

BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

Bab ini memaparkan konsep-konsep fundamental dan rujukan literatur terkait kecerdasan buatan, model bahasa, keterbatasan perangkat keras, serta teknik optimasi memori komputasi yang menjadi landasan teoritis bagi penelitian ini.

2.1.1. Natural Language Processing (NLP) dan Pergeseran ke Edge Computing

Natural Language Processing (NLP) merupakan cabang ilmu kecerdasan buatan yang memungkinkan mesin mengurai, memahami, dan membangkitkan bahasa alami manusia secara kontekstual. Perkembangan NLP terus meningkat seiring meluasnya penerapan algoritma *Deep Learning*. Salah satu contoh penerapan NLP di industri lokal adalah *chatbot* kesehatan berbasis kecerdasan buatan yang dapat meningkatkan efisiensi layanan rumah sakit dan menjadi kanal konsultasi *real-time* bagi masyarakat (Fadillah Ahmad et al., 2024).

Namun, pemrosesan bahasa yang kompleks sering membutuhkan komputasi server *cloud*. Ketergantungan pada *cloud computing* ini menimbulkan tantangan pada data yang bersifat rahasia, seperti rekam medis dan diagnosis pasien. Untuk mengurangi risiko keamanan, pengembangan NLP mulai diarahkan ke pemrosesan lokal pada perangkat seluler (*Mobile Edge Computing*). Pendekatan *offline* ini menjaga sistem AI tetap berfungsi sekaligus melindungi privasi data dari paparan jaringan internet terbuka (Zhan et al., 2025).

Untuk mendukung operasi AI pada perangkat dengan RAM terbatas, diperlukan pendekatan *Edge Intelligence Optimization*. Pendekatan ini menyesuaikan *software stack* agar perangkat Android berspesifikasi rendah dapat menjalankan beban kerja AI tanpa memicu *Force Close* yang diakibatkan oleh keterbatasan arsitektur memori bawaan (Zhang et al., 2024). Secara hierarkis, ekosistem *Edge Intelligence* terbagi menjadi tiga lapisan: lapisan *cloud* (server pusat untuk pelatihan dan model berskala raksasa), lapisan *edge server* (*gateway* perantara dengan latensi rendah), dan lapisan *edge device* (perangkat akhir seperti *smartphone*); penelitian ini memfokuskan kontribusi pada lapisan paling bawah, yaitu inferensi SLM pada *smartphone* Android.

2.1.2. Large Language Models (LLM) dan Fleksibilitasnya

Large Language Model (LLM) merupakan arsitektur *neural network* berskala besar yang dilatih menggunakan data teks dalam jumlah triliunan token. Pendekatan paradigmatis LLM modern, mulai dari *pre-training*, *prompting*, hingga *alignment* dan *inference*, secara komprehensif disusun oleh (Xiao & Zhu, 2025) dalam buku ajar *Foundations of Large Language Models*. Sebagai fondasi dasar, LLaMA merupakan salah satu pelopor *open-source foundation language model* berskala besar yang dibangun murni menggunakan dataset publik, dengan kapasitas parameter mulai 7 miliar hingga 65 miliar (Touvron et al., 2023). Arsitektur LLM bahkan telah dimanfaatkan untuk membantu otomatisasi pengelolaan jaringan nirkabel di kawasan urban (Sevim et al., 2024), yang menunjukkan luasnya peran model bahasa modern.

2.1.3. Small Language Models (SLM) dan Limitasi Perangkat Seluler

Untuk menekan tingginya biaya inferensi LLM berukuran besar, dikembangkan *Small Language Model (SLM)*. SLM memiliki jumlah parameter yang lebih sedikit (umumnya 1–7 miliar parameter), tetapi tetap mampu mempertahankan

kemampuan analisis yang memadai. Survei komprehensif atas lebih dari 60 SLM oleh Lu et al. (2025) menunjukkan bahwa SLM *state-of-the-art* dapat mengungguli model 7B pada tugas umum, sehingga viabilitas praktis SLM sebagai pilihan utama *on-device* terbukti secara empiris. Ukurannya yang ringkas membuat SLM cocok untuk dijalankan langsung pada *smartphone*. Namun tantangan utamanya adalah arsitektur *shared memory*, di mana kapasitas RAM 8 GB pada Android harus dibagi untuk sistem operasi, antarmuka layar, dan aplikasi latar belakang. Apabila ukuran SLM melebihi memori yang tersedia, *Out of Memory Killer* akan menghentikan proses AI secara paksa (*Force Close*) untuk mencegah sistem mengalami *freeze*.

2.1.4. Arsitektur Dasar Transformer

Kemampuan LLM maupun SLM bertumpu pada arsitektur *Transformer* dengan mekanisme *Self-Attention*, yang memungkinkan model membaca seluruh kata dalam satu kalimat secara bersamaan dan menimbang hubungan makna antar-token secara kontekstual (Alammar & Grootendorst, 2024; Prince, 2026). Kalkulasi bobot atensi antar-token ini melibatkan operasi matriks masif yang dilakukan lapis demi lapis (*layer-by-layer*), sehingga *Transformer* membutuhkan memori yang besar dan konsumsinya meningkat seiring bertambah panjangnya *prompt* dari pengguna.

2.1.5. Konsep Kuantisasi, Presisi Campuran, dan Rentang 3-Bit hingga 5-Bit

Kuantisasi adalah teknik kompresi untuk mengecilkan ukuran file model AI dan menghemat konsumsi RAM (Prince, 2026). Pada kondisi aslinya, bobot jaringan model bahasa disimpan dalam format desimal berpresisi tinggi (FP16). Format ini memiliki akurasi tinggi, tetapi membutuhkan kapasitas memori yang besar. Kuantisasi menyederhanakan rangkaian pecahan desimal ini menjadi bilangan bulat yang lebih padat (INT5, INT4, atau bahkan INT3) (Dettmers et al., 2023). Pembahasan teknik

inferensi LLM yang efisien beserta strategi kuantisasi modern pada tahap pasca-pelatihan dirangkum secara komprehensif oleh (Xiao & Zhu, 2025). Selain keluarga PTQ klasik (GPTQ, AWQ, dan GGUF *k-quants*), studi kontemporer juga mengusulkan metode PTQ yang khusus dirancang ramah eksekusi *on-device*, seperti MobileQuant dari Samsung AI yang mengoptimasi *weight transformation* dan rentang aktivasi secara *end-to-end* untuk menekan *latency* hingga 20–50% pada perangkat seluler (Tan et al., 2024).

Pada awalnya, teknik kompresi standar di *llama.cpp* menerapkan jumlah bit yang sama untuk seluruh lapisan model. Pendekatan ini memiliki kelemahan, yaitu rusaknya bobot penting sehingga model lebih mudah menghasilkan keluaran yang keliru (halusinasi). Untuk mengatasi penurunan kualitas ini, dikembangkan metode *K-Quants* (ditandai huruf “K”) yang merupakan kontribusi komunitas pengembang *llama.cpp*. Metode ini tidak memukul rata semua ukuran kompresi, melainkan menggunakan presisi campuran (*mixed-precision*). Lapisan jaringan yang penting untuk logika dipertahankan di ukuran bit yang lebih besar (seperti 6-bit). Sementara itu, bagian jaringan yang dianggap kurang krusial akan ditekan lebih ekstrem ke ukuran 3-bit atau 5-bit agar RAM tidak penuh. Dalam eksplorasi *resource-constrained edge*, tiga varian *k-quants* yang lazim dievaluasi secara bertahap adalah Q5_K_M (rata-rata mendekati 5-bit), Q4_K_M (mendekati 4-bit), dan Q3_K_M (mendekati 3-bit). Pencarian titik optimal (*sweet spot*) menjadi penting: kompresi yang terlalu ringan (>5-bit) belum cukup menekan penggunaan RAM 8 GB, sedangkan kompresi yang terlalu agresif (≤ 3 -bit) berisiko menurunkan kualitas model secara drastis. Konsep dasar PTQ yang melandasi varian Q*_K_M tersebut bersumber dari keluarga kuantisasi terdahulu seperti QLoRA (Dettmers et al., 2023) dan GPTQ (Frantar et al., 2023), sedangkan implikasinya pada perangkat seluler dianalisis secara mendalam

oleh (Tan et al., 2024). Visualisasi praktis arsitektur *Transformer* dan teknik tokenisasi serta *embedding* yang menjadi prasyarat pemahaman mekanisme kuantisasi pada lapisan model dijelaskan secara komprehensif oleh (Alammar & Grootendorst, 2024).

2.1.6. Karakteristik Inferensi Mobile: Limitasi CPU dan Bandwidth Memori

Pada arsitektur *System-on-Chip* (SoC) ARM big.LITTLE, kecepatan inferensi model AI dipengaruhi oleh dua faktor utama. Pertama adalah keterbatasan *memory bandwidth*, yaitu kondisi ketika CPU dapat menghitung dengan cepat, tetapi aliran data parameter SLM justru terhambat pada jalur transfer RAM. Ketika model FP16 dikuantisasi ke 5-bit hingga 3-bit, berkas model menjadi sangat ringan sehingga kemacetan transfer data dari RAM ke CPU ARM terurai. Hal inilah yang menjelaskan mengapa pengurangan ukuran model berkorelasi langsung dengan peningkatan *Tokens per Second* (TPS).

2.1.7. Format File GGUF (GPT-Generated Unified Format)

GGUF adalah format biner yang dirancang khusus untuk menyimpan arsitektur *neural network* AI generatif dalam satu berkas. Diciptakan sebagai evolusi dari GGML, GGUF mampu menampung seluruh parameter identitas model beserta metadatanya. Format ini penting bagi lingkungan Linux Android melalui *Termux* karena strukturnya dioptimalkan untuk eksekusi pada CPU dan mendukung berbagai skema kompresi presisi campuran (*K-Quants*).

2.1.8. Framework Inferensi llama.cpp di Android

Pada perangkat konsumen ringan, komputasi *Machine Learning* banyak bergantung pada *llama.cpp*. Berbeda dengan *framework* berbasis Python seperti *PyTorch* atau *TensorFlow* yang memiliki *overhead* pustaka besar, *llama.cpp* ditulis dalam bahasa C/C++ secara *bare-metal*. Struktur ini memungkinkan kompilasi manual

yang disesuaikan dengan instruksi CPU ARM pada Android, sehingga perangkat dapat menjalankan komputasi *Transformer* secara efisien (Ray & Pradhan, 2026).

2.1.9. Tantangan Gangguan Matematis (Perturbation)

Pengurangan presisi bobot dari format desimal menjadi bilangan bulat selalu menimbulkan efek samping. Dalam kajian kuantisasi, dampak simplifikasi numerik ini disebut sebagai *perturbation*. Kendala utama adalah keberadaan nilai pencilan (*outliers*) yang sulit diprediksi dan rentan hilang saat nilai dibulatkan ke presisi 3-bit hingga 5-bit. Hilangnya *outliers* ini dapat melemahkan parameter atensi model, sehingga model lebih mudah menghasilkan keluaran yang keliru atau jawaban kosong (Gong et al., 2024). Oleh karena itu, mengevaluasi titik batas kerusakan kompresi atau *degradation* memegang peranan krusial.

2.1.10. Metrik Evaluasi Performa Edge Computing

Kelayakan operasional SLM pada Android atau *Mobile Edge* dievaluasi melalui rasio efisiensi perangkat keras. Penelitian ini menggunakan dua parameter utama. Pertama, *System Memory Usage* (konsumsi RAM sistem) sebagai indikator utama untuk mengukur kapasitas aktual memori yang dialokasikan oleh model saat proses pemuatan (*loading*). Parameter ini harus dipertahankan pada ambang batas aman antara 1 hingga 3 GB untuk mencegah sistem operasi Android mengaktifkan mekanisme *Out-of-Memory (OOM) Killer*. Kedua, *Tokens per Second (TPS)* sebagai tolok ukur kecepatan pemrosesan *prompt* dan kecepatan pembangkitan token pada tahap generasi teks.

2.1.11. Standar Pengujian Kognitif dan Linguistik

Untuk memastikan bahwa penghematan memori tidak menurunkan kemampuan model secara signifikan, diperlukan beberapa pengujian standar. Metrik pertama adalah *Perplexity* (PPL) untuk menilai kualitas prediksi Bahasa, semakin rendah nilai PPL, semakin baik kalimat yang dihasilkan. Metrik kedua adalah *Massive Multitask Language Understanding* (MMLU), yaitu dataset untuk menguji pengetahuan model dan mendeteksi penurunan pengetahuan faktual setelah kompresi. Metrik ketiga adalah *Grade School Math 8K* (GSM8K) untuk menguji kemampuan penalaran matematika. Terakhir, *HumanEval* digunakan untuk menguji kemampuan logika pemrograman Python model.

2.2. Penelitian Terkait

Untuk menempatkan orisinalitas penelitian ini di dalam literatur yang relevan, peneliti merujuk pada berbagai studi terdahulu yang berfokus pada analisis teknis implementasi kecerdasan buatan (AI) di lingkungan dengan memori terbatas. (Dettmers et al., 2023) memperkenalkan teknik kompresi *QLoRA* yang berkaitan dengan efisiensi memori. Studi tersebut menunjukkan bahwa pemadatan bobot ke presisi 4-bit (NF4) dapat menurunkan *memory footprint* hingga sepertiganya. Metode ini juga mencegah sistem kehabisan memori (*Out of Memory*) tanpa mengurangi kinerja arsitektur dasarnya yang berpresisi 16-bit. Argumen efisiensi ini diperkuat oleh dua pendekatan pengembangan *Post-Training Quantization* (PTQ) yang berbeda namun saling melengkapi.

(Frantar et al., 2023) mengusulkan *GPTQ*, yaitu metode kuantisasi bobot per lapisan (*one-shot layer-wise*) yang berbasis informasi orde kedua. Metode ini mampu menekan ukuran bobot *Large Language Model* (LLM) hingga 3 sampai 4 bit dengan

penurunan nilai *perplexity* yang minimal. Sementara itu, (Lin et al., 2026) memperkenalkan AWQ (*Activation-aware Weight Quantization*) yang mengidentifikasi dan mempertahankan 1% bobot paling penting (*salient weights*) berdasarkan distribusi aktivasinya. Pendekatan ini sangat sesuai untuk eksekusi langsung di perangkat (*on-device*) karena tidak memerlukan operasi presisi campuran yang dapat memperlambat waktu respons pada perangkat seluler.

Akan tetapi, peningkatan efisiensi memori tersebut sering kali disertai penurunan kualitas model. Studi ekstensif oleh (Gong et al., 2024) menggunakan pendekatan perturbasi menunjukkan bahwa penyederhanaan bobot menjadi bilangan bulat terkompresi sering kali menghilangkan nilai-nilai ekstrem yang penting (*outliers*). Hal ini berdampak pada menurunnya kemampuan pemahaman bahasa pada AI, yang ditandai dengan memburuknya nilai *perplexity*.

Terlepas dari risiko penurunan akurasi penalaran tersebut, urgensi untuk menjalankan AI generatif secara mandiri di perangkat pengguna semakin tinggi. Hal ini didorong oleh masalah keterlambatan respons (latensi) dan keterbatasan pita jaringan (*bandwidth*) pada layanan komputasi awan di kawasan urban (Sevim et al., 2024). Dari perspektif privasi data, (Zhan et al., 2025) menegaskan bahwa instansi yang mengelola data rahasia sebaiknya tidak menyerahkan proses pengolahan teks kepada layanan pihak ketiga (*Cloud API*). Dalam konteks layanan publik, kebutuhan akan asisten cerdas yang responsif sangatlah tinggi. Sebagai contoh, implementasi *chatbot* di RS Brawijaya (Fadillah Ahmad et al., 2024) saat ini masih bergantung pada layanan *Cloud API* komersial. Kondisi inilah yang memperkuat urgensi pengembangan alternatif pemrosesan lokal pada perangkat pengguna (*edge device*).

Untuk menyusun parameter pengujian *Small Language Model* (SLM) secara objektif, penelitian ini merujuk pada metodologi evaluasi dari (Jin et al., 2024) dan kerangka pengujian latensi dari (Zhang et al., 2024). Kedua studi tersebut menegaskan bahwa kualitas LLM tidak cukup dinilai dari tingkat akurasi saja, tetapi harus dipadukan dengan metrik *perplexity* dan kecepatan pembuatan teks (*Token Per Second* atau TPS). Sebagai landasan utama untuk studi terapan ini, riset observasional dari (Ray & Pradhan, 2026) digunakan sebagai acuan validasi. Studi tersebut membuktikan bahwa pengujian format PTQ GGUF menggunakan bahasa pemrograman C++ (*llama.cpp*) pada perangkat Raspberry Pi berkapasitas RAM 8 GB mampu menghasilkan nilai TPS yang stabil. Bukti empiris ini dipadukan dengan keunggulan arsitektur model lokal seperti Qwen (Nurohim et al., 2025) dan LLaMA (Touvron et al., 2023). Hal ini mengonfirmasi bahwa eksekusi AI secara luring (*offline*) pada sistem operasi Android dengan spesifikasi RAM yang setara adalah suatu potensi terapan yang sangat krusial untuk dieksplorasi.

Pemaparan ringkas mengenai tujuh penelitian pilar paling representatif yang mendasari riset ini disajikan pada Tabel II.1 sebagai berikut:

Tabel II. 1
Matriks Perbandingan Penelitian Terkait

Penulis & Judul Penelitian	Tujuan Penelitian	Metode Penelitian	Hasil Penelitian
(Dettmers et al., 2023), <i>QLoRA: Efficient Finetuning of Quantized LLMs</i>	Mengembangkan teknik <i>fine-tuning</i> hemat memori bagi LLM agar dapat dilatih pada satu GPU melalui kompresi presisi rendah.	Kompresi bobot ke presisi 4-bit (<i>NormalFloat-4</i>), <i>double quantization</i> , dan <i>paged optimizer</i> untuk LoRA.	Pemadatan 4-bit (NF4) terbukti mencegah <i>Out of Memory</i> dan mempertahankan akurasi setara FP16 pada <i>benchmark</i> pasca- <i>finetuning</i> .

Penulis & Judul Penelitian	Tujuan Penelitian	Metode Penelitian	Hasil Penelitian
(Frantar et al., 2023), <i>GPTQ: Accurate Post-Training Quantization for Generative Pre-Trained Transformers</i>	Mengusulkan metode PTQ <i>one-shot</i> yang akurat untuk model generatif berskala besar.	Pendekatan <i>layer-wise weight quantization</i> berbasis pendekatan <i>second-order</i> (OBQ) yang diadaptasi untuk LLM.	GPTQ berhasil mengompresi LLM hingga 3–4-bit dengan degradasi <i>perplexity</i> minim dan menjadi salah satu fondasi PTQ untuk LLM.
(Lin et al., 2026), <i>AWQ: Activation-Aware Weight Quantization for On-Device LLM Compression and Acceleration</i>	Menyusun metode kuantisasi PTQ yang ramah implementasi <i>on-device</i> dengan perlindungan terhadap bobot kritis.	Identifikasi 1% bobot paling penting berdasarkan distribusi aktivasi, lalu menerapkan <i>per-channel scaling</i> untuk mempertahankannya.	Perlindungan <i>salient weights</i> menurunkan <i>quantization error</i> dan menjadi rujukan penting untuk inferensi <i>on-device</i>
(Jin et al., 2024), <i>A Comprehensive Evaluation of Quantization Strategies for Large Language Models</i>	Menyusun kerangka evaluasi holistik untuk membandingkan berbagai strategi kuantisasi pada LLM.	Evaluasi tiga dimensi (kapasitas, kelugasan, dan efisiensi) terhadap beragam metode kuantisasi pada banyak model LLM.	Studi tersebut menekankan penggunaan <i>Perplexity</i> dan beban RAM sebagai indikator kelayakan kuantisasi. Kerangka evaluasi ini diadopsi sebagai acuan metodologi dalam skripsi.
(Ray & Pradhan, 2026), <i>Performance Analysis of Localised Large Language Models in Resource-Constrained Edge for Python and Rust APIs</i>	Mengevaluasi kelayakan eksekusi LLM terlokalisasi pada perangkat <i>edge</i> dengan sumber daya terbatas.	Eksperimen empiris LLM format GGUF melalui <i>llama.cpp</i> pada Raspberry Pi RAM 8 GB.	LLM GGUF dapat dieksekusi stabil pada arsitektur ARM dengan RAM 8 GB. Temuan ini menjadi dasar replikasi pengujian pada SoC ARM Android melalui <i>Termux</i>
(Laskaridis et al., 2024), <i>MELTING point: Mobile Evaluation of</i>	Mengevaluasi kelayakan dan keterbatasan eksekusi LLM	Studi empiris pertama yang membandingkan beberapa LLM (1–	Mengonfirmasi bahwa inferensi LLM bersifat <i>memory-bound</i>

Penulis & Judul Penelitian	Tujuan Penelitian	Metode Penelitian	Hasil Penelitian
<i>Language Transformers</i>	pada <i>smartphone</i> Android dan iOS secara sistematis.	13B) pada perangkat seluler nyata dengan metrik <i>throughput</i> , RAM, dan <i>thermal</i> .	pada <i>smartphone</i> . Kuantisasi menurunkan <i>memory footprint</i> dengan trade-off pada akurasi. Acuan utama klaim <i>memory-bound</i> di skripsi.
(Lu et al., 2025), <i>Demystifying Small Language Models for Edge Deployment</i>	Memetakan lanskap SLM yang layak diimplementasikan pada perangkat <i>edge</i> termasuk <i>smartphone</i> .	Survei komprehensif >60 SLM mencakup arsitektur, ukuran, <i>benchmark</i> akurasi, dan profil inferensi pada perangkat <i>edge</i> .	SLM <i>state-of-the-art</i> dapat mengungguli model 7B pada beberapa tugas umum. Temuan ini mendukung pemilihan LFM 2.5 (1,2B) dan Qwen 3.5 (2B) pada penelitian ini.

2.3. Tinjauan Objek Penelitian

Karena penelitian ini merupakan studi eksperimental berbasis laboratorium, subbab ini memaparkan objek penelitian. Objek penelitian merupakan satuan analisis (*unit of analysis*) yang diukur dalam eksperimen kompresi dan pengujian performa *on-device*. Seluruh rangkaian pengujian, mulai dari persiapan berkas model hingga eksekusi pengujian performa (*benchmark*), dilakukan secara mandiri menggunakan perangkat komputasi pribadi. Objek penelitian ini mencakup dua kategori mendasar yang saling melengkapi. Kategori pertama adalah ekosistem perangkat keras dan perangkat lunak Android yang bertindak sebagai sistem utama target penerapan (*target deployment*). Selanjutnya, kategori kedua adalah arsitektur *Small Language Model* (SLM) yang menjadi sasaran pengujian kompresi *Post-Training Quantization* (PTQ). Pembagian menjadi dua komponen ini bertujuan untuk memfasilitasi analisis titik keseimbangan (*trade-off*) antara aspek efisiensi perangkat keras, yang meliputi

penggunaan RAM, beban CPU, dan nilai TPS, dengan aspek kualitas perangkat lunak, yang mencakup tingkat *perplexity* dan akurasi *benchmark*. Pendekatan evaluasi komprehensif ini dilakukan sebagaimana dianjurkan dalam kerangka evaluasi tiga dimensi yang diusulkan oleh (Jin et al., 2024).

2.3.1. Infrastruktur Pengujian: Tecno Pova 5 dan Termux

Objek operasional dibatasi pada satu *smartphone* kelas reguler, yaitu Tecno Pova 5. Perangkat ini ditenagai SoC MediaTek Helio G99 berbasis ARM, dengan RAM sistem 8 GB. Mengingat status gawai *non-rooted* (bawaan pabrik tanpa akses *root*), lingkungan pengujian virtual dibangun di atas Termux, sebuah terminal emulator dan lingkungan paket perangkat lunak berbasis Linux untuk Android yang berjalan tanpa akses *root*. Melalui kompilasi mandiri di Termux inilah *llama.cpp* dikompilasi ulang di *Termux* dalam konfigurasi C/C++ agar instruksi CPU ARM dapat dieksekusi lebih optimal.

2.3.2. Objek Arsitektur SLM: LFM 2.5 (1,2B) dan Qwen 3.5 (2B)

Riset komparatif ini menggunakan dua Model Bahasa Skala Kecil (*Small Language Model* atau SLM). Model pertama adalah LFM 2.5 yang memiliki ukuran 1,2 miliar parameter. Berdasarkan laporan teknis resmi (Amini et al., 2025), arsitektur LFM 2 dirancang menggunakan struktur dasar campuran (*hybrid backbone*). Struktur ini memadukan teknik penyaringan data berjarak pendek yang terkontrol (*gated short convolutions*) dengan mekanisme pemusatan informasi secara berkelompok (*grouped query attention*). Model ini secara khusus dioptimalkan untuk pemrosesan langsung di perangkat pengguna (*edge inference*), dengan kecepatan pembacaan instruksi awal (*prefill*) dan kecepatan pembuatan teks jawaban (*decode*) lebih cepat dibandingkan model setara yang dijalankan pada prosesor biasa (CPU). Struktur internalnya yang

efisien menjadikan model ini sangat berpotensi untuk dipadatkan ukurannya secara bertahap pada rentang 3-bit hingga 5-bit guna menekan penggunaan RAM. Versi asli dengan presisi standar (FP16) akan dibandingkan dengan tiga varian hasil pemadatan (*K-Quants*), yaitu Q5_K_M, Q4_K_M, dan Q3_K_M. Model kedua adalah Qwen 3.5 dengan ukuran 2 miliar parameter. Keluarga model Qwen secara khusus memperkenalkan dua sistem dalam satu model, yaitu mode berpikir (*thinking mode*) untuk menyelesaikan masalah yang memerlukan penalaran bertahap (*multi-step reasoning*) dan mode standar (*non-thinking mode*) untuk menghasilkan respons yang cepat (Team, 2025). Pemilihan dua arsitektur berbeda ini bertujuan untuk memvalidasi generalisasi temuan lintas-arsitektur, sekaligus menguji batas toleransi memori sistem ketika perangkat Android dengan RAM 8 GB dan *chipset* Helio G99 menjalankan model di atas 2 miliar parameter pascakuantisasi.

