BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Berikut adalah penelitian-penelitian terdahulu yang relevan dan memiliki keterkaitan dengan topik yang diangkat dalam penelitian ini. Penelitian-penelitian tersebut membahas penerapan algoritma YOLO dalam deteksi kendaraan, vehicle counting, vehicle tracking, hingga sistem monitoring lalu lintas berbasis computer vision. Ringkasan penelitian terdahulu ditunjukkan pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No	Penelitian	Metode	Hasil
1	*R. Mahadshetti, J. Kim, and T. W. Um, "Sign-YOLO: Traffic Sign Detection Using Attention-Based YOLOv7," IEEE Access, vol. 12, pp. 132689–132700, 2024.*	*Attention-Based YOLOv7*	Mencapai mAP 99.10% dan mengurangi ukuran model hingga 98%, sangat efisien untuk *real-time*. [18]
2	*F. M. Talaat, R. M. El-Balka, S. Sweidan, S. A. Gamel, and A. M. Al-Zoghby, "Smart traffic management system using YOLOv11 for real-time vehicle detection and dynamic flow optimization in smart cities," Neural Comput. Appl., vol. 37, no. 24, pp. 19957–19974, 2025.*	*YOLOv11*	Akurasi mAP 92.4% dan IoU 0.85, efektif untuk optimasi lalu lintas *real-time*. [19]
3	*J. Azimjonov and A. Özmen, "A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways," Advanced Engineering Informatics, vol. 50, 2021*	*YOLO + Tracking*	Akurasi meningkat hingga 95.45% dengan kombinasi classifier. [20]
4	*M. A. A. Al-qaness, A. A. Abbasi, H. Fan, R. A. Ibrahim, S. H. Alsamhi, and A. Hawbani, "An improved YOLO-based road traffic monitoring system," Computing, vol. 103, no. 2, pp. 211–230, 2021.*	*YOLOv3*	Sistem mampu mendeteksi, melacak, dan menghitung kendaraan secara stabil dalam berbagai kondisi. [21]
5	*M. A. Mohd Nazri, F. Samsuri, and V. Narayanamurthy, "Design and Analysis of Visual Vehicle Tracking System for Traffic Surveillance Using Deep learning," International Journal of Intelligent Transportation Systems Research, vol. 23, no. 1, pp. 592–602, 2025.*	*YOLOv8*	*Precision* 0.85, *Recall* 0.82, mAP 0.80, lebih unggul dari YOLO-NAS. [22]
6	*M. Bakirci, "Utilizing YOLOv8 for enhanced traffic monitoring in Intelligent Transportation Systems (ITS) applications," Digital Signal Processing: A Review Journal, vol. 152, 2024.*	*YOLOv8*	Meningkatkan *Precision* hingga 18% dibanding YOLOv5 serta lebih cepat dalam inference. [23]
7	*Mohanapriya, K.; Shankar, R.; Duraisamy, S. (2025), "Vehicle Density Estimation Using Improved Yolov8 Object Detection Model with Kernel Density Estimation", CCIS Springer.*	*YOLOv8 + Kernel Density Estimation*	Meningkatkan akurasi estimasi kepadatan dan performa sistem *real-time*. [15]
8	*M. Majumder and C. Wilmot, "Automated Vehicle Counting from Pre-Recorded Video Using You Only Look Once (YOLO) Object Detection Model," J. Imaging, vol. 9, no. 7, 2023, doi: 10.3390/jimaging9070131.*	*YOLO*	Mencapai akurasi sekitar 90% pada proses vehicle counting dari video dan menunjukkan bahwa kualitas video memengaruhi hasil perhitungan kendaraan. [24]

9	*D. Shokri, C. Larouche, and S. Homayouni, "A Comparative Analysis of Multi-Label Deep Learning Classifiers for Real-Time Vehicle Detection to Support Intelligent Transportation Systems," Smart Cities, vol. 6, no. 5, pp. 2982–3004, 2023, doi: 10.3390/smartcities6050134.*	*SSD, RCNN, YOLOv5, YOLOv6, YOLOv7, YOLOv8*	Menunjukkan bahwa YOLO memberikan performa deteksi kendaraan yang lebih baik dibandingkan SSD dan RCNN. YOLOv8 memiliki waktu inferensi yang lebih cepat, sedangkan YOLOv7 memberikan akurasi yang lebih tinggi. [25]
10	*C. H. Lim, T. Connie, and K. O. M. Goh, "Vehicle Detection Based on Improved YOLOv8," in 6th IEEE International Conference on Artificial Intelligence in Engineering and Technology, IICAIET 2024, 2024.*	*YOLOv8*	Mengidentifikasi masalah class imbalance pada *dataset* lokal yang mempengaruhi performa awal model.[26]

