagai instrumen penelitian untuk menentukan Ground Truth (Standar Emas) bagi pengujian model ASR Whisper.

Lampiran C - Metadata Audio Uji

ID Data	Nama File Audio	Jenis Doa	Durasi	Sumber Data	Tanggal Pengambilan	ID Penutur	Inisial Nama	Usia	Catatan
Audio-001	20260422_165913_Doa_Sebelum_Tidur.wav	Doa Sebelum Tidur	00:00:07	Rekam Manual	22/04/2026	AU-001	A.	6 tahun	TPA An-Nahrowi Plosokuning
Audio-002	20260422_172018_Doa_Sebelum_Tidur.wav	Doa Sebelum Tidur	00:00:05	Rekam Manual	22/04/2026	AU-002	A.	10 tahun	TPA An-Nahrowi Plosokuning
Audio-003	20260422_173108_Doa_Sebelum_Tidur.wav	Doa Sebelum Tidur	00:00:09	Rekam Manual	22/04/2026	AU-003	U.	10 tahun	TPA An-Nahrowi Plosokuning
Audio-004	20260422_174738_Doa_Sebelum_Tidur.wav	Doa Sebelum Tidur	00:00:04	Rekam Manual	22/04/2026	AU-004	U.	10 tahun	TPA An-Nahrowi Plosokuning
Audio-005	20260422_181426_Doa_Sebelum_Tidur.wav	Doa Sebelum Tidur	00:00:05	Rekam Manual	22/04/2026	AU-005	A.	8 tahun	TPA An-Nahrowi Plosokuning
Audio-006	20260422_171321_Doa_Bangun_Tidur.wav	Doa Bangun Tidur	00:00:12	Rekam Manual	22/04/2026	AU-001	A.	6 tahun	TPA An-Nahrowi Plosokuning
Audio-007	20260422_170321_Doa_Sebelum_Makan.wav	Doa Sebelum Makan	00:00:09	Rekam Manual	22/04/2026	AU-001	A.	6 tahun	TPA An-Nahrowi Plosokuning
Audio-008	20260422_171819_Doa_Sebelum_Makan.wav	Doa Sebelum Makan	00:00:08	Rekam Manual	22/04/2026	AU-002	A.	10 tahun	TPA An-Nahrowi Plosokuning

ID Data	Nama File Audio	Jenis Doa	Durasi	Sumber Data	Tanggal Pengambilan	ID Penutur	Inisial Nama	Usia	Catatan
Audio-009	20260422_173511_Doa_Sebelum_Makan.wav	Doa Sebelum Makan	00:00:10	Rekam Manual	22/04/2026	AU-003	U.	10 tahun	TPA An-Nahrowi Plosokuning
Audio-010	20260422_174919_Doa_Sebelum_Makan.wav	Doa Sebelum Makan	00:00:05	Rekam Manual	22/04/2026	AU-004	U.	10 tahun	TPA An-Nahrowi Plosokuning
Audio-011	20260422_181635_Doa_Sebelum_Makan.wav	Doa Sebelum Makan	00:00:09	Rekam Manual	22/04/2026	AU-005	A.	8 tahun	TPA An-Nahrowi Plosokuning
Audio-012	20260422_182546_Doa_Sebelum_Makan.wav	Doa Sebelum Makan	00:00:05	Rekam Manual	22/04/2026	AU-006	Q.	12 tahun	TPA An-Nahrowi Plosokuning

Lampiran D – Teks Doa Acuan

No	Jenis Doa	Teks Doa
1	Doa Sebelum Tidur	بِسْمِكَ اللَّهُمَّ أَحْيَا وَبِسْمِكَ أَمُوتُ
2	Doa Bangun Tidur	الْحَمْدُ لِلَّهِ الَّذِي أَحْيَانَا بَعْدَ مَا أَمَاتَنَا وَإِلَيْهِ النُّشُورُ
3	Doa Sebelum Makan	اللَّهُمَّ بَارِكْ لَنَا فِيْمَا رَزَقْتَنَا وَقِنَا عَذَابَ النَّارِ

Lampiran E – Dokumentasi Pengumpulan Data

