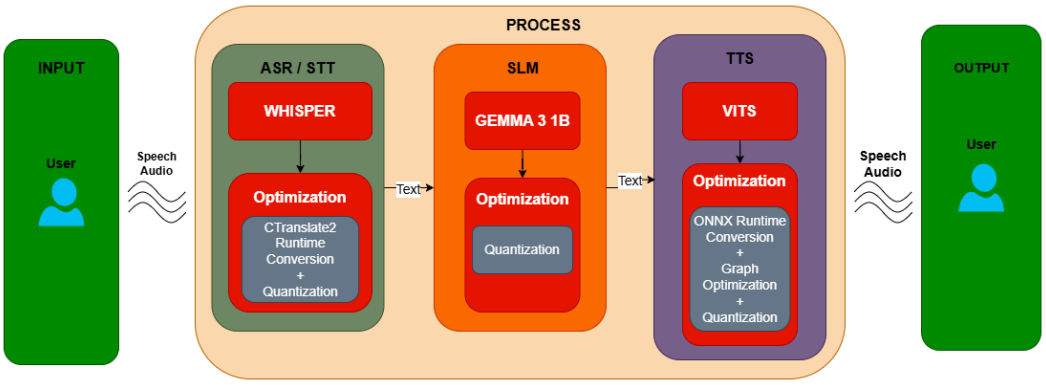
BAB III

METODOLOGI PENELITIAN

3.1 Gambaran Umum Objek Penelitian

Sistem *voice conversational* AI merupakan salah satu bentuk teknologi kecerdasan buatan yang dirancang untuk memungkinkan interaksi alami antara manusia dan komputer melalui percakapan berbasis suara. Sistem ini terdiri dari rangkaian komponen yang bekerja secara berurutan, yaitu ASR untuk mengubah suara menjadi teks, SLM untuk memahami konteks dan menghasilkan respons, serta TTS untuk mengubah kembali teks menjadi *audio*.



Gambar 3. 1 Kerangka Sistem *Voice conversational AI*

Gambar 3.1 menunjukkan kerangka sistem yang dibangun dalam penelitian ini. Objek yang dikaji adalah pengembangan dan optimasi sistem *voice conversational* AI berbasis arsitektur *cascaded* yang memanfaatkan tiga model utama, yaitu Whisper untuk komponen ASR, Gemma 3 1B untuk SLM, serta VITS untuk TTS. Ketiga model tersebut dipilih karena keunggulannya dalam kualitas transkripsi, kemampuan pemahaman bahasa, dan naturalness suara yang dihasilkan. Meskipun demikian, ukuran model dan kebutuhan komputasi yang tinggi menyebabkan performa yang kurang optimal ketika dijalankan pada perangkat berdaya rendah, sehingga diperlukan strategi optimasi agar sistem dapat berjalan secara efisien. Fokus penelitian ini bukan hanya menilai performa sistem secara keseluruhan, tetapi juga menganalisis bagaimana teknik optimasi seperti *quantization*, ONNX *Runtime optimization*, dan *decoding optimization* dapat diterapkan untuk meningkatkan kecepatan *inference* tanpa menurunkan akurasi atau kualitas

keluaran secara signifikan. Dengan demikian, penelitian ini memberikan landasan bagi pengembangan sistem *voice conversational* AI yang lebih ringan, responsif, dan dapat diimplementasikan secara lebih luas pada perangkat dengan kemampuan komputasi terbatas.