Berdasarkan Tabel 2.1, perkembangan penelitian pada bidang *computer vision* menunjukkan bahwa algoritma YOLO telah banyak dimanfaatkan untuk mendukung sistem *monitoring* lalu lintas secara otomatis. Berbagai penelitian berhasil menerapkan YOLO untuk mendeteksi kendaraan, mengklasifikasikan jenis kendaraan, menghitung jumlah kendaraan (*vehicle counting*), hingga melakukan pelacakan (*vehicle tracking*) pada video lalu lintas secara *real-time* [20], [21], [22]. Salah satu algoritma yang banyak digunakan dalam penelitian tersebut adalah YOLO (*You Only Look Once*), yang dikenal memiliki keunggulan dalam kecepatan dan akurasi sehingga sangat cocok untuk aplikasi *real-time* seperti sistem *monitoring* lalu lintas berbasis CCTV [18], [23], [27], [28].

Penelitian sebelumnya menunjukkan bahwa algoritma YOLO telah banyak diterapkan dalam berbagai sistem transportasi cerdas (*Intelligent Transportation System*/ITS), khususnya dalam mendeteksi kendaraan serta mendukung proses *monitoring* lalu lintas secara *real-time* [19], [29]. Beberapa penelitian mengembangkan sistem manajemen lalu lintas cerdas yang mampu mendeteksi kendaraan secara *real-time* sekaligus mengoptimalkan aliran lalu lintas [21], [27], [30]. Sistem tersebut tidak hanya berfungsi sebagai alat deteksi, tetapi juga mampu mendukung proses *monitoring* lalu lintas serta membantu pengambilan keputusan secara lebih efektif.

Selain itu, terdapat penelitian yang mengombinasikan algoritma YOLO dengan metode *tracking* untuk meningkatkan akurasi dalam menghitung jumlah kendaraan [20], [24], [31]. Dengan memanfaatkan teknik pelacakan objek, sistem dapat mengurangi kesalahan perhitungan akibat deteksi ganda maupun objek yang hilang sehingga menghasilkan data yang lebih akurat dan konsisten dalam analisis lalu

lintas. Pendekatan ini menunjukkan bahwa integrasi antara deteksi dan pelacakan objek menjadi solusi yang efektif dalam meningkatkan performa sistem *monitoring* lalu lintas.

Penggunaan algoritma YOLO dalam sistem *monitoring* berbasis video juga telah menunjukkan performa yang baik dalam mendeteksi, melacak, dan menghitung kendaraan pada berbagai kondisi lingkungan [24], [32], [33]. Hal ini menunjukkan bahwa YOLO memiliki kemampuan generalisasi yang cukup baik dan dapat diimplementasikan pada sistem berbasis CCTV untuk kebutuhan pemantauan lalu lintas secara *real-time*. Namun demikian, performa sistem masih dipengaruhi oleh beberapa faktor eksternal, seperti kondisi pencahayaan, resolusi video, sudut pengambilan gambar, serta kepadatan objek dalam satu frame.

Perkembangan terbaru dalam algoritma YOLO menunjukkan bahwa versi YOLOv8 memiliki peningkatan performa dibandingkan versi sebelumnya. Selain itu, hasil perbandingan dengan algoritma deteksi objek lain seperti SSD dan RCNN menunjukkan bahwa keluarga YOLO memiliki performa yang lebih baik untuk deteksi kendaraan, sedangkan YOLOv8 menawarkan keunggulan dalam hal kecepatan inferensi dibandingkan beberapa versi YOLO sebelumnya [25], [34]. Beberapa penelitian menunjukkan bahwa YOLOv8 memiliki peningkatan akurasi deteksi, kecepatan inferensi, serta kemampuan mendeteksi objek kecil sehingga lebih optimal digunakan pada sistem deteksi kendaraan berbasis *real-time* [22], [26].

