BAB III

METODE PENELITIAN

3.1. Desain Penelitian

Penelitian ini menggunakan metode Design and Development (D&D) sebagaimana dijelaskan oleh Richey dan Klein (2014), yang menekankan pengembangan produk atau sistem secara sistematis disertai evaluasi untuk menghasilkan pengetahuan praktis maupun teoretis. Aspek kuantitatif dipilih untuk memfasilitasi pengukuran performa sistem melalui data numerik yang terukur. Sifat evaluatif dari penelitian ini sangat relevan untuk menilai efektivitas dan fungsionalitas prototipe dalam kondisi yang mendekati lingkungan nyata. Tujuan utamanya adalah untuk mengukur sejauh mana sistem dapat beroperasi secara efisien, serta mengidentifikasi kekuatan dan kelemahannya berdasarkan metrik performa yang terukur. Inti dari desain penelitian ini terletak pada implementasi model YOLOv8 ke dalam aplikasi web dan kuantifikasi efektivitasnya dalam mendeteksi serta menghitung kendaraan dari rekaman video CCTV.

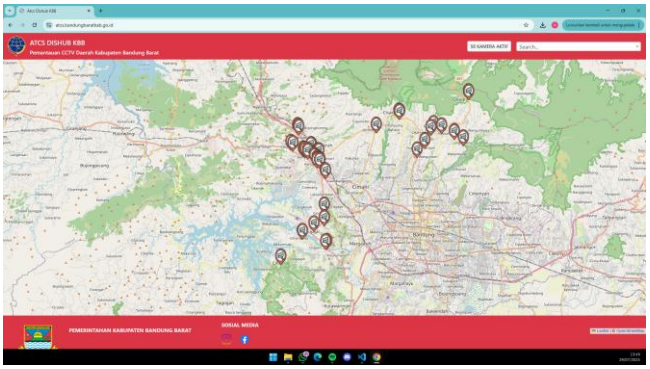
3.2. Objek Penelitian

Objek penelitian ini berfokus pada dua aspek utama yang saling terkait: rekaman video CCTV sebagai data input dan sistem deteksi kendaraan berbasis metode "You Only Look Once" (YOLO) dengan implementasi YOLOv8 sebagai entitas yang diteliti performanya. Pendekatan ini memungkinkan analisis mendalam terhadap kemampuan *deep learning* dalam memecahkan permasalahan monitoring lalu lintas riil.

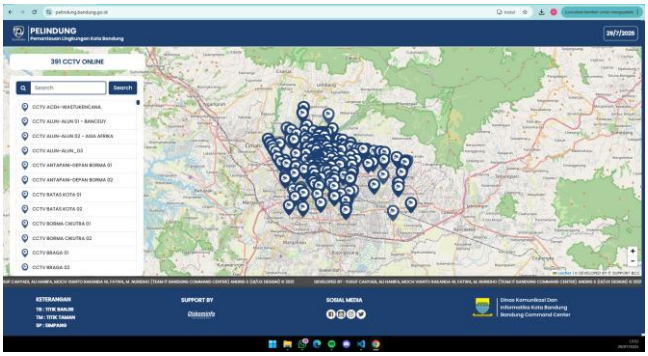
Pertama, rekaman video CCTV yang berisi aktivitas lalu lintas kendaraan menjadi data primer. Pemilihan rekaman video ini didasarkan pada ketersediaannya yang luas dari berbagai sumber dan kemampuannya untuk merepresentasikan kondisi lalu lintas yang dinamis dan beragam dalam skenario perkotaan. Video-video ini akan diperoleh dari empat lokasi spesifik di wilayah Bandung dan sekitarnya: Rumah Sakit Gedebage, Batujajar, Farmhouse, dan Simpang BBS yang didapatkan dari website “PELINDUNG” Pemantauan

Muhammad Azfa Faadhilah, 2025
IMPLEMENTASI SISTEM DETEKSI DAN COUNTING KENDARAAN BERBASIS VIDEO CCTV DENGAN METODE DEEP LEARNING YOLOv8
Universitas Pendidikan Indonesia | repository.upi.edu | perpustakaan.upi.edu

Lingkungan Kota Bandung pada halaman (https://pelindung.bandung.go.id/) pada gambar 3.2 dan website ATCS Dishub Kabupaten Bandung Barat pada halaman (https://atcs.bandungbaratkab.go.id/) pada gambar 3.1.



Gambar 3. 1 *Capture* Halaman web ATCS KBB



Gambar 3. 2 *Capture* Halaman web PELINDUNG

Setiap lokasi akan menyajikan rekaman pada tiga kondisi waktu berbeda: siang, sore, dan malam hari. Variasi kondisi pencahayaan ini krusial untuk menguji *robustness* (ketahanan) sistem terhadap perubahan visibilitas yang signifikan, seperti bayangan kuat di siang hari, bias cahaya dari kendaraan atau minimnya cahaya di malam hari, yang dapat mempengaruhi akurasi deteksi. Durasi setiap rekaman pengujian akan diseragamkan untuk memastikan konsistensi dalam analisis performa selama 2 menit. Jenis kendaraan yang menjadi fokus deteksi dan penghitungan meliputi kategori umum seperti motor, mobil, truk, bis, angkot, dan pickup, yang merupakan jenis kendaraan dominan di Indonesia. Dengan detail klasifikasi jenis kendaraan sebagai berikut:

Tabel 3.1 Klasifikasi Kendaraan

NO	Klasifikasi Kendaraan	Gambar 1	Gambar 2	Keterangan
1	Mobil	 Gambar 3. 3 Ilustrasi Mobil Tipe *Sport* (sumber: https://setirkanan.co.id/article/deretan-mobil-sport-toyota-paling-legend-hingga-saat-ini)	 Gambar 3. 4 Ilustrasi Mobil Tipe SUV (sumber: https://www.tokopedia.com/blog/jenis-mobil-toyota-oto/?utm_source=google&utm_medium=organic)	Termasuk tipe mobil sedan, sport, suv, mpv. Memiliki 4 roda
2	Motor	 *Gambar 3. 5 Ilustrasi Motor Matic/Skuter* (sumber: https://www.tokopedia.com/ellao/variasi-stiker-motor-mio-sporty-2005-merah-basygi-3774ad?utm_source=google&utm_medi	 Gambar 3. 6 Ilustrasi Motor Sport (sumber: https://www.autofun.co.id/motor/bmw/m-1000-rr)	Termasuk tipe motor matic, skuter, sport, bebek. Memiliki 2 roda

		um=organic&utm_campaign=pdp-seo)		
3	Truk	 Gambar 3. 7 Ilustrasi Truk Besar (sumber: https://indoautozone.co.id/jenis-jenis-truk-dan-kegunaannya-dalam-berbagai-ukuran/)	 Gambar 3. 8 Ilustrasi Truk Tronton (sumber: https://sewatrukbalen.com/wp-content/uploads/2019/12/truk-tronton-kontainer.png)	Termasuk tipe truk, tronton, trailer, box. Memiliki ≥ 4 roda
4	Bus	 Gambar 3. 9 Ilustrasi Mini Bus (sumber: https://bangfirman.com/2019/05/toyota-hiace-mini-bus-paling-nyaman-untuk-bisnis-travel-ini-harga-hiace-jakarta-di-auto2000/)	 Gambar 3. 10 Ilustrasi Big Bus (sumber: https://bus-news.com/new-mercedes-benz-euro-vi-touring-coaches-for-taiwanese-market/)	Termasuk mini bus, bus besar, bus double decker. Memiliki ≥ 4 roda

5	Angkot	 Gambar 3. 11 Ilustrasi Angkot (sumber: https://stickearn.com/wp-content/uploads/2025/02/banner-product-stickangkot-32.png)	 Gambar 3. 12 Ilustrasi Angkot Lembang (sumber: https://imgx.gridoto.com/crop/0x0:0x0/700x500/photo/gridoto/2017/11/17/781571208.jpg)	termasuk angkot dengan berbagai tipe mobil dan beragam jenis warna angkot.
6	Pickup	 Gambar 3. 13 Ilustrasi Pickup (sumber: https://www.sunstarmotor.id/mobil/colt-l300/)	 Gambar 3. 14 Ilustrasi Double Cabin (sumber: https://astradigitaldigiroomuat.blob.core.windows.net/storage-uat-001/apa-yang-dimaksud-dengan-double-cabin-pada-hilux.png)	Termasuk pickup dan double cabin . Memiliki 4 roda dengan ruang muatan di belakang mobil.

Total dataset yang digunakan dalam penelitian ini berjumlah 4.767 data, terdiri dari 3.221 data publik dan 1.546 data pribadi yang telah dilabeli dengan *bounding box*. Dataset ini kemudian dibagi menjadi tiga bagian yaitu data pelatihan (*train*), validasi (*valid*), dan pengujian (*test*) dengan rasio perbandingan 80/15/5 dari keseluruhan dataset. Pembagian ini menghasilkan

3.897 data pelatihan, 690 data validasi, dan 180 data pengujian. Jumlah kelas model yang digunakan disesuaikan menjadi 6. Proses *training* dilakukan selama 100 *epoch* dengan ukuran *batch* 8, ukuran gambar 1024 piksel, dan jumlah *workers* 6. Proses ini menggunakan *optimizer* AdamW dan memanfaatkan akselerasi GPU NVIDIA GeForce RTX 3070 8GB (device="cuda") untuk efisiensi komputasi.

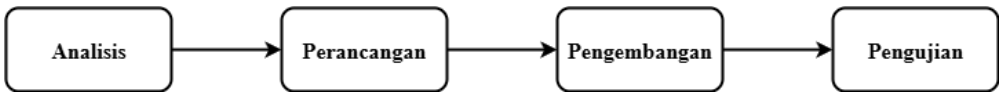
Kedua, objek penelitian juga mencakup sistem deteksi kendaraan itu sendiri, yang diimplementasikan menggunakan metode *deep learning* "You Only Look Once" (YOLOv8). Penelitian ini secara khusus menyelidiki bagaimana implementasi YOLOv8 dapat meningkatkan kecepatan pemantauan dan akurasi identifikasi berbagai jenis kendaraan dalam lingkungan lalu lintas perkotaan yang dinamis dan kompleks. Penelitian ini akan mengeksplorasi sejauh mana kemampuan pemrosesan data single-shot YOLOv8 dapat memberikan dampak positif pada kinerja sistem deteksi kendaraan, terutama dalam menghadapi tantangan yang umum terjadi di Indonesia, seperti kepadatan lalu lintas, variasi ukuran dan bentuk kendaraan, serta kondisi jalan yang beragam. Adaptabilitas YOLOv8 terhadap karakteristik lalu lintas Indonesia, termasuk jenis kendaraan yang dominan dan perilaku lalu lintas lokal, akan menjadi poin krusial dalam analisis. Dengan demikian, objek penelitian ini tidak hanya mencakup aspek teknis dan operasional dari implementasi YOLOv8 dalam deteksi kendaraan, tetapi juga menitikberatkan pada relevansi praktis dan keberlanjutan penggunaannya sebagai solusi inovatif untuk pengelolaan lalu lintas di perkotaan Indonesia.

3.3. Prosedur Penelitian

Penelitian ini dilaksanakan melalui serangkaian prosedur yang sistematis dan terstruktur menggunakan metode D&D, mencakup tahapan-tahapan utama yang saling berhubungan untuk menjamin kelengkapan dan validitas hasil. Proses diawali dengan analisis, di mana permasalahan penelitian diidentifikasi secara mendalam, literatur relevan mengenai *deep learning* YOLOv8 dan teknologi pendukungnya dikaji, serta kebutuhan fungsional sistem dianalisis. Selanjutnya, perenca dilaksanakan dengan mendokumentasikan rekaman video dari empat lokasi spesifik yaitu Rumah Sakit Gedebage, Batujajar, Farmhouse,

dan Simpang BBS yang diambil pada kondisi waktu siang, sore, dan malam hari. Data ground truth manual juga disiapkan melalui anotasi visual pada rekaman tersebut untuk keperluan validasi performa sistem.

Tahap berikutnya, perancangan sistem, melibatkan definisi arsitektur sistem secara komprehensif, termasuk integrasi model YOLOv8, desain logika pelacakan objek, mekanisme penghitungan, dan perancangan antarmuka berbasis web. Selanjutnya, pengembangan sistem dilakukan untuk mengimplementasikan semua rancangan tersebut menjadi sebuah aplikasi yang berfungsi. Fase ini mencakup pengembangan backend menggunakan Flask, integrasi model *deep learning* YOLOv8, pembangunan modul pelacakan dan penghitungan dengan OpenCV-Python, serta pengembangan frontend menggunakan HTML, CSS, dan JavaScript. Setelah sistem berhasil dikembangkan, Pengujian Sistem dilaksanakan untuk memvalidasi fungsionalitas dan mengevaluasi performa kuantitatifnya menggunakan *Confusion matrix* dan metrik *Micro average* pada data video yang telah dikumpulkan dan dianotasi. Terakhir, Analisis Hasil dan Pembahasan dilakukan untuk menginterpretasikan data kuantitatif yang diperoleh dari pengujian, menarik kesimpulan yang relevan, dan merumuskan rekomendasi berdasarkan kinerja sistem secara keseluruhan. Gambar 3.15 menunjukan diagram dari tahapan penelitian ini dengan metode Design & Development (D&D).



Gambar 3. 15 Diagram Tahapan Penelitian D&D (Richey & Klein, 2014)

3.4.Tahap Analisis

Tahap analisis dalam penelitian ini berfokus pada analisis data kuantitatif yang diperoleh dari hasil pengujian model deteksi dan penghitungan kendaraan. Analisis ini dilakukan setelah model deteksi objek YOLOv8 terintegrasi dengan modul pelacakan dan penghitungan selesai dibangun dan diuji pada rekaman video CCTV.

Analisis ini melibatkan:

3.4.1. Perhitungan Total Kendaraan

Menghitung jumlah total kendaraan yang berhasil dideteksi dan dihitung oleh sistem, baik secara keseluruhan maupun terperinci per kelas kendaraan (motor, mobil, truk, bis, angkot, pickup). Penghitungan ini didasarkan pada logika perlintasan garis yang memicu pencatatan satu kali per objek yang teridentifikasi secara unik menggunakan ID.

3.4.2. Evaluasi Metrik Performa Model

Performa model deteksi dan penghitungan diukur secara kuantitatif menggunakan metrik standar dalam bidang *computer vision* dan *deep learning*:

3.4.2.1. *Confusion matrix*

Confusion matrix adalah sebuah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi pada sekumpulan data uji. Dalam konteks penelitian ini, *confusion matrix* akan menyajikan ringkasan performa sistem deteksi dan penghitungan kendaraan untuk setiap kelas (motor, mobil, truk, bis, angkot, pickup) dengan membandingkan hasil prediksi model terhadap kondisi aktual (ground truth). Komponen-komponen utama dalam *confusion matrix* adalah:

1. *True Positive* (TP)

   Mengacu pada jumlah instance atau objek kendaraan dari kelas tertentu yang secara aktual ada dalam frame video dan berhasil dideteksi serta diklasifikasikan dengan benar oleh sistem. Ini adalah prediksi yang benar dan positif.

   Kontekstual: Jika sistem mendeteksi sebuah "mobil" di frame video, dan pada ground truth memang terdapat "mobil" di lokasi yang sesuai, maka ini dihitung sebagai satu *True Positive* untuk kelas "mobil". Dalam konteks penghitungan, ini berarti sebuah

kendaraan aktual dari kelas yang benar melintasi garis dan dicatat oleh sistem.

2. True Negative (TN)

Mengacu pada jumlah instance atau area dalam frame di mana secara aktual tidak ada objek dari kelas tertentu, dan sistem dengan benar tidak mendeteksi objek dari kelas tersebut (atau tidak mengklasifikasikan objek lain sebagai kelas tersebut). Ini adalah prediksi yang benar dan negatif. TN umumnya sulit diukur secara langsung dalam deteksi objek karena melibatkan area yang tidak terhingga yang tidak mengandung objek. Namun, dalam konteks *micro average* yang melihat keseluruhan background atau objek lain sebagai 'negatif' untuk kelas spesifik, TN dapat merefleksikan kemampuan model untuk tidak membuat deteksi palsu di mana seharusnya tidak ada.

Kontekstual: Jika sistem tidak mendeteksi "truk" di suatu area dalam frame, dan pada ground truth memang tidak ada "truk" di area tersebut, maka ini dihitung sebagai satu *True Negative* untuk kelas "truk". Atau, ketika sistem tidak salah mengklasifikasikan sebuah motor sebagai truk.

3. *False Positive* (FP)

Mengacu pada jumlah instance atau deteksi objek oleh sistem yang sebenarnya bukan objek dari kelas yang diprediksi, atau objek dari kelas lain yang salah diklasifikasikan ke kelas tersebut. Ini adalah kesalahan Tipe I, di mana sistem memberikan prediksi positif padahal kenyataannya negatif.

Kontekstual: Jika sistem mendeteksi "mobil" di suatu frame, tetapi pada ground truth area tersebut kosong (tidak ada kendaraan), maka ini adalah *False Positive* untuk kelas "mobil". Jika sistem mendeteksi "truk", tetapi pada ground truth objek tersebut sebenarnya adalah "bis", maka ini adalah *False Positive* untuk

kelas "truk" (dan *False Negative* untuk kelas "bis"). Dalam penghitungan, ini terjadi ketika sistem mencatat adanya kendaraan melintasi garis, padahal tidak ada kendaraan aktual atau itu adalah objek yang salah diklasifikasikan dari kelas lain.

4. *False Negative* (FN)

Mengacu pada jumlah *instance* atau objek kendaraan dari kelas tertentu yang secara aktual ada dalam frame video, tetapi gagal dideteksi oleh sistem (tidak terdeteksi sama sekali) atau salah diklasifikasikan sebagai kelas lain. Ini adalah kesalahan Tipe II, di mana sistem memberikan prediksi negatif padahal kenyataannya positif.

Kontekstual: Jika pada ground truth terdapat "motor" di frame video, tetapi sistem tidak mendeteksinya sama sekali, maka ini adalah *False Negative* untuk kelas "motor". Jika pada ground truth terdapat "pickup", tetapi sistem mendeteksinya dan mengklasifikasikannya sebagai "mobil", maka ini adalah *False Negative* untuk kelas "pickup" (dan *False Positive* untuk kelas "mobil"). Dalam penghitungan, ini terjadi ketika kendaraan aktual melintasi garis tetapi sistem gagal mencatatnya.

Pemahaman dan penghitungan yang akurat dari TP, TN, FP, dan FN ini menjadi dasar untuk menghitung metrik performa utama lainnya seperti *Micro Average Accuracy, Precision, Recall,* dan *F1-Score*, yang akan memberikan gambaran komprehensif mengenai kinerja sistem deteksi dan penghitungan kendaraan.

3.4.2.2. Micro average

Berdasarkan total TP, FN, dan FP yang disatukan dari semua kelas, dihitunglah *Accuracy, Precision, Recall*, dan *F1-Score* secara *micro average*. Metode *micro average* dipilih secara khusus untuk memberikan gambaran performa model secara keseluruhan, tanpa bias terhadap kelas dengan jumlah sampel yang mungkin tidak

seimbang, karena ia memperlakukan setiap instance prediksi (baik benar maupun salah) secara setara di seluruh kelas (Sokolova, M., & Lapalme, G., 2009). Rumus perhitungan untuk masing-masing metrik ini akan diterapkan untuk setiap hasil pengujian sistem.

3.4.3. Analisis Deskriptif

Hasil metrik performa disajikan dalam bentuk tabel dan grafik untuk memudahkan interpretasi. Analisis deskriptif juga mencakup perbandingan performa model di berbagai lokasi pengujian (Rumah Sakit Gedebage, Batujajar, Farmhouse, Simpang BBS) dan pada tiga kondisi waktu yang berbeda (siang, sore, malam), untuk mengidentifikasi pola atau perbedaan kinerja model di bawah variasi kondisi lingkungan dan pencahayaan.

3.4.4. Interpretasi Hasil

Temuan kuantitatif yang diperoleh akan diinterpretasikan secara mendalam. Ini melibatkan perbandingan performa model dengan data ground truth manual yang telah disiapkan sebelumnya, untuk menilai akurasi, efektivitas, dan ketahanan model secara objektif. Pembahasan juga akan mengaitkan hasil analisis dengan landasan teori *deep learning* dan YOLOv8. Analisis ini juga akan mengidentifikasi kekuatan, kelemahan, dan potensi perbaikan pada model deteksi dan penghitungan.

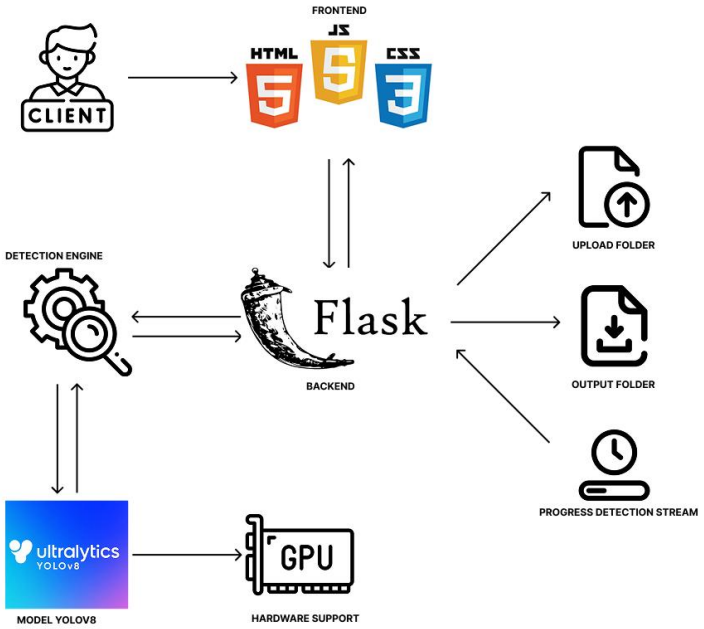
3.5. Tahap Perancangan

Tahap perancangan merupakan fase krusial dalam penelitian ini di mana desain arsitektur sistem dan semua komponen utamanya ditentukan secara detail sebelum proses pengembangan dimulai. Fokus utamanya adalah menerjemahkan kebutuhan fungsional dan non-fungsional ke dalam spesifikasi teknis dan desain logis yang jelas.

3.5.1. Perancangan Arsitektur Sistem Deteksi

Pada tahap ini, arsitektur keseluruhan sistem deteksi dan penghitungan kendaraan dirancang. Ini mencakup penentuan

bagaimana model *deep learning* YOLOv8 akan diintegrasikan sebagai inti pemrosesan deteksi. Perancangan juga meliputi alur data mulai dari input video, pemrosesan frame oleh model, hingga output hasil deteksi. Perancangan modul pelacakan objek dengan desain detail untuk modul pelacakan objek dilakukan untuk memastikan setiap kendaraan yang terdeteksi dapat diidentifikasi secara unik dan pergerakannya terlacak secara konsisten antar frame video. Hal ini vital untuk akurasi penghitungan, mencegah duplikasi atau penghilangan objek. Logika untuk mempertahankan object ID dan memperbarui posisi objek akan dirancang. Pada Gambar 3.16 menunjukan rancangan diagram arsitektur sistem dengan aplikasi berbasis *website*.



Gambar 3. 16 Diagram Arsitektur Sistem

Secara umum sistem ini menggunakan model monolitik di mana semua komponen seperti *frontend, backend,* dan modul pemrosesan objek berada dalam satu kesatuan aplikasi web. Desain ini bertujuan untuk meminimalkan kompleksitas dan memaksimalkan integrasi antar komponen. Pada bagian *frontend* menggunakan *user interface* dan *user experience* yang dibuat menggunakan HTML, CSS untuk struktur tampilan dan JavaScript untuk fungsionalitas, yang memungkinkan pengguna mengunggah video, menggambar garis, dan memantau pemroresan deteksi video melalui *Server-Sent Events* (SSE).

Bagian *backend* sistem ini adalah inti yang dibangun dengan *framework* web Flask. *Backend* berperan sebagai *bridge* atau yang menjembatani, mengelola permintaan dari *frontend* dan memicu proses di *detection engine*. Ia menyediakan *endpoint* API untuk mengunggah video ke upload folder dan menyimpan koordinat garis yang telah dibuat. Selain itu , *backend* juga mengelola pengiriman data progres secara terus menerus ke *user.* Inti dari aplikasi ini adalah sistem deteksi yang dibuat menggunakan python. *Engine* ini mengintregasikan model YOLOv8 untuk mendeteksi objek. Untuk mempercepat analisis, sistem dirancang untuk mendeteksi ketersediaan GPU dan secara otomatis memanfaatkannya. *Detection engine* memproses video dari *upload folder,* menerapkan model AI, dan menghasilkan video *output* yang telah dianotasi oleh model ke *output folder* sebelum akhirnya siap di unduh oleh *user*.

3.5.2. Perancangan Logika Penghitungan

Tahap ini fokus pada bagaimana mekanisme penghitungan kendaraan akan bekerja. Ini melibatkan penentuan konsep garis virtual yang akan digambar oleh pengguna pada video. Perancangan logika akan mendefinisikan bagaimana sistem mendeteksi bahwa sebuah objek telah melintasi garis tersebut, dan bagaimana penghitungan akan dipicu hanya sekali per objek yang melintas, memanfaatkan informasi dari modul pelacakan objek.

3.5.3. Perancangan User Interface Berbasis Web

Aspek user *experience* dan *user interfac*e menjadi perhatian utama dalam perancangan antarmuka. Desain akan mencakup tata letak halaman web untuk mengunggah video, area visual untuk menentukan koordinat garis deteksi pada frame video, serta bagian untuk menampilkan progres deteksi secara real-time dan menyajikan hasil akhir (total hitungan, hitungan per kelas, dan link unduh video hasil). Perancangan ini bertujuan untuk memastikan sistem mudah digunakan dan memberikan informasi yang relevan kepada pengguna.

3.6. Tahap Pengembangan

Tahap pengembangan merupakan fase krusial dalam penelitian ini, di mana seluruh rancangan konseptual dan arsitektur sistem yang telah

dirumuskan pada tahap perancangan diimplementasikan menjadi sebuah prototipe fungsional. Fase ini esensial untuk menerjemahkan spesifikasi teknis dan desain logis ke dalam representasi kode program yang dapat dieksekusi, serta mengintegrasikan berbagai komponen dan pustaka yang diperlukan untuk konstruksi sistem deteksi dan penghitungan kendaraan berbasis video CCTV. Proses pengembangan ini bersifat iteratif, memungkinkan optimisasi dan penyesuaian berkelanjutan sepanjang siklus implementasi. Pengembangan sistem ini distrukturkan menjadi dua pilar utama yang saling terintegrasi dan berinteraksi secara harmonis:

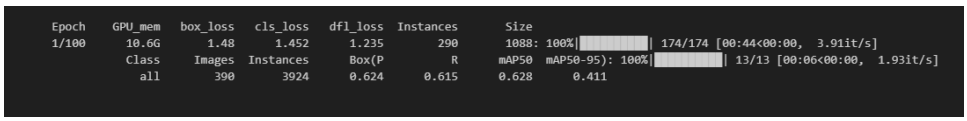
3.6.1. Pengembangan *Model Detection* dan *Counting*

Bagian ini memfokuskan pada konstruksi logika fundamental untuk identifikasi, pelacakan, dan kuantifikasi kendaraan dari rekaman video. Pustaka OpenCV-Python, Ultralytics, dan PyTorch merupakan fondasi teknis dalam implementasi ini. Pada Gambar 3.17 menunjukan proses atau kode untuk instalasi YOLOv8

Gambar 3. 17 Penggalan Kode Instalasi YOLOv8

Implementasi model YOLOv8, sebagai inti kemampuan deteksi, dilakukan melalui pemuatan model terlatih (best.pt) menggunakan pustaka Ultralytics. Tahap ini memastikan model terintegrasi secara optimal dalam alur pemrosesan video. Penentuan perangkat komputasi (GPU atau CPU) difasilitasi oleh PyTorch melalui pengecekan ketersediaan CUDA, dengan model ditempatkan pada perangkat yang paling efisien untuk memaksimalkan performa inferensi. Pengelolaan memori GPU juga dioptimalkan melalui prosedur pembersihan cache CUDA, yang esensial untuk menjaga stabilitas sistem selama

pemrosesan video berdurasi panjang. Pada Gambar 3.18 menunjukan proses training model YOLOv8 pada Jupyter Notebook.



Gambar 3. 18 Penggalan Proses *Training* model TOLOv8

Menyusul proses deteksi awal oleh YOLOv8, modul object tracking diimplementasikan untuk mempertahankan identitas unik setiap kendaraan yang terdeteksi secara konsisten antar frame. Kemampuan pelacakan bawaan dari *library* Ultralytics dimanfaatkan untuk menghasilkan dan mengelola ID tracking yang persisten untuk setiap objek seperti yang ada pada Gambar 3.19 yang menunjukan kode logika untuk *ID Tracking*. Secara internal, posisi pusat objek (centroid) dari frame sebelumnya dan frame saat ini disimpan untuk setiap ID yang terlacak, data ini krusial untuk inferensi lintasan gerak objek.

```
156                 for box in boxes:
157                     x1, y1, x2, y2 = map(int, box.xyxy[0])
158                     obj_id = int(box.id[0])
159                     class_name = model.names[int(box.cls[0])]
160                     center = ((x1 + x2) // 2, (y1 + y2) // 2)
161
162                     if obj_id not in tracked:
163                         tracked[obj_id] = {'prev_center': center, 'curr_center': center}
164                     else:
165                         tracked[obj_id]['prev_center'] = tracked[obj_id]['curr_center']
166                         tracked[obj_id]['curr_center'] = center
```

Gambar 3. 19 Penggalan kode logika ID *tracking*

Logika penghitungan dikembangkan berdasarkan konsep garis virtual yang didefinisikan oleh pengguna. Algoritma geometris, yang melibatkan prinsip orientasi tiga titik dan persimpangan segmen garis, diimplementasikan untuk mendeteksi secara matematis apakah lintasan gerak objek (dari posisi sebelumnya ke posisi saat ini) telah memotong segmen garis perhitung yang ditetapkan. Identifikasi ID objek yang belum pernah dihitung sebelumnya menjadi syarat utama untuk pemicuan penambahan jumlah kendaraan pada kategori kelas yang bersangkutan, sehingga memastikan kuantifikasi yang akurat tanpa duplikasi. Seperti pada Gambar 3.20 yang menunjukan logika dari

perhitungan kendaraan dan pada Gambar 3.21 ilustrasi garis perhitungan.

```python
if obj_id not in tracked:
    tracked[obj_id] = {'prev_center': center, 'curr_center': center}
else:
    tracked[obj_id]['prev_center'] = tracked[obj_id]['curr_center']
    tracked[obj_id]['curr_center'] = center

if obj_id not in counted_ids:
    if intersect(tracked[obj_id]['prev_center'], tracked[obj_id]['curr_center'], line_start, line_end):
        counted_ids.add(obj_id)
        count_per_class[class_name] = count_per_class.get(class_name, 0) + 1
        logger.info(f"✅ {class_name} #{obj_id} melintasi garis.")

color = (0, 255, 0) if obj_id in counted_ids else (255, 182, 0)
cv2.rectangle(output_frame, (x1, y1), (x2, y2), color, 2)
label = f"{class_name} #{obj_id}"
(w_text, h_text), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 2)
cv2.rectangle(output_frame, (x1, y1 - h_text - 5), (x1 + w_text, y1), color, -1)
cv2.putText(output_frame, label, (x1, y1 - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)
```

Gambar 3. 20 Penggalan kode logika penghitungan

Gambar 3. 21 Ilustrasi garis penghitungan

Library OpenCV-Python dimanfaatkan secara ekstensif untuk semua operasi *computer vision* yang berkaitan dengan manipulasi video. Ini mencakup akuisisi frame individual dari video input (cv2.VideoCapture()) dan penulisan frame yang telah diproses ke file video output (cv2.VideoWriter()). Selain itu, OpenCV digunakan untuk menggambar anotasi visual secara langsung pada frame video, seperti bounding boxes di sekitar kendaraan, garis penghitung, serta menyematkan informasi tekstual (label kelas, ID objek, dan ringkasan hitungan total dan per kelas) menggunakan fungsi cv2.putText(), yang secara signifikan meningkatkan interpretasi visual dari hasil deteksi.

```
        color = (0, 255, 0) if obj_id in counted_ids else (255, 182, 0)
        cv2.rectangle(output_frame, (x1, y1), (x2, y2), color, 2)
        label = f"{class_name} #{obj_id}"
        (w_text, h_text), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 2)
        cv2.rectangle(output_frame, (x1, y1 - h_text - 5), (x1 + w_text, y1), color, -1)
        cv2.putText(output_frame, label, (x1, y1 - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

cv2.line(output_frame, line_start, line_end, (0, 255, 255), 3)

total_count = sum(count_per_class.values())
info_text = f"Total Terdeteksi: {total_count}"
cv2.putText(output_frame, info_text, (20, 40), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 0), 3)
cv2.putText(output_frame, info_text, (20, 40), cv2.FONT_HERSHEY_DUPLEX, 1, (255, 255, 255), 2)

y_offset = 70
for class_name, count in count_per_class.items():
    class_text = f"- {class_name}: {count}"
    cv2.putText(output_frame, class_text, (20, y_offset), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,0), 2)
    cv2.putText(output_frame, class_text, (20, y_offset), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,255), 1)
    y_offset += 25

out.write(output_frame)
```

Gambar 3. 22 Penggalan kode Logika Visualisasi Deteksi dengan OpenCV

Gambar 3.22 dapat dilihat menunjukan logika visualisasi deteksi dengan OpenCV.

3.6.2. Pengembangan Antarmuka Aplikasi Web

Pengembangan Backend Aplikasi Web (Flask): Sisi backend sistem diimplementasikan menggunakan framework Flask, yang berfungsi sebagai API server sentral untuk manajemen komunikasi. Pengembangan ini mencakup perancangan dan implementasi berbagai *endpoint* API (misalnya, /upload, /set_line, /detect, /progress_stream, /stop_detection, /download/<filename>). Fungsi-fungsi ini memfasilitasi penerimaan permintaan dari frontend, pemrosesan data terkait (seperti file video yang diunggah dan koordinat garis), serta pengiriman respons dalam format yang sesuai (JSON atau file). Struktur direktori untuk penyimpanan file input (uploads), file output (outputs), dan file konfigurasi (line_position.json) juga dikonfigurasi pada tahap ini.

Pengembangan Frontend Aplikasi Web (HTML, CSS, JavaScript): Antarmuka pengguna (frontend) dikembangkan untuk memfasilitasi interaksi intuitif dengan sistem. Struktur dasar halaman (index.html) dibangun menggunakan HTML, yang mendefinisikan elemen-elemen presentasi dan interaksi. Gaya visual dan tata letak diatur oleh CSS (static/style.css) untuk memastikan tampilan yang responsif dan estetis.

JavaScript (static/index.js) merupakan komponen utama yang mengelola seluruh logika interaktivitas sisi klien. Ini mencakup implementasi fitur penanganan unggah file video, kemampuan untuk menggambar dan mengatur koordinat garis deteksi secara visual pada elemen HTML <canvas> dengan konversi koordinat yang akurat, pengiriman data ke backend melalui permintaan API, serta pengelolaan koneksi Server-Sent Events (SSE) untuk menerima dan menampilkan pembaruan progres deteksi secara real-time. Selain itu, JavaScript bertanggung jawab untuk menyajikan hasil deteksi akhir, termasuk statistik rinci dan tautan unduh video keluaran yang telah dianotasi, serta mengimplementasikan fungsi kontrol seperti tombol reset dan penghentian proses.

Melalui kedua bagian pengembangan ini, semua komponen individu dirangkai menjadi satu kesatuan sistem yang kohesif dan fungsional, menjadikan sistem siap untuk tahap pengujian dan evaluasi performa secara komprehensif.

3.7. Tahap Pengujian

Tahap pengujian merupakan fase krusial dalam siklus hidup penelitian ini, yang berfungsi ganda untuk memvalidasi fungsionalitas sistem yang telah dikembangkan dan mengevaluasi performanya secara kuantitatif pada data riil. Fase ini esensial untuk memastikan bahwa sistem deteksi dan penghitungan kendaraan berbasis YOLOv8 beroperasi sesuai dengan spesifikasi yang dirancang dan mampu memberikan hasil yang akurat serta dapat diandalkan.

Pengujian dalam penelitian ini terbagi menjadi dua jenis utama, yaitu pengujian fungsional dan pengujian performa kuantitatif:

3.7.1. Pengujian Fungsional (Functional Testing)

Pengujian fungsional difokuskan untuk memverifikasi apakah setiap fitur dan modul yang diimplementasikan dalam sistem beroperasi sesuai dengan persyaratan fungsional yang telah ditetapkan pada tahap analisis kebutuhan. Proses ini dilakukan melalui interaksi langsung

dengan antarmuka web yang telah dibangun, memastikan setiap alur kerja berjalan sebagaimana mestinya.

Aspek-aspek yang diuji dalam pengujian fungsional meliputi:

1. Fungsionalitas Unggah Video: Memastikan pengguna dapat berhasil mengunggah file video dalam berbagai format yang didukung, dan file tersebut tersimpan dengan benar di direktori uploads pada server.
2. Fungsionalitas Pengaturan Garis Deteksi: Memverifikasi bahwa pengguna dapat secara visual menentukan dua titik pada frame video yang akan membentuk garis penghitung. Ini termasuk validasi konversi koordinat dari kanvas ke dimensi video asli dan penyimpanan koordinat garis yang benar ke dalam file line_position.json.
3. Pemicuan Proses Deteksi: Memastikan bahwa klik tombol "Mulai Deteksi" berhasil mengirimkan permintaan ke backend dan memicu fungsi run detection di modul deteksi.
4. Pembaruan Progres Real-time: Mengonfirmasi bahwa antarmuka pengguna menerima dan menampilkan pembaruan progres deteksi secara real-time melalui koneksi Server-Sent Events (SSE), termasuk persentase progres, jumlah objek terdeteksi, dan hitungan per kelas yang sedang berlangsung.
5. Fungsionalitas Penghentian Proses: Memastikan bahwa sinyal dari tombol "Reset" atau "Hentikan Deteksi" berhasil diterima oleh backend (stop_detection *endpoint*) dan secara efektif menghentikan proses deteksi yang sedang berjalan pada modul detect.py melalui mekanisme STOP_PROCESSING_FLAG.
6. Generasi Video Output: Memverifikasi bahwa setelah proses deteksi selesai, sistem berhasil menghasilkan file video baru yang telah dianotasi (dengan bounding boxes, label, garis penghitung, dan informasi hitungan) dan menyimpannya di direktori outputs.
7. Fungsionalitas Unduh Hasil: Memastikan bahwa pengguna dapat mengunduh file video hasil deteksi dari server dan melihat ringkasan hasil penghitungan secara akurat di antarmuka web.

Pengujian fungsional ini akan dilakukan secara sistematis untuk setiap fitur, mencatat keberhasilan atau kegagalan, dan melakukan perbaikan jika ditemukan bug atau ketidaksesuaian.

3.7.2. Pengujian Performa Kuantitatif (Quantitative Performance Testing)

Setelah sistem terbukti fungsional, pengujian performa kuantitatif dilakukan untuk mengukur efektivitas dan akurasi model deteksi dan penghitungan secara objektif. Pengujian ini menggunakan data video yang representatif dari lingkungan lalu lintas yang beragam. Pendekatan ini merupakan bagian integral dari desain penelitian kuantitatif yang bertujuan untuk memperoleh data terukur dan dapat digeneralisasi mengenai kinerja sistem (Creswell, J. W., & Creswell, J. D., 2018).

1. Dataset Pengujian: Pengujian akan dilaksanakan pada 12 rekaman video CCTV yang telah dikumpulkan, yang terdiri dari 4 lokasi berbeda (Rumah Sakit Gedebage, Batujajar, Farmhouse, Simpang BBS) dengan masing-masing lokasi memiliki rekaman pada 3 kondisi waktu berbeda (siang, sore, dan malam). Setiap rekaman video memiliki durasi standar (misalnya, 2 menit) untuk memastikan konsistensi dalam perbandingan.
2. Data Ground Truth: Setiap frame relevan dari video pengujian akan dilengkapi dengan data ground truth manual yang telah dibuat secara teliti. Data ini berfungsi sebagai referensi kebenaran absolut untuk:

a. Jumlah aktual objek kendaraan (MOTOR, MOBIL, TRUK, BIS, ANGKOT, PICKUP) dalam setiap frame.
b. Posisi bounding box yang akurat untuk setiap objek.
c. Jumlah aktual objek kendaraan yang melintasi garis penghitung yang telah ditentukan.