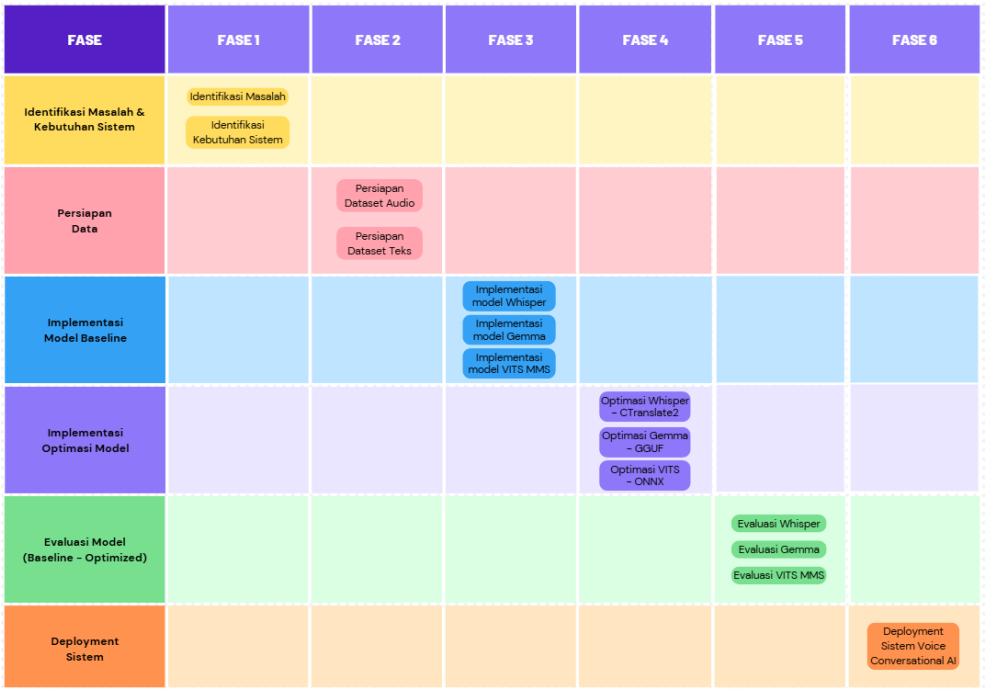
3.2 Metode Penelitian

Penelitian ini merupakan penelitian eksperimental dengan pendekatan kuantitatif yang bertujuan untuk mengukur dan menganalisis efektivitas teknik optimasi *inference* pada sistem *voice conversational* AI berbasis CPU. Penelitian eksperimental dipilih karena penelitian ini melibatkan manipulasi variabel independen berupa teknik-teknik optimasi yang diterapkan pada model, serta pengukuran dampaknya terhadap variabel dependen berupa metrik performa komputasi seperti *latency*, *throughput*, dan efisiensi penggunaan sumber daya. Pendekatan kuantitatif digunakan untuk mengumpulkan data numerik yang dapat dianalisis secara statistik, sehingga memungkinkan perbandingan objektif antara performa model *baseline* dan model yang telah dioptimasi. Fokus penelitian berada pada tahap *inference* atau *deployment* dari model *deep learning*, bukan pada tahap *training* atau data mining, sehingga seluruh eksperimen dan evaluasi dilakukan terhadap model yang sudah matang atau sudah dilatih sebelumnya tanpa melakukan modifikasi pada *weight* model melalui proses re-*training*.

Pendekatan penelitian yang digunakan dalam penelitian ini adalah pendekatan berbasis pengujian performa komputasi pada tahap inferensi, dengan fokus utama pada optimasi *runtime* execution dari model-model *deep learning* yang membentuk sistem *voice conversational AI*. Pendekatan ini berbeda dengan pendekatan *training-based optimization* yang memerlukan modifikasi arsitektur model dan proses pelatihan ulang, penelitian ini menerapkan *inference-level optimization* yang dapat dilakukan pada *pre-trained model* tanpa mengubah *knowledge* yang telah dipelajari oleh model. Teknik optimasi yang diterapkan meliputi *post-training quantization* untuk mengurangi presisi numerik dari *floating-point* 32-bit menjadi *integer* 8-bit, model *runtime conversion* ke format yang lebih efisien seperti CTranslate2 dan ONNX, serta *graph optimization* untuk mengurangi operasi

komputasi yang redundan. Seluruh eksperimen dilakukan pada lingkungan CPU-*only* untuk mencerminkan skenario *deployment* pada perangkat dengan daya komputasi terbatas seperti perangkat atau *server* tanpa GPU dengan biaya operasional rendah, sehingga hasil penelitian dapat memberikan temuan yang relevan untuk aplikasi praktis dalam kondisi terbatas.

3.2.1 Alur Penelitian



Gambar 3. 2 Alur Penelitian

Sesuai dengan gambar 3.2 mengenai alur penelitian dalam penelitian ini dirancang secara sistematis untuk memastikan bahwa setiap tahapan dilaksanakan dengan terstruktur dan terukur, sehingga menghasilkan *output* yang dapat direproduksi dan divalidasi. Alur penelitian ini mengadopsi pendekatan iteratif pada tahap optimasi dan evaluasi, di mana hasil evaluasi digunakan untuk memvalidasi efektivitas teknik optimasi yang diterapkan sebelum melanjutkan ke tahap *deployment* sistem. Berikut merupakan tahapan-tahapan alur penelitian yang dilakukan:

1. **Identifikasi Masalah dan Kebutuhan Sistem**

   Tahap identifikasi masalah dan kebutuhan sistem merupakan langkah awal yang krusial dalam penelitian ini untuk memahami

tantangan yang dihadapi dalam implementasi sistem *voice conversational* AI pada lingkungan dengan daya komputasi terbatas. Masalah utama yang diidentifikasi adalah *latency inference* yang tinggi pada sistem *voice conversational* AI berbasis arsitektur *cascaded* yang terdiri dari komponen ASR, SLM, dan TTS ketika dijalankan pada lingkungan CPU-*only* tanpa akselerasi GPU. *Latency* tinggi ini disebabkan oleh beban komputasi yang besar dari model-model yang digunakan memerlukan operasi *floating-point arithmetic* yang intensif dan penggunaan memori yang besar. Pada lingkungan CPU-*only*, keterbatasan *processing power* dan *memory bandwidth* menyebabkan waktu *inference* untuk setiap komponen menjadi sangat lambat, yang kemudian terakumulasi menjadi total response time yang tidak realistis untuk aplikasi *real-time conversational* AI.

