BAB I
PENDAHULUAN

1.1 Latar Belakang

Pendapatan Asli Daerah (PAD) merupakan salah satu sumber pendapatan utama bagi pemerintah daerah untuk membiayai pembangunan dan pelayanan publik. Salah satu komponen yang berkontribusi dalam meningkatkan PAD adalah pendapatan dari sektor perparkiran, baik melalui pajak parkir maupun retribusi parkir. Sayangnya, kontribusi pendapatan parkir terhadap PAD di Kota Makassar masih sangat rendah. Berdasarkan data Laporan Keuangan Pemerintah Kota Makassar, realisasi pajak parkir pada tahun 2020 hanya mencapai Rp 9,96 miliar dengan kontribusi 0,92% terhadap total PAD. Pada tahun 2021, realisasi meningkat menjadi Rp 10,92 miliar namun kontribusinya tetap rendah yaitu 0,96%. Pada tahun 2022, penerimaan pajak parkir tercatat sebesar Rp 14,98 miliar dengan kontribusi 1,06% terhadap PAD. Sementara itu, kontribusi bagi hasil Perumda Parkir Makassar Raya terhadap PAD pada periode 2020-2022 bahkan lebih kecil, berkisar antara Rp 443 juta hingga Rp 650 juta dengan kontribusi hanya 0,03%-0,06% terhadap total PAD. Rendahnya realisasi pendapatan parkir ini disebabkan oleh berbagai permasalahan dalam pengelolaan perparkiran, antara lain keberadaan juru parkir liar yang tidak terkelola dengan baik, tarif parkir yang tidak sesuai ketentuan, penyalahgunaan lokasi parkir, serta mekanisme pembayaran yang tidak menggunakan karcis sehingga setoran juru parkir tidak dapat dikendalikan dengan baik. Kondisi ini mengakibatkan terjadinya kebocoran pendapatan dan menyebabkan realisasi retribusi parkir pada tahun 2022 hanya mampu mencapai 31,05% dari target yang ditetapkan (Tim Pengkaji Kajian Akademik Strategi Pengelolaan dan Peningkatan PAD Perumda Parkir Kota Makassar, 2023).

Solusi untuk permasalahan ini adalah dengan mengembangkan sistem parkir cerdas (*smart parking*) berbasis *Artificial Intelligence* (AI), yang memungkinkan teknologi untuk mengelola dan mengawasi proses pembayaran parkir secara otomatis sehingga pengelolaan menjadi lebih efektif, transparan, dan berpotensi mengurangi risiko kebocoran pendapatan oleh pihak yang tidak bertanggung jawab (Fahim et al., 2021).

Salah satu komponen penting dalam sistem *smart parking* adalah teknologi visi Komputer. Namun, implementasi teknologi ini tidak lepas dari berbagai tantangan. Salah satu tantangannya adalah kurangnya integrasi antara deteksi dan klasifikasi kendaraan serta pengenalan karakter pelat nomor dalam satu sistem yang komprehensif. Sistem yang ada saat ini sering kali hanya fokus pada salah satu aspek, seperti deteksi kendaraan atau pengenalan pelat nomor, sehingga memerlukan proses tambahan untuk mengintegrasikan data tersebut. Hal ini meningkatkan kompleksitas sistem dan mengurangi efisiensi dalam pengelolaan parkir secara *real-time*. Selain itu, keterbatasan daya komputasi juga menjadi

permasalahan dalam pengimplementasian sistem *smart parking* (Tarumingkeng, 2024).

Model AI berbasis visi komputer umumnya membutuhkan daya komputasi tinggi, sehingga tanpa optimasi yang tepat, implementasi sistem dapat mengalami keterlambatan proses dan menurunkan efisiensi operasional. Untuk mengatasi keterbatasan ini, penelitian ini mengusulkan pengembangan model YOLOv11 yang dioptimalkan melalui teknik *pruning* atau kuantisasi (*quantization*). YOLOv11 dipilih karena kemampuannya untuk mencapai *balanced accuracy* yang tinggi, mencapai akurasi 99,39% dalam mendeteksi *multiple object* berupa kendaraan, dengan F1-score 0,994, sekaligus memiliki latensi inferensi yang relatif rendah dibandingkan model lain seperti YOLOv9e (Luz et al., 2024). Selain itu YOLOv11 juga kuat dalam mendeteksi kendaraan dengan berbagai ukuran dan jenis (Sivakoti, 2024).

Pruning sendiri adalah teknik pengurangan parameter model dengan menghilangkan bobot yang tidak signifikan tanpa mengorbankan akurasi secara drastis. Penelitian Yuang Jiang et al. (2022) menunjukkan bahwa model yang dioptimalkan melalui proses *pruning* memungkinkan pelatihan model yang lebih efisien dari sisi komputasi dan komunikasi, dengan tetap mempertahankan akurasi mendekati model asli. Penelitian ini membuktikan bahwa *pruning* mengurangi waktu pelatihan model hingga 33% dan mengurangi kebutuhan *floating-point operations* per *second* (FLOPs) hingga lebih dari 50% (Jiang et al., 2023).

Quantization, di sisi lain, adalah teknik optimasi yang mengurangi presisi numerik bobot model (misalnya, dari *floating-point* 32-bit ke integer 8-bit), sehingga mengurangi kebutuhan memori dan meningkatkan efisiensi komputasi tanpa kehilangan akurasi yang signifikan. Penelitian Younes El Bouzekraoui et al. (2024) membuktikan bahwa *quantization* memungkinkan model dapat diimplementasikan pada perangkat dengan sumber daya terbatas untuk melakukan deteksi *real-time*, dengan peningkatan kecepatan inferensi hingga 409,85%, sementara tetap mempertahankan akurasi mAP@0.5 sebesar 0,965 dan mAP@0.5:0.95 sebesar 0,833 (El Bouzekraoui et al., 2024).

Penelitian ini juga mengusulkan integrasi YOLOv11 dengan PaddleOCR untuk menciptakan sistem parkir cerdas yang komprehensif. Model YOLOv11 digunakan untuk mengklasifikasikan kendaraan (roda dua atau roda empat) sekaligus mendeteksi satu atau beberapa pelat nomor kendaraan secara *real-time*. Hasil deteksi pelat nomor dari YOLOv11 kemudian diproses oleh PaddleOCR untuk mengenali karakter alfanumerik pada pelat nomor kendaraan yang terdeteksi. PaddleOCR dipilih karena kemampuannya yang superior dalam mengenali teks dengan akurasi tinggi pada kondisi pencahayaan yang bervariasi dan berbagai orientasi teks. Penelitian oleh Du et al. (2020) menunjukkan bahwa PaddleOCR mencapai akurasi deteksi teks 89.8% dan akurasi pengenalan 95.2% pada *benchmark* dataset publik, mengungguli banyak *framework* OCR lainnya dalam hal kecepatan dan akurasi (Du et al., 2020).

Kombinasi ini tidak hanya meningkatkan akurasi deteksi dan pengenalan karakter, tetapi juga memastikan sistem dapat berjalan secara efisien pada perangkat dengan daya komputasi rendah. Dengan solusi ini, diharapkan pengelolaan parkir di Kota Makassar menjadi lebih transparan, efisien, dan berkontribusi pada peningkatan PAD.

1.2 Landasan Teori

1.2.1 Visi Komputer (*Computer Vision)*

Computer Vision atau visi komputer adalah sub bidang dari Kecerdasan Buatan (AI) yang berfokus pada pengembangan sebuah sistem otonom dalam meniru kemampuan manusia khususnya dalam bidang visual. Visi komputer bekerja dengan cara melakukan ekstraksi informasi dari data yang disediakan berupa gambar digital atau video, untuk menghasilkan informasi yang memungkinkan pengambilan keputusan ataupun rekomendasi berdasarkan pada data-data yang telah diubah menjadi sebuah informasi (Alsakka et al., 2023). Perkembangan visi Komputer membuat teknologi ini dapat mengatasi berbagai macam tugas visual seperti deteksi objek, klasifikasi gambar, pelacakan objek, pengenalan wajah dan lainnya (Alsakka et al., 2023).

Kemajuan dalam *deep Learning*, didukung dengan ketersediaan *dataset* yang besar, serta peningkatan kekuatan komputasi komputer, terutama melalui teknologi GPU, telah mendorong perkembangan Visi Komputer ke tingkat kemampuan yang sangat bermanfaat dan menjadikannya sebagai salah satu bidang dari Kecerdasan Buatan yang paling dinamis dan berpengaruh saat ini (Khan et al., 2018). Perkembangan Visi Komputer telah memberikan berbagai macam pengaruh yang signifikan di masa ini, salah satunya adalah kemampuan visi komputer dalam mendeteksi pelat kendaraan secara akurat dengan akurasi yang tinggi (Alwateer et al., 2024).

1.2.2 Deep Learning

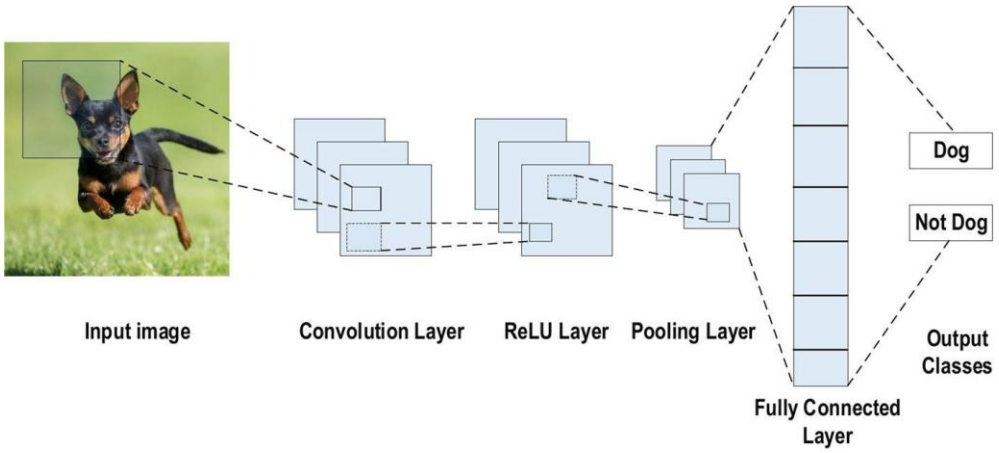
Deep Learning merupakan sebuah metode pembelajaran mesin (*machine learning*) yang menggunakan model komputasi berlapis-lapis untuk mempelajari representasi data dengan berbagai tingkat abstraksi. Metode ini memungkinkan komputer untuk meniru cara kerja otak manusia sehingga mesin dapat membangun konsep-konsep yang kompleks dengan cara menyusunnya dari konsep-konsep yang lebih sederhana. Metode *Deep Learning* memanfaatkan arsitektur *neural network* yang dalam kemudian dilatih menggunakan algoritma *backpropagation*, yang memberikan sistem *deep learning* kemampuan untuk menemukan pola-pola yang rumit dalam data, seperti mengenali objek pada gambar, tanpa perlu diprogram secara eksplisit (LeCun et al., 2015).

Kemajuan unit pemrosesan grafis (GPU) serta ketersediaan data dalam skala besar di era digital ini, memungkinkan perkembangan *Deep Learning* menjadi semakin pesat untuk belajar secara lebih efektif dan lebih akurat. Perkembangan

Deep Learning menjadikannya salah satu teknologi transformatif di berbagai bidang, seperti Kesehatan, *Natural Language Processing*, hingga *Computer Vision*, menjadikan teknologi *deep learning* sebagai teknologi yang sangat relevan dan vital di masa ini (Sharifani & Amini, 2023).

1.2.3 Convolutional Neural Network (CNN)

CNN (*Convolutional Neural Network*) adalah jenis jaringan saraf tiruan yang menggunakan operasi konvolusi untuk dapat mengekstraksi pola dan fitur dari data. Pengembangan CNN awalnya hanya dirancang untuk melakukan operasi pengolahan gambar, namun seiring berjalannya waktu, kegunaan CNN berkembang hingga kini CNN dapat digunakan untuk melakukan operasi pada data berurutan seperti teks, suara, dan bahasa tergantung pada kemampuan yang diberikan atau diajarkan kepada model (Chen & Wu, 2017). CNN bekerja dengan meniru cara kerja sistem visual manusia dalam mengenali pola, melalui lapisan-lapisan yang mempelajari fitur dari yang sederhana hingga kompleks.



Gambar 1. Ilustrasi arsitektur *convolutional neural network* (CNN) untuk klasifikasi objek

Secara umum, CNN terdiri dari beberapa komponen utama, yaitu *convolution layer, pooling layer,* dan *fully connected layer*. Lapisan konvolusi berfungsi mengekstraksi fitur penting seperti tepi atau sudut dari citra menggunakan filter, kemudian dilanjutkan dengan fungsi aktivasi non-linear (ReLU) untuk memungkinkan model mempelajari pola kompleks. Lapisan *pooling* digunakan untuk mengurangi dimensi data dan meningkatkan efisiensi komputasi, sedangkan lapisan *fully connected* berperan dalam mengklasifikasikan hasil ekstraksi fitur menjadi kategori tertentu (Alzubaidi et al., 2021).

Seiring perkembangan teknologi, arsitektur CNN terus berevolusi dari LeNet menuju model modern seperti AlexNet, VGGNet, ResNet, dan EfficientNet. Menurut (Alzubaidi et al., 2021), perkembangan ini difokuskan pada peningkatan akurasi,

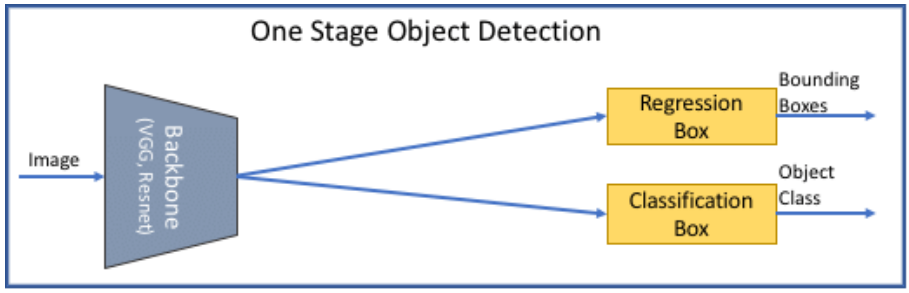
efisiensi komputasi, serta integrasi mekanisme perhatian (*attention mechanism*) agar CNN mampu mengenali fitur peting secara lebih efektif. Dengan keunggulan tersebut, CNN kini menjadi fondasi utama dalam pengembangan sistem visi komputer.

1.2.4 Deteksi Objek (Object Detection)

Object Detection atau deteksi objek merupakan cabang dari visi komputer yang bertujuan untuk mengenali dan menentukan lokasi berbagai objek yang terdapat dalam suatu citra atau video. Berbeda dengan *image classification* yang hanya mengidentifikasi jenis objek dalam satu gambar, deteksi objek tidak hanya mengenali kategori objek tetapi juga memberikan informasi posisi berupa *bounding box* yang mengelilingi objek tersebut (Zou et al., 2023). Dengan demikian, *object detection* memiliki dua tugas utama, yaitu *classification* (menentukan label objek) dan *localization* (menentukan posisi objek di dalam citra).

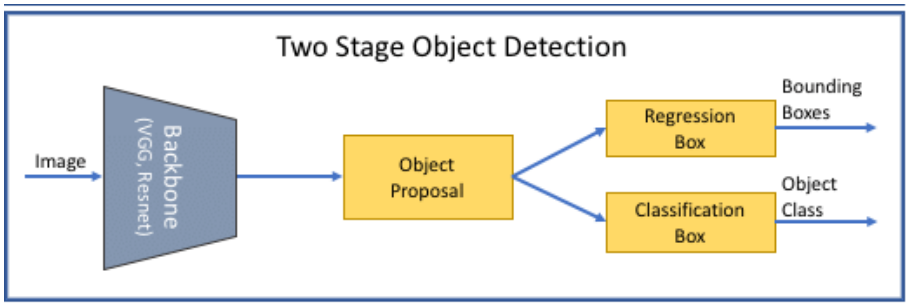
Perkembangan pesat *deep learning* membawa perubahan besar dalam bidang deteksi objek melalui penerapan jaringan saraf konvolusional (*Convolutional Neural Network* atau CNN). Menurut (Zou et al., 2023), metode modern deteksi objek berbasis deep *learning* terbagi menjadi dua kategori utama, yaitu *two-stage detector* dan *one-stage detector*. Kedua metode ini memiliki perbedaan mendasar dalam alur kerja, kecepatan deteksi, serta tingkat akurasi hasil deteksinya.

Two-Stage Detector. Metode *two-stage detector* bekerja melalui dua tahapan utama. Tahap pertama disebut *region proposal stage*, yaitu proses menghasilkan sejumlah kandidat area yang kemungkinan besar berisi objek. Tahap kedua adalah *classification and bounding box regression stage*, di mana setiap kandidat area tersebut dianalisis untuk menentukan jenis objek dan memperbaiki posisi *bounding box*-nya. Contoh Populer dari pendekatan ini adalah R-CNN, Fast R-CNN, dan Faster R-CNN. Metode ini memiliki tingkat akurasi yang tinggi karena proses identifikasi dilakukan secara selektif terhadap area yang relevan, namun kelemahannya terletak pada waktu komputasi yang relatif lebih lama (Zhang et al., 2021).



Gambar 2. Ilustrasi arsitektur dasar *one-stage object detector*

One-Stage Detector. Berbeda dengan pendekatan dua tahap, *one-stage detector* melakukan proses deteksi dan klasifikasi secara langsung dalam satu jaringan CNN terpadu. Model ini menghilangkan tahap *region proposal* dan langsung melakukan prediksi terhadap posisi serta kelas objek pada seluruh citra. Pendekatan ini dikenal sebagai metode berbasis regresi (*regression-based detection*). Contoh arsitektur yang termasuk dalam kategori ini antara lain YOLO (*You Only Look Once*), SSD (*Single Shot Multibox Detector*), dan DSSD (*Deconvolutional Single Shot Detector*). YOLO memproses seluruh citra hanya sekali dan secara langsung menghasilkan *bounding box* beserta probabilitas kelas untuk setiap objek. Hal ini menjadikan YOLO memiliki kecepatan deteksi yang sangat tinggi, jauh lebih cepat dibandingkan model *two-stage*. Namun demikian, metode *one-stage* cenderung memiliki akurasi yang sedikit lebih rendah, terutama dalam mendeteksi objek kecil atau berdekatan (Zhang et al., 2021).



Gambar 3. Ilustrasi arsitektur dasar one-stage object detector

1.2.5 *You Only Look Once* (YOLO)

YOLO (You Only Look Once) merupakan salah satu algoritma object detection berbasis *deep learning* yang dikembangkan oleh Joseph Redmon pada tahun 2016. Berbeda dengan metode deteksi dua tahap seperti R-CNN yang memisahkan proses *region proposal* dan klasifikasi, YOLO memperlakukan deteksi objek sebagai masalah regresi tunggal (*single-pass regression problem*). Model ini memproses seluruh citra hanya sekali melalui jaringan saraf konvolusional, kemudian secara langsung menghasilkan prediksi *bounding box* dan probabilitas kelas secara simultan (Gallagher & Oughton, 2025).

Secara umum, arsitektur YOLO terdiri dari tiga komponen utama, yaitu *backbone*, *neck*, dan *head*. *Backbone* berfungsi mengekstraksi fitur dari citra *input*, *neck* melakukan fusi informasi multi-skala untuk memperkaya representasi fitur, sedangkan *head* menghasilkan prediksi akhir berupa koordinat *bounding box*, skor kepercayaan (*confidence score*), dan probabilitas kelas. Pendekatan ini memungkinkan YOLO untuk mencapai kecepatan deteksi tinggi dengan akurasi yang kompetitif, menjadikannya populer dalam berbagai aplikasi *real-time*.

Sejak versi pertamanya (YOLOv1), YOLO telah mengalami perkembangan signifikan. YOLOv2 memperkenalkan *batch normalization* dan *anchor boxes* untuk meningkatkan akurasi deteksi, sementara YOLOv3 menggunakan *Darknet-53* sebagai *backbone* yang lebih dalam dan melakukan prediksi pada tiga skala untuk mendeteksi objek berukuran kecil. YOLOv4 menambahkan teknik augmentasi seperti *Mosaic data augmentation* dan *CSPDarknet53*, diikuti oleh YOLOv5 yang lebih ringan serta fleksibel untuk implementasi industri. Versi-versi selanjutnya seperti YOLOv7, YOLOv8, hingga YOLOv10 terus meningkatkan efisiensi arsitektur dengan memperkenalkan modul seperti *E-ELAN, C2f,* serta mekanisme penghapusan *Non-Maximum suppression* (*NMS*) untuk mempercepat inferensi (Gallagher & Oughton, 2025). Kemampuan YOLO dalam menyeimbangkan kecepatan, akurasi, dan efisiensi komputasi menjadikan model ini sangat cocok di implementasikan dalam sistem pantauan cerdas modern.

1.2.6 YOLOv11

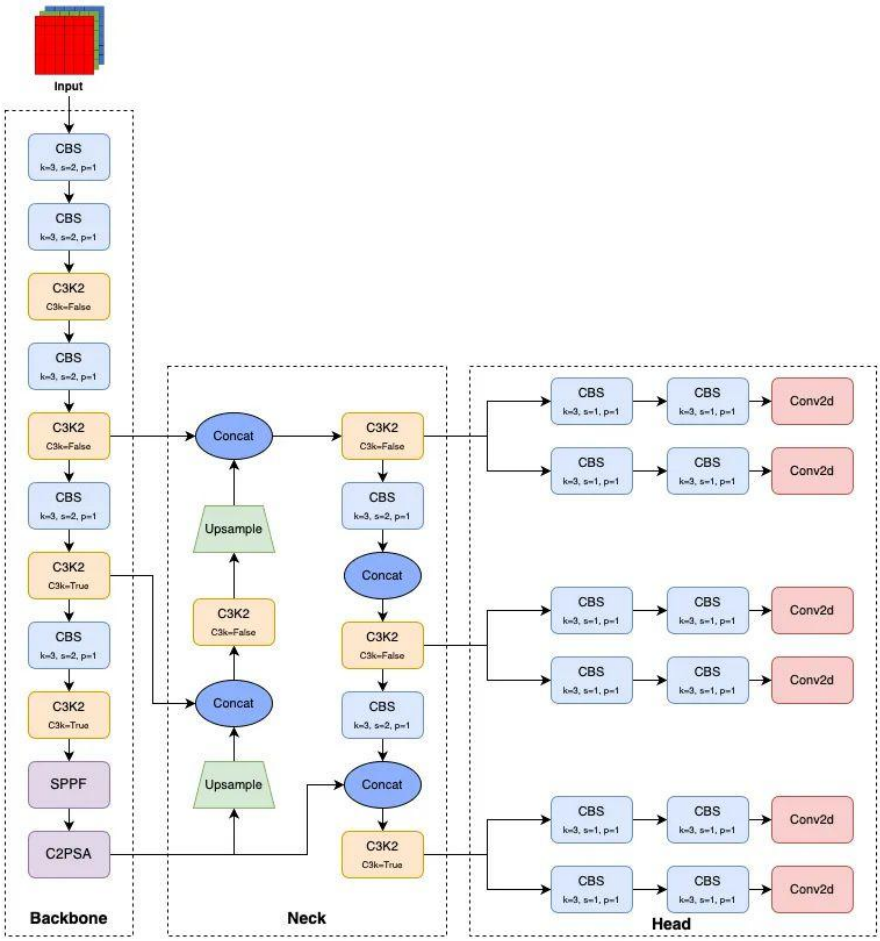
YOLOv11 merupakan versi terbaru dari keluarga *You Only Look Once (YOLO)* yang dikembangkan oleh Ultralytics pada tahun 2024 dan diperkenalkan secara resmi pada konferensi YOLO Vision 2024 (Khanam & Hussain, 2024). Model ini hadir sebagai penyempurnaan dari YOLOv10 dengan peningkatan yang berfokus pada efisiensi komputasi, kecepatan inferensi, serta kemampuan mendeteksi objek kecil dan saling tumpang tindih dengan lebih akurat.

Arsitektur YOLOv11 mempertahankan struktur tiga komponen utama YOLO, yaitu *backbone*, *neck*, dan *head*, namun membawa sejumlah inovasi penting yang membedakan versi ini dari versi sebelumnya.

> ***Backbone***, YOLOv11 memperkenalkan C3k2 *block*, yaitu bentuk modifikasi dari *Cross Stage Partial (CSP) Bottleneck* yang menggunakan dua konvolusi kecil (*kernel size 2*) untuk menggantikan satu konvolusi besar seperti pada versi YOLOv8. Pendekatan ini meningkatkan efisiensi komputasi tanpa mengorbankan akurasi (Jegham et al., 2025). YOLOv11 juga memperkenalkan C2PSA (*Cross-Stage Partial with Spatial Attention*) yang menggabungkan mekanisme *cross-stage connection* dengan *spatial attention* yang memungkinkan mode berfokus pada area penting dalam citra. Kombinasi ini meningkatkan kemampuan model dalam mendeteksi objek berukuran kecil serta objek yang sebagian tertutupi.
>
> ***Neck*** dan ***Head***, YOLOv11 menggunakan kembali blok C3k2 yang disusun untuk mempercepat proses fusi fitur dari berbagai skala serta memperbaiki representasi spasial. Blok ini juga dikombinasikan dengan lapisan CBS (*Convolution-BatchNorm-SiLU*) yang meningkatkan stabilitas pelatihan dan kualitas model dalam memprediksi *bounding box* (Khanam & Hussain, 2024).

Performa YOLOv11 dibandingkan dengan model YOLO versi sebelumnya menunjukkan hasil yang lebih tinggi dalam hal keseimbangan antara akurasi dan

efisiensi. Model versi ini secara konsisten menghasilkan nilai mAP50-95 yang tinggi dengan waktu inferensi yang lebih singkat dibandingkan YOLOv9 dan YOLOv10 (Jegham et al., 2025). Peningkatan ini menjadikan YOLOv11 unggul dan sesuai dalam aplikasi *real-time* yang membutuhkan kecepatan dan ketepatan komputasi.



Gambar 4. Diagram blok arsitektur YOLOv11 (*Backbone, Neck, dan Head*)

1.2.7 *Object Tracking*

Object Tracking atau pelacakan objek merupakan salah satu bidang penting dalam visi komputer yang bertujuan untuk mendeteksi, mengenali, dan mengikuti pergerakan suatu objek secara berurutan dalam rangkaian citra atau video (Soleimanitaleb et al., 2019). Cara kerja *object tracking* dimulai dengan mendeteksi posisi awal objek pada *frame* pertama, kemudian mempertahankan identitas dan posisinya pada *frame-frame* berikutnya.

Dalam beberapa tahun terakhir, pendekatan *deep learning* menjadi dominan karena kemampuannya menghasilkan pelacakan yang lebih akurat dan adaptif terhadap perubahan lingkungan. Salah satu arah perkembangan terkini adalah penerapan *Deep Reinforcement Learning* (DRL) pada *object tracking* (Nguyen et al., 2025). Pendekatan ini memungkinkan sistem pelacakan untuk mengantisipasi gerakan objek di masa depan, menjaga konsistensi identitas objek, dan belajar secara adaptif. Selain itu, penelitian terbaru memperkenalkan konsep *Multi-Agent Deep Reinforcement Learning* (MADRL), di mana beberapa agen bekerja secara kolaboratif untuk melakukan pelacakan (Nguyen et al., 2025). Perkembangan metode pelacakan objek dari pendekatan berbasis fitur hingga *deep reinforcement learning* menunjukkan kemajuan signifikan dalam efisiensi dan akurasi.

1.2.8 Optimalisasi Model

Optimalisasi model (*model optimization*) merupakan proses untuk meningkatkan kinerja dan efisiensi jaringan saraf dengan menyesuaikan arsitektur, parameter, maupun teknik kompresi agar model dapat bekerja lebih cepat dengan tetap menjaga akurasi. Dalam konteks *deep learning*, optimalisasi tidak hanya bertujuan untuk meningkatkan akurasi prediksi, tetapi juga untuk mengurangi kompleksitas model, mempercepat inferensi, dan menyesuaikan model agar dapat dijalankan pada perangkat dengan sumber daya terbatas seperti *embedded systems* atau GPU berdaya rendah (Hu et al., 2023).

Salah satu pendekatan populer untuk mempercepat kinerja model adalah dengan menggunakan optimalisasi metode *pruning* atau *quantization*. *Pruning* dilakukan dengan menghapus bobot atau neuron yang kontribusinya kecil terhadap hasil akhir, sehingga model menjadi lebih ringan dan cepat tanpa adanya penurunan performa yang signifikan. Sementara itu, *quantization* adalah metode optimalisasi yang bekerja dengan cara mengubah representasi bobot dari presisi tinggi seperti float32, ke presisi yang lebih rendah seperti int8, yang secara langsung menurunkan kebutuhan memori dan meningkatkan kecepatan inferensi (Gunawan, 2020).

1.2.9 *Optical Character Recognition* (OCR)

Optical Character Recognition (OCR) merupakan teknologi yang digunakan untuk mengonversi teks pada dokumen cetak atau gambar menjadi bentuk digital yang dapat dikenali dan diolah oleh komputer (Islam et al., 2017). Teknologi ini bekerja dengan meniru kemampuan manusia dalam membaca teks, namun menggunakan mekanisme optik dan algoritma pengenalan pola untuk mengidentifikasi karakter dari citra *input*. OCR memiliki peranan penting dalam mengotomatisasi proses pembacaan teks, seperti pada sistem pengarsipan dokumen, pembacaan cek bank, serta pengenalan pelat nomor kendaraan.

Secara umum, proses kerja OCR terdiri dari beberapa tahapan utama, yaitu *scanning*, *preprocessing*, *segementation*, *feature extraction*, dan *recognition*. Tahap *scanning* berfungsi mengubah dokumen fisik menjadi citra digital, biasanya melalui kamera atau pemindai. Selanjutnya, pada tahap *preprocessing*, citra tersebut diolah

untuk meningkatkan kualitasnya melalui proses *binarization, noise removal,* dan *normalization*. Tahap *segmentation* memisahkan karakter satu per satu dari citra, agar setiap huruf dapat dianalisis secara individual. Pada tahap *feature extraction*, sistem mengekstraksi ciri khas dari setiap karakter, seperti bentuk, garis, sudut, dan lengkungan. Fitur-fitur ini kemudian digunakan oleh algoritma klasifikasi untuk mengenali huruf atau angka yang terdapat pada citra pada tahap *recognition* (Mithe et al., 2013).

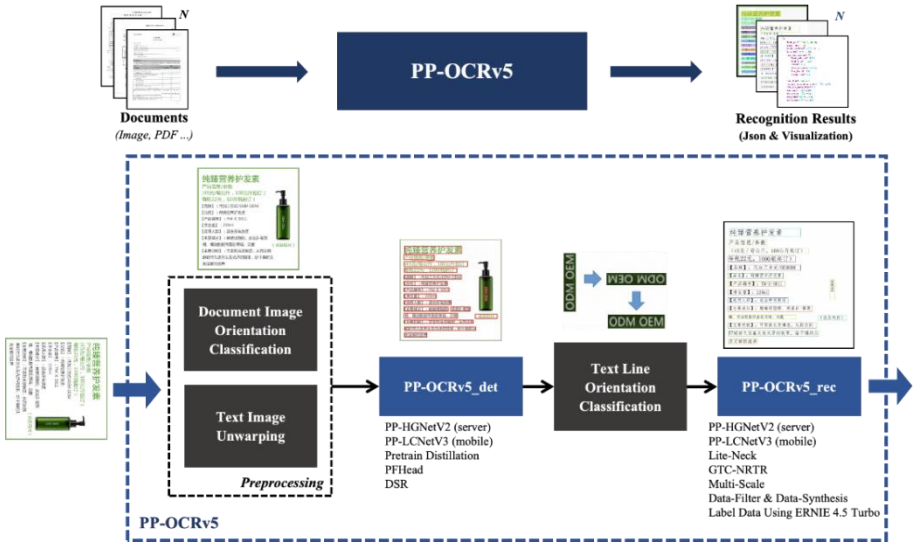
Keakuratan sistem OCR sangat bergantung pada kualitas citra *input* serta jenis teks yang dikenali, seperti perbedaan font, bahasa, dan gaya tulisan tangan. Oleh karena itu, tahap *post-processing* sering ditambahkan untuk meningkatkan hasil akhir dengan memanfaatkan teknik seperti pemeriksaan ejaan (*spell checking*) atau model probabilistik berbasis *n-gram* (Islam et al., 2017).

1.2.10 PaddleOCR

PaddleOCR merupakan salah satu pustaka (*library*) *open-source* berbasis *deep learning* yang dikembangkan oleh Baidu melalui *platform* PaddlePaddle. *Framework* ini dirancang untuk melakukan proses *text detection* (deteksi teks) dan *text recognition* (pengenalan teks) secara efisien. Keunggulan utamanya terletak pada kecepatan inferensi yang tinggi dan ukuran model yang ringan (Hakasin et al., 2025).

Secara arsitektural, PaddleOCR mengadopsi pendekatan dua tahap (*two-stage*) dalam sistem pengenalannya.

> ***Text Detection* (Deteksi Teks)**. Tahap ini bertujuan untuk mendeteksi dan melokalisasi area teks pada citra. PaddleOCR menggunakan metode *Differentiable Binarization* (DB), yang berfungsi sebagai mekanisme segmentasi adaptif yang dapat dilatih (*trainable*). Mekanisme ini memungkinkan model untuk secara dinamis menentukan area teks dan memisahkannya dari latar belakang dengan presisi tinggi (Hakasin et al., 2025).
>
> ***Text Recognition* (Pengenalan Teks)**. Setelah area teks terdeteksi, proses pengenalan karakter dilakukan menggunakan arsitektur *Convolutional Recurrent Neural Network* (CRNN) atau arsitektur berbasis *Transformer*. Arsitektur CRNN mengombinasikan kemampuan CNN dalam mengekstraksi fitur spasial dengan kemampuan RNN dalam mengenali dependensi sekuel (urutan karakter). Kombinasi ini memungkinkan PaddleOCR untuk membaca teks dalam berbagai orientasi dan kondisi pencahayaan (Karadag et al., 2025).



Gambar 5. Alur pemrosesan citra untuk pengenalan teks dengan PaddleOCR

PaddleOCR juga mampu menghasilkan keluaran pada level karakter (*character-level output*) yang disertai dengan skor keyakinan (*confidence score*). *Output* terperinci ini memudahkan proses analisis dan evaluasi kuantitatif terhadap hasil pengenalan teks (Karadag et al., 2025).

1.2.11 Evaluasi Model

Evaluasi model merupakan proses pengukuran performa suatu sistem pembelajaran mesin dalam mengenali pola dan menghasilkan prediksi yang akurat terhadap data uji. Pada penelitian ini, proses evaluasi model bertujuan untuk mengukur kemampuan model YOLOv11 dalam mendeteksi kendaraan, model OCR dalam mengenali karakter pada pelat nomor, serta algoritma ByteTrack dalam melakukan pelacakan objek secara berkelanjutan. Proses ini dilakukan untuk memastikan bahwa sistem tidak hanya akurat, tetapi juga efisien dan stabil saat digunakan. Ada beberapa metrik utama yang digunakan dalam pengukuran model pada penelitian ini.

Precision, mengukur proporsi prediksi positif yang benar dari keseluruhan prediksi positif. Nilai tinggi menunjukkan sedikit *false positive* (deteksi palsu). *Precision* penting untuk memahami tingkat “kesalahan alarm” model deteksi.

Recall, digunakan untuk mengukur proporsi objek sebenarnya yang berhasil dideteksi oleh model. Nilai tinggi menunjukkan sedikit *false negative* (objek terlewat). *Recall* membantu menilai kemampuan model menemukan semua *instance* objek di gambar (Zhu et al., 2020).

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

Keterangan:

- TP (True Positive) = objek yang terdeksi dengan benar,
- FP (False Positive) = deteksi salah (objek tidak ada tetapi terdeteksi),
- FN (False Negative) = objek nyata yang tidak terdeteksi.

Mean Average Precision (mAP), merupakan metrik utama yang digunakan untuk menilai kinerja deteksi objek pada model YOLO. mAP mengukur seberapa baik model mengenali objek dari berbagai kelas dan seberapa tepat posisi *bounding box* yang dihasilkan dibandingkan dengan *ground truth* (Zhu et al., 2020).

$$AP = \int_0^1 p(r)dr$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

Keterangan:

- r = recall
- $p(r)$ = Precision pada saat *Recall* r
- N = Jumlah kelas objek yang diuji
- AP_i = Nilai *Average Precision* untuk kelas ke-i

Dalam penelitian berbasis YOLOv11, nilai mAP dihitung menggunakan batas *Intersection over Union* (IoU) sebesar 0.5 (mAP@0.5) atau rentang 0.5-0.95 (mAP@0.5:0.95), yang menggambarkan ketepatan prediksi model terhadap posisi objek sebenarnya.

Multiple Object Tracking Accuracy (MOTA) dan **Multipel Object Tracking Precision (MOTP)**, merupakan dua metrik utama yang digunakan dalam evaluasi akurasi sistem pelacakan model.

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t}$$

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t}$$

Keterangan:

- FN = jumlah *false negative*,
- FP = jumlah *false positives*,
- $IDSW_t$ = jumlah kesalahan identitas objek selama pelacakan,
- GT = jumlah total *ground truth* objek,
- $d_{i,t}$ = jarak kesalahan antara posisi prediksi dan posisi sebenarnya,
- c_t = jumlah objek yang terdeteksi dengan benar pada waktu t.

MOTA menilai akurasi keseluruhan pelacakan, sedangkan MOTP mengukur ketepatan spasial dari posisi objek yang diikuti. Nilai MOTA dan MOTP yang tinggi menunjukkan sistem pelacakan yang stabil, minim kehilangan target, serta tepat dalam mempertahankan identitas objek dari satu *frame* ke *frame* berikutnya (Soleimanitaleb & Keyvanrad, 2022).

Identity Switch (IDSW) atau pergantian identitas, merupakan metrik yang menghitung frekuensi algoritma pelacak (*tracker*) mengubah identitas (ID) dari satu objek yang sama selama durasi pelacakan. Pergantian ID biasanya terjadi ketika pelacak kehilangan jejak objek untuk sesaat (misalnya akibat oklusi atau *motion blur*) dan kemudian menginisialisasi ulang objek tersebut dengan ID yang baru saat terdeteksi kembali (Wojke et al., 2017).

$$IDSW = \sum_{t=1}^{T} idsw_t$$

Keterangan:

- $idsw_t$ = jumlah *identity switch* yang terjadi pada frame ke-t
- T = jumlah total frame dalam sekuens video

Character Error Rate (CER) merupakan metrik evaluasi yang digunakan untuk model OCR. CER adalah metrik standar untuk mengukur kesalahan karakter, semakin kecil CER semakin baik (Neudecker et al., 2021).

$$CER = \frac{i + s + d}{n}$$

Keterangan:

- i = jumlah karakter yang ditambahkan (*insertions*),
- s = jumlah karakter yang salah (*substitutions*),
- d = jumlah karakter yang dihapus (*deletions*),
- n = total karakter pada *ground truth*.

1.2.12 ONNX

ONNX (*Open Neural Network Exchange*) merupakan *intermediate representation* (IR) terbuka yang dikembangkan untuk memfasilitasi interoperabilitas model *deep learning* antar berbagai *framework* seperti PyTorch dan TensorFlow. Format ini memungkinkan model yang telah dilatih pada satu *framework* dapat diekspor dan dijalankan pada *framework* atau perangkat keras lain tanpa memerlukan perubahan arsitektur. Struktur model dalam ONNX direpresentasikan dalam bentuk *computational graph* yang berisi informasi operator, aliran data, serta bobot model, sehingga proses konversi dan eksekusi dapat dilakukan secara efisien (Jajal et al., 2024).

Ekosistem ONNX dilengkapi dengan ONNX Runtime (ORT), yaitu mesin inferensi lintas platform (*cross-platform*) berperforma tinggi. ORT mendukung berbagai *execution providers* (akselerator perangkat keras) seperti CUDA, TensorRT, OpenVino, CoreML, dan NNAPI.

Keunggulan utama ONNX antara lain adalah *portabilitas* yang tinggi, efisiensi komputasi, serta dukungan terhadap berbagai teknik optimasi seperti *graph optimization*, *operator fusion*, dan *constant folding*. Selain itu, ONNX juga mendukung proses kompresi model melalui metode seperti *quantization* (kuantisasi) dan *pruning* (pemangkasan) untuk mempercepat proses inferensi tanpa mengorbankan akurasi secara signifikan (Joshua et al., 2025).

1.3 Rumusan Masalah

Adapun fokus pertanyaan yang ingin dijawab melalui penelitian ini meliputi:

1. Bagaimana mengoptimalkan model YOLOv11 menggunakan teknik *pruning* atau *quantization* untuk mendeteksi kendaraan pada sistem *smart parking*?
2. Bagaimana performa model YOLOv11 sebelum dan sesudah dilakukan optimasi menggunakan teknik *pruning* atau *quantization*?
3. Bagaimana merancang dan mengimplementasikan sistem *smart parking* yang mengintegrasikan deteksi kendaraan dan pengenalan pelat nomor secara *real-time*?

1.4 Tujuan dan Manfaat Penelitian

1.4.1 Tujuan

Tujuan utama yang ingin dicapai melalui penelitian ini adalah sebagai berikut:

1. Mengembangkan model YOLOv11 yang dioptimalkan menggunakan teknik *pruning* atau *quantization* untuk mendeteksi kendaraan secara efisien pada sistem *smart parking*.
2. Membandingkan dan mengevaluasi performa model YOLOv11 sebelum dan sesudah optimasi untuk memastikan peningkatan efisiensi dan akurasi sistem deteksi kendaraan dan pengenalan pelat nomor.

3. Merancang dan mengimplementasikan sistem *smart parking* yang mengintegrasikan deteksi kendaraan, *tracking* kendaraan, dan pengenalan pelat nomor secara otomatis dan *real-time*.

1.4.2 Manfaat

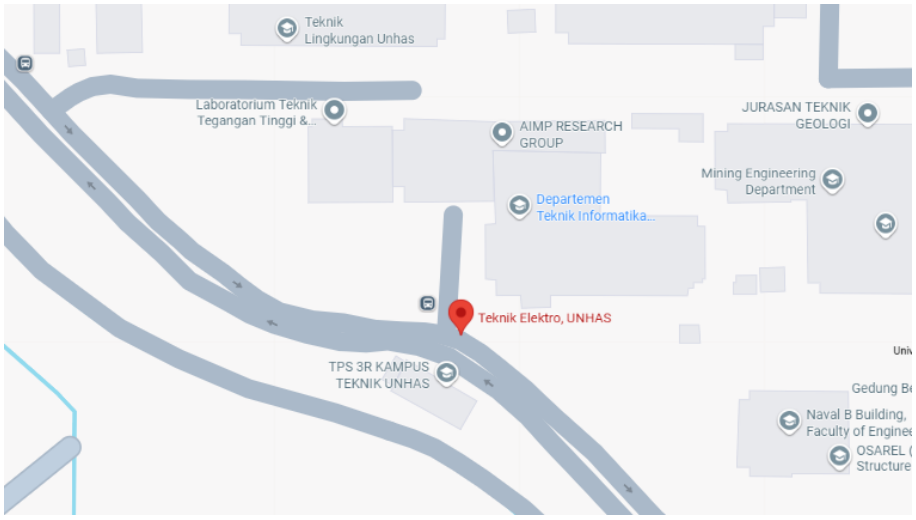
Penelitian ini bertujuan untuk memberikan kontribusi dalam bentuk manfaat berikut:

1. Meningkatkan efisiensi sistem parkir cerdas melalui integrasi deteksi kendaraan, klasifikasi jenis kendaraan dan pengenalan karakter pelat nomor dalam satu alur kerja otomatis
2. Memfasilitasi pengelolaan parkir yang lebih efisien dan otomatis melalui deteksi kendaraan dan identifikasi pelat nomor secara *real-time*.
3. Memberikan kemudahan dan kenyamanan dalam pelayanan parkir dengan sistem yang lebih cepat, aman, dan transparan.

BAB II
METODE PENELITIAN

2.1 Tempat dan Waktu Penelitian

Penelitian ini dilaksanakan selama periode Januari hingga Oktober 2025, terhitung sejak disetujuinya judul penelitian oleh pihak pembimbing. Pengambilan data dilakukan di area parkir Gedung Elektro, Fakultas Teknik, Universitas Hasanuddin, yang berlokasi di Jl. Malino, Kelurahan Romang Lompoa, Kecamatan Bontomarannu, Kabupaten Gowa. Seluruh proses pengembangan dan pengujian sistem dilaksanakan di Laboratorium *Artificial Intelligence* (AI), Departemen Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin.



Gambar 6. Lokasi Penelitian Fakultas Teknik, Universitas Hasanuddin.

2.2 Instrumen Penelitian

Berikut instrumen atau alat yang digunakan dalam penelitian ini:

1. Perangkat lunak
   a. Visual Studio Code
   b. Kaggle
   c. Microsoft Word
   d. Microsoft Excel
   e. Microsoft PowerPoint
   f. Google Chrome
   g. *Operating System* Windows 11 64-bit
   h. Zotero
2. Perangkat keras
   a. CCTV Vivotek IP9165-LPC

   b. Lenovo V14 ADA, *Processor* AMD Ryzen 3 3250U, *Graphic Card* AMD Radeon Graphics, RAM 12 GB
   c. Kaggle Cloud Platform, GPU 2x NVIDIA Tesla T4 15 GB VRAM/GPU, RAM 30 GB
3. Bahasa Pemrograman
   a. Python v3.11.9
4. Algoritma
   a. YOLOv11
   b. ByteTrack
   c. PaddleOCR
5. *Library*
   a. ONNX v1.16.1, Digunakan untuk konversi model PyTorch ke format *interoperable*.
   b. Onnx Runtime v1.23.2, Digunakan untuk inferensi model ONNX yang dioptimalkan pada CPU.
   c. Ultralytics v8.3.221, Digunakan untuk implementasi YOLOv11 dengan API lengkap untuk *training* dan *inference*.
   d. NumPy v1.26.4, Digunakan untuk komputasi numerik dan manipulasi *array* dalam *pipeline processing*.
   e. Pandas v2.1.4, Digunakan untuk manipulasi dan analisis data hasil evaluasi model.
   f. Matplotlib v3.8.4, Digunakan untuk visualisasi data dan hasil evaluasi model.
   g. Seaborn v0.13.2, Digunakan untuk visualisasi statistik dan grafik hasil evaluasi.
   h. Pillow v10.4.0, Digunakan untuk *image processing* dan manipulasi citra.
   i. PyTorch v2.6.0, Digunakan untuk *training* dan inferensi model YOLOv11.
   j. PaddleOCR v3.3.0, Digunakan untuk *optical character recognition* pada citra pelat nomor.
   k. Motmetrics v1.4.0, Digunakan untuk evaluasi metrik *multi-object tracking*.
   l. Jiwer v4.0.0, Digunakan untuk perhitungan *character error rate* pada evaluasi OCR.
   m. Thop v0.1.1, Digunakan untuk perhitungan jumlah parameter dan FLOP pada model PyTorch
   n. Onnx-tool v0.8.0, Digunakan untuk perhitungan jumlah parameter dan FLOP pada model ONNX

2.3 Tahapan Penelitian

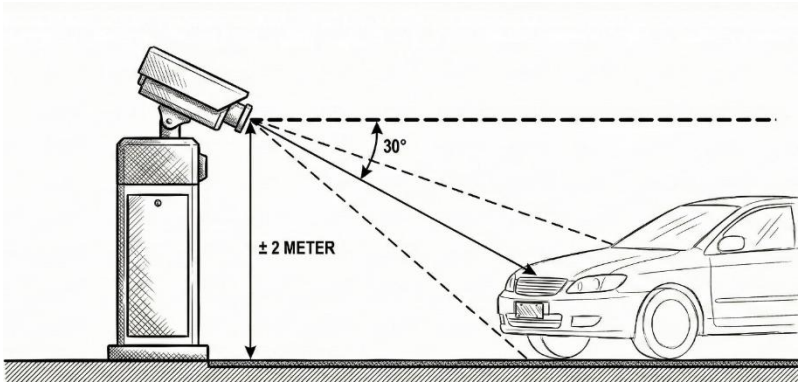
Gambar 7. Tahapan penelitian

Tahapan penelitian yang dilakukan dalam studi ini dapat dilihat pada Gambar 7. Penelitian diawali dengan studi literatur untuk menelaah berbagai sumber referensi yang relevan sebagai dasar penguatan teori dan pemahaman terhadap topik yang diteliti. Tahap berikutnya adalah analisis kebutuhan penelitian, yang bertujuan untuk mengidentifikasi permasalahan, merumuskan tujuan, serta menentukan data dan metode yang akan digunakan. Setelah kebutuhan penelitian dirumuskan, dilakukan pengambilan data sebagai bahan utama dalam proses pengembangan sistem. Data yang telah terkumpul kemudian dimanfaatkan pada tahap perancangan sistem, meliputi penyusunan alur kerja, pemilihan arsitektur model, serta penentuan komponen pendukung. Sistem yang telah dirancang selanjutnya diimplementasikan dan dilakukan pengujian untuk mengevaluasi performa serta tingkat akurasi hasilnya. Tahap akhir berupa analisis hasil dilakukan untuk menilai efektivitas sistem secara keseluruhan, dan seluruh rangkaian proses penelitian ini kemudian disusun dalam bentuk laporan akhir secara sistematis.

2.4 Teknik Pengambilan Data

2.4.1 Data Primer

Data primer yang digunakan dalam penelitian ini diperoleh langsung dari lokasi pengambilan data. Data yang dikumpulkan berupa *dataset* untuk pelatihan model *object detection*. *Dataset* ini berisi kumpulan gambar atau *frame* yang diambil dari hasil ekstraksi video yang merekam aktivitas kendaraan masuk dan keluar pada area parkir. Pengambilan data dilakukan menggunakan kamera CCTV Vivotek IP9165-LPC yang diarahkan langsung ke arah jalur masuk dan keluar kendaraan, seperti ditunjukkan pada gambar.



Gambar 8. Ilustrasi konfigurasi pemasangan kamera untuk pengambilan dataset

Berdasarkan gambar 8, kamera dipasang di atas besi palang parkir dengan ketinggian ± 2 meter dari permukaan jalan dengan kemiringan sekitar 30 derajat. Posisi ini dipilih untuk mendapatkan sudut yang cukup mendapatkan gambaran kendaraan yang melewati portal parkir, agar gambar kendaraan yang didapatkan cukup untuk model dapat mendeteksi dan mengklasifikasi kendaraan yang lewat, serta cukup dekat sehingga pelat kendaraan dapat terlihat jelas dan dapat dibaca oleh model. Contoh *dataset object detection* terdapat pada gambar 9.

Gambar 9. Contoh *dataset object detection*

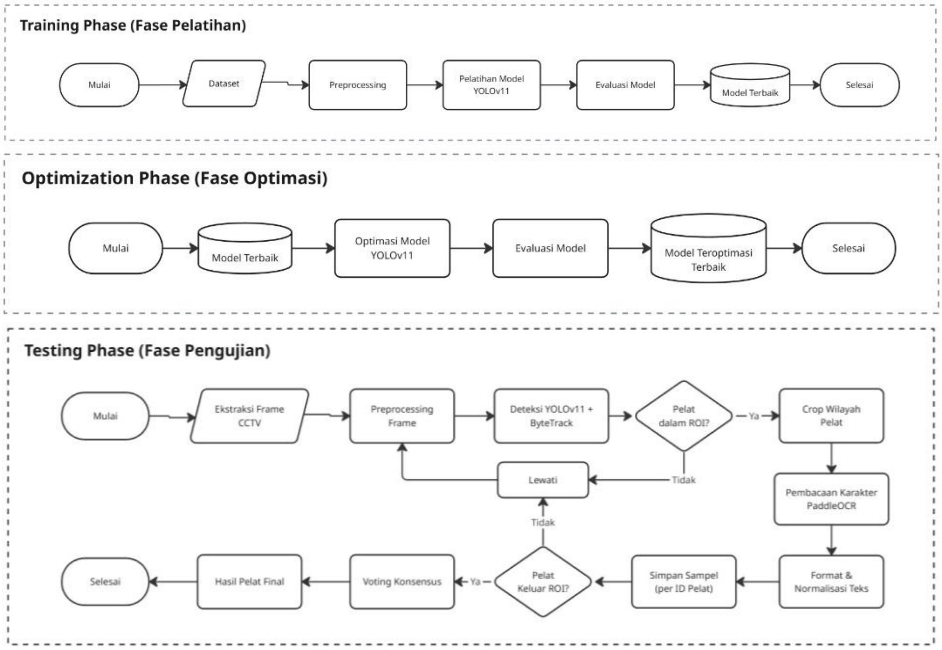
2.4.2 Data Sekunder

Data sekunder pada penelitian ini diperoleh dari dataset publik yang tersedia pada platform Roboflow. Dataset ini digunakan untuk melengkapi dan menyeimbangkan jumlah data pada salah satu kelas objek, yaitu *motorcycle*. Penambahan data sekunder dilakukan karena pada dataset utama terdapat ketidakseimbangan jumlah data (*class imbalance*) antara kelas *car* dan *motorcycle*, di mana jumlah citra kendaraan roda empat lebih banyak dibandingkan kendaraan roda dua.

Proses anotasi ulang (*re-annotation*) dilakukan terhadap seluruh citra yang diunduh sebelum dataset publik tersebut digabungkan dengan dataset utama, untuk memastikan kesesuaian label dan akurasi area deteksi (*bounding box*). Langkah ini bertujuan untuk menjaga konsistensi format anotasi serta meningkatkan kualitas dataset secara keseluruhan, sehingga model *object detection* dapat belajar dengan baik dan menghasilkan performa deteksi yang optimal pada seluruh kelas objek. Contoh dataset yang ditambahkan, terdapat pada gambar 10.

Gambar 10. Contoh dataset tambahan

2.5 Perancangan dan Implementasi Sistem



Gambar 11. Perancangan Alur Sistem

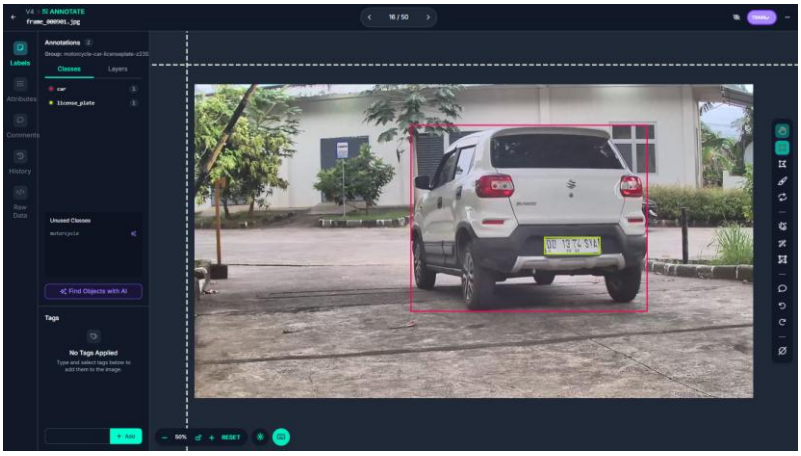
2.5.1 Tahap Training Model YOLOv11

2.5.1.1 Persiapan Dataset

Tahap persiapan dataset merupakan langkah kritis dalam proses *training*, karena kualitas dan struktur dataset akan sangat memengaruhi performa model yang dihasilkan. Dataset yang digunakan dalam penelitian ini merupakan gabungan dari data primer dan data sekunder untuk memastikan variasi dan representasi yang memadai. Data primer terdiri dari 622 citra unik yang diperoleh dari ekstraksi *frame* video *surveillance* dengan interval dua detik, memberikan variasi temporal dan spasial yang cukup. Untuk meningkatkan kemampuan generalisasi model terhadap berbagai tipe kendaraan dan kondisi visual yang mungkin tidak terwakili sepenuhnya dalam data lokal, dataset diperkaya dengan 425 citra sekunder yang bersumber dari repositori publik Roboflow. Kombinasi kedua sumber ini menghasilkan total dataset akhir sebanyak 1047 citra.

Proses anotasi atau pelabelan objek pada dataset primer dilakukan menggunakan platform Roboflow. Roboflow dipilih karena menyediakan antarmuka yang efisien untuk manajemen dataset kolaboratif serta fitur-fitur *built-in* untuk konversi format label yang kompatibel dengan YOLO. Seluruh citra dianotasi dengan tiga kelas objek utama, yaitu:

- *car* : Kendaraan roda empat
- *motorcycle* : Kendaraan roda dua
- *license_plate* : Pelat nomor kendaraan



Gambar 12. Proses pelabelan data menggunakan platform Roboflow

Setelah proses anotasi dan validasi kualitas label selesai, dataset dibagi menjadi dua subset dengan rasio 80:20. Sebanyak 838 citra (80%) dialokasikan sebagai *training* set untuk melatih model, sedangkan 209 citra (20%) digunakan sebagai *validation* set untuk mengevaluasi performa model dan mencegah *overfitting* selama proses pelatihan. Setiap citra dalam dataset dinormalisasi ke resolusi standar

640×640 piksel untuk memastikan konsistensi *input* sesuai dengan arsitektur *default* YOLOv11. Distribusi kelas, khususnya *license_plate*, dipastikan cukup merata di seluruh *subset* agar model dapat mempelajari representasi visual yang *robust* terhadap variasi pencahayaan, sudut pandang, dan jenis pelat kendaraan.

2.5.1.2 Preprocessing dan Augmentasi Data

Sebelum memasuki tahap pelatihan model, seluruh dataset akan melalui tahap *preprocessing* yang mencakup *resize* dan augmentasi. Proses *preprocessing* ini dieksekusi menggunakan platform Roboflow, yang merupakan platform sama dengan yang digunakan untuk pelabelan dan anotasi *bounding box*. Pendekatan ini bertujuan untuk memastikan format anotasi dan transformasi *bounding box* tetap konsisten pasca-*preprocessing*. Tahap pertama adalah proses *resize*, di mana setiap gambar diseragamkan ukurannya menjadi resolusi 640×640 piksel. Dimensi ini dipilih karena sesuai dengan ukuran *input* standar yang diterima oleh arsitektur YOLOv11, sehingga menjamin konsistensi *input* dan efisiensi proses pelatihan.

Tahap selanjutnya adalah proses augmentasi data. Proses ini diterapkan secara eksklusif pada data latih (*training set*). Penerapan transformasi ini bertujuan untuk memperkaya representasi data latih dan mengurangi risiko *overfitting* pada model. Parameter augmentasi yang diterapkan dalam proses ini meliputi:

Rotation (± 10°): Rotasi acak diterapkan pada citra dengan rentang -10 hingga +10 derajat. Augmentasi ini bertujuan untuk mensimulasikan variasi sudut kamera dan perubahan orientasi kendaraan saat bergerak memasuki area parkir. Dalam praktik lapangan, kamera CCTV mungkin tidak selalu dipasang dengan sudut yang sempurna terhadap alur lalu lintas, sehingga pelat nomor dapat terekam dalam berbagai orientasi. Dengan melatih model pada citra yang telah mengalami rotasi ringan (± 10°), model dapat belajar untuk mendeteksi pelat nomor dengan *robust* terhadap variasi orientasi ini, meningkatkan performa deteksi pada kondisi *real-world*.

(a)

(b)

(c)

Gambar 13. Contoh hasil perubahan rotasi: (a) sebelum perubahan rotasi; (b) setelah perubahan rotasi -10%; (c) setelah perubahan rotasi +10%;

Shearing (± 10° horizontal, ± 10° vertikal): *Shearing* adalah menerapkan transformasi geser secara acak dengan rentang ± 10° pada sumbu horizontal dan ± 10° pada sumbu vertikal. Augmentasi ini bertujuan untuk mensimulasikan distorsi perspektif ringan yang umum terjadi di lapangan akibat sudut pemasangan kamera yang tidak ideal atau pergeseran posisi kendaraan relatif terhadap lensa kamera. Transformasi *shear* menghasilkan efek visual di mana objek terlihat "miring" atau "condong" dari perspektif yang berbeda. Dengan melatih model pada data yang telah mengalami distorsi perspektif ringan ini, ketahanan (*robustness*) model terhadap variasi perspektif di dunia nyata dapat ditingkatkan secara efektif, sehingga deteksi pelat nomor tetap akurat meskipun kendaraan berada pada posisi atau sudut pandang yang tidak ideal.

(a)

(b)

(c)

Gambar 14. Contoh hasil perubahan *shear*: (a) sebelum perubahan *shear*; (b) setelah perubahan *shear* 10° horizontal; (c) setelah perubahan *shear* 10° vertikal;

Brightness (± 15%): *Brightness augmentation* menerapkan perubahan kecerahan citra secara acak dalam rentang -15% hingga +15% dari nilai original. Teknik ini mensimulasikan variasi kondisi pencahayaan yang terjadi sepanjang hari, mulai dari cahaya pagi yang terang, cahaya siang yang intens, hingga cahaya sore yang redup atau bayangan dari struktur bangunan. CCTV area parkir yang beroperasi 24 jam akan merekam objek dalam berbagai kondisi pencahayaan, dan model harus mampu mendeteksi pelat nomor dengan andal di semua kondisi ini. Dengan augmentasi *brightness*, model belajar untuk invarian terhadap perubahan pencahayaan, sehingga performa deteksi tetap konsisten di berbagai waktu.

(a)

(b)

(c)

Gambar 15. Contoh hasil perubahan *brightness*: (a) sebelum perubahan *brightness*; (b) setelah perubahan *brightness* -15%; (c) setelah perubahan *brightness* +15%;

Exposure (± 15%): *Exposure* mengubah *exposure* level citra (menyesuaikan kurva *tone mapping*) dalam rentang -15% hingga +15%. Berbeda dengan *brightness* yang mengubah tingkat kecerahan secara linear, *exposure* bertujuan untuk mensimulasikan perbedaan dalam *exposure time* kamera atau pengaturan ISO sensor. Pada praktiknya, kamera CCTV yang berbeda atau konfigurasi kamera yang berubah dapat menghasilkan citra dengan *exposure* yang berbeda-beda, beberapa tampak *overexposed* (terang berlebihan) atau *underexposed* (gelap berlebihan). Augmentasi ini memastikan model dapat menangani variasi *exposure* ini dan tetap mendeteksi pelat nomor dengan baik bahkan pada citra yang *underexposed* atau *overexposed*.

(a)

(b)

(c)

Gambar 16. Contoh hasil perubahan *exposure*: (a) sebelum perubahan *exposure*; (b) setelah perubahan *exposure* -15%; (c) setelah perubahan *exposure* +15%;

Blur (up to 1 px): Augmentasi *blur* menerapkan *Gaussian blur* dengan kernel kecil (hingga 1 piksel) secara acak pada beberapa citra. Dalam dunia real, *motion blur* terjadi ketika kendaraan bergerak cepat atau kamera memiliki *shutter speed* yang terlalu lambat, menghasilkan citra yang sedikit buram atau kabur. *Blur* juga dapat berasal dari lensa kamera yang kotor atau *autofocus* yang tidak sempurna. Dengan melatih model pada citra yang mungkin buram, model dapat belajar fitur yang lebih *robust* dan tidak terlalu bergantung pada detail frekuensi tinggi yang hilang pada citra buram. Hal ini penting karena di lapangan, tidak semua *frame* akan memiliki kualitas ketajaman yang sempurna, dan model tetap harus mampu mendeteksi pelat nomor dengan andal.

(a)

(b)

Gambar 17. Contoh hasil perubahan blur: (a) sebelum perubahan blur; (b) setelah perubahan blur;

Noise (up to 0.85%): Augmentasi *Noise* menambahkan *Gaussian noise* dengan level hingga 0.85% dari nilai piksel. *Noise* simulasi ini mewakili sensor 'bising' yang umum terjadi pada CCTV, terutama pada kondisi cahaya rendah atau pada sensor yang kurang baik. *Gaussian noise* dengan level kecil (0.85% atau ~2,17 dari 255) menciptakan gangguan *granular* pada citra yang menyerupai karakteristik *noise* pada kasus nyata. Dengan mengekspos model pada data yang telah ditambahkan *noise*, model belajar untuk ekstrak fitur yang penting dan mengabaikan *noise spurious*, sehingga meningkatkan ketahanan model terhadap sensor *noise* yang akan dijumpai pada kamera CCTV di lapangan sesungguhnya.

(a) (b)

Gambar 18. Contoh hasil perubahan *noise*: (a) sebelum perubahan *noise*; (b) setelah perubahan *noise*;

Hasil augmentasi menghasilkan dataset *training* yang lebih besar dan lebih beragam. Dataset 838 citra *original* setelah augmentasi ekspansi menjadi 2.514 citra (rasio augmentasi ~3x), memastikan model menerima data yang sangat variatif di setiap *epoch* dan mengurangi risiko *overfitting* pada pola spesifik dalam dataset *original*.

2.5.1.3 Hyperparameter Training

Proses pelatihan model YOLOv11 dilakukan pada platform kaggle dengan menggunakan akselerator 2× NVIDIA Tesla T4 GPU (total 30 VRAM) dengan konfigurasi *hyperparameter* sebagai berikut:

Tabel 1. Konfigurasi *hyperparameter training* model

Parameter	Nilai
Varian Model	YOLOv11n, YOLOv11s, YOLOv11m,
Optimizer	SGD
Learning Rate	0.01

Epoch	300
Batch Size	32

Pemilihan *hyperparameter* dijelaskan sebagai berikut:

Optimizer SGD (*Stochastic Gradient Descent*) dipilih karena SGD menghasilkan model dengan generalisasi yang lebih baik pada testing set dibandingkan *optimizer* adaptif seperti Adam (Wilson et al., 2017).

Learning rate 0.01 dipilih sesuai dengan rekomendasi dari dokumentasi resmi (*Model Training with Ultralytics YOLO - Ultralytics YOLO Docs*, n.d.) untuk *optimizer* SGD pada model-model YOLO. Pemilihan ini didukung oleh penelitian empiris (Choi et al., 2020) yang menunjukkan bahwa dengan *hyperparameter tuning* yang tepat, SGD dengan *learning rate* 0.01 dapat mencapai performa yang kompetitif atau lebih baik dari Adam pada berbagai tugas *computer vision*.

Epoch 300 dipilih untuk memberikan waktu yang cukup pada model untuk mencapai konvergensi stabil.

Batch size 32 dipilih berdasarkan kapasitas GPU yang tersedia (12 GB) dan merupakan ukuran *batch* standar untuk *training* model deteksi objek.

2.5.1.4 Hasil Training

Hasil *training* ketiga varian model pada *validation set* disajikan pada Tabel 2 berikut

Tabel 2. Hasil evaluasi model pada data validasi

Model	*Precision*	*Recall*	mAP50	mAP50-95
YOLOv11n	0.972	0.984	0.991	0.941
YOLOv11s	0.980	0.985	0.991	0.950
YOLOv11m	0.978	0.981	0.988	0.947

Hasil *training* menunjukkan bahwa ketiga varian model YOLOv11 telah mencapai akurasi deteksi yang sangat tinggi pada *validation set*. YOLOv11s mencatat performa terbaik pada metrik mAP50-95 dengan nilai 0.9504, diikuti YOLOv11m (0.9470) dan YOLOv11n (0.9418). Pada metrik mAP50, perbedaan antar model sangat minimal dengan YOLOv11n mencapai 0.9917, YOLOv11s mencapai 0.9914, dan YOLOv11m mencapai 0.9881.

Precision dan *recall* ketiga model juga menunjukkan nilai yang sangat tinggi (>0.97 dan >0.98), mengindikasikan bahwa model-model ini telah mencapai keseimbangan yang sempurna antara mendeteksi semua kelas yang ada dan menghindari *false positives*. Semua varian model telah menunjukkan kemampuan yang *robust* dalam mendeteksi objek pada *validation set* dan memenuhi persyaratan akurasi untuk tahap selanjutnya yaitu optimalisasi model.

2.5.2 Tahap Optimalisasi Model

Meskipun ketiga varian model YOLOv11 telah mencapai akurasi deteksi yang tinggi pada tahap *training*, efisiensi komputasi model masih perlu ditingkatkan untuk memungkinkan *deployment* pada perangkat dengan daya komputasi terbatas, seperti sistem yang ditempatkan di area parkir atau server dengan daya pemrosesan terbatas. Tahap optimalisasi model bertujuan untuk mengurangi ukuran model dan mempercepat waktu inferensi sambil mempertahankan akurasi deteksi. Strategi optimalisasi yang dipilih adalah kuantisasi model dari presisi *floating-point* 32-bit (FP32) ke integer 8-bit (INT8), sebuah teknik yang terbukti efektif dalam mengurangi ukuran model dan mempercepat inferensi tanpa menghasilkan degradasi akurasi yang signifikan.

2.5.2.1 Eksplorasi Awal: Unstructured Pruning

Pada tahap awal penelitian, metode *pruning* sempat dipertimbangkan sebagai salah satu teknik optimalisasi model. *Pruning* bekerja dengan menghilangkan bobot (*weights*) yang memiliki *magnitude* kecil sehingga dianggap kurang berkontribusi terhadap prediksi model. Teknik ini secara teori dapat mengurangi jumlah parameter dan mempercepat inferensi tanpa degradasi akurasi yang signifikan.

Eksperimen dilakukan menggunakan teknik *unstructured L1 pruning* pada seluruh lapisan konvolusi (Conv2d) dengan rasio pruning 30%. Metode ini diimplementasikan menggunakan modul *torch.nn.utils.prune* dari PyTorch. Setelah proses *pruning*, model kemudian di *fine-tune* selama 50 *epoch* untuk mencoba memulihkan performa yang hilang akibat penghapusan bobot.

Hasil eksperimen menunjukkan degradasi performa yang sangat parah. Tabel 3 berikut menyajikan perbandingan performa model sebelum dan sesudah *pruning*.

Tabel 3. Tabel hasil *unstructured pruning*

Model	mAP50-95 awal	mAP50-95 Pruned	mAP50-95 Pruned (*fine-tuned*)
YOLOv11n	0.941	0.025	0.030
YOLOv11s	0.952	0.025	0.016
YOLOv11m	0.962	0.025	0.019

Hasil eksperimen menunjukkan bahwa proses *fine-tuning* selama 50 *epoch* tidak mampu memulihkan performa model yang telah di-*prune*. Nilai mAP50-95 tetap stagnan di kisaran 0.02–0.04, jauh dari performa model asli yang mencapai 0.94–0.96. Bahkan deteksi pada kelas *license_plate* menghasilkan mAP50-95 sebesar 0 (nol), menunjukkan bahwa model kehilangan kemampuan krusial untuk mendeteksi objek kecil yang menjadi inti dari sistem ini.

Kegagalan ini terutama disebabkan oleh sensitivitas arsitektur YOLOv11, yang mengandalkan struktur blok C3k2 dan mekanisme *spatial attention* (C2PSA) yang kompleks. Penghapusan bobot secara acak (*unstructured*) merusak integritas representasi fitur pada layer-layer tersebut, di mana bobot bermagnitude kecil ternyata tetap memegang peran vital. Selain itu, *unstructured pruning* hanya menghasilkan *sparse tensor* yang sulit dioptimalkan oleh perangkat keras modern tanpa pengurangan dimensi komputasi yang nyata, sehingga tidak memberikan manfaat efisiensi yang sebanding dengan drastisnya penurunan akurasi. Berdasarkan temuan ini, metode *unstructured pruning* tidak dilanjutkan pada penelitian ini.

2.5.2.2 Eksplorasi Lanjutan: Structured Pruning

Selain *unstructured pruning*, penelitian ini juga mengeksplorasi metode *structured pruning* yang bertujuan untuk menghilangkan keseluruhan filter atau *channel* dari jaringan, sehingga dapat memberikan percepatan inferensi yang nyata pada perangkat keras standar tanpa memerlukan dukungan *sparse matrix accelerator*.

Implementasi dilakukan menggunakan *library torch-pruning* dengan strategi *dependency-aware pruning* untuk menjaga konsistensi konektivitas antar-layer. Target pengurangan parameter ditetapkan sebesar 30%. Namun, eksperimen ini menghadapi kendala arsitektural yang signifikan. Arsitektur YOLOv11 banyak memanfaatkan *Depthwise Separable Convolutions* dan *Grouped Convolutions* (terutama pada blok C3k2 dan modul C2PSA) di mana parameter *groups* pada lapisan konvolusi terikat erat dengan jumlah *channel input* dan *output*.

Dalam praktiknya, percobaan *structured pruning* pada arsitektur YOLOv11 tidak menghasilkan pengurangan parameter yang berarti (0.0% *reduction*), meskipun sudah dilakukan beberapa iterasi *pruning*. Hal ini mengindikasikan bahwa mekanisme *dependency graph* pada *library* yang digunakan cenderung mempertahankan *channel* pada layer-layer dengan *grouped* dan *depthwise convolution* demi menjaga konsistensi dimensi *tensor* di seluruh jalur komputasi. Oleh karena itu, *structured pruning* dinilai kurang kompatibel untuk arsitektur YOLOv11 tanpa modifikasi arsitektur yang lebih radikal.

2.5.2.3 Pemilihan Metode Akhir: Post-Training Quantization

Berdasarkan hasil eksperimen *unstructured pruning* dan *structured pruning* yang tidak memberikan hasil memuaskan, penelitian beralih ke metode optimalisasi yang lebih sesuai dengan arsitektur YOLOv11 modern. Metode yang dipilih adalah *Post-Training Quantization* (PTQ), sebuah teknik yang berbeda fundamental dari *pruning* dalam pendekatan optimalnya.

Quantization atau kuantisasi adalah proses konversi nilai-nilai parameter dan *aktivasi* dalam model *neural network* dari representasi presisi tinggi (FP32, 32-bit *floating point*) ke representasi presisi lebih rendah (INT8, 8-bit *integer*). Proses ini mengurangi ukuran memori model secara signifikan dan mempercepat inferensi

karena operasi integer 8-bit dapat dieksekusi jauh lebih cepat dibandingkan operasi *floating-point* 32-bit pada sebagian besar CPU modern. Berbeda dengan *pruning* yang membuang parameter sama sekali, kuantisasi mempertahankan semua parameter tetapi dengan presisi numerik yang lebih rendah, sehingga struktur arsitektur tetap utuh dan kompatibel dengan berbagai *hardware*.

Strategi kuantisasi yang diterapkan dalam penelitian ini adalah *Static Post-Training Quantization* (PTQ) dengan format QDQ (*Quantize-DeQuantize*) menggunakan pendekatan *Selective Quantization*. Pendekatan ini dirancang untuk memaksimalkan rasio kompresi model dengan menerapkan kuantisasi pada mayoritas lapisan ekstraksi fitur, namun secara selektif mempertahankan *node* spesifik yang vital terhadap akurasi dalam format aslinya.

Langkah krusial dalam metode ini adalah pemisahan antara lapisan *robust* yang dikonversi ke format INT8, dan lapisan sensitif yang dipertahankan dalam format FP32. Berdasarkan analisis graf komputasi model, beberapa komponen diidentifikasi sangat rentan terhadap *quantization noise* dan wajib dikecualikan dari proses kuantisasi.

Komponen sensitif yang dipertahankan dalam presisi penuh (FP32) meliputi:

1. **Detection Head (Layer 19–23):** Merupakan lapisan akhir arsitektur yang bertugas melakukan prediksi koordinat *bounding box* dan klasifikasi objek. Penelitian ini secara eksplisit mengecualikan rentang lapisan ini (model.19 hingga model.23) karena kuantisasi pada tahap ini berisiko tinggi menyebabkan penyimpangan koordinat deteksi yang signifikan.
2. **Lapisan Transisi dan Attention (Layer 10):** Lapisan menengah (model.10) diidentifikasi sebagai *node* krusial, yaitu blok C2PSA yang menjembatani fitur antar-*stage*. Mempertahankan lapisan ini dalam FP32 bertujuan menjaga integritas fitur sebelum didistribusikan ke bagian *Neck* dan *Head*.

Metode *selective quantization* dipilih dibandingkan *full quantization* (di mana semua lapisan di-kuantisasi) karena arsitektur YOLOv11 mengandung lapisan-lapisan *detection head* yang sangat sensitif terhadap kehilangan presisi. Pengujian awal dengan *full quantization* menghasilkan penurunan akurasi sangat buruk (mAP50-95 turun dari 0.9418 menjadi 0.00), di mana model tidak dapat menghasilkan deteksi yang valid. Hal ini terjadi karena *error* akumulatif dari *quantization* pada lapisan-lapisan krusial seperti *Softmax*, *MatMul*, dan *coordinate transformation* merambat melalui *pipeline* dan menyebabkan *numerical instability*. Dengan *selective quantization*, lapisan-lapisan kritis dipertahankan dalam presisi FP32, memastikan stabilitas numerik dan mempertahankan akurasi sambil tetap mencapai rasio kompresi yang signifikan.

2.5.2.4 Proses Optimalisasi

Proses kuantisasi dilakukan melalui beberapa tahapan. Tahapan pertama adalah ekspor PyTorch (.pt) ke ONNX FP32 menggunakan utilitas ekspor Ultralytics. Proses ini mengonversi operasi-operasi pada PyTorch menjadi operasi ONNX yang standar dan kompatibel lintas platform. Format ONNX memungkinkan model untuk dioptimalkan serta dijalankan menggunakan berbagai *framework* dan *runtime*. Model hasil optimalisasi kemudian disimpan dalam format ONNX untuk kebutuhan inferensi menggunakan ONNX *Runtime*. Konversi balik ONNX ke PyTorch tidak dibahas lebih lanjut dikarenakan ONNX digunakan sebagai format *deployment* atau inferensi, serta proses konversi lintas format tidak selalu terjamin ekuivalensinya pada implementasi yang kompleks

Setelah model tersedia dalam format ONNX FP32, proses kuantisasi dilakukan dengan langkah-langkah berikut:

1. **Persiapan Dataset Kalibrasi:** Subset 500 citra dari set *training* dipilih secara acak untuk digunakan sebagai data kalibrasi. Setiap citra diproses sebelumnya dengan cara yang sama seperti saat inferensi (ubah ukuran menjadi 640×640, normalisasi nilai piksel).
2. **Jalankan Kalibrasi:** Model ONNX FP32 dijalankan pada seluruh dataset kalibrasi untuk mengumpulkan statistik tentang rentang dan distribusi dari aktivasi di setiap lapisan. Statistik ini digunakan untuk mempelajari pemetaan yang optimal antara nilai FP32 dan nilai INT8.
3. **Tentukan Lapisan Sensitif:** Lapisan-lapisan yang sensitif terhadap kuantisasi diidentifikasi dan dikecualikan dari proses kuantisasi berdasarkan analisis arsitektur model YOLOv11. Analisis ini menunjukkan bahwa 84 lapisan pada *Detection Head* (model.19–23) sangat sensitif terhadap pengurangan presisi numerik, terutama pada operasi kritis seperti *Softmax* untuk perhitungan probabilitas, *MatMul* untuk transformasi koordinat *bounding box*, dan fungsi *Sigmoid* untuk *confidence score*. Selain itu, layer C2PSA pada *Neck* (model.10) juga dikecualikan karena merupakan modul *Attention* yang memerlukan presisi tinggi dalam menjembatani fitur antar-*stage*. Sementara itu, lapisan-lapisan lain yang lebih *robust* terutama lapisan *convolutional* pada *Backbone* dan sebagian besar *Neck* dipilih untuk di kuantisasi ke INT8 tanpa mengorbankan akurasi secara signifikan.
4. **Buat Parameter Kuantisasi:** Berdasarkan statistik yang dikumpulkan, parameter kuantisasi (faktor skala dan titik nol) dihitung untuk setiap lapisan yang akan di kuantisasi, memastikan bahwa rentang FP32 dipetakan secara optimal ke rentang INT8 [-128, 127].
5. **Konversi ke INT8:** Model ONNX FP32 dikonversi ke ONNX INT8 dengan mempertahankan lapisan-lapisan sensitif dalam presisi FP32. Hasilnya adalah model ONNX INT8 *hybrid* (sebagian INT8, sebagian FP32) yang siap untuk inferensi.

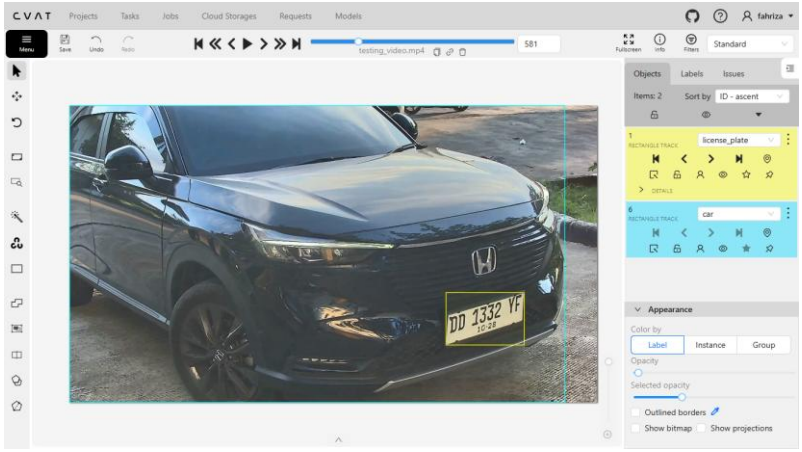
Setelah proses kuantisasi selesai, model INT8 hasil kuantisasi kemudian divalidasi pada set validasi untuk memastikan bahwa degradasi akurasi tetap berada dalam batas yang dapat diterima (target < 5% penurunan akurasi). Selain itu, *benchmark* dilakukan untuk mengukur pengurangan dalam ukuran model dan peningkatan dalam kecepatan inferensi dibandingkan dengan model FP32 orisinal.

2.5.3 Tahap Persiapan Dataset Pengujian

2.5.3.1 Dataset Pengujian

Dataset pengujian terdiri dari citra pelat nomor dan kendaraan yang tidak pernah dilihat oleh model selama proses *training* maupun validasi. Dataset ini dikumpulkan dari rekaman video CCTV area parkir pada periode waktu yang berbeda dan mencakup variasi kendaraan yang tidak terdapat dalam set *training*.

Proses anotasi (*labeling*) untuk dataset testing dilakukan menggunakan CVAT (*Computer Vision Annotation Tool*), sebuah platform *open-source* yang menyediakan *interface* untuk anotasi objek dalam citra dan video dengan fitur-fitur seperti *semi-automatic annotation*, *support* untuk *multiple object classes*, dan *export* format yang kompatibel dengan YOLO.



Gambar 19. Proses pelabelan data menggunakan platform CVAT

Dataset testing ini menjadi indikator utama generalisasi model terhadap data baru yang akan dijumpai pada *deployment* nyata sistem parkir cerdas.

2.5.3.2 Skenario Pengujian

Pengujian sistem dilakukan pada tiga skenario lalu lintas yang merepresentasikan kondisi berbeda di area parkir. Setiap skenario dirancang dengan komposisi kendaraan dengan durasi yang berbeda untuk mengevaluasi *robustness* sistem dalam berbagai kondisi operasional.

Skenario 1 (Kendaraan Sepi): dirancang sebagai *baseline* pengujian dengan kompleksitas rendah di mana kendaraan memasuki area parkir secara individual dengan jarak dan waktu terpisah, sehingga oklusi minimal dan trajektori kendaraan jelas. Skenario ini memiliki durasi video sepanjang 1 menit 23 detik dengan *frame rate* 30 FPS dan total 2.503 *frame*, mencakup pengujian pada 5 kendaraan yang terdiri dari 2 mobil (*car*) dan 3 motor (*motorcycle*) dengan total 5 pelat nomor (*license plate*) untuk mengevaluasi kemampuan model pada kondisi ideal.

Skenario 2 (Kendaraan Antri): fokus menguji kemampuan sistem dalam menangani oklusi parsial dan kedekatan objek (*object proximity*) yang tinggi pada satu jalur linear. Pada skenario ini, pengujian dilakukan menggunakan video berdurasi 51 detik dengan *frame rate* 30 FPS dan total 1.535 *frame* yang menampilkan 4 mobil (*car*) dengan total 4 pelat nomor (*license plate*) yang mengantre dalam jarak dekat. Kondisi ini memberikan tantangan spesifik bagi sistem dalam membedakan kendaraan yang berimpitan dan membaca pelat nomor yang mungkin sebagian terhalang oleh kendaraan di depannya.

Skenario 3 (Kendaraan Paralel): merupakan skenario pengujian paling kompleks karena kendaraan bergerak berdampingan di jalur berbeda, menyebabkan pelat nomor dapat berdekatan atau saling *overlapping* dari perspektif kamera. Durasi video pengujian adalah 15 detik dengan *frame rate* 30 FPS dan total 468 *frame*, skenario ini sangat krusial untuk menguji fitur diskriminatif sistem dalam mempertahankan identitas objek. Pengujian ini menggunakan 3 motor (*motorcycle*) dengan total 3 pelat nomor (*license plate*) untuk mengevaluasi performa model dalam kondisi pergerakan dinamis yang menantang.

Skenario 4 (Monitor Petak Parkir): berfokus pada pemantauan ketersediaan petak pada lahan parkir dan pencatatan aktivitas parkir secara otomatis. Pada skenario ini, sistem tidak hanya melakukan deteksi dan pembacaan pelat nomor kendaraan, tetapi juga memonitor setiap petak parkir untuk menentukan kapan sebuah kendaraan mulai mengisi petak dan kapan kendaraan tersebut meninggalkan petak. Informasi tersebut digunakan untuk mencatat pelat kendaraan yang mengisi petak serta menghitung durasi kendaraan menempati petak parkir sebagai indikator waktu parkir. Video pengujian pada skenario ini berdurasi 3 menit 7 detik dengan *frame rate* 30 FPS dan total 5.629 *frame.* Total petak parkir yang dipantau dalam pengujian adalah 3 petak, dengan kendaraan uji sebanyak 3 mobil (*car*) dan 1 motor (*motorcycle*) dengan total 4 pelat nomor (*license plate*) yang menjadi objek pengujian.

2.5.4 Tahap Pengujian Performa Model Deteksi

Setelah proses *training* dan optimalisasi model selesai, tahap selanjutnya adalah melakukan evaluasi komprehensif terhadap performa model dalam mendeteksi objek pada citra kendaraan dan pelat nomor. Pengujian dilakukan pada model FP32 (*baseline*) dan model INT8 (hasil kuantisasi) untuk ketiga varian YOLOv11, dengan tujuan untuk mengukur akurasi deteksi dan efisiensi komputasi.

2.5.4.1 Environment Pengujian

Seluruh pengujian performa model dilakukan pada unit pemrosesan pusat (CPU) untuk mengevaluasi performa model pada perangkat dengan daya komputasi terbatas. Spesifikasi perangkat keras yang digunakan untuk pengujian adalah sebagai berikut:

- Perangkat: Lenovo V14 ADA
- *Processor*: AMD Ryzen 3 3250U (2 *cores*, 4 *threads*, *base clock* 2.6 GHz)
- RAM: 12 GB DDR4
- *Storage*: SSD 256 GB
- Sistem Operasi: Windows 11

2.5.4.2 Metrik Evaluasi Performa Deteksi

Performa deteksi model dievaluasi menggunakan metrik standar untuk tugas deteksi objek, mencakup mAP50, mAP50-95, *Precision*, *Recall*, *Inference Time* (ms), dan *Model Size* (MB). Kombinasi metrik ini memberikan gambaran komprehensif tentang trade-off antara performa deteksi dan kebutuhan sumber daya komputasi.

2.5.5 Tahap Pengujian Performa Tracking

Tahap pengujian *tracking* atau pelacakan bertujuan untuk mengevaluasi kemampuan sistem dalam melacak (*tracking*) objek khususnya pelat nomor kendaraan (*license_plate*) di sepanjang *sequence* video. Pengujian dilakukan menggunakan tiga skenario lalu lintas yang berbeda untuk mensimulasikan berbagai kondisi operasional yang mungkin dijumpai pada sistem parkir cerdas riil.

Tracking model hanya ditargetkan pada satu kelas objek, yaitu *license_plate*, karena fokus penelitian adalah pada pencatatan nomor pelat kendaraan untuk log data parkir. Setiap pelat nomor yang terdeteksi harus diberi identitas unik (ID) yang konsisten di sepanjang *sequence* video, yang memungkinkan sistem untuk melacak perjalanan satu pelat dari *frame* ke *frame* tanpa kehilangan atau melakukan penukaran identitas.

2.5.5.1 Algoritma Tracking

Tracking dilakukan menggunakan ByteTrack, sebuah *tracker* berbasis deteksi yang terintegrasi dengan *pipeline* YOLOv11. ByteTrack menggunakan pendekatan *data association* berbasis *distance metrics* (*IoU-based*) untuk menghubungkan deteksi pada *frame* berurutan dengan identitas yang konsisten. Penghubungan deteksi

dilakukan dengan strategi *multi-hypothesis tracking* yang mempertimbangkan *spatial proximity* dan *motion* model untuk memprediksi posisi objek pada *frame* berikutnya.

2.5.5.2 Metrik Evaluasi Performa Tracking

Performa *tracking* dievaluasi menggunakan metrik standar untuk *multi-object tracking* mencakup MOTA, MOTP, IDF1, FP, FN, dan *ID Switches*.

2.5.6 Desain Alur Sistem (End-to-End)

2.5.6.1 Alur Sistem Portal Parkir

Pada alur sistem portal parkir, video *input* diproses secara *real-time* dengan cara membaca *frame* secara berurutan untuk mensimulasikan kondisi operasional sistem pada CCTV. Setiap *frame* yang masuk akan melewati rangkaian pemrosesan *end-to-end* yang terintegrasi, mulai dari deteksi dan pelacakan objek hingga pembacaan teks pelat nomor dan pencatatan hasil akhir. Rangkaian tahapan tersebut disusun agar sistem mampu mempertahankan identitas objek secara konsisten antar-*frame*, meminimalkan kesalahan pembacaan akibat *noise* sesaat, serta menghasilkan keluaran berupa pelat nomor yang siap dicatat ke dalam log. Adapun alur sistem portal parkir sebagai berikut:

1. Deteksi Objek dan Pelacakan (YOLOv11 - ByteTrack)
   Setiap *frame* diumpankan ke model YOLOv11 yang terintegrasi dengan algoritma pelacakan ByteTrack. *Output* tahap ini berupa daftar deteksi dengan informasi label kelas, koordinat *bounding box*, serta *tracking ID* mentah untuk setiap objek yang terdeteksi.
2. Penyaringan Kelas dan Normalisasi ID (*Canonical ID*)
   Hanya deteksi dengan label kelas *license_plate* yang dipertahankan. Untuk mengurangi kasus *identity switch*, sistem menerapkan skema *Canonical ID (cid)* sebagai ID utama yang stabil. Jika muncul *tracking ID* baru yang secara spasial sangat mirip dengan *track* lama, ID baru tersebut akan dipetakan kembali ke cid lama untuk menjaga kontinuitas pelacakan..
3. Pemfilteran Spasial dengan *Region of Interest* (ROI)
   Sistem mendefinisikan sebuah *Region of Interest* (ROI) di dalam *frame*. Hanya pelat dengan cid yang terdeteksi berada di dalam ROI yang akan diproses lebih lanjut untuk pembacaan teks.
4. Pemotongan Citra Pelat (*Cropping*)
   Untuk setiap pelat aktif yang berada di dalam ROI, sistem melakukan *cropping* citra berdasarkan koordinat *bounding box* dengan menambahkan margin.
5. Ekstraksi dan Normalisasi Teks OCR
   Citra *crop* pelat diumpankan ke model PaddleOCR. Teks mentah hasil OCR kemudian dibersihkan dan diformat menggunakan *regular expression* untuk menyusun kembali pola teks ke format standar pelat nomor Indonesia.

6. Akumulasi Sampel dan Voting Konsensus
   Selama sebuah cid masih berada di dalam ROI, semua hasil pembacaan OCR yang valid akan dikumpulkan. Setelah cid keluar dari ROI, mekanisme voting konsensus diterapkan untuk menentukan teks pelat final.
7. Pencatatan Log Hasil Sistem
   Hasil akhir pembacaan pelat nomor untuk setiap cid kemudian dicatat ke dalam berkas log.

2.5.6.2 Alur Sistem Monitor Petak Parkir

Pada alur sistem monitor petak parkir, video *input* diproses secara *real-time* dengan membaca *frame* secara berurutan untuk mensimulasikan pengawasan kontinu area parkir oleh CCTV. Setiap *frame* digunakan untuk memperbarui status masing-masing petak parkir berdasarkan keberadaan kendaraan di dalam ROI petak, melakukan pembacaan pelat nomor kendaraan, serta menghitung durasi kendaraan yang menempati petak tersebut. Rangkaian tahapan ini dirancang agar sistem dapat menjaga konsistensi identitas kendaraan pada tiap petak, menghindari deteksi terisi/kosong yang bersifat sementara, dan menghasilkan catatan *event* parkir yang berisi pelat nomor, jenis kendaraan, waktu masuk-keluar, serta durasi parkir yang siap disimpan ke dalam log. Adapun alur sistem monitor petak parkir sebagai berikut:

1. Deteksi dan Pelacakan Kendaraan (YOLOv11 – ByteTrack)
   Setiap *frame* diumpankan ke model YOLOv11 dengan *tracking* untuk menghasilkan daftar objek terdeteksi beserta koordinat *bounding box*, label kelas, dan *tracking* ID. Dari keluaran ini, sistem memisahkan data menjadi dua, yaitu deteksi kendaraan (*car/motorcycle*) untuk pemantauan okupansi petak, dan pelat nomor (*license_plate*) sebagai pemegang ID utama serta untuk kebutuhan OCR.
2. Normalisasi Identitas Kendaraan (*Canonical ID*)
   Saat pelat kendaraan baru terdeteksi dalam frame, data pelat hasil *tracking* tersebut kemudian dinormalisasikan menjadi *Canonical ID* (cid) agar identitas kendaraan lebih stabil antar-*frame*. Ketika muncul *tracking ID* baru yang secara spasial mirip dengan *track* sebelumnya (berdasarkan IoU dan jarak *centroid*), ID tersebut akan dipetakan ke cid yang sudah ada sehingga identitas kendaraan tetap konsisten walaupun *tracking ID* mentah telah berubah.
3. Asosiasi Pelat ke kendaraan Terdekat
   Untuk setiap pelat yang sudah memiliki cid, sistem mencari kendaraan terdekat (*car*/*motorcycle*) yang paling relevan dengan pelat tersebut dengan mempertimbangkan *overlap* minimal (30%) antara *bounding box* pelat dan kendaraan, lalu memilih kandidat dengan jarak terdekat terhadap titik referensi kendaraan, yaitu jarak antara pusat *bounding box* pelat (*plate center*) dan titik *bottom-center* dari *bounding box* kendaraan. Kendaraan dengan jarak terkecil dipilih sebagai pasangan relevan untuk pelat tersebut. Hasil tahap ini adalah pasangan data cid

pelat, *bounding box* kendaraan, dan jenis kendaraan yang menjadi dasar pemantauan petak.

4. Pemfilteran Spasial dengan *Region of Interest* (ROI Petak Parkir)
   Setiap petak parkir yang ada, didefinisikan sebagai ROI poligon, kemudian sistem menghitung *overlap* antara *bounding box* kendaraan dengan ROI petak. Pelat (cid) yang kendaraan asosiasinya memiliki *overlap* melewati ambang *threshold* (25%), akan dianggap sebagai kandidat yang sedang menempati petak tersebut.
5. Penentuan Status Petak dan Stabilitas Okupansi
   Ketika ada kandidat kendaraan yang berada pada ROI, petak tidak langsung dianggap terisi, tetapi kandidat harus terdeteksi secara kontinu selama durasi minimum yang ditetapkan (2.5 detik) agar bisa dianggap menempati petak dan status berubah menjadi *occupied*.

   Saat status petak sudah *occupied*, sistem akan mempertahankan identitas kendaraan/pelat yang tercatat pada petak tersebut agar tidak mudah berubah akibat oklusi atau ketidakstabilan deteksi. Sistem akan membandingkan posisi kandidat yang terlihat sekarang dengan posisi terakhir objek yang menempati petak menggunakan tingkat *overlap bounding box* (ioU). Selama nilai IoU masih berada di atas ambang batas tertentu (≥ 50%), petak dianggap masih terisi oleh kendaraan yang sama dan cid-nya dipertahankan.

   CID akan dianggap “keluar” jika objek yang menempati petak tidak lagi terdeteksi secara konsisten selama minimal 5 detik, kemudian sistem akan menganggap *event* parkir selesai dan status petak akan di finalisasi menjadi kosong. Untuk mengantisipasi kejadian di mana cid menghilang cukup lama sehingga dianggap keluar karena kasus tertentu seperti terhalang kendaraan lain, sistem juga menerapkan logika *plate based merge,* di mana ketika ada cid baru dengan pembacaan pelat yang sama dengan cid yang telah tercatat sebelumnya, selama masih berada dalam rentang waktu yang ditentukan (60 detik), sistem akan menganggap kedua cid ini adalah cid yang sama, dan menggabungkan cid dan durasi parkir kendaraan yang telah ada dengan yang baru.
6. Pemotongan Cita Pelat dan Ekstraksi Teks (PaddleOCR)
   Selama petak dalam status *occupied* dan cid pelat masih konsisten, sistem melakukan sampling OCR secara periodik. Citra pelat dipotong berdasarkan *bounding box* pelat dengan penambahan margin 10 piksel, kemudian diproses menggunakan PaddleOCR untuk menghasilkan teks mentah (*raw*) yang selanjutnya akan dinormalisasi ke format pelat Indonesia.
7. Akumulasi Sampel, Voting Konsensus, dan Pencatatan Log
   Selama petak masih terisi, sistem akan mengumpulkan hasil OCR yang valid sebagai sampel hingga batas maksimum 50 sampel per kejadian parkir. Ketika petak dinyatakan kosong, sistem melakukan voting konsensus dari kumpulan hasil OCR yang sudah di normalisasi untuk menentukan pelat final, dengan catatan jumlah sampel yang terkumpul

mencapai minimum 5 sampel. Hasil akhir kemudian dicatat ke dalam log kejadian parkir dengan data kendaraan lainnya (nomor petak, CID, Pelat Final, Jenis Kendaraan, Waktu masuk dan keluar, dan durasi parkir).

2.5.6.3 Integrasi OCR & Mekanisme Voting Konsensus Pada Sistem

Tahap integrasi OCR bertujuan untuk menghubungkan modul deteksi dan pelacakan YOLOv11-ByteTrack dengan modul pembacaan teks pelat nomor menggunakan PaddleOCR. Integrasi ini memungkinkan sistem untuk secara otomatis membaca karakter pada pelat nomor yang telah terdeteksi dan dilacak, kemudian menghasilkan teks pelat final yang telah dinormalisasi dan diverifikasi melalui mekanisme voting konsensus.

Ekstraksi dan Normalisasi Teks. Proses pembacaan teks pelat nomor dilakukan melalui beberapa tahap pemrosesan:

1. Ekstraksi Teks dari Potongan Citra Pelat
   Setelah sistem melakukan *cropping* pada region pelat yang berada di dalam ROI, potongan citra diumpankan ke model PaddleOCR. *Engine* OCR akan mendeteksi region teks dalam citra dan melakukan *recognition* untuk mengekstrak karakter-karakter yang terbaca. Hasil *raw* OCR berupa list teks yang terdeteksi dari berbagai region dalam citra *crop*.

   Untuk memastikan urutan pembacaan yang benar (kiri ke kanan), sistem menerapkan mekanisme *ordering* berdasarkan posisi *x-coordinate* dari setiap *text region*. *Text regions* diurutkan berdasarkan posisi horizontal (*x-coordinate*) sebelum digabungkan menjadi satu *string*, sehingga karakter-karakter pelat nomor terbaca dalam urutan yang tepat.
2. Format dan Normalisasi Pelat Nomor Indonesia
   Teks *raw* hasil OCR kemudian diproses melalui fungsi *formatting* yang dirancang khusus untuk format pelat nomor Indonesia. Proses ini diawali dengan tahapan tokenisasi, di mana teks *raw* diubah menjadi huruf kapital dan dipisahkan menjadi token-token dengan membuang karakter non-alfanumerik. Selanjutnya, sistem melakukan ekstraksi nomor pelat dengan mencari urutan angka 3-4 digit sebagai nomor utama pelat menggunakan *regular expression*.

   Setelah nomor pelat ditemukan, sistem mengekstrak prefix (kode wilayah) dengan mengambil dua huruf terakhir dari token-token sebelum nomor pelat. Langkah terakhir adalah ekstraksi dan koreksi suffix, di mana sistem mengambil hingga tiga huruf dari token-token setelah nomor pelat, dengan menerapkan *character correction mapping* untuk mengoreksi kesalahan OCR umum, seperti angka '4' yang dikoreksi menjadi huruf 'A', angka '1' menjadi 'I', atau angka '5' menjadi 'S'. Hasil akhir dari proses ini adalah *string* dengan format standar [PREFIX] [NOMOR] [SUFFIX] (contoh: "DD 1234 ABC"). Jika proses

formatting gagal karena nomor pelat tidak terdeteksi, sistem akan mengembalikan status "FORMATTING_FAILED".

Mekanisme Voting Konsensus Untuk meningkatkan akurasi pembacaan dan mengatasi kesalahan sesekali dari OCR, sistem menerapkan mekanisme voting konsensus pada level *canonical ID* (cid). Mekanisme ini bekerja sebagai berikut:

1. Akumulasi Sampel Pembacaan
   Selama sebuah pelat nomor (diidentifikasi dengan cid) masih berada di dalam ROI, setiap hasil pembacaan OCR yang valid disimpan sebagai satu sampel. Sistem mengakumulasi semua sampel pembacaan untuk cid tersebut dalam *list samples_formatted*.
2. *Trigger* Voting saat ID keluar dari ROI
   Ketika pelat nomor keluar dari ROI atau menghilang dari *frame*, sistem memeriksa jumlah sampel yang terkumpul untuk menentukan apakah *trigger* voting dapat dijalankan. *Trigger* voting dijalankan jika jumlah sampel pembacaan minimal mencapai 20 sampel.

   Threshold minimum sebanyak 20 sampel ini dipilih berdasarkan dua pertimbangan penting. Pertama, minimal 20 suara diperlukan untuk menghasilkan keputusan mayoritas yang valid dan stabil dengan tujuan untuk memastikan pembacaan pelat akhir yang akurat. Kedua, *threshold* ini berfungsi sebagai mekanisme *filtering* untuk memastikan bahwa hanya pelat nomor kendaraan yang secara konsisten berada dalam ROI (*Region of Interest*) selama periode waktu tertentu yang akan diproses. Dengan menetapkan *threshold* sebanyak 20 sampel, sistem dapat mengabaikan pelat nomor yang hanya tertangkap secara tidak sengaja, seperti pelat kendaraan lain yang hanya muncul sekali atau beberapa kali melintas di *edge frame* atau sudut pandang kamera.

   Mekanisme ini penting untuk memastikan integritas data parkir, di mana hanya kendaraan yang secara sengaja dan permanen masuk ke portal parkir yang akan di voting dan dimasukkan ke dalam sistem log pencatatan akhir. Jika jumlah sampel kurang dari 20, pelat nomor tersebut tidak akan melalui proses voting dan akan di skip dari pencatatan log akhir, karena durasi kehadiran pelat dalam ROI dianggap terlalu singkat dan data sampel tidak cukup representatif untuk menghasilkan hasil pembacaan yang dapat dipercaya sebagai kendaraan yang benar-benar masuk ke area parkir.
3. Voting Terpisah per Komponen
   Voting tidak dilakukan pada keseluruhan teks pelat sekaligus, melainkan dipisah menjadi tiga komponen: *prefix* (Kode Wilayah), nomor (Angka Registrasi), dan *suffix* (Kode Huruf Belakang). Untuk setiap komponen, sistem menghitung frekuensi kemunculan setiap nilai dan memilih nilai yang paling sering muncul sebagai hasil final.

   Sebagai contoh, jika dari 15 sampel pembacaan:

- ***Prefix*:** "DD" muncul 12 kali, "OD" muncul 3 kali → Pilih "DD"
- **Nomor:** "1234" muncul 14 kali, "1284" muncul 1 kali → Pilih "1234"
- **Suffix:** "ABC" muncul 10 kali, "EFG" muncul 5 kali → Pilih "ABC"

Hasil voting akhir adalah gabungan dari ketiga bagian yang terpilih pada voting per bagian masing-masing, yaitu: "DD 1234 ABC"

4. Pencatatan Log Hasil Akhir

   Hasil voting konsensus dicatat ke dalam *file* log final beserta informasi cid, *timestamp*, dan jumlah sampel yang digunakan. Jika jumlah sampel tidak mencukupi *threshold*, pelat nomor tersebut di-*skip* dan tidak dicatat ke hasil final.

2.5.7 Tahap Pengujian Sistem

Tahap pengujian sistem bertujuan untuk mengukur kinerja dan *throughput* dari keseluruhan alur pemrosesan (*end-to-end pipeline*) secara terintegrasi. Pengujian ini difokuskan untuk mengevaluasi performa model saat dijalankan pada perangkat keras CPU dalam mensimulasikan beban kerja operasional yang sesungguhnya.

2.5.7.1 Skenario dan Input Video Pengujian

Pengujian sistem dilakukan menggunakan rekaman video representatif yang mensimulasikan kondisi operasional standar di area parkir dengan total durasi video yang mencakup 1.535 *frame*. Video ini berisi kendaraan yang melintas pada kecepatan normal dengan variasi kondisi yang wajar, seperti oklusi ringan dan pergerakan antrean, yang cukup untuk memvalidasi keandalan sistem dalam mendeteksi, melacak, dan memproses data pelat nomor dari awal hingga akhir.

Sistem dirancang untuk memproses setiap *frame* video secara berurutan (*sequentia*l). Selama proses pengujian berlangsung, sistem mencatat waktu inferensi rata-rata secara kumulatif setiap interval 50 *frame* untuk memantau stabilitas performa seiring berjalannya waktu. Pengukuran waktu inferensi ini difokuskan secara spesifik pada durasi komputasi inti model AI, yang mencakup seluruh proses deteksi objek (*object detection*) hingga pelacakan objek (*object tracking*), namun tidak memperhitungkan waktu *overhead* sistem seperti proses *decoding* video dan *rendering* visualisasi hasil ke layar monitor.

Untuk mengevaluasi efektivitas strategi optimalisasi yang diusulkan, pengujian dilakukan dengan membandingkan performa antara model *baseline* dengan presisi penuh (*floating-point* 32-bit) dan model hasil optimasi kuantisasi (integer 8-bit). Perbandingan ini bertujuan untuk mengukur sejauh mana peningkatan kecepatan (*speedup*) yang dapat dicapai melalui teknik kuantisasi tanpa mengorbankan akurasi sistem secara signifikan.

2.5.7.2 Metrik Evaluasi Sistem

Efisiensi komputasi sistem diukur melalui waktu inferensi rata-rata (ms), waktu inferensi minimum dan maksimum (ms), serta standar deviasi waktu inferensi (ms). Metrik-metrik ini memberikan gambaran mengenai kecepatan pemrosesan dan stabilitas performa model dalam kondisi operasional nyata. Analisis ini bertujuan untuk memvalidasi bahwa model hasil optimalisasi dapat beroperasi secara responsif dan konsisten pada perangkat keras yang digunakan.

2.5.8 Pengujian OCR

Tahap pengujian OCR dilakukan untuk mengevaluasi performa pembacaan pelat nomor model PaddleOCR untuk performa *raw* OCR dan performa keluaran akhir sistem setelah melalui tahap normalisasi format dan mekanisme voting konsensus. Evaluasi *raw* OCR bertujuan untuk mengukur kemampuan murni PaddleOCR dalam mengenali karakter dari citra pelat hasil *cropping*, dengan membandingkan hasil pembacaan teks mentah dengan *ground truth*.

2.5.8.1 Metrik Evaluasi OCR

Performa OCR dievaluasi menggunakan metrik *Character Error Rate* (CER) dan *Plate Read Success* Rate yang telah dijelaskan pada Bab I. Evaluasi dilakukan dengan membandingkan hasil pembacaan sistem (setelah voting konsensus) dengan *ground truth* yang telah dilabeli secara manual, memberikan gambaran objektif mengenai akurasi pembacaan teks pelat nomor pada berbagai kondisi pencahayaan dan kualitas citra.