SISTEM DETEKSI HELM MENGGUNAKAN ALGORITMA YOLOv8 PADA PENGENDARA SEPEDA MOTOR

SKRIPSI

Disusun sebagai salah satu syarat untuk memperoleh Sarjana Teknik (S.T.)

Disusun oleh:

ISMAIL BINTANG

NPM. 3332190037

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS SULTAN AGENG TIRTAYASA

2024

LEMBAR PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya sebagai penulis Skripsi berikut:

Judul	:	Sistem Deteksi Helm Menggunakan Algoritma YOLOv8 Pada Pengendara Sepeda Motor
Nama Mahasiswa	:	Ismail Bintang
NPM	:	3332190037
Fakultas/Jurusan	:	Teknik/Teknik Elektro

Menyatakan dengan sesungguhnya bahwa Skripsi tersebut di atas adalah benar-benar hasil karya asli saya dan tidak memuat hasil karya orang lain, kecuali dinyatakan melalui rujukan yang benar dan dapat dipertanggungjawabkan. Apabila dikemudian hari ditemukan hal-hal yang menunjukkan bahwa sebagian atau seluruh karya ini bukan karya saya, maka saya bersedia dituntut melalui hukum yang berlaku. Saya juga bersedia menanggung segala akibat hukum yang timbul dari pernyataan yang secara sadar dan sengaja saya nyatakan melalui lembar ini.

Cilegon, 12 Juni 2024

MA

Ismail Bintang

NPM. 3332190037

LEMBAR PENGESAHAN

Dengan ini ditetapkan bahwa Skripsi berikut:

Judul	:	Sistem Deteksi Helm Menggunakan Algoritma YOLOv8 Pada Pengendara Sepeda Motor
Nama Mahasiswa	:	Ismail Bintang
NPM	:	3332190037
Fakultas/Jurusan	:	Teknik/Teknik Elektro

Telah diuji dan dipertahankan pada tanggal 12 Juni 2024 melalui Sidang Skripsi di Fakultas Teknik Universitas Sultan Ageng Tirtayasa Cilegon dan dinyatakan LULUS / ~~TIDAK LULUS.~~

		Dewan Penguji	Tanda Tangan
Pembimbing I	:	Imamul Muttakin, S.T., M.Eng. Ph.D	.....
Pembimbing II	:	Fadil Muhammad, S.T., MT	.....
Penguji I	:	Rian Fahrizal, S.T., M.Eng.	.....
Penguji II	:	Ceri Ahendyarti, S.T., M.Eng.	.....

Mengetahui,

Ketua Jurusan

Dr. Eng. Rocky Alfanz, S.T., M.Sc.

NIP. 198103282010121001

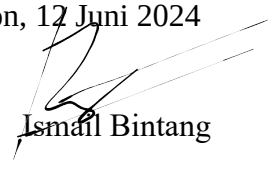
PRAKATA

Puji syukur saya panjatkan kepada Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga laporan skripsi ini dapat saya selesaikan. Laporan ini merupakan bagian dari penugasan akademik dalam menyelesaikan pendidikan S1 di Jurusan Teknik Elektro Universitas Sultan Ageng Tirtayasa. Banyak pihak telah memberikan bantuan, arahan, dan dorongan selama penulisan ini. Oleh sebab itu, saya ucapkan terima kasih kepada.

1. Dr. Eng. Rocky Alfanz, S.T., M.Sc., selaku Ketua Jurusan Teknik Elektro Universitas Sultan Ageng Tirtayasa, yang telah memberikan fasilitas dan dukungan yang diperlukan selama proses penulisan skripsi ini.
2. Imamul Muttakin, S.T., M.Eng. Ph.D dan Fadil Muhammad, S.T., MT sebagai dosen pembimbing skripsi yang telah memberikan bimbingan, arahan, serta kritik konstruktif selama proses penulisan skripsi ini.
3. Rian Fahrizal, S.T., M.Eng. dan Ceri Ahendyarti, S.T., M.Eng. selaku penguji skripsi yang telah menguji dan memberikan masukan yang berharga guna penyempurnaan laporan skripsi ini.
4. Fadil Muhammad, S.T., M.T., sebagai dosen pembimbing akademik yang telah memberikan bimbingan serta arahan selama perjalanan studi saya di Universitas Sultan Ageng Tirtayasa.
5. Keluarga yang senantiasa memberikan doa, dukungan moral, dan motivasi dalam setiap langkah penelitian ini.
6. Rekan-rekan di Kontrakan Jajatgakure yang senantiasa memberikan semangat dan motivasi, serta berkontribusi dalam penyelesaian proyek ini.

Akhir kata, semoga hasil penelitian ini dapat memberikan manfaat serta kontribusi positif bagi pengembangan ilmu pengetahuan.

Cilegon, 12 Juni 2024



Ismail Bintang

ABSTRAK

Ismail Bintang
Teknik Elektro

Sistem Deteksi Helm Menggunakan Algoritma YOLOv8 Pada Pengendara Sepeda Motor

Di Indonesia, angka kecelakaan lalu lintas terus meningkat setiap tahun, terutama yang melibatkan pengendara motor. Mengendarai motor sangat berbahaya jika tidak menggunakan perlengkapan keselamatan yang sesuai, seperti helm. Oleh karena itu, penggunaan helm sangat krusial untuk keselamatan berkendara. Penelitian ini bertujuan untuk mengembangkan teknologi pendeteksi helm pada pengendara motor dengan menggunakan algoritma YOLOv8 berbasis *deep learning*, yang diterapkan pada perangkat tepi seperti Jetson Nano secara *real-time*. Hasil penelitian menunjukkan bahwa algoritma YOLOv8 memiliki kemampuan deteksi helm yang tinggi dalam pengujian langsung, dengan f1-score mencapai 91,1% untuk kelas 'Helm', 81,7% untuk kelas 'Rider', dan 33,0% untuk kelas 'Tidak Helm'. Analisis komputasi menunjukkan penggunaan CPU rata-rata sebesar 78,0%, penggunaan RAM rata-rata sebesar 77,4%, suhu komponen berkisar antara 33°C hingga 65°C, total daya pemakaian rata-rata 6.5 W, penggunaan GPU yang bervariasi (dari 0.1% hingga 99%), serta FPS rata-rata 11.

Kata Kunci: Helm, Deteksi Objek, YOLOv8, Jetson Nano

ABSTRACT

Ismail Bintang
Teknik Elektro

Helmet Detection System For Motorcyclists Using YOLOv8 Algorithm

In Indonesia, the number of traffic accidents continues to increase every year, especially those involving motorcyclists. Riding a motorcycle is very dangerous without the proper safety equipment, such as a helmet. Therefore, helmet use is crucial for riding safety. This research aims to develop helmet detection technology for motorcyclists using the YOLOv8 algorithm based on deep learning, implemented on edge devices like the Jetson Nano in real-time. The results show that the YOLOv8 algorithm has a high helmet detection capability in live testing, with an f1-score of 91.1% for the 'Helmet' class, 81.7% for the 'Rider' class, and 33.0% for the 'No Helmet' class. Computational analysis shows an average CPU usage of 78.0%, average RAM usage of 77.4%, component temperatures ranging from 33°C to 65°C, total average power consumption of 6.5 W, varying GPU usage (from 0.1% to 99%), and an average FPS of 11.

Keywords: Helmet, Object Detection, YOLOv8, Jetson Nano

DAFTAR ISI

HALAMAN JUDUL ..... **i**
ABSTRAK ..... **v**
ABSTRACT ..... **vi**
DAFTAR GAMBAR ..... **ix**
DAFTAR TABEL ..... **x**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan Penelitian 4
1.5 Manfaat Penelitian 4
1.6 Sistematika Penulisan 4
BAB II TINJAUAN PUSTAKA **6**
2.1 Regulasi Pelanggaran Lalu Lintas 6
2.2 *Artificial Intelligence* 6
2.3 *Machine Learning* 7
2.4 Computer Vision 8
2.5 Deteksi Objek 9
2.5.1 *Object Tracking* 10
2.6 *Deep Learning* 11
2.7 YOLO 12
2.7.1 YOLOv8 13
2.8 Teknik *Cropping* untuk Sistem 15
2.9 Metrik Evaluasi dalam *Computer Vision* 15
2.10 Jetson Nano 16
2.11 Kajian Terdahulu 17

BAB III METODOLOGI PENELITIAN 19
3.1 Diagram Metodologi 19
3.2 Komponen Penelitian 20
3.3 Metode Penelitian 23
3.3.1 Studi Literatur 24
3.3.2 Akuisisi Data 24
3.3.3 *Pre-processing Data* 25
3.3.4 *Pelatihan Dataset* 27
3.3.5 Perancangan Sistem Deteksi 30
3.3.6 *Deployment* dan Pengujian Sistem 31
3.3.7 Evaluasi 33
BAB IV ANALISIS DAN PEMBAHASAN 35
4.1 Analisis Model *Training* YOLOv8 35
4.2 Analisis *Testing* Model Pada Video 37
4.3 Analisis Pengujian Sistem *Live Feed* menggunakan Jetson Nano 39
4.3.2 Hasil Analisis Pengujian Sistem pada Pagi Hari 40
4.3.3 Hasil Analisis Pengujian Sistem pada Siang Hari 43
4.3.4 Hasil Analisis Pengujian Sistem pada Sore Hari 46
4.4 Analisa Performa Sistem Deteksi Helm 50
BAB V PENUTUP 55
5.1 Kesimpulan 55
5.2 Saran 55
DAFTAR PUSTAKA A-1
LAMPIRAN A *OUTPUT* GRAFIK MODEL A-1
LAMPIRAN B SAMPEL DATA B-1
LAMPIRAN C KONFIGURASI PENGUJIAN C-1

DAFTAR GAMBAR

Gambar 2.1 Kerangka Umum *Artificial Intelligence*[19] 7
Gambar 2.2 Jenis Sub-bidang *Computer Vision* [23]..... 9
Gambar 2.3 Pembagian gambar menjadi sel-sel grid dan prediksi yang sesuai untuk satu sel grid [31]..... 13
Gambar 2.4 Arsitekur YOLOv8 [34] 14
Gambar 3.1 Diagram Alir Penelitian 19
Gambar 3.2 Diagram Alir *Pre-processing* Data *Training* 25
Gambar 3.3 Diagram Alir *Pre-processing* Data *Validation*..... 26
Gambar 3.4 Diagram Alir Pelatihan Dataset..... 28
Gambar 3.5 Diagram Alir Sistem Deteksi Helm 31
Gambar 3.6 Skema Rangkaian Pengujian Sistem 32
Gambar 3.7 Skema Lingkungan saat Pengujian 33
Gambar 3.8 Blok Diagram Pengujian Sistem 33
Gambar 4.1 *Confusion Matrix* Model *Training* 35
Gambar 4.2 Kurva *Precision-Recall* Model *Training* 37
Gambar 4.3 Tampilan Sistem Deteksi Helm *Real-time* menggunakan Jetson Nano 40
Gambar 4.4 Grafik Penggunaan GPU Pengujian Pagi Hari..... 43
Gambar 4.5 Grafik Penggunaan GPU Pengujian Siang Hari..... 46
Gambar 4.6 Grafik Penggunaan GPU Pengujian Sore Hari 49
Gambar 4.7 Kasus *False Positives* pada Pengujian Langsung 51
Gambar 4.8 Kasus *False Negatives* pada Pengujian Langsung 52
Gambar 4.9 Hasil *Cropping* tanpa Sistem *Tracking* 53
Gambar 4.10 Hasil *Cropping* dengan menggunakan Sistem *Tracking*..... 53

DAFTAR TABEL

Tabel 3.1 Spesifikasi Laptop 21
Tabel 3.2 Spesifikasi Jetson Nano 22
Tabel 3.3 Spesifikasi *Webcam* 23
Tabel 3.4 Jumlah Total Aktual Objek Pada Video *Testing* 26
Tabel 3.5 Jumlah Total Anotasi Pada Tiap Kategori Kelas 27
Tabel 3.6 Konfigurasi *Hyperparameter* 28
Tabel 3.7 Kondisi Lingkungan Pengujian Sistem 31
Tabel 4.1 Hasil Metrik Evaluasi Model *Training* 36
Tabel 4.2 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Pagi Hari 37
Tabel 4.3 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Siang Hari 38
Tabel 4.4 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Siang Hari 39
Tabel 4.5 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Pagi Hari 40
Tabel 4.6 Hasil Kecepatan Pemrosesan Deteksi Per-*frame* Pada Pagi Hari 41
Tabel 4.7 Hasil Performa Jetson Nano pada Pagi Hari 42
Tabel 4.8 Hasil Prediksi Dan Evaluasi Model Pada Pengujian *Real-time* Sistem Siang Hari 44
Tabel 4.9 Hasil Kecepatan Pemrosesan Deteksi Per-*frame* Pada Siang Hari 44
Tabel 4.10 . Hasil Performa Jetson Nano pada Siang Hari 45
Tabel 4.11 Hasil Prediksi Dan Evaluasi Model Pada Pengujian *Real-time* Sistem Sore Hari 47
Tabel 4.12 Hasil Kecepatan Pemrosesan Deteksi Per-*frame* Pada Sore Hari 47
Tabel 4.13 Hasil Performa Jetson Nano pada Sore Hari 48
Tabel 4.14 Hasil Pengujian Secara Keseluruhan 50

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kecelakaan lalu lintas adalah ancaman serius bagi pengguna jalan di seluruh dunia dan berakibat fatal. Di Indonesia, angka kecelakaan terus meningkat. Pada 2021, tercatat lebih dari 103.645 kecelakaan, naik dari 100.028 kasus pada tahun sebelumnya menurut data Kementerian Perhubungan dan Korlantas Polri [1]. Kecelakaan lalu lintas berkontribusi besar sebagai salah satu penyebab kematian di Indonesia dengan tingkat fatalitas yang tinggi. Data menunjukkan sekitar 25 ribu orang meninggal dunia setiap tahunnya akibat kecelakaan lalu lintas [2]. Sepeda motor paling rentan dan sering terlibat kecelakaan di Indonesia, sehingga perlu tindakan untuk mengurangi angka tersebut [3].

Penggunaan helm saat berkendara motor sangat penting untuk keselamatan jalan. Helm melindungi kepala dan dapat mengurangi risiko cedera kepala hingga 69% dan kematian hingga 42% [4]. Di Indonesia, penggunaan helm yang memenuhi standar keamanan nasional telah diwajibkan oleh Undang-Undang Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan [5]. Namun, tantangan tetap ada dalam memastikan pengendara mematuhi standar helm. Oleh karena itu, pengembangan sistem deteksi helm otomatis penting untuk penegakan aturan dan meningkatkan keselamatan [6].

Salah satu cara terbaru untuk meningkatkan kepatuhan pengendara motor dalam menggunakan helm adalah dengan memanfaatkan teknologi, seperti *Electronic Traffic Law Enforcement* (E-TLE) di Indonesia. E-TLE adalah sistem yang memanfaatkan teknologi untuk melakukan pengawasan dan penilangan pelanggar lalu lintas secara otomatis [7]. Kamera *Closed Circuit Television* (CCTV) mengambil gambar pelanggaran, lalu sistem memprosesnya untuk mendeteksi pelanggaran [8]. Keuntungan E-TLE adalah mengurangi beban kerja polisi dan memastikan penegakan hukum yang lebih adil dan objektif [9]. Namun, E-TLE memiliki kelemahan dalam mendeteksi pelanggaran tertentu, seperti penggunaan helm pada pengendara sepeda motor [10], [11]. Oleh karena itu, perlu

evaluasi dan pengembangan teknologi pendukung untuk mengatasi keterbatasan ini.

Dalam E-TLE, teknologi *deep learning* diimplementasikan pada CCTV untuk mendeteksi objek dalam gambar atau video [12]. *Deep learning* adalah cabang dari *machine learning*, menggunakan jaringan saraf tiruan untuk memproses data kompleks [13]. Deteksi objek adalah tugas penting dalam computer vision, dengan aplikasi seperti deteksi wajah, mobil otonom, dan pengawasan [14]. Metode umum deteksi objek termasuk *Region Based Convolutional Neural Networks* (R-CNN), *Faster* R-CNN, *Single-shot Detector* (SSD), dan *You Only Look Once* (YOLO) [15]. Penelitian ini memilih YOLO karena kemampuannya mendeteksi objek secara *real-time* dengan akurasi tinggi, cocok untuk deteksi helm yang memerlukan respons cepat.

Dalam penelitian ini, digunakan algoritma terbaru YOLOv8 yang dikembangkan oleh Ultralytics [16]. Struktur YOLOv8 berbeda dengan versi sebelumnya, yang memungkinkannya untuk bekerja lebih akurat dan memiliki nilai mAP (*mean Average Precision*) yang lebih tinggi saat diuji pada dataset *Common Object* (COCO) [17]. Algoritma ini memiliki arsitektur kompleks dan tidak sepenuhnya transparan, membuatnya sulit dipahami dan direproduksi [17]. Meski begitu, YOLOv8 dapat diaplikasikan untuk berbagai tugas deteksi objek dan segmentasi gambar, serta dilatih pada dataset besar dan dijalankan di berbagai *platform hardware* [17]. Meskipun masih sedikit digunakan, YOLOv8 terus dikembangkan untuk meningkatkan akurasi dan kestabilannya dalam aplikasi praktis [17].

Penelitian ini berharap teknologi seperti algoritma YOLOv8 meningkatkan akurasi, efektivitas, dan efisiensi deteksi helm. Hasil penelitian diharapkan membantu mengembangkan E-TLE untuk mendeteksi helm dengan lebih akurat dan andal, membantu penegak hukum meminimalkan kesalahan tilang. Selain itu, diharapkan bahwa penelitian ini dapat memberikan kontribusi dalam meningkatkan kesadaran masyarakat untuk lebih patuh terhadap aturan berkendara dan pentingnya penggunaan perlengkapan keamanan berkendara yang tepat, khususnya penggunaan helm, guna menciptakan lingkungan berkendara yang lebih aman.

1.2 Rumusan Masalah

Terdapat beberapa permasalahan pada penelitian sistem deteksi helm pada pengendara sepeda motor menggunakan algoritma YOLOv8 berbasis *deep learning* yang diharapkan dapat diselesaikan pada penelitian ini, diantaranya:

1. Bagaimana cara melakukan pendeteksian helm bagi pengendara motor menggunakan berbasis deep learning menggunakan model YOLOv8?
2. Bagaimana kinerja dari model YOLOv8 pada pendeteksian helm bagi pengendara sepeda motor?
3. Bagaimana kinerja dari keseluruhan sistem pendeteksian helm bagi pengendara sepeda motor yang telah dibuat dalam pengujian secara langsung?

1.3 Batasan Masalah

Terdapat beberapa batasan masalah yang sudah ditentukan pada penelitian sistem deteksi helm pada pengendara sepeda motor menggunakan algoritma YOLOv8, diantaranya:

1. Sistem deteksi ini mendeteksi penggunaan helm bagi pengendara kendaraan sepeda motor.
2. Pengujian sistem dilakukan dalam kondisi pencahayaan yang memadai, dengan durasi pengujian selama 10-25 menit pada masing-masing periode waktu berikut: antara jam 7 sampai 9 pagi, jam 1 sampai 3 siang, dan jam 4 sampai 6 sore.
3. Pengujian sistem dilakukan di jalan raya Cilegon, penempatan sistem diletakkan diatas jalan penyeberangan orang.
4. Komponen utama yang digunakan yaitu papan komputasi Jetson Nano untuk sistem deteksi penggunaan helm serta komponen visi yaitu *webcam* Logitech C270.
5. Pengevaluasian dilakukan dengan tolak ukur *precision*, *recall*, f_1*-score*, *mean Average Precision*, FPS, dan penggunaan komponen pada Jetson Nano.