Analisis terhadap karakteristik masing-masing komponen mengidentifikasi bahwa *latency* bersifat akumulatif dalam arsitektur *cascaded*, di mana *output* dari satu komponen harus menunggu hingga komponen sebelumnya selesai memproses, sehingga *bottleneck* pada satu komponen akan berdampak pada keseluruhan performa sistem. Berdasarkan identifikasi masalah tersebut, kebutuhan sistem yang ditetapkan adalah penerapan teknik optimasi *inference* yang dapat mengurangi *latency* dan efisiensi penggunaan sumber daya komputasi pada setiap komponen tanpa memerlukan *re-training* model, sehingga dapat diimplementasikan dengan cepat dan efisien pada model-model yang telah ada.

2. Persiapan Data

Tahap persiapan data dilakukan untuk memastikan ketersediaan dataset yang sesuai dan representatif dalam mengevaluasi performa model *baseline* dan model yang telah dioptimasi pada setiap komponen sistem. Dataset yang digunakan mencakup data *audio* dan data teks berbahasa Indonesia, yang dipilih untuk merepresentasikan skenario penggunaan nyata pada komponen ASR, SLM, dan TTS. Seluruh data disiapkan dengan memperhatikan keberadaan *ground truth* yang valid,

variasi karakteristik *input*, serta kesesuaian dengan tujuan evaluasi masing-masing komponen. Proses *sampling*, kurasi, dan validasi data dilakukan secara konsisten untuk memastikan kualitas data yang digunakan, sehingga hasil evaluasi yang diperoleh dapat mencerminkan performa model secara objektif dan dapat dibandingkan secara adil antara kondisi sebelum dan sesudah optimasi.

3. Implementasi Model *Baseline*

Tahap implementasi model *baseline* dilakukan untuk menetapkan standar performa awal sebelum penerapan teknik optimasi, dengan menjalankan model-model original menggunakan konfigurasi *default* dan *framework inference* standar. Model Whisper Small yang digunakan sebagai komponen ASR yang di-import dari Hugging Face Model Hub dengan *identifier* openai/whisper-small dan diimplementasikan menggunakan *library* Transformers dengan PyTorch sebagai *backend*, serta menggunakan presisi *default floating-point* 32-bit untuk memaksimalkan akurasi transkripsi. Proses *inference* dilakukan dengan menggunakan *pipeline* ASR yang menyediakan *interface high-level* untuk melakukan transkripsi *audio*, dengan *parameter* language *di-set* ke bahasa Indonesia dan *task* diatur ke transcription.

Model Gemma 3 1B instruction-tuned yang digunakan sebagai komponen SLM dimuat dari Hugging Face Model Hub dengan *identifier* google/gemma-3-1b-it dan diimplementasikan menggunakan *library* Transformers dengan PyTorch, menggunakan presisi *default floating-point* 32-bit dan konfigurasi *generation parameter*s *default* untuk mempertahankan kualitas generasi.

Model VITS MMS Indonesian yang digunakan sebagai komponen TTS dimuat dari Hugging Face Model Hub dengan *identifier* facebook/mms-tts-ind dan diimplementasikan menggunakan *library* Transformers dengan PyTorch sebagai *backend*. Proses *inference* dilakukan dengan tokenisasi *input* teks menggunakan VitsTokenizer untuk mengkonversi karakter menjadi *phoneme IDs*, kemudian

memproses melalui model untuk menghasilkan *waveform audio* dengan *sampling rate* 16 kHz. Seluruh implementasi *baseline* dilakukan pada lingkungan CPU-*only* untuk mencerminkan kondisi target *deployment*.

4. **Penerapan Optimasi Model**

Tahap penerapan teknik optimasi model merupakan inti dari penelitian ini, di mana berbagai teknik *inference*-level *optimization* diterapkan pada setiap komponen sistem untuk mengurangi *latency* dan meningkatkan efisiensi komputasi tanpa memerlukan re-*training* model. Optimasi pada komponen Whisper dilakukan melalui dua pendekatan utama yaitu *runtime model conversion* dengan mengkonversi model dari format PyTorch ke format CTranslate2 yang merupakan *inference engine* yang dioptimalkan untuk *deployment* model transformer, serta *post-training quantization* dengan mengurangi presisi numerik dari *floating-point* 32-bit menjadi *integer* 8-bit untuk mengurangi ukuran model dan mempercepat operasi *arithmetic*. Optimasi pada komponen Gemma 3 1B dilakukan melalui penerapan INT8 *quantization* dengan mengubah model ke format ekstensi .gguf. Optimasi pada komponen VITS dilakukan melalui pendekatan *multi-stage* yang mencakup konversi model dari format PyTorch ke ONNX format menggunakan torch.onnx.export, penerapan graph optimization menggunakan ONNX *Runtime* untuk mengurangi operasi redundan melalui *operator fusion* dan *constant folding*, serta *post-training quantization* INT8. Setiap teknik optimasi dipilih berdasarkan karakteristik arsitektur model dan *framework inference* yang paling sesuai, dengan pertimbangan *trade-off* antara peningkatan kecepatan *inference* dan potensi penurunan kualitas *output*.