Selain deteksi kendaraan, beberapa penelitian juga mengembangkan metode tambahan, seperti *Kernel Density Estimation* (KDE), untuk menganalisis distribusi kendaraan pada suatu ruas jalan [15]. Pendekatan tersebut digunakan untuk memperoleh informasi yang lebih rinci mengenai pola lalu lintas, namun tidak menjadi fokus dalam penelitian ini.

Meskipun demikian, beberapa penelitian menunjukkan bahwa performa sistem deteksi kendaraan masih memiliki keterbatasan. Faktor seperti kualitas data video, kondisi lingkungan yang kompleks, serta objek yang saling berdekatan dapat menyebabkan kesalahan deteksi maupun penurunan akurasi model [26], [35], [36].

Hal ini menunjukkan bahwa kualitas data input dan proses pelatihan model memiliki peran yang sangat penting dalam menentukan keberhasilan sistem deteksi kendaraan.

Berdasarkan berbagai penelitian terdahulu tersebut, dapat disimpulkan bahwa algoritma YOLO, khususnya YOLOv8, memiliki performa yang sangat baik dalam mendeteksi kendaraan secara *real-time* serta mendukung proses *monitoring* lalu lintas berbasis CCTV. Namun, sebagian besar penelitian masih berfokus pada proses deteksi dan analisis saja tanpa mengintegrasikan hasil tersebut ke dalam sistem *monitoring* berbasis web yang interaktif dan mudah diakses oleh pengguna.

Berdasarkan penelitian-penelitian terdahulu yang telah dijelaskan, sebagian besar penelitian hanya berfokus pada proses deteksi kendaraan dan evaluasi performa model tanpa mengimplementasikan sistem *monitoring* berbasis web secara *real-time* [19], [20], [35]. Selain itu, beberapa penelitian masih menggunakan *dataset* umum dan belum memanfaatkan video streaming CCTV jalan tol secara langsung sebagai sumber input utama sistem [15], [26].

Penelitian sebelumnya umumnya masih berfokus pada peningkatan performa model *object detection* dan evaluasi metrik seperti *Precision*, *Recall*, serta *mAP* untuk deteksi kendaraan [15], [23]. Namun, sebagian besar penelitian tersebut belum menyediakan sistem *monitoring* lalu lintas berbasis web yang interaktif dan terintegrasi langsung dengan CCTV jalan tol secara *real-time*.

Oleh karena itu, penelitian ini dilakukan untuk mengisi *research gap* tersebut dengan mengembangkan sistem *monitoring* lalu lintas berbasis CCTV secara *real-time* menggunakan YOLOv8m yang diimplementasikan pada aplikasi berbasis web menggunakan *Streamlit*. Sistem yang dikembangkan tidak hanya melakukan deteksi kendaraan dan menghitung jumlah kendaraan secara otomatis (*vehicle counting*), tetapi juga menyajikan hasil deteksi dalam bentuk dashboard berbasis web yang interaktif, sehingga memudahkan proses *monitoring* lalu lintas dari berbagai titik CCTV secara *real-time*.

2.2 Tinjauan Teori

2.2.1 Kamera CCTV untuk *Monitoring* Lalu Lintas

CCTV (*Closed Circuit Television*) merupakan sistem kamera pengawasan yang digunakan untuk memantau suatu area secara terus-menerus melalui jaringan tertutup. Dalam bidang transportasi, CCTV telah menjadi salah satu komponen penting dalam sistem pemantauan lalu lintas karena mampu menyediakan data visual secara *real-time* yang dapat digunakan mengamati kondisi jalan, memantau arus kendaraan, serta mengidentifikasi kejadian tertentu seperti kecelakaan atau pelanggaran lalu lintas [37], [38].

Penggunaan CCTV dalam *monitoring* lalu lintas memiliki berbagai keunggulan dibandingkan metode konvensional. CCTV mampu memberikan informasi visual secara langsung tanpa memerlukan kehadiran petugas di lokasi serta dapat mencakup area pengamatan yang luas. Selain itu, sistem CCTV dapat beroperasi selama 24 jam secara kontinu sehingga memungkinkan pengawasan lalu lintas dilakukan secara berkelanjutan. Dengan perkembangan teknologi jaringan dan internet, video dari CCTV kini dapat diakses secara online melalui berbagai platform sehingga memudahkan pengguna maupun pihak pengelola dalam memantau kondisi lalu lintas dari berbagai lokasi [38].