1.4 Tujuan Penelitian

Terdapat beberapa tujuan yang diharapkan tercapai pada penelitian sistem deteksi helm pada pengendara sepeda motor menggunakan algoritma YOLOv8 berbasis *deep learning*, diantaranya:

1. Mengembangkan sistem deteksi helm pada pengendara sepeda motor yang otomatis dan *real-time* dengan basis *deep learning* menggunakan algoritma YOLOv8.
2. Mengembangkan sistem deteksi helm pada pengendara sepeda motor yang akurat dengan basis *deep learning* menggunakan algoritma YOLOv8.
3. Menerapkan sistem deteksi helm pada pengendara sepeda motor dengan menggunakan algoritma YOLOv8 berbasis *deep learning* pada perangkat Jetson Nano.

1.5 Manfaat Penelitian

Terdapat beberapa manfaat yang dapat diperoleh pada penelitian sistem deteksi helm pada pengendara sepeda motor menggunakan algoritma YOLOv8 berbasis deep learning, diantaranya:

1. Membantu pengembangan teknologi deteksi helm pada pengendara sepeda motor yang dapat beroperasi secara otomatis dan real-time, sehingga dapat meningkatkan keamanan berkendara di jalan raya.
2. Menjadi referensi bagi penelitian-penelitian selanjutnya yang berkaitan dengan pengembangan teknologi deteksi helm pada pengendara sepeda motor berbasis deep learning, terutama dengan menggunakan algoritma YOLOv8 yang masih relatif baru.
3. Meningkatkan penggunaan Jetson Nano pada sistem deteksi objek berbasis *deep learning* untuk memperluas penggunaannya di berbagai aplikasi teknologi lainnya.

1.6 Sistematika Penulisan

Struktur keseluruhan skripsi ini disusun dalam lima bab. Setiap babnya memiliki fokus khusus yang akan dijelaskan sebagai berikut.

BAB I

Bab ini menjelaskan latar belakang masalah yang mendasari penelitian, disertai dengan tujuan penelitian, manfaat penelitian, rumusan masalah, batasan penelitian, serta sistematika penulisan.

BAB II

Bab ini mencakup tinjauan terhadap penelitian terdahulu yang relevan dengan topik penelitian yang sedang dilaksanakan. Selain itu, bab ini juga menguraikan dasar-dasar teoritis mengenai sepeda motor, helm, *machine learning*, *deep learning*, *computer vision*, pendeteksian objek, *object tracking*, YOLO, *edge computing*, dan Jetson Nano.

BAB III

Bab ini mencakup penjabaran mengenai metode yang digunakan, alur penelitian, komponen penelitian, pengolahan data yang akan digunakan, perancangan sistem, *deployment* serta pengujian sistem.

BAB IV

Bab ini memuat hasil-hasil dari penelitian yang telah dilakukan beserta analisisnya, yang dikaji berdasarkan pada parameter penelitian dan batasan yang telah ditetapkan sebelumnya.

BAB V

Berisi kesimpulan yang diambil dari hasil penelitian beserta saran yang dapat menjadi arahan untuk penelitian berikutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Regulasi Pelanggaran Lalu Lintas

Di Indonesia, regulasi mengenai pelanggaran lalu lintas ditetapkan melalui Undang-Undang Republik Indonesia Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan. UU ini mencakup berbagai elemen, termasuk pentingnya lalu lintas dan angkutan jalan, pengembangan potensinya, serta peranannya dalam memastikan keamanan, keselamatan, ketertiban, dan kelancaran lalu lintas, serta akuntabilitas dalam penyelenggaraan negara [5].

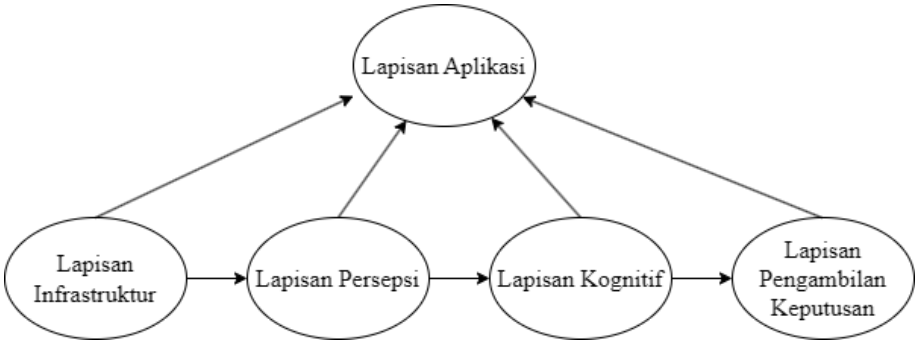
Salah satu poin penting dalam regulasi ini adalah kewajiban penggunaan helm bagi pengendara dan penumpang sepeda motor. Helm berfungsi sebagai perlengkapan keselamatan yang dirancang untuk melindungi kepala dari cedera serius saat terjadi kecelakaan. Aturan ini diatur secara spesifik dalam Pasal 57 ayat (1) dan ayat (2) yang menyatakan bahwa pengendara dan penumpang sepeda motor wajib menggunakan helm yang memenuhi standar nasional Indonesia [5].

Meskipun aturan ini telah diberlakukan sejak lama, tingkat kepatuhan pengendara masih beragam. Banyak pengendara belum menyadari pentingnya penggunaan helm, terutama di daerah-daerah terpencil. Kurangnya pengawasan dan penegakan hukum yang konsisten turut memperburuk situasi ini. Untuk itu, pemerintah Indonesia terus mencari cara untuk meningkatkan kesadaran dan kepatuhan terhadap aturan ini, salah satunya melalui penerapan teknologi dalam sistem pengawasan dan penegakan hukum lalu lintas, yaitu *Electronic Traffic Law Enforcement* (E-TLE). Dalam E-TLE, teknologi *Artificial Intelligence* diimplementasikan pada CCTV untuk mendeteksi objek dalam gambar atau video. E-TLE bekerja dengan menggunakan kamera pengawas yang ditempatkan di lokasi-lokasi strategis untuk memantau lalu lintas dan merekam pelanggaran, seperti melanggar lampu merah, melanggar garis berhenti, dan tidak menggunakan helm [8].

2.2 *Artificial Intelligence*

Artificial Intelligence (AI) atau kecerdasan buatan merujuk pada simulasi kecerdasan manusia dalam sistem berbasis digital, di mana tugas yang

memerlukan kecerdasan manusia dapat dilakukan tanpa masukan kecerdasan manusia [18]. Tujuannya adalah mengembangkan teknologi yang mampu memikirkan, merasakan, dan bertindak seperti manusia, termasuk dalam aspek-aspek seperti persepsi, penalaran, pembelajaran, perencanaan, prediksi, dan lain-lain. Berikut kerangka umum kecerdasan buatan yang disajikan pada Gambar 2.1.



Gambar 2.1 Kerangka Umum *Artificial Intelligence*[19]

Pada Gambar 2.1, kerangka pengembangan AI terdiri dari beberapa lapisan. Lapisan infrastruktur mencakup data, penyimpanan, daya komputasi, algoritma, dan *framework* AI yang mendukung kemampuan persepsi dan kognitif. Lapisan persepsi memberikan kemampuan dasar dalam penglihatan dan pendengaran, memungkinkan mesin untuk "melihat" dan "mendengar". Lapisan kognitif menyediakan kemampuan induksi, penalaran, dan akuisisi pengetahuan. Lapisan pengambilan keputusan memungkinkan mesin membuat keputusan optimal, seperti perencanaan otomatis dan sistem pendukung keputusan. Kemampuan ini mendukung aplikasi AI dalam berbagai bidang seperti ilmu pengetahuan, manufaktur, kehidupan, tata kelola sosial, dan dunia maya, yang mempengaruhi pekerjaan dan gaya hidup [19].

2.3 *Machine Learning*

Machine Learning adalah sebuah cabang ilmu komputer yang bertujuan untuk mengembangkan algoritma atau model yang mampu belajar dari data, mengidentifikasi pola, dan membuat prediksi atau keputusan tanpa adanya pemrograman eksplisit [20]. *Machine Learning* bertujuan untuk memungkinkan mesin untuk belajar secara mandiri, mengidentifikasi pola dan membuat prediksi yang akurat dan dapat diandalkan.

Machine learning menggunakan pendekatan matematis dan statistik untuk mengidentifikasi pola dan relasi yang tidak terlihat secara langsung dalam data, juga untuk meningkatkan efisiensi dan ketepatan model pembelajaran. Beragam masalah dapat diselesaikan dengan menggunakan teknik pembelajaran mesin, termasuk klasifikasi, regresi, pengelompokan, rekomendasi, pengenalan suara, pengenalan gambar, analisis bahasa alami, dan lain-lain [20].

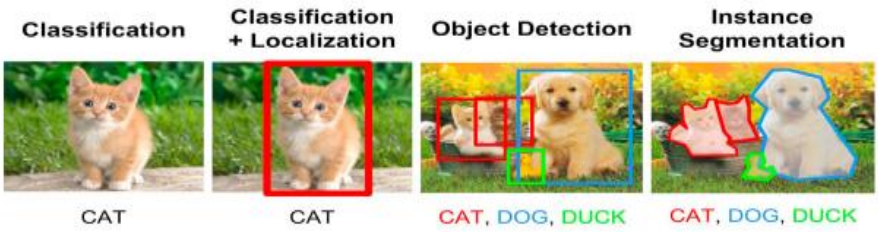
Machine learning dapat dibagi menjadi tiga jenis utama berdasarkan cara pembelajarannya, yaitu:

1. *Supervised learning*, yaitu pembelajaran dengan menggunakan data yang telah diberi label atau kelas. Tujuannya adalah untuk membuat model yang dapat memprediksi label atau kelas dari data baru yang belum diketahui sebelumnya. Contohnya adalah klasifikasi email menjadi spam atau tidak spam, klasifikasi objek pada gambar menjadi kategori tertentu, regresi harga rumah berdasarkan fitur-fiturnya, dan lain-lain [20].
2. *Unsupervised learning*, yaitu pembelajaran dengan menggunakan data yang tidak diberi label atau kelas. Tujuannya adalah untuk menemukan struktur atau kelompok dalam data tanpa adanya informasi sebelumnya. Contohnya adalah klastering pelanggan berdasarkan preferensi atau perilaku mereka, reduksi dimensi data untuk mempermudah visualisasi atau analisis, deteksi anomali atau outlier dalam data, dan lain-lain [20].
3. *Reinforcement learning*: yaitu pembelajaran dengan menggunakan umpan balik dari lingkungan sebagai penguat atau hukuman. Tujuannya adalah untuk membuat model yang dapat belajar dari pengalaman dan meningkatkan performa dalam mencapai tujuan. Contohnya adalah pelatihan robot untuk berjalan atau berlari, pelatihan agen untuk bermain game atau catur, pelatihan mobil otonom untuk mengemudi dengan aman, dan lain-lain [20].

2.4 Computer Vision

Computer vision adalah bidang ilmu yang berkembang pesat sejak beberapa dekade terakhir. Tujuan utama dari computer vision adalah untuk membuat komputer dan sistem mampu melihat, mengamati, dan memahami dunia

visual dengan cara yang mirip dengan manusia [21]. *Computer vision* awalnya berurusan dengan konsep-konsep seperti meniru sistem visual manusia melalui berbagai pemahaman tentang bagaimana skema kamera, proyeksi, dan fotogrametri bekerja. Dengan demikian, computer vision dapat dianggap sebagai cabang dari pengolahan citra, yang berfokus pada manipulasi dan analisis gambar digital [22]. Berikut contoh jenis sub-bidang pada *computer vision* yang diilustrasikan pada Gambar 2.2.



Gambar 2.2 Jenis Sub-bidang *Computer Vision* [23]

Seperti pada Gambar 2.2, *Computer vision* mencakup berbagai sub-bidang dan aplikasi yang berkaitan dengan dunia visual. Beberapa sub-bidang computer vision adalah rekonstruksi adegan, deteksi objek, pelacakan video, pengenalan objek, segmentasi objek, estimasi pose 3D, pembelajaran, pengindeksan, estimasi gerak, servo visual, pemodelan adegan 3D, dan restorasi gambar [23]. Beberapa aplikasi computer vision adalah deteksi wajah, pengenalan karakter optik, analisis medis, navigasi robot, pengawasan keamanan, dan sistem kendaraan pintar [22]. Pada penelitian ini akan berkaitan dengan penggunaan metode deteksi objek untuk perancangan sistem deteksi helm.

2.5 Deteksi Objek

Objek ditentukan oleh fitur seperti bentuk, ukuran, warna, tekstur, dan atribut lainnya. Deteksi objek tidak hanya menunjukkan keberadaan benda tetapi juga lokasinya dalam gambar. Proses ini melibatkan menemukan dan mengidentifikasi objek dunia nyata dalam gambar, serta menentukan posisinya menggunakan *bounding box*. Deteksi objek berkaitan erat dengan klasifikasi objek, segmentasi semantik, dan segmentasi instan[24][25].

Penerapan deteksi objek dapat dibagi menjadi berikut.

1. *Dedicated Object Detection*

Penerapan ini merujuk pada sistem deteksi objek yang dirancang untuk objek atau skenario tertentu, seperti deteksi pejalan kaki, deteksi kendaraan, atau mendeteksi sel kanker dalam gambar medis. Ini biasanya dilakukan dengan menggunakan model yang telah dioptimalkan atau dilatih khusus untuk tujuan tersebut, sehingga dapat memberikan hasil yang sangat spesifik dan optimal untuk jenis objek atau situasi yang ditargetkan [25].

2. *Generic Object Detection*

Pada penerapan deteksi objek ini, bertujuan untuk menemukan contoh objek dunia nyata dalam gambar tanpa terbatas pada kelas objek tertentu. Ini mencakup deteksi objek yang lebih umum seperti kendaraan, manusia, bangunan, dan lainnya. Dalam deteksi objek generik, model mungkin lebih umum dan tidak dioptimalkan secara khusus untuk satu jenis objek atau situasi tertentu, tetapi mampu mengenali berbagai jenis objek dengan tingkat akurasi yang cukup baik [25].

Dalam bidang deteksi objek, terdapat dua pendekatan utama yaitu sebagai berikut.

1. Metode Tradisional

Metode ini bergantung pada fitur yang dibuat secara manual dan algoritma untuk mendeteksi objek. Metode ini melibatkan langkah-langkah seperti pemilihan wilayah informatif, ekstraksi fitur, dan klasifikasi. Metode tradisional menggunakan teknik seperti deteksi tepi dan analisis tekstur untuk mengidentifikasi objek dalam gambar [26].

2. Metode *Deep Learning*

Metode ini menggunakan jaringan saraf untuk mempelajari fitur secara langsung dari data. Metode *deep learning* secara otomatis mengekstrak fitur hierarkis yang lebih tahan terhadap variasi dalam penampilan objek, skala, dan pose. Biasanya, mereka melibatkan pelatihan model pada dataset besar untuk mengenali pola dan membuat prediksi tentang lokasi dan kelas objek [26].

2.5.1 *Object Tracking*

Deteksi objek mengidentifikasi objek target dalam satu gambar, sementara pelacakan objek memprediksi posisi objek sepanjang rangkaian video.

Kemampuan ini penting untuk tugas visi komputer seperti pengawasan dan navigasi otonom. Pelacakan objek biasanya terjadi dalam dua konteks yang berbeda. Pelacakan objek tunggal (*Single Object Tracking*) berfokus pada satu target sepanjang video, contohnya adalah model yang berbasis *Siamese-Network* dan pelacak berbasis *Correlation Filter* [27]. Sedangkan pelacakan objek ganda (*Multiple Object Tracking*) atau pelacakan target ganda (*Multiple Target Tracking*) melibatkan pelacakan banyak objek secara bersamaan, model yang biasa digunakan misalnya SORT, DeepSORT, dan ByteTrack [28].

Dalam penelitian ini, digunakan pelacakan objek untuk proses *cropping* agar tidak terjadi pengulangan gambar yang di-*crop* pada setiap inferensi. Model ByteTrack digunakan karena ByteTrack mempertahankan kecepatan pemrosesan yang tinggi, melampaui SORT dan DeepSORT dalam hal efisiensi komputasi [28].

2.6 *Deep Learning*

Deep learning adalah cabang dari machine learning yang menggunakan jaringan saraf tiruan untuk menangani masalah kompleks dan membutuhkan banyak data. Jaringan ini memiliki banyak lapisan dan bobot sinapsis, yang memungkinkan menemukan pola atau hubungan tersembunyi antara *input* dan *output*. Hal ini membuat deep learning berbeda dari multilayer perceptron yang hanya memiliki tiga lapisan [29]. Dengan menggunakan banyak lapisan, *deep learning* meniru cara berpikir manusia dan dapat mengekstrak fitur tingkat tinggi dari data mentah seperti gambar, teks, suara, atau angka. Fitur-fitur ini kemudian digunakan untuk berbagai tugas seperti klasifikasi, regresi, klastering, dan generasi [30].

Pada kasus deteksi objek, penggunaan *deep learning* pada deteksi objek terbagi menjadi dua jenis sebagai berikut.

1. *One-stage Detector*

Pada jenis ini, tidak ada tugas perantara yang dilakukan dalam detektor satu tahap untuk menghasilkan output akhir. Hal ini membuat arsitekturnya lebih cepat dan sederhana. Sebagai contoh, YOLO dan SSD adalah detektor satu tahap

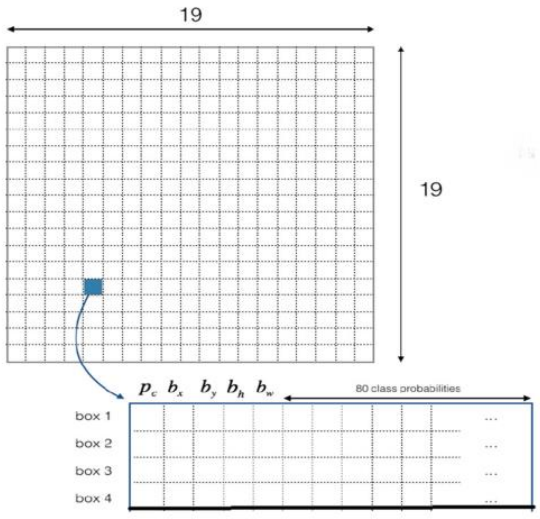
yang digunakan untuk menetapkan kotak pembatas pada posisi tertentu. Jadi, detektor ini melakukan pembelajaran pada lokasi objek tertentu [26].

2. *Two-stage Detector*

Metode ini menggunakan dua tahap untuk mendeteksi objek: pertama, menghasilkan kandidat area objek, kemudian membuat prediksi berdasarkan area tersebut. Di tahap pertama, detektor mengidentifikasi area semua objek. Detektor gambar menghasilkan area dengan tingkat *recall* tinggi, dan objek berada di setidaknya satu area ini. Pada tahap kedua, model *deep learning* melakukan klasifikasi. Area yang dihasilkan bisa berisi objek dengan label tertentu atau latar belakang. Model pada tahap pertama juga dapat memperbaiki lokalisasi[26].

2.7 YOLO

YOLO (*You Only Look Once*) merupakan suatu model deteksi objek yang memanfaatkan metode *one-stage* atau satu jaringan saraf konvolusi untuk memprediksi kotak pembatas dan kelas objek dalam sebuah gambar. YOLO hanya melihat gambar sekali (*You Only Look Once*), sehingga dapat melakukan deteksi objek secara *real-time*. Berbeda dengan model deteksi objek lainnya yang menggunakan dua jaringan saraf terpisah, YOLO dapat memperoleh hasil yang lebih cepat dan akurat karena memiliki arsitektur yang lebih efisien dan sederhana [31]. Untuk mendeteksi gambar dengan lebih cepat dan efisien, dilakukan pembagian grid gambar untuk melakukan deteksi seperti contoh pada Gambar 2.3.