5. **Evaluasi performa model**

Tahap evaluasi kinerja model dilakukan untuk mengukur efektivitas teknik optimasi yang telah diterapkan dengan membandingkan performa model *baseline* terhadap model yang telah dioptimasi menggunakan berbagai metrik yang relevan untuk setiap komponen. Evaluasi dilakukan secara terpisah untuk setiap komponen dalam

pipeline, yaitu komponen ASR, komponen SLM, dan komponen TTS. Pada komponen ASR, evaluasi mencakup pengukuran ukuran model, pengukuran metrik akurasi berupa *Word Error Rate* dan *Character Error Rate* untuk menilai kualitas transkripsi, serta metrik performa komputasi berupa *latency inference* dan *Real-time Factor* untuk menilai kecepatan pemrosesan terhadap durasi *audio*. Pada komponen SLM, evaluasi mencakup pengukuran ukuran model, pengukuran *latency* untuk menilai kecepatan respons, *throughput* dalam *tokens per second* untuk menilai efisiensi *generation*, serta *perplexity* untuk menilai kualitas model bahasa. Pada komponen TTS, evaluasi mencakup pengukuran *latency* sintesis untuk menilai kecepatan *generation audio*, *Real-time Factor* untuk menilai kemampuan sintesis secara *real-time*, serta ukuran model.

6. ***Deployment* Sistem**

Tahap *deployment* sistem merupakan tahap akhir dari penelitian ini yang bertujuan untuk mengintegrasikan seluruh komponen yang telah dioptimasi ke dalam satu *pipeline end-to-end* yang membentuk sistem *voice conversational* AI yang utuh dan siap untuk digunakan. *Deployment* dilakukan setelah evaluasi individual component selesai dan hasil evaluasi menunjukkan bahwa teknik optimasi yang diterapkan berhasil meningkatkan performa *inference* dengan *trade-off* yang *acceptable* pada kualitas *output*. Proses *deployment* mencakup pembuatan class *pipeline* yang mengenkapsulasi ketiga model *optimized* beserta konfigurasi *inference* masing-masing komponen. *Pipeline* terintegrasi dirancang untuk dapat memproses *input audio* dari pengguna dan menghasilkan *output audio* sebagai respons dengan *interface* sederhana.

3.3 Teknik Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan dengan memanfaatkan data sekunder yang berasal dari dataset publik yang telah tersedia dan tervalidasi untuk

tugas evaluasi model AI berbahasa Indonesia. Penggunaan data sekunder dipilih karena dataset-dataset tersebut telah melalui proses kurasi dan validasi oleh peneliti sebelumnya, sehingga memiliki kualitas dan reliabilitas yang terjamin untuk keperluan evaluasi performa model. Seluruh dataset yang digunakan dalam penelitian ini bersifat *open-source* dan dapat diakses secara bebas melalui *platform* repositori data publik.

3.3.1 *Dataset* Evaluasi ASR (Whisper)

Tabel 3. 1 Informasi *Dataset* Evaluasi ASR

Aspek	**Keterangan**
Nama	Mozilla Common Voice Indonesian - Scripted Speech 23.0
Sumber	Mozilla Foundation
URL	https://datacollective.mozillafoundation.org/datasets/cmflnuzw5yf6ccihtcnfs6ll3
Bahasa	Bahasa Indonesia
Jenis Data	*Audio recordings* dengan *ground truth transcription*
Format *Audio*	MP3, 48kHz *sampling rate*
Jumlah Data	59.500 klip *audio*
Jumlah Sampel Data	50 sampel *audio*
Durasi *Audio*	Bervariasi (3-10 detik/klip)
Metode *Sampling*	*Random sampling*
Karakteristik *Speaker*	Beragam pembicara, berbagai gender, aksen regional Indonesia
Kualitas Rekaman	Kualitas suara bersih dari *noise*
Ground Truth	Transkrip teks
Lisensi	CC0 (Public *Domain*)
Penggunaan	Evaluasi performa ASR (*input*: *audio* → *output*: teks)

Merujuk pada tabel 3.1, dataset untuk evaluasi komponen ASR diperoleh dari Mozilla Common Voice Indonesian, yaitu dataset Common Voice Scripted Speech 23.0 – Indonesian [87]. Dataset ini merupakan kumpulan rekaman suara dalam bahasa Indonesia yang dikontribusikan oleh penutur asli dengan variasi karakteristik akustik yang mencakup perbedaan aksen regional, gender *speaker*, kualitas perekaman, dan kondisi lingkungan *recording*. Setiap sampel *audio* dalam dataset dilengkapi dengan *ground truth transcription* yang akurat. Proses

sampling dilakukan untuk memilih 50 sampel *audio* secara random dari keseluruhan dataset Common Voice, di mana jumlah 50 sampel ini merepresentasikan ukuran dataset evaluasi yang optimal dan ideal untuk assessment performa model ASR berdasarkan *best practices* dalam penelitian *speech recognition* [70]. Ketentuan sampel yang diambil adalah sampel yang memiliki durasi *audio* berkisar antara 3-10 detik sesuai dengan simulasi waktu rata-rata *input* pertanyaan atau instruksi pengguna ke sistem, dan kualitas rekaman yang stabil dan bersih dari *noise*. Ketentuan ini ditetapkan untuk melihat performa model secara stabil dan sesuai dengan kebutuhan pengguna