Dalam implementasinya, CCTV merupakan bagian dari konsep *Intelligent Transportation System* (ITS), yaitu sistem transportasi yang memanfaatkan teknologi informasi dan komunikasi untuk meningkatkan efisiensi, keamanan, dan kenyamanan dalam pengelolaan lalu lintas. Data visual yang dihasilkan oleh CCTV dapat diproses lebih lanjut menggunakan teknologi *Computer vision* dan *Deep learning* untuk mendeteksi objek kendaraan, mengklasifikasikan jenis kendaraan, serta menghitung jumlah kendaraan secara otomatis [7], [39], [40].

Perkembangan teknologi *computer vision* memungkinkan sistem CCTV tidak hanya digunakan sebagai alat pemantauan visual, tetapi juga sebagai sumber data otomatis untuk analisis lalu lintas. Dengan memanfaatkan algoritma *object detection* seperti YOLO, sistem dapat mengenali berbagai jenis kendaraan pada video CCTV secara otomatis dan *real-time*. Proses tersebut

membantu mengurangi ketergantungan terhadap pengawasan manual oleh operator serta meningkatkan efisiensi proses *monitoring* lalu lintas.

Meskipun memiliki banyak keunggulan, penggunaan CCTV dalam *monitoring* lalu lintas juga memiliki beberapa keterbatasan. Kualitas data yang dihasilkan sangat dipengaruhi oleh kondisi lingkungan, seperti pencahayaan, cuaca, resolusi kamera, serta sudut pengambilan gambar. Selain itu, objek kendaraan yang saling berdekatan atau tertutup (*occlusion*) dapat menyebabkan kesalahan dalam proses deteksi. Gangguan jaringan internet pada proses video streaming juga dapat mempengaruhi stabilitas *monitoring* secara *real-time*. Oleh karena itu, diperlukan algoritma yang mampu mengatasi berbagai kondisi tersebut agar sistem *monitoring* dapat bekerja dengan lebih optimal [6], [41].

2.2.2 *Deep learning Monitoring* Lalu Lintas

Deep learning merupakan salah satu cabang dari *Machine Learning* yang memanfaatkan jaringan saraf tiruan (*Artificial Neural Network*) dengan banyak lapisan (*deep neural network*) untuk mempelajari pola data secara kompleks. Metode ini sangat efektif dalam mengolah data tidak terstruktur, seperti gambar dan video, karena proses ekraksi fitur dapat dilakukan secara otomatis tanpa memerlukan proses manual. Dalam beberapa tahun terakhir, *Deep learning* telah menjadi pendekatan utama dalam berbagai aplikasi *Computer vision*, termasuk dalam deteksi objek dan analisis video [42], [43].

Dalam konteks *monitoring* lalu lintas, *Deep learning* digunakan untuk mendeteksi, mengklasifikasikan, dan menghitung kendaraan secara otomatis dari data visual yang diperoleh melalui kamera CCTV. Dengan memanfaatkan model berbasis *Convolutional Neural Network* (CNN), sistem mampu mengenali berbagai jenis kendaraan berdasarkan pola visual yang dipelajari dari *dataset* pelatihan. Keunggulan utama dari pendekatan ini adalah kemampuannya dalam menangani variasi kondisi lingkungan, seperti perubahan pencahayaan, sudut pandang kamera, serta klasifikasi kepadatan lalu lintas. [44], [45].

Penerapan *Deep learning* dalam *monitoring* lalu lintas memberikan berbagai manfaat dibandingkan metode konvensional. Sistem berbasis *Deep learning* mampu bekerja secara *real-time* dengan tingkat akurasi yang tinggi sehingga dapat digunakan untuk mendukung pengambilan keputusan secara cepat. Selain itu, proses deteksi dan analisis yang dilakukan secara otomatis dapat mengurangi ketergantungan terhadap tenaga manusia serta meningkatkan efisiensi dalam proses pemantauan lalu lintas [46], [47].