3. Protokol Eksekusi: Sistem akan dijalankan untuk memproses setiap dari 12 rekaman video tersebut. Untuk setiap video, output sistem (yaitu, jumlah kendaraan yang dideteksi per kelas dan total) akan dicatat secara otomatis. Selama proses ini, model YOLOv8 akan menjalankan

inferensi pada setiap frame, diikuti oleh modul pelacakan objek dan logika penghitungan garis.

4. Matrik Evaluasi Performa: Performa sistem akan dievaluasi menggunakan matrik kuantitatif standar dalam computer vision dan *deep learning*:

    a. *Confusion matrix*: Untuk setiap video dan setiap kelas kendaraan, sebuah *confusion matrix* akan dibangun. Matriks ini akan merangkum:

    i. *True Positive* (TP): Jumlah kendaraan dari kelas tertentu yang secara aktual ada dan berhasil dideteksi serta dihitung dengan benar oleh sistem.

    ii. *True Negative* (TN): Jumlah kendaraan di mana model dengan benar memprediksi kelas negatif pada kendaraan yang memang sebenarnya negatif.

    iii. *False Negative* (FN): Jumlah kendaraan dari kelas tertentu yang secara aktual ada tetapi gagal dideteksi atau dihitung oleh sistem, atau salah diklasifikasikan ke kelas lain.

    iv. *False Positive* (FP): Jumlah deteksi atau hitungan kendaraan yang dilakukan oleh sistem, padahal tidak ada kendaraan aktual atau itu adalah objek dari kelas lain yang salah diklasifikasikan.

    b. *Micro average Metrics*: Berdasarkan total TP, FN, dan FP yang diagregasikan dari semua kelas kendaraan di setiap sesi pengujian, metrik performa *Accuracy, Precision, Recall,* dan *F1-Score* akan dihitung secara *micro average*. Pemilihan *micro average* memastikan bahwa metrik merefleksikan performa keseluruhan sistem secara global, mengurangi dampak dari potensi ketidakseimbangan jumlah sampel antar kelas. Rumus umum untuk masing-masing metrik ini akan diterapkan.

    i. *Accuracy*

Accuracy mengukur proporsi total prediksi yang benar (baik itu prediksi positif yang benar maupun prediksi negatif yang benar) dari seluruh jumlah observasi. Ini memberikan gambaran umum mengenai seberapa sering model membuat prediksi yang benar.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Keterangan:

a. *True Positive* (TP). Jumlah prediksi benar untuk kelas positif
b. *True Negative* (TN). Jumlah prediksi benar untuk kelas negatif
c. *False Positive* (FP). Jumlah prediksi salah untuk kelas positif
d. *False Negative* (FN). Jumlah prediksi salah untuk kelas negatif

Akurasi menunjukkan persentase frame atau instance di mana sistem dengan benar mendeteksi objek yang ada dan tidak mendeteksi objek di mana seharusnya tidak ada. Dalam konteks penghitungan, ini mencerminkan sejauh mana sistem secara keseluruhan benar dalam mengidentifikasi atau mengabaikan keberadaan kendaraan.

ii. *Precision*

Precision mengukur proporsi prediksi positif yang benar (TP) dari semua *instance* yang diprediksi sebagai positif (yaitu, gabungan *True Positive* dan *False Positive*). Metrik ini penting untuk menilai seberapa andal model ketika ia mengklaim telah mendeteksi sesuatu; presisi tinggi berarti model memiliki tingkat *false positive* yang rendah. Berikut merupakan rumus dalam menghitung *Precision :*

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

Precision menunjukkan seberapa besar proporsi kendaraan yang benar-benar terdeteksi adalah kendaraan yang valid. Presisi tinggi berarti ketika sistem mengidentifikasi sebuah kendaraan (misalnya, sebuah "MOBIL"), kemungkinan besar itu memang "MOBIL" yang sebenarnya dan bukan deteksi palsu

(misalnya, bukan bayangan atau objek lain yang salah diklasifikasi).

iii. *Recall*

Recall (juga dikenal sebagai sensitivitas atau *True Positive Rate*) mengukur proporsi aktual positif yang teridentifikasi dengan benar (TP) dari semua instance yang sebenarnya positif (yaitu, gabungan *True Positive* dan *False Negative*). Metrik ini penting untuk menilai seberapa baik model dalam menemukan semua instance positif yang relevan; *Recall* tinggi berarti model memiliki tingkat *false negative* yang rendah.

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

Recall menunjukkan seberapa besar proporsi semua kendaraan aktual yang berhasil dideteksi oleh sistem. *Recall* tinggi berarti sistem mampu mendeteksi sebagian besar kendaraan yang benar-benar ada di video, meminimalkan jumlah kendaraan yang terlewatkan.

iv. *F1-Score*

F1-Score adalah rata-rata harmonik dari *Precision* dan *Recall*. Metrik ini sangat berguna ketika terdapat ketidakseimbangan kelas dalam data, atau ketika kita ingin keseimbangan antara *Precision* dan *Recall*. *F1-Score* memberikan ukuran tunggal yang mempertimbangkan baik *false positives* maupun *false negatives*.

$$F1\ score = \frac{2 \times Precison \times Recall}{Precision + Recall}$$

5. Penyajian dan Agregasi Data Hasil:

Hasil *micro average* akan dikompilasi ke dalam tabel terstruktur untuk setiap sesi pengujian (misalnya, RUMAH SAKIT GEDEBAGE

- SIANG). Selanjutnya, data akan dikumpulkan untuk menghitung *micro average* per lokasi (misalnya, rata-rata semua pengujian di Rumah Sakit Gedebage), rata-rata per kondisi waktu (misalnya, rata-rata semua pengujian di waktu Siang), dan rata-rata performa keseluruhan dari semua pengujian. Data ini dapat divisualisasikan menggunakan grafik atau chart untuk mempermudah perbandingan dan analisis tren.

Tahap pengujian yang detail ini akan menghasilkan data kuantitatif yang kuat, yang kemudian akan menjadi dasar analisis mendalam dan pembahasan dalam Bab IV untuk menarik kesimpulan yang valid mengenai kinerja sistem