Gambar 2.3 Pembagian gambar menjadi sel-sel grid dan prediksi yang sesuai untuk satu sel grid [31]

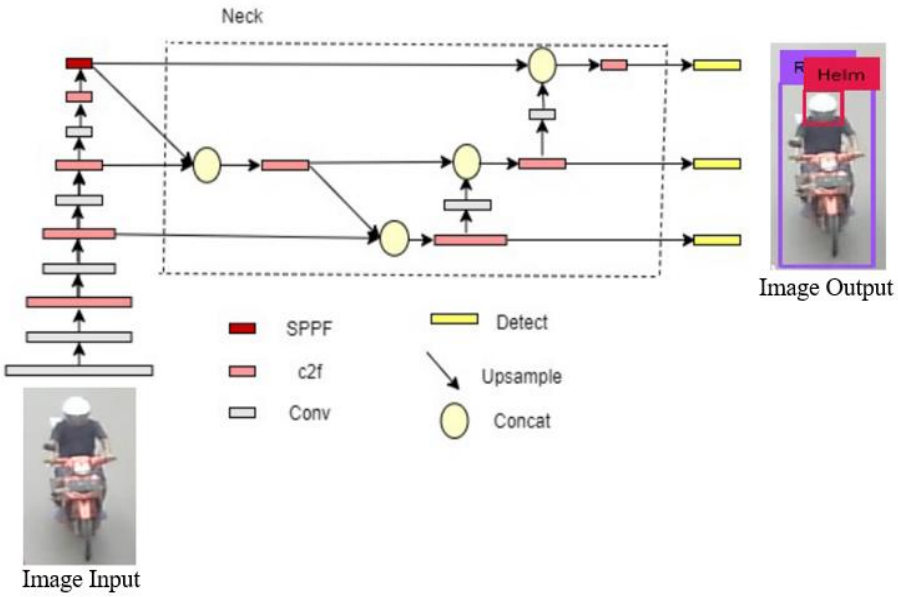
Pada Gambar 3.2, proses deteksi objek menggunakan YOLO membagi gambar menjadi *grid* berukuran $S \times S$, di mana setiap sel *grid* memprediksi B kotak pembatas, posisi dan dimensinya, probabilitas adanya objek, dan probabilitas kelasnya. Pusat objek harus berada dalam sel *grid* agar dapat terdeteksi. Setiap kotak pembatas diprediksi dengan parameter probabilitas objek (p_c), koordinat pusat kotak (b_x, b_y), dimensi kotak (b_h, b_w), dan probabilitas kelas (p_{ci}). Setiap sel *grid* memprediksi ($B \times 5 + n$) nilai, di mana B adalah jumlah kotak pembatas per sel dan n adalah jumlah kelas. Output tensor memiliki bentuk $S \times S \times (B \times 5 + n)$. Skor kepercayaan dihitung untuk setiap kotak pembatas dengan mengalikan p_c dengan *Intersection over Union* (IoU) antara kotak prediksi dan kotak asli. Jika tidak ada objek di sel *grid*, skor kepercayaannya adalah nol [31].

YOLO memiliki beberapa versi, seperti YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, dan YOLOv8. Setiap versi memiliki perbedaan dalam arsitektur, fitur, dan kinerja [32]. Pada penelitian kali ini, model YOLOv8 akan digunakan sebagai basis algoritma untuk sistem deteksi helm. Alasan pemilihan model ini akan dijelaskan lebih lanjut pada bagian teori yang menjelaskan tentang YOLOv8.

2.7.1 YOLOv8

YOLOv8 adalah algoritma deteksi objek dan segmentasi gambar terbaru dari Ultralytics, dirilis pada Januari 2023. Keunggulannya termasuk akurasi yang lebih baik dibandingkan model YOLO sebelumnya dan mendukung berbagai tugas seperti deteksi objek, instance segmentation, dan klasifikasi gambar. YOLOv8 fokus pada peningkatan akurasi dan efisiensi dengan arsitektur jaringan yang dioptimalkan, implementasi anchor box yang didesain ulang, dan fungsi loss yang dimodifikasi, menjadikannya lebih presisi dan fleksibel untuk berbagai aplikasi [33]. YOLOv8 menawarkan lima model berbeda yaitu N (*Nano*), S (*Small*), M (*Medium*), L *(Large*), dan X (*Extra Large*), masing-masing dengan kedalaman saluran dan jumlah filter yang bervariasi. Untuk arsitektur *backbone*

pada penelitian ini, dipilih versi *nano* dari YOLOv8 karena lebih ringan dibandingkan dengan versi lainnya [32]. Berikut arsitektur YOLOv8 yang digunakan pada penelitian kali ini, yang disajikan pada Gambar 2.4.



Gambar 2.4 Arsitekur YOLOv8 [34]

Pada Gambar 2.4 menjelaskan bahwa YOLOv8 memiliki struktur dasar yang terdiri dari beberapa lapisan, seperti lapisan konvolusi, lapisan C2F (*Cross Stage Partial Networks with Fusion*), dan lapisan SPPF (*Spatial Pyramid Pooling – Fast*). Lapisan C2F membantu mempercepat proses belajar dan konvergensi jaringan. Lapisan SPPF memungkinkan model mendeteksi objek dalam berbagai ukuran. Bagian leher (*neck*) menggabungkan dua jenis jaringan, yaitu FPN (*Feature Pyramid Network*) dan PAN (*Path Aggregation Network*), untuk membantu mengelola informasi dengan lebih baik. Fitur dari lapisan-lapisan yang berdekatan digabungkan dalam modul C2F. Jaringan ini menggabungkan fitur-fitur tingkat tinggi dengan fitur-fitur dasar saat melalui struktur tersebut. Bagian *output* memisahkan proses deteksi dan klasifikasi, menggunakan fitur tingkat bawah untuk mendeteksi objek kecil dan fitur tingkat atas untuk objek besar, dengan setiap lapisan deteksi menghasilkan informasi tentang lokasi dan kelas objek [34].

2.8 Teknik *Cropping* untuk Sistem

Teknik *cropping* pada sistem ini adalah komponen kunci dalam deteksi helm untuk memeriksa apakah dua objek tumpang tindih dalam suatu gambar. Dalam konteks ini, *overlap detection* digunakan untuk mendeteksi apakah ada pengendara yang terlihat tidak mengenakan helm. Deteksi ini dilakukan dengan memeriksa apakah *bounding box* 'Rider' tumpang tindih dengan *bounding box* 'Tidak Helm'. Dua *bounding box* dianggap tumpang tindih jika koordinat horizontal dan vertikalnya saling bertemu dalam area tertentu. Koordinat ini didapat dari *library* Ultralytics untuk YOLOv8 itu sendiri [35]. Secara matematis, kondisi tumpang tindih antara dua *bounding box* dapat dinyatakan pada Persamaan 2.1 dan Persamaan 2.2 sebagai berikut.

$$\max(x1_1, x2_2) < \min(x2_1, x2_2) \quad (2.1)$$

$$\max(y1_1, y2_2) < \min(y2_1, y2_2) \quad (2.2)$$

Fungsi ini, yang dijelaskan dalam Persamaan (2.1) dan Persamaan (2.2), memanfaatkan prinsip matematika untuk mengevaluasi apakah dua *bounding boxes* memiliki area yang tumpang tindih atau *overlap*. Persamaan (2.1) menguji posisi horizontal kedua kotak dengan membandingkan sisi kiri kotak yang posisinya lebih ke kanan (nilai x maksimum) terhadap sisi kanan kotak yang posisinya lebih ke kiri (nilai x minimum). Sementara itu, Persamaan (2.2) menilai posisi vertikal dengan membandingkan bagian atas kotak yang lebih rendah (nilai y maksimum) terhadap bagian bawah kotak yang lebih tinggi (nilai y minimum). Jika kedua kondisi ini dipenuhi, maka dapat disimpulkan bahwa kedua kotak tersebut memiliki irisan.

2.9 Metrik Evaluasi dalam *Computer Vision*

Dalam visi komputer, terdapat beberapa metrik evaluasi yang penting untuk mengukur kinerja dari suatu sistem atau model [23]. Berikut adalah beberapa metrik evaluasi yang digunakan pada penelitian.

1. Presisi (*Precision*): Presisi dalam deteksi objek mengukur kemampuan sistem untuk secara akurat mengidentifikasi objek yang terdeteksi. Dalam deteksi helm, tidak helm, dan pengendara, presisi tinggi menunjukkan

sedikitnya *false positive*, atau objek yang salah terdeteksi sebagai objek aktual.

2. Sensitivitas (*Recall*): Sensitivitas dalam deteksi objek mengukur seberapa baik sistem mendeteksi semua objek yang ada. Pada deteksi helm, tidak helm, dan pengendara, sensitivitas tinggi menunjukkan sistem dapat mendeteksi sebagian besar objek, mengurangi *false negative*.
3. F_1*-Score*: F_1*-Score* adalah rata-rata harmonis dari presisi dan sensitivitas. Dalam deteksi helm, tidak helm, dan pengendara, F_1*-Score* menunjukkan keseimbangan antara akurasi dan kemampuan deteksi. F_1*-Score* yang tinggi menandakan sistem memiliki presisi dan sensitivitas yang baik.
4. *Mean Average Precision* (mAP): mAP adalah nilai rata-rata presisi pada berbagai tingkat sensitivitas, memberikan gambaran kemampuan sistem mendeteksi objek dengan presisi yang konsisten. IoU mengevaluasi sejauh mana prediksi *bounding box* bertumpang tindih dengan objek sebenarnya, menjadi faktor kunci dalam perhitungan mAP. mAP50 dihitung dengan threshold IoU 0.5, sedangkan mAP50-95 memperhitungkan rentang IoU dari 0.5 hingga 0.95, memberikan gambaran lengkap tentang kualitas deteksi sistem pada berbagai tingkat kesulitan.

2.10 Jetson Nano

Jetson Nano adalah sebuah komputer *embedded* yang dikembangkan oleh NVIDIA. Jetson Nano dirancang untuk memberikan kemampuan komputasi yang tinggi dan efisien pada perangkat *edge* [36]. Jetson Nano menggunakan prosesor NVIDIA Maxwell dengan arsitektur GPU CUDA (*Compute Unified Device Architecture*), yang memungkinkan operasi komputasi paralel yang efisien. GPU dengan 128 inti CUDA dan CPU quad-core ARM Cortex-A57 memberikan daya komputasi yang handal [37]. Sistem operasi Jetson Nano adalah JetPak, yang merupakan versi khusus Linux yang telah dioptimalkan untuk komputasi AI. JetPak menyediakan lingkungan pengembangan yang lengkap dengan *library* dan *framework* AI yang sering digunakan seperti TensorFlow, PyTorch, dan Caffe [38].

Jetson Nano dapat digunakan dalam berbagai aplikasi AI, termasuk pengenalan objek, deteksi wajah, analisis citra, dan sebagainya secara *real-time*. Platform ini juga dilengkapi dengan berbagai antarmuka I/O seperti USB, HDMI, Ethernet, dan GPIO, yang memungkinkan integrasi dengan perangkat dan sensor eksternal [36]. Dengan performa tinggi dan fleksibilitas dalam integrasi perangkat, Jetson Nano dapat digunakan dalam berbagai proyek AI di perangkat edge seperti kendaraan otonom, robotika, sistem keamanan, dan IoT. Platform ini memberikan solusi yang efisien untuk pengembangan dan implementasi aplikasi AI yang membutuhkan daya komputasi tinggi dalam lingkungan *edge* [39].

2.11 Kajian Terdahulu

Dalam bagian ini dijelaskan beberapa penelitian yang terkait dengan topik penelitian tentang deteksi pelanggaran penggunaan helm pada pengendara motor. Beberapa studi ini melibatkan penggunaan model CNN seperti YOLO dan platform Jetson Nano untuk pengolahan data. Studi-studi ini dijadikan referensi dan landasan bagi penelitian yang sedang berlangsung.

Pada penelitian sebelumnya telah menjelaskan teknik-teknik yang digunakan untuk mengembangkan sistem deteksi pelanggaran penggunaan helm pada sepeda motor dengan menggunakan YOLO untuk mendeteksi sepeda motor, GoogleNet untuk mengklasifikasikan pelanggaran helm, dan *Kristan's method* untuk melacak objek [40]. Selain itu, penelitian ini juga mempersembahkan desain arsitektur sistem CPU-GPU berbiaya rendah namun berkinerja tinggi yang mampu memproses beberapa aliran kamera secara bersamaan. Hasil penelitian menunjukkan bahwa sistem CPU-GPU berbiaya rendah yang diusulkan mampu mendeteksi 97% pelanggaran penggunaan helm dengan tingkat kesalahan alarm sebesar 15%. Selain itu, model YOLO juga terbukti lebih efektif dalam mendeteksi objek pada sistem pelanggaran penggunaan helm pengendara motor dibandingkan dengan penggunaan HOG dan Haar Cascade dengan nilai *precision*, *recall*, dan *f1 score* yang lebih tinggi [40].

Penelitian selanjutnya yaitu mengenai sistem pendeteksi penggunaan masker dengan menggunakan Jetson Nano dan TensorFlow, serta teknik computer vision berbasis MobileNetV2 [41]. Sistem ini mencapai akurasi rata-rata 99,48%

dalam mendeteksi pengguna yang tidak mengenakan masker atau mengenakan masker dengan tidak benar, serta 99,12% dalam mendeteksi pengguna yang menggunakan masker dengan benar. Pada pengujian sistem ini, *platform* Jetson Nano memproses data gambar dalam waktu 0,114 detik, sebagai perbandingan, jauh lebih unggul daripada Raspberry Pi 4 yang memerlukan waktu kurang lebih 7 detik. Sehingga, dapat disimpulkan bahwa Jetson Nano memiliki performa yang lebih baik dalam memproses dan mengidentifikasi objek secara langsung [41].

Penelitian berikutnya membahas penggunaan *computer vision* dan *deep learning* untuk mendeteksi unta di jalan dan mencegah tabrakan antara kendaraan dan unta yang merupakan masalah serius di Arab Saudi dan daerah lainnya [42]. Penelitian ini membandingkan lima algoritma deteksi objek: CenterNet, EfficientDet, Faster R-CNN, SSD, dan YOLOv8, dengan menggunakan *dataset* khusus yang terdiri dari 250 gambar unta dalam berbagai konteks. Evaluasi dilakukan berdasarkan *mean Average Precision* (mAP) dan *mean Average Recall* (AR) pada berbagai threshold *intersection over union* (IoU). Hasil penelitian menunjukkan bahwa YOLOv8 adalah algoritma terbaik dari segi akurasi dan efisiensi, diikuti oleh CenterNet. Peneliti dari kajian ini menyimpulkan bahwa YOLOv8 adalah algoritma yang direkomendasikan untuk pengembangan sistem penghindaran tabrakan antara kendaraan dan unta berbasis *deep learning* di masa depan [42].

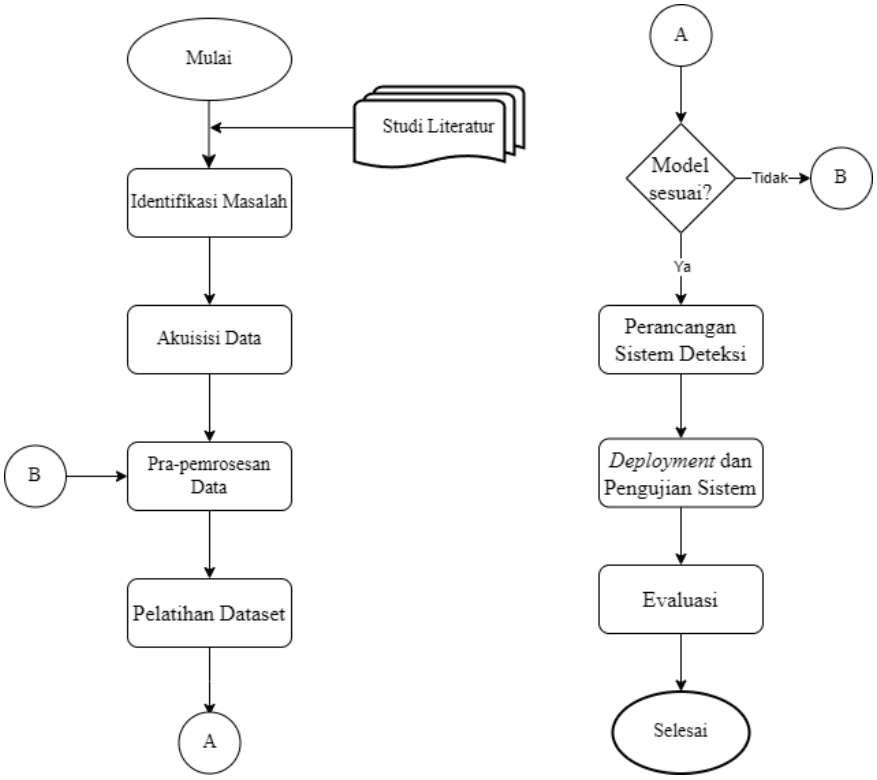
Selanjutnya pada penelitian ini menghadirkan sebuah proyek penelitian yang menggunakan YOLOv5 dan *ensemble learning* untuk secara otomatis mendeteksi dan mengklasifikasikan pengendara sepeda motor dan penumpangnya berdasarkan penggunaan helm [43]. Hasil menunjukkan skor mAP yang tinggi yaitu 0,526 pada data uji, dengan benar menandai sebagian besar kelas terlepas dari kondisi pencahayaan atau cuaca dari video. Meskipun demikian, model ini belum dapat diterapkan *deployment* secara *real-time* untuk menghadapi tantangan di dunia nyata seperti pengaruh perangkat *deployment* yang digunakan variasi kondisi pencahayaan, adanya halangan, dan berbagai jenis sepeda motor dan helm. Faktor-faktor ini dapat memengaruhikinerja dan keandalan model saat diterapkan dalam situasi praktis [43].

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.

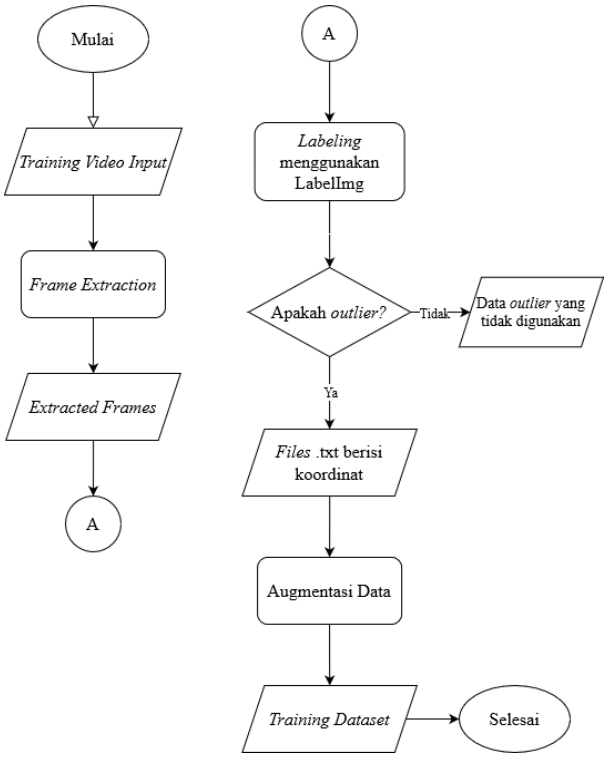
Video untuk kelas ‘Helm’ dan ‘Rider’ diambil pada Jalan Raya Cilegon direkam dari Jembatan Penyeberangan Orang (JPO) Cilegon, tempat yang dipilih karena frekuensi tinggi pengendara motor yang menggunakan helm. Sementara itu, data untuk kelas ‘Tidak Helm’ diambil dari lokasi yang memiliki banyak orang 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*

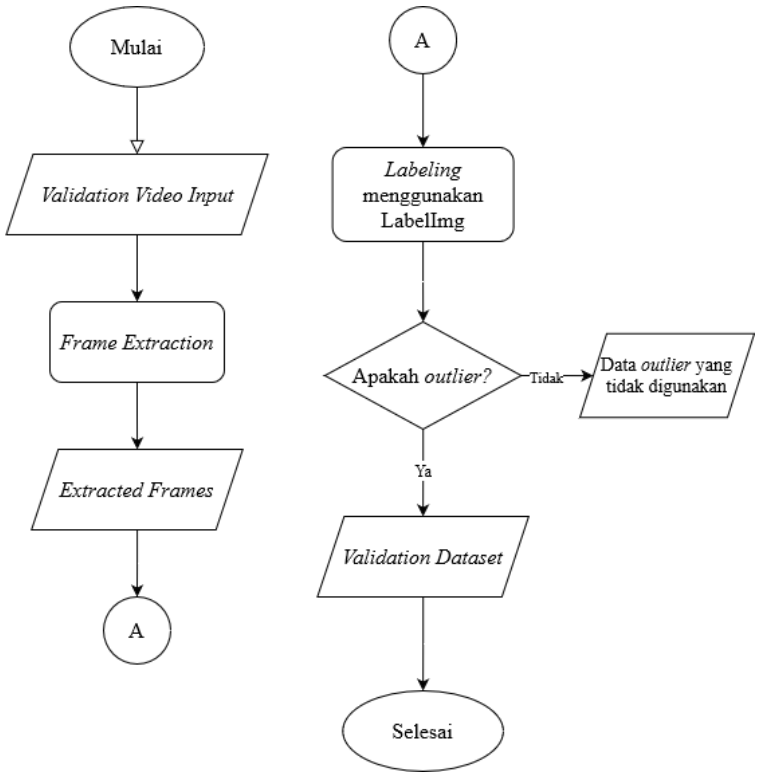
Proses ini bertujuan untuk membuat dataset yang akan digunakan untuk melatih model YOLOv8. Dataset terdiri dari tiga bagian: *training*, validation, dan testing, yang diambil dari video yang telah direkam sebelumnya. Video tersebut kemudian diproses melalui tahap pre-processing menggunakan tiga proses 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*.

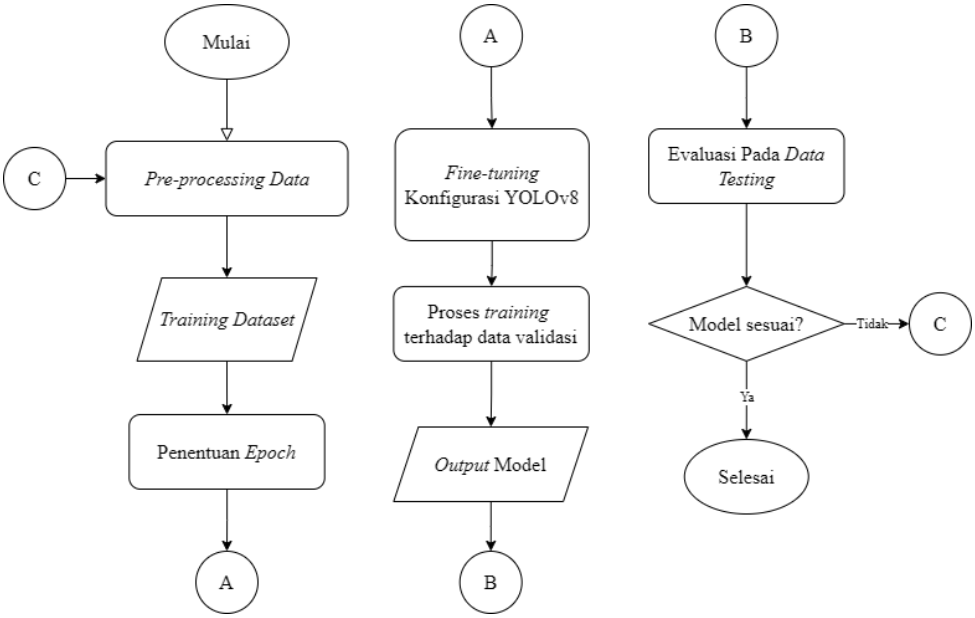
Tabel 3.5 Jumlah Total Anotasi Pada Tiap Kategori Kelas

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.

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

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

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

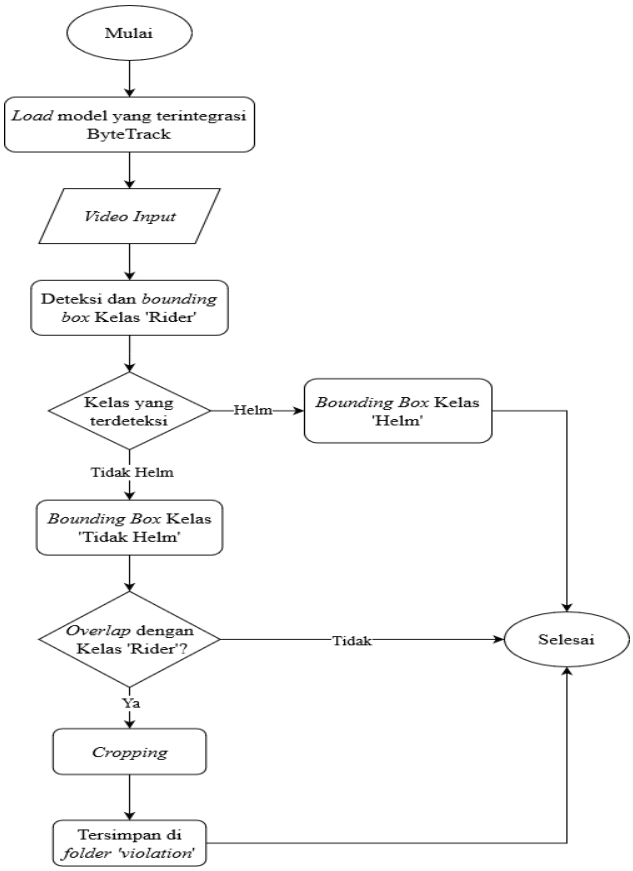
$$AP_i = \int_0^1 Prec(Rec)d(Rec) + C \tag{3.4}$$

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

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.

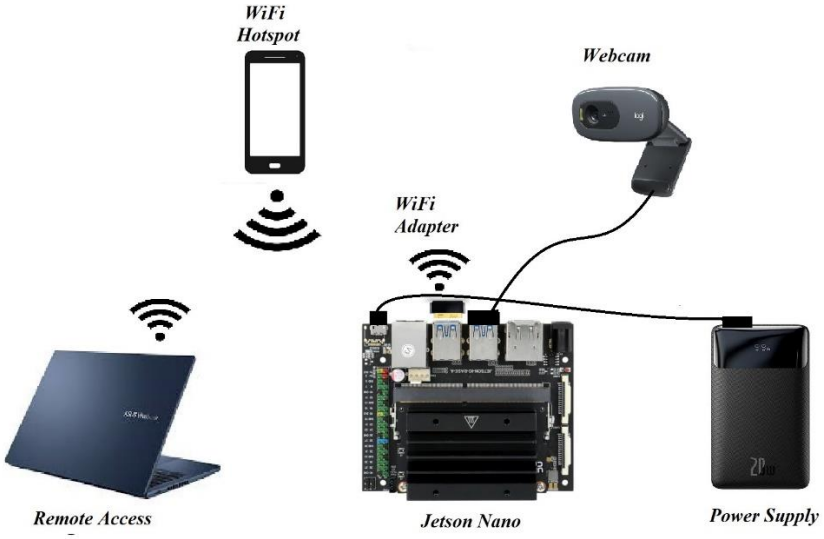
3.3.6 *Deployment* dan Pengujian Sistem

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

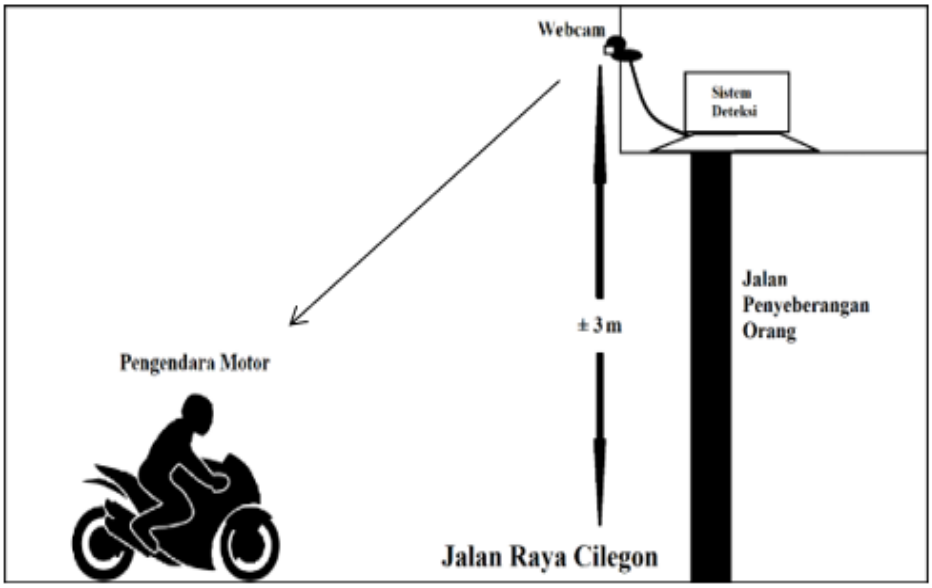
Parameter	Keterangan
Lokasi	Jl. SA. Tirtayasa, Cilegon
Penempatan Sistem	JPO Simpang Tiga
Waktu/Kondisi Pencahayaan	Pagi Hari, Siang Hari, Sore Hari
Jumlah Lajur	1 Lajur
Arah Kamera	Barat

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.



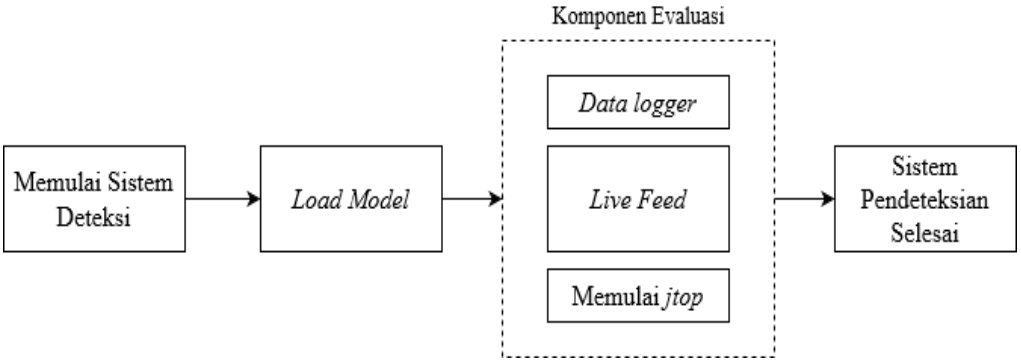
Gambar 3.6 Skema Rangkaian Pengujian Sistem

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.

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

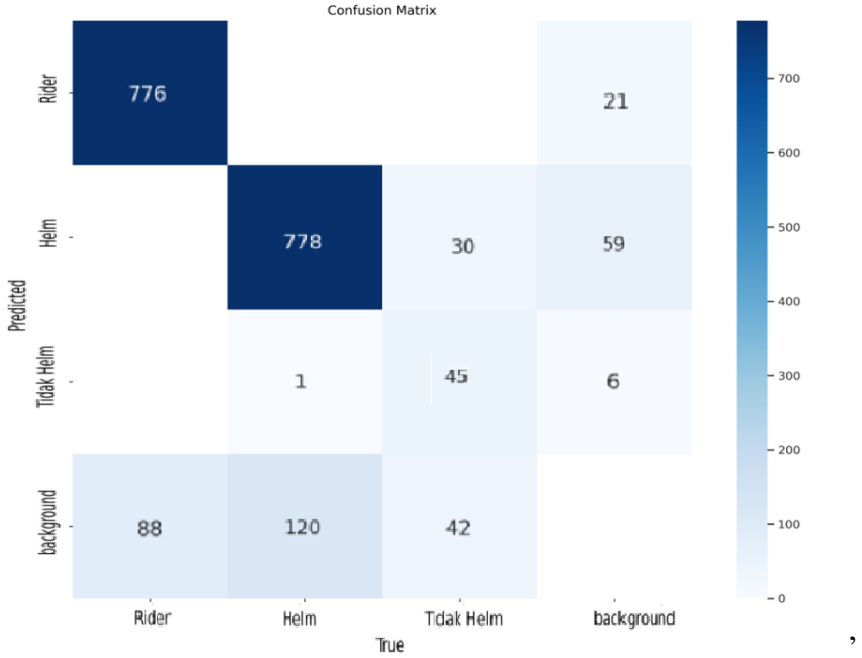
Komponen perhitungan FPS yang digunakan pada Persamaan (3.6) yaitu merupakan rata-rata kecepatan waktu *pre-processing* pada tiap *frame*, waktu inferensi pada *tiap frame*, dan waktu *post-processing* pada tiap *frame* yang dihasilkan *data logger* pada waktu sistem beroperasi secara *real-time*.

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*.

BAB IV
ANALISIS DAN PEMBAHASAN

4.1 Analisis Model *Training* YOLOv8

Proses eksperimental dalam pengembangan model YOLOv8 yang disesuaikan dilakukan berulang kali pada dataset yang telah diolah sebelumnya. Upaya ini diarahkan untuk mendapatkan konfigurasi yang menghasilkan kinerja optimal. Pemantauan eksperimen dijalankan dengan menggunakan ClearML, yang memfasilitasi *tracking* yang rinci dan terstruktur. Dihasilkan metrik evaluasi dan *confusion matrix* dari model *training* yang diterapkan pada data validasi, berikut *confusion matrix* yang dihasilkan seperti pada Gambar 4.1.



Gambar 4.1 *Confusion Matrix* Model *Training*

Pada Gambar 4.1, *confusion matrix* mengungkapkan bahwa model dengan tepat mengklasifikasikan ‘Rider’ sebanyak 776 kali dari 864 anotasi dan ‘Helm’ sebanyak 778 kali dari 899 anotasi, menunjukkan tingkat *true positives* yang tinggi untuk kedua kelas tersebut. Di sisi lain, ‘Tidak Helm’ diidentifikasi benar hanya 45 kali dari 117 anotasi, yang lebih rendah dibandingkan dengan dua kelas lainnya. Kelas ‘*background*’ merupakan area yang dihasilkan oleh model YOLOv8, kelas ini merupakan area non-objek, yang akan berkorelasi dengan

prediksi *false negative* dan *false positive* agar lebih komprehensif. Kesalahan deteksi terjadi saat:

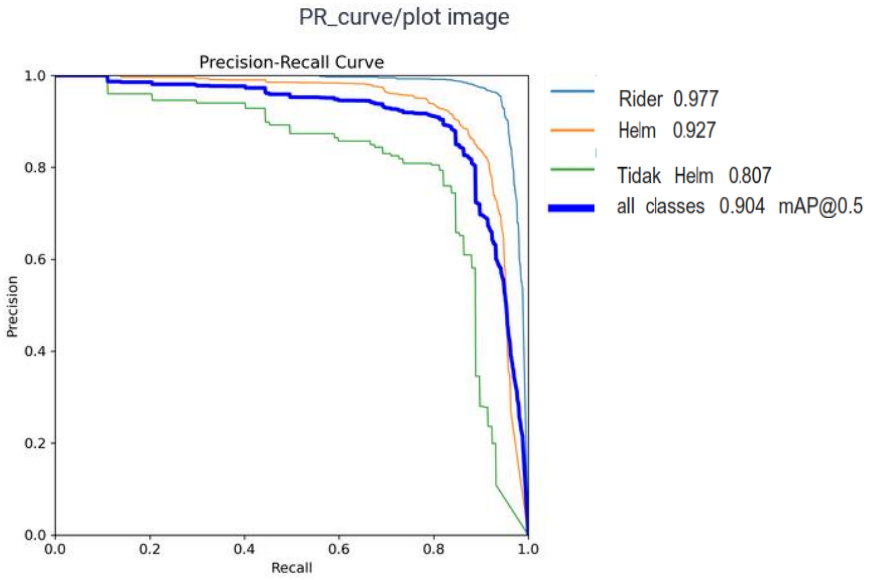
a. Kelas ‘Rider’ diprediksi sebagai ‘*background*’ sebanyak 88 kali.
b. Kelas ‘Helm’ diprediksi sebagai kelas ‘Tidak Helm’ 1 kali, dan ‘*background*’ 120 kali.
c. Kelas ‘Tidak Helm’ diprediksi sebagai kelas ‘Helm’ sebanyak 30 kali, dan ‘*background*’ 42 kali.

Model ini juga sering mengklasifikasikan ‘*background*’ sebagai kelas objek, dengan sebanyak 6 untuk kelas ‘Rider’, 59 untuk kelas ‘Helm’, 21 untuk kelas ‘Tidak Helm’. Dari hasil analisis *confusion matrix*, dapat dilihat korelasinya dengan nilai metrik evaluasi seperti pada Tabel 4.1.

Tabel 4.1 Hasil Metrik Evaluasi Model *Training*

Kelas	***Precision***	***Recall***	**mAP50**	**mAP50-95**	***F_1-score***
Semua Kelas	85.9%	87.9%	90.4%	50.5%	86.8%
Rider	95.6%	93.8%	97.7%	68.0%	94.5%
Helm	81.3%	91.5%	92.7%	46.2%	85.9%
Tidak Helm	80.6%	78.3%	80.7%	37.3%	79.4%

Pada Tabel 4.2, hasil evaluasi YOLOv8 menunjukkan *precision* 85.9% dan *recall* 87.9%. Namun, mAP50-95 rendah menunjukkan kesulitan dalam mendeteksi objek pada IoU tinggi. Deteksi ‘Rider’ unggul dengan nilai *precision*, *recall*, mAP50, mAP50-95, dan *f_1-score* yang tinggi. Deteksi ‘Helm’ memiliki *recall* tinggi namun *precision* rendah, cenderung menghasilkan *false positive* pada IoU tinggi. Deteksi ‘Tidak Helm’ menunjukkan keseimbangan, meskipun kesulitan pada IoU tinggi. Kesulitan pada IoU tinggi disebabkan oleh bentuk objek yang rumit, variasi pencahayaan, dan gangguan lingkungan, serta dataset *training* yang kurang representatif atau ketidakseimbangan sampel kelas. Selain itu, kurva *precision-recall* seperti pada Gambar 4.2 memberikan pemahaman lebih rinci tentang performa model.



Gambar 4.2 Kurva *Precision-Recall* Model *Training*

Pada Gambar 4.2, kurva *precision-recall* menunjukkan keseimbangan antara *precision* dan recall untuk berbagai nilai threshold pada tiga kelas dan keseluruhan kelas. Kelas 'Rider' memiliki *Average Precision* (AP) tertinggi sebesar 0.977, diikuti oleh 'Helm' pada 0.927, dan 'Tidak Helm' pada 0.807, dengan mAP untuk semua kelas sebesar 0.904 pada IoU 0.5. Kurva menunjukkan *precision* tinggi pada recall rendah, menandakan keakuratan model dalam mengidentifikasi kasus positif yang mudah, namun *precision* menurun dengan peningkatan recall. Penurunan *precision* pada recall tinggi menunjukkan kesulitan model dalam mengidentifikasi semua kasus positif. *Area under curve* (AUC) mengukur seberapa baik model membedakan antara kelas, dan kurva yang mendekati sudut kanan atas menunjukkan model yang akurat.

4.2 Analisis *Testing* Model Pada Video

Dalam tahap ini, model pelatihan yang telah jadi akan melanjutkan proses *testing* dengan mengintegrasikan model ke dalam algoritma *tracking* Bytetrack dan menganalisis kinerjanya pada video, dengan parameter evaluasi yang sama dengan sebelumnya. Pada kasus pertama yaitu kasus pencahayaan pagi hari, memiliki hasil deteksi serta evaluasi seperti Tabel 4.2.

Tabel 4.2 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Pagi Hari

Kelas	Aktual	Hasil Prediksi			*Precision*	*Recall*	F_1-*score*
		TP	FP	FN			
Rider	350	350	12	0	96.7%	100.0%	98.3%
Helm	443	443	48	0	90.2%	100.0%	94.9%
Tidak Helm	30	24	10	6	70.6%	80.0%	75.0%

Pada Tabel 4.2, hasil prediksi dan evaluasi model untuk kondisi pencahayaan pagi hari menunjukkan bahwa kelas ‘Rider’ memiliki *precision* tertinggi sebesar 96.7%. Kelas ‘Rider’ dan ‘Helm’ mencapai nilai recall sempurna sebesar 100.0%, sedangkan kelas ‘Tidak Helm’ memiliki recall sebesar 80.0%. Mean harmonik antara *precision* dan *recall* tertinggi ditemukan pada kelas ‘Rider’, yaitu 98.3%. Meskipun model akurat untuk kelas ‘Rider’ dan ‘Helm’, terdapat kasus *false positive* pada objek non-aktual dan latar belakang, serta prediksi salah untuk kelas lain. Kelas ‘Tidak Helm’ mengalami *false positive* dan *false negative* yang mempengaruhi nilai evaluasi. Pada kasus kedua yaitu kasus pencahayaan siang hari, memiliki hasil deteksi serta evaluasi seperti Tabel 4.3.

Tabel 4.3 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Siang Hari

Kelas	Aktual	Hasil Prediksi			*Precision*	*Recall*	F_1-*score*
		TP	FP	FN			
Rider	381	381	24	0	94.1%	100.0%	96.9%
Helm	481	480	44	1	91.6%	99.8%	95.5%
Tidak Helm	19	13	25	6	34.2%	68.4%	45.6%

Pada Tabel 4.3, hasil prediksi dan evaluasi model pada kondisi pencahayaan siang hari menunjukkan bahwa kelas ‘Rider’ dan ‘Helm’ memiliki *precision* tinggi, sementara kelas ‘Tidak Helm’ memiliki precision rendah yaitu 34.2%. Nilai *recall* sempurna dicapai pada kelas ‘Rider’ dan hampir sempurna pada kelas ‘Helm’, tetapi rendah pada kelas ‘Tidak Helm’ yaitu 68.4%. Mean harmonik antara *precision* dan *recall* tertinggi ditemukan pada kelas ‘Rider’ dengan 96.9%, sedangkan kelas ‘Tidak Helm’ hanya 45.6%. Kesalahan prediksi pada kelas ‘Tidak Helm’ disebabkan oleh *false positive* pada objek non-aktual dan latar

belakang serta kesalahan prediksi pada kelas lain. Beberapa pelanggaran ‘Tidak Helm’ juga terlewat atau *false negative*, mempengaruhi nilai evaluasi kelas tersebut. Pada kasus terakhir yaitu kasus pencahayaan sore hari, memiliki hasil deteksi serta evaluasi seperti Tabel 4.4.

Tabel 4.4 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Siang Hari

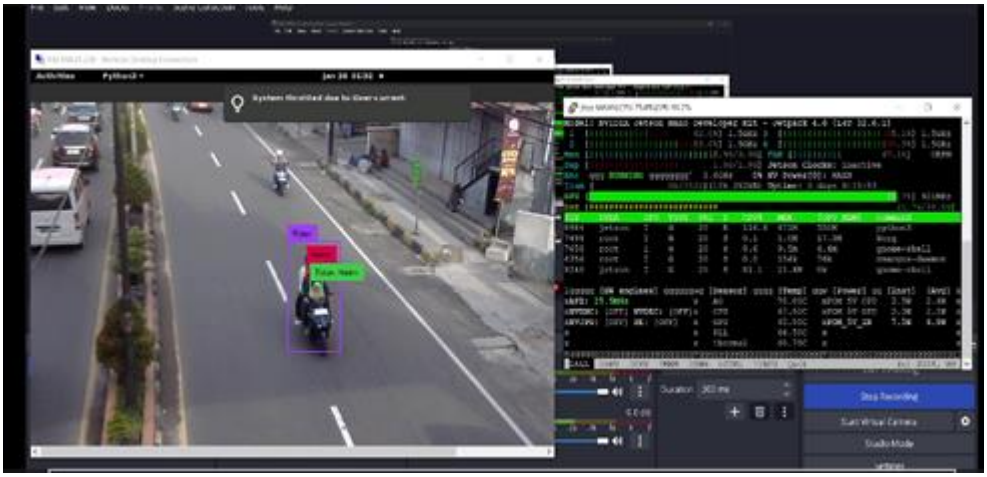
Kelas	**Aktual**	**Hasil Prediksi**			***Precision***	***Recall***	***F_1-score***
		TP	**FP**	**FN**			
Rider	610	607	5	3	99.2%	99.5%	99.4%
Helm	728	721	53	7	93.2%	99.0%	96.0%
Tidak Helm	16	11	23	5	32.4%	68.8%	44.0%

Pada Tabel 4.4, hasil evaluasi model pada kondisi pencahayaan siang hari menunjukkan kelas ‘Rider’ dan ‘Helm’ memiliki *precision* tinggi, masing-masing 99.2% dan 93.2%, sedangkan *precision* kelas ‘Tidak Helm’ rendah, hanya 32.4%. *Recall* kelas ‘Rider’ dan ‘Helm’ baik, tetapi rendah untuk kelas ‘Tidak Helm’ yaitu 68.8%. Mean harmonik antara *precision* dan *recall* tertinggi pada kelas ‘Rider’ mencapai 99.5%, tetapi rendah pada kelas ‘Tidak Helm’ yaitu 44.0%. Kesalahan prediksi pada kelas ‘Tidak Helm’ disebabkan oleh *false positive* pada objek non-aktual dan latar belakang serta prediksi yang salah pada kelas lain. Beberapa kasus pada ketiga kelas terlewat dalam prediksi atau terdeteksi sebagai *false negative*, memengaruhi evaluasi performa ketiga kelas tersebut.

4.3 Analisis Pengujian Sistem *Live Feed* menggunakan Jetson Nano

Pada tahap ini, model dan sistem yang sudah diuji dan dikembangkan sebelumnya akan langsung diuji dalam lingkungan operasional menggunakan perangkat Jetson Nano. Pengujian dilakukan untuk mengevaluasi kinerja sistem dalam mendeteksi objek secara *real-time*. Pengujian dilaksanakan pada ketiga kondisi pencahayaan yang berbeda, yakni pagi, siang, dan sore, dan berlangsung selama 25 menit. Analisis dilakukan terhadap metrik evaluasi seperti *precision*, *recall*, dan *f_1-score*, serupa dengan tahap sebelumnya. Selain itu, evaluasi juga mempertimbangkan performa mikrokomputer Jetson Nano dalam menjalankan sistem, dengan memperhatikan penggunaan sumber daya. Analisis ini bertujuan untuk

mendapatkan pemahaman yang lebih mendalam tentang kinerja sistem secara keseluruhan dalam konteks pengujian yang dilakukan. Pada Gambar 4.3, merupakan tampilan dari sistem deteksi helm dengan penggunaan Jetson Nano.



Gambar 4.3 Tampilan Sistem Deteksi Helm *Real-time* menggunakan Jetson Nano

Pada Gambar 4.3, merupakan *Graphical User Interface* (GUI) untuk sistem deteksi helm yang dioperasikan pada laptop melalui aplikasi *remote access*, yaitu PuTTY. Jendela di sebelah kanan menampilkan *jtop* yang digunakan untuk mengevaluasi kinerja dan menganalisis sumber daya Jetson Nano ketika sistem deteksi helm sedang berjalan. Evaluasi mempertimbangkan performa mikrokomputer Jetson Nano dalam menjalankan sistem, dengan memperhatikan penggunaan sumber daya seperti CPU1 hingga CPU4, RAM, dan GPU, suhu komponen, daya termal serta daya total saat operasi sistem.

4.3.2 Hasil Analisis Pengujian Sistem pada Pagi Hari

Pada kasus pertama, yang merujuk pada kondisi pencahayaan pada pagi hari, pengujian dilakukan pada pukul 07.18 pagi, dimulai dengan periode pengoperasian sistem *live feed* selama 25 menit. Hasil pengujian yang mencakup kinerja model dalam mengidentifikasi objek disajikan pada Tabel 4.5.

Tabel 4.5 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Pagi Hari

Kelas	**Aktual**	**Hasil Prediksi**			***Precision***	***Recall***	***F_1-score***
		TP	**FP**	**FN**			

Rider	1294	1196	6	98	99.5%	92.4%	95.8%
Helm	1533	1473	13	60	99.1%	96.1%	97.6%
Tidak Helm	17	12	47	5	20.3%	70.6%	31.6%
Semua Kelas x̄					73.0%	86.3%	75.0%

Pada Tabel 4.5, model menunjukkan *precision* dan *recall* yang sangat baik untuk kelas 'Rider' dan 'Helm' dengan kedua metrik tersebut lebih dari 90% dan nilai mean harmonik atau f_1*-score* masing-masing sebesar 95.8% dan 97.6%. Ini menandakan kemampuan yang sangat baik untuk mengidentifikasi pengendara dan helm dengan benar. Namun, kinerja model menurun secara signifikan untuk kelas 'Tidak Helm' dengan *precision* hanya 20.3%, meskipun *recall*-nya relatif lebih tinggi pada 70.6%. Ini menghasilkan f_1*-score* yang rendah sebesar 31.6%, menunjukkan bahwa model sering salah mengklasifikasikan objek lain atau latar belakang sebagai 'Tidak Helm'. Performa yang dihasilkan bisa dipengaruhi oleh beberapa faktor, termasuk performa penggunaan Jetson Nano secara *real-time*, kualitas *webcam* yang digunakan, serta kondisi pencahayaan di pagi hari. Secara keseluruhan, sementara model sangat efektif untuk dua kelas, model memerlukan peningkatan dalam membedakan kelas "Tidak Helm" untuk mengurangi *false positives* dan meningkatkan kinerja secara keseluruhan. Berikut adalah hasil kecepatan rata-rata saat menjalankan sistem untuk kasus pagi hari yang disajikan pada Tabel 4.6.

Tabel 4.6 Hasil Kecepatan Pemrosesan Deteksi Per-*frame* Pada Pagi Hari

Parameter	**x̄ Waktu**
Pre-processing Speed	7.8 ms
Inference Time	72.4 ms
Post-processing Speed	6.6 ms

Pada Tabel 4.6, terdapat hasil rata-rata *output log* kecepatan pemrosesan per-*frame* dari pengujian sistem deteksi secara *real-time* menggunakan Jetson Nano. Rata-rata waktu pra-pemrosesan *frame* sekitar 7.8 ms menunjukkan kemampuan sistem dalam menyiapkan gambar untuk inferensi dengan cepat. Selama proses inferensi frame, waktu sekitar 72.4 ms dianggap wajar untuk sistem deteksi objek *real-time* pada perangkat *edge* seperti Jetson Nano. Meskipun ada sistem yang

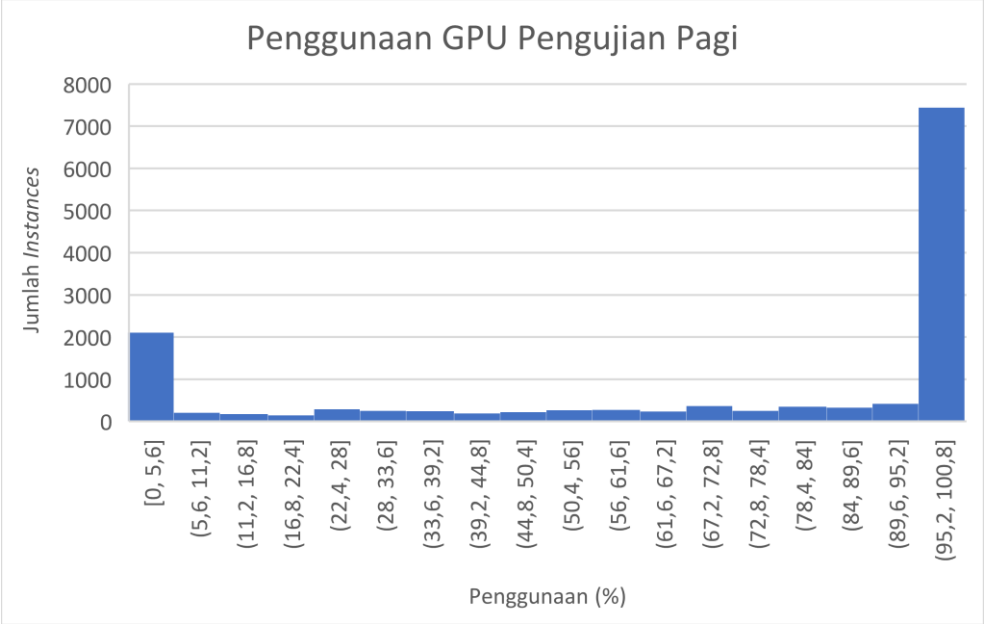
lebih cepat, waktu ini masih cukup baik untuk banyak aplikasi *real-time*. Waktu pasca-pemrosesan sekitar 6.6 ms menunjukkan kemampuan sistem dalam menyelesaikan tugas seperti pelacakan objek dan klasifikasi tambahan dengan cepat setelah deteksi. Dari analisis waktu pemrosesan yang dilakukan, diperoleh hasil kalkulasi untuk *frame per second* (FPS) sekitar 11.7. Meskipun angka FPS ini masih dianggap cukup untuk sebagian besar kasus, untuk mendeteksi objek kendaraan yang bergerak secara *real-time*, nilai FPS ini masih dianggap belum optimal. Faktor-faktor seperti sumber daya yang tersedia pada Jetson Nano serta kompleksitas model dan sistem dapat memengaruhi nilai FPS. Dari pengujian ini, didapat hasil kinerja komputasi Jetson Nano seperti pada Tabel 4.7.

Tabel 4.7 Hasil Performa Jetson Nano pada Pagi Hari

Metrik Performa	**Penggunaan**
CPU1	73.9%
CPU2	79.4%
CPU3	80.1%
CPU4	80.9%
RAM	76.3%
Temperatur CPU	34°C -70°C
Temperatur GPU	33.5°C -67.5°C
Daya Total	6.7 W

Pada Tabel 4.7, merupakan hasil performa komputasi Jetson Nano pada kasus pencahayaan di pagi hari. Perangkat ini menunjukkan kinerja yang intensif dengan penggunaan CPU mencapai 73.9% hingga 80.9% untuk keempat *core*-nya. Ini menandakan kemampuan Jetson Nano dalam menangani beban kerja dari algoritma deteksi objek. Meskipun demikian, masih ada ruang untuk pemrosesan data tambahan, sebagaimana ditunjukkan oleh penggunaan RAM yang mencapai 76.3%. Terjadi fluktuasi suhu yang signifikan, namun kemudian suhu kembali turun hingga mencapai kisaran 65°C untuk kedua komponen, menandakan adanya mekanisme pendinginan yang efektif untuk menjaga keandalan sistem dalam jangka panjang. Selain itu, terlihat efisiensi energi yang baik dari konsumsi daya perangkat sekitar 6.7 W selama operasi, berada di bawah batas operasional standar 10 W. Hal ini menunjukkan bahwa Jetson Nano mampu menjaga keseimbangan

beban operasional secara optimal, terutama untuk tugas-tugas yang memerlukan pemrosesan intensif seperti deteksi objek. Untuk penggunaan GPU, berikut disajikan grafik histogram pada Gambar 4.4 untuk persentase penggunaan GPU.



Gambar 4.4 Grafik Penggunaan GPU Pengujian Pagi Hari

Pada Gambar 4.4, merupakan grafik histogram untuk persentase penggunaan GPU selama operasi pada Jetson Nano untuk kasus pencahayaan pagi hari. Penggunaan GPU pada sistem deteksi helm menunjukkan pola fluktuasi yang signifikan. Fluktuasi ini mencapai puncak hingga 99.7%, yang menandakan periode di mana GPU bekerja hampir pada kapasitas maksimumnya. Setelah mencapai puncak, penggunaan GPU kemudian turun drastis mendekati 0%, menunjukkan bahwa ada interval di mana beban kerja GPU sangat ringan atau tidak ada sama sekali. Fluktuasi ini terjadi beberapa kali dan secara acak, yang bisa menandakan bahwa sistem melakukan pemrosesan data dalam *batch* pada interval yang tidak teratur, menunggu hingga akumulasi data tertentu terkumpul sebelum memulai proses inferensi, dan setelah *batch* data diproses, penggunaan GPU turun kembali karena sistem menunggu kumpulan data berikutnya.

4.3.3 Hasil Analisis Pengujian Sistem pada Siang Hari

Pada kasus kedua, yang merujuk pada kondisi pencahayaan pada siang hari, pengujian dilakukan pada pukul 14.25 siang, dimulai dengan periode

pengoperasian sistem *live feed* selama 25 menit. Hasil pengujian yang mencakup kinerja model dalam mengidentifikasi objek disajikan pada Tabel 4.8.

Tabel 4.8 Hasil Prediksi Dan Evaluasi Model Pada Pengujian *Real-time* Sistem Siang Hari

Kelas	**Aktual**	**Hasil Prediksi**			***Precision***	***Recall***	***F_1-score***
		TP	**FP**	**FN**			
Rider	768	264	0	504	100.0%	34.4%	51.2%
Helm	949	605	2	344	99.7%	63.8%	77.8%
Tidak Helm	34	16	42	18	27.6%	47.1%	34.8%
Semua Kelas x̄					75.8%	48.4%	54.6%

Pada Tabel 4.8, model menunjukkan tingkat presisi yang tinggi untuk kelas 'Rider' dan 'Helm', melebihi 90%. Namun, *recall* untuk kedua kelas tersebut relatif rendah, menyebabkan penurunan *f_1-score*. Hal ini menandakan kemampuan model dalam mengenali pengendara dan helm dengan baik, namun seringkali melewatkan objek yang sebenarnya ada. Kinerja model juga rendah untuk kelas 'Tidak Helm', dengan presisi hanya sekitar 27.6% dan *recall* sekitar 47.1%, menghasilkan *f_1-score* rendah sekitar 34.8%. Kemungkinan performa yang rendah dipengaruhi oleh beberapa faktor, termasuk kinerja Jetson Nano, kualitas *webcam*, dan kondisi pencahayaan. Secara keseluruhan, model kurang efektif dalam mengklasifikasikan ketiga kelas, memerlukan peningkatan untuk membedakan kelas 'Tidak Helm' dan meningkatkan kinerja secara keseluruhan. Berikut adalah hasil kecepatan rata-rata saat menjalankan sistem untuk kasus siang hari yang disajikan pada Tabel 4.9.

Tabel 4.9 Hasil Kecepatan Pemrosesan Deteksi Per-*frame* Pada Siang Hari

Parameter	**x̄ Waktu**
Pre-processing Speed	7.9 ms
Inference Time	78.3 ms
Post-processing Speed	4.9 ms

Pada Tabel 4.9, terdapat hasil rata-rata *output log* kecepatan pemrosesan per-*frame* dari pengujian sistem deteksi secara *real-time* menggunakan Jetson Nano.

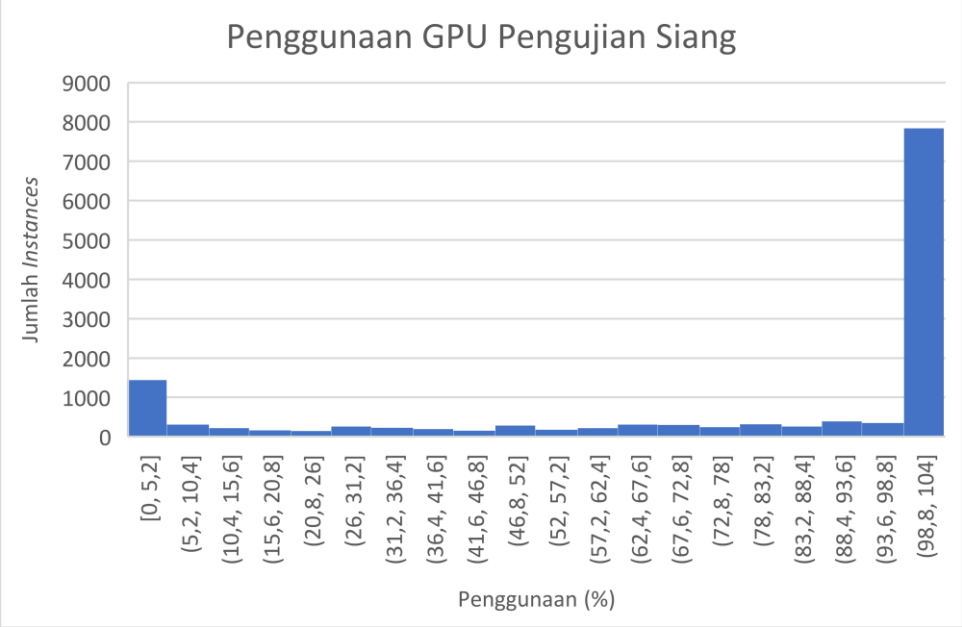
Rata-rata waktu pra-pemrosesan *frame* sekitar 7.9 ms menunjukkan kemampuan sistem dalam menyiapkan gambar untuk inferensi dengan cepat. Selama proses inferensi frame, waktu sekitar 78.3 ms dianggap wajar untuk sistem deteksi objek *real-time* pada perangkat *edge* seperti Jetson Nano. Meskipun ada sistem yang lebih cepat, waktu ini masih cukup baik untuk banyak aplikasi *real-time*. Waktu pasca-pemrosesan sekitar 4.9 ms menunjukkan kemampuan sistem dalam menyelesaikan tugas seperti pelacakan objek dan klasifikasi tambahan dengan lebih cepat dari pengujian sebelumnya setelah deteksi. Dari analisis waktu pemrosesan yang dilakukan, diperoleh hasil kalkulasi untuk *frame per second* (FPS) sekitar 11.1. Meskipun angka FPS ini masih dianggap cukup untuk sebagian besar kasus, untuk mendeteksi objek kendaraan yang bergerak secara *real-time*, nilai FPS ini masih dianggap belum optimal. Faktor-faktor seperti sumber daya yang tersedia pada Jetson Nano serta kompleksitas model dan sistem dapat memengaruhi nilai FPS. Dari pengujian ini, didapat hasil kinerja komputasi Jetson Nano seperti pada Tabel 4.10.

Tabel 4.10 . Hasil Performa Jetson Nano pada Siang Hari

Metrik Performa	Penggunaan
CPU1	77.1%
CPU2	81.1%
CPU3	82.6%
CPU4	83.3%
RAM	79.6%
Temperatur CPU	34°C -50.5°C
Temperatur GPU	34°C -49°C
Daya Total	6.2 W

Pada Tabel 4.10 ditunjukkan hasil performa komputasi Jetson Nano pada kasus pencahayaan di siang hari. Perangkat ini menunjukkan penggunaan CPU antara 77.1% hingga 83.3% untuk keempat *core*-nya, menandakan kemampuannya dalam menangani beban kerja dari algoritma deteksi objek. Penggunaan RAM mencapai 79.6%, menunjukkan masih ada ruang untuk pemrosesan data tambahan. Terjadi peningkatan suhu yang cukup signifikan, namun suhu kembali stabil di kisaran 45°C untuk kedua komponen, menunjukkan mekanisme pendinginan yang efektif. Konsumsi daya perangkat sekitar 6.2 W, berada di

bawah batas operasional standar 10 W, menunjukkan efisiensi energi yang baik. Untuk penggunaan GPU, disajikan grafik histogram pada Gambar 4.5.



Gambar 4.5 Grafik Penggunaan GPU Pengujian Siang Hari

Pada Gambar 4.5, merupakan grafik histogram untuk persentase penggunaan GPU selama operasi pada Jetson Nano untuk kasus pencahayaan pagi hari. Penggunaan GPU pada sistem deteksi helm menunjukkan pola fluktuasi yang signifikan. Fluktuasi ini mencapai puncak hingga 99.9%, yang menandakan periode di mana GPU bekerja hampir pada kapasitas maksimumnya. Setelah mencapai puncak, penggunaan GPU kemudian turun drastis mendekati 0%, menunjukkan bahwa ada interval di mana beban kerja GPU sangat ringan atau tidak ada sama sekali. Fluktuasi ini terjadi beberapa kali dan secara acak, yang bisa menandakan bahwa sistem melakukan pemrosesan data dalam *batch* pada interval yang tidak teratur, menunggu hingga akumulasi data tertentu terkumpul sebelum memulai proses inferensi, dan setelah *batch* data diproses, penggunaan GPU turun kembali karena sistem menunggu kumpulan data berikutnya.

4.3.4 Hasil Analisis Pengujian Sistem pada Sore Hari

Pada kasus terakhir, yang merujuk pada kondisi pencahayaan pada siang hari, pengujian dilakukan pada pukul 18.02 sore, dimulai dengan periode

pengoperasian sistem *live feed* selama 25 menit. Hasil pengujian yang mencakup kinerja model dalam mengidentifikasi objek disajikan pada Tabel 4.11.

Tabel 4.11 Hasil Prediksi Dan Evaluasi Model Pada Pengujian *Real-time* Sistem Sore Hari

Kelas	**Aktual**	**Hasil Prediksi**			***Precision***	***Recall***	***F_1-score***
		TP	**FP**	**FN**			
Rider	1336	1291	7	45	99.5%	96.6%	98.0%
Helm	1567	1533	33	34	97.9%	97.8%	97.9%
Tidak Helm	35	9	11	26	45.0%	25.7%	32.7%
Semua Kelas x̄					80.8%	73.4%	76.2%

Pada Tabel 4.11, model menunjukkan *precision* dan *recall* yang sangat baik untuk kelas ‘Rider’ dan ‘Helm’ dengan kedua metrik tersebut lebih dari 90% dan nilai mean harmonik atau *f_1-score* masing-masing sebesar 98.0% dan 97.9%. Ini menandakan kemampuan yang sangat baik untuk mengidentifikasi pengendara dan helm dengan benar. Namun, kembali kinerja model lebih rendah untuk kelas ‘Tidak Helm’ dengan *precision* hanya 45.0% dan *recall* sebesar 25.7%. Ini menghasilkan *f_1-score* yang rendah sebesar 32.7%, menunjukkan bahwa model sering salah mengklasifikasikan objek 'Tidak Helm' atau melewatkan deteksinya sebagai latar belakang. Performa yang dihasilkan bisa dipengaruhi oleh beberapa faktor, termasuk performa penggunaan Jetson Nano secara *real-time*, kualitas *webcam* yang digunakan, serta kondisi pencahayaan di sore hari yang cenderung lebih gelap. Secara keseluruhan, sementara model sangat efektif untuk dua kelas, model memerlukan peningkatan dalam membedakan kelas “Tidak Helm” untuk mengurangi *false positives* dan meningkatkan kinerja secara keseluruhan. Berikut adalah hasil kecepatan rata-rata saat menjalankan sistem untuk kasus sore hari yang disajikan pada Tabel 4.12.

Tabel 4.12 Hasil Kecepatan Pemrosesan Deteksi Per-*frame* Pada Sore Hari

Parameter	**x̄ Waktu**
Pre-processing Speed	8.0 ms

Inference Time	85.6 ms
Post-processing Speed	5.0 ms

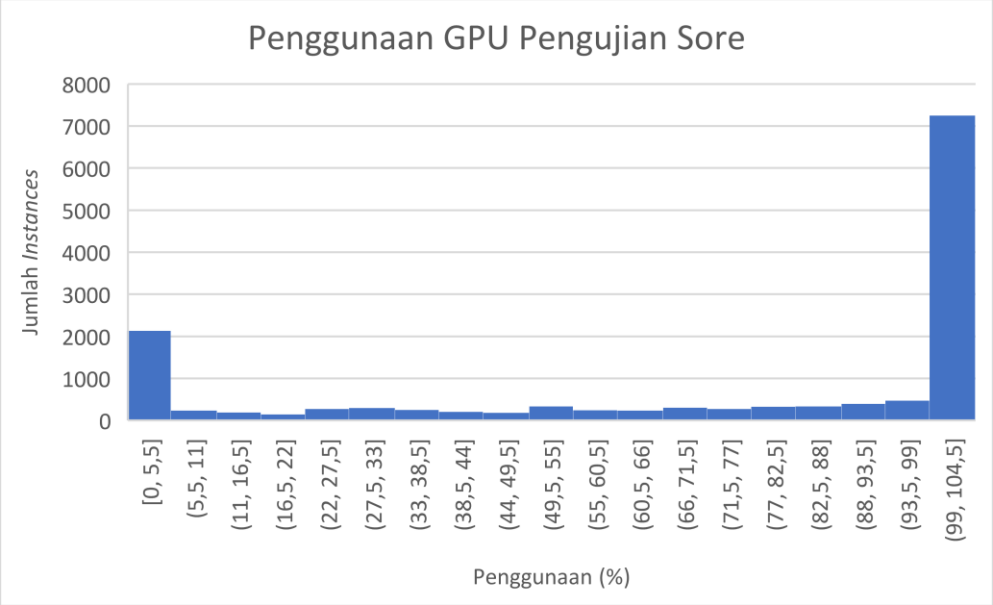
Pada Tabel 4.12, hasil rata-rata *output log* kecepatan pemrosesan per-*frame* dari pengujian sistem deteksi secara *real-time* menggunakan Jetson Nano ditampilkan. Waktu *pre-processing frame* sekitar 8.0 ms, menunjukkan kemampuan sistem dalam menyiapkan gambar untuk inferensi dengan cepat. Waktu inferensi *frame* sekitar 85.6 ms, sedikit lebih lama dari pengujian sebelumnya, namun masih wajar untuk perangkat *edge* seperti Jetson Nano. Waktu *post-processing* sekitar 5.0 ms, menunjukkan kemampuan sistem dalam menyelesaikan tugas tambahan dengan cepat setelah deteksi. Hasil analisis menunjukkan *frame per second* (FPS) sekitar 10.2, meskipun sedikit lebih rendah dari pengujian sebelumnya. Meskipun FPS ini cukup untuk banyak kasus, untuk deteksi objek kendaraan yang bergerak secara real-time, nilai FPS masih dianggap belum optimal. Faktor-faktor seperti sumber daya yang tersedia pada Jetson Nano dan kompleksitas model dan sistem dapat memengaruhi nilai FPS. Dari pengujian ini, didapat hasil kinerja komputasi Jetson Nano seperti pada Tabel 4.13.

Tabel 4.13 Hasil Performa Jetson Nano pada Sore Hari

Metrik Performa	**Penggunaan**
CPU1	70.8%
CPU2	75.8%
CPU3	76.9%
CPU4	77.3%
RAM	76.4%
Temperatur CPU	33.3°C -72°C
Temperatur GPU	35°C -69°C
Daya Total	6.5 W

Pada Tabel 4.13, merupakan hasil performa komputasi Jetson Nano pada kasus pencahayaan di siang hari. Perangkat ini menunjukkan kinerja yang intensif dengan penggunaan CPU mencapai 70.8% hingga 77.3% untuk keempat *core*-nya, namun tidak se-intensif kedua pengujian sebelumnya. Ini menandakan kemampuan Jetson Nano dalam menangani beban kerja dari algoritma deteksi objek. Meskipun demikian, masih ada ruang untuk pemrosesan data tambahan,

sebagaimana ditunjukkan oleh penggunaan RAM yang mencapai 76.4%. Terjadi peningkatan suhu yang signifikan, namun kemudian suhu kembali stabil di kisaran 50°C untuk kedua komponen, menandakan adanya mekanisme pendinginan yang efektif untuk menjaga keandalan sistem dalam jangka panjang. Selain itu, terlihat efisiensi energi yang baik dari konsumsi daya perangkat sekitar 6.5 W selama operasi, berada di bawah batas operasional standar 10 W. Hal ini menunjukkan bahwa Jetson Nano mampu menjaga keseimbangan beban operasional secara optimal, terutama untuk tugas-tugas yang memerlukan pemrosesan intensif seperti deteksi objek. Untuk penggunaan GPU, berikut disajikan grafik histogram pada Gambar 4.6 untuk persentase penggunaan GPU.



Gambar 4.6 Grafik Penggunaan GPU Pengujian Sore Hari

Pada Gambar 4.6, merupakan grafik histogram untuk persentase penggunaan GPU selama operasi pada Jetson Nano untuk kasus pencahayaan sore hari. Penggunaan GPU pada sistem deteksi helm menunjukkan pola fluktuasi yang signifikan. Sama seperti pengujian di kondisi sebelumnya, fluktuasi ini mencapai puncak hingga 99.9%, yang menandakan periode di mana GPU bekerja hampir pada kapasitas maksimumnya. Setelah mencapai puncak, penggunaan GPU kemudian turun drastis mendekati 0%, menunjukkan bahwa ada interval di mana beban kerja GPU sangat ringan atau tidak ada sama sekali. Fluktuasi ini terjadi beberapa kali dan secara acak, yang bisa menandakan bahwa sistem

melakukan pemrosesan data dalam *batch* pada interval yang tidak teratur, menunggu hingga akumulasi data tertentu terkumpul sebelum memulai proses inferensi, dan setelah *batch* data diproses, penggunaan GPU turun kembali karena sistem menunggu kumpulan data berikutnya.

4.4 Analisa Performa Sistem Deteksi Helm

Secara keseluruhan, faktor yang sangat berpengaruh pada performa sistem deteksi helm secara real-time adalah model yang digunakan dan beban komputasi itu sendiri. Dari model yang diperoleh untuk berbagai kondisi pencahayaan, diperoleh rata-rata untuk setiap kelas, serta rata-rata keseluruhan sistem di semua pengujian pencahayaan seperti yang disajikan pada Tabel 4.14 berikut.

Tabel 4.14 Hasil Pengujian Secara Keseluruhan

Kelas	***Precision*** $\bar{x}$	***Recall*** $\bar{x}$	F_1***-score*** $\bar{x}$
Rider	99.7%	74.5%	81.7%
Helm	98.9%	85.9%	91.1%
Tidak Helm	31.0%	47.8%	33.0%
Semua Kelas $\bar{x}$	76.5%	69.4%	68.6%

Pada Tabel 4.14, terlihat bahwa secara keseluruhan, pada kasus pengujian sistem deteksi, kelas ‘Helm’ memiliki nilai f_1*-score* yang lebih tinggi dibandingkan dengan kelas lainnya. Kelas dengan kinerja terendah ada pada kasus ‘Tidak Helm’. Hal ini menunjukkan bahwa model kesulitan membedakan kelas ‘Tidak Helm’ dari kelas lainnya, terutama dari kelas ‘Helm’. Rendahnya kinerja kelas ‘Tidak Helm’ memengaruhi nilai evaluasi model secara keseluruhan, menghasilkan nilai f_1*-score* yang tidak terlalu tinggi, yaitu 68.6%. Penyebabnya adalah banyaknya kasus *false positive* dan *false negative*. Kasus *false positive* dapat digambarkan pada sampel gambar pada pengujian langsung yang disajikan pada Gambar 4.7.

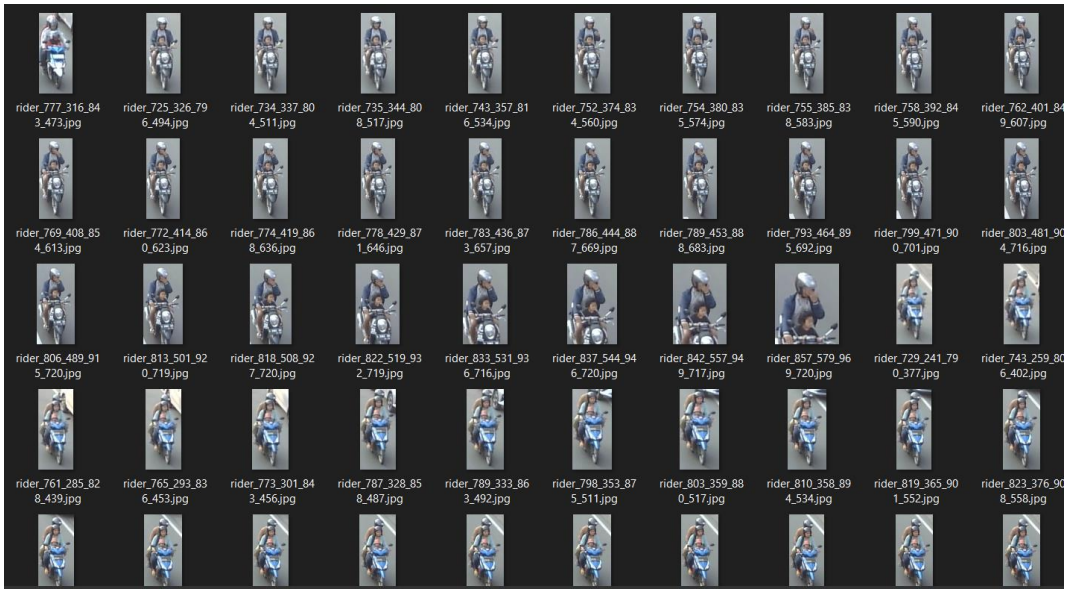
Gambar 4.7 Kasus *False Positives* pada Pengujian Langsung

Pada Gambar 4.7, terlihat beberapa contoh kasus *false positive* yang terjadi selama pengujian langsung. Untuk pengendara motor yang tidak menggunakan helm, sering kali prediksi yang muncul adalah kelas ‘Helm’, terutama saat objek mendekati kamera hingga keluar dari pandangan, namun dalam beberapa *frame* saat objek bergerak keluar dari pandangan, terkadang terdeteksi sebagai kelas ‘Tidak Helm’, menyebabkan deteksi yang tidak konsisten. Kasus *false positive* pada kelas ‘Tidak Helm’ sering terjadi pada pejalan kaki, yang bukan merupakan target penelitian sistem deteksi helm ini. Hal serupa juga terjadi pada kelas ‘Rider’, di mana pengendara motor yang melawan arah atau berada di jalur berbeda terkadang terdeteksi, meskipun kasus ini jarang. Ini disebabkan oleh sudut pandang kamera yang terlalu luas sehingga menampilkan trotoar, yang lebih efektif jika fokus pada jalan raya saja. Kasus lainnya pada kelas ‘Helm’ mencakup kesulitan membedakan antara aksesoris kepala seperti topi, kerudung, dan helm yang tidak khusus untuk pengendara motor. Pada kelas ‘Rider’, kesulitan muncul dalam membedakan pengendara sepeda, gerobak, dan kendaraan non-motor serupa lainnya. Selain itu, artefak lain di area jalan raya, seperti benda bulat, memengaruhi deteksi pada kelas ‘Helm’ dan ‘Tidak Helm’. Masalah ini bisa disebabkan oleh pencahayaan dan dataset yang kurang representatif. Kasus *false negative* dapat digambarkan pada sampel gambar pada pengujian langsung yang disajikan pada Gambar 4.7.

Gambar 4.8 Kasus *False Negatives* pada Pengujian Langsung

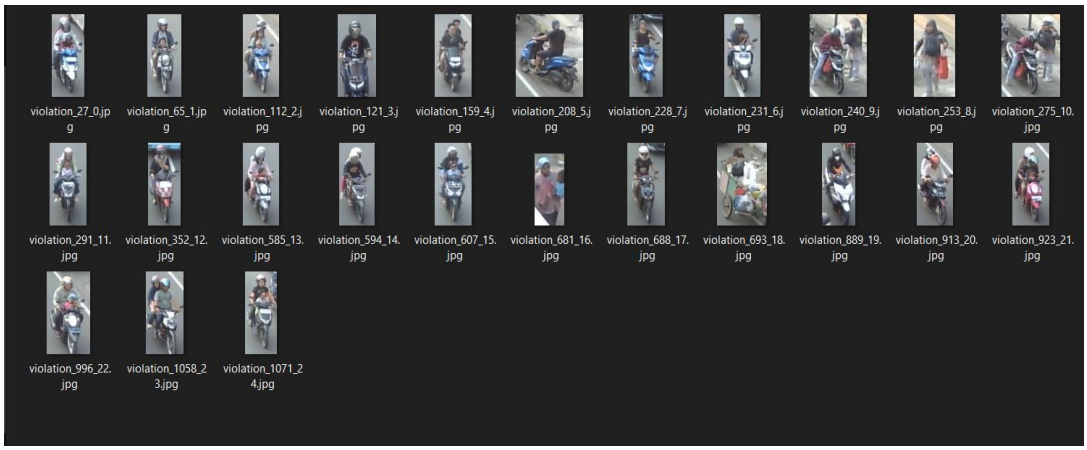
Pada Gambar 4.8, terlihat beberapa contoh kasus *false negative* yang terjadi selama pengujian langsung. Kasus *false negative* ini lebih sering melewatkan deteksi kelas ‘Rider’ dan ‘Tidak Helm’ dibandingkan dengan kelas ‘Helm’. Pada pengujian siang dan sore, kurangnya deteksi *bounding box* terjadi pada ketiga kelas saat siang hari, dan lebih sering melewatkan kelas ‘Rider’ saat cahaya redup di sore hari. Hal ini disebabkan oleh intensitas cahaya yang terlalu tinggi pada siang hari, sehingga *webcam* kesulitan mendeteksi objek dengan resolusi yang ada, serta cahaya yang redup pada sore hari yang menyebabkan lampu motor menyala, mengakibatkan *blur* ketika objek bergerak cepat. Kasus *false negative* juga muncul ketika pengendara bergerak terlalu cepat atau mengebut.

Penggunaan ByteTrack untuk mengurangi pengulangan atau duplikasi berlebih dalam hasil *cropping* pengendara motor terbukti cukup efisien dan efektif, meskipun terdapat sedikit *error* yang mungkin disebabkan oleh model itu sendiri. Tanpa sistem *tracking*, isi dari *folder* 'violation' terlihat seperti pada Gambar 4.9 berikut.



Gambar 4.9 Hasil *Cropping* tanpa Sistem *Tracking*

Pada Gambar 4.9, ditampilkan *folder* 'violation' yang berisi hasil *cropping* tanpa sistem tracking. Terlihat banyak duplikasi gambar dari objek yang sama, terutama ketika kelas 'Tidak Helm' bertumpang tindih dengan kelas 'Rider', menghasilkan sebanyak 218 gambar dalam satu *folder*. Berikut adalah tampilan folder 'violation' dengan sistem tracking ByteTrack pada Gambar 4.10.



Gambar 4.10 Hasil *Cropping* dengan menggunakan Sistem *Tracking*

Pada Gambar 4.10, ditampilkan *folder* 'violation' yang berisi hasil *cropping* dengan sistem *tracking* yang diintegrasikan. Terlihat bahwa duplikasi gambar dari objek yang sama berkurang secara signifikan ketika kelas 'Tidak Helm' bertumpang tindih dengan kelas 'Rider', sehingga hanya menghasilkan 25 gambar saja dalam satu folder.

Performa sistem juga bisa dipengaruhi oleh komputasi sistem, terutama saat menggunakan *edge computing* dengan mikrokomputer Jetson Nano. Saat melakukan pengujian langsung, terkadang FPS yang dihitung berdasarkan waktu inferensi tidak selalu sesuai dengan apa yang terlihat secara *real-time* di layar monitor. Hal ini mungkin disebabkan oleh beberapa faktor, termasuk proses pengujian yang dilakukan melalui *remote access*. Penggunaan *remote access* menyebabkan GUI Jetson Nano ditampilkan pada laptop, yang seringkali mengalami latensi dan membebani *bandwidth*, terutama ketika data harus dikirim bolak-balik antara Jetson Nano dan perangkat *remote*. Selain itu, pemrosesan *live feed* dan penggunaan model AI juga dapat memengaruhi komputasi secara keseluruhan, menyebabkan pemrosesan yang intensif pada komponen Jetson Nano. Akibatnya, suhu komponen dapat meningkat, dan ini dapat memicu *thermal throttling*. Di lingkungan luar ruangan, di mana suhu udara mungkin lebih tinggi, risiko *thermal throttling* bisa lebih besar. Notifikasi "*System throttled due to over-current*" muncul ketika Jetson Nano mendeteksi aliran listrik yang melampaui kapasitas maksimum yang dapat ditangani oleh sistem. Hal ini dapat terjadi jika Jetson Nano menggunakan perangkat tambahan seperti perangkat USB, terutama jika perangakt USB tersebut memiliki konsumsi daya yang tinggi, sehingga dapat menyebabkan tidak konsistennya total daya yang dikonsumsi oleh sistem secara keseluruhan.

BAB V
PENUTUP

5.1 Kesimpulan

Dari hasil penelitian yang telah diuraikan sebelumnya, dapat disimpulkan seperti berikut:

1. Sistem deteksi helm pada pengendara sepeda motor yang menggunakan model *deep learning* berbasis algoritma YOLOv8 mampu mendeteksi helm secara *real-time* dengan menginferensikan model YOLOv8 menggunakan Jetson Nano dan *webcam* pada jalan raya, serta melakukan proses *cropping* otomatis terhadap pelanggar yang tidak mengenakan helm.
2. Sistem deteksi helm pada pengendara sepeda motor yang menggunakan model *deep learning* berbasis algoritma YOLOv8 mencapai performa akurasi yang tinggi dalam ketiga pengujian langsung dengan mencatat nilai f_1*-score* sebesar 91.1% untuk kelas 'Helm', 81.7% untuk kelas 'Rider', dan f_1*-score* terendah 33.0% untuk kelas 'Tidak Helm'.
3. Sistem deteksi helm pada pengendara sepeda motor dengan menggunakan algoritma YOLOv8 berbasis *deep learning* pada perangkat Jetson Nano dapat dilakukan, dengan penggunaan CPU rata-rata 78,0%, penggunaan RAM rata-rata 77,4%, suhu komponen kisaran rentang antara 33°C hingga 65°C, daya total pemakaian rata-rata 6.5 W, penggunaan GPU yang tidak konsisten (dari 0.1% hingga 99%), serta FPS berkisar di rata-rata 11.

5.2 Saran

Berikut adalah beberapa saran yang dapat diberikan berdasarkan hasil penelitian yang telah dilakukan:

1. Penggunaan *line detector* meningkatkan efektivitas dan efisiensi sistem, khususnya di dalam *region of interest* (ROI).
2. Implementasi *edge computing* dan penggunaan *webcam* yang lebih baik bertujuan mengatasi beban komputasi sistem.
3. Dataset yang lebih variatif dan representatif diperlukan untuk meningkatkan kualitas model.

DAFTAR PUSTAKA

[1] H. Akbar and S. A. Kamaruddin, "Faktor-Faktor yang Berhubungan dengan Perilaku Tidak Aman pada Pengendara Ojek dan Becak Motor di Kota Kotamobagu Factors Related to Unsafe Behavior Among Motorcycle Taxi and Auto Rickshaw Drivers in the City of Kotamobagu," *PROMOTIF: Jurnal Kesehatan Masyarakat*, vol. 12, no. 1, pp. 36–42, 2022, doi: 10.56338/pjkm.v12i1.2443.

[2] F. Lestari, L. Febria Lina, N. D. Puspaningtyas, and I. Cahya Pratama, "PENINGKATAN PENGETAHUAN PATUH BERLALU LINTAS DAN BERKENDARA AMAN PADA SISWA SMA 1 NATAR," *Journal of Technology and Social for Community Service (JTSCS)*, vol. 3, no. 2, pp. 249–253, 2022, doi: 10.33365/jsstcs.v3i2.2118.

[3] L. A. Rizqandini, D. Lintang Trenggonowati, and L. Lady, "EFEK USIA, PENGALAMAN BERKENDARA, DAN TINGKAT KECELAKAAN TERHADAP DRIVER BEHAVIOR PENGENDARA SEPEDA MOTOR," *J Teknol*, vol. 12, no. 1, pp. 57–64, 2020, doi: 10.24853/jurtek.12.1.57-64.

[4] M. Tabary *et al.*, "The effectiveness of different types of motorcycle helmets – A scoping review," *Accid Anal Prev*, vol. 154, May 2021, doi: 10.1016/j.aap.2021.106065.

[5] M. Yulianingsih and Fridino, "IMPLEMENTASI UU NO. 22 TAHUN 2009 YANG BERKAITAN DENGAN PENGGUNAAN HELM SNI DI KECAMATAN TEBAS," *Jurnal Pendidikan Kewarganegaraan*, vol. 3, no. 2, pp. 111–120, 2019, doi: 10.31571/pkn.v3i2.1434.

[6] B. Sugandi and S. Lifitri, "Deteksi Pelanggaran Lampu Lalu Lintas Berdasarkan Sensor Visual," *JST (Jurnal Sains dan Teknologi)*, vol. 11, no. 2, pp. 315–323, Aug. 2022, doi: 10.23887/jstundiksha.v11i2.50287.

[7] A. S. Nugroho, "ELECTRONIC TRAFFIC LAW ENFORCEMENT (ETLE) MOBILE SEBAGAI DIFUSI INOVASI, INTEROPERABILITAS MENUJU ETLE NASIONAL (STUDI IMPLEMENTASI ETLE MOBILE

DI WILAYAH PROPINSI JAWA TENGAH)," *Jurnal Ilmu Kepolisian*, vol. 16, no. 3, pp. 157–176, 2022, doi: 10.35879/jik.v16i3.358.

[8] A. R. Admoko and Supriyadi, "Penerapan Sanksi Denda Tilang Elektronik Traffic Law Enforcement (E-TLE) Berdasarkan Undang-Undang No. 22 Tahun 2009 Tentang Lalu Lintas dan Angkutan Jalan," *MLJ Merdeka Law Journal*, vol. 3, no. 2, pp. 148–156, 2022, doi: 10.26905/mlj.v3i2.9220.

[9] F. Nurany, A. Nasya Damayanti, F. Aetika Wulandari, F. Nuzul Furqonia, A. Sulthon AHK, and U. Bhayangkara Surabaya, "KUALITAS PELAYANAN PUBLIK PADA LAYANAN E-TILANG SURABAYA," *Jurnal Aplikasi Administrasi*, vol. 24, no. 1, pp. 9–22, 2021, doi: 10.30649/aamama.v24i1.51.

[10] T. Sutrisna, "Kelemahan E-TLE, dari Belum Bisa Ciduk Pengendara Tak Pakai Helm hingga Salah Tilang," KOMPAS.com. Accessed: May 09, 2023. [Online]. Available: https://megapolitan.kompas.com/read/2022/11/12/12001381/kelemahan-e-tle-dari-belum-bisa-ciduk-pengendara-tak-pakai-helm-hingga

[11] T. Sutrisna, "Polda Metro: ETLE Belum Bisa Tilang Pengendara Tak Pakai Helm dan Kendaraan Kelebihan Penumpang ," KOMPAS.com. Accessed: May 09, 2023. [Online]. Available: https://megapolitan.kompas.com/read/2022/11/11/16550771/polda-metro-etle-belum-bisa-tilang-pengendara-tak-pakai-helm-dan

[12] K. Adi, C. E. Widodo, and A. Pujiwidodo, "DESIGN AND IMPLEMENTATION OF TRAFFIC VIOLATION DETECTION SYSTEMS WITH DEEP LEARNING TO SUPPORT ELECTRONIC TRAFFIC LAW ENFORCEMENT (e-TLE)," *ARPN Journal of Engineering and Applied Sciences*, vol. 16, no. 10, pp. 1062–1070, 2021.

[13] J. Chai, H. Zeng, A. Li, and E. W. T. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, pp. 1–13, 2021, doi: 10.24433/CO.0411648.v1.

[14] Z.-Q. Zhao, P. Zheng, S. Xu, and X. Wu, “Object Detection with Deep Learning: A Review,” *IEEE Trans Neural Netw Learn Syst*, vol. 30, no. 11, pp. 3212–3232, Jul. 2019, doi: 10.1109/TNNLS.2018.2876865.

[15] S. Srivastava, A. V. Divekar, C. Anilkumar, I. Naik, V. Kulkarni, and V. Pattabiraman, “Comparative analysis of deep learning image detection algorithms,” *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00434-w.

[16] D. Bhabad, S. Kadam, T. Malode, G. Shinde, and D. Bage, “Object Detection for Night Vision using Deep Learning Algorithms,” *International Journal of Computer Trends and Technology*, vol. 71, no. 2, pp. 87–92, Feb. 2023, doi: 10.14445/22312803/IJCTT-V71I2P113.

[17] S. Wu, Z. Li, S. Li, Q. Liu, and W. Wu, “Static Gesture Recognition Algorithm Based on Improved YOLOv5s,” *Electronics (Switzerland)*, vol. 12, no. 3, Feb. 2023, doi: 10.3390/electronics12030596.

[18] A. A. Mirza *et al.*, “The Use of Artificial Intelligence in Medical Imaging: A Nationwide Pilot Survey of Trainees in Saudi Arabia,” *Clin Pract*, vol. 12, no. 6, pp. 852–866, Dec. 2022, doi: 10.3390/clinpract12060090.

[19] Y. Xu *et al.*, “Artificial intelligence: A powerful paradigm for scientific research,” *The Innovation*, vol. 2, no. 4. Cell Press, Nov. 28, 2021. doi: 10.1016/j.xinn.2021.100179.

[20] I. H. Sarker, “Machine Learning: Algorithms, Real-World Applications and Research Directions,” *SN Comput Sci*, vol. 2, no. 3, May 2021, doi: 10.1007/s42979-021-00592-x.

[21] M. Abdullah-Al-Noman, A. N. Eva, T. B. Yeahyea, and R. Khan, “Computer Vision-based Robotic Arm for Object Color, Shape, and Size Detection,” *Journal of Robotics and Control (JRC)*, vol. 3, no. 2, pp. 180–186, Mar. 2022, doi: 10.18196/jrc.v3i2.13906.

[22] J. Peddie, E. Fonseka, K. Akeley, M. Mangan, P. Debevec, and M. Raphael, “A vision for computer vision: Emerging technologies,” in *ACM SIGGRAPH 2016 Panels, SIGGRAPH 2016*, Association for Computing Machinery, Inc, Jul. 2016. doi: 10.1145/2927383.2933233.

[23] A. I. Khan and S. Al-Habsi, “Machine Learning in Computer Vision,” in *Procedia Computer Science*, Elsevier B.V., 2020, pp. 1444–1451. doi: 10.1016/j.procs.2020.03.355.

[24] K. L. Masita, A. N. Hasan, and T. Shongwe, “Deep Learning in Object Detection: a Review,” in *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, 2020, pp. 1–11. doi: DOI:10.1109/icABCD49160.2020.9183866.

[25] Y. Xiao *et al.*, “A review of object detection based on deep learning,” *Multimed Tools Appl*, vol. 79, no. 33–34, pp. 23729–23791, Sep. 2020, doi: 10.1007/s11042-020-08976-6.

[26] J. Kaur and W. Singh, “Tools, techniques, datasets and application areas for object detection in an image: a review,” *Multimed Tools Appl*, vol. 81, no. 27, pp. 38297–38351, Nov. 2022, doi: 10.1007/s11042-022-13153-y.

[27] B. Mirzaei, H. Nezamabadi-pour, A. Raoof, and R. Derakhshani, “Small Object Detection and Tracking: A Comprehensive Review,” *Sensors*, vol. 23, no. 15, Aug. 2023, doi: 10.3390/s23156887.

[28] M. Abouelyazid, “Comparative Evaluation of SORT, DeepSORT, and ByteTrack for Multiple Object Tracking in Highway Videos,” *nternational Journal of Sustainable Infrastructure for Cities and Societies*, vol. 8, no. 11, pp. 42–52, 2023.

[29] V. A. Golovko, “Deep learning: an overview and main paradigms,” *Optical Memory and Neural Networks (Information Optics)*, vol. 26, no. 1, pp. 1–17, Jan. 2017, doi: 10.3103/S1060992X16040081.

[30] L. Alzubaidi *et al.*, “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[31] T. Diwan, G. Anirudh, and J. V. Tembhurne, “Object detection using YOLO: challenges, architectural successors, datasets and applications,” *Multimed Tools Appl*, vol. 82, no. 6, pp. 9243–9275, Mar. 2023, doi: 10.1007/s11042-022-13644-y.

[32] J. Terven and D. Cordova-Esparza, “A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond,” Apr. 2023, doi: 10.48550/arXiv.2304.00501.

[33] M. Lalinia and A. Sahafi, “Colorectal polyp detection in colonoscopy images using YOLO-V8 network,” *Signal Image Video Process*, vol. 18, no. 3, pp. 2047–2058, Apr. 2024, doi: 10.1007/s11760-023-02835-1.

[34] Z. Huang, L. Li, G. C. Krizek, and L. Sun, “Research on Traffic Sign Detection Based on Improved YOLOv8,” *Journal of Computer and Communications*, vol. 11, no. 07, pp. 226–232, 2023, doi: 10.4236/jcc.2023.117014.

[35] “results - Ultralytics YOLOv8 Docs.” Accessed: May 17, 2024. [Online]. Available: https://docs.ultralytics.com/reference/engine/results/

[36] Provost James, “NVIDIA MAKES IT EASY TO EMBED AITHE JETSON NANO PACKS A LOT OF MACHINE-LEARNING POWER INTO DIY PROJECTS,” 2020. doi: 10.1109/MSPEC.2020.9126102.

[37] Z. Deng, C. Yao, and Q. Yin, “Safety Helmet Wearing Detection Based on Jetson Nano and Improved YOLOv5,” *Advances in Civil Engineering*, vol. 2023, pp. 1–12, May 2023, doi: 10.1155/2023/1959962.

[38] M. Oliveira Jr., G. Sedrez, G. de Souza, and G. Cavalheiro, “An Application with Jetson Nano for Plant Stress Detection and On-field Spray Decision,” Scitepress, Feb. 2022, pp. 215–222. doi: 10.5220/0010983900003118.

[39] W. Rahmaniar and A. Hernawan, “Real-time human detection using deep learning on embedded platforms: A review,” *Journal of Robotics and Control (JRC)*, vol. 2, no. 6. Department of Agribusiness, Universitas Muhammadiyah Yogyakarta, pp. 462-468Y, Nov. 01, 2021. doi: 10.18196/jrc.26123.