Perkembangan teknologi *Deep learning* juga mendorong munculnya berbagai algoritma *object detection* yang lebih cepat dan akurat, salah satunya adalah YOLO (*You Only Look Once*)[28], [48]. Algoritma YOLO mampu melakukan proses deteksi objek dalam satu tahap (*single stage detector*) sehingga memiliki kecepatan inferensi yang tinggi dan sangat cocok digunakan untuk aplikasi *monitoring* lalu lintas berbasis video secara *real-time*. Dengan kemampuan tersebut, sistem dapat mendeteksi kendaraan secara langsung pada setiap frame video CCTV tanpa memerlukan proses komputasi yang terlalu kompleks.

Namun demikian, penggunaan *Deep learning* juga memiliki beberapa tantangan, seperti kebutuhan akan *dataset* yang besar dan beragam untuk melatih model, serta kebutuhan komputasi yang tinggi terutama pada tahap pelatihan model. Selain itu, performa model sangat dipengaruhi oleh kualitas data yang digunakan sehingga kondisi video yang kurang baik, seperti resolusi rendah, pencahayaan buruk, atau objek kendaraan yang saling bertumpuk (*occlusion*), dapat mempengaruhi akurasi deteksi [49].

2.2.3 Sistem *Monitoring* Lalu Lintas *Real-time*

Sistem *monitoring* lalu lintas *real-time* merupakan sistem yang dirancang untuk memantau kondisi lalu lintas secara langsung dan berkelanjutan dengan memanfaatkan teknologi digital. Sistem ini umumnya menggunakan perangkat seperti kamera CCTV sebagai sumber data utama yang kemudian diproses untuk menghasilkan informasi mengenai kondisi lalu lintas dalam waktu yang hampir bersamaan dengan kejadian sebenarnya. Dengan demikian, sistem *real-*

time memungkinkan pengguna maupun pengelola jalan untuk memperoleh informasi yang aktual dan responsif terhadap perubahan kondisi di lapangan [50], [51].

Dalam implementasinya, sistem *monitoring* lalu lintas *real-time* biasanya terintegrasi dengan konsep *Intelligent Transportation System* (ITS). Konsep ini menggabungkan teknologi informasi, komunikasi, serta analisis data untuk meningkatkan efisiensi, keamanan, dan kenyamanan dalam sistem transportasi. Data yang diperoleh dari CCTV diproses menggunakan teknologi *Computer vision* dan *Deep learning* untuk mendeteksi kendaraan, mengklasifikasikan jenis kendaraan, serta menghitung jumlah kendaraan secara otomatis [52], [53]. Proses tersebut memungkinkan pengambilan keputusan yang lebih cepat dan akurat dibandingkan metode konvensional yang masih bergantung pada pengamatan manual.

Sistem *monitoring* lalu lintas *real-time* memiliki berbagai keunggulan dibandingkan sistem *monitoring* konvensional. Sistem mampu memberikan informasi kondisi jalan secara cepat dan kontinu sehingga dapat digunakan untuk memantau arus kendaraan, memperoleh informasi jumlah kendaraan, serta membantu proses pengambilan keputusan secara langsung. Selain itu, sistem juga dapat membantu pihak pengelola jalan dalam melakukan evaluasi dan pengelolaan lalu lintas secara lebih efektif dan efisien. Informasi yang dihasilkan dari sistem *monitoring* juga dapat digunakan sebagai bahan analisis untuk mendukung pengembangan sistem transportasi cerdas di masa mendatang.

Perkembangan teknologi *Deep learning* dan *object detection* membuat sistem *monitoring* lalu lintas *real-time* menjadi semakin efektif dalam melakukan analisis kondisi jalan. Dengan memanfaatkan algoritma seperti YOLO, sistem mampu mendeteksi kendaraan secara otomatis pada setiap frame video CCTV dengan kecepatan inferensi yang tinggi. Hasil deteksi tersebut dapat digunakan untuk menghitung jumlah kendaraan, mengidentifikasi jenis kendaraan, serta menyajikan informasi hasil deteksi secara *real-time*.

Namun demikian, sistem *monitoring real-time* juga memiliki beberapa tantangan. Sistem membutuhkan koneksi jaringan internet yang stabil untuk mengakses video streaming CCTV secara langsung. Selain itu, proses deteksi kendaraan secara *real-time* juga membutuhkan kemampuan komputasi yang memadai, terutama ketika jumlah kendaraan pada video CCTV cukup banyak atau resolusi video yang digunakan cukup tinggi. Kualitas video CCTV, kondisi pencahayaan, serta sudut pengambilan gambar juga dapat mempengaruhi performa sistem dalam melakukan proses deteksi kendaraan [53], [54].

2.3 Framework dan Algoritma yang digunakan

2.3.1 Framework

Dalam penelitian berbasis data dan *machine learning*, diperlukan suatu kerangka kerja (*framework*) yang sistematis untuk memastikan proses pengolahan data dilakukan secara terstruktur dan menghasilkan model yang optimal. Framework membantu peneliti dalam mengorganisasi tahapan penelitian mulai dari pemahaman permasalahan hingga implementasi sistem sehingga penelitian menjadi lebih terarah dan mudah dievaluasi [55].

Terdapat beberapa framework yang umum digunakan dalam proses *data mining* dan *machine learning*, di antaranya adalah CRISP-DM (*Cross-Industry Standard Process for Data Mining*), SEMMA (*Sample, Explore, Modify, Model, Assess*), dan KDD (*Knowledge Discovery in Databases*). Ketiga framework tersebut memiliki tujuan yang sama, yaitu menghasilkan informasi atau model yang bermanfaat dari data, namun memiliki perbedaan pada tahapan dan fokus prosesnya [55], [56], [57].

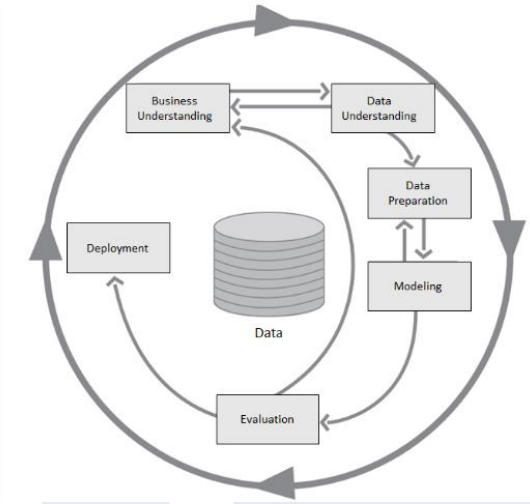
Tabel 2.2 Perbandingan Metode Pengembangan Sistem

Aspek	CRISP-DM	SEMMA	KDD
Fokus Utama	Menyelesaikan masalah bisnis dengan pendekatan berbasis data	Teknik eksplorasi data dan pembuatan model statistik	Penemuan pola atau pengetahuan baru dalam data
Tahapan Utama	1. *Business Understanding* 2. *Data Understanding* 3. *Data Preparation* 4. *Modeling* 5. *Evaluation* 6. *Deployment*	1. *Sampling* 2. *Exploration* 3. *Modification* 4. *Modeling* 5. *Assessment*	1. *Selection* 2. *Preprocessing* 3. *Transformation* 4. *Data Mining* 5. *Evaluation*

Kelebihan	Fleksibel dan terstruktur sehingga dapat disesuaikan dengan kebutuhan	Berfokus pada model analitik dengan pendekatan statistik	Dapat menangani data besar dan kompleks.
Kekurangan	Membutuhkan data yang besar dan memerlukan waktu yang lama.	Tidak sefleksibel CRISP-DM dan lebih berfokus pada statistik.	Memiliki tingkat kompleksitas yang tinggi

CRISP-DM merupakan framework yang banyak digunakan karena mencakup seluruh tahapan proses secara lengkap, mulai dari pemahaman masalah hingga implementasi sistem. Framework ini terdiri dari enam tahapan utama, yaitu *business understanding*, *data understanding*, *data preparation*, *modeling*, *evaluation*, dan *deployment*. Tahapan tersebut memungkinkan peneliti untuk tidak hanya fokus pada pemodelan, tetapi juga memastikan bahwa solusi yang dihasilkan sesuai dengan kebutuhan dan dapat diimplementasikan secara nyata [55], [58], [59].

Sementara itu, SEMMA merupakan framework yang lebih berfokus pada proses analisis data dengan tahapan *sample*, *explore*, *modify*, *model*, dan *assess*. Framework ini menitikberatkan pada proses teknis dalam pengolahan dan pemodelan data, namun tidak mencakup tahapan pemahaman masalah secara mendalam maupun implementasi sistem secara langsung. Di sisi lain, KDD merupakan proses penemuan pengetahuan dari data yang terdiri dari tahapan *selection*, *preprocessing*, *transformation*, *data mining*, dan *interpretation*. Framework ini lebih berorientasi pada proses ekstraksi pola dari data dan tidak secara eksplisit mencakup tahap *deployment* atau implementasi sistem [56], [57], [60].



Gambar 2.1 Framework CRISP-DM

Berdasarkan Gambar 2.1, framework CRISP-DM memiliki alur proses yang bersifat iteratif dan saling terhubung antar tahapan. Proses dimulai dari tahap *Business Understanding* untuk memahami permasalahan dan tujuan penelitian. Selanjutnya dilakukan tahap *Data Understanding* untuk memahami karakteristik data yang digunakan. Setelah itu dilakukan tahap *Data Preparation* untuk mempersiapkan data sebelum digunakan pada proses pelatihan model. Tahap berikutnya adalah *Modeling* untuk membangun model deteksi kendaraan menggunakan algoritma YOLOv8m. Model yang telah dibuat kemudian dievaluasi pada tahap *Evaluation* untuk mengetahui performa model. Setelah model dinilai optimal, tahap terakhir yaitu *Deployment* dilakukan untuk mengimplementasikan model ke dalam sistem *monitoring* lalu lintas berbasis web [57], [59].

2.3.2 Algoritma

a) Perbandingan Algoritma YOLO dan CNN

Dalam bidang *Computer vision*, algoritma berbasis *Deep learning* banyak digunakan untuk melakukan proses pengenalan dan deteksi objek pada gambar maupun video. Salah satu algoritma yang paling umum digunakan adalah CNN (*Convolutional Neural Network*), yaitu arsitektur jaringan saraf tiruan yang dirancang khusus untuk mengolah data visual melalui proses konvolusi untuk mengekstraksi fitur dari gambar [32], [61], [62]. CNN memiliki kemampuan yang baik dalam mengenali pola, bentuk,

tekstur, dan karakteristik objek sehingga banyak diterapkan pada proses klasifikasi gambar dan pengenalan objek.

Pada perkembangannya, CNN digunakan sebagai dasar dalam berbagai metode deteksi objek, seperti R-CNN, Fast R-CNN, dan Faster R-CNN. Metode tersebut menggunakan pendekatan *two-stage detection*, yaitu melakukan proses pencarian area objek (*region proposal*) terlebih dahulu sebelum melakukan klasifikasi objek [10]. Pendekatan ini mampu menghasilkan tingkat akurasi yang tinggi karena proses analisis dilakukan secara detail pada setiap kandidat objek. Namun, proses tersebut membutuhkan waktu komputasi yang lebih lama sehingga kurang optimal untuk aplikasi berbasis *real-time*.

Sementara itu, YOLO (*You Only Look Once*) merupakan algoritma deteksi objek berbasis *Deep learning* yang menggunakan pendekatan *one-stage detection*. YOLO melakukan proses deteksi dan klasifikasi objek secara langsung dalam satu tahap tanpa memerlukan proses *region proposal* secara terpisah [63]. Algoritma ini membagi gambar menjadi beberapa grid dan memprediksi lokasi objek (*bounding box*) beserta kelas objek secara simultan. Pendekatan tersebut membuat YOLO memiliki keunggulan dalam hal kecepatan dan efisiensi dibandingkan metode berbasis CNN konvensional.

Keunggulan utama YOLO terletak pada kemampuannya dalam mendeteksi banyak objek secara bersamaan dengan waktu inferensi yang cepat sehingga sangat cocok digunakan pada aplikasi *monitoring* berbasis video *real-time*, seperti sistem *monitoring* lalu lintas berbasis CCTV. Selain itu, YOLO juga memiliki efisiensi komputasi yang lebih baik sehingga dapat diimplementasikan pada berbagai perangkat dengan performa yang tetap stabil [64], [65].

Meskipun demikian, YOLO tetap memiliki beberapa keterbatasan, terutama dalam mendeteksi objek berukuran kecil atau objek yang saling bertumpang tindih (*occlusion*). Di sisi lain, CNN berbasis *two-stage*

detector umumnya memiliki kemampuan yang lebih baik dalam analisis detail objek, namun membutuhkan sumber daya komputasi yang lebih besar. Oleh karena itu, pemilihan algoritma perlu disesuaikan dengan kebutuhan sistem yang dikembangkan.

Dalam penelitian ini, algoritma YOLO dipilih karena lebih sesuai untuk sistem *monitoring* lalu lintas berbasis CCTV yang membutuhkan proses deteksi kendaraan dan *vehicle counting* secara cepat dan *real-time*. Perbandingan antara algoritma YOLO dan CNN dapat dilihat pada Tabel 2.3.

Tabel 2.3 Perbandingan Algoritma

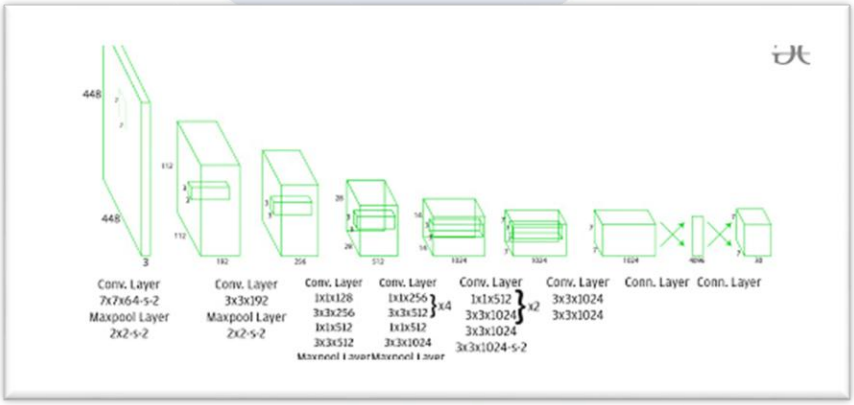
Aspek	YOLO	CNN
Tujuan Utama	Digunakan untuk mendeteksi dan mengklasifikasikan objek secara langsung pada gambar maupun video dalam waktu *real-time*	Digunakan untuk mengenali pola, mengekstraksi fitur, dan melakukan klasifikasi objek pada gambar
Kecepatan	Memiliki proses deteksi yang sangat cepat karena menggunakan metode *one-stage detection*, sehingga cocok untuk aplikasi *real-time*	Relatif lebih lambat karena proses deteksi dilakukan melalui beberapa tahapan dan membutuhkan komputasi yang lebih kompleks
Kelebihan	Mampu mendeteksi banyak objek secara simultan dalam satu frame, memiliki efisiensi komputasi yang baik, serta cocok digunakan pada sistem *monitoring* berbasis video *real-time*	Memiliki kemampuan yang baik dalam mempelajari fitur visual secara mendalam sehingga efektif untuk proses klasifikasi dan analisis gambar yang kompleks
Kekurangan	Akurasi dapat menurun pada objek berukuran kecil, objek yang saling bertumpang tindih, atau kondisi gambar dengan kualitas rendah	Membutuhkan waktu pemrosesan dan penggunaan memori yang lebih besar sehingga kurang optimal untuk implementasi sistem *real-time*

Berdasarkan perbandingan tersebut, dapat disimpulkan bahwa algoritma YOLO lebih sesuai digunakan pada penelitian ini karena memiliki kecepatan deteksi yang tinggi dan mampu bekerja secara *real-time* pada video CCTV. Kemampuan YOLO dalam mendeteksi banyak objek secara

simultan juga mendukung proses *monitoring* lalu lintas dan perhitungan jumlah kendaraan secara otomatis. Sementara itu, meskipun CNN memiliki kemampuan analisis fitur yang baik, proses komputasinya yang lebih kompleks membuatnya kurang optimal untuk implementasi sistem *monitoring* lalu lintas berbasis *real-time*.

b) YOLO

YOLO (*You Only Look Once*) merupakan algoritma deteksi objek berbasis *Deep learning* yang dikembangkan untuk melakukan proses deteksi objek secara cepat dan akurat dalam satu tahap proses (*one-stage detection*). Berbeda dengan metode deteksi objek konvensional yang melakukan proses pencarian objek dan klasifikasi secara terpