ANALISIS AKURASI PENGHITUNG KENDARAAN MENGGUNAKAN TEKNOLOGI *COMPUTER VISION* DENGAN ALGORITMA YOLOV8 PADA CCTV JALAN RAYA DI KOTA BANDAR LAMPUNG

Analysis Of Vehicle CountingAccuracy Using Computer Vision Technology Using Yolov8 Algorithm On Street CCTV In Bandar Lampung City

Skripsi

Diusulkan oleh:
M. RAFLI JULIAN
19315031

Acc Revisi
17/10/2023

Acc Revisi
17/10/2023
Auliya Rahman.

PROGRAM STUDI TEKNIK ELEKTRO
UNIVERSITAS TEKNOKRAT INDONESIA
BANDAR LAMPUNG
April, 2023

LEMBAR PERSETUJUAN

Usulan Penelitian

ANALISIS AKURASI PENGHITUNG KENDARAAN MENGGUNAKAN TEKNOLOGI *COMPUTER VISION* DENGAN ALGORITMA YOLOV8 PADA CCTV JALAN RAYA DI KOTA BANDAR LAMPUNG

Yang diajukan oleh :
M. RAFLI JULIAN
19315031

Telah disetujui
Tanggal : 6 November 2023

Mengetahui : Program Studi Teknik Elektro Ketua,	Disetujui : Pembimbing,
Qadhli Jafar Adrian, Bmm., MIT. NIK. 022 16 07 02	**Auliya Rahman Isnain, S.Kom., M.Cs.** NIK. 022 16 02 02

LEMBAR PENGESAHAN

Skripsi

ANALISIS AKURASI PENGHITUNG KENDARAAN MENGGUNAKAN TEKNOLOGI *COMPUTER VISION* DENGAN ALGORITMA YOLOV8 PADA CCTV JALAN RAYA DI KOTA BANDAR LAMPUNG

Dipersiapkan dan disusun oleh

M. RAFLI JULIAN
19315031

Telah dipertahankan di depan Dewan Penguji
Pada tanggal 19 September 2023

Pembimbing,	Penguji,
Auliya Rahman Isnain, S.Kom., M.Cs.	**Akhmad Jayadi, S.Kom., M.Cs.**
NIK. 022 16 02 02	NIK. 022 20 09 01

Skripsi ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar sarjana
Tanggal 6 November 2023

Fakultas Teknik dan Ilmu Komputer,	Program Studi Teknik Elektro,
Dekan,	Ketua,
Dr. H. Mahathir Muhammad, SE., MM.	**Qadhli Jafar Adrian, Bmm., MIT.**
NIK. 023 05 00 09	NIK. 022 16 07 02

LEMBAR PERNYATAAN

Yang bertanda tangan dibawah ini :

Nama : M. Rafli Julian

NPM : 19315031

Program Studi : Teknik Elektro

Dengan ini menyatakan bahwa tugas akhir :

Judl : Analisis Akurasi Penghitung Kendaraan Menggunakan Teknologi *Computer Vision* Dengan Algoritma Yolov8 Pada CCTV Jalan Raya Di Kota Bandar Lampung

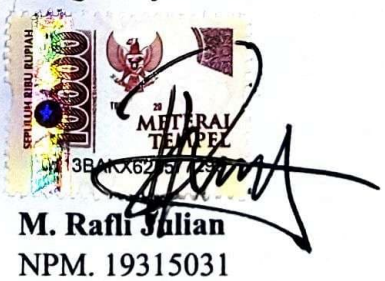
Pembimbing : Auliya Rahman Isnain, S.Kom., M.Cs.

Belum pernah diajukan untuk diuji sebagai persyaratan untuk memperoleh gelar akademik pada berbagai tingkatan di universitas/perguruan tinggi manapun. Tidak ada bagian dalam skripsi ini yang pernah dipublikasikan oleh pihak lain, kecuali bagian yang digunakan sebagai referensi, berdasarkan kaidah penulisan ilmiah yang benar.

Apabila dikemudian hari ternyata laporan tugas akhir yang saya tulis terbukti hasil saduran/plagiat, maka saya akan bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan sebenar-benarnya.

Bandar Lampung, 22 September 2023
Yang menyatakan,



M. Rafli Julian
NPM. 19315031

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia, saya yang bertanda tangan di bawah ini:

Nama : M. Rafli Julian
NPM : 19315031
Program Studi : Teknik Elektro

Jenis karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia, **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

Analisis Akurasi Penghitung Kendaraan Menggunakan Teknologi *Computer Vision* Dengan Algoritma YOLOv8 Pada CCTV Jalan Raya Di Kota Bandar Lampung

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Bandar Lampung

Pada tanggal : 22 September 2023

Yang menyatakan,

M. Rafli Julian

NPM. 19315031

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar sarjana pada Program Studi S1 Teknik Elektro, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia. Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk menyelesaikan laporan ini. Oleh karena itu, penulis mengucapkan terima kasih kepada :

1. Bapak Dr. H. M. Nasrullah Yusuf, S.E., M.B.A., selaku Rektor Universitas Teknokrat Indonesia.
2. Bapak H. Mahatir Muhammad, S.E., M.M., selaku Dekan Fakultas Teknik dan Ilmu Komputer Universitas Teknokrat Indonesia.
3. Bapak Qadhli Jafar Adrian, BMM., MIT., selaku Ketua Program Studi Teknik Elektro.
4. Bapak Auliya Rahman Isnain, S.Kom., M.Cs., selaku Pembimbing yang telah meluangkan waktunya untuk membimbing penulis menyelesaikan skripsi ini.
5. Bapak Akhmad Jayadi, S.Kom., M.Cs., selaku penguji
6. Pembimbing Skripsi serta Bapak dan Ibu Dosen Universitas Teknokrat Indonesia yang telah memberikan motivasi, semangat dan arahan dalam penyususnan skripsi ini.
7. Kedua orang tua yang selalu saya sayangi dan saya cintai, Ayah dan Emak, yang mendidik dan merawat serta mendukung saya, dan tidak pernah henti hentinya memberikan dukungan doa serta motivasi agar bisa mencapai apa yang di inginkan, menjadi kebanggaan serta meraih kesuksesan.
8. Kakak - Kakak saya yang selalu memberikan dukungan moral, motivasi, serta materi.
9. Teman-teman Angkatan 2019 yang selalu bersama-sama berjuang dan saling membantu untuk memberikan semangat dan motivasi.

10. Kepada orang-orang terdekat saya, yang selalu mendukung, selalu memberikan masukan untuk skripsi saya dan dapat menyelesaikan skripsi dengan tepat waktu.
11. Dan masih banyak lagi, yang tidak bisa saya sebutkan satu persatu.

Akhir kata, penulis berharap semoga Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu dan skripsi ini membawa manfaat bagi pengambangan ilmu.

Bandar Lampung, 22 September 2023
Penulis,

M. Rafli Julian
NPM. 19315031

HALAMAN MOTO

DONE IS BETTER THAN PERFECT.

ABSTRAK

Pertumbuhan kendaraan bermotor di Indonesia menjadi ancaman serius bagi masalah transportasi. Kondisi ini terlihat dari kemacetan yang semakin parah dan polusi udara yang semakin meningkat akibat lalu lintas yang semakin padat. Dalam situasi yang semakin sulit seperti ini, diperlukan solusi yang dapat mengatasi permasalahan transportasi. Salah satu solusi untuk mengatasi masalah ini adalah dengan menggunakan teknologi *Computer Vision* untuk menghitung jumlah kendaraan di jalan raya. Hal ini sangat efektif dalam memberikan informasi berharga terkait kondisi arus lalu lintas di jalan raya. Jika informasi berharga ini di analisis lebih lanjut maka akan sangat berguna untuk menunjang keputusan dalam menyelesaikan permasalahan transportasi seperti kemacetan di jalan raya. YOLOv8 adalah versi terkini dari Algoritma YOLO yang menggunakan teknologi *Deep Learning* untuk mendeteksi objek pada gambar. Pada penelitian ini model yang dilatih dengan 100 Epochs menghasilkan nilai akurasi pendeteksian terbaik dibandingkan model lainnya yaitu 0.931 dan nilai *Error Relative* paling rendah untuk objek *motorcycle* sebesar 3.3% dan 0% untuk objek *car*.

Kata kunci : Kendaraan, Kemacetan, Penghitungan Kendaraan Otomatis, Deteksi Kendaraan, *Computer Vision*, YOLO

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Transportasi merupakan aspek penting dalam kehidupan manusia *modern*. Dalam kehidupan sehari-hari, manusia memerlukan transportasi untuk bergerak dari satu tempat ke tempat lainnya. Transportasi menjadi salah satu sarana yang sangat penting dalam kegiatan ekonomi dan sosial manusia (Hidayat & AH, 2017). Transportasi memungkinkan manusia untuk mengakses pekerjaan, pendidikan, perdagangan, dan berbagai kegiatan lainnya.

Seiring dengan perkembangan zaman, jumlah transportasi kendaraan semakin meningkat. Peningkatan ini disebabkan oleh peningkatan jumlah penduduk dan kemajuan teknologi yang memungkinkan produksi kendaraan dalam jumlah yang besar dan efisien. Namun, peningkatan jumlah kendaraan ini juga membawa dampak negatif seperti kemacetan, polusi udara, dan kecelakaan lalu lintas (BPS, 2023).

Menurut data BPS pada tahun 2017 di Indonesia terdapat 100.200.245 unit kendaraan sepeda motor dan pada tahun 2022 meningkat menjadi 125.267.349 unit. Sedangkan untuk kendaraan mobil penumpang pada tahun 2017 terdapat 13.968.202 unit dan pada tahun 2022 meningkat menjadi 17.175.632 unit. Data ini menunjukkan selama 5 tahun terakhir di Indonesia sudah terjadi peningkatan jumlah unit kendaraan sepeda motor sebesar 25% dan untuk kendaraan mobil penumpang mengalami peningkatan sebesar 22,96%.

Pertumbuhan jumlah kendaraan bermotor di Indonesia pada 5 tahun terakhir menunjukkan sebuah pertumbuhan yang signifikan. Menurut Nisak dan Prakoso (2012), pertumbuhan kendaraan bermotor di Indonesia menjadi ancaman serius bagi masalah transportasi. Kondisi ini terlihat dari kemacetan yang semakin parah dan polusi udara yang semakin meningkat akibat lalu lintas yang semakin padat. Situasi ini dapat menjadi ancaman serius bagi kelancaran transportasi di Indonesia, khususnya di kota-kota besar yang terkena dampak parah akibat padatnya lalu lintas. Dalam situasi yang semakin sulit seperti ini, diperlukan solusi yang dapat mengatasi permasalahan transportasi.

Salah satu solusi untuk mengatasi masalah ini adalah dengan menggunakan teknologi *Computer Vision* untuk menghitung jumlah kendaraan di jalan raya. Teknologi *Computer Vision* telah digunakan secara luas untuk menghitung kendaraan di jalan raya di berbagai negara. Teknologi ini menggunakan kamera CCTV yang dipasang di jalan raya untuk mengambil gambar kendaraan yang lewat, dan kemudian menerapkan algoritma *Computer Vision* untuk menghitung jumlah kendaraan. Kamera seperti CCTV sudah banyak digunakan sebagai implementasi untuk sistem pengawasan lalu lintas (Kejriwal et al., 2022). Hal ini sangat efektif dalam memberikan informasi berharga terkait kondisi arus lalu lintas di jalan raya. Jika informasi berharga ini di analisis lebih lanjut maka akan sangat berguna untuk menunjang keputusan dalam menyelesaikan permasalahan transportasi seperti kemacetan di jalan raya.

Salah satu algoritma *Computer Vision* yang paling populer digunakan untuk tugas pendeteksian objek adalah YOLO (*You Only Look Once*). Algortima ini memiliki keseimbangan antara kecepatan dan akurasi yang tinggi sehingga sangat handal dalam tugas mengidentifikasi objek (Terven & Cordova-Esparza, 2023). YOLOv8 adalah versi terkini dari Algoritma YOLO (*You Only Look Once*) yang menggunakan teknologi *Deep Learning* untuk mendeteksi objek pada gambar.

Algoritma ini dikembangkan oleh Joseph Redmon dan timnya pada tahun 2021 sebagai versi terbaru dari algoritma YOLO (*You Only Look Once*) yang pertama kali diperkenalkan pada tahun 2015. Salah satu kelebihan YOLOv8 adalah kecepatan deteksinya yang sangat cepat, sehingga memungkinkan penggunaan algoritma ini dalam situasi *Real-time*, seperti di industri keamanan, transportasi, kendaraan otonom, dan robotika (Terven & Cordova-Esparza, 2023). Selain itu, YOLOv8 memiliki akurasi dan efisiensi yang lebih unggul dibandingkan dengan algoritma pendeteksian objek lainnya yaitu CenterNet, EfficientDet, Fast R-CNN dan SSD (Alnujaidi et al., 2023). Algoritma YOLO (*You Only Look Once*) terus dikembangkan untuk menghasilkan deteksi objek yang semakin akurat dan lebih cepat, sehingga memberikan manfaat yang besar bagi berbagai industri dan aplikasi di masa depan.

Karena Algoritma YOLOv8 masih terbilang baru, keakuratan sistem penghitung kendaraan menggunakan teknologi *Computer Vision* dengan algoritma

YOLOv8 pada CCTV jalan raya masih memerlukan analisis yang lebih lanjut, terutama di kota-kota yang memiliki kondisi lalu lintas yang berbeda. Oleh karena itu, penelitian ini bertujuan untuk menganalisis akurasi sistem penghitung kendaraan menggunakan teknologi *Computer Vision* dengan algoritma YOLOv8 pada CCTV jalan raya di Kota Bandar Lampung.

Berdasarkan permasalahan yang dihadapi penulis, maka penulis memutuskan untuk melakukan penelitian yang berjudul **"Analisis Akurasi Penghitung Kendaraan Menggunakan Teknologi *Computer Vision* Dengan Algoritma Yolov8 Pada CCTV Jalan Raya Di Kota Bandar Lampung"**. Algoritma YOLOv8 akan diimplementasikan untuk mengolah data yang di dapatkan dari CCTV jalan raya yang ada di Bandar Lampung, lalu akan dilakukan analisis terhadap hasil dari akurasi penghitungungan kendaraan dan membandingkannya dengan penghitungan manual, dengan adanya penelitian ini diharapkan dapat memberikan informasi terkait efektivitas dari penerapan teknologi *Computer Vision* sehingga dapat menjadi solusi yang lebih efektif dalam mengelola lalu lintas di Kota Bandar Lampung dan membantu meningkatkan keselamatan jalan raya secara keseluruhan.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan sebelumnya maka dapat ditarik rumusan masalah dalam penelitian ini tentang bagaimana tingkat akurasi pendeteksian dan penghitungan kendaraan menggunakan teknologi *Computer Vision* dengan algoritma YOLOv8 pada CCTV jalan raya di kota Bandar Lampung.

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai pada penelitian ini adalah:

1. Untuk mengevaluasi akurasi pendeteksian kendaraan menggunakan teknologi *Computer Vision* dengan algoritma YOLOv8 pada CCTV jalan raya kota Bandar Lampung.
2. Untuk mengevaluasi akurasi penghitungan kendaraan menggunakan teknologi *Computer Vision* dengan algoritma YOLOv8 pada CCTV jalan raya kota Bandar Lampung.

1.4 Batasan Masalah

Adapun Batasan masalah pada penelitian ini adalah:

1. Penelitian ini hanya difokuskan pada teknologi *Computer Vision* dengan menggunakan algoritma YOLOv8 untuk mendeteksi dan menghitung kendaraan pada CCTV jalan raya di Kota Bandar Lampung.
2. CCTV yang digunakan dalam penelitian ini adalah CCTV jalan raya Simpang Gramedia.
3. Data rekaman CCTV didapatkan dengan cara merekam *streaming* CCTV pada *website* https://www.cctvkotabandarlampung.com/
4. Penelitian ini tidak membahas mengenai implementasi teknologi *Computer Vision* dengan algoritma lain selain YOLOv8.
5. Penelitian ini hanya akan mendeteksi dan menghitung kendaraan mobil dengan label *car* dan motor dengan label *motorcycle*.
6. Penelitian ini hanya menggunakan 1000 label *car* dan 1000 label *motorcycle*
7. Parameter yang di setel saat pelatihan terbatas hanya pada *Epochs* dan hanya menggunakan model YOLOv8.
8. Penelitian ini tidak membahas mengenai teknologi *Computer Vision* dengan algoritma YOLOv8 pada malam hari atau kondisi cahaya yang kurang memadai.
9. Pengujian dilakukan pada *platform Google Colaboratory.*
10. Bahasa pemrograman yang digunakan adalah python.
11. Pengujian tidak dilakukan secara *Real-time*.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Memberikan informasi tentang tingkat akurasi teknologi *Computer Vision* dengan menggunakan algoritma YOLOv8 dalam mendeteksi dan menghitung kendaraan pada CCTV jalan raya di Kota Bandar Lampung.
2. Memberikan masukan kepada pihak-pihak terkait, seperti dinas perhubungan, untuk meningkatkan efektivitas sistem pemantauan lalu lintas di Kota Bandar Lampung.

3. Menjadi referensi bagi peneliti lain dalam melakukan penelitian serupa di masa depan.
4. Meningkatkan pemahaman dan pengetahuan tentang penggunaan teknologi *Computer Vision* dan algoritma YOLOv8 pada bidang penghitungan kendaraan di jalan raya.

BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Penelitian ini didukung oleh tinjauan pustaka yang relevan, yang rinciannya tercantum dalam Tabel 2.1.

Tabel 2. 1 Penelitian Terdahulu

No	Peneliti	Judul Penelitian	Hasil
1	(Amwin, 2021)	Deteksi Dan Klasifikasi Kendaraan Berbasis Algoritma *You Only Look Once* (Yolo)	Nilai AP pada mobil sebesar 99,88%, pada sepeda motor 97,79%, becak sebesar 100%, truk sebesar 100%, dan bus 99,09%. Sedangkan nilai mAP sebesar 99,35% dengan waktu pemrosesan selama 4 detik. Kualitas video dapat mempengaruhi pendeteksian karena semakin tinggi kualitas video maka hasil klasifikasi dan *bounding box* semakin tinggi pula
2	(Cholid, 2021)	Penerapan Metode *You Only Look Once* (Yolo) Dan *Support Vector Regression* (Svr) Untuk Perhitungan Kepadatan Lalu Lintas Berdasarkan Area *Occupancy*	Metode YOLO dapat mengenali jenis kendaraan dan memperoleh akurasi 75,16% pada kondisi lalu lintas siang hari. Untuk estimasi kepadatan lalu lintas berdasarkan okupansi diterapkan metode YOLO dan SVR. Hal ini direpresentasikan dengan *kernel olinomial* dengan parameter

Tabel Lanjutan 2. 1 Penelitian Terdahulu

No	Peneliti	Judul Penelitian	Hasil
			optimasi epsilon = 1,0, derajat = 1, gamma = 0,0, dan coef0 = 2,0 memperoleh skor MAPE 53,59; nilai ini lebih kecil dari penggunaan *kernel* linier mendapatkan nilai MAPE sebesar 55,5
3	(Pratama & Rasywir, 2021)	Eksperimen Penerapan Sistem *Traffic Counting* dengan Algoritma YOLO (*You Only Look Once*) V.4.	Diperoleh hasil deteksi yang mempunyai akurasi yang cukup baik pada hasil pemisahan *frame-frame* dari video data. Walaupun tidak seluruh hasil penggunaan algoritma ini sempurna pada semua data, namun didapatkan hasil cenderung baik.
4	(Khatami, 2022)	Deteksi Kendaraan Menggunakan Algoritma *You Only Look Once* (Yolo) V3	Didapatkan nilai *Precision* yang sempurna, nilai *Recall* di atas 70%, dan F1 *Score* di atas 80%. Dari kedua pengujian bobot jaringan tersebut, diketahui bahwa objek mobil dan kendaraan besar merupakan objek yang paling berhasil dapat dideteksi oleh ketiga jaringan tersebut. Hal ini disebabkan oleh ukuran dan fitur fisik objek mobil dan kendaraan besar yang hampir mirip antara satu sama lain.

Tabel Lanjutan 2. 2 Penelitian Terdahulu

No	Peneliti	Judul Penelitian	Hasil
5	(Isnaini, 2020)	Aplikasi Penghitung Kendaraan yang Melintas di Jalan Raya Berdasarkan Metode Yolo *Object Detection*	Dengan pengujian menggunakan 3 video yang diambil dari hasil rekam dan dari internet yang sudah bisa mendeteksi dan menghitung dengan akurat yang mempunyai nilai keakuratan paling tinggi 56% dan paling rendah 10%.
6	(Fachrie, 2020)	*A Simple Vehicle Counting System Using Deep Learning with YOLOv3 Model*	Penghitungan didasarkan pada empat jenis kendaraan yaitu mobil, sepeda motor, bus, dan truk. Hasilnya, sistem yang di usulkan mampu menghitung kendaraan yang melintasi jalan berdasarkan video yang direkam oleh kamera dengan akurasi tertinggi 97,72%.
7	(Sumarudin et al., 2019)	Aplikasi Penghitung Kendaraan Pada Jalur Pantura Menggunakan Blob Deteksi dan Kalman Filter	Dapat disimpulkan penggunaan kalman *filter* memiliki nilai deteksi 78.81% sehingga dapat diimplementasikan dalam sistem pendeteksian jumlah kendaraan di jalur pantura.
8	(Faisal & Abadi, 2021)	Implementasi SSD_Resnet50_V1 Untuk Penghitung Kendaraan	Aplikasi dengan model deteksi SSD_Resnet50_v1 mendapatkan akurasi sebesar 56,49% untuk kendaraan berjenis motor, dan 54,43% untuk mobil. Hasil akurasi terhitung rendah dikarenakan model deteksi.

Tabel Lanjutan 2. 3 Penelitian Terdahulu

			kurang bisa mengenali objek kendaraan pada malam hari, sehingga banyak kendaraan yang tidak terdeteksi dan ada kendaraan berjenis motor yang dideteksi sebagai mobil
9	(Majumder & Wilmot, 2023)	*Automated Vehicle Counting from Pre-Recorded Video Using You Only Look Once (YOLO) Object Detection Model*	Keakuratan penghitungan otomatis dievaluasi dibandingkan dengan penghitungan manual, dan ditemukan sekitar 90 persen. Selain itu, analisis manfaat-biaya (B/C) menunjukkan bahwa menerapkan metode penghitungan otomatis menghasilkan 1,76 kali lipat investasi.
10	(Ramadhan et al., 2023)	Prototype penghitung Jumlah Dan Kecepatan Kendaraan Otomatis Secara *Real Time* Berbasis *Computer Vision* Menggunakan Metode *Background Subtraction*	Untuk raspberry pi mendapatkan rata-rata persentasi akurasi ketepatan sebesar 15% sedangkan untuk laptop mendapat rata-rata persentasi akurasi ketepatan sebesar 90%. Proses perhitungan jumlah kendaraan kurang baik dilakukan menggunakan raspberry pi yang disebabkan *drop* FPS (*frame rate per second*) yang sangat jauh.