3.3.2 *Dataset* Evaluasi SLM (Gemma 3 1B) dan TTS (VITS)

Tabel 3. 2 Informasi *Dataset* Evaluasi SLM dan TTS

Aspek	Keterangan
Nama	*Dataset* Evaluasi SLM & TTS Bahasa Indonesia
Sumber	*Manual synthetic dataset*
Bahasa	Bahasa Indonesia
Jenis Data	Teks tertulis dengan variasi panjang kalimat (pendek, sedang, panjang)
Format Data	30 kalimat dalam format *plain text* dengan anotasi kategori dan panjang
Kategori Kalimat	Teknologi, kesehatan, pendidikan, bisnis, kehidupan sehari-hari
Jenis Kalimat	Deklaratif, interogatif, imperatif
Variasi Panjang	Pendek (5-10 kata), Sedang (11-20 kata), Panjang (21-40 kata)
Jumlah Sampel	30 kalimat
Jumlah Sampel Data	100 pertanyaan
Durasi *Audio*	Bervariasi (3-10 detik/klip)
Metode Sampling	*Manual curation* dengan kriteria diversitas linguistik dan kompleksitas semantik
Penggunaan	*Perplexity measurement* (SLM), *Real-time Factor* (TTS)

Sesuai dengan Tabel 3.2, *Dataset* evaluasi untuk SLM dan TTS disusun secara manual dengan fokus pada diversitas konten dan kompleksitas linguistik berbahasa Indonesia. Dataset ini terdiri dari 30 kalimat yang dirancang dengan

variasi panjang (pendek: 5-10 kata, sedang: 11-20 kata, panjang: 21-40 kata) dan mencakup berbagai kategori semantik seperti teknologi, kesehatan, pendidikan, bisnis, dan kehidupan sehari-hari, dengan jenis kalimat yang beragam meliputi deklaratif, interogatif, dan imperatif untuk mengevaluasi kemampuan generalisasi model pada berbagai konteks komunikatif. Pendekatan *synthetic dataset* dengan *controlled diversity* ini telah divalidasi dalam literatur terkini sebagai metode efektif untuk evaluasi model bahasa dan *speech synthesis*, yang menggunakan *synthetic data generation* dengan struktur dan konten beragam untuk meningkatkan performa *text embedding models*, dengan berbagai variasi konten untuk evaluasi sistem TTS yang *robust* [88]. Metode *sampling* manual dengan kriteria diversitas linguistik memastikan *coverage* yang komprehensif terhadap karakteristik bahasa Indonesia, memungkinkan pengukuran *perplexity* untuk evaluasi SLM dan *Real-time Factor* (RTF) untuk TTS secara reliabel.

3.4 Teknik Optimasi Model

Teknik optimasi model yang diterapkan dalam penelitian ini berfokus pada *inference*-level optimization yang tidak memerlukan *re-training* model, sehingga dapat diimplementasikan secara efisien pada model-model *pre-trained* yang telah ada. Pendekatan optimasi dilakukan secara terpisah untuk setiap komponen dalam *pipeline* dengan mempertimbangkan karakteristik arsitektur model dan *framework inference* yang paling sesuai untuk masing-masing komponen.

3.4.1 Optimasi Model ASR (Whisper)

Optimasi model Whisper sebagai komponen ASR dilakukan melalui dua pendekatan utama yaitu konversi model ke CTranslate2 *runtime* dan penerapan *post-training quantization* INT8. CTranslate2 merupakan *inference engine* yang dirancang khusus untuk *deployment* model transformer dengan fokus pada efisiensi eksekusi di CPU dan GPU, yang menyediakan implementasi operator yang dioptimalkan untuk arsitektur transformer melalui teknik-teknik seperti *kernel fusion*, *memory layout optimization*, dan *efficient batching*. Proses konversi model dilakukan dengan menggunakan *tool* ct2-transformers-*converter* yang mengambil *checkpoint* model Whisper dalam format Hugging

Face Transformers dan mengkonversinya ke format CTranslate2 yang dioptimalkan untuk *inference* cepat. Selama proses konversi, dilakukan *restructuring computation graph* untuk mengeliminasi operasi yang tidak diperlukan pada waktu *inference* dan mengoptimalkan pola *memory access* agar sesuai dengan karakteristik *cache hierarchy* pada CPU. *Post-training quantization* INT8 diterapkan dengan mengkonversi *weight* dan *activation* dari presisi *floating-point* 32-bit menjadi *integer* 8-bit menggunakan *symmetric quantization scheme*, di mana nilai *floating-point* di-map ke *range integer* [-127, 127] menggunakan *scaling factor* yang dihitung *berdasarkan distribution statistics* dari *weight* atau *activation*.