[40] A. Chairat, M. N. Dailey, and D. Raj, “Low Cost, High Performance Automatic Motorcycle Helmet Violation Detection,” in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 3560–3568. doi: 10.1109/WACV45572.2020.9093538.

[41] M. Luqman Bukhori and E. E. Prasetiyo, “Jetson Nano-Based Mask Detection System with TensorFlow Deep Learning Framework,” *JURNAL NASIONAL TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI*, vol. 12, no. 1, pp. 15–21, 2022, doi: 10.22146/jnteti.v12i1.5472.

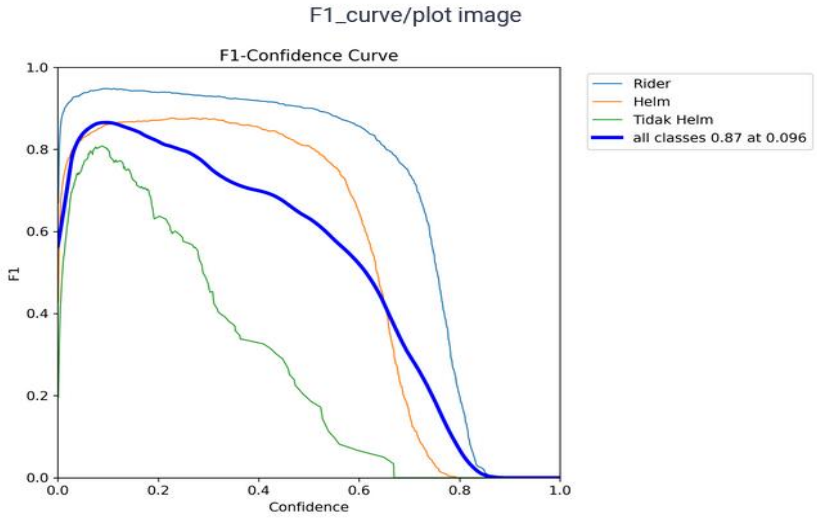
[42] K. Al-Nujaidi, G. Al-Habib, and A. Al-Odhieb, “Spot-the-Camel: Computer Vision for Safer Roads,” *International Journal of Artificial Intelligence & Applications*, vol. 14, no. 2, pp. 1–10, Mar. 2023, doi: 10.5121/ijaia.2023.14201.

[43] G. Agorku *et al.*, “Real-Time Helmet Violation Detection Using YOLOv5 and Ensemble Learning,” *arXiv e-prints*, Apr. 2023, doi: 10.48550/arXiv.2304.09246.

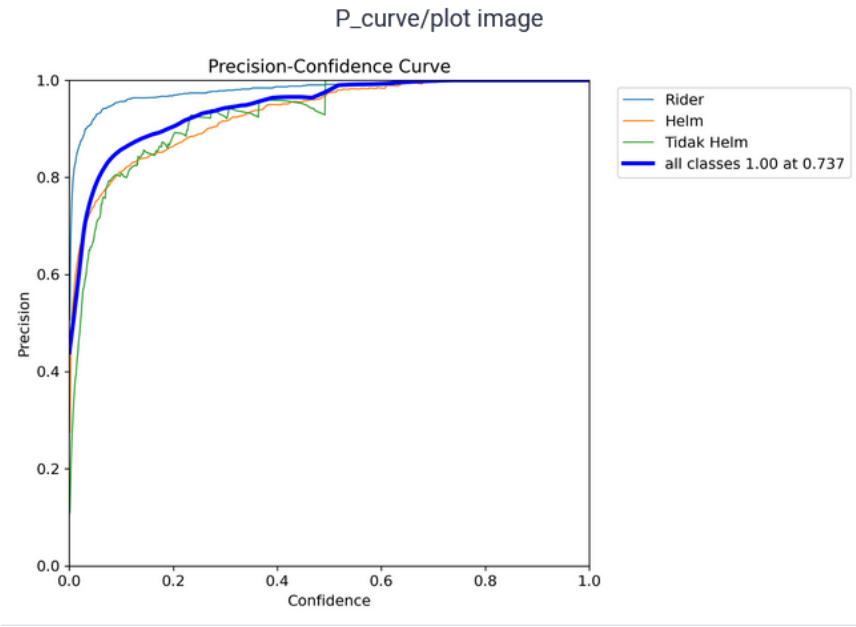
LAMPIRAN A *OUTPUT* GRAFIK MODEL

2023-12-05 19:00:17	all 240 1880 0.779 0.614 0.701 0.382
2023-12-05 19:00:18	50 epochs completed in 3.202 hours.
2023-12-05 19:00:19	Optimizer stripped from runs/detect/train2/weights/last.pt, 6.2MB Optimizer stripped from runs/detect/train2/weights/best.pt, 6.2MB Validating runs/detect/train2/weights/best.pt... Ultralytics YOLOv8.0.222 🚀 Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB) Model summary (fused): 168 layers, 3006233 parameters, 0 gradients, 8.1 GFLOPs
2023-12-05 19:00:25	Class Images Instances Box(P R mAP50 mAP50-95): 38%| | 3/8 [00:05<00:10, 2.(
2023-12-05 19:00:31	Class Images Instances Box(P R mAP50 mAP50-95): 100%| | 8/8 [00:10<00:00, 1.3
2023-12-05 19:00:35	all 240 1880 0.859 0.879 0.904 0.505 Rider 240 864 0.956 0.938 0.977 0.68 Helm 240 899 0.813 0.915 0.927 0.462 Tidak Helm 240 117 0.806 0.783 0.807 0.373
2023-12-05 19:00:36	Speed: 0.2ms preprocess, 2.5ms inference, 0.0ms loss, 4.6ms postprocess per image Results saved to **runs/detect/train2**
2023-12-05 19:00:57	2023-12-05 12:00:57,248 - clearml.Task - INFO - Completed model upload to https://files.clear.ml/YOLOv8/train2.6392ced696d94e8882

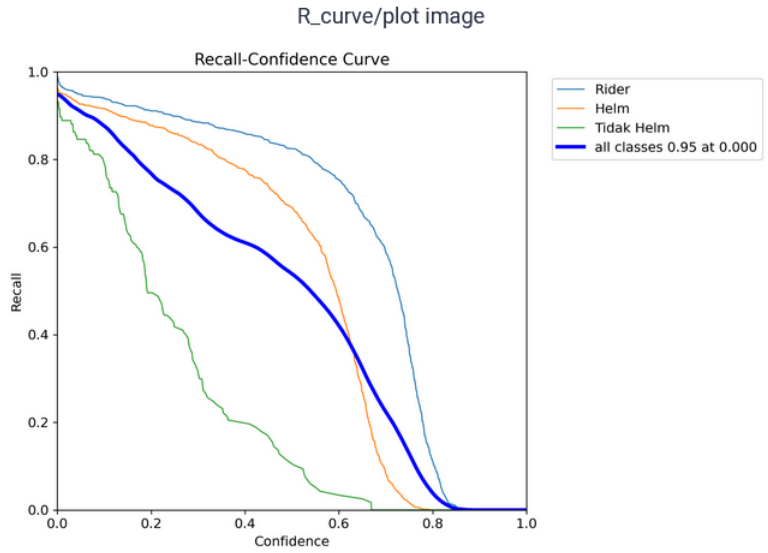
Gambar A.1 Output Model Terbaik pada ClearML



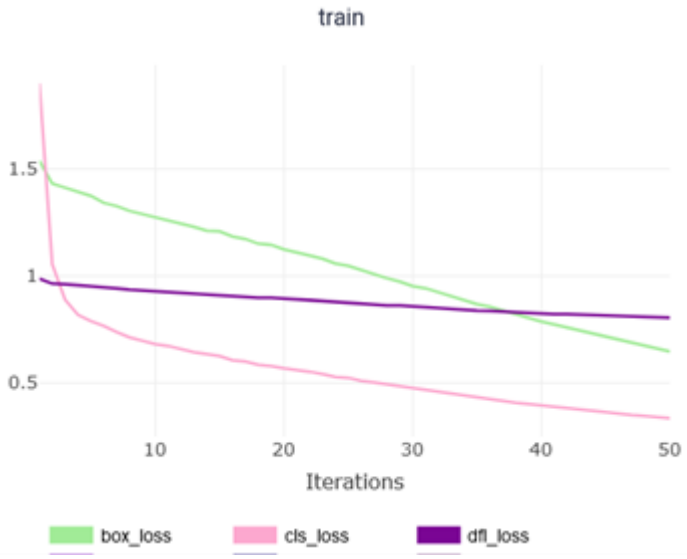
Gambar A.2 Kurva Kepercayaan *F_1-score* Model *Training*



Gambar A.3 Kurva Kepercayaan *Precision* Model *Training*



Gambar A.4 Kurva Kepercayaan *Recall* Model *Training*



Gambar A.5 Kurva Kerugian Model *Training*

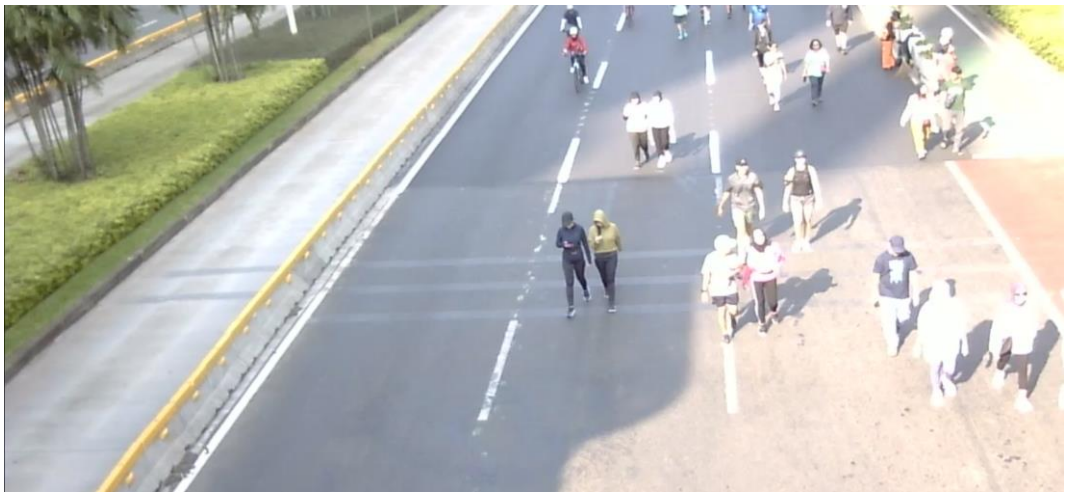
LAMPIRAN B SAMPEL DATA

Gambar B.1 Sampel *Frame* untuk Video Keperluan Dataset *Training* 1

Gambar B.2 Sampel *Frame* untuk Video Keperluan Dataset *Training* 2

Gambar B.3 Sampel *Frame* untuk Video Keperluan Dataset *Training* 3

Gambar B.4 Sampel *Frame* untuk Video Keperluan Dataset *Training* 4



Gambar B.5 Sampel *Frame* untuk Video Keperluan Dataset *Training* 5

Gambar B.6 Sampel *Frame* untuk Video Keperluan Dataset *Training* 6

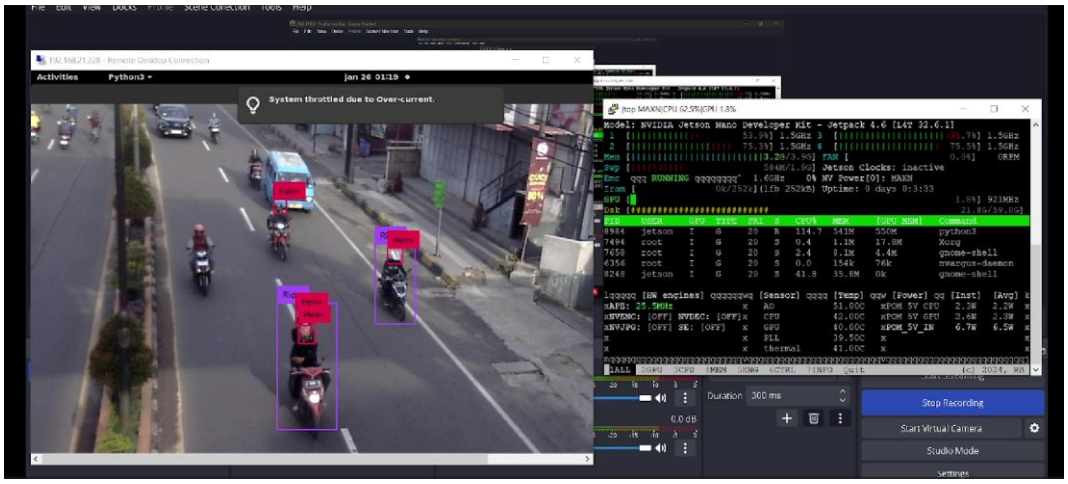
Gambar B.7 Sampel *Frame* untuk Video Keperluan Dataset *Training* 7

Gambar B.8 Sampel *Frame* untuk Video Keperluan Dataset Validasi

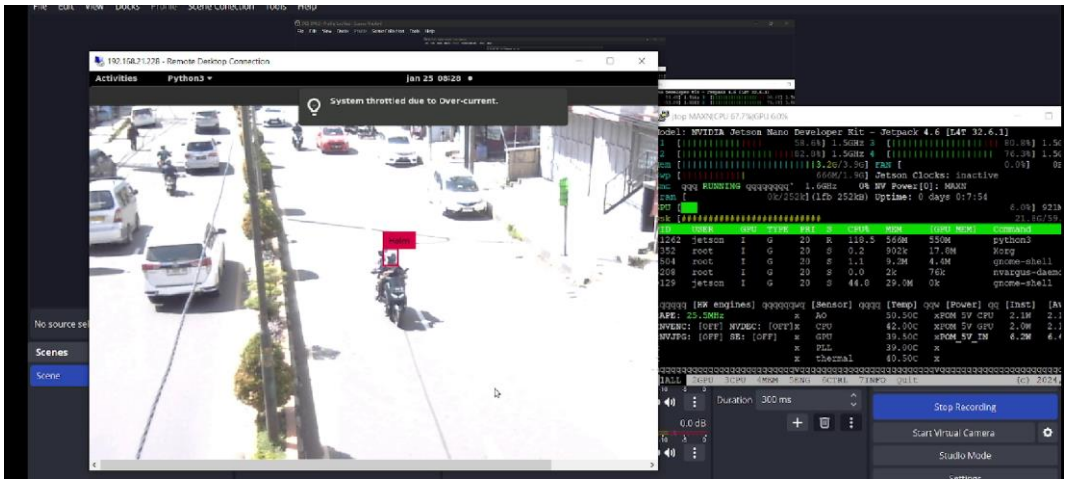
Gambar B.9 Sampel *Frame* untuk Video Keperluan *Testing* Sore

Gambar B.10 Sampel *Frame* untuk Video Keperluan *Testing* Siang

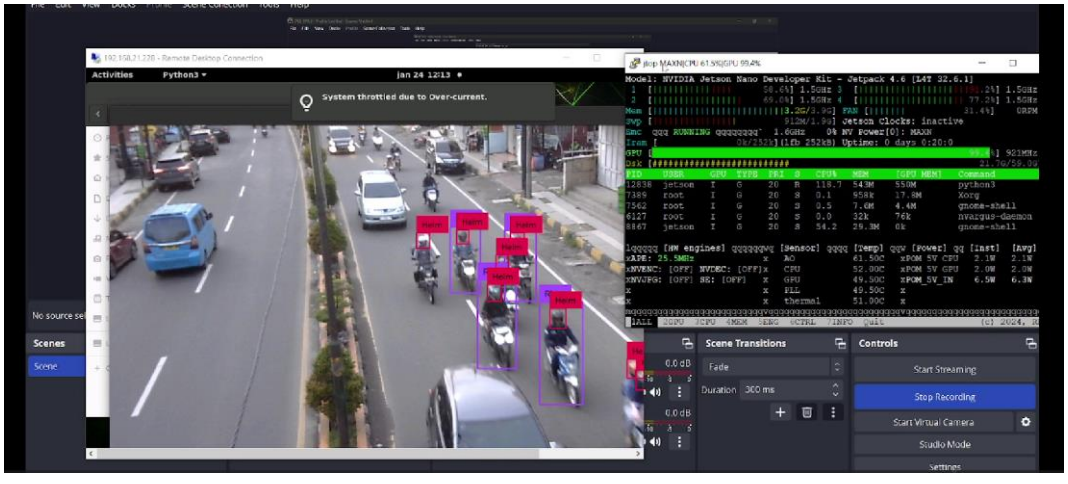
Gambar B.11 Sampel *Frame* untuk Video Keperluan *Testing* Pagi



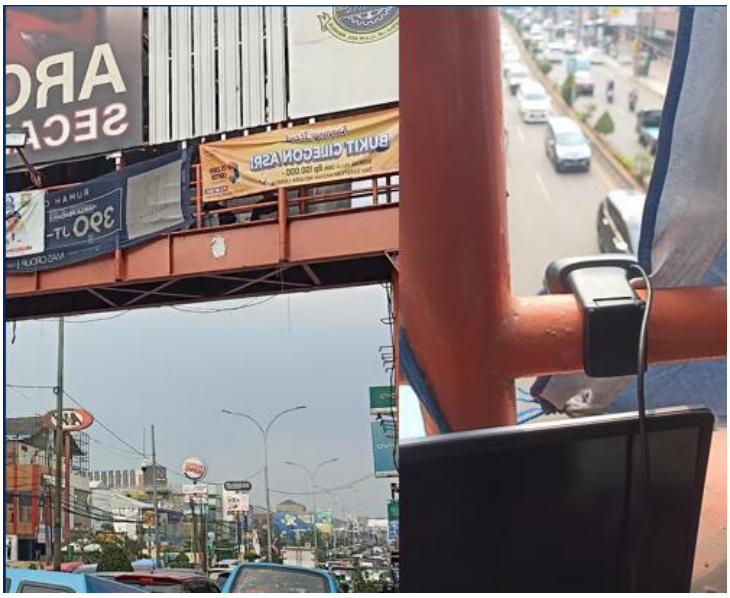
Gambar B.12 Sampel Tampilan Penujian *Live Feed* Pagi



Gambar B.13 Sampel Tampilan Penujian *Live Feed* Siang



Gambar B.14 Sampel Tampilan Penujian *Live Feed* Sore



Gambar B.15 Posisi Kamera dan Penempatan Sistem

LAMPIRAN C KONFIGURASI PENGUJIAN

1. Melakukan *remote access* antara *laptop* dan Jetson Nano
2. Memindahkan *folder* Sistem Deteksi Helm YOLOv8 bernama 'Yolov8-Helmet-Detection' ke OS Linux Jetson Nano melalui *remote access* yang diakses menggunakan SSHFS dan PuTTY.
3. Menjalankan Terminal pada GUI OS Linux pada NVIDIA Jetson Nano.
4. Lalu install model YOLOv8 satu paket dengan *library* yang dibutuhkan pada Terminal OS Linux Jetson Nano dengan menggunakan perintah berikut.

```
pip install ultralytics
```

5. Lalu install model *tracking ByteTrack* pada Terminal OS Linux Jetson Nano dengan menggunakan perintah berikut.

```
pip install supervision
```

6. Menyesuaikan *working directory* dengan *folder* sistem yang telah dipindahkan dengan menggunakan perintah berikut pada terminal.

```
cd Yolov8-Helmet-Detection
```

7. Setelah semua pengaturan selesai, lalu menjalankan file *testbtlive.py* untuk menjalankan *live feed* dengan perintah berikut.

```
python3 testbtlive.py
```