Pada Tabel 2.1 telah di perlihatkan tentang penelitian terdahulu yang sejenis dengan penelitian ini. Namun, penelitian ini memiliki fokus yang berbeda, yaitu pada analisis akurasi pendeteksian dan penghitungan kendaraan menggunakan teknologi *Computer Vision* dengan algoritma YOLOv8 pada CCTV jalan raya di Kota Bandar Lampung. Selain itu, penelitian ini juga menggunakan perangkat laptop dengan *platform Google Colaboratory* untuk melakukan pengujian. Terdapat empat metrik yang akan digunakan untuk mengevaluasi akurasi hasil pengujian, yaitu *Accuracy, Precision*, *Recall*, dan *F1-Score*. Oleh karena itu, penelitian ini diharapkan dapat memberikan kontribusi baru pada pengembangan teknologi penghitung kendaraan dengan tingkat akurasi yang lebih baik.

2.2 Kendaraan

Kendaraan merupakan alat transportasi yang berguna sebagai pengangkut orang atau barang dari suatu tempat ke tempat lainnya. Kendaraan dapat berupa mobil, motor, truk, bus, kereta api, kapal laut, dan pesawat terbang. Jenis kendaraan yang digunakan tergantung pada tujuan dan jarak tempuh yang ingin dicapai. Kendaraan umumnya terdiri dari beberapa komponen utama seperti mesin, roda, sistem suspensi, sistem pengereman, dan sistem kemudi. Setiap komponen ini memiliki peran penting dalam menghasilkan kinerja yang optimal dari kendaraan. Selain itu, ada juga teknologi terbaru yang digunakan pada kendaraan seperti teknologi *Computer Vision* dan *Machine Learning* untuk meningkatkan efisiensi dan keamanan dalam berkendara.

Seiring berjalannya waktu pertumbuhan jumlah kendaraan terus meningkat. dalam penggunaannya sendiri, kendaraan dapat memberikan berbagai dampak positif dan negatif. Di satu sisi, kendaraan dapat meningkatkan mobilitas dan aksesibilitas ke berbagai tempat. Namun, di sisi lain, penggunaan kendaraan juga dapat menimbulkan masalah seperti kemacetan, polusi udara, dan kecelakaan lalu lintas. Pertumbuhan jumlah kendaraan perlu disikapi secara bijak (Priyambodo, 2018). Oleh karena itu, perlu dilakukan pengaturan dan pengawasan yang baik dalam penggunaan kendaraan untuk meminimalkan dampak negatif dan memaksimalkan manfaatnya bagi masyarakat.

Setiap jenis kendaraan memiliki karakteristik yang berbeda-beda tergantung dari jenis kendaraannya (Khatami, 2022). Salah satu karakteristik kendaraan yang penting adalah dimensinya, yaitu panjang, lebar, dan tinggi kendaraan. Dimensi kendaraan menjadi sangat penting untuk memastikan bahwa kendaraan tersebut dapat melewati jalan atau jembatan yang ada dengan aman dan tanpa mengganggu lalu lintas yang lain. Selain itu, dimensi kendaraan juga memengaruhi kapasitas muatan yang dapat diangkut oleh kendaraan tersebut. Selain dimensi, karakteristik kendaraan lainnya adalah kecepatan maksimum dan akselerasi kendaraan. Kecepatan maksimum kendaraan menunjukkan seberapa cepat kendaraan dapat berjalan, sedangkan akselerasi kendaraan menunjukkan seberapa cepat kendaraan dapat meningkatkan kecepatannya dari keadaan diam.

2.3 Jalan Raya

Jalan raya merupakan sebuah infrastruktur transportasi darat yang biasanya dilalui oleh kendaraan bermotor untuk menghubungkan berbagai wilayah atau daerah. Jalan raya dipergunakan sebagai tempat akumulasi dari berbagai kendaraan bermotor maupun tak bermotor (Wibisana, 2009). Jalan raya umumnya terdiri dari beberapa jalur yang dipisahkan oleh pembatas jalan atau marka jalan. Karakteristik jalan raya bervariasi tergantung pada tipe jalan, lokasi, dan lalu lintas yang melaluinya. Namun secara umum, jalan raya memiliki karakteristik seperti lebar jalan yang memadai, permukaan jalan yang rata dan halus, dan terdapat aturan lalu lintas yang jelas untuk menjaga keselamatan pengguna jalan. Jalan raya juga dapat dilengkapi dengan fasilitas pendukung seperti jembatan, *flyover*, *underpass*, *rest area*, serta CCTV untuk memantau lalu lintas dan keamanan pengguna jalan.

2.4 *Deep Learning*

Deep Learning adalah salah satu teknik pembelajaran mesin (*Machine Learning*) yang memungkinkan mesin atau komputer untuk mempelajari representasi yang lebih abstrak dan kompleks dari data. Teknik ini dilakukan dengan menggunakan arsitektur jaringan saraf tiruan (*neural network*) yang sangat dalam dengan banyak lapisan (*layer*) dan parameter yang sangat besar. Pada *Deep Learning*, data yang diberikan kepada mesin tidak perlu diproses secara manual atau dipilih fiturnya secara terpisah. Sebaliknya, mesin akan mempelajari fitur-fitur

yang paling penting secara otomatis dengan melakukan optimasi pada parameter jaringan saraf tiruan yang terdiri dari ratusan bahkan ribuan *neuron* . Dengan demikian, *Deep Learning* sangat berguna dalam mengolah data yang kompleks seperti gambar, suara, dan teks. Teknik ini telah banyak digunakan dalam berbagai bidang seperti visi komputer, pengenalan suara, pengenalan tulisan tangan, dan bahkan prediksi kejadian masa depan.

Deep Learning dapat didefinisikan sebagai salah satu kelompok algoritma dalam *Machine Learning* yang menggunakan beberapa lapisan pemrosesan nonlinier yang disusun secara berturut-turut untuk melakukan ekstraksi fitur dan transformasi data (Diponegoro et al., 2021). Dalam *Deep Learning*, terdapat banyak lapisan yang berbeda dalam memproses data secara bertahap untuk mencapai tujuan tertentu. Setiap lapisan tersebut akan mempelajari representasi data yang berbeda dan berguna untuk mengekstraksi fitur yang diperlukan dalam memahami pola data yang rumit. Oleh karena itu, DL sering digunakan dalam aplikasi yang membutuhkan pemrosesan data yang kompleks, seperti pengenalan suara, pengenalan wajah, pengenalan objek, dan sebagainya.

Deep Learning memiliki banyak domain aplikasi yang berkembang pesat, termasuk diantaranya adalah *Computer Vision* (Shorten et al., 2021). Penerapan *Deep Learning* pada *Computer Vision* memungkinkan komputer untuk memproses dan memahami gambar dan video dengan cara yang sama seperti manusia, termasuk melakukan tugas-tugas seperti deteksi objek, pengenalan wajah, dan klasifikasi gambar secara otomatis dan akurat. Oleh karena itu, *Deep Learning* dan *Computer Vision* adalah bidang yang sangat menjanjikan untuk dijelajahi dan dikembangkan lebih lanjut di masa depan.

2.5 *Image Processing*

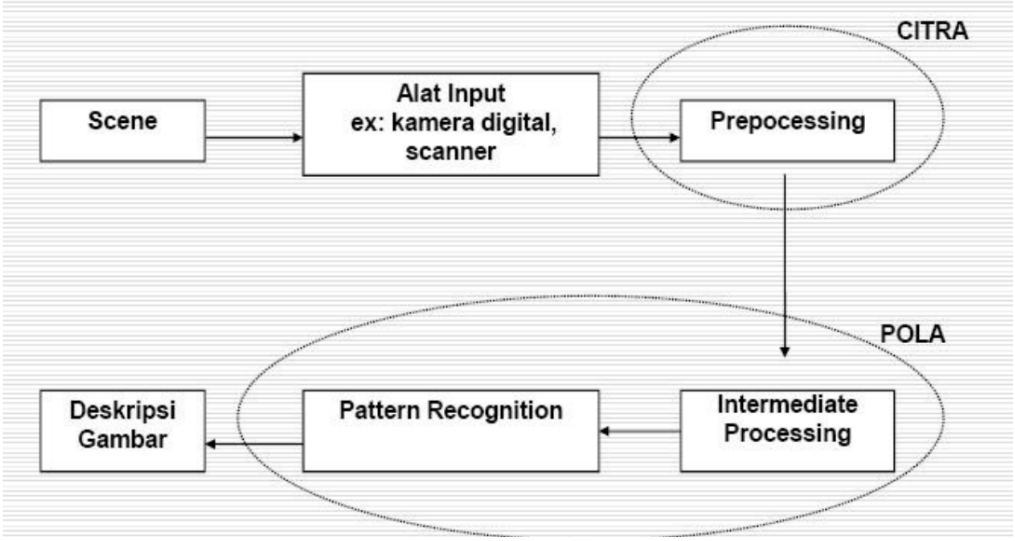
Image Processing adalah suatu proses pengolahan citra atau gambar untuk mendapatkan informasi yang berguna dari gambar tersebut. Proses ini dapat dilakukan dengan menggunakan teknik-teknik seperti *filtering*, *enhancement*, *restoration*, dan *compression*. Tujuan dari *Image Processing* adalah untuk menyempurnakan citra dan mendapatkan informasi yang berguna (Maulana et al., 2023). Seperti memperjelas atau menghilangkan *noise* pada citra, atau mengubah

tampilan citra untuk memudahkan analisis. *Image Processing* sering digunakan dalam berbagai bidang, seperti pengolahan medis, pengenalan wajah, pengolahan citra satelit, pengenalan pola, dan lain-lain. Teknologi *Image Processing* sangat penting dalam pengembangan algoritma *Computer Vision*, termasuk dalam pengembangan algoritma *Deep Learning*.

Sebuah citra adalah representasi visual dari objek atau fenomena yang disajikan dalam bentuk gambar. Citra dapat disimpan dalam berbagai format *digital* dan dapat membawa banyak informasi (Adhinata et al., 2020). Sebuah citra dapat berisi informasi tentang warna, bentuk, ukuran, tekstur, dan banyak lagi. Informasi ini dapat diekstraksi dan dimanipulasi menggunakan teknik pengolahan citra seperti segmentasi citra, deteksi fitur, dan pengenalan pola. Citra adalah sumber daya yang sangat berharga dalam dunia komputasi visual dan penelitian terus dilakukan untuk meningkatkan teknik pengolahan citra yang ada.

2.6 *Computer Vision*

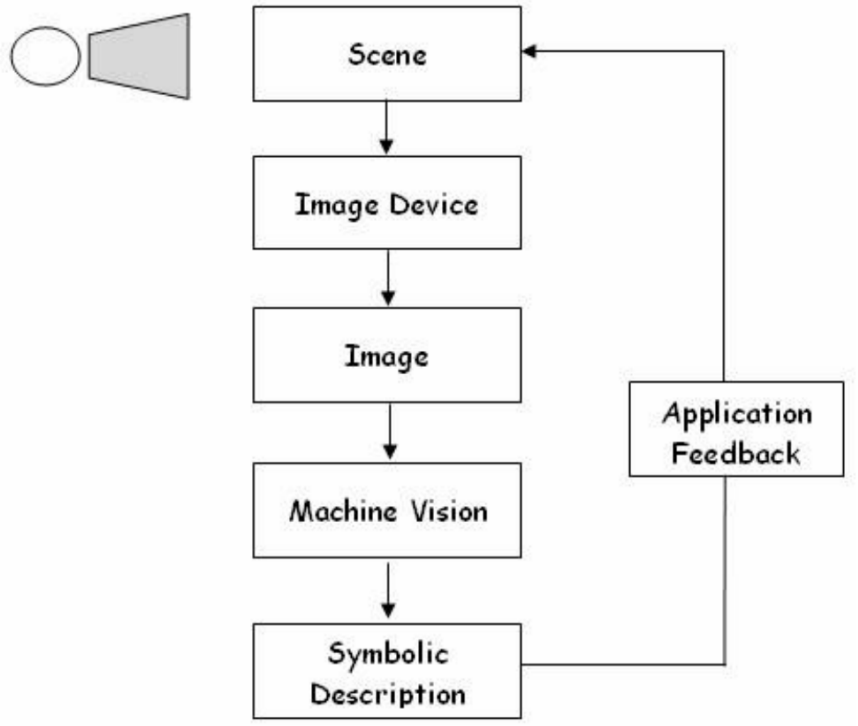
Computer Vision adalah bidang studi yang memungkinkan komputer untuk memproses, menganalisis, dan memahami gambar dan video dalam cara yang mirip dengan penglihatan manusia. Salah satu tujuan dari *Computer Vision* adalah untuk membuat komputer dapat "melihat" dunia seperti yang dilihat oleh manusia (Wahyudi & Kartowisastro, 2011). Konsep utama dalam *Computer Vision* adalah pengolahan citra, yang mencakup teknik untuk memanipulasi gambar untuk mendapatkan informasi yang berguna, seperti pengenalan objek, ekstraksi fitur, segmentasi objek, dan deteksi objek. Berikut adalah gambaran dari proses dalam visi komputer yang dapat dilihat pada gambar 2.1.



Gambar 2. 1 Proses pada Visi Komputer

Computer Vision dapat diklasifikasikan berdasarkan fungsinya menjadi dua kategori utama, yaitu inspeksi dan *sorting* (Neethu & Anoop, 2015). Inspeksi merupakan salah satu aplikasi *Computer Vision* yang digunakan untuk memeriksa kondisi fisik dan kemungkinan kecacatan pada sebuah objek. Contohnya adalah inspeksi pada produk manufaktur atau mesin, seperti memeriksa cacat pada produk atau memeriksa keandalan mesin pada suatu proses produksi. Sedangkan *sorting*, seperti namanya, digunakan untuk mengurutkan atau memisahkan objek-objek yang memiliki kriteria tertentu. Contohnya adalah memisahkan buah yang sudah matang atau tidak matang dalam industri pertanian, atau memisahkan sampah berdasarkan jenisnya di industri pengolahan sampah. Kedua kategori aplikasi ini memiliki beragam metode dan teknik yang berbeda untuk memproses data citra dan mencapai tujuan yang diinginkan.

Dalam bidang visi komputer, terdapat 7 elemen utama yang membentuk struktur dasar suatu sistem visi komputer, yaitu *light sources*, *scene*, *image device*, *machine vision*, *symbolic description*, dan *possible application feetback* (Amrizal & Aini, 2013). Berikut ini adalah struktur visual dari sistem visi komputer yang dapat dilihat pada gambar 2.2.



Gambar 2. 2 Struktur Visi Komputer

Adapun komponen dari *Computer Vision* sebagai berikut:

1. *Light source*, yang merupakan sumber cahaya yang digunakan dalam *Computer Vision*.
2. *Scene*, yaitu kumpulan dari objek-objek,
3. *Image device*, yaitu alat yang digunakan untuk mengubah letak benda yang direpresentasikan.
4. *Machine vision*, merupakan sistem yang menafsirkan gambar melalui ciri-ciri, pola maupun objek yang dapat ditelusuri oleh sistem.
5. *Symbolic description*, yaitu sistem yang dapat menggambarkan kinerja sistem kedalam simbol-simbol yang telah dimengerti oleh sistem.
6. *Possible application feedback*, merupakan suatu keadaan yang bisa memberikan respon menerima gambar dari suatu sistem penglihatan

2.7 *Object Detection*

Object Detection atau pengenalan objek adalah salah satu tugas mendasar dalam visi komputer (Vahab et al., 2019). *Object Detection* adalah salah satu cabang dari *Computer Vision* yang bertujuan untuk mendeteksi keberadaan dan

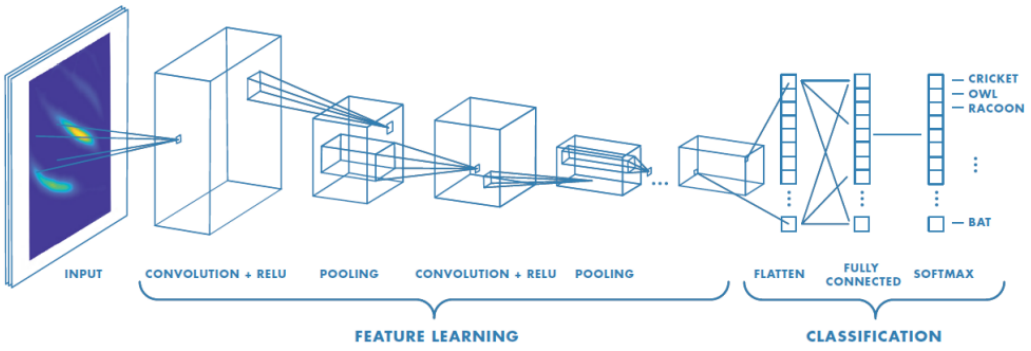
lokasi objek pada suatu gambar atau video. Teknik ini umumnya digunakan untuk melakukan pengenalan objek secara otomatis pada suatu gambar atau video. Tujuannya adalah untuk memberikan kemampuan pada mesin untuk mengenali objek seperti manusia, hewan, kendaraan, atau benda lainnya dalam gambar dan memperoleh informasi tentang kuantitas dan lokasi objek-objek tersebut. Pada dasarnya, *Object Detection* terdiri dari beberapa tahapan, yaitu:

1. *Preprocessing*: tahap awal yang dilakukan adalah mengubah gambar menjadi format yang dapat diproses oleh model *Deep Learning*, seperti mengubah ukuran gambar menjadi ukuran yang seragam atau normalisasi intensitas piksel.
2. *Feature extraction*: tahap ini melibatkan penggunaan algoritma *Deep Learning* untuk melakukan ekstraksi fitur pada gambar. Pada tahap ini, gambar akan diproses oleh beberapa lapisan jaringan saraf tiruan untuk mengekstrak fitur yang spesifik dan kompleks dari gambar.
3. *Region proposal*: tahap ini akan menentukan daerah-daerah pada gambar yang kemungkinan berisi objek, dengan *car*a memproses gambar menggunakan algoritma seperti *selective search* atau *region-based Convolutional Neural Network* (R-CNN).
4. *Classification*: pada tahap ini, objek yang terdeteksi akan diklasifikasikan ke dalam kategori yang sudah ditentukan. Ini dilakukan dengan menggunakan teknik klasifikasi seperti *Convolutional Neural Network* (CNN) atau *Support Vector Machine* (SVM).
5. *Post-processing*: tahap akhir dalam *Object Detection* adalah *post-processing*, di mana hasil deteksi objek yang sudah diklasifikasikan dianalisis untuk menghilangkan deteksi duplikat atau menggabungkan deteksi yang terpisah menjadi satu.

Dalam aplikasinya, *Object Detection* dapat digunakan dalam berbagai bidang, seperti otomotif, keamanan, transportasi, pengawasan, atau bahkan dalam industri kreatif seperti film dan videografi. Salah satu algoritma populer yang digunakan dalam *Object Detection* adalah YOLO (*You Only Look Once*).

2.8 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) atau dalam bahasa Indonesia sering disebut sebagai Jaringan Saraf Tiruan Konvulisional merupakan sebuah algoritma *Deep Learning* yang banyak digunakan untuk mengolah data gambar atau visual. CNN termasuk dalam kategori *Deep Neural Network* dikarenakan memiliki arsitektur dengan kedalaman yang tinggi (Putra, 2016). Umumnya, CNN banyak digunakan pada data citra untuk mempelajari dan mendeteksi fitur-fitur penting dalam gambar. Pada umumnya, *Neural Network* menggunakan data *array* satu dimensi sebagai *input*. Namun, pada algoritma CNN, data yang digunakan sebagai *input* berupa data dua dimensi (Ovtcharov et al., 2015). Berikut pada gambar 2.3 adalah gambaran dari arsitektur CNN.



Gambar 2. 3 Arsitektur CNN

CNN memiliki empat lapisan utama yang terdiri dari lapisan konvolusi (*Convolution Layer*), lapisan aktivasi (*Activation Layer*), lapisan penggabungan (*Pooling Layer*), dan lapisan terhubung penuh (*Fully-Connected Layer*) (Hibatullah, 2019). Berikut adalah penjelasan dari masing-masing *layer* yang ada pada CNN:

1. *Convolution layer*, adalah lapisan pertama dalam CNN yang melakukan konvolusi pada *input* gambar dengan sejumlah *filter* untuk menghasilkan *feature map*.
2. *Activation layer,* adalah lapisan yang mengaktifkan atau mematikan *neuron* pada *feature map* berdasarkan ambang batas tertentu.
3. *Pooling layer*, digunakan untuk mengurangi dimensi *feature map* dengan cara menggabungkan beberapa piksel menjadi satu nilai.

4. *Fully-connected layer,* adalah lapisan terakhir dalam CNN yang melakukan klasifikasi dengan menghubungkan *neuron* pada *layer* sebelumnya ke *neuron* pada *layer output*.

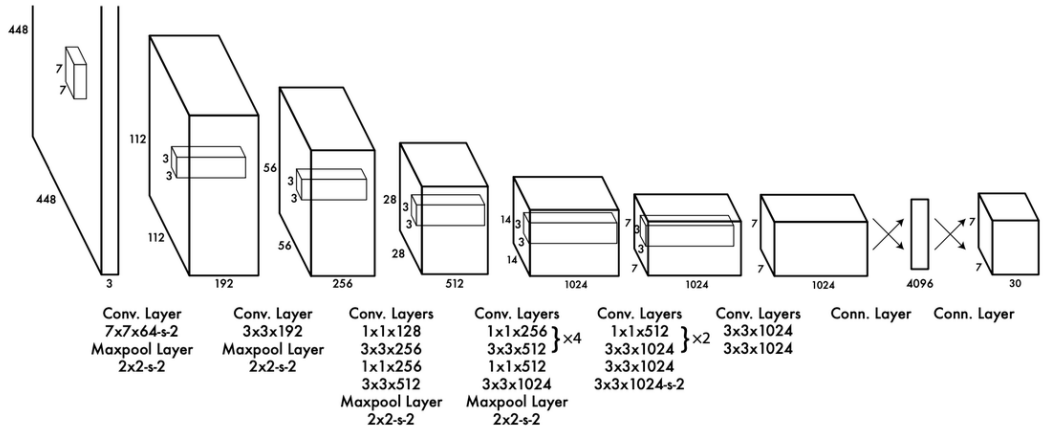
CNN bekerja dengan cara mempelajari *feature* pada gambar dengan cara melakukan konvolusi pada citra tersebut. Pada lapisan konvolusi, CNN menggunakan *filter* atau *kernel* yang berfungsi untuk memisalkan informasi pada citra secara bertahap, sehingga menghasilkan *feature map* yang merepresentasikan bagian-bagian penting pada citra. Selanjutnya, pada lapisan aktivasi, *feature map* yang dihasilkan pada lapisan konvolusi diproses dengan fungsi aktivasi untuk memperkuat sinyal yang relevan dan menghilangkan *noise* atau sinyal yang tidak relevan. Kemudian, pada lapisan *pooling*, *feature map* yang sudah diaktivasi akan dikompresi dengan melakukan operasi reduksi ukuran citra sehingga menjadikannya lebih efisien dan efektif. Setelah melalui beberapa lapisan konvolusi dan *pooling*, data yang sudah diolah kemudian diteruskan ke lapisan terhubung penuh atau *fully-connected layer*. Pada lapisan ini, CNN melakukan klasifikasi dan penentuan kelas berdasarkan *feature* yang sudah diekstrak sebelumnya.

Dengan demikian, secara keseluruhan cara kerja CNN adalah dengan mempelajari *feature* pada gambar secara bertahap dengan memanfaatkan lapisan konvolusi, aktivasi, dan *pooling* untuk menghasilkan representasi yang lebih efektif dan efisien. Selanjutnya, hasil representasi tersebut digunakan untuk melakukan klasifikasi pada lapisan *fully-connected layer*.

2.9 YOLO

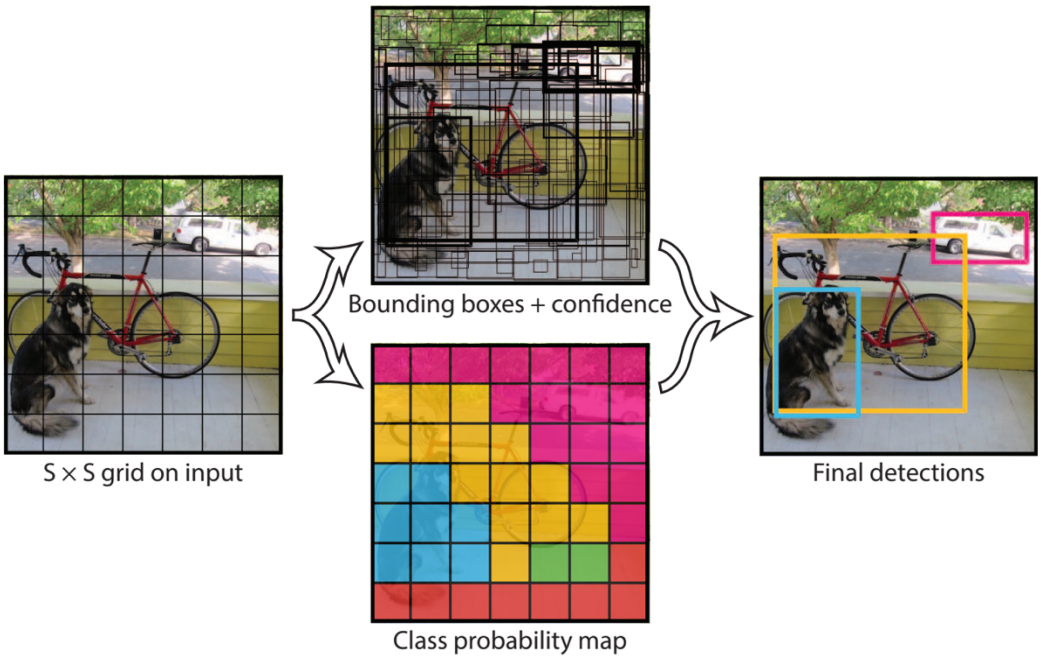
You Only Look Once (YOLO) adalah salah satu algoritma dalam *Computer Vision* yang digunakan untuk melakukan deteksi objek secara *Real-time* pada gambar maupun video. Inti dari algoritma YOLO adalah memiliki ukuran model yang kecil serta memiliki kecepatan perhitungan yang cepat (Jiang et al., 2022). Hal ini memungkinkan YOLO untuk melakukan deteksi objek secara *Real-time* dengan akurasi yang tinggi. Model YOLO menggunakan arsitektur *Convolutional Neural Network* (CNN) dengan banyak lapisan *convolutional*, yang memungkinkan model untuk mengekstraksi fitur gambar secara efisien.

Arsitektur YOLO asli terdiri dari 24 lapisan konvolusi yang diikuti oleh dua lapisan terhubung sepenuhnya (*fully connected layer*) (Jiang et al., 2022). Pada setiap lapisan konvolusi, YOLO menggunakan *filter* berukuran 3x3 dengan *stride* 1, dan dilakukan *zero-padding* pada tepi citra *input*. Setelah lapisan konvolusi, dilakukan lapisan *pooling* berukuran 2x2 dengan *stride* 2 untuk mengurangi dimensi citra *input* dan meningkatkan efisiensi komputasi. Pada bagian akhir, YOLO menghasilkan prediksi objek dengan menggunakan lapisan terhubung sepenuhnya (*Fully Connected Layer*) dengan *neuron* yang berkorespondensi dengan setiap *grid* pada citra *input*. Setiap *neuron* ini menentukan apakah suatu objek terdeteksi pada *grid* tersebut, dan memberikan informasi tentang ukuran dan lokasi objek relatif terhadap *grid*. Arsitektur dari YOLO dapat dilihat pada Gambar 2.4



Gambar 2. 4 Arsitektur YOLO (Redmon et al., 2016)

Proses deteksi objek pada YOLO dilakukan dengan membagi gambar ke dalam *grid-cell* yang sama besar, kemudian setiap *grid-cell* akan bertanggung jawab untuk mendeteksi objek yang ada di dalamnya. Selanjutnya, setiap *grid-cell* akan menghasilkan beberapa *bounding box* dengan ukuran dan posisi yang berbeda-beda, kemudian dilakukan *non-max suppression* untuk memilih *bounding box* yang paling sesuai dengan objek yang sebenarnya. Gambar 2.5 merupakan ilustrasi dari cara kerja YOLO



Gambar 2. 5 Ilustrasi Deteksi Objek YOLO (Redmon et al., 2016)

Salah satu keunggulan dari algoritma YOLO adalah kemampuan untuk melakukan deteksi objek secara *Real-time* dengan kecepatan yang tinggi. Dalam YOLO, deteksi objek dipandang sebagai masalah regresi, sehingga algoritma ini tidak memerlukan alur yang rumit. YOLO hanya menjalankan *Convolutional Neural Network* pada gambar untuk memprediksi deteksi objek. Oleh karena itu, YOLO dapat memproses citra secara *Real-time* dengan waktu latensi kurang dari 25 milidetik (Pramestya, 2018). Selain itu, YOLO juga memiliki kemampuan untuk mendeteksi objek dengan ukuran dan bentuk yang berbeda-beda, serta mampu mendeteksi objek yang saling tumpang tindih dengan akurat.

2.10 *Vehicle Counting*

Vehicle Counting atau penghitung kendaraan merupakan sebuah teknik dalam *Computer Vision* yang dapat digunakan untuk menghitung jumlah kendaraan yang melintas pada suatu lokasi. Teknik ini biasanya dilakukan dengan menggunakan kamera CCTV yang dipasang di atas jalan raya atau lintasan kendaraan lainnya. Dalam *Vehicle Counting*, kamera akan merekam gambar kendaraan yang melewati lintasan tertentu, kemudian gambar tersebut akan diproses menggunakan teknik *Image Processing* dan *Computer Vision* untuk menghitung jumlah kendaraan yang telah melintas.

Teknik *Vehicle Counting* umumnya dilakukan dengan menggunakan algoritma *Deep Learning* seperti *Convolutional Neural Network* (CNN) dan *You Only Look Once* (YOLO). Dalam penggunaan algoritma YOLO, gambar dari kamera akan dibagi-bagi menjadi beberapa *grid*, kemudian setiap *grid* akan diproses menggunakan teknik CNN untuk mengenali kendaraan pada gambar tersebut. Hasil dari pengenalan tersebut kemudian akan digunakan untuk menghitung jumlah kendaraan yang melewati suatu lintasan pada suatu periode waktu tertentu.

Vehicle Counting sangat penting untuk banyak aplikasi dunia nyata, seperti manajemen lalu lintas (Zhang et al., 2017). Dengan menggunakan teknik ini, pihak berwenang dapat memantau dan menganalisis tingkat lalu lintas kendaraan pada suatu jalan, menentukan kecepatan rata-rata kendaraan, dan mengidentifikasi jam-jam sibuk yang memerlukan peningkatan kapasitas jalan atau tindakan pengaturan lalu lintas. Selain itu, teknik ini juga dapat digunakan untuk memantau kondisi lalu lintas pada waktu tertentu dan menentukan pola penggunaan jalan dalam jangka waktu tertentu.

2.11 *Confusion Matrix*

Confusion Matrix adalah alat penting dalam evaluasi klasifikasi yang digunakan untuk mengukur performa model dalam memprediksi kelas target. *Confusion Matrix* digunakan sebagai ukuran objektif untuk mengukur kualitas pendeteksian objek dengan akurat (Wu, 2022). *Confusion Matrix* berisi data tentang hasil klasifikasi yang sesungguhnya dan hasil klasifikasi yang diprediksi oleh model. Matriks ini terdiri dari empat komponen utama, yaitu *true positive* (TP), *true negative* (TN), *false positive* (FP), dan *false negative* (FN). Pada Tabel 2.2 berikut merupakan contoh dari tabel *Confusion Matrix*.

Tabel 2. 2 Contoh Tabel *Confusion Matrix*

		Aktual	
		Kelas 1	Kelas 2
Prediksi	Kelas 1	TP	FP
	Kelas 2	FN	TN

Keterangan:

1. TP mewakili jumlah sampel yang benar-benar diklasifikasikan dengan benar sebagai positif.
2. TN mewakili jumlah sampel yang benar-benar diklasifikasikan dengan benar sebagai negatif.
3. FP mewakili jumlah sampel yang seharusnya diklasifikasikan sebagai negatif tetapi salah diklasifikasikan sebagai positif.
4. FN mewakili jumlah sampel yang seharusnya diklasifikasikan sebagai positif tetapi salah diklasifikasikan sebagai negatif.

Confusion Matrix memberikan gambaran yang jelas tentang sejauh mana model klasifikasi mampu membedakan kelas yang berbeda . Dengan menggunakan matriks kebingungan, kita dapat menghitung beberapa metrik evaluasi klasifikasi yang penting, seperti akurasi (*accuracy*), presisi (*precision*), *Recall* (*Recall*), dan nilai F1 (*F1-Score*). Berikut penjelasannya:

A. *Accuracy*

Akurasi (*Accuracy*) mengukur sejauh mana model mampu mengklasifikasikan dengan benar seluruh sampel. Dengan rumus sebagai berikut:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad ..(2.1)$$

B. *Precision*

Presisi (*Precision*) mengukur sejauh mana model memberikan hasil positif yang benar dari seluruh hasil yang diprediksi positif. Dengan rumus sebagai berikut:

$$Precision = \frac{TP}{TP+FP} = \frac{TP}{Total\ Prediksi\ Positif} \quad ..(2.2)$$

C. *Recall*

Recall mengukur sejauh mana model mampu mendeteksi secara benar semua sampel yang sebenarnya positif. Dengan rumus sebagai berikut:

$$Recall = \frac{TP}{TP+FN} = \frac{TP}{Total\ Data\ Positif} \quad ..(2.3)$$

D. *F1-Score*

Nilai *F1-Score* merupakan ukuran gabungan dari *Precision* dan *Recall*. Dengan rumus sebagai berikut:

$$F1 - Score = 2 \times \left(\frac{(Precision \times Recall)}{(Precision+Recall)}\right) \quad ..(2.4)$$

BAB III
METODOLOGI PENELITIAN

3.1 Metode Penelitian

Untuk memperjelas arah penelitian, pelaksanaan penelitian ini dilakukan melalui beberapa tahap. Adapun tahapan yang dilakukan oleh penulis untuk menyelesaikan penelitian ini adalah sebagai berikut:

1. Perumusan masalah

   Pada tahap perumusan masalah, langkah awal yang dilakukan adalah mengidentifikasi tema penelitian yang ingin diangkat serta masalah-masalah yang perlu dicari solusinya.

2. Landasan teori

   Pada tahap landasan teori, penulis akan melakukan studi teori yang berkaitan dengan tema penelitian yaitu deteksi kendaraan menggunakan teknologi *Computer Vision* dan algoritma YOLOv8.

3. Pengumpulan data

   Data dikumpulkan dari periode waktu tertentu, kemudian dipilih dan diolah untuk digunakan pada tahapan selanjutnya.

4. Pengolahan dan pelabelan data

   Data yang telah dikumpulkan kemudian diolah dan diberi label untuk selanjutnya di lakukan pelatihan model.

5. Pelatihan model

   Setelah proses pengolahan dan pelabelan data selesai dilakukan, langkah selanjutnya adalah melakukan pelatihan model menggunakan data tersebut dengan algortima YOLOv8.

6. Pengujian model

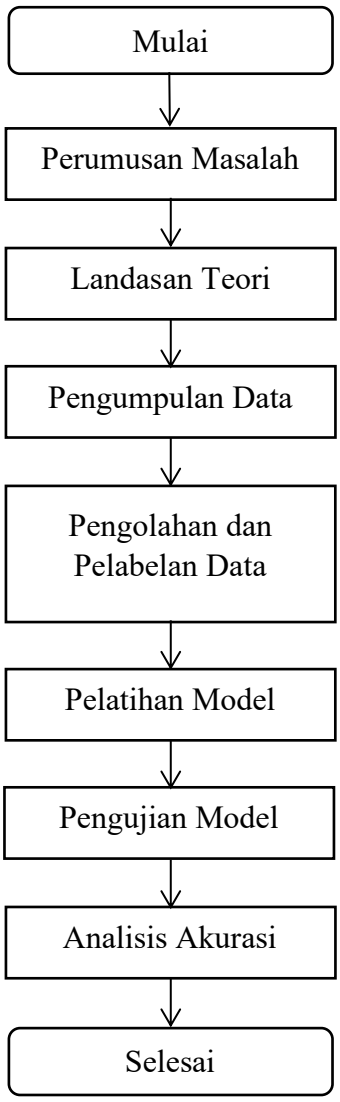
   Setelah model selesai dilatih kemudian dilakukan pengujian model dengan data rekaman CCTV jalan raya di Bandar Lampung yang belum pernah di lihat oleh model sebelumnya

7. Analisis akurasi

   Melakukan analisis terhadap tingkat akurasi model dalam menghitung kendaraan di jalan raya untuk mengetahui seberapa akurat hasil pengujian yang telah dilakukan.

3.2 Diagram Alir Penelitian

Diagram alir penelitian yang digunakan dalam penelitian ini dapat dilihat pada Gambar 3.1 sebagai berikut:



Gambar 3. 1 Diagram Alir Penelitian

3.2.1 Tahap Perumusan Masalah

Dalam penelitian ini, tema yang diangkat adalah pendeteksian kendaraan, dan masalah yang akan dipecahkan adalah bagaimana tingkat akurasi penghitung kendaraan menggunakan teknologi *Computer Vision* dengan algoritmaa YOLOv8 pada CCTV jalan raya di kota bandar lampung serta faktor apa saja yang mempengaruhi tingkat akurasi nya. Selain itu, pada tahap ini juga ditentukan tujuan dari penelitian, yakni untuk mengevaluasi akurasi penghitung kendaraan dan

mengevaluasi faktor-faktor apa saja yang dapat mempengaruhi akurasi penghitungan kendaraan. Dengan langkah perumusan masalah yang jelas dan terperinci, diharapkan penelitian dapat dilaksanakan dengan fokus dan tujuan yang jelas, sehingga dapat menghasilkan hasil yang berkualitas.

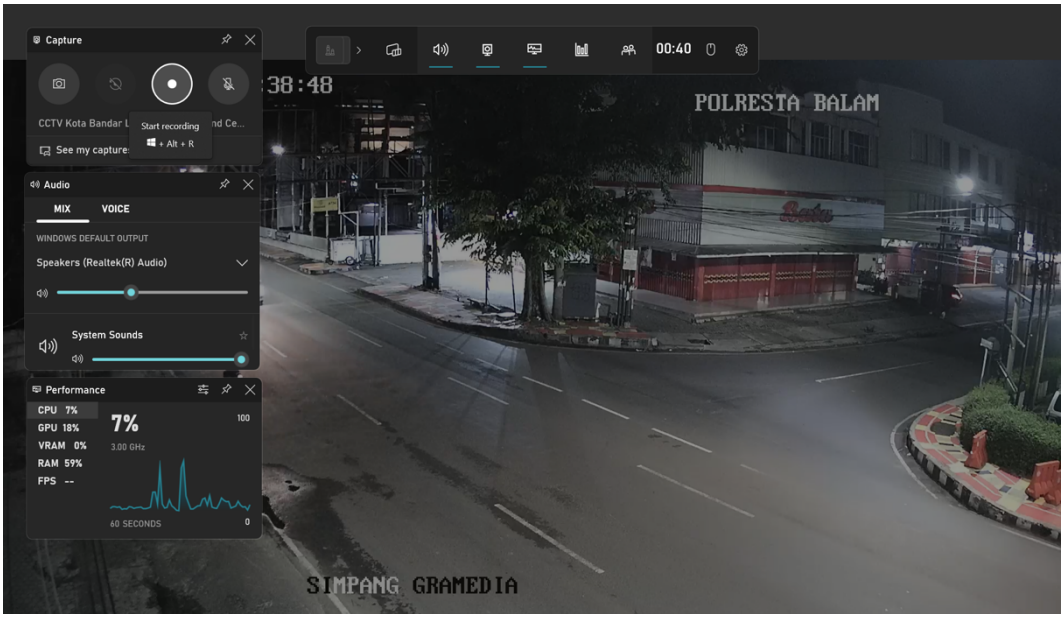
3.2.2 Tahap Landasan Teori

Tahap ini bertujuan untuk memperoleh pemahaman yang mendalam mengenai konsep dasar, prinsip, serta teknik yang berkaitan dengan deteksi kendaraan menggunakan teknologi *Computer Vision* dan algoritma YOLOv8. Selain itu, penulis juga akan mengkaji pustaka penelitian terdahulu yang berkaitan dengan deteksi kendaraan untuk memperoleh referensi yang dapat digunakan dalam membangun pendeteksi kendaraan yang efektif.

Untuk mempelajari teori yang berkaitan dengan tema penelitian, penulis akan mengumpulkan berbagai sumber informasi seperti artikel, buku, dan video yang berhubungan dengan deteksi kendaraan. Sumber informasi ini akan diperoleh dari berbagai situs seperti medium.com dan sejenisnya, *Youtube*, serta situs-situs akademik seperti *Google Scholar*, dan *Research Gate*. Penulis akan mengumpulkan dan mempelajari berbagai sumber informasi yang relevan dan berkualitas untuk memastikan bahwa landasan teori yang digunakan dalam penelitian ini akurat dan komprehensif.

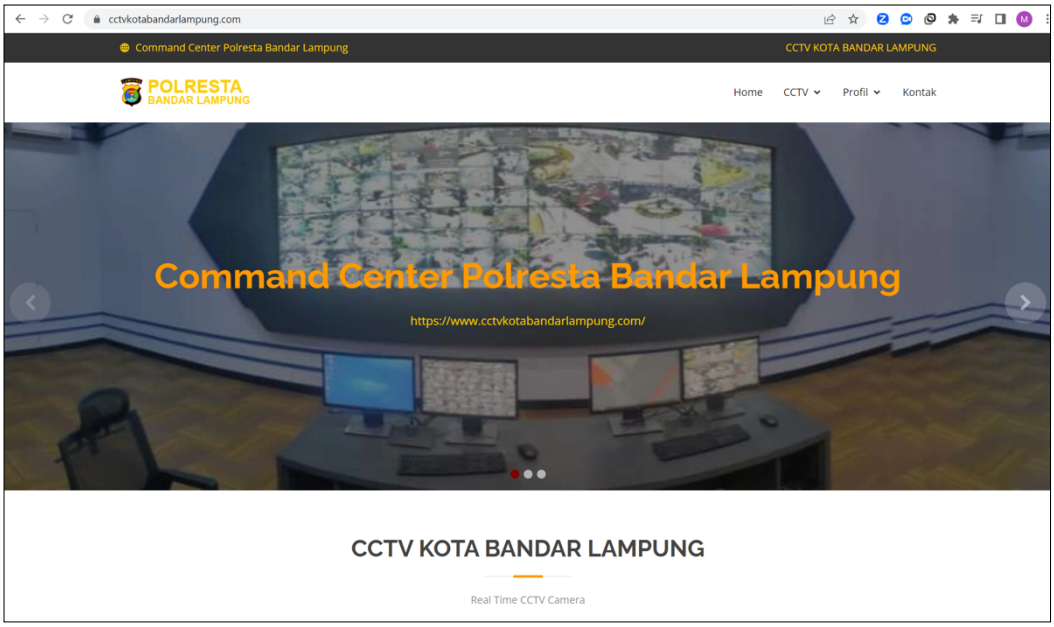
3.2.3 Tahap Pengumpulan Data

Pada penelitian ini, pengumpulan data dilakukan dengan cara merekam video dari CCTV menggunakan *software xbox game bar*.

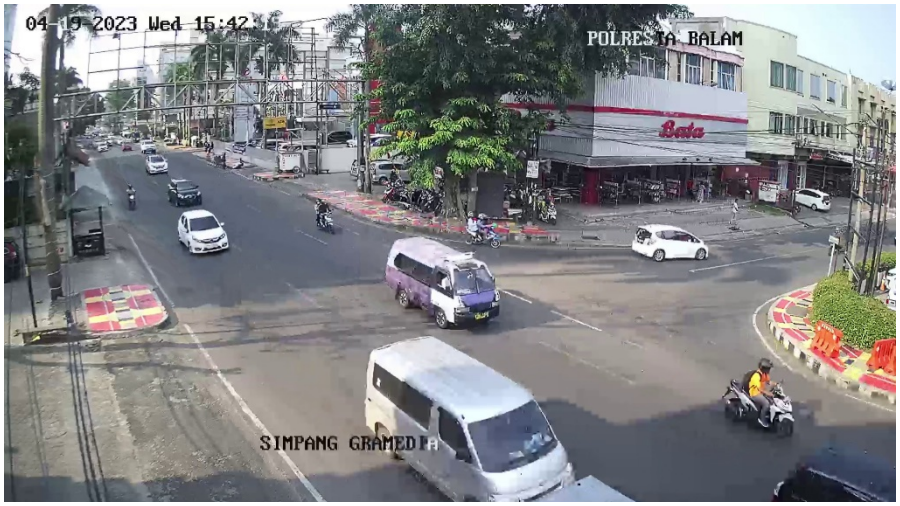


Gambar 3. 2 Proses Pengumpulan Data

Data rekaman CCTV yang dikumpulkan berasal dari situs *website* https://www.cctvkotabandarlampung.com dan direkam selama 10 menit dari sumber CCTV Simpang Gramedia.



Gambar 3. 3 Tampilan *Website* CCTV Kota Bandar Lampung



Gambar 3. 4 Tampilan Rekaman CCTV Simpang Gramedia

Tujuan dari pengumpulan data ini adalah untuk mendapatkan data yang mencakup berbagai jenis kendaraan yang melintas di jalan raya, sehingga dapat digunakan untuk melatih model deteksi kendaraan yang akan dibangun. Proses pengambilan data dilakukan dengan cermat dan terencana untuk memastikan kualitas data yang baik dan representatif.

3.2.4 Tahap Pengolahan dan Pelabelan Data

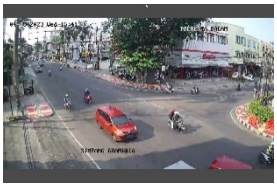
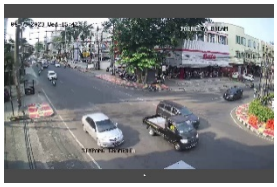
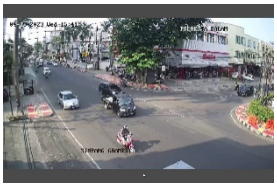
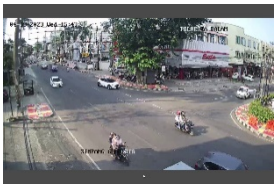
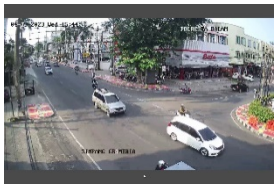
Tahap pengolahan dan pelabelan data bertujuan untuk membuat *Dataset* yang akan digunakan untuk pelatihan model, pada tahap ini data rekaman CCTV yang telah dikumpulkan diproses menggunakan *script* python dan dijalankan di jupyter *notebook* dengan memanfaatkan *library open cv* untuk memecah video menjadi *frame per frame* berupa gambar untuk memudahkan pelabelan. Berikut adalah *script* python yang dijalankan di jupyter *notebook* untuk memproses video menjadi gambar *frame per frame*.

```
In [ ]: import cv2

In [ ]: vid = cv2.VideoCapture("simpang_gramedia.mp4")

In [17]: # Function to save image frame from video
def getFrame(sec):
    vid.set(cv2.CAP_PROP_POS_MSEC,sec*1000)
    hasFrames,image = vid.read()
    if hasFrames:
        cv2.imwrite("image"+str(count)+".jpg", image) # save frame as JPG file
    return hasFrames

sec = 0
frameRate = 1 #capture image in each 1 second
count=1
success = getFrame(sec)
while success:
    count = count + 1
    sec = sec + frameRate
    sec = round(sec, 2)
    success = getFrame(sec)
```

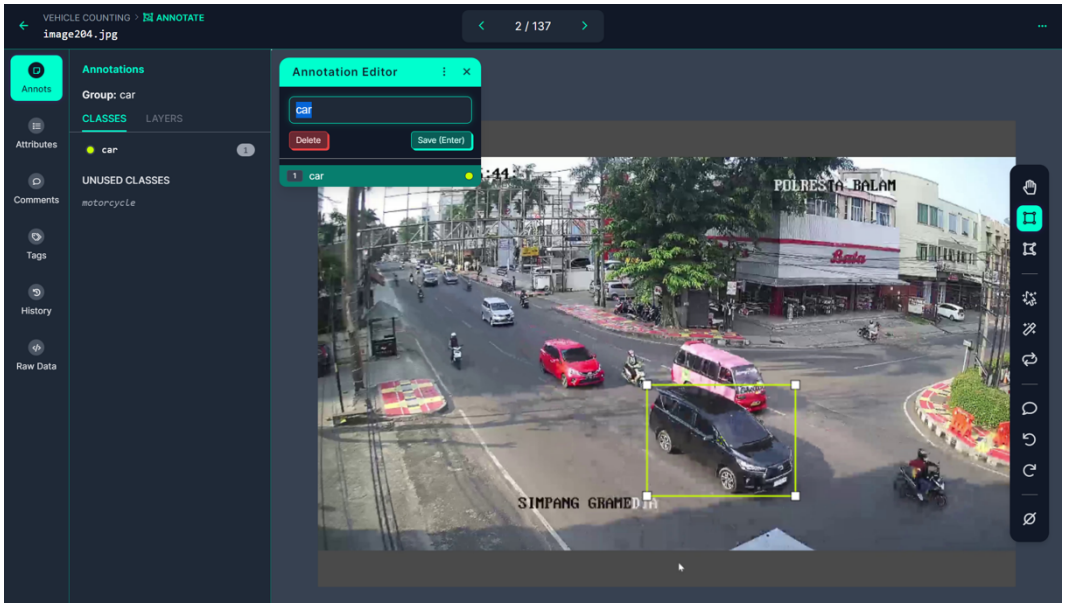
Gambar 3. 5 *Script* Python Untuk Memproses Video Menjadi Gambar

Dari pemrosesan tersebut, video yang berdurasi 10 menit menghasilkan sebanyak 600 gambar. Hasil *output* setelah di lakukan pemecahan *frame* menggunakan *library open cv* dapat dilihat pada Tabel 3.1 berikut.

Tabel 3. 1 *Output* Hasil Konversi Video ke Gambar

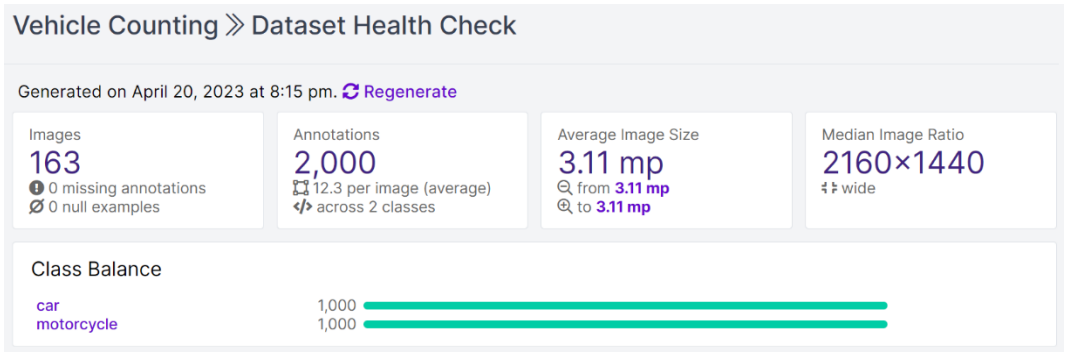
		
		

Setelah dilakukan pemecahan *frame* video menjadi gambar, selanjutnya dilakukan pelabelan data di *platform* Roboflow. Dalam pelabelan data, terdapat 2 jenis label yang digunakan untuk mengidentifikasi kendaraan yaitu label *car* dan *motorcycle*. Dalam proses pelatihan model, nanti nya hanya akan digunakan 1000 label *car* dan 1000 label *motorcycle* untuk dibuat *Dataset* pelatihan. Proses pelabelan data di *platform* Roboflow dapat dilihat pada gambar berikut.



Gambar 3. 6 Proses Pelabelan Data di *Platform* Roboflow

Proses pelabelan data dilakukan dengan hati-hati dan teliti untuk memastikan setiap objek kendaraan teridentifikasi dengan tepat. Berikut adalah gambar hasil dari pengecekan kesehatan *Dataset* pada *platform* Roboflow.



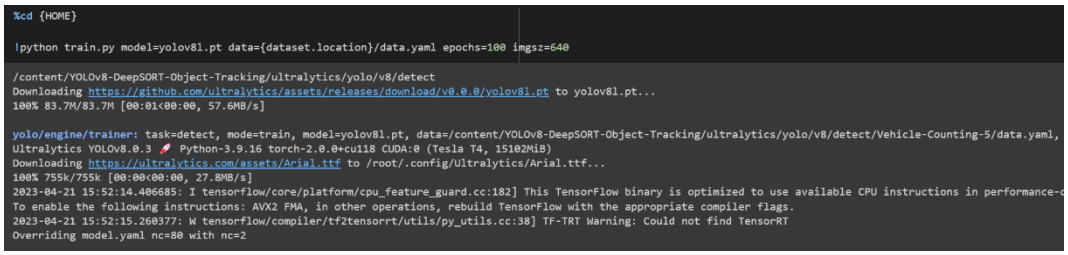
Gambar 3. 7 *Dataset Health Check*

Setelah proses pelabelan data selesai dilakukan, maka selanjutnya akan dibuatkan *Dataset*. *Dataset* akan berisi total 2000 label objek kendaraan yang terbagi menjadi 2 yaitu 1000 label untuk kendaraan bermotor (*Motorcycle*), dan 1000 label untuk kendaraan mobil (*Car*). Pada *Dataset* akan dilakukan pembagian data sebanyak 70% gambar untuk *set* pelatihan, 20% gambar untuk *set* validasi, dan 10% gambar untuk *set* uji. Pada *Dataset* juga akan diterapkan augmentasi gambar.

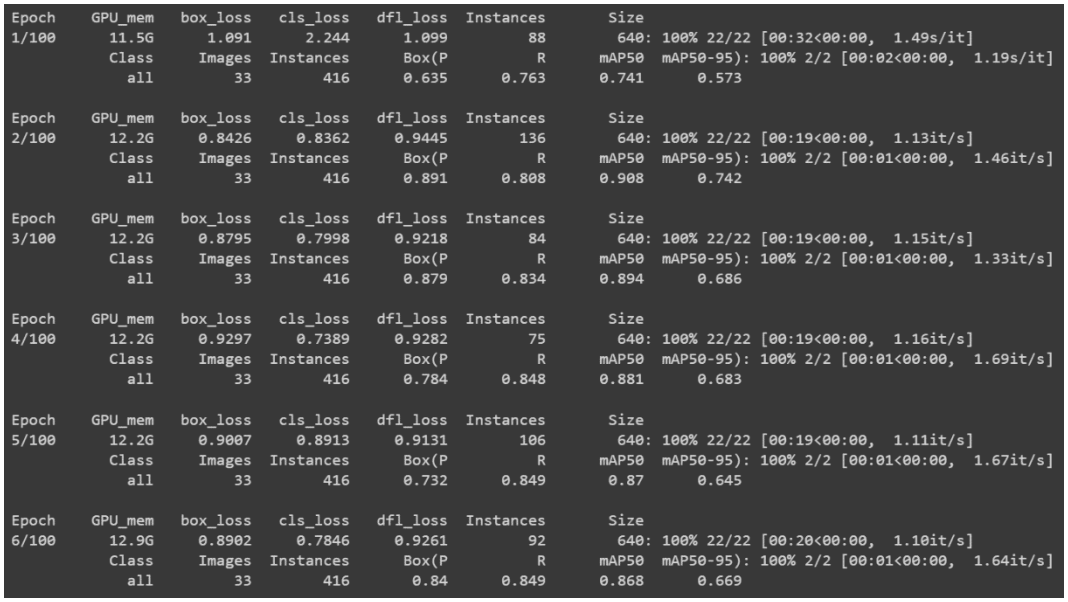
3.2.5 Pelatihan Model

Setelah *Dataset* selesai dibuat, selanjutnya *Dataset* tersebut akan dilatih menggunakan algoritma YOLOv8 dengan menggunakan bahasa pemrograman Python pada *platform* Google Colaboratory. YOLOv8 merupakan salah satu algoritma terbaru dalam deteksi objek, yang memiliki kemampuan deteksi yang lebih baik dan lebih cepat dibandingkan algoritma sebelumnya.

Pelatihan model dilakukan dengan mengatur parameter *Epochs* untuk mengoptimalkan hasil pelatihan. Selama pelatihan, model akan belajar untuk mengenali objek yang telah diberi label pada *set* latih dan akan divalidasi menggunakan data pada *set* validasi. Hasil pelatihan tersebut akan digunakan untuk menguji keakuratan deteksi objek pada data uji yang belum pernah dilihat sebelumnya. Berikut adalah gambaran proses pelatihan model pada *platform Google Colaboratory*.



Gambar 3. 8 Penyetelan Parameter *Epoch*



Gambar 3. 9 Proses Pelatihan Model

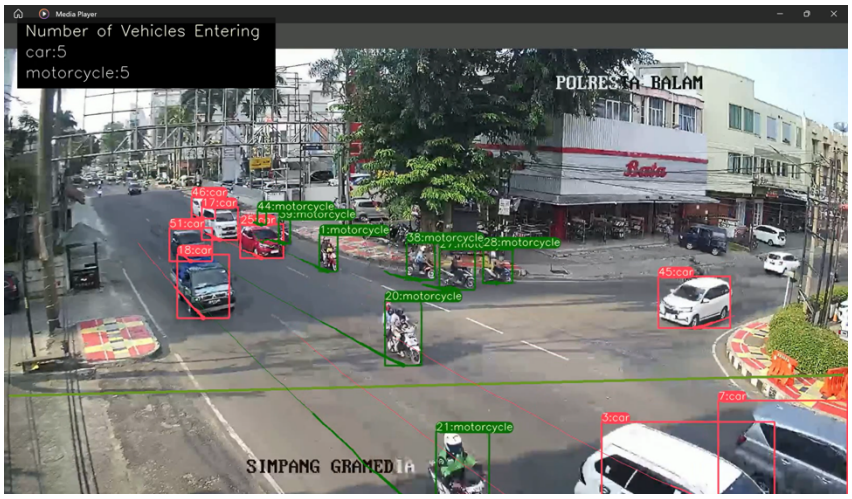
Setelah model selesai dilatih maka selanjutnya akan dilakukan pengujian pada *set* validasi dan *set* uji, jika hasil pendeteksian masih dirasa kurang maka akan di lakukan pelatihan model ulang dengan menyetel ulang parameter *Epochs*. Hal ini dilakukan agar mendapatkan model dengan akurasi terbaik sehingga siap untuk di uji pada data rekaman CCTV yang sebelumnya belum pernah di lihat oleh model.

3.2.6 Tahap Pengujian Model

Setelah proses pelatihan model selesai dilakukan, maka pengujian model siap dilakukan. Pada tahap ini model di uji langsung menggunakan video rekaman CCTV yang belum pernah dilihat sebelumnya. Video CCTV untuk pengujian model berdurasi 30 detik. Pengujian model juga dilakukan menggunakan *platform Google Colaboratory*. Berikut tampilan proses pengujian model pada *platform Google Colaboratory*.

```
!python predictc.py model='/content/drive/MyDrive/skripsi/model/best.pt' source='/content/drive/MyDrive/skripsi/sgtest.mp4'

[2023-04-21 22:59:12,093][root.tracker][INFO] - Loading weights from deep_sort_pytorch/deep_sort/deep/checkpoint/ckpt.t7... Done!
2023-04-21 22:59:12.567073: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use availab
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-04-21 22:59:13.444372: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Ultralytics YOLOv8.0.3 🚀 Python-3.9.16 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Fusing layers...
Model summary: 268 layers, 43608150 parameters, 0 gradients, 164.8 GFLOPs
video 1/1 (1/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 4 cars, 2 motorcycles, 62.3ms
video 1/1 (2/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 3 cars, 2 motorcycles, 49.8ms
video 1/1 (3/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 3 cars, 2 motorcycles, 49.8ms
video 1/1 (4/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 3 cars, 2 motorcycles, 49.7ms
video 1/1 (5/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 3 cars, 2 motorcycles, 35.9ms
video 1/1 (6/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 2 cars, 2 motorcycles, 37.1ms
video 1/1 (7/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 2 cars, 2 motorcycles, 34.9ms
video 1/1 (8/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 2 cars, 2 motorcycles, 34.9ms
video 1/1 (9/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 2 cars, 2 motorcycles, 37.7ms
video 1/1 (10/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 2 cars, 2 motorcycles, 33.1ms
video 1/1 (11/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 2 cars, 2 motorcycles, 36.3ms
video 1/1 (12/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 2 cars, 2 motorcycles, 26.2ms
video 1/1 (13/918) /content/drive/MyDrive/skripsi/sgtest.mp4: 448x640 2 cars, 2 motorcycles, 26.0ms
```

Gambar 3. 10 Proses Pengujian Model



Gambar 3. 11 *Output* Pengujian Model

3.2.7 Tahap Analisis Akurasi

Setelah semua proses dilalui dan menghasilkan *output* pengujian model, maka selanjutnya akan dilakukan Analisa terhadap *output* dari pengujian model. Pada tahap ini akan dilakukan analisis terhadap akurasi pendeteksian kendaraan dan penghitungan kendaraan. Metrik yang akan digunakan untuk menganalisis akurasi pendeteksian pada *output* pengujian model adalah *Accuracy, Precision*, *Recall* dan *F1-Score*. Sedangkan untuk menganalisis seberapa akurat penghitungan kendaraan, akan dilakukan pembandingan antara penghitungan kendaraan otomatis dengan penghitungan kendaraan secara manual, dan dihitung nilai *Error Relative* nya.

Hal pertama yang akan dilakukan untuk menganalisis akurasi pendeteksian kendaraan adalah memecah video *output* pengujian menjadi *frame per frame* berupa gambar, kemudian dibuatkan tabel *Confusion Matrix* seperti pada Tabel 3.2.

Tabel 3. 2 *Confusion Matrix*

		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	TP	FP
	Car	FN	TN

Tabel ini akan menyajikan hasil prediksi model dalam bentuk matriks, *Confusion Matrix* menyediakan informasi tentang berapa banyak data yang diklasifikasikan dengan benar (*true positive* dan *true negative*) serta berapa banyak data yang salah (*false positive* dan *false negative*). Dari tabel *Confusion Matrix* dapat dihitung nilai *Accuracy, Precision*, *Recall*, dan *F1-Score* dengan persamaan berikut.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad ..(2.1)$$

$$Precision = \frac{TP}{TP+FP} \quad ..(2.2)$$

$$Recall = \frac{TP}{TP+FN} \quad ..(2.3)$$

$$F1 - Score = 2 \times \left(\frac{(Precision \times Recall)}{(Precision + Recall)}\right) \quad ..(2.4)$$

Accuracy (Akurasi) mengukur sejauh mana model klasifikasi mampu memprediksi dengan benar semua kelas yang ada dalam *Dataset*. Dalam konteks deteksi objek atau klasifikasi, akurasi mengukur sejauh mana model mampu mengidentifikasi dengan benar seluruh kelas yang ada, termasuk kelas positif dan negatif. Akurasi dihitung sebagai rasio antara jumlah prediksi benar (TP+TN) dengan total jumlah sampel atau data dalam *Dataset*. *Precision* (presisi) menggambarkan seberapa akurat model dalam mengidentifikasi kelas positif dari hasil prediksinya. *Precision* dihitung sebagai rasio antara *true positive* (TP) dan total hasil prediksi positif (TP + FP), di mana TP adalah jumlah data positif yang diklasifikasikan dengan benar dan FP adalah jumlah data negatif yang salah diklasifikasikan sebagai positif. *Recall* (ingatan) mengukur seberapa baik model dalam mengidentifikasi seluruh data positif yang ada dalam *Dataset* . *Recall* dihitung sebagai rasio antara TP dan total data positif (TP + FN), di mana FN adalah jumlah data positif yang salah diklasifikasikan sebagai negatif. *F1-Score* (skor F1) adalah suatu metrik yang menggabungkan antara *precision* dan *Recall* menjadi satu angka yang mencerminkan keseimbangan antara keduanya. *F1-Score* dihitung sebagai rata-rata harmonik antara *precision* dan *Recall*, dan dapat memberikan gambaran yang holistik tentang kinerja keseluruhan model klasifikasi. Penggunaan keempat metrik ini akan memberikan informasi yang komprehensif tentang akurasi, ketelitian, dan kelengkapan model pada penelitian ini, dan dapat menjadi dasar untuk mengambil kesimpulan yang tepat dan akurat terhadap hasil penelitian.

Selanjutnya akan dilakukan pula analisis terhadap tingkat akurasi penghitungan kendaraan. Untuk menilai seberapa akurat model dalam menghitung kendaraan maka akan digunakan rumus *Error Relative* dengan persamaan sebagai berikut.

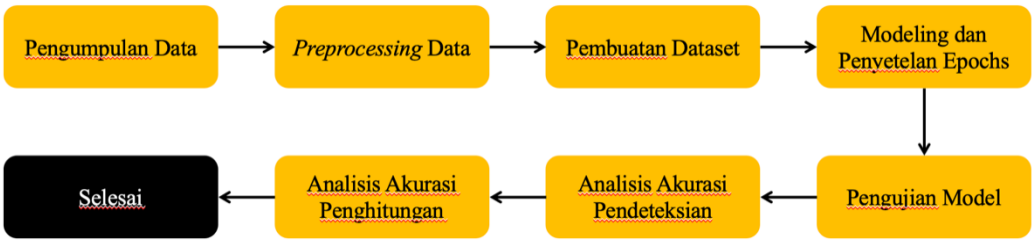
$$Error\ Relative\ \% = \frac{|Nilai\ Sebenarnya - Nilai\ Penghitungan|}{|Nilai\ Sebenarnya|}\ x\ 100\% \quad ..(2.5)$$

Dengan menggunakan rumus diatas maka akan diketahui persentase *error* penghitungan kendaraan.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi

Pada bab ini, akan dipaparkan hasil dari penelitian yang telah dilaksanakan dengan cermat dan sistematis. Sebagaimana telah dipaparkan dalam bab sebelumnya, penelitian ini dilaksanakan untuk menginvestigasi akurasi dari penghitung kendaraan menggunakan teknologi *Computer Vision* dengan penerapan algoritma YOLOv8 pada kamera CCTV yang terpasang di jalan raya di Kota Bandar Lampung. Berikut adalah gambar yang menjelaskan alur proses permodelan sampai dengan Analisis akurasi Algoritma YOLOv8 yang diterapkan pada penelitian ini.



Gambar 4. 1 Alur Proses Permodelan Sampai Dengan Analisis Akurasi

Data dan informasi yang dihasilkan dari penelitian ini dapat bermanfaat dalam konteks pengembangan teknologi pendeteksian dan penghitungan kendaraan secara otomatis. Namun, tak dapat dielakkan bahwa penelitian ini juga menghadapi sejumlah tantangan dan keterbatasan, yang akan dipaparkan dengan cermat dalam bab hasil dan pembahasan ini. Dengan demikian, pada bab ini akan diulas mengenai temuan-temuan yang relevan dan peneliti akan memberikan interpretasi yang akurat, sehingga dapat memberikan kontribusi yang bermakna dalam ranah ilmu *Computer Vision* dan aplikasinya dalam penghitungan kendaraan.

4.1.1 Pengumpulan Data dan Pembuatan Dataset

Pengumpulan data dilakukan dengan cara merekam *Streaming* video CCTV yang berasal dari *website* resmi CCTV kota Bandar Lampung yang bisa di akses di https://www.cctvkotabandarlampung.com. Video CCTV direkam menggunakan *software* bawaan windows yaitu *software* xbox game bar. Pengumpulan data

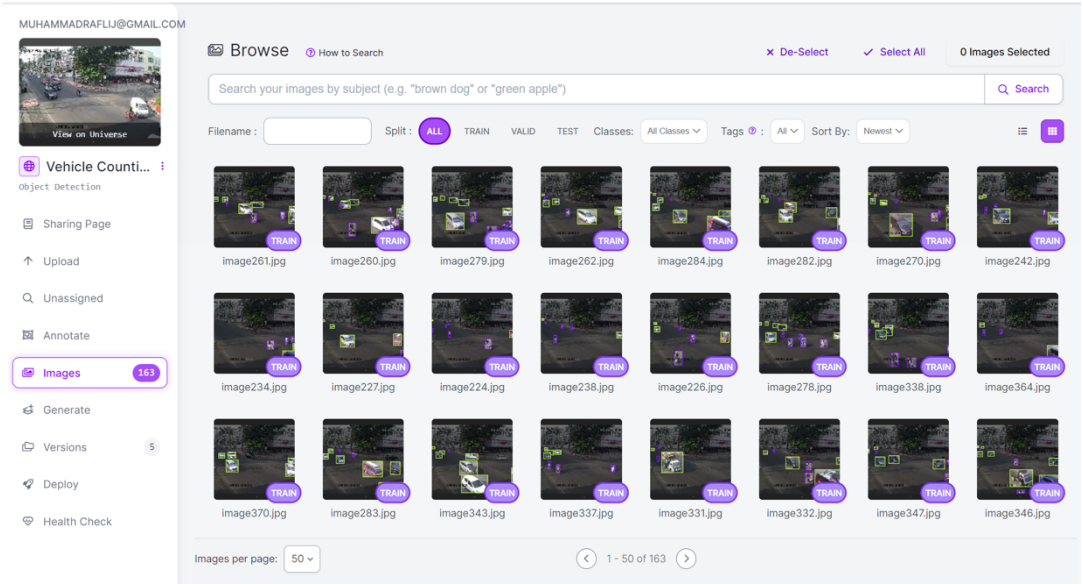
dilakukan pada sore hari tanggal 19 April 2023 pada jam 15.41 sampai dengan jam 15.50. Data yang dikumpulkan berisi video rekaman CCTV Simpang Gramedia berdurasi 10 menit.

Data yang telah dikumpulkan kemudian di proses menjadi sebuah *Dataset* yang akan digunakan dalam pelatihan model. Data mentah yang masih berbentuk video di pecah menjadi *frame-per-frame* dengan *framerate* 1 detik menggunakan *script* python, data berbentuk video berdurasi 10 menit menghasilkan 575 gambar. *Framerate* yang disetel saat menkonversi video menjadi gambar adalah 1 detik yang berarti video akan di pisah perdetik menjadi gambar, seharusnya dari 10 menit video akan menghasilkan 600 gambar, namun disini peneliti hanya mendapatkan 575 gambar, hal ini terjadi karena video mengalami *freeze* pada menit ke 9.59. berikut adalah gambar *output* video setelah dilakukan pengkonversian menjadi gambar perdetik.

Gambar 4. 2 *Output* Pengkonversian Video Menjadi Gambar

Setelah proses pengkonversian video menjadi rangkaian gambar-gambar berhasil dilakukan, langkah selanjutnya adalah melakukan pelabelan yang teliti terhadap setiap gambar. Dalam upaya membangun *Dataset* yang kaya dan representatif, peneliti merancang *Dataset* yang terdiri dari 1000 label kendaraan tipe mobil dengan label Bernama (*car*) dan 1000 label kendaraan tipe sepeda motor dengan label Bernama (*motorcycle*). Dengan perpaduan ini, keseluruhan *Dataset*

terdiri dari 2000 label kendaraan. Proses penyiapan *Dataset* ini melibatkan proses pemilihan 163 gambar saja dari total 575 gambar hasil dari langkah sebelumnya, yakni pengonversian video ke gambar-gambar. Peneliti melakukan pemilihan gambar secara selektif untuk memastikan bahwa *Dataset* yang digunakan dalam penelitian ini mencerminkan keragaman kendaraan yang ada di jalan raya. Dalam proses pelabelan data, peneliti melakukannya dengan langkah yang sangat teliti untuk memastikan bahwa setiap kendaraan dapat diidentifikasi dengan ketepatan. Berikut adalah tampilan *platform* Roboflow yang digunakan sebagai alat pelabelan serta pembuatan *Dataset* yang efektif dan efisien.



Gambar 4. 3 *Platform* Roboflow

Setelah pelabelan gambar telah selesai dilakukan dengan teliti, langkah berikutnya adalah mengelola *Dataset* yang telah terbentuk. *Dataset* yang berisi total 163 gambar dengan 2000 label kendaraan di bagi menjadi 3 bagian, masing-masing berfungsi untuk tujuan yang berbeda. Bagian pertama, yang merupakan sekitar 70% dari keseluruhan *Dataset*, diperuntukkan untuk proses pelatihan model. Kemudian, bagian kedua, sebesar 20%, dialokasikan untuk validasi model. Bagian terakhir, yakni 10% dari *Dataset*, akan digunakan untuk menguji kinerja model dalam tahap uji coba.

Setelah melakukan pembagian *Dataset*, selanjutnya masuk ke tahap *preprocessing* dan augmentasi gambar. Proses *preprocessing* melibatkan serangkaian langkah untuk mempersiapkan data sebelum memasukkannya ke

dalam model. Selain itu, juga diterapkan augmentasi gambar, yaitu teknik untuk memperluas variasi data dengan membuat variasi minor dalam gambar asli. Hal ini membantu model untuk lebih tanggap terhadap variasi yang ada dalam *Dataset* sebenarnya, sehingga meningkatkan kemampuan generalisasi dan akurasi model dalam mengenali objek kendaraan dalam berbagai situasi. Proses ini menjadi kunci dalam meningkatkan kinerja keseluruhan model deteksi yang akan dibuat. hal-hal yang diterapkan pada tahap *preprocessing* dan augmentasi gambar tercantum pada Tabel 4.1.

Tabel 4. 1 *Preprocessing* dan Augmentasi Gambar

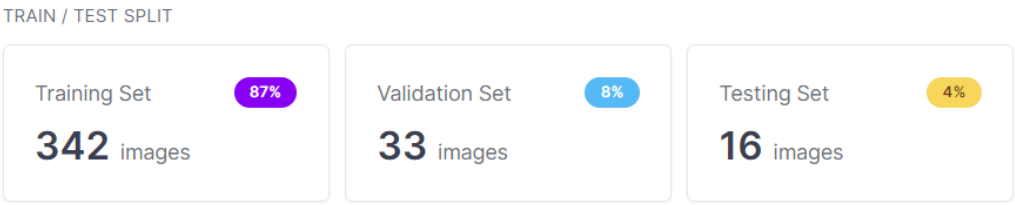
Preprocessing	**Resize**: Stretch to 640x640
Augmentations	***Output*s per training example**: 3
	Grayscale: Apply to 15% of images
	Saturation: Between -40% and +40%
	Brightness: Between -35% and +35%
	Blur: Up to 1.25px
	Noise: Up to 2% of pixels

Pada tahap *preprocessing*, setiap gambar yang awalnya memiliki dimensi 2160 x 1440 piksel di *resize* menjadi 640 x 640 piksel. Penyesuaian ukuran ini dilakukan karena *input* model YOLOv8 membutuhkan ukuran yang konsisten, dan ini merupakan langkah penting untuk memastikan bahwa gambar-gambar dalam *Dataset* memiliki dimensi yang sesuai dengan tuntutan model.

Setelah tahap *resizing* selesai, maka dilanjutkan ke tahap augmentasi gambar. Tujuan utama dari augmentasi ini adalah untuk memperkaya variasi data dalam *Dataset*. Oleh karena itu, dilakukan manipulasi pada gambar dengan menerapkan berbagai teknik augmentasi yang tercantum dalam Tabel 4.1 diatas. Hal ini bertujuan untuk menciptakan variasi yang lebih besar dalam *Dataset*, sehingga model dapat lebih baik dalam mengenali objek dalam berbagai situasi yang mungkin terjadi dalam penggunaan sehari-hari.

Dampak dari penerapan augmentasi gambar sangat signifikan. *Dataset* awal yang terdiri dari 163 gambar tumbuh menjadi 391 gambar setelah proses augmentasi. *Dataset* yang lebih besar ini memberikan model lebih banyak informasi untuk dipelajari, sehingga model dapat lebih baik dalam menangkap ciri-

ciri yang relevan dari objek kendaraan. Akhirnya *Dataset* telah selesai dibuat dan diperkaya melalui langkah-langkah *preprocessing* dan augmentasi, *Dataset* sekarang siap digunakan dalam tahap pelatihan model. Pada Gambar berikut, disajikan informasi tentang pembagian akhir *Dataset*, yang meliputi data *training*, validasi, dan tes, serta jumlah gambar dalam masing-masing bagian.



Gambar 4. 4 *Dataset Split*

4.1.2 Pelatihan Model

Dalam penelitian ini, pelatihan model deteksi kendaraan dilakukan dengan memanfaatkan *platform Google Colaboratory*. Peneliti melakukan penyetelan parameter pelatihan model dengan fokus pada penyesuaian jumlah *Epochs*. Model yang dipilih untuk pelatihan adalah YOLOv8L, sebuah model yang telah terbukti efektif dalam tugas deteksi objek. Tahap pelatihan dilakukan secara iteratif, dengan empat percobaan pelatihan yang masing-masing menggunakan jumlah *Epochs* yang berbeda-beda.

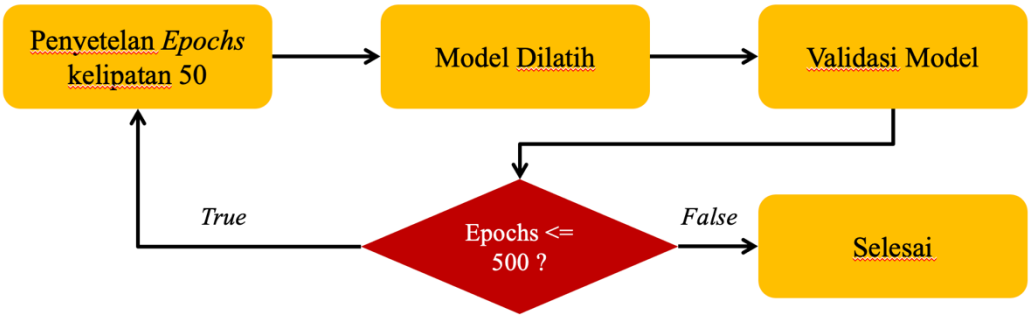
Setiap percobaan pelatihan dilakukan dengan jumlah *Epochs* yang berbeda untuk mengamati bagaimana perubahan tersebut mempengaruhi kinerja model. *Epochs* merupakan parameter yang menentukan berapa kali seluruh *Dataset* akan dilewatkan ke dalam model selama proses pelatihan. Dengan memvariasikan jumlah *Epochs*, selanjutnya akan di analisis bagaimana kinerja model berubah seiring dengan peningkatan jumlah iterasi pelatihan. Pada tabel berikut disajikan informasi mengenai pelatihan model dan nilai parameter *Epochs* yang digunakan.

Tabel 4. 2 *Epochs* dan Model yang Digunakan Saat Pelatihan Model

Pelatihan	***Epochs***	**Model**
1	50	YOLOv8L
2	100	YOLOv8L
3	150	YOLOv8L
4	200	YOLOv8L
5	250	YOLOv8L
6	300	YOLOv8L

7	350	YOLOv8L
8	400	YOLOv8L
9	450	YOLOv8L
10	500	YOLOv8L

Percobaan pertama menggunakan 50 *Epochs*, yang diikuti oleh percobaan dengan 100 *Epochs*, 150 *Epochs*, sampai dengan 500 *Epochs*. Berikut adalah gambar alur modeling dan penyetelan *Epochs*.



Gambar 4. 5 Alur Modeling dan Penyetelan *Epochs*

Tahap awal dalam pelatihan model melibatkan penyetelan parameter *Epochs* serta pemilihan model yang akan digunakan. Parameter *Epochs* penting dalam menentukan seberapa banyak iterasi pelatihan yang akan dilakukan pada model. Selain itu, pemilihan model juga memegang peranan penting, dalam penelitian ini digunakan model YOLOv8L. Proses penyesuaian parameter *Epochs* dan pemilihan model tersebut dilakukan dengan mempertimbangkan tujuan penelitian untuk mencapai hasil yang optimal dalam tugas pendeteksian kendaraan menggunakan CCTV.

Setelah tahap penyetelan parameter, langkah selanjutnya adalah menjalankan *script* python yang telah disiapkan untuk memulai proses pelatihan model. Proses pelatihan ini memungkinkan model untuk "belajar" dari *Dataset* dengan *cara* mengoptimalkan parameter-parameter internalnya sehingga dapat mengenali kendaraan pada gambar dengan akurasi yang lebih baik. Berikut adalah *script* python yang digunakan untuk melakukan pelatihan model.

```
[ ]  %cd {HOME}
     !python train.py model=yolov8l.pt data={dataset.location}/data.yaml epochs=100 imgsz=640
```

Gambar 4. 6 *Script* Python Pelatihan Model

Lama waktu yang diperlukan selama proses pelatihan model sangatlah bervariasi dan dipengaruhi oleh beberapa faktor, terutama ukuran *Dataset* yang digunakan dan nilai parameter *Epochs* yang telah diatur sebelumnya. Dalam konteks pelatihan model YOLOv8L pada penelitian ini, kita perlu memahami bahwa semakin besar ukuran *Dataset* dan semakin banyak nilai *Epochs* yang diterapkan, maka waktu pelatihan akan semakin lama.

Pada setiap *Epoch* yang dijalankan, model akan mengalami iterasi untuk mengoptimalkan parameter internalnya. Oleh karena itu, setiap *Epochs* akan membutuhkan waktu yang berbeda-beda tergantung pada kompleksitas *Dataset* dan model yang digunakan. Untuk memberikan gambaran yang lebih jelas mengenai waktu pelatihan pada masing-masing *Epochs*, berikut ini disajikan informasi terkait Waktu pelatihan model yang dibutuhkan pada masing-masing *Epochs* yang berbeda dalam Tabel 4.3 berikut.

Tabel 4. 3 Jumlah *Epochs* dan Waktu Pelatihan

Pelatihan	***Epochs***	**Waktu Pelatihan**
1	50	0.367 Jam
2	100	0.702 Jam
3	150	1.075 Jam
4	200	1.434 Jam
5	250	1.860 Jam
6	300	2.141 Jam
7	350	2.535 Jam
8	400	2.835 Jam
9	450	3.158 Jam
10	500	3.681 Jam

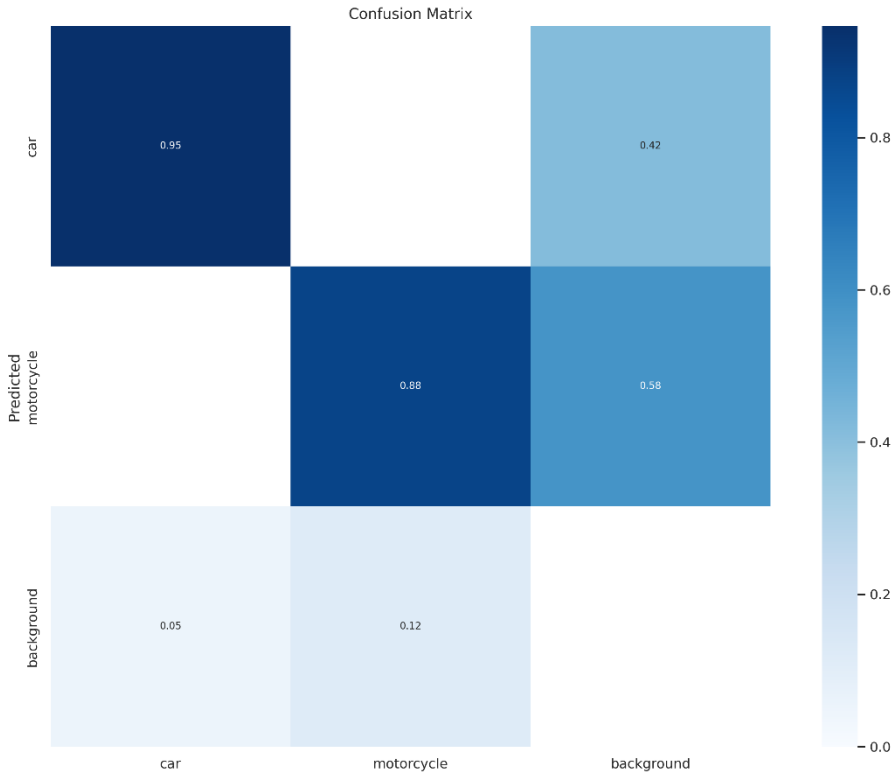
Tabel 4.3 di atas menggambarkan bahwa semakin tinggi nilai *Epochs* yang digunakan, waktu pelatihan pada setiap *Epochs* akan cenderung meningkat. Perlu diperhatikan bahwa tingkat kenaikan waktu pelatihan tidak bersifat linier, tetapi bisa mengalami lonjakan terutama pada tahap-tahap akhir pelatihan. Ini menunjukkan pentingnya penyesuaian parameter *Epochs* secara tepat sesuai dengan kebutuhan serta ketersediaan sumber daya komputasi. Dengan memperhatikan informasi ini, peneliti dapat mengatur strategi pelatihan yang sesuai untuk mencapai hasil yang diinginkan dengan efisien dalam hal waktu dan sumber daya yang digunakan. Hasil pelatihan model pada masing-masing nilai parameter *Epochs* yang berbeda dapat dilihat pada Tabel 4.4 sebagai berikut.

Tabel 4. 4 Hasil Pelatihan Model

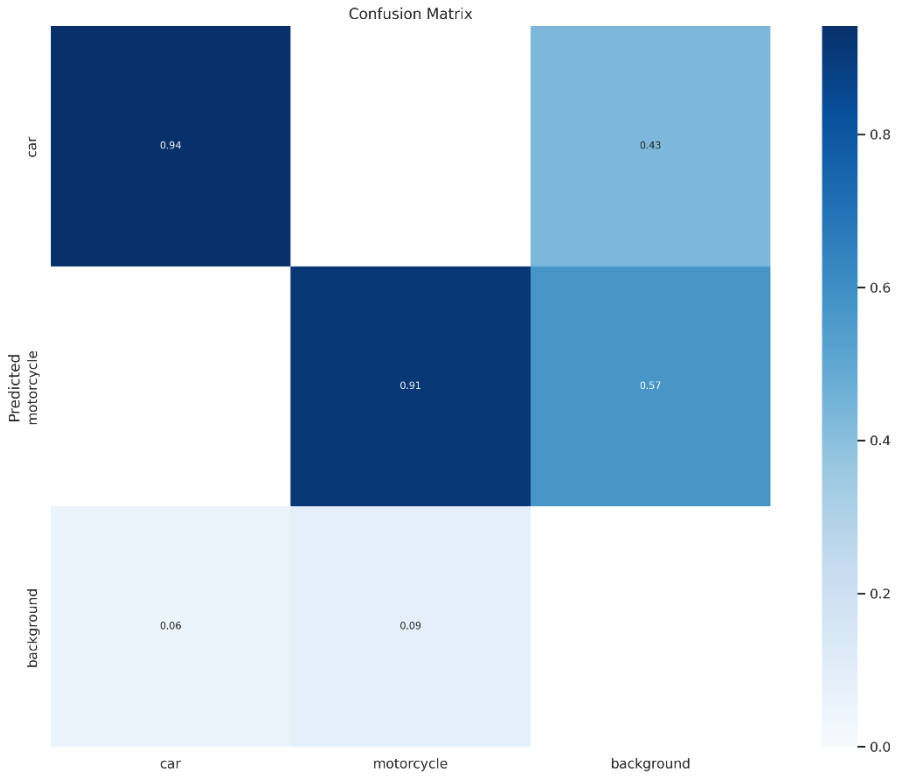
No	Epochs	Class	Box P	Box R	mAP50	mAP50-95
1	50	*All*	0.864	0.905	0.940	0.806
		Car	0.878	0.938	0.960	0.882
		Motorcycle	0.851	0.872	0.919	0.730
2	100	*All*	0.860	0.915	0.939	0.816
		Car	0.870	0.941	0.954	0.884
		Motorcycle	0.849	0.890	0.923	0.748
3	150	*All*	0.877	0.900	0.941	0.817
		Car	0.857	0.929	0.955	0.890
		Motorcycle	0.897	0.871	0.926	0.744
4	200	*All*	0.886	0.899	0.936	0.820
		Car	0.886	0.941	0.949	0.887
		Motorcycle	0.866	0.856	0.922	0.752
5	250	*All*	0.882	0.926	0.938	0.824
		Car	0.894	0.947	0.954	0.892
		Motorcycle	0.870	0.904	0.923	0.755
6	300	*All*	0.901	0.888	0.941	0.824
		Car	0.895	0.932	0.954	0.892
		Motorcycle	0.907	0.844	0.929	0.757
7	350	*All*	0.894	0.902	0.946	0.830
		Car	0.891	0.932	0.959	0.894
		Motorcycle	0.897	0.873	0.934	0.767
8	400	*All*	0.892	0.899	0.939	0.827
		Car	0.893	0.932	0.953	0.894
		Motorcycle	0.892	0.865	0.926	0.760
9	450	*All*	0.867	0.927	0.949	0.829
		Car	0.872	0.950	0.959	0.890
		Motorcycle	0.863	0.904	0.939	0.768
10	500	*All*	0.918	0.878	0.943	0.828
		Car	0.928	0.928	0.951	0.892
		Motorcycle	0.907	0.828	0.934	0.764

Selain melihat waktu pelatihan model pada setiap nilai parameter *Epochs* serta akurasi yang dihasilkan, sangat penting untuk mengevaluasi hasil dari pelatihan model tersebut. Untuk itu, akan disajikan juga grafik berupa *Confusion Matrix* yang dihasilkan dari proses pelatihan model dengan variasi nilai parameter *Epochs* yang berbeda-beda. Grafik ini memberikan gambaran yang lebih terperinci tentang kinerja model dalam melakukan deteksi objek. Grafik *Confusion Matrix* menggambarkan hasil prediksi model terhadap data sebenarnya dalam bentuk matriks.

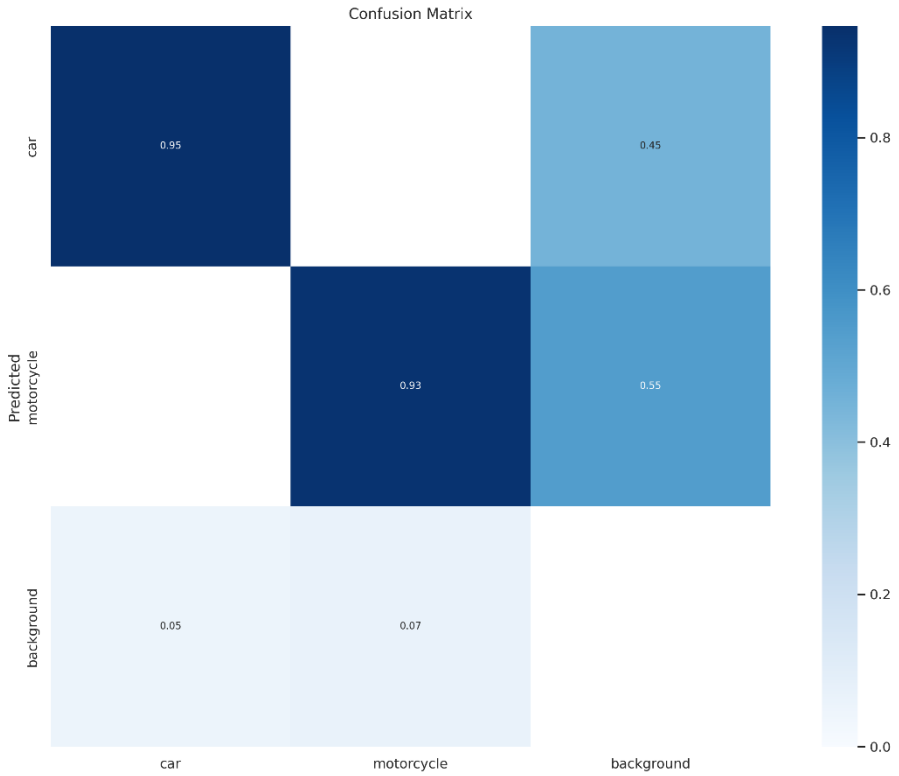
Dengan melihat grafik *Confusion Matrix* pada setiap nilai parameter *Epochs* yang berbeda, peneliti dapat menganalisis pola hasil prediksi model secara lebih rinci. Grafik tersebut dapat memberikan wawasan tentang sejauh mana model mampu mengenali dan membedakan antara kelas objek yang berbeda. Informasi ini sangat penting dalam mengidentifikasi potensi kelemahan atau kekuatan model. pada gambar-gambar berikut disajikan grafik *Confusion Matrix* dari hasil pelatihan model dengan nilai parameter *Epochs* yang berbeda-beda



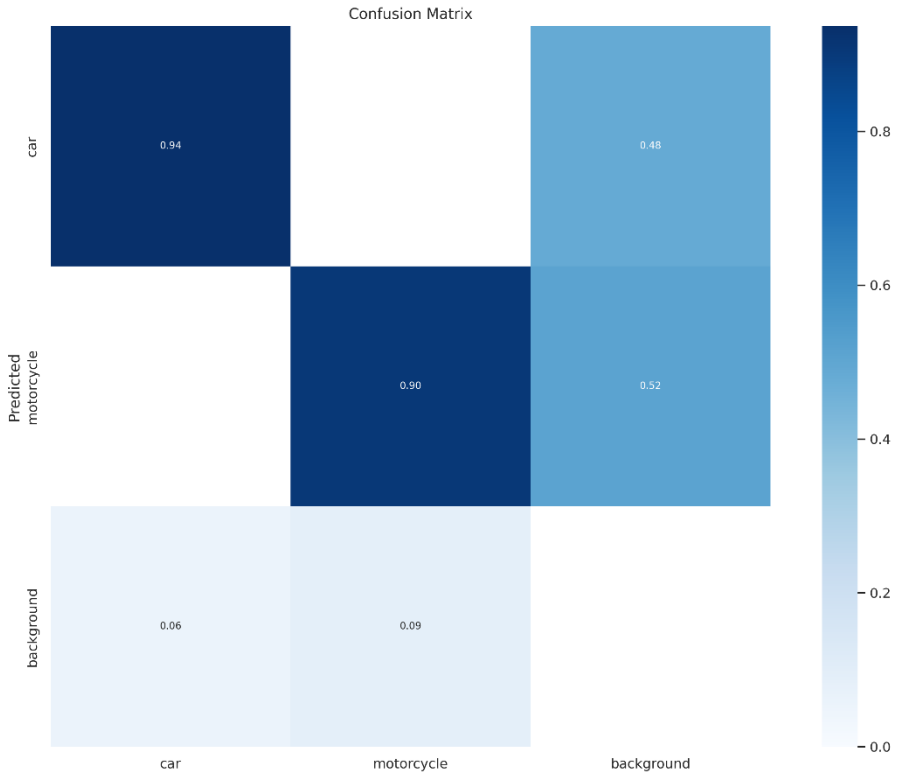
Gambar 4. 7 *Confusion Matrix* Pelatihan Model dengan *Epochs* 50



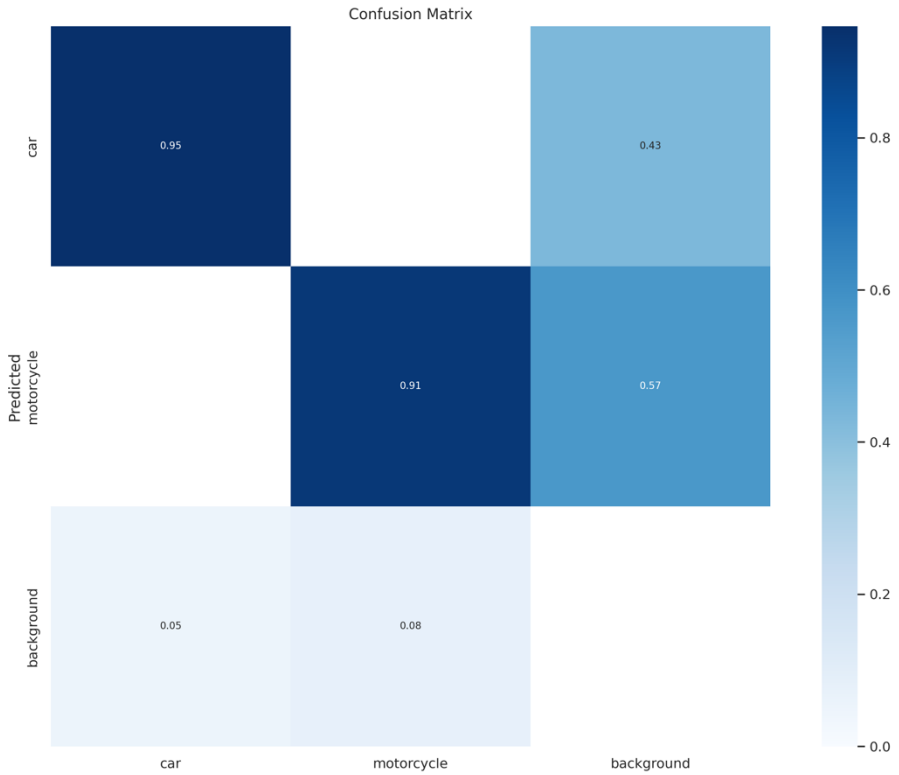
Gambar 4. 8 *Confusion Matrix* Pelatihan Model dengan *Epochs* 100



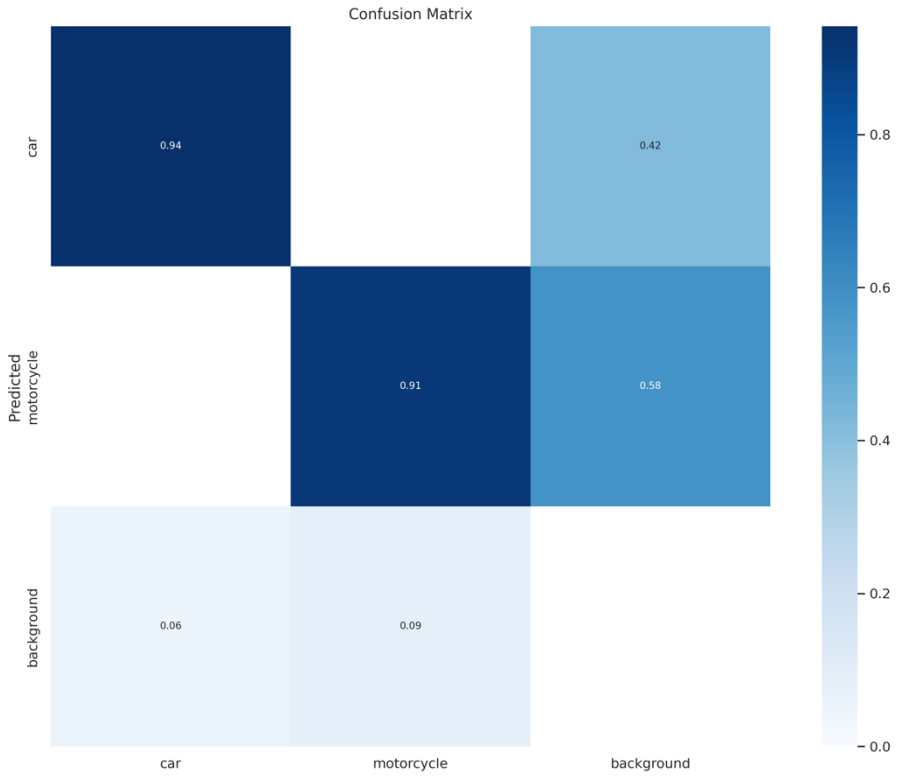
Gambar 4. 9 *Confusion Matrix* Pelatihan Model dengan *Epochs* 150



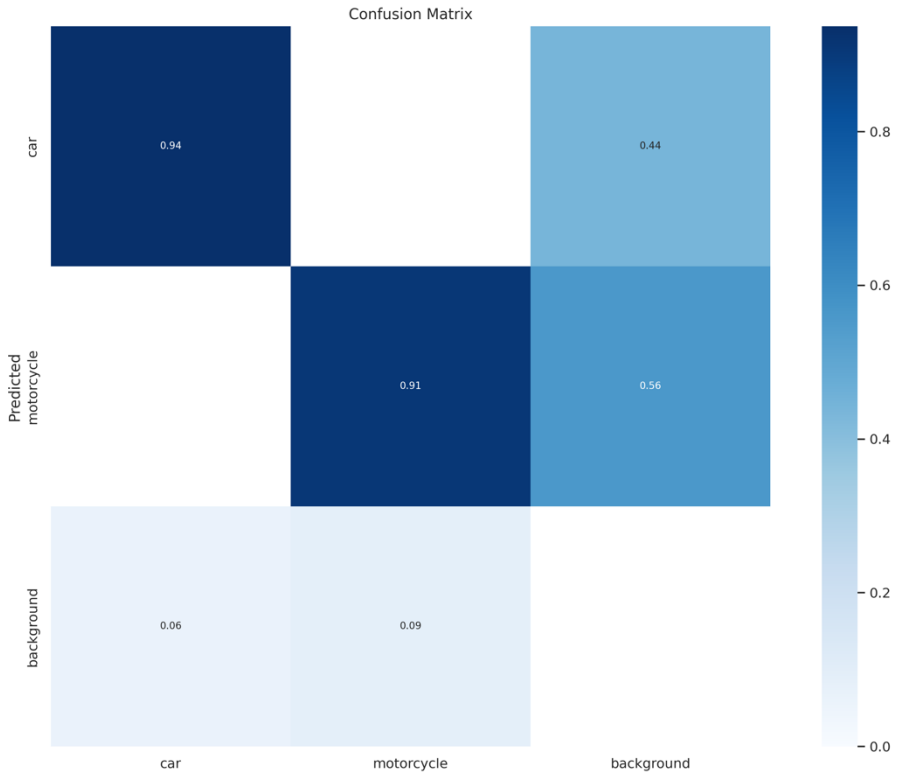
Gambar 4. 10 *Confusion Matrix* Pelatihan Model dengan *Epochs* 200



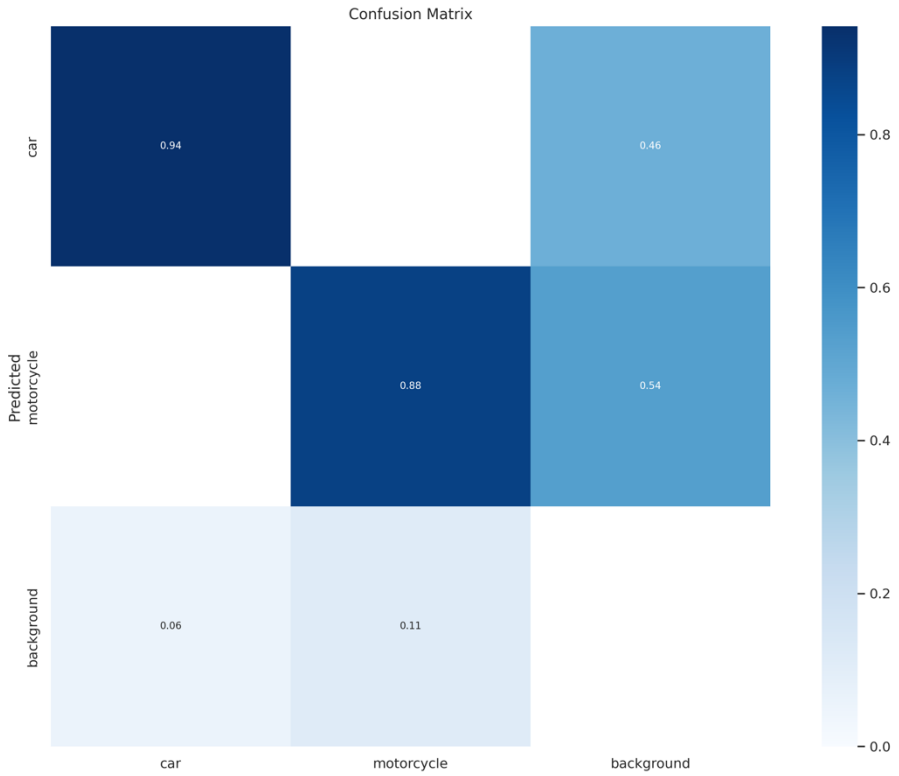
Gambar 4. 11 *Confusion Matrix* Pelatihan Model dengan *Epochs* 250



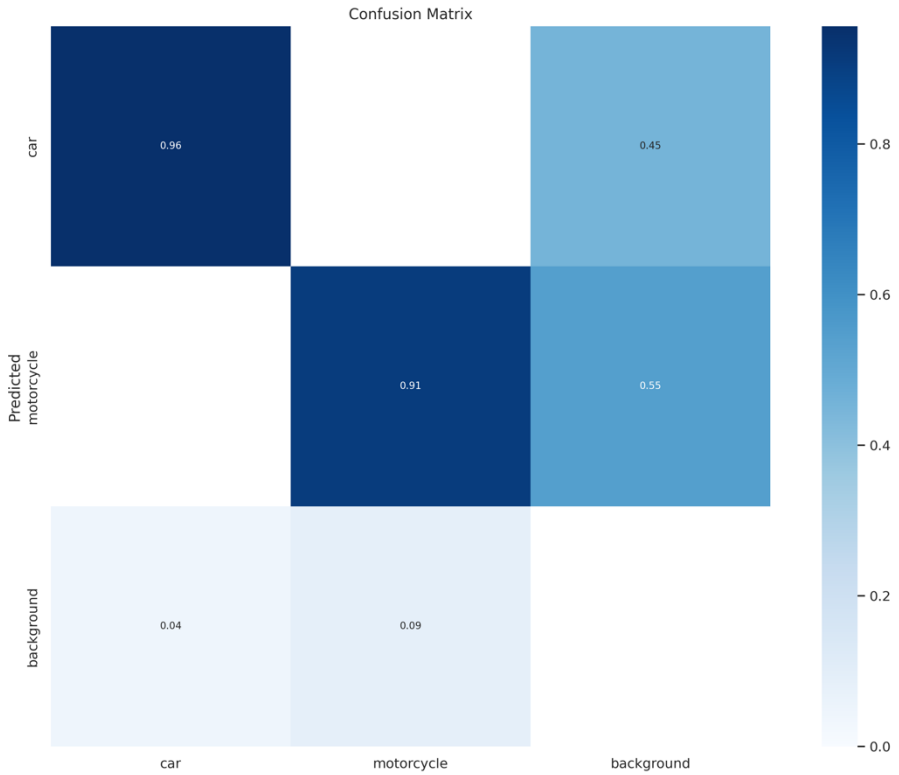
Gambar 4. 12 *Confusion Matrix* Pelatihan Model dengan *Epochs* 300



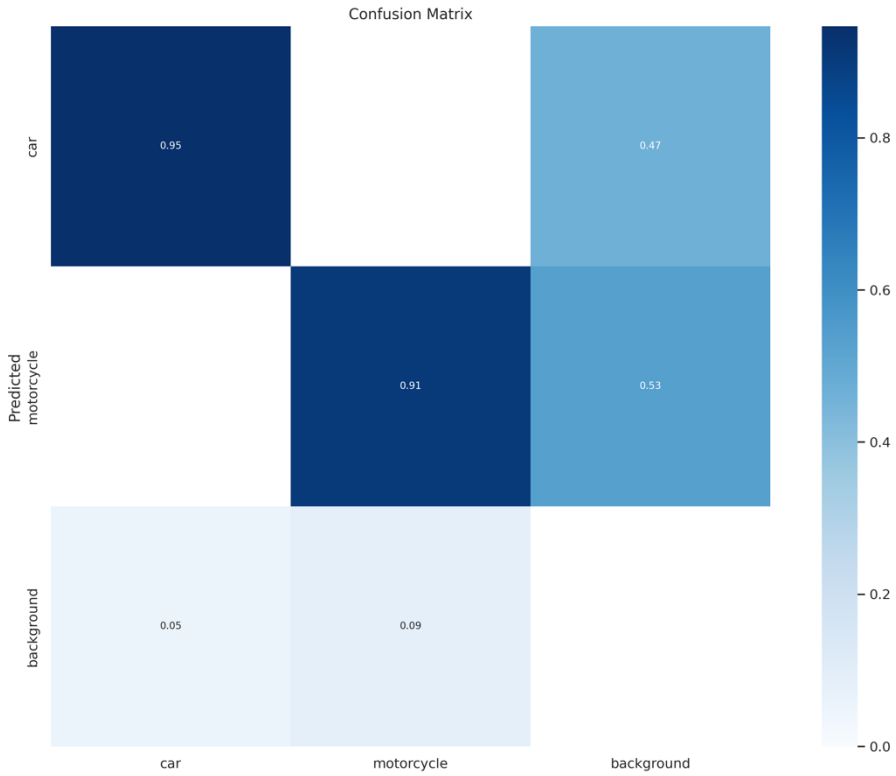
Gambar 4. 13 *Confusion Matrix* Pelatihan Model dengan *Epochs* 350



Gambar 4. 14 *Confusion Matrix* Pelatihan Model dengan *Epochs* 400



Gambar 4. 15 *Confusion Matrix* Pelatihan Model dengan *Epochs* 450



Gambar 4. 16 *Confusion Matrix* Pelatihan Model dengan *Epochs* 450

Setelah melalui proses pelatihan yang intensif, langkah berikutnya yang harus dilakukan adalah mengukur performa model dengan melakukan validasi terhadap *Dataset* validasi yang telah disiapkan sebelumnya. Tujuan dari tahap ini adalah untuk menguji sejauh mana model yang telah dilatih mampu melakukan generalisasi terhadap data yang belum pernah dilihat sebelumnya. Validasi ini sangat penting untuk memastikan bahwa model tidak hanya menghafal data pelatihan, tetapi juga memiliki kemampuan untuk mengenali pola-pola yang umum dalam data baru. Hasil dari validasi model terhadap *Dataset* validasi disajikan dalam Tabel 4.5 di bawah ini.

Tabel 4. 5 Hasil Validasi Model

No	Epochs	Class	Box P	Box R	mAP50	mAP50-95
1	50	*All*	0.866	0.905	0.940	0.803
		Car	0.878	0.938	0.960	0.878
		Motorcycle	0.855	0.872	0.920	0.728
2	100	*All*	0.860	0.916	0.939	0.815
		Car	0.870	0.941	0.955	0.884
		Motorcycle	0.849	0.890	0.923	0.745
3	150	*All*	0.877	0.900	0.941	0.815

		Car	0.857	0.929	0.955	0.887
		Motorcycle	0.897	0.871	0.927	0.743
4	200	*All*	0.886	0.899	0.936	0.820
		Car	0.866	0.941	0.949	0.886
		Motorcycle	0.866	0.856	0.922	0.754
5	250	*All*	0.880	0.926	0.938	0.822
		Car	0.894	0.947	0.954	0.889
		Motorcycle	0.865	0.904	0.923	0.755
6	300	*All*	0.901	0.888	0.941	0.823
		Car	0.895	0.932	0.954	0.891
		Motorcycle	0.907	0.845	0.929	0.755
7	350	*All*	0.894	0.902	0.946	0.833
		Car	0.891	0.932	0.959	0.898
		Motorcycle	0.897	0.872	0.934	0.767
8	400	*All*	0.892	0.899	0.940	0.827
		Car	0.893	0.932	0.953	0.895
		Motorcycle	0.892	0.865	0.926	0.759
9	450	*All*	0.867	0.927	0.948	0.828
		Car	0.872	0.950	0.959	0.889
		Motorcycle	0.863	0.904	0.937	0.767
10	500	*All*	0.918	0.878	0.943	0.829
		Car	0.928	0.928	0.951	0.891
		Motorcycle	0.907	0.828	0.935	0.767

Informasi yang disajikan pada Tabel 4.5 diatas memiliki peranan yang sangat penting dalam menggambarkan kinerja model deteksi objek yang telah dilatih. Dalam konteks evaluasi model, terdapat berbagai metrik evaluasi yang digunakan untuk memberikan wawasan yang komprehensif tentang kemampuan model dalam mengidentifikasi dan mengklasifikasikan objek pada gambar. Metrik-metrik ini mencakup Box P, Box R, mAP50, dan mAP50-95.

Box P dan Box R merujuk pada *precision* dan *Recall* yang dihitung untuk deteksi objek dengan koordinat *bounding box*. *Precision* (Box P) mengukur seberapa akurat model dalam mengidentifikasi objek dalam *bounding box*, sedangkan *Recall* (Box R) mengukur seberapa baik model dalam menemukan semua objek yang sebenarnya ada. Dalam kedua metrik ini, nilai yang lebih tinggi menunjukkan kualitas yang lebih baik. Sementara itu, mAP50 dan mAP50-95 merujuk pada *Mean Average Precision* pada IoU (*Intersection over Union*) *threshold* 0.5 dan 0.5 hingga 0.95. mAP adalah metrik yang mengukur sejauh mana model dapat mengklasifikasikan objek dengan tepat dalam berbagai tingkat

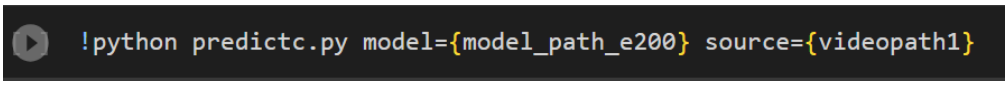
keakuratan. Penggunaan beberapa IoU *threshold* memberikan gambaran lebih komprehensif tentang performa model dalam situasi yang berbeda-beda.

Hasil validasi model terhadap *Dataset* validasi memberikan pandangan yang lebih jelas tentang kemampuan model dalam mengenali objek pada situasi dunia nyata. Hasil ini akan memberikan informasi yang penting untuk mengambil keputusan mengenai apakah model sudah cukup baik untuk diterapkan dalam aplikasi nyata atau perlu diperbaiki lagi.

4.1.3 Pengujian Model

Setelah menyelesaikan tahap validasi model, langkah berikutnya dalam penelitian ini adalah mengujikan model yang telah dilatih pada video CCTV berdurasi 30 detik. Proses pengujian ini bertujuan untuk mengukur sejauh mana model yang telah dikembangkan dapat berperforma dalam kondisi nyata. Dalam penelitian ini, pengujian model dilakukan dengan memanfaatkan *platform Google Colaboratory*. Pengujian model dilakukan dengan menggunakan video CCTV yang telah direkam sebelumnya dan belum pernah dihadapkan pada model sebelumnya.

