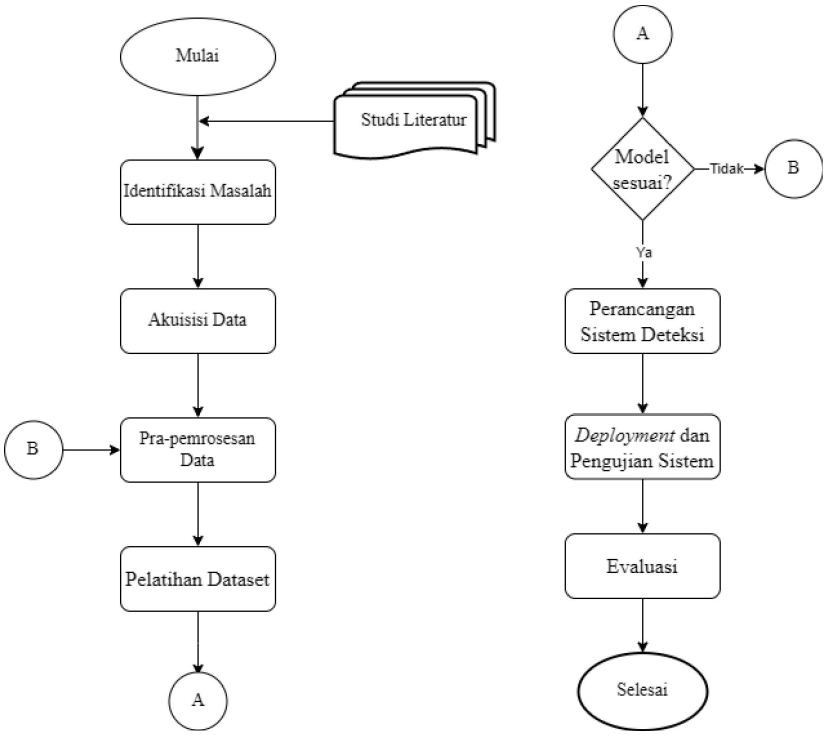
BAB III
METODOLOGI PENELITIAN

Dalam penelitian ini, diimplementasikan sebuah sistem deteksi helm menggunakan *Convolutional Neural Network* (CNN) sebagai metodologi. Model yang digunakan adalah YOLOv8, sebuah model deteksi objek *state-of-the-art* yang telah dioptimalkan untuk kecepatan dan akurasi yang lebih tinggi dari versi YOLO sebelumnya. Sistem ini dirancang untuk melakukan deteksi secara *real-time*, dan akan menggunakan platform Jetson Nano sebagai prosesor pengolahan modelnya serta kamera *webcam* sebagai sumber *input* data.

3.1 Diagram Metodologi

Tahapan-tahapan tertentu harus dijalani dalam penelitian ini untuk mencapai hasil yang diharapkan, termasuk di dalamnya adalah sebagai berikut.



Gambar 3.1 Diagram Alir Penelitian

3.2 Komponen Penelitian

Berikut adalah komponen yang dibutuhkan dalam penelitian ini, sebagai berikut:

1. *Python*

   Python merupakan bahasa pemrograman yang akan digunakan dalam penelitian ini, komponen utama ini memiliki banyak *library* dan *framework* yang berguna dalam pengolahan data serta pengembangan sistem kecerdasan buatan. *Python* akan memproses input data dari *webcam*, melakukan *pre-processing* data, membuat model deteksi helm pada pengendara motor menggunakan algoritma *deep learning* seperti YOLOv8, menguji model, dan menampilkan output berupa hasil deteksi helm pada pengendara motor dalam suatu bentuk visualisasi. Dalam proses tersebut, Python akan memanfaatkan *library* dan *framework* untuk mempercepat dan mempermudah proses pengolahan data dan pembuatan model.

2. LabelImg

   LabelImg adalah alat untuk memberi anotasi pada gambar, dibuat menggunakan Python dan Qt untuk antarmuka grafisnya. Anotasi yang dihasilkan disimpan dalam file XML dengan format PASCAL VOC, dan juga mendukung format YOLO. Tujuan komponen LabelImg pada penelitian ini adalah untuk memberikan label atau kategori pada gambar-gambar pengendara motor dengan menggunakan kotak pembatas (*bounding box*) yang menunjukkan lokasi helm pada gambar. Label ini akan digunakan sebagai data latih untuk model YOLOv8 yang akan mendeteksi pelanggaran helm pengendara motor.

3. Google Colab

   Google Colab digunakan sebagai alat untuk membangun dan mengembangkan model deteksi helm menggunakan algoritma YOLOv8. Google Colab menyediakan lingkungan kerja berbasis web yang memungkinkan penulisan dan eksekusi kode Python secara interaktif. Dengan menggunakan Google Colab, proses pembuatan model sistem deteksi helm dapat dijelaskan dengan lebih terperinci dan terstruktur. Selain itu, akses gratis ke GPU T4 yang disediakan oleh Google Colab

mempercepat pelatihan model, yang sangat berguna dalam mengoptimalkan performa deteksi.