3.4.2 Optimasi Model SLM (Gemma 3 1B)

Optimasi model Gemma 3 1B sebagai komponen SLM dilakukan melalui konversi model ke format GGUF (GPT-Generated Unified Format) yang merupakan format *binary* teroptimasi untuk *inference Large Language Model* berbasis CPU. Format GGUF dikembangkan sebagai evolusi dari format GGML dengan menambahkan *metadata* terstruktur yang mencakup informasi arsitektur model, *vocabulary*, dan *hyperparameter* dalam *single file format* yang *self-contained*, sehingga memungkinkan *loading* model yang lebih efisien dan *portable* di beragam *inference engine*s. Proses optimasi dimulai dengan konversi model Gemma 3 1B dari format Hugging Face Transformers ke format GGUF menggunakan *conversion script* yang memetakan *weight tensors*, *vocabulary embeddings*, dan *layer configurations* ke dalam sktruktur GGUF yang telah didefinisikan. Setelah konversi, dilakukan *quantization* dengan teknik Q8_0 (8-bit *quantization*) di mana *weight matrix* di-*quantize* menjadi 8-bit *integer values* menggunakan k-means *clustering* untuk mengelompokkan nilai *weight* ke dalam 16 representasi *clusters*, dengan mempertahankan *separate scaling factors per group of values* untuk meminimalkan *quantization error*.

Format GGUF mendukung *mixed-precision quantization* di mana *critical layers* seperti *attention mechanism* dan *output layers* dapat dipertahankan dalam presisi lebih tinggi sementara *fully connected layers* di-*quantize* ke 8-bit,

sehingga memberikan keseimbangan optimal antara *memory footprint reduction* dan *model accuracy preservation*. *Inference* menggunakan GGUF model dilakukan melalui llama.cpp *backend* atau *compatible inference engine*s yang mengimplementasikan *optimized* CPU *kernels* untuk *matrix multiplication operations* dengan *quantized weight*s, termasuk *Single Instruction Multiple Data* (SIMD) *optimizations* dan *cache-aware memory access patterns* yang secara signifikan meningkatkan *throughput* pada CPU *inference*. Pendekatan GGUF *quantization* ini dipilih karena memberikan *compression ratio* sambil mempertahankan degradasi minimal pada kualitas model melalui careful *weight clustering* dan *selective layer quantization* strategies.

3.4.3 Optimasi Model TTS (VITS)

Optimasi model VITS sebagai komponen Text-to-Speech dilakukan melalui pendekatan multi-stage yang mencakup konversi model dari format PyTorch ke ONNX format, penerapan *graph optimization* menggunakan ONNX *Runtime*, dan *post-training quantization* INT8. Konversi model ke ONNX dilakukan menggunakan torch.onnx.export yang melakukan *tracing* terhadap model PyTorch dengan *sample input* untuk menghasilkan *computation graph* dalam format ONNX. ONNX format dipilih karena menyediakan *standardized representation* yang dapat dioptimasi oleh berbagai *inference engine*s dan mendukung *portability* antar *platform*s. Setelah konversi ke ONNX, dilakukan *graph optimization* menggunakan ONNX *Runtime* yang menerapkan berbagai *optimization passes* seperti *operator fusion* untuk menggabungkan beberapa operasi menjadi *single fused operation, constant folding* untuk mengevaluasi operasi dengan *constant input*s pada saat kompilasi sehingga tidak perlu dieksekusi pada waktu *inference*, dan *dead code elimination* untuk menghapus opearasi yang *output*-nya tidak digunakan dalam hasil akhir. *Graph optimization* juga melakukan *layout transformation* untuk mengoptimalkan *memory access pattern* dan *tensor reshaping* untuk mengurangi *overhead* dari *dynamic shape operations*.

Setelah model tersedia dalam format ONNX, ONNX *Runtime* digunakan untuk melakukan *graph optimization* seperti *node fusion*, penghapusan operasi

yang redundan, serta pemilihan kernel eksekusi yang lebih efisien untuk CPU. Selain itu, dilakukan juga *post-training quantization* pada model ONNX untuk menurunkan presisi bobot dan, bila relevan, aktivasi menjadi representasi *integer* yang lebih rendah. Kombinasi optimasi graf dan *quantization* ini diharapkan dapat mengurangi *latency* proses sintesis suara dan menurunkan penggunaan memori tanpa mengurangi kualitas *audio* secara signifikan.

3.5 Teknik Evaluasi Model

Evaluasi performa model dalam penelitian ini dilakukan secara terpisah untuk setiap komponen dalam *pipeline* dengan menggunakan metrik yang relevan untuk mengukur baik kualitas *output* maupun efisiensi komputasi. Pendekatan evaluasi dirancang untuk memberikan perbandingan objektif antara model *baseline* yang menggunakan presisi *floating-point* 32-bit dengan *framework inference* standar terhadap model yang telah dioptimasi menggunakan teknik *quantization* dan *runtime model conversion*. Metrik evaluasi mencakup dua kategori utama yaitu metrik kualitas *output* untuk menilai apakah optimasi menyebabkan degradasi pada akurasi atau kualitas dari hasil *inference*, dan metrik performa komputasi untuk menilai seberapa efektif teknik optimasi dalam mengurangi *latency*, meningkatkan *throughput*, dan mengurangi ukuran model. Seluruh evaluasi dilakukan pada dataset yang telah disiapkan sebelumnya dengan jumlah sampel yang representatif, serta pengukuran dilakukan pada lingkungan dan dataset yang sama sebagai kontrol eksperimen untuk memastikan perbandingan yang adil dan konsisten.