Pengujian model dengan video CCTV adalah tahap kritis dalam memastikan kualitas dan kemampuan sebenarnya dari model. Hasil dari pengujian ini sangat penting dalam mengukur keakuratan dan keandalan model dalam pendeteksian dan penghitungan kendaraan secara otomatis di lingkungan nyata. Hasil dari pengujian model dengan video CCTV akan memberikan gambaran nyata tentang sejauh mana model mampu beradaptasi dengan berbagai variabilitas situasi di lapangan. Berikut adalah *script* python yang digunakan untuk melakukan pengujian model.



Gambar 4. 17 *Script* Python Pengujian Model

Dalam pengujian model pada penelitian ini, pendekatan yang diambil adalah dengan menggunakan rekaman video daripada menguji secara *Real-time*. Keputusan ini didasarkan pada fokus utama penelitian, yakni menganalisis akurasi pendeteksian dan penghitungan kendaraan. Oleh karena itu, pengujian dilakukan dengan merekam video yang mencakup situasi dunia nyata di jalan raya. Dalam konteks ini, setiap video rekaman dijadikan sebagai *input* bagi model yang telah

dilatih sebelumnya. Tujuan utama adalah untuk menilai bagaimana model dapat melakukan pendeteksian dan penghitungan kendaraan dalam situasi yang sudah terekam sebelumnya. Hasil pengujian ini kemudian menjadi bagian penting dalam evaluasi performa model dalam kondisi nyata.

Sebagai bagian dari proses evaluasi, pengujian dilakukan pada empat model yang berbeda, masing-masing telah dilatih dengan berbagai variasi nilai *Epochs*. Variasi *Epochs* ini memungkinkan peneliti untuk menganalisis bagaimana performa model dapat berubah seiring dengan perubahan jumlah iterasi dalam proses pelatihan. Gambaran proses pengujian model dapat dilihat pada Gambar 4.18 berikut.

```
Model summary: 268 layers, 43608150 parameters, 0 gradients, 164.8 GFLOPs
video 1/1 (1/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 3 cars, 2 motorcycles, 84.6ms
video 1/1 (2/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 3 cars, 2 motorcycles, 38.2ms
video 1/1 (3/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 38.2ms
video 1/1 (4/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 38.3ms
video 1/1 (5/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 33.3ms
video 1/1 (6/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 33.1ms
video 1/1 (7/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 33.1ms
video 1/1 (8/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 33.2ms
video 1/1 (9/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 25.6ms
video 1/1 (10/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 25.6ms
video 1/1 (11/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 25.6ms
video 1/1 (12/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 25.6ms
video 1/1 (13/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 29.3ms
video 1/1 (14/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 26.8ms
video 1/1 (15/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 23.8ms
video 1/1 (16/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 25.5ms
video 1/1 (17/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 23.6ms
video 1/1 (18/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 23.9ms
video 1/1 (19/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 23.9ms
video 1/1 (20/1836) /content/drive/MyDrive/skripsi/video_test/vt1.mp4: 448x640 2 cars, 2 motorcycles, 23.6ms
```

Gambar 4. 18 Proses Pengujian Model

Gambar 4.18 menggambarkan tahap yang signifikan dalam proses pengujian model pada penelitian ini. Video yang memiliki durasi 30 detik dibagi menjadi 1836 *frame* individu, yang mewakili gambar diam pada setiap titik waktu dalam video, yang berarti setiap detik dari video di pecah menjadi 60 *frame*. Dalam proses ini, setiap frame melewati algoritma deteksi objek YOLOv8 yang telah dilatih sebelumnya. Tujuan utama adalah untuk mengidentifikasi dan memetakan objek kendaraan yang ada dalam setiap *frame*.

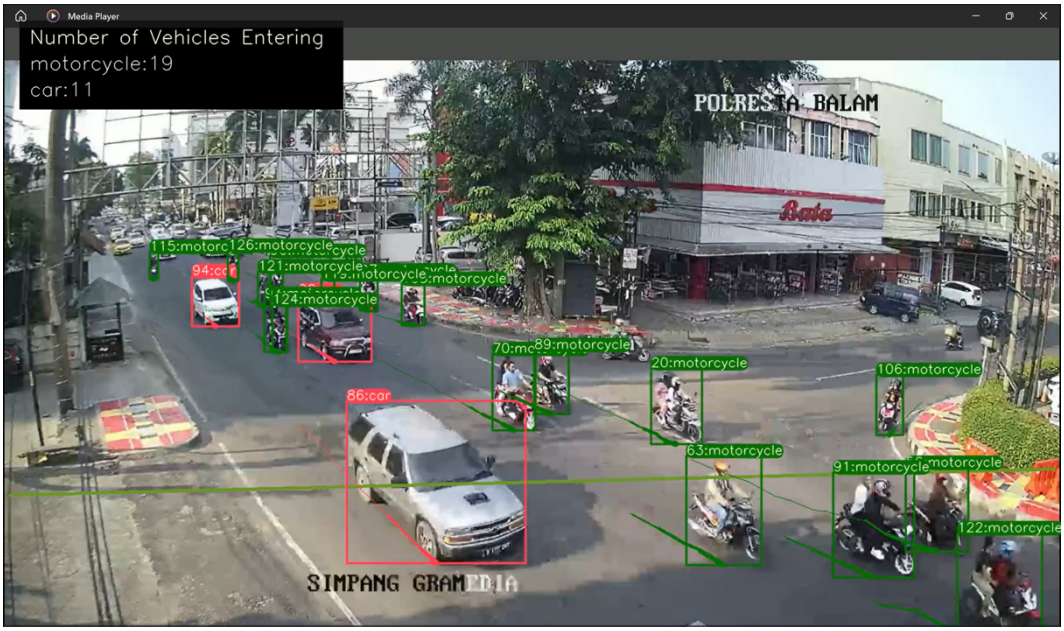
Tahap pendeteksian objek dalam penelitian ini memiliki peran kunci dalam analisis akurasi penghitungan kendaraan. Ketika model YOLOv8 mendeteksi kendaraan dalam gambar atau frame video, ia tidak hanya menghasilkan bounding box yang mengelilingi setiap kendaraan yang berhasil terdeteksi, tetapi juga

memberikan setiap kendaraan sebuah ID unik yang bersifat sementara. Ini merupakan elemen penting dalam memastikan akurasi penghitungan kendaraan.

Setiap kendaraan diberikan ID unik yang hanya berlaku dalam konteks penghitungan saat itu. Dengan kata lain, saat kendaraan pertama kali muncul dalam frame video, ia akan diberikan ID unik, dan ID ini akan terus digunakan untuk mengidentifikasi kendaraan tersebut selama muncul dalam frame-frame video berikutnya. Sehingga, ketika kendaraan melewati garis pelacakan, sistem dapat dengan pasti mengidentifikasi dan menghitung kendaraan tersebut tanpa adanya penghitungan ganda.

Garis pelacakan ini memainkan peran krusial dalam mengukur jumlah kendaraan yang melewati suatu titik tertentu di jalan raya selama periode waktu tertentu. Garis ini memberikan referensi yang jelas dan konstan untuk perhitungan jumlah kendaraan yang berhasil melintasinya. Ketika sebuah kendaraan melintasi garis ini, sistem merekam peristiwa tersebut dan mencatatnya sesuai dengan ID unik kendaraan yang bersangkutan.

Hasil dari seluruh proses ini adalah video *output* hasil pengujian model yang sangat informatif. Bounding box yang mengelilingi setiap kendaraan memberikan informasi visual yang jelas tentang lokasi objek dalam setiap frame video. Selain itu, garis pelacakan yang secara dinamis bergerak bersama dengan kendaraan memberikan pandangan langsung tentang bagaimana kendaraan bergerak dalam video dan bagaimana mereka melintasi garis pelacakan tersebut. Dengan memadukan elemen-elemen visual ini dan ID unik yang diberikan kepada setiap kendaraan, analisis akurasi penghitungan kendaraan dapat dilakukan dengan sangat efisien dan akurat. Pentingnya tahap ini tidak hanya terletak pada deteksi objek itu sendiri tetapi juga pada kemampuan sistem untuk memastikan setiap kendaraan dihitung dengan benar tanpa ada penghitungan ganda, yang merupakan elemen kunci dalam analisis akurasi penghitungan kendaraan pada proyek ini. Berikut adalah gambar hasil dari pengujian model yang di tangkap dari *output* video pengujian model.



Gambar 4. 19 *Output* Video Pengujian Model

Namun, informasi visual saja tidaklah cukup. Oleh karena itu, dalam *output* video ini, terdapat juga keterangan yang menunjukkan jumlah kendaraan yang berhasil terdeteksi dan dihitung saat melintas di depan kamera. Keterangan ini memberikan wawasan langsung tentang efektivitas model dalam melakukan pendeteksian dan penghitungan objek secara akurat dalam situasi dunia nyata. Dengan begitu, *output* video ini menjadi suatu hasil yang informatif dan memberikan pemahaman yang jelas tentang performa model dalam tugas penghitungan kendaraan.

4.2 Analisis

Pada sub bab ini, akan dilakukan analisis yang mendalam terhadap hasil *output* dari pengujian model yang telah dilakukan sebelumnya. Proses analisis ini merupakan tahapan krusial dalam penelitian, yang bertujuan untuk memperoleh pemahaman yang komprehensif mengenai performa model dalam pendeteksian dan penghitungan kendaraan. Dalam analisis ini, fokus utama akan diberikan pada dua aspek utama, yakni akurasi pendeteksian kendaraan dan tingkat keakuratan dalam penghitungan jumlah kendaraan.

Untuk menganalisis akurasi pendeteksian kendaraan, digunakan metrik-metrik penting seperti Akurasi, *Precision*, *Recall*, dan *F1-Score*. Keempat metrik ini memberikan informasi rinci tentang kinerja model dalam mengenali dan

membedakan objek yang diidentifikasi sebagai kendaraan. *Accuracy* mengukur sejauh mana model klasifikasi mampu memprediksi dengan benar semua kelas yang ada dalam *Dataset*. *Precision* mengukur seberapa akurat model dalam mengidentifikasi kendaraan dari objek yang sebenarnya bukan kendaraan. *Recall*, di sisi lain, menunjukkan seberapa baik model dalam mengenali seluruh objek kendaraan yang seharusnya diidentifikasi. *F1-Score* merupakan harmonisasi antara *Precision* dan *Recall*, yang memberikan pandangan holistik tentang performa model dalam klasifikasi objek kendaraan.

Selanjutnya, analisis dilakukan untuk mengukur seberapa akurat penghitungan kendaraan yang dilakukan oleh model secara otomatis. Untuk mengukur ini, perbandingan akan dilakukan antara hasil penghitungan kendaraan secara otomatis dengan penghitungan kendaraan yang dilakukan secara manual lalu dihitung *Error Relative* nya. Proses ini akan memungkinkan peneliti untuk mendapatkan gambaran sejauh mana model mampu melakukan penghitungan yang akurat dan konsisten dengan penghitungan manual yang dilakukan oleh manusia. Analisis tersebut akan memberikan pemahaman yang lebih lengkap mengenai performa model dalam penghitungan kendaraan. Dengan analisis yang mendalam terhadap kedua aspek tersebut, diharapkan penelitian ini dapat memberikan pandangan yang jelas dan teruji mengenai kemampuan dan batasan model dalam melakukan pendeteksian dan penghitungan kendaraan.

4.2.1 Analisis Akurasi Pendeteksian Kendaraan

Hasil video yang diperoleh dari pengujian model akan dipelajari dengan cermat untuk melakukan evaluasi mendalam terhadap akurasi pendeteksian yang telah dicapai oleh masing-masing model. Pada bagian analisis ini akan disajikan tabel matriks kebingungan (*Confusion Matrix*) untuk setiap model yang telah diujicobakan. Tabel *Confusion Matrix* untuk setiap model yang telah diujicobakan dapat dilihat pada tabel-tabel berikut.

Tabel 4. 6 *Confusion Matrix* Pengujian Model *Epochs* 50

		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	71	8
	Car	5	78

Tabel 4. 7 *Confusion Matrix* Pengujian Model *Epochs* 100

		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	70	8
	Car	3	80

Tabel 4. 8 *Confusion Matrix* Pengujian Model *Epochs* 150

		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	72	6
	Car	5	78

Tabel 4. 9 *Confusion Matrix* Pengujian Model *Epochs* 200

		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	68	10
	Car	7	76

Tabel 4. 10 *Confusion Matrix* Pengujian Model *Epochs* 250

		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	72	6
	Car	4	79

Tabel 4. 11 *Confusion Matrix* Pengujian Model *Epochs* 300

		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	69	10
	Car	9	76

Tabel 4. 12 *Confusion Matrix* Pengujian Model *Epochs* 350

		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	72	8
	Car	6	77

Tabel 4. 13 *Confusion Matrix* Pengujian Model *Epochs* 400

		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	70	10
	Car	7	77

Tabel 4. 14 *Confusion Matrix* Pengujian Model *Epochs* 450

		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	73	5
	Car	8	78

Tabel 4. 15 *Confusion Matrix* Pengujian Model *Epochs* 500

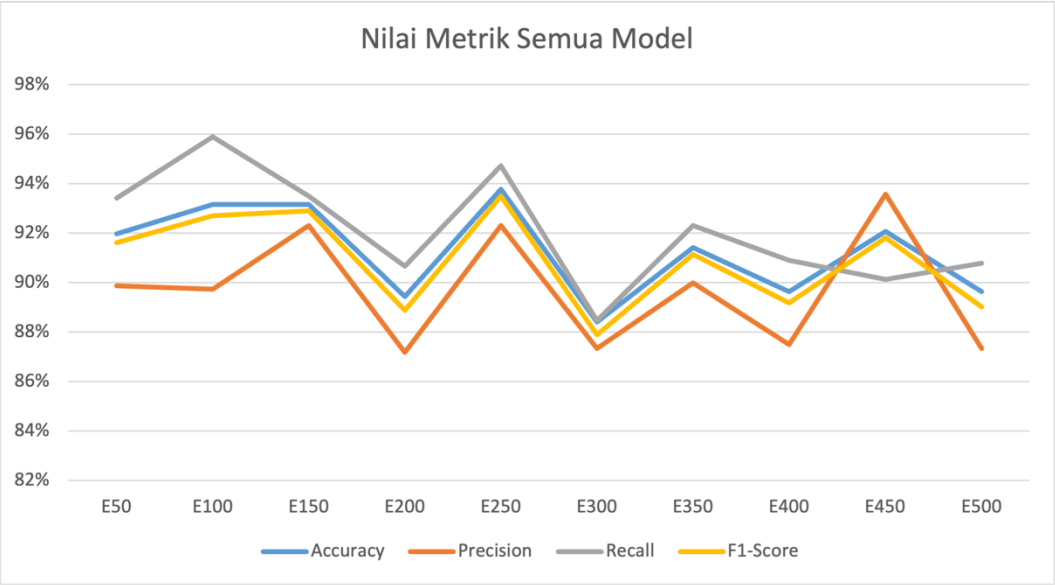
		Aktual	
		Motorcycle	*Car*
Prediksi	*Motorcycle*	69	10
	Car	7	78

Matriks kebingungan akan menunjukkan secara jelas jumlah dari *true positive* (TP), *true negative* (TN), *false positive* (FP), dan *false negative* (FN) yang diperoleh oleh model dalam pendeteksian objek kendaraan. Dengan menggunakan informasi yang terdapat pada matriks kebingungan, selanjutnya akan dihitung nilai dari metrik *Accuracy*, *Precision*, *Recall*, dan *F1-Score* nya yang mana metrik ini merupakan metrik evaluasi utama dalam mengukur akurasi pendeteksian. Pada Tabel berikut disajikan nilai dari metrik *Accuracy*, *Precision*, *Recall*, dan *F1-Score* dari masing-masing model yang telah dilatih.

Tabel 4. 16 Nilai Metrik *Accuracy*, *Precision*, *Recall* dan *F1-Score* Model

Model	***Accuracy***	***Precision***	***Recall***	***F1-Score***
***Epochs* 50**	0,920	0,899	0,934	0,916
***Epochs* 100**	0,932	0,897	0,959	0,927

***Epochs* 150**	0,932	0,923	0,935	0,929
***Epochs* 200**	0,894	0,872	0,907	0,889
***Epochs* 250**	0,938	0,923	0,947	0,935
***Epochs* 300**	0,884	0,873	0,885	0,879
***Epochs* 350**	0,914	0,900	0,923	0,911
***Epochs* 400**	0,896	0,875	0,909	0,892
***Epochs* 450**	0,921	0,936	0,901	0,918
***Epochs* 500**	0,896	0,873	0,908	0,890



Gambar 4. 20 *Chart* Perbandingan Nilai Metrik Semua Model

Hasil evaluasi kinerja model pada berbagai nilai *Epochs* menunjukkan variasi dalam metrik yang digunakan untuk mengukur akurasi dan keandalan model deteksi. Pada model dengan 50 *Epochs*, terlihat akurasi sekitar 0.919, yang menunjukkan kemampuan model dalam mengidentifikasi dengan benar seluruh

kelas yang ada dalam *Dataset*. Namun, nilai *Precision* pada model ini sekitar 0.898, yang menunjukkan bahwa sejumlah kecil data mungkin salah diklasifikasikan sebagai positif. *Recall*, di sisi lain, memiliki nilai sekitar 0.934, yang menggambarkan kemampuan model dalam mengidentifikasi sebagian besar data positif yang ada dalam *Dataset*. *F1-Score* pada model ini sekitar 0.916, yang mencerminkan keseimbangan antara *Precision* dan *Recall*.

Pada model dengan 100 *Epochs*, terlihat peningkatan yang signifikan dalam *Recall*, mencapai sekitar 0.958. Ini menunjukkan bahwa model lebih baik dalam mengidentifikasi data positif dalam *Dataset*. Namun, *Precision* sedikit menurun menjadi sekitar 0.897, yang berarti ada sedikit peningkatan dalam jumlah data negatif yang salah diklasifikasikan sebagai positif. Akurasi pada model ini adalah sekitar 0.931, dan *F1-Score* sekitar 0.927, menunjukkan performa keseluruhan model yang baik.

Model dengan 150 *Epochs* menunjukkan akurasi yang hampir sama dengan model 100 *Epochs*, yaitu sekitar 0.931, tetapi dengan nilai *Precision* yang sedikit lebih tinggi, yaitu sekitar 0.923. Hal ini menunjukkan bahwa model ini lebih akurat dalam mengklasifikasikan data positif. *Recall* sekitar 0.935 dan *F1-Score* sekitar 0.929, mencerminkan kinerja yang baik secara keseluruhan.

Pada model dengan 200 *Epochs*, terlihat penurunan akurasi menjadi sekitar 0.894, yang mengindikasikan bahwa model mungkin mulai *overfitting* pada data pelatihan. *Precision* sekitar 0.871, *Recall* sekitar 0.906, dan *F1-Score* sekitar 0.888, yang semuanya menunjukkan penurunan performa model dalam mengidentifikasi dan mengklasifikasikan data positif.

Model dengan 250 *Epochs* menunjukkan peningkatan yang signifikan dalam *Recall*, mencapai sekitar 0,947. Ini menunjukkan bahwa model ini memiliki kemampuan yang sangat baik dalam mengidentifikasi data positif dalam *dataset*. Presisi yang sekitar 0,923 juga menunjukkan tingkat akurasi yang tinggi dalam mengklasifikasikan data positif. Akurasi model ini adalah sekitar 0,938, dan F1-*Score* sekitar 0,935, mencerminkan performa yang sangat baik secara keseluruhan.

Model dengan 300 *Epochs* menunjukkan penurunan akurasi menjadi sekitar 0,884, yang mungkin mengindikasikan bahwa model kembali mengalami *overfitting* pada data pelatihan. Presisi yang sekitar 0,873 dan *Recall* yang sekitar

0,885 menunjukkan penurunan performa model dalam mengidentifikasi dan mengklasifikasikan data positif. Meskipun demikian, F1-*Score* sekitar 0,879 masih mencerminkan performa yang cukup baik secara keseluruhan.

Model dengan 350 *Epochs* menunjukkan peningkatan akurasi menjadi sekitar 0,914, yang menunjukkan kinerja yang lebih baik daripada model dengan 300 *Epochs*. Presisi yang sekitar 0,900 dan *Recall* yang sekitar 0,923 juga menunjukkan kinerja yang baik dalam mengidentifikasi dan mengklasifikasikan data positif. F1-*Score* sekitar 0,911 mencerminkan kinerja keseluruhan yang baik.

Model dengan 400 *Epochs* menunjukkan peningkatan akurasi menjadi sekitar 0,896 dibandingkan dengan model 350 *Epochs*. Presisi sekitar 0,875 dan *Recall* sekitar 0,909 menunjukkan kinerja yang baik dalam mengidentifikasi dan mengklasifikasikan data positif. Meskipun akurasi lebih rendah daripada model sebelumnya, F1-*Score* sekitar 0,892 masih mencerminkan performa yang baik secara keseluruhan.

Model dengan 450 *Epochs* menunjukkan peningkatan akurasi menjadi sekitar 0,921, dengan nilai Presisi sekitar 0,936 yang menunjukkan tingkat akurasi yang tinggi dalam mengklasifikasikan data positif. *Recall* sekitar 0,901 menunjukkan kemampuan model dalam mengidentifikasi data positif. F1-*Score* sekitar 0,918 mencerminkan performa yang baik secara keseluruhan.

Model dengan 500 *Epochs* menunjukkan akurasi sekitar 0,896, dengan Presisi sekitar 0,873 dan Recall sekitar 0,908. Meskipun akurasi lebih rendah daripada model dengan 450 Epochs, F1-*Score* sekitar 0,890 masih mencerminkan performa yang cukup baik secara keseluruhan.

Dalam analisis ini, dapat diamati bahwa peningkatan nilai *Epochs* tidak selalu menghasilkan peningkatan kinerja model yang signifikan. Model dengan 100, 150 dan 250 *Epochs* menunjukkan performa yang baik, sementara model dengan 200, 300, 400 dan 500 *Epochs* mengalami penurunan performa. Oleh karena itu, pemilihan nilai *Epochs* yang optimal sangat penting dalam pelatihan model deteksi objek ini.

Berdasarkan hasil analisis, dapat disimpulkan bahwa performa model deteksi objek pada *Dataset* ini paling baik dicapai oleh model yang memiliki jumlah *Epochs* sebesar 100, 150, dan 250. Model dengan 100 *Epochs* memiliki akurasi

sekitar 0.931, sementara model dengan 150 *Epochs* memiliki akurasi yang sama, yaitu sekitar 0.931, dan model dengan 250 *Epochs* memiliki akurasi sekitar 0.938. Oleh karena itu, dari segi akurasi, ketiga model ini menunjukkan kinerja yang setara. Namun, jika kita mempertimbangkan metrik lain seperti *Precision, Recall*, dan F1-*Score*, kita dapat melihat perbedaan yang lebih jelas. Model dengan 100 *Epochs* memiliki *Recall* yang tinggi, yaitu sekitar 0.958, mengindikasikan kemampuan model ini dalam mengidentifikasi semua kendaraan yang ada. Model dengan 150 *Epochs* memiliki *Precision* yang lebih tinggi, yaitu sekitar 0.923, dan F1-*Score* sekitar 0.929, menunjukkan bahwa model ini lebih baik dalam mengidentifikasi area yang bukan kendaraan sebagai kendaraan. Model dengan 250 *Epochs* juga menunjukkan performa yang baik dengan *Recall* sekitar 0.947 dan F1-*Score* sekitar 0.935, menggambarkan kemampuan model dalam mengidentifikasi kendaraan. Dengan demikian, jika kinerja model dalam mengidentifikasi kendaraan secara umum adalah prioritas, maka model dengan 100 atau 250 *Epochs* mungkin merupakan pilihan yang baik. Sementara itu, jika mengurangi kesalahan dalam mengidentifikasi area bukan kendaraan sebagai kendaraan adalah prioritas, model dengan 150 *Epochs* tetap menjadi pilihan yang baik.

Kesimpulannya, pemilihan model terbaik harus disesuaikan dengan tujuan dan kebutuhan khusus dari proyek yang berhubungan. Harus dipertimbangkan tingkat akurasi yang diperlukan, keterbatasan sumber daya, tenggat waktu proyek, dan prioritas-prioritas khusus proyek tersebut. Dengan demikian, pemilihan akhir model harus disesuaikan dengan kebutuhan unik dan situasi proyek yang sedang dihadapi.

4.2.2 Analisis Akurasi Penghitungan Kendaraan

Pada bagian analisis ini, untuk mengukur akurasi penghitungan kendaraan dilakukan dengan membandingkan hasil penghitungan otomatis yang dihasilkan oleh model dengan data sesungguhnya dilapangan yang didapat dari penghitungan manual yang dilakukan oleh peneliti, selanjutnya dari penghitungan tersebut akan dicari nilai *Error Relative* nya. Dengan demikian, kita dapat mengukur sejauh mana model dapat menggantikan peran manusia dalam menghitung kendaraan di jalan raya. Hasil penghitungan ini akan memberikan wawasan yang berharga tentang

efektivitas model dalam memenuhi tujuan utama penelitian ini. Pada tabel berikut ini akan disajikan hasil penghitungan kendaraan secara otomatis yang dilakukan oleh model dan hasil sesungguhnya yang dihitung oleh peneliti.

Tabel 4. 17 Hasil Penghitungan Secara Otomatis dan Manual

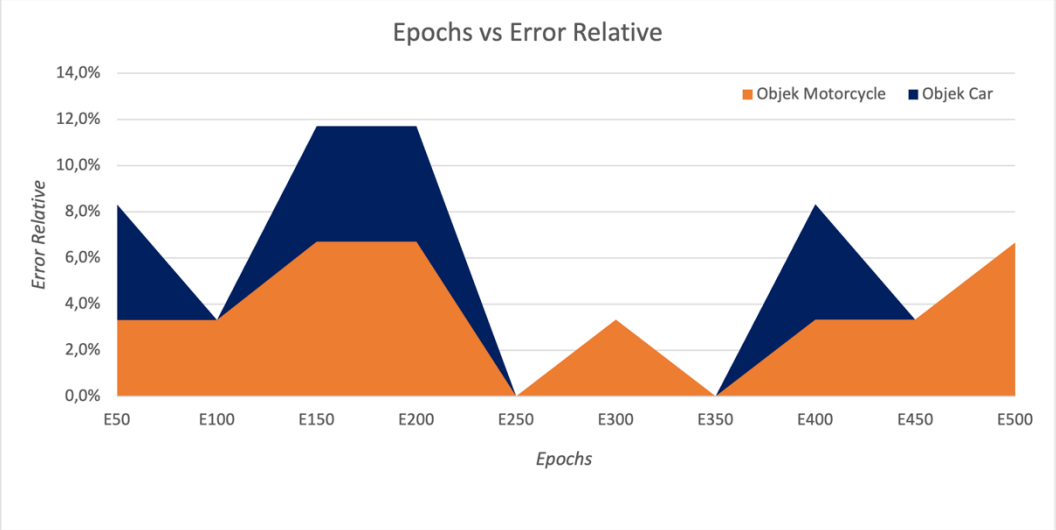
Penghitungan	**Objek**	
	Motorcycle	***Car***
Peneliti	30	20
Epochs 50	29	21
Epochs 100	29	20
Epochs 150	28	21
Epochs 200	28	21
Epochs 250	30	20
Epochs 300	29	20
Epochs 350	30	20
Epochs 400	29	21
Epochs 450	31	20
Epochs 500	28	20

Setelah memperoleh hasil penghitungan jumlah kendaraan, langkah berikutnya adalah melakukan perhitungan nilai *Error Relative*. *Error Relative* adalah ukuran statistik yang digunakan untuk mengukur sejauh mana perbedaan antara dua nilai numerik (seperti hasil penghitungan atau estimasi) dan nilai referensi atau nilai yang dianggap sebagai nilai yang benar atau akurat. Ini memberikan gambaran tentang sejauh mana hasil pengukuran atau perhitungan mendekati nilai yang seharusnya. Dengan menghitung *Error Relative*, kita dapat memahami seberapa akurat dan representatif penghitungan kendaraan otomatis dibandingkan dengan jumlah kendaraan yang sebenarnya ada di dalam situasi tersebut. Hasil perhitungan nilai *Error Relative* di tunjukkan pada Tabel 4.18 sebagai berikut.

Tabel 4. 18 Hasil Perhitungan Nilai *Error Relative* Model

Penghitungan	**Objek**		
	Motorcycle	***Car***	***Average All***
Epochs 50	3,3%	5,0%	4,2%
Epochs 100	3,3%	0,0%	1,7%
Epochs 150	6,7%	5,0%	5,9%
Epochs 200	6,7%	5,0%	5,9%
Epochs 250	0,0%	0,0%	0,0%
Epochs 300	3,3%	0,0%	1,7%
Epochs 350	0,0%	0,0%	0,0%
Epochs 400	3,3%	5,0%	4,2%
Epochs 450	3,3%	0,0%	1,7%
Epochs 500	6,7%	0,0%	3,3%

Dalam proses analisis hasil penelitian ini, fokus tertuju pada nilai *Error Relative* sebagai ukuran yang signifikan untuk mengukur tingkat akurasi model-model yang telah dilatih dengan jumlah *Epochs* yang berbeda dalam tugas pendeteksian dan penghitungan kendaraan. *Error Relative* memberikan pandangan mendalam tentang sejauh mana model mendekati hasil yang benar dalam penghitungan kendaraan, yang terbagi menjadi dua kategori utama, yaitu *motorcycle* dan *car*.



Gambar 4. 21 Chart Perbandingan *Error Relative* Masing-Masing Model

Model dengan 50 *Epochs* menunjukkan hasil yang cukup baik dalam mengukur akurasi deteksi objek. *Error Relative* untuk label *motorcycle* adalah sekitar 3.3%, sedangkan label *car* mencapai 5.0%. Secara rata-rata, model ini memiliki *Error Relative* sekitar 4.2%. Ini menggambarkan bahwa model ini memiliki akurasi yang layak dalam mengidentifikasi kendaraan, meskipun dengan sedikit ketidakpastian dalam menghitung kendaraan.

Selanjutnya, model dengan 100 *Epochs* menunjukkan peningkatan yang signifikan dalam akurasi deteksi kendaraan. *Error Relative* untuk label *motorcycle* tetap konsisten pada 3.3%, menunjukkan kestabilan dalam deteksinya. Yang menarik, model ini berhasil mencapai akurasi penuh dalam penghitungan label *car,* dengan *Error Relative* mencapai angka 0%. Hasil ini mengindikasikan bahwa model ini mampu mengidentifikasi label *car* tanpa melakukan kesalahan dalam prosesnya. Secara rata-rata, model ini memiliki *Error Relative* sekitar 1.7%.

Namun, terdapat peningkatan *Error Relative* pada model dengan 150 *Epochs*, terutama pada label *motorcycle* yang mencapai 6.7%. Meskipun label *car* tetap pada 5.0%, ini mengindikasikan bahwa dengan peningkatan jumlah *Epochs*, terdapat sedikit penurunan akurasi dalam deteksi sepeda motor. Secara rata-rata, model ini memiliki *Error Relative* sekitar 5.9%. Model dengan 200 *Epochs* menunjukkan hasil serupa dengan model 150 *Epochs*. *Error Relative* untuk label *motorcycle* dan *car* adalah 6.7%, dengan rata-rata *Error Relative* sekitar 5.9%. Ini

mengindikasikan bahwa peningkatan jumlah *Epochs* setelah 100 tidak membawa perbaikan dalam akurasi deteksi kendaraan.

Model dengan 250 *Epochs* menunjukkan hasil yang signifikan, dengan *Error Relative* mencapai 0% untuk kedua label, yaitu *motorcycle* dan *car*. Secara rata-rata, model ini memiliki *Error Relative* sekitar 0.0%. Ini menunjukkan bahwa model ini memiliki akurasi yang sangat baik dalam mengidentifikasi kendaraan pada kedua kategori tersebut. Namun, model dengan 300 *Epochs* menunjukkan sedikit penurunan akurasi dalam deteksi kendaraan. *Error Relative* untuk label *motorcycle* adalah 3.3%, dengan *Error Relative* untuk label *car* tetap pada 0.0%. Secara rata-rata, model ini memiliki *Error Relative* sekitar 1.7%.

Model dengan 350 *Epochs* menunjukkan hasil yang sangat baik, dengan *Error Relative* mencapai 0% untuk kedua label, yaitu *motorcycle* dan *car*. Secara rata-rata, model ini memiliki *Error Relative* sekitar 0.0%. Model dengan 400 *Epochs* menunjukkan hasil yang serupa dengan model 50 *Epochs*, dengan *Error Relative* sekitar 4.2%. Ini mengindikasikan bahwa peningkatan jumlah *Epochs* setelah 350 tidak membawa perbaikan dalam akurasi deteksi kendaraan.

Model dengan 450 *Epochs* menunjukkan hasil yang baik, dengan *Error Relative* untuk label *motorcycle* sekitar 3.3% dan label *car* tetap pada 0.0%. Secara rata-rata, model ini memiliki *Error Relative* sekitar 1.7%. Model dengan 500 *Epochs* menunjukkan sedikit penurunan akurasi dalam deteksi kendaraan, dengan *Error Relative* untuk label *motorcycle* adalah 6.7% dan label *car* tetap pada 0.0%. Secara rata-rata, model ini memiliki *Error Relative* sekitar 3.3%.

Hasil ini menggambarkan perubahan dalam akurasi deteksi kendaraan seiring dengan peningkatan jumlah *Epochs*. Model dengan 250 *Epochs* menunjukkan akurasi yang sangat baik, tetapi peningkatan selanjutnya tampaknya tidak membawa perbaikan signifikan dalam deteksi kendaraan. Dalam konteks ini, kita dapat menyimpulkan bahwa model dengan 250 *Epochs* menunjukkan performa terbaik dalam hal akurasi yang mana mencapai akurasi 100%. Meskipun model dengan jumlah *Epochs* yang lebih tinggi mencapai akurasi yang baik, hasil ini menegaskan bahwa peningkatan *Epochs* tidak selalu berdampak positif pada peningkatan akurasi, dan bahkan dapat menguranginya dalam beberapa kasus. Oleh karena itu, pemilihan model terbaik harus mempertimbangkan faktor efisiensi dan

kinerja secara bersamaan, sesuai dengan konteks proyek yang ada. Dengan demikian, penting untuk merinci kapan dan bagaimana model-model tersebut dapat paling efektif digunakan, serta sejauh mana hasil yang diinginkan dapat dicapai dalam batasan sumber daya yang tersedia.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan rangkuman hasil analisis dan temuan yang diperoleh dalam penelitian ini tentang penggunaan teknologi *Computer Vision* dengan algoritma YOLOv8 untuk analisis pendeteksian dan penghitungan kendaraan pada CCTV jalan raya di Kota Bandar Lampung, dapat disimpulkan:

1. Hasil penelitian menunjukkan bahwa penggunaan teknologi *Computer Vision* dengan algoritma YOLOv8 telah berhasil dalam melakukan pendeteksian dan penghitungan kendaraan pada rekaman video CCTV jalan raya di Kota Bandar Lampung.
2. Berdasarkan analisis, model dengan 250 *Epochs* menunjukkan performa paling baik dalam hal akurasi, presisi, *Recall*, dan *F1-Score* serta penghitungan kendaraan. Hal ini menunjukkan bahwa model ini mampu memberikan keseimbangan yang baik antara pendeteksian objek yang akurat dan jumlah kesalahan yang rendah.
3. Evaluasi model dilakukan dengan membandingkan hasil penghitungan kendaraan secara otomatis dengan hasil penghitungan kendaraan secara manual. Meskipun terdapat kecenderungan model otomatis untuk sedikit menghasilkan overestimasi, namun tingkat kesalahan relatif ini tetap dalam batas yang dapat diterima.
4. *Error Relative* dari model-model yang diuji mengindikasikan bahwa model dengan 250 *Epochs* mengungguli model-model lainnya dalam mengurangi kesalahan pendeteksian kendaraan.
5. Kesimpulan penelitian ini memberikan landasan kuat untuk pengembangan lebih lanjut dalam bidang pendeteksian objek dan penghitungan kendaraan. Hasil penelitian ini dapat memiliki aplikasi penting dalam manajemen lalu lintas dan keamanan jalan raya.
6. Penelitian ini juga menggarisbawahi bahwa pemilihan model yang optimal harus mempertimbangkan faktor kontekstual dan persyaratan proyek yang khusus, serta mempertimbangkan antara akurasi dan efisiensi komputasi.

5.2 Saran

Dari hasil penelitian yang telah dilakukan tentang "Analisis Akurasi Penghitungan Kendaraan Menggunakan Teknologi *Computer Vision* Dengan Algoritma YOLOv8 pada CCTV Jalan Raya di Kota Bandar Lampung", berikut beberapa saran yang dapat menjadi panduan bagi penelitian dan pengembangan selanjutnya di bidang ini:

1. Ekstensifikasi *Dataset*: Perluasan *Dataset* dengan mengumpulkan data dari berbagai sudut pandang dan kondisi lalu lintas yang berbeda di Kota Bandar Lampung dapat membantu meningkatkan keakuratan model dan generalisasi.
2. Penyetelan *Hyperparameter*: Lakukan eksperimen lebih lanjut dalam menyetel parameter-model YOLOv8, seperti ukuran *input*, jumlah lapisan, atau learning rate, untuk mengoptimalkan akurasi deteksi kendaraan.
3. Validasi terhadap Variabilitas Cuaca: Selain melibatkan pengujian pada kondisi cahaya yang optimal, penting juga untuk menguji model dalam berbagai kondisi cuaca seperti hujan, kabut, atau malam hari yang dapat memengaruhi kinerja deteksi.
4. Evaluasi Terhadap *Hardware*: Pertimbangkan penggunaan perangkat keras yang lebih canggih, seperti GPU atau TPU, yang dapat mempercepat pelatihan model dan analisis video secara *Real-time*.
5. Perkembangan Antarmuka Pengguna: Membangun antarmuka pengguna yang lebih intuitif dan *user-friendly* dapat memudahkan penggunaan sistem oleh pihak berwenang atau pengguna akhir.
6. Efisiensi Energi: Selidiki penggunaan daya dan cara untuk mengoptimalkannya agar sistem lebih efisien dalam penggunaan energi.
7. Dampak pada Lalu Lintas: Melakukan penelitian lebih lanjut tentang pengaruh implementasi teknologi ini pada pergerakan lalu lintas, termasuk kemungkinan perbaikan pengaturan lalu lintas.
8. Kerjasama dengan Pihak Terkait: Kolaborasi dengan instansi terkait, seperti otoritas lalu lintas atau pihak berwenang lainnya, untuk pengujian dan implementasi di lapangan.

9. Analisis Biaya-Manfaat: Lakukan analisis biaya-manfaat yang lebih mendalam untuk menilai efektivitas implementasi teknologi ini dalam jangka panjang.

]

DAFTAR PUSTAKA

Adhinata, F. D., Wardhana, A. C., Rakhmadani, D. P., & Jayadi, A. (2020). Peningkatan Kualitas Citra pada Citra Digital Gelap. *Jurnal E-Komtek*, *4*(2), 136–144.

Alnujaidi, K., Alhabib, G., & Alodhieb, A. (2023). Spot-the-Camel: *Computer Vision* for Safer Roads. *ArXiv Preprint ArXiv:2304.00757*.

Amrizal, V., & Aini, Q. (2013). *Kecerdasan Buatan*. repository.uinjkt.ac.id. https://repository.uinjkt.ac.id/dspace/bitstream/123456789/44538/2/naskah%20kecerdasan%20buatan.pdf

Amwin, A. (2021). *Deteksi dan Klasifikasi Kendaraan Berbasis Algoritma You Only Look Once (YOLO)*.

Cholid, F. A. (2021). *Penerapan Metode You Only Look Once (Yolo) Dan Support Vector Regression (Svr) Untuk Perhitungan Kepadatan Lalu Lintas Berdasarkan Area Occupancy*. Universitas Diponegoro.

Diponegoro, M. H., Kusumawardani, S. S., & Hidayah, I. (2021). Tinjauan Pustaka Sistematis: Implementasi Metode Deep Learning pada Prediksi Kinerja Murid. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi*, *10*(2), 131–138.

Fachrie, M. (2020). *A Simple Vehicle Counting System Using Deep Learning with YOLOv3 Model*. https://doi.org/10.13140/RG.2.2.15026.56001

Faisal, M. R., & Abadi, F. (2021). *Implementasi SSD_Resnet50_V1 untuk Penghitung Kendaraan*.

Hibatullah, A. (2019). *Penerapan Metode Convolutional Neural Network Pada Pengenalan Pola Citra Sandi Rumput*. elibrary.unikom.ac.id. https://elibrary.unikom.ac.id/id/eprint/1529/

Hidayat, R., & AH, D. S. (2017). Dampak Kemacetan Terhadap Sosial Ekonomi Pengguna Jalan di Kota Banda Aceh. *Jurnal Ilmiah Mahasiswa Ekonomi Pembangunan*, *2*(1), 176–186.

Isnaini, R. I. (2020). *Aplikasi Penghitung Kendaraan yang Melintas di Jalan Raya Berdasarkan Metode Yolo Object Detection*. Universitas DInamika.

Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A Review of Yolo algorithm developments. *Procedia Computer Science*, *199*, 1066–1073.

Kejriwal, R., Ritika, H. J., & Arora, A. (2022). Vehicle Detection and Counting using Deep Learning basedYOLO and Deep SORT Algorithm for Urban Traffic Management System. *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, 1–6.

KHATAMI, M. S. (2022). *Deteksi Kendaraan Menggunakan Algoritma You Only Look Once (Yolo) V3*.

Majumder, M., & Wilmot, C. (2023). Automated Vehicle Counting from Pre-Recorded Video Using You Only Look Once (YOLO) Object Detection Model. *Journal of Imaging*, *9*, 131. https://doi.org/10.3390/jimaging9070131

Maulana, E., Setianingsih, C., & Paryasto, M. W. (2023). Sistem Deteksi Pelanggaran Kelebihan Penumpang Pada Kendaraan Sepeda Motor Roda Dua Menggunakan Algoritma Faster RCNN. *EProceedings of Engineering*, *10*(1).

Neethu, N. J., & Anoop, B. K. (2015). Role of *Computer Vision* in automatic inspection systems. *International Journal of Computer Applications*, *123*(13).

Ovtcharov, K., Ruwase, O., Kim, J. Y., Fowers, J., & ... (2015). Accelerating deep convolutional neural networks using specialized hardware. *Microsoft Research* …. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=209932cd2e3f5da071c4f6341a3b8b29cf50cc4a

Pramestya, R. H. (2018). Deteksi dan klasifikasi kerusakan jalan aspal menggunakan metode yolo berbasis citra digital. *Institut Teknologi Sepuluh Nopember*. https://repository.its.ac.id/59044/1/06111650010019-Master_Thesis.pdf

Pratama, Y., & Rasywir, E. (2021). Eksperimen Penerapan Sistem Traffic Counting dengan Algoritma YOLO (You Only Look Once) V. 4. *Jurnal Media Informatika Budidarma*, *5*(4), 1438–1446.

Priyambodo, P. (2018). Analisis korelasi jumlah kendaraan dan pengaruhnya terhadap PDRB di Provinsi Jawa Timur. *Warta Penelitian Perhubungan*, *30*(1), 59–65.

Putra, W. S. E. (2016). Klasifikasi citra menggunakan convolutional neural network (CNN) pada caltech 101. *Jurnal Teknik ITS*. http://ejurnal.its.ac.id/index.php/teknik/article/view/15696

Ramadhan, N. M., Krisyantho, D., Irsal, I., & Rizal, M. (2023). PROTOTYPE PENGHITUNG JUMLAH DAN KECEPATAN KENDARAAN OTOMATIS SECARA REAL TIME BERBASIS *COMPUTER VISION* MENGGUNAKAN METODE BACKGROUND SUBTRACTION. *Dipanegara Komputer Teknologi Informatika*, *15*(2), 160–172.

Redmon, J., Divvala, S., Girshick, R., & ... (2016). You only look once: Unified, *Real-time* object detection. *Proceedings of the IEEE …*. https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html

Shorten, C., Khoshgoftaar, T. M., & Furht, B. (2021). Deep Learning applications for COVID-19. *Journal of Big Data*, *8*(1), 1–54.

Sumarudin, A., Darsih, D., Iryanto, I., & Suheryadi, A. (2019). Aplikasi Penghitung Kendaraan Pada Jalur Pantura Menggunakan Blob Deteksi Dan Kalman Filter. *Journal of Applied Informatics and Computing*, *3*(1), 8–11.

Terven, J., & Cordova-Esparza, D. (2023). A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond. *ArXiv Preprint ArXiv:2304.00501*.

Vahab, A., Naik, M. S., Raikar, P. G., & Prasad, S. R. (2019). Applications of object detection system. *International Research Journal of Engineering and Technology (IRJET)*, *6*(4), 4186–4192.

Wahyudi, D. A., & Kartowisastro, I. H. (2011). Menghitung Kecepatan Menggunakan *Computer Vision*. *Universitas Binus*. http://research-dashboard.binus.ac.id/uploads/paper/document/publication/Journal/Teknik%20Komputer/Vol%2019%20No%202%20Agustus%202011/01_Iman_menghitung%20kecepatan-Ok.pdf

Wibisana, H. (2009). Efektifitas model karakteristik arus lalu lintas di ruas jalan raya rungkut madya kota madya surabaya (perbandingan model greenshield dan greenberg). *Jurnal Teknik Sipil Unika Soegijapranata*, *4*(1).

Wu, M. T. (2022). *Confusion Matrix* and minimum cross-entropy metrics based motion recognition system in the classroom. *Scientific Reports*. https://www.nature.com/articles/s41598-022-07137-z

Zhang, S., Wu, G., Costeira, J. P., & Moura, J. M. F. (2017). Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. *Proceedings of the IEEE International Conference on Computer Vision*, 3667–3676.