4. ClearML

   Penggunaan ClearML dalam penelitian ini membantu dalam memantau dan mengoptimalkan performa model YOLOv8. Sebagai bagian dari infrastruktur pengembangan sistem deteksi helm, ClearML diadopsi untuk memperkuat proses pelacakan dan manajemen eksperimen. Platform ini memungkinkan otomatisasi alur kerja pembelajaran mesin, mulai dari pelacakan eksperimen, manajemen dataset, hingga eksekusi eksperimen jarak jauh. Dengan ClearML, eksperimen yang dilakukan dapat direproduksi dengan konsistensi, memudahkan proses verifikasi dan validasi hasil.

5. Laptop

   Laptop atau *personal computer* ini digunakan untuk pra-pemrosesan data serta pelatihan dataset dengan basis YOLOv8 untuk model latih yang digunakan untuk sistem deteksi. Berikut spesifikasi laptop yang digunakan pada penelitian ini yang tertera pada Tabel 3.1.

Tabel 3.1 Spesifikasi Laptop

Komponen	**Spesifikasi**
Model Sistem	ASUS Vivobook 15 X512DA
Prosesor	AMD Ryzen 7 3700U *with* Radeon Vega Mobile Gfx (8 CPUs), ~2.3GHz
GPU	AMD Radeon(TM) RX Vega 10 Graphics
Storage	1TB SSD
Sistem Operasi	Windows 10 Home 64-bit (10.0, Build 19044)
Dimensi	357.2 x 230.4 x 19.9 mm
Memori	16GB RAM

6. Jetson Nano

Jetson Nano digunakan pada berbagai aplikasi *embedded system* yang membutuhkan kemampuan AI, seperti robotika, kendaraan otonom, dan sistem deteksi objek. Jetson Nano didukung oleh berbagai *library* dan *framework machine learning* populer seperti PyTorch, TensorFlow, Keras, serta CUDA sehingga memudahkan pengembangan dan implementasi sistem pada *platform* ini. Berikut spesifikasi Jetson Nano yang digunakan pada penelitian ini yang tertera pada Tabel 3.2.

Tabel 3.2 Spesifikasi Jetson Nano

Komponen	Spesifikasi
GPU	128-core Maxwell
CPU	Quad-core ARM A57 64-bit
Memori	4 GB LPDDR4
Storage	16 GB eMMC 5.1
Video Encode	1x 4K @ 30 (HEVC) atau 2x 1080p @ 60 (HEVC) atau 4x 1080p @ 30 (HEVC) atau 4x 720p @ 60 (HEVC) atau 9x 720p @ 30 (HEVC)
Video Decode	1x 4K @ 60 (HEVC) atau 2x 4K @ 30 (HEVC) atau 4x 1080p @ 60 (HEVC) atau 8x 1080p @ 30 (HEVC) atau 9x 720p @ 60 (HEVC)
Kamera	12 lanes MIPI CSI-2 D-PHY
Konektifitas	Gigabit Ethernet, M.2 Key E
Display	HDMI 2.0 dan eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B
Powe rate	5W-10W

7. *Webcam*

Webcam digunakan sebagai sumber data masukan yang *real-time* untuk sistem deteksi helm yang menggunakan model YOLOv8 pada penelitian ini. *Webcam* merekam gambar atau video dari orang-orang yang menggunakan atau tidak menggunakan helm pada jalan raya secara kontinu, lalu mengirimkannya ke Jetson Nano untuk diproses oleh model YOLOv8 dengan kecepatan tinggi. *Webcam* berperan penting dalam penelitian ini karena dapat memberikan data masukan yang bervariasi dan realistis secara *real-time*. *Webcam* yang digunakan pada penelitian ini

yaitu Logitech C270 HD *Webcam*, dengan spesifikasi yang tertera pada Tabel 3.3.

Tabel 3.3 Spesifikasi *Webcam*

Spesifikasi	Nilai
Resolusi video	HD 720p / 30 FPS
Bidang pandang	55° diagonal
Fokus	*Fixed*
Auto light correction	RightLight 2
Koneksi	USB-A *plug-and-play*
Panjang kabel	1,5 m
Mikrofon	*Built-in, Noise Reducing*

8. *WiFi Adapter*

   WiFi Adapter TP-Link WN725N yang digunakan pada Jetson Nano memungkinkan perangkat ini untuk terhubung ke *hotspot WiFi*, sehingga memberikan kemampuan koneksi nirkabel yang tidak tersedia secara bawaan. Konektivitas ini sangat penting dalam penelitian ini, karena memfasilitasi transfer data yang dikumpulkan dan penerimaan perintah dari laptop secara jarak jauh melalui SSH.

9. *Power Bank*

   Penggunaan *power bank* sebagai sumber daya untuk Jetson Nano memungkinkan penelitian lapangan yang lebih fleksibel, karena tidak tergantung pada sumber listrik tetap. Penggunaan *power bank* harus memenuhi spesifikasi teknis Jetson Nano, termasuk output tegangan yang stabil dan kapasitas arus yang cukup. Merk yang digunakan yaitu Baseus PD 20W *Power Bank* dengan *micro input* tegangan dan arus sebesar 5V/2A dan 9V/2A maksimum.

3.3 Metode Penelitian

Metodologi yang dijalankan dalam penelitian ini bertujuan memberikan penjelasan yang terperinci dan terstruktur tentang tahapan-tahapan yang dilakukan, sebagai berikut.

3.3.1 Studi Literatur

Tujuan dari studi literatur pada penelitian ini adalah untuk mengumpulkan, menelaah, dan menganalisis sumber-sumber pustaka yang relevan dengan topik penelitian yang akan dilakukan. Studi literatur ini difokuskan pada pengujian model YOLO yang diterapkan untuk sistem deteksi objek, khususnya helm pengendara motor, dan juga untuk mengevaluasi kinerja komputasi Jetson Nano pada model-model CNN tertentu.

3.3.2 Akuisisi Data

Dalam tahap pengambilan data ini, pemilihan lokasi, waktu, dan metode rekaman sangat penting. Akuisisi data yang berkualitas adalah langkah esensial dalam pengembangan sistem deteksi helm. Proses ini melibatkan pengumpulan gambar atau video yang relevan dan anotasi yang tepat, yang akan digunakan untuk melatih dan menguji model YOLOv8. Data dikumpulkan secara mandiri dalam bentuk video yang direkam menggunakan *webcam* Logitech C270, yang kemudian diolah menjadi *frame-frame* gambar pada tahap *pre-processing* data. Kelas yang digunakan dalam penelitian ini adalah 'Rider' untuk pengendara motor, 'Helm' untuk pengendara yang memakai helm, dan 'Tidak Helm' untuk pengendara yang tidak memakai helm.

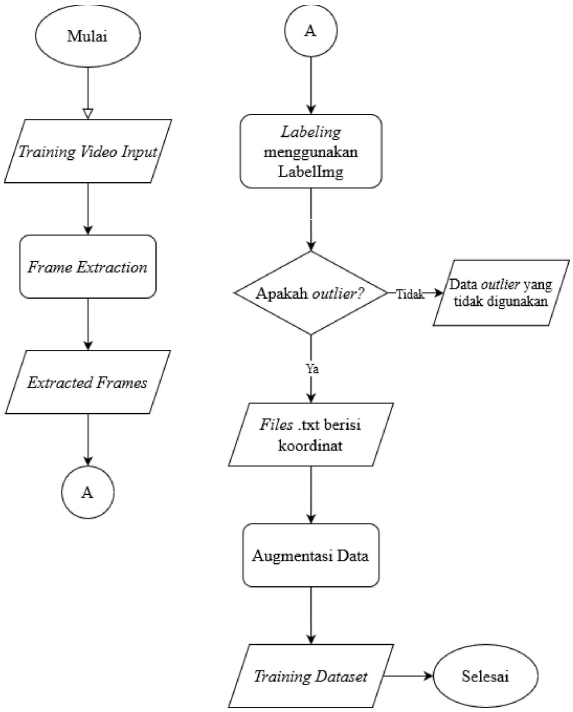
Video untuk kelas 'Helm' dan 'Rider' diambil pada Jalan Raya Cilegon direkam dari Jembatan Penyeberangan Orang (JPO) Cilegon, tempat yang dipilih karena frekuensi tinggi pengendara motor yang menggunakan helm. Sementara itu, data untuk kelas 'Tidak Helm' diambil dari lokasi yang memiliki banyak orang terlibat dalam video, seperti area publik yang ramai, untuk mendapatkan variasi data yang lebih luas, pada penelitian ini diambil pada Jembatan Penyebrangan Orang dan Sepeda (JPOS) Phinisi, dan Taman Literasi Martha Tiahahu.

Dari tahap ini, didapatkan sebanyak 11 video rekaman, di mana 8 di antaranya ditujukan untuk data latih, 1 video untuk data validasi, dan sisanya, yaitu 3 video, digunakan untuk *testing* model dan sistem. Semua video direkam menggunakan *webcam* Logitech C270 yang menangkap gambar pada resolusi 1280x720 dan *frame rate* 30 FPS. Durasi rekaman juga disesuaikan sesuai dengan tujuan masing-masing, dengan data video *training* direkam selama 15-25 menit,

data video validation selama 15 menit, dan data video testing selama 10 menit, memastikan semuanya mencakup skenario lalu lintas yang beragam. Ketiga video pengujian direkam pada waktu yang berbeda — pagi, siang, dan sore — untuk mengevaluasi performa sistem dalam berbagai kondisi pencahayaan, dari terang hingga redup.

3.3.3 *Pre-processing Data*

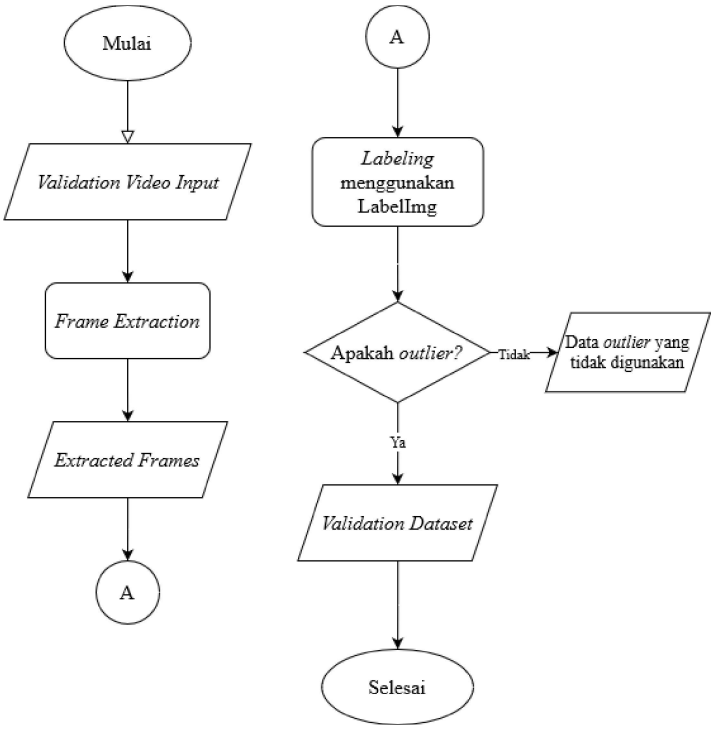
Proses ini bertujuan untuk membuat dataset yang akan digunakan untuk melatih model YOLOv8. Dataset terdiri dari tiga bagian: *training*, validation, dan testing, yang diambil dari video yang telah direkam sebelumnya. Video tersebut kemudian diproses melalui tahap pre-processing menggunakan tiga proses yang berbeda. Metode pre-processing untuk pembuatan dataset *training* dijelaskan secara detail dalam diagram alir pada Gambar 3.2.



Gambar 3.2 Diagram Alir *Pre-processing* Data *Training*

Pada Gambar 3.2, dimana dari tahap ini, *frame* yang berhasil diekstrak disimpan dalam *format file* .jpg. Dari total 8 video yang ditujukan untuk pelatihan, berhasil diekstrak sebanyak 7922 *frames,* yang kemudian dianotasikan dengan

kelas yang sesuai sehingga berkurang menjadi sebanyak 3513 *frames*, yang dimana sisanya merupakan data *frames* yang kurang relevan. Kemudian dari *frames* tersebut diaugmentasikan menggunakan *library albumentations* dengan metode *horizontal flip* sehingga menghasilkan jumlah dua kali lipatnya sebanyak 7026 *frames*. Metode *pre-processing* untuk pembuatan dataset *validation* dijelasakan pada grafik diagram alir pada Gambar 3.3.



Gambar 3.3 Diagram Alir *Pre-processing* Data *Validation*

Pada Gambar 3.3, dimana dari tahap ini, *frame* yang berhasil diekstrak disimpan dalam *format file* .jpg. Dari 1 video yang ditujukan untuk validasi model, berhasil diekstrak sebanyak 641 *frames,* yang kemudian dianotasikan dengan kelas yang sesuai sehingga berkurang menjadi sebanyak 240 *frames*, yang dimana sisanya merupakan data *frames* yang kurang relevan.

Untuk data *testing,* digunakan data berupa video untuk pengujian ini mencakup tiga kondisi pencahayaan yang berbeda, yakni pagi, siang, dan sore. Ketiga video tersebut mencakup detail jumlah aktual objek dan kelas yang terdapat pada Tabel 3.4.

Tabel 3.4 Jumlah Total Aktual Objek Pada Video *Testing*

Waktu	Kelas Rider	Kelas Helm	Kelas Tidak Helm
Pagi	350	443	30
Siang	381	481	19
Sore	610	728	16

Pada Tabel 3.4, terdapat rincian jumlah total objek pada video *testing* untuk ketiga kondisi pencahayaan. Dalam tabel ini, objek kelas dihitung berdasarkan satu entitas yang bergerak dari saat muncul hingga tidak terlihat lagi oleh kamera. Namun, objek pengendara motor yang berlawanan arah dari kamera dan pejalan kaki tanpa helm serta yang bukan pengendara kendaraan motor tidak termasuk sebagai nilai aktual kelas. Pada Tabel 3.4, disajikan jumlah total anotasi untuk setiap kelas berdasarkan *frame* yang telah melewati proses *labeling*.

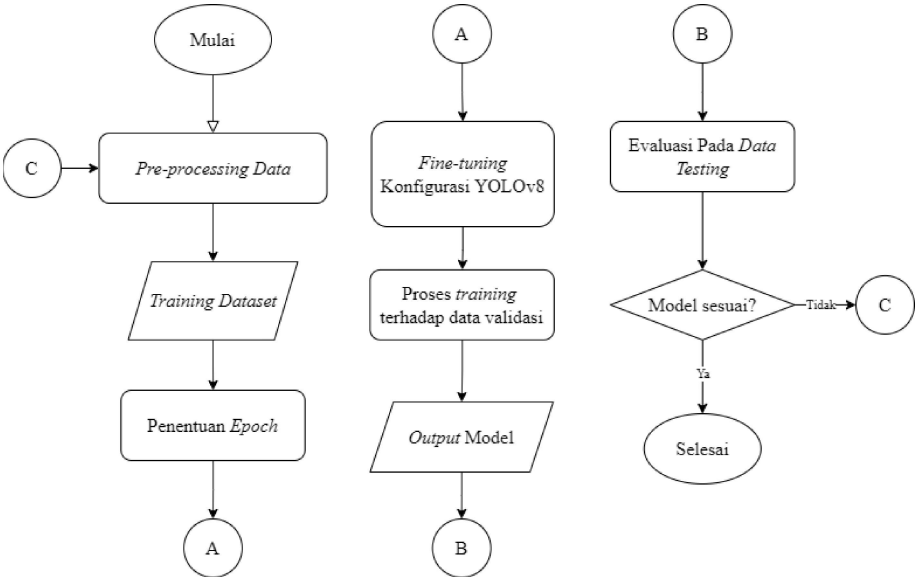
Tabel 3.5 Jumlah Total Anotasi Pada Tiap Kategori Kelas

Kelas	*Training*	*Validasi*
Rider	3836	864
Helm	4782	899
Tidak Helm	4128	117
Total	**12746**	**1880**

Pada Tabel 3.6, disajikan jumlah anotasi yang telah dilakukan pada data pelatihan dan validasi. Terdapat sejumlah 12746 anotasi yang dikumpulkan untuk data pelatihan, yang akan digunakan dalam pengembangan model YOLOv8. Selanjutnya, untuk data validasi, dicatat sebanyak 1880 anotasi, yang berperan dalam evaluasi performa dan keefektifan model.

3.3.4 *Pelatihan Dataset*

Proses pelatihan dataset pada penelitian ini dimulai dengan mempersiapkan dataset yang telah dibuat sebelumnya. Dalam proses pelatihan model deteksi pelanggaran helm pada penelitian ini, digunakan arsitektur model YOLOv8 ukuran paling kecil yaitu YOLOv8n. Berikut diagram alir pelatihan model yang digunakan dalam sistem deteksi helm yang disajikan pada Gambar 3.4.



Gambar 3.4 Diagram Alir Pelatihan Dataset

Pada Gambar 3.4, dimana tahap ini proses pelatihan pada model dilakukan dengan bantuan Google Colab, ClearML dan library PyTorch. Hasil dari pelatihan ini adalah model deteksi yang akan digunakan pada tahap pengujian. Pada tahap pengujian, model akan diuji dengan menggunakan data yang yang belum pernah digunakan sebelumnya yang berasal dari data *testing*. Data yang digunakan pada tahap pengujian harus memiliki karakteristik yang sama dengan data pada tahap pelatihan agar hasil pengujian dapat dianggap *valid*.

Hyperparameter yang menentukan hasil model pelatihan yang digunakan ditampilkan pada Tabel 3.6.

Tabel 3.6 Konfigurasi *Hyperparameter*

Hyperparameter	**Keterangan**
Epochs	50
HSV-Hue augmentation	0.01
HSV-Saturation augmentation	0.5
HSV-Value augmentation	0.0
Translate	0.0
Degrees	0.0
Scale	0.1
Flip Left-Right	0.0

Flip Up-Down	0.0
Mosaic	0.0

Pada Tabel 3.6, merupakan konfigurasi *hyperparameter* terbaik untuk model kustom YOLOv8 setelah dilakukan beberapa kali eksperimen serta *fine-tuning*. Konfigurasi yang tercantum dalam tabel merupakan hasil perubahan yang dilakukan, sementara konfigurasi yang tidak tercantum diberlakukan dengan nilai *default*-nya. Selain itu, memperkecil nilai augmentasi tambahan atau bahkan mengaturnya menjadi nol menghasilkan model yang lebih baik daripada menggunakan nilai augmentasi yang lebih tinggi atau nilai bawaan YOLOv8.

Selanjutnya pada tahap evaluasi dilakukan dengan menggunakan metrik-metrik seperti *confusion matrix, precision, recall, mean Average Precision,* serta *f_1-score*. Penggunaan *confusion matrix* digunakan untuk mengukur kinerja deteksi helm berdasarkan *true positive* (TP), *false positive* (FP), dan *false negative* (FN) untuk pendeteksian multi kelas. TP merupakan jumlah objek yang terdeteksi dengan benar, FP merupakan jumlah objek yang salah terdeteksi, serta FN merupakan jumlah objek yang tidak terdeteksi, penggunaan *true negative* (TN) tidak masuk dalam evaluasi dikarenakan semua titik pada *background* atau objek non-aktual pada pendeteksian objek termasuk TN. Dari confusion matrix, dapat dihitung *precision*, *recall*, serta *f_1-score* dari model deteksi helm. Hasil evaluasi ini dapat membantu dalam menentukan apakah model deteksi yang telah dibangun cukup baik atau perlu dilakukan perbaikan seperti *cross-check* pada dataset serta konfigurasi pada proses pelatihannya. Persamaan (3.1), Persamaan (3.2), Persamaan (3,3), Persamaan (3.4) dan Persamaan (3.5) dapat digunakan sebagai alat untuk menghitung nilai parameter yang terukur.

$$Precision = \frac{TP}{TP+FP} \tag{3.1}$$

$$Recall = \frac{TP}{TP+FN} \tag{3.2}$$

$$f_1 - score = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{3.3}$$

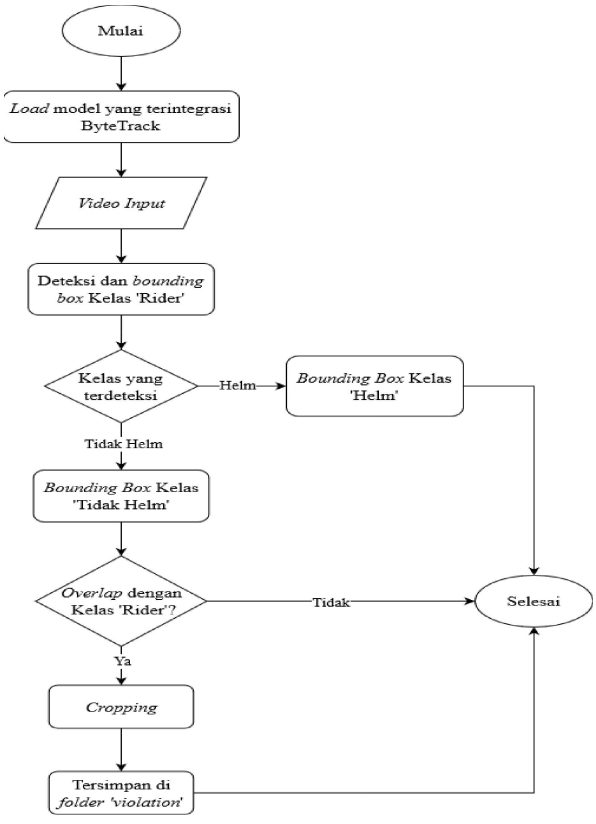
$$AP_i = \int_0^1 Prec(Rec)d(Rec) + C \tag{3.4}$$

$$mAP = \frac{1}{N}\sum_i^N AP_i \tag{3.5}$$

Diketahui bahwa Persamaan (3.1) merupakan rumus untuk *precision*, yang digunakan sebagai parameter untuk mengukur seberapa akurat prediksi positif yang dilakukan oleh model. Persamaan (3.2) menggambarkan rumus untuk *recall*, yang merupakan parameter untuk mengukur seberapa baik model dapat mengidentifikasi semua kasus positif yang sebenarnya. Persamaan (3.3) adalah rumus untuk f_1*-score*, yang merupakan rata-rata harmonik dari *precision* dan *recall*, memberikan keseimbangan antara kedua metrik tersebut. Persamaan (3.4) menyajikan rumus untuk *Average Precision* (AP_i), yang merupakan pengukuran *precision* rata-rata pada semua *threshold recall*. Terakhir, Persamaan (3.5) adalah rumus untuk *mean Average Precision* (mAP) yang menggambarkan rata-rata dari nilai AP (AP_i) untuk semua kelas atau kategori (N) pada kurva dari grafik *precision-recall*.

3.3.5 Perancangan Sistem Deteksi

Proses ini bertujuan untuk membuat sistem deteksi pelanggaran helm berbasis model YOLOv8 yang telah dibuat. Berikut diagram alir perancangan sistem deteksi yang diperlihatkan pada Gambar 3.5



Gambar 3.5 Diagram Alir Sistem Deteksi Helm

Pada Gambar 3.5, merupakan skema sistem ketika dilakukan pengujian secara keseluruhan. Model kustom yang sudah jadi diintegrasikan dengan *ByteTrack* terlebih dahulu. Pengintegrasian ByteTrack pada model YOLOv8 yang telah dilatih meningkatkan efisiensi sistem deteksi helm dengan meminimalkan *cropping* yang diperlukan. Sistem akan mendeteksi pengguna helm dengan *bounding box* berwarna hijau, sedangkan untuk pelanggar ditandai dengan *bounding box* berwarna merah, dan dilakukan *snapshot* berupa data *file* gambar dari pelanggar yang kemudian dimasukkan kedalam *folder* pelanggar.

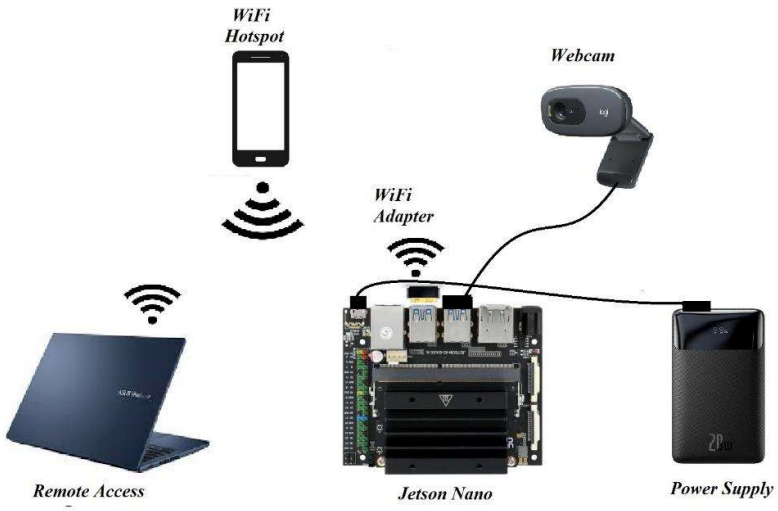
3.3.6 *Deployment* dan Pengujian Sistem

Pada tahap ini, sistem diuji secara langsung di lapangan pada Jalan Raya Cilegon. Sistem yang telah dirancang diimplementasikan pada Jetson Nano, dan akan diuji dalam berbagai kondisi lingkungan. Detail kondisi lingkungan yang akan diuji disajikan dalam Tabel 3.7.

Tabel 3.7 Kondisi Lingkungan Pengujian Sistem

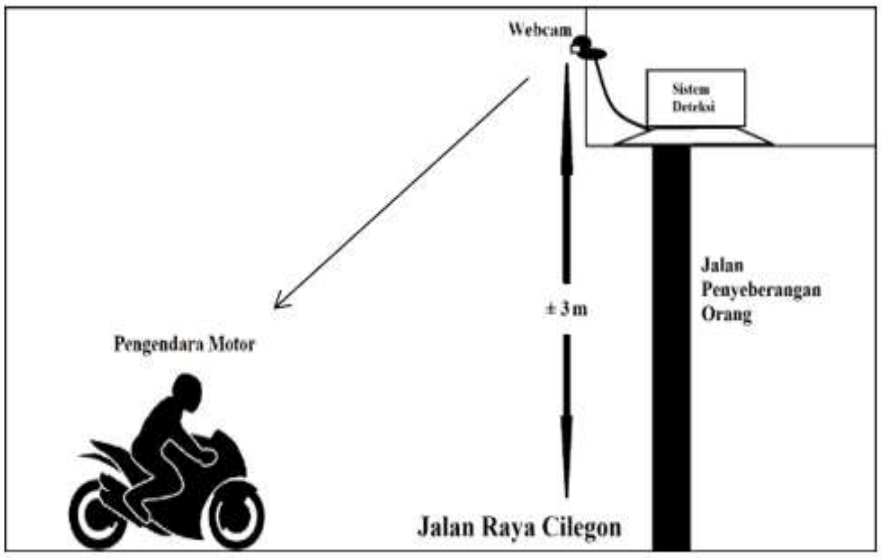
Parameter	Keterangan
Lokasi	Jl. SA. Tirtayasa, Cilegon
Penempatan Sistem	JPO Simpang Tiga
Waktu/Kondisi Pencahayaan	Pagi Hari, Siang Hari, Sore Hari
Jumlah Lajur	1 Lajur
Arah Kamera	Barat

Pada Tabel 3.7, kondisi tersebut dipilih karena arus lalu lintas ke arah timur memiliki volume yang lebih tinggi dibandingkan dengan arus lalu lintas ke arah barat. Selain itu, ini juga mengikuti karakteristik kamera pada sistem E-TLE di Kota Cilegon yang menghadap ke arah barat, terutama di dekat Tugu Kota Cilegon. Hal tersebut juga menjadi salah satu alasan penempatan sistem di Jembatan Penyeberangan Orang (JPO) Simpang Tiga yang berdekatan dengan Tugu Kota Cilegon. Berikut adalah skema rangkaian yang akan diterapkan dalam pengujian real-time, seperti yang ditunjukkan dalam Gambar 3.6.



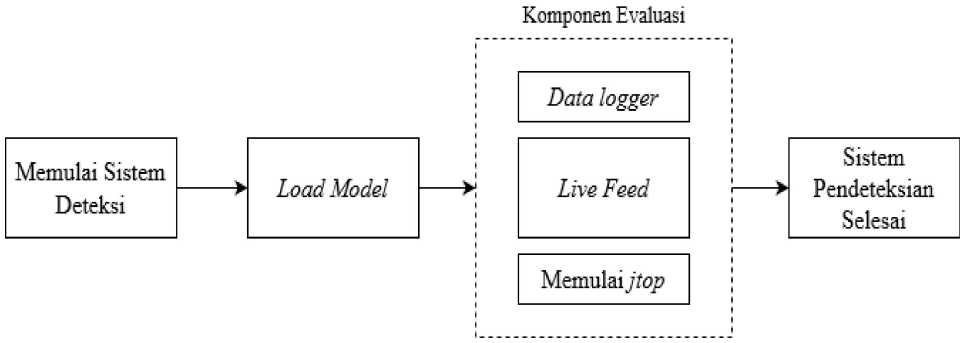
Gambar 3.6 Skema Rangkaian Pengujian Sistem

Pada skema yang diperlihatkan dalam Gambar 3.7, Jetson Nano akan menerima daya dari *power bank* dan terkoneksi ke *hotspot WiFi* yang dihasilkan oleh *smartphone*. Selanjutnya, laptop akan bergabung dengan *hotspot* yang sama. Setelah kedua perangkat terhubung ke jaringan *WiFi*, *Graphical User Interface* (GUI) dari Jetson Nano dapat diakses melalui koneksi desktop jarak jauh atau akses jarak jauh melalui SSH. Dari sini, sistem deteksi *real-time* akan dimulai melalui laptop yang terhubung dengan *webcam*. Berikut skema lingkungan pada saat pengujian secara *real-time* yang disajikan dalam bentuk ilustrasi pada Gambar 3.7.



Gambar 3.7 Skema Lingkungan saat Pengujian

Pada Gambar 3.7, *webcam* dipasang pada penghalang JPO yang menghadap Jalan Raya Cilegon dengan ketinggian sekitar 3 meter. Penempatan ini sesuai dengan mengikuti sudut pandang E-TLE yang memberikan tampilan luas kondisi jalan raya. Berikut blok diagram diagram blok pengujian sistem ketika sistem dioperasikan di lapangan yang disajikan pada Gambar 3.8.



Gambar 3.8 Blok Diagram Pengujian Sistem

Pada Gambar 3.8, untuk menguji kelayakan performa sistem, ketika menjalankan sistem dan memulai live feed, terlebih dahulu menampilkan *jtop* yang memantau statistik perangkat untuk mengevaluasi performa Jetson Nano saat live feed dijalankan. Sistem juga telah terintegrasi dengan *data logger* yang menyimpan data dalam format file .txt untuk evaluasi lanjutan Jetson Nano.

3.3.7 Evaluasi

Pada tahapan ini, dilakukan evaluasi terhadap performa sistem deteksi helm menggunakan model YOLOv8 pada perangkat Jetson Nano. Evaluasi untuk kinerja model pada *live feed* dilakukan dengan menggunakan tiga metrik yaitu *precision*, *recall,* serta *f1-score* berdasarkan persamaan yang sudah tertera pada Persamaan (3.1), Persamaan (3.2), Persamaan (3,3).

Sementara itu, untuk evaluasi performa pada Jetson Nano dilakukan dengan dengan analisis dari hasil *data logger* dan *jtop*. Dari *data logger* didapat proses per-*instances* yang terbagi menjadi dua yaitu kinerja kecepetan inferensi atau pendeteksian per-*instance* serta informasi dari sumber daya komponen dari Jetson Nano secara *real-time*. Untuk kecepatan inferensi, akan dihitung *frames*

per second FPS guna mengukur performa sistem deteksi helm pada perangkat Jetson Nano, mengindikasikan jumlah *frame* per detik yang dapat diproses oleh sistem, di mana semakin tinggi nilai FPS, semakin cepat sistem dalam memproses gambar. Persamaan 3.6 digunakan untuk perhitungan FPS sebagai berikut.

$$FPS = \frac{1}{t_{preprocess} + t_{inference\ time} + t_{postprocess}} \quad (3.6)$$

Komponen perhitungan FPS yang digunakan pada Persamaan (3.6) yaitu merupakan rata-rata kecepatan waktu *pre-processing* pada tiap *frame*, waktu inferensi pada *tiap frame*, dan waktu *post-processing* pada tiap *frame* yang dihasilkan *data logger* pada waktu sistem beroperasi secara *real-time*.

Sedangkan untuk sumber daya komponen pada perangkat Jetson Nano, akan dianalisis kinerja penggunaan keempat *cores* CPU, GPU, RAM, temperature komponen, serta daya total yang digunakan dari Jetson Nano, yang sudah dihasilkan oleh *data logger*.