3.5.1 Evalulasi Model ASR (Whisper)

Tabel 3. 3 Metrik Evaluasi Model ASR

Metrik	Keterangan
Word Error Rate (WER)	Persentase kesalahan transkrip kata
Character Error Rate (CER)	Persentase kesalahan transkrip karakter
Real-time Factor	Rasio waktu pemrosesan transkrip dengan durasi *input*
Latency	Waktu pemrosesan transkrip
Model Size	Ukuran model

Sesuai dengan Tabel 3.3, evaluasi model Whisper sebagai komponen ASR dilakukan dengan mengukur metrik akurasi transkripsi untuk menilai sejauh mana optimasi mempertahankan kualitas hasil pengenalan ujaran. Evaluasi akurasi menggunakan *Word Error Rate* (WER) sebagai metrik utama, yang mengukur persentase kesalahan transkripsi pada level kata dengan menghitung jumlah *substitution*, *insertion*, dan *deletion* yang diperlukan untuk mengubah transkrip hasil prediksi menjadi transkrip *ground truth* [89]. WER memberikan indikasi langsung terhadap *usability* hasil transkripsi untuk aplikasi *downstream*, di mana nilai WER yang tinggi mengindikasikan banyaknya kesalahan yang dapat mengganggu pemahaman konten *audio*. Sebagai metrik komplementer, *Character Error Rate* (CER) digunakan untuk mengukur kesalahan pada level karakter dengan pendekatan perhitungan yang serupa, namun dilakukan pada unit karakter [90]. CER memberikan *insight* tambahan mengenai pola kesalahan transkripsi dan cenderung lebih *robust* terhadap kesalahan kecil seperti *typo* atau variasi ejaan dibandingkan WER yang menghitung satu kata sebagai kesalahan meskipun hanya satu karakter yang salah. Pengukuran WER dan CER dilakukan dengan membandingkan hasil transkripsi model terhadap transkrip *ground truth* dari dataset Common Voice Bahasa Indonesia yang telah dinormalisasi, menggunakan *library* evaluasi seperti *jiwer* untuk menghitung *edit distance* antara transkripsi prediksi dan referensi [90], [91].

Selain akurasi, evaluasi performa komputasi dilakukan untuk menilai efisiensi *inference* model dalam skenario *voice conversational* AI. Metrik *Real-time Factor* (RTF) digunakan untuk mengukur rasio antara total waktu pemrosesan transkripsi terhadap total durasi *audio input*, di mana nilai RTF yang lebih kecil dari 1.0 menunjukkan bahwa model mampu memproses *audio* lebih cepat daripada durasi aslinya [89]. *Inference latency* diukur sebagai waktu yang diperlukan untuk memproses satu sampel *audio* sejak *audio* diterima sebagai *input* hingga transkrip dihasilkan sebagai *output*, mencakup tahapan *preprocessing*, *forward pass* melalui *encoder* dan *decoder*, serta *post-processing* hasil transkripsi [91]. Untuk mengurangi variansi akibat

measurement noise, pengukuran *latency* dilakukan dengan *multiple runs* pada setiap sampel dan dirangkum dalam bentuk statistik deskriptif berupa nilai rata-rata dan simpangan baku. Selain itu, *model size* diukur sebagai ukuran file model dalam satuan megabytes, yang merepresentasikan kebutuhan penyimpanan model pada saat *deployment* dan menjadi indikator efisiensi implementasi, khususnya pada lingkungan dengan keterbatasan sumber daya [21].

Perbandingan antara model *baseline* dan model yang telah dioptimasi dilakukan dengan menghitung seluruh metrik evaluasi pada dataset yang sama menggunakan prosedur pengujian yang konsisten, sebagaimana direkomendasikan dalam literatur evaluasi ASR untuk menjamin keadilan dan keterbandingan hasil. Hasil evaluasi kemudian dianalisis dengan membandingkan besarnya peningkatan performa komputasi, seperti penurunan nilai RTF dan *latency* serta pengurangan ukuran model, terhadap potensi degradasi akurasi yang ditunjukkan oleh kenaikan WER dan CER. Analisis ini bertujuan untuk mengidentifikasi *trade-off* antara kecepatan dan kualitas transkripsi, serta menentukan apakah optimasi yang diterapkan menghasilkan performa yang masih dapat diterima (*acceptable trade-off*) untuk kebutuhan aplikasi target.

3.5.2 Evaluasi Model SLM (Gemma 3 1B)

Tabel 3. 4 Metrik Evaluasi SLM

Metrik	Keterangan
Perplexity	Tingkat *confident* model dalam memprediksi *next token*
Latency	Rasio waktu pemrosesan transkrip dengan durasi *input*
Throughput	Kecepatan generasi token per detik
Model Size	Ukuran model

Sesuai dengan table 3.4, evaluasi model Gemma 3 1B sebagai komponen Small Language Model mencakup pengukuran metrik kualitas model bahasa dan metrik performa generation untuk menilai efektivitas *quantization* dalam mempercepat *text generation* sambil mempertahankan kemampuan model

dalam menghasilkan respons yang relevan dan koheren. Model size diukur sebagai ukuran *file* model dalam satuan *megabytes* antara model *baseline* dengan presisi *floating-point* 32-bit dan model *quantized* dengan presisi *integer* 8-bit, di mana reduksi model size mengindikasikan manfaat dari *quantization* dalam mengurangi kebutuhan *storage* [92]. *Latency generation* diukur sebagai total waktu yang diperlukan untuk menghasilkan *complete response* dari model, mencakup waktu untuk *tokenization input, iterative generation* process di mana model menghasilkan token satu per satu secara *autoregressive*, dan *detokenization output* [92]. *Latency* generation sangat bergantung pada panjang *output* yang dihasilkan karena *nature autoregressive* dari *language model*, sehingga normalisasi terhadap *output length* diperlukan untuk perbandingan yang adil antar sampel dengan kompleksitas yang berbeda.

Tokens per second merupakan metrik *throughput* yang mengukur kecepatan *generation* dalam satuan *tokens* yang dihasilkan per detik [93]. TPS memberikan indikasi langsung mengenai efisiensi proses generasi. Peningkatan TPS mengindikasikan bahwa optimasi berhasil mempercepat *per-token generation time*, yang secara agregat menghasilkan reduksi dalam total *latency* untuk menghasilkan respons dengan panjang tertentu. *Perplexity* digunakan sebagai metrik untuk menilai kualitas dari model bahasa dengan mengukur seberapa *confident* model dalam memprediksi token selanjutnya dalam *sequence* [94]. *Perplexity* dihitung sebagai eksponensial dari rata-rata negatif *log-likelihood* seluruh token pada dataset uji [95]. *Perplexity* yang lebih rendah mengindikasikan bahwa model memiliki *confidence* yang lebih tinggi dan *better fit* terhadap *data distribution*, sementara *perplexity* yang tinggi mengindikasikan peningkatan ketidaktentuan yang dapat diindikasikan sebagai rendahnya kualitas atau *less coherent generated text*. Perbandingan *baseline* dengan *optimized model* dilakukan dengan menganalisis persentase perubahan pada setiap metrik, di mana optimasi yang ideal menghasilkan peningkatan signifikan pada *latency* dan TPS dengan peningkatan seminimal mungkin pada perplexity.

3.5.3 Evaluasi Model TTS (VITS)

Tabel 3. 5 Metrik Evaluasi Model TTS

Metrik	Keterangan
Real-time Factor	Rasio waktu pemrosesan transkrip dengan durasi *input*
Latency	Waktu pemrosesan transkrip
Model Size	Ukuran model

Sesuai dengan Tabel 3.5, evaluasi model TTS VITS dilakukan pada pengukuran kualitas dan performa model. Evaluasi performa sintesis dilakukan untuk menilai efektivitas optimasi dalam mempercepat proses generasi *audio*. *Inference latency* diukur sebagai total waktu yang diperlukan untuk mensintesis *audio output* dari *input* teks, mencakup tahapan *tokenization* teks menjadi urutan fonem, *forward pass* melalui *text encoder* untuk menghasilkan fitur linguistik, prediksi durasi setiap fonem oleh *duration predictor*, serta generasi gelombang *audio* oleh *neural vocoder* [96]. Pengukuran *latency* dilakukan dengan merekam *timestamp* sebelum dan sesudah *synthesis call* untuk setiap kalimat dalam dataset evaluasi, dengan *multiple runs* untuk mengurangi variansi pengukuran dan menghasilkan statistik yang reliabel [96]. Untuk memastikan perbandingan yang adil antar sampel, *latency* dianalisis dengan mempertimbangkan panjang teks *input* dan durasi *audio output* yang dihasilkan. *Real-time Factor* (RTF) diukur sebagai rasio antara waktu sintesis terhadap durasi *audio* yang dihasilkan, di mana nilai RTF < 1.0 mengindikasikan bahwa model mampu mensintesis *audio* lebih cepat dari waktu nyata, yang merupakan kebutuhan krusial untuk aplikasi *voice conversational* AI. Selain itu, *model size* diukur sebagai ukuran file model dalam satuan *megabytes* untuk membandingkan *memory footprint* antara model PyTorch *baseline* dan model ONNX yang telah di-*quantize* [97]. Perbandingan performa dilakukan dengan menghitung *percentage reduction* ukuran model dan *speedup ratio* dari setiap tahapan optimasi, serta merangkum seluruh metrik menggunakan statistik deskriptif seperti nilai rata-rata, simpangan baku, minimum, dan maksimum untuk memastikan konsistensi evaluasi antar varian model.

Perbandingan antara model VITS *baseline* dan model yang telah dioptimasi dilakukan dengan mengukur seluruh metrik kualitas dan performa pada dataset evaluasi yang sama menggunakan prosedur pengujian yang konsisten, sebagaimana direkomendasikan dalam literatur evaluasi TTS untuk menjamin keterbandingan hasil. Evaluasi metrik performa seperti *latency*, *Real-time Factor*, dan *model size* digunakan untuk mengukur peningkatan efisiensi *inference* dan kelayakan sistem untuk aplikasi *voice conversational* AI yang bersifat interaktif.