Implementasi Dan Optimasi Sistem Parkir Terintegrasi Menggunakan Yolo Dan Cnn Pada Lingkungan *Edge Computing* Berbasis Raspberry Pi

Muhammad Caesar Abhista Raya Bima Saputra[1*], Imadudin Harjanto[2], Ganjar Winasis[3]

[123]*Universitas PGRI Semarang*

Jl.Sidodadi Timur No.24-Dr Cipto, Semarang, Jawa Tengah 50232, Indonesia

[*] *Email: muhammadcaesar68@gmail.com*

Abstrak*— Pertumbuhan jumlah kendaraan di lingkungan universitas seperti Universitas PGRI Semarang (UPGRIS) memerlukan sistem parkir otomatis yang efisien. Implementasi *deep learning* pada perangkat *edge* seperti Raspberry Pi menghadapi kendala latensi dan keterbatasan sumber daya. Penelitian ini bertujuan mengoptimalkan model YOLOv8n untuk deteksi plat nomor dan MobileNetV2 untuk deteksi lahan parkir melalui dua skema optimasi: konversi TensorFlow Lite (*float32*) dan gabungan *structured pruning* 20% dengan *Post-Training Quantization* (INT8). Hasil penelitian menunjukkan bahwa optimasi gabungan (Pruning+INT8) berhasil memangkas ukuran model YOLOv8n dari 11,71 MB menjadi 3,19 MB dan MobileNetV2 dari 10,84 MB menjadi 2,74 MB. Pada Raspberry Pi 4B, model YOLOv8n INT8 mencapai kecepatan stabil 2,72 FPS dengan efisiensi RAM sebesar 27,4%. Sementara pada MobileNetV2, optimasi mempercepat waktu inferensi hingga empat kali lipat dan menekan konsumsi RAM sebesar 35,6% (dari 940 MB ke 605 MB). Meskipun terjadi kompromi akurasi sebesar 2%-8% pada kondisi cahaya redup, model tetap mempertahankan performa deteksi yang reliabel. Penelitian ini menyimpulkan bahwa skema gabungan *pruning* dan kuantisasi INT8 merupakan solusi paling efisien untuk menyeimbangkan kecepatan inferensi dan keterbatasan memori pada perangkat *edge* guna mendukung sistem parkir cerdas secara *real-time*.

Kata kunci：Raspberry Pi; *Edge Computing*; YOLOv8; MobileNetV2; *Pruning*; *Quantization*; TensorFlow Lite.

I. PENDAHULUAN

Pertumbuhan jumlah kendaraan yang pesat di lingkungan universitas, seperti di Universitas PGRI Semarang (UPGRIS), menuntut adanya sistem tata kelola parkir yang lebih efisien, terotomatisasi, dan hemat sumber daya. Sistem parkir konvensional yang mengandalkan tenaga manusia memiliki berbagai kelemahan, mulai dari waktu tunggu yang lama hingga kerentanan terhadap kesalahan pencatatan data. Integrasi teknologi pengolahan citra berbasis kecerdasan buatan (Artificial Intelligence) dan perangkat Internet of Things (IoT) menawarkan solusi sistem parkir modern yang bekerja secara real-time. Komponen utama dalam sistem ini mencakup algoritma YOLO untuk identifikasi plat nomor dan Convolutional Neural Network (CNN) untuk mendeteksi ketersediaan slot parkir secara otomatis.

Implementasi model deep learning yang kompleks pada perangkat dengan sumber daya terbatas (edge computing) seperti Raspberry Pi menghadapi tantangan besar terkait tuntutan komputasi yang tinggi. Model baseline berisiko mengalami latensi tinggi dan konsumsi daya yang tidak efisien, sehingga menghambat tercapainya kinerja deteksi objek secara real-time. Oleh karena itu, penelitian ini berfokus pada optimasi kinerja model YOLO dan CNN pada Raspberry Pi 4B melalui penerapan metode pruning untuk menghapus bobot yang tidak signifikan, quantization untuk mereduksi ukuran model, serta konversi ke format TensorFlow Lite.

Tujuan utama dari penelitian ini adalah mengaplikasikan model YOLO dan CNN pada perangkat Raspberry Pi serta membandingkan performa antara model baseline dengan model yang telah teroptimasi. Evaluasi dilakukan dengan membandingkan parameter waktu inferensi, akurasi, dan ukuran model terhadap perangkat komputer konvensional sebagai pembanding kinerja. Penelitian ini diharapkan dapat memberikan kontribusi literatur terkait metode optimasi deep learning pada perangkat edge serta menunjukkan efektivitas sistem deteksi lahan parkir dan plat nomor dalam kondisi keterbatasan perangkat keras.

II. STUDI PUSTAKA

Penelitian mengenai sistem parkir cerdas berbasis visi komputer telah banyak dikembangkan untuk meningkatkan efisiensi manajemen kendaraan. Purwalaksana et al. [1]dan Pradhan et al. [2]merancang sistem parkir berbasis Raspberry Pi dan IoT yang terintegrasi dengan Optical Character Recognition (OCR). Meskipun sistem tersebut terbukti fungsional, keduanya belum menekankan pada aspek optimasi model deep learning untuk mengatasi keterbatasan komputasi pada perangkat keras. Dari sisi algoritma, Satya et al. [3] membuktikan bahwa YOLOv8 memiliki akurasi tinggi (mAP 99,3%) untuk deteksi plat nomor, sementara Suhartono et al. [4] memanfaatkan CNN untuk deteksi lahan parkir. Namun, implementasi model-model kompleks ini pada perangkat edge seringkali terkendala oleh latensi tinggi jika tidak dilakukan penyesuaian arsitektur.

Untuk mengatasi kendala sumber daya pada perangkat edge, teknik optimasi model menjadi solusi krusial. Ameen et al. [5] dan Mohandas et al. [6] menunjukkan bahwa

penerapan pruning dan kuantisasi pada Raspberry Pi mampu mereduksi ukuran model secara signifikan dan mempercepat waktu inferensi tanpa penurunan akurasi yang drastis. Tank et al.[7] juga menegaskan bahwa penggunaan arsitektur model yang ringan sangat menentukan keberhasilan diagnosis berbasis AI pada perangkat embedded. Berdasarkan kajian tersebut, penelitian ini mengintegrasikan metode optimasi pada model YOLO dan CNN untuk menjembatani celah antara kompleksitas model dengan keterbatasan perangkat keras.

Metode optimasi yang diterapkan dalam penelitian ini meliputi Structured Pruning, Kuantisasi, dan Konversi Model. (1) Pruning adalah teknik mereduksi parameter model dengan menghapus filter atau neuron yang memiliki kontribusi minim terhadap hasil prediksi. Penelitian ini menerapkan structured pruning untuk menghapus channel konvolusi secara sistematis, sehingga struktur jaringan menjadi lebih ringkas [8]. (2) Quantization (Kuantisasi) merupakan proses pemetaan nilai kontinu ke dalam nilai diskrit. Dalam konteks ini, digunakan Post-Training Quantization (PTQ) untuk mengubah representasi bobot dan aktivasi dari presisi tinggi (float32) menjadi presisi rendah (int8). Teknik ini bertujuan mereduksi ukuran model hingga empat kali lipat dan mempercepat inferensi aritmatika pada CP[9]. (3) Konversi Model dilakukan menggunakan kerangka kerja TensorFlow Lite (TFLite). Format ini dirancang khusus untuk lingkungan edge computing, memungkinkan model hasil optimasi dijalankan melalui interpreter ringan tanpa memerlukan akselerasi GPU eksternal [10].

III. METODE/DESAIN

A. Pendekatan Penelitian

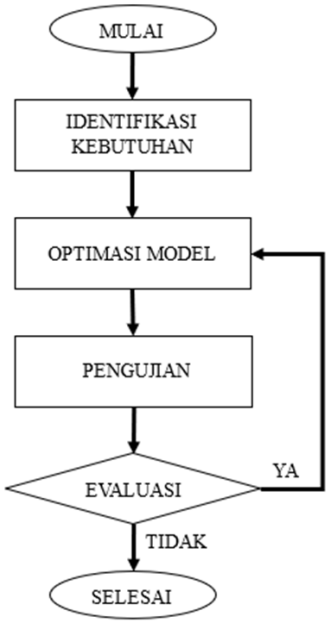
Penelitian ini menerapkan pendekatan kuantitatif yang mengombinasikan metode eksperimental dan komparatif untuk mengevaluasi efektivitas optimasi model. Metode eksperimental dilakukan dengan memberikan perlakuan khusus pada model melalui teknik pemangkasan parameter dan penurunan presisi data. Pendekatan komparatif digunakan untuk membedakan performa model secara objektif pada dua platform komputasi yang kontras, yakni laptop sebagai representasi komputasi awan/lokal berperforma tinggi, dan Raspberry Pi 4B sebagai representasi perangkat edge dengan sumber daya terbatas. Parameter numerik yang diukur meliputi waktu inferensi, penggunaan memori, dan akurasi, guna menarik kesimpulan yang valid secara statistik.

B. Alat dan Bahan

Perangkat keras dan lunak yang digunakan dalam penelitian ini meliputi:

- Dataset PKLot dan LPR, sebagai sumber data pelatihan dan optimasi model.
- Model *Baseline* MobileNetV2 dan YOLOv8n, sebagai model awal atau acuan utama model sebelum optimasi.
- Laptop HP 245 G8, sebagai perangkat utama proses optimasi dan pengembangan kode program
- Raspberry Pi 4B, sebagai perangkat utama target pengujian model baseline dan model teroptimasi
- Webcam Jete W9 dan adaptor daya USB Type-C 5V 3A, sebagai perangkat pendukung
- Visual Studio Code, Google Colab, Python, sebagai perangkat lunak pendukung proses optimasi dan pengembanga kode program
- RealVNC, sebagai perangkat pengoperasian Raspberry Pi secara remote.

C. Tahapan Optimasi Model



Gambar 1 Diagram air tahapan penelitian

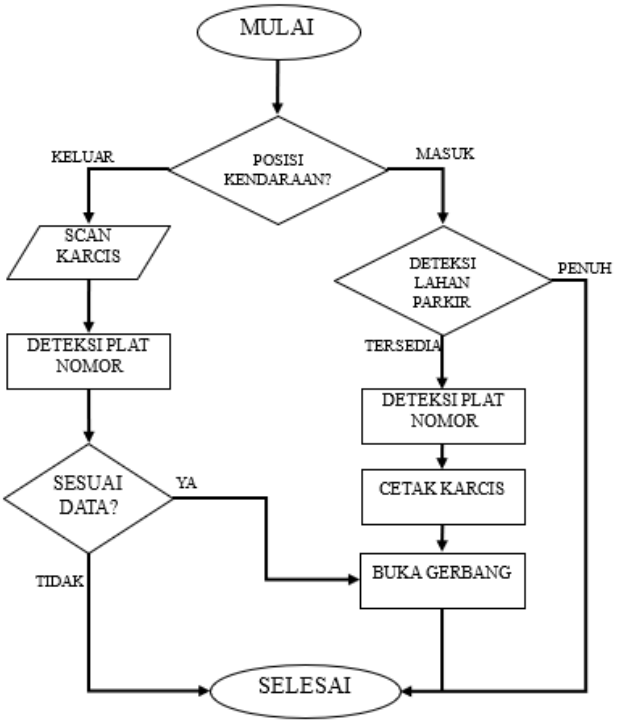
Tahapan penelitian dimulai dengan fase identifikasi kebutuhan yang mencakup pengumpulan informasi, analisis kebutuhan sistem, serta persiapan data dan model baseline. Peneliti melakukan studi literatur secara mendalam guna memperoleh landasan teoretis terkait metode deteksi objek, optimasi deep learning, serta implementasi teknis pada perangkat Raspberry Pi. Pada fase ini, dataset publik PKLot dan LPR Roboflow dipilih dan disiapkan bersama dengan data tambahan berupa foto uji lapangan manual. Model YOLOv8n ditetapkan sebagai baseline untuk tugas deteksi plat nomor kendaraan, sedangkan MobileNetV2 CNN dialokasikan untuk tugas klasifikasi ketersediaan lahan parkir.

Selanjutnya, proses optimasi model dilaksanakan menggunakan platform Google Colab untuk memanfaatkan akselerasi GPU cloud dalam mentransformasikan model baseline menjadi model yang lebih efisien. Optimasi dijalankan melalui dua skema pendekatan utama: Skema I melibatkan konversi langsung ke format TensorFlow Lite (FP32) untuk menyederhanakan struktur graf model agar kompatibel dengan perangkat edge; sedangkan Skema II menerapkan optimasi gabungan yang lebih kompleks. Skema II diawali dengan structured pruning untuk memangkas 20% parameter redundan pada struktur jaringan, yang kemudian diikuti dengan Post-Training Full Integer Quantization (INT8) untuk mengubah representasi bobot dari presisi floating point menjadi integer 8-bit. Langkah ini bertujuan untuk mereduksi ukuran model secara drastis serta menurunkan latensi inferensi pada CPU Raspberry Pi.

Setelah model teroptimasi berhasil dibuat, dilakukan tahap pengujian dan evaluasi komparatif pada perangkat laptop dan Raspberry Pi 4B. Pengujian dilakukan secara

konsisten menggunakan dataset uji yang sama untuk mengukur parameter performa yang meliputi waktu inferensi (latency), ukuran file model, tingkat akurasi deteksi, serta penggunaan sumber daya CPU dan RAM. Seluruh data hasil pengujian kemudian dievaluasi secara kritis untuk menganalisis rasio peningkatan efisiensi terhadap stabilitas akurasi. Jika performa model telah memenuhi kriteria kinerja real-time dan menunjukkan keunggulan dibandingkan model baseline, maka penelitian dinyatakan selesai dan model final direkomendasikan untuk diintegrasikan ke dalam sistem manajemen parkir kampus yang otonom.

D. Alur Kerja Sistem



Gambar 2. Flowchart system

Alur kerja sistem parkir otomatis yang dirancang dalam penelitian ini secara komprehensif diilustrasikan melalui Gambar 2 Proses operasional dimulai dengan identifikasi posisi kendaraan melalui sensor input untuk menentukan apakah kendaraan berada di pintu masuk atau pintu keluar. Pada skenario kendaraan masuk, sistem terlebih dahulu mengeksekusi model klasifikasi lahan parkir (MobileNetV2) untuk memverifikasi ketersediaan slot. Apabila lahan parkir terdeteksi penuh, sistem secara otomatis menghentikan proses deteksi. Namun, jika lahan tersedia, sistem melanjutkan ke tahap deteksi plat nomor menggunakan algoritma YOLOv8n untuk mencatat identitas kendaraan. Data hasil deteksi tersebut kemudian diintegrasikan ke dalam karcis berbasis *barcode* yang dicetak secara otomatis bersamaan dengan pembukaan gerbang masuk selama sepuluh detik.

Sebaliknya, pada alur kendaraan keluar, sistem memulai proses dengan memindai *barcode* pada karcis untuk memanggil data kendaraan dari basis data. Secara bersamaan, kamera pada pintu keluar menangkap citra plat nomor untuk dilakukan verifikasi ulang menggunakan model deteksi objek. Sistem akan melakukan pencocokan data antara nomor plat yang terdeteksi saat keluar dengan data yang tersimpan saat masuk. Jika data dinyatakan sesuai (*match*), gerbang keluar akan terbuka selama sepuluh detik untuk memberikan akses jalan. Namun, apabila ditemukan ketidaksesuaian data, sistem akan memberikan peringatan dan kembali ke tahap awal pemindaian untuk memastikan keamanan kendaraan.

Definisikan singkatan dan akronim pada saat pertama sekali digunakan pada teks, walaupun singkatan dan akronim tersebut telah didefinisikan di abstrak.

Singkatan seperti IEEE, SI, MKS, CGS, sc, dc, dan rms tidak perlu didefinisikan kembali. Jangan gunakan singkatan pada judul, kecuali tidak ada pilihan lain.

E. Persamaan

Nomori *persamaan* secara berurutan dengan nomor persamaan dalam tanda kurung rata dengan margin kanan, seperti pada (1). Supaya representasi persamaan lebih kompak, gunakan fungsi exp function atau eksponen yang sesuai. Untuk kuantitas dan variable, cetak miring (italic) symbol Roman, bukan symbol Greek. Gunakan en dash (–) dan bukan hypen untuk tanda minus. Gunakan tanda kurung untuk memperjelas bagian penyebut pada bilangan pecahan. Pisahkan persamaan dengan koma jika persamaan tersebut merupakan bagian dari kalimat seperti contoh berikut

$$\frac{d[F_1]}{d\omega_2} = SAm_2 \cos\omega, \ \frac{d[F_1]}{d\omega_3} = SAm_2 \cos\omega \qquad (1)$$

Simbol pada persamaan seharusnya didefinisikan terlebih dahulu sebelum persamaannya muncul atau segera sesudahnya. Gunakan "(1)," bukan "Pers. (1)" atau "persamaan (1)," kecuali pada permulaan kalimat: "Persamaan (1) adalah ..."

IV. Hasil dan Pembahasan

A. Implementasi dan optimasi model

Pada penelitian ini dilakukan optimasi model deep learning yang digunakan pada sistem parkir otomatis, yaitu model YOLOv8n untuk deteksi plat nomor kendaraan dan MobileNetV2 untuk deteksi ketersediaan lahan parkir. Optimasi diperlukan untuk menyesuaikan keterbatasan sumber daya perangkat edge seperti Raspberry Pi, sehingga model yang diimplementasikan mampu mencapai keseimbangan antara efisiensi komputasi, kecepatan inferensi, dan akurasi.

Strategi optimasi dilakukan melalui dua skenario utama, yaitu konversi model ke format TensorFlow Lite (FP32) dan optimasi lanjutan menggunakan kombinasi pruning dan quantization INT8. Seluruh proses optimasi dilakukan pada lingkungan Google Colab, sedangkan Raspberry Pi digunakan secara khusus untuk tahap pengujian performa nyata guna memastikan kesiapan model dalam implementasi edge computing.

B. Hasil pengujian model MobileNetV2

Berdasarkan data pada Tabel 1 dan Tabel 2, dilakukan analisis komparatif performa berdasarkan variasi kondisi pencahayaan. Pada kondisi siang hari dengan cahaya optimal, laptop menunjukkan akurasi sempurna sebesar

100% dengan waktu inferensi tercepat pada varian TFLite FP32 (5,226 ms). Sementara itu, Raspberry Pi mampu mempertahankan akurasi stabil di angka 97%, meskipun waktu inferensinya jauh lebih lambat dengan varian INT8 sebagai yang tercepat (18,652 ms). Hal ini mengonfirmasi adanya keterbatasan *hardware* pada perangkat *edge* dalam memproses beban komputasi visi komputer.

Tabel 1. Hasil pengujian model MobileNetV2 di Laptop

Performa di Laptop

Kondisi Cahaya	Varian model	Waktu Inferensi (ms)	Konsumsi RAM (mb)	Konsumsi CPU (%)	Size (mb)	AKURASI
SIANG	Baseline	42,370	404,050	2,631	10,84%	100%
	TFLine FP32	5,226	349,652	2,442	9,08	100%
	TFLite INT8	7,754	351,421	5,978	2,743	100%
SORE	Baseline	31,987	405,027	1,816	10,847	93%
	TFLine FP32	6,362	350,276	1,328	9,08	92%
	TFLite INT8	7,734	352,056	8,504	2,743	92%
MALAM	Baseline	31,660	404,315	1,762	10,847	94%
	TFLine FP32	5,649	348,915	1,701	9,08	94%
	TFLite INT8	8,032	350,385	6,058	2,743	92%

Tabel 2. Hasil penguji model mobileNetV2 di Raspberry Pi

Performa di Raspberry Pi

Kondisi Cahaya	Varian model	Waktu Inferensi (ms)	Konsumsi RAM (mb)	Konsumsi CPU (%)	Size (mb)	AKURASI
SIANG	Baseline	68,633	946,101	70,543	10,847	97%
	TFLine FP32	36,114	626,375	71,165	9,08	97%
	TFLite INT8	18,652	606,601	71,230	2,743	97%
SORE	Baseline	65,384	943,494	73,379	10,847	91%
	TFLine FP32	36,632	626,591	72,952	9,08	91%
	TFLite INT8	17,692	606,883	74,047	2,743	89%
MALAM	Baseline	67,856	891,699	71,049	10,847	94%
	TFLine FP32	36,011	634,122	70,612	9,08	95%
	TFLite INT8	18,062	605,394	72,522	2,743	92%

Memasuki kondisi sore hari dengan cahaya redup, terjadi penurunan akurasi pada kedua perangkat. Pada Raspberry Pi, penurunan paling signifikan terlihat pada varian INT8 (89%), yang menunjukkan bahwa kondisi cahaya rendah menjadi tantangan teknis terbesar bagi model yang telah terkuantisasi. Namun, pada kondisi malam hari dengan bantuan cahaya lampu, model menunjukkan pemulihan performa yang cukup baik dengan akurasi pada Raspberry Pi berkisar antara 92% hingga 95%.

Dari sisi penggunaan sumber daya, pengujian menunjukkan perbedaan beban kerja yang kontras. Penggunaan CPU pada laptop sangat rendah (1,3% - 8,5%), mengindikasikan bahwa MobileNetV2 bukan merupakan tugas yang berat bagi prosesor laptop. Sebaliknya, Raspberry Pi bekerja secara intensif dengan utilisasi CPU mencapai 74,1%. Efisiensi penggunaan RAM menjadi poin krusial pada perangkat *edge*, di mana model INT8 berhasil menekan penggunaan memori secara signifikan menjadi 605 MB dibandingkan varian *baseline* yang mencapai 940 MB.

Secara keseluruhan, model TFLite FP32 terbukti sebagai pilihan paling efektif untuk penggunaan pada laptop karena menghasilkan akurasi maksimal tanpa kendala latensi. Untuk implementasi pada Raspberry Pi, model TFLite INT8 menjadi solusi paling ideal karena mampu mempercepat waktu proses hingga empat kali lipat serta memangkas ukuran model dan penggunaan RAM secara signifikan, meskipun terdapat sedikit kompromi pada margin akurasi di kondisi cahaya tertentu.

C. *Hasil pengujian model YOLOv8n*

Tabel 3. Hasil pengujian model YOLOv8n di Laptop

LAPTOP						
Model	Avg Latency (ms)	Avg FPS	Avg CPU (%)	Avg RAM (MB)	Akurasi (%)	Size (mb)
Baseline .pt	354,4909063	2,820946834	60,93%	536,697	91%	5,69
Fp32 .tflite	287,4247678	3,479171289	94,93%	764,936	91%	11,712
INT8 .tflite	28088,4035	0,035601881	93,76%	183,826	83%	3,193

Tabel 4. Hasil pengujian model YOLOv8n di Raspberry Pi

LAPTOP						
Model	Avg Latency (ms)	Avg FPS	Avg CPU (%)	Avg RAM (MB)	Akurasi (%)	Size (mb)
Baseline .pt	354,4909063	2,820946834	60,93%	536,697	91%	5,69
Fp32 .tflite	287,4247678	3,479171289	94,93%	764,936	91%	11,712
INT8 .tflite	28088,4035	0,035601881	93,76%	183,826	83%	3,193

Berdasarkan data pada Tabel 3 dan 4, dilakukan analisis mendalam mengenai pengaruh optimasi terhadap kinerja sistem pada dua platform yang berbeda. Dari aspek kecepatan (*latency* dan FPS), laptop menunjukkan performa yang cenderung stabil pada model *baseline*, namun model INT8 .tflite justru mengalami penurunan kecepatan yang sangat drastis hingga 0,035 FPS. Hal ini mengindikasikan bahwa arsitektur prosesor laptop tertentu belum teroptimasi secara *native* untuk instruksi kuantisasi tersebut. Sebaliknya, pada Raspberry Pi, model INT8 menunjukkan hasil yang impresif dengan kecepatan 2,72 FPS, yang hampir identik dengan model *baseline*. Optimasi kuantisasi terbukti berhasil memulihkan kecepatan inferensi yang sempat melambat signifikan pada varian FP32 di perangkat *edge*.

Ditinjau dari efisiensi sumber daya, penggunaan CPU pada Raspberry Pi tergolong sangat tinggi, mencapai puncaknya pada model FP32 (334,53%) yang menunjukkan beban kerja prosesor yang berat. Namun, dari sisi penggunaan RAM, model INT8 berhasil memberikan efisiensi yang signifikan dengan menekan konsumsi memori kembali ke angka 795,76 MB pada Raspberry Pi, serta menjadi varian paling hemat pada laptop dengan penggunaan hanya 183,82 MB.

Terkait akurasi dan dimensi model, proses optimasi berhasil memangkas ukuran file secara drastis dari 11,71 MB menjadi 3,19 MB. Walaupun pada laptop model INT8 mengalami sedikit penurunan akurasi menjadi 83%, pada pengujian di Raspberry Pi, seluruh varian model termasuk

INT8 berhasil mempertahankan akurasi 100% pada sampel uji yang digunakan. Secara keseluruhan, model INT8 .tflite terbukti sebagai varian paling optimal untuk diimplementasikan pada Raspberry Pi karena mampu memberikan keseimbangan terbaik antara kecepatan, efisiensi memori, dan penghematan ruang penyimpanan tanpa mengorbankan akurasi deteksi.

D. Pembahasan

Berdasarkan seluruh rangkaian pengujian yang telah dilakukan, terdapat beberapa temuan krusial mengenai implementasi deep learning pada lingkungan edge computing. Pertama, hasil optimasi menunjukkan adanya trade-off yang berbeda antara arsitektur model YOLOv8n dan MobileNetV2. Pada YOLOv8n, penggunaan teknik quantization INT8 berhasil mereduksi ukuran model hingga 46% tanpa mengorbankan akurasi pada perangkat Raspberry Pi, yang secara konsisten tetap berada di angka 100%. Sebaliknya, pada MobileNetV2, meskipun reduksi ukuran model jauh lebih drastis (74,7%), terdapat margin penurunan akurasi sebesar 1% hingga 5% tergantung pada kondisi pencahayaan. Hal ini mengindikasikan bahwa model klasifikasi (CNN) cenderung lebih sensitif terhadap kehilangan presisi data dibandingkan model deteksi objek (anchor-based) saat melewati proses kuantisasi.

Kedua, fenomena perlambatan drastis model INT8 pada perangkat laptop (x86) dibandingkan akselerasi signifikan pada Raspberry Pi (ARM) menegaskan bahwa optimasi TensorFlow Lite sangat bergantung pada dukungan instruksi hardware yang spesifik. Prosesor ARM pada Raspberry Pi 4B terbukti lebih efisien dalam mengeksekusi aritmatika integer, yang memungkinkan model INT8 memulihkan kecepatan inferensi yang sempat melambat pada varian FP32. Temuan ini sangat krusial bagi pengembangan sistem embedded di masa depan, di mana pemilihan format model harus diselaraskan dengan arsitektur komputasi perangkat target, bukan sekadar mengejar ukuran file yang kecil.

Terakhir, dari aspek keberlanjutan operasional, efisiensi penggunaan RAM yang mencapai 35,6% pada MobileNetV2 memberikan ruang bagi sistem untuk menjalankan tugas tambahan seperti manajemen basis data atau komunikasi jaringan secara simultan tanpa risiko system crash akibat out-of-memory. Meskipun kondisi cahaya redup masih menjadi tantangan bagi akurasi model terkuantisasi, secara keseluruhan kombinasi strategi pruning dan quantization telah memenuhi kriteria kinerja yang diharapkan untuk sistem parkir otomatis yang responsif, hemat energi, dan memiliki throughput yang stabil pada perangkat dengan sumber daya terbatas.

V. KESIMPULAN

Penelitian ini berhasil menunjukkan bahwa optimasi model deep learning melalui teknik gabungan pruning dan kuantisasi INT8 sangat efektif untuk implementasi pada perangkat edge. Berdasarkan analisis komparatif, proses optimasi ini mampu mereduksi ukuran model secara drastis, hingga 46% pada YOLOv8n dan 75% pada MobileNetV2, yang menjadikannya solusi ideal bagi keterbatasan ruang penyimpanan. Meskipun terdapat trade-off berupa penurunan akurasi pada rentang 2%–8%—terutama pada kondisi cahaya redup—sistem tetap mempertahankan nilai recall yang tinggi sehingga fungsionalitas sistem parkir otomatis tetap terjaga. Kecepatan inferensi ditemukan sangat bergantung pada kecocokan format model dengan arsitektur perangkat keras, di mana format INT8 memberikan performa paling stabil di Raspberry Pi (2,72 FPS) dibandingkan format FP32. Selain itu, penggunaan model teroptimasi berhasil menekan konsumsi RAM hingga 35,6%, sebuah aspek krusial untuk menjaga stabilitas sistem agar tidak mengalami crash saat menjalankan proses paralel seperti kontroler Arduino dan unit pencetakan karcis. Namun, tantangan utama masih ditemukan pada akurasi EasyOCR yang menurun hingga 70% pada kondisi real-time akibat motion blur dan penurunan detail bobot pasca-kuantisasi.

Sebagai langkah pengembangan di masa depan, penelitian dapat diarahkan pada eksplorasi arsitektur YOLO versi terbaru yang lebih ringan serta penerapan teknik image preprocessing tambahan, seperti perspective correction dan thresholding, guna meningkatkan akurasi pengenalan karakter pada engine OCR. Metode optimasi lain seperti Knowledge Distillation dan Weight Clustering juga berpotensi untuk dieksplorasi lebih lanjut guna menyeimbangkan presisi dan kecepatan. Terakhir, pengembangan sistem dapat ditingkatkan menuju integrasi penuh pada prototipe sistem parkir yang memiliki mekanisme keamanan berlapis berbasis mikrokontroler untuk menciptakan ekosistem parkir otomatis yang lebih komprehensif dan tangguh.

REFERENSI

[1] A. Z. Purwalaksana, D. Siburian, I. Sianturi, and S. Sianturi, “Camera-Based Parking System Management Using Raspberry Pi,” *Piston: Journal of Technical Engineering*, vol. 5, no. 1, pp. 22–34, 2021, doi: 10.32493/pjte.v5i1.14721.

[2] G. Pradhan, M. R. Prusty, V. S. Negi, and S. Chinara, “Advanced IoT-integrated parking systems with automated license plate recognition and payment management,” *Sci. Rep.*, vol. 15, no. 1, p. 2388, 2025, doi: 10.1038/s41598-025-86441-w.

[3] B. Satya, D. Manongga, Hendry, and A. Aminuddin, “Optimized YOLOv8 for Automatic License Plate Recognition on Resource Constrained Devices,” *Engineering, Technology and Applied Science Research*, vol. 15, no. 2, pp. 21976–21981, Apr. 2025, doi: 10.48084/etasr.9983.

[4] Suhartono, Satria Gunawan Zain, and A. F. Sadaruddin, “Implementation of Convolutional Neural Network Algorithm for Detecting Empty Parking Area Based on Raspberry Pi,” *Jurnal E-Komtek (Elektro-Komputer-Teknik)*, vol. 8, no. 1, pp. 154–167, Jun. 2024, doi: 10.37339/e-komtek.v8i1.1220.

[5] S. Ameen, K. Siriwardana, and T. Theodoridis, “Optimizing Deep Learning Models For Raspberry Pi,” Apr. 2023. Accessed: Sep. 19, 2025.[Online].Available: http://dx.doi.org/10.48550/arXiv.2304.13039

[6] R. Mohandas, M. Bhattacharya, M. Penica, K. Van Camp, and M. J. Hayes, “TensorFlow Enabled Deep Learning Model Optimization for enhanced Realtime Person Detection using Raspberry Pi operating at the Edge,” 2020.

[7] M. B. Tank, M. Patel, and K. Deshmukh, “Evaluating Deep Learning Model Performance on Raspberry Pi 4

for COVID-19 Diagnosis," 2024. [Online]. Available: https://www.jisem-journal.com/

[8] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," Mar. 2017, [Online]. Available: http://arxiv.org/abs/1608.08710

[9] B. Jacob *et al.*, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," Dec. 2017, [Online]. Available: http://arxiv.org/abs/1712.05877

[10] Google, "Convert Models to TensorFlow Lite," Google AI Documentation. Accessed: Nov. 10, 2025. [Online]. Available: https://ai.google.dev/edge/litert/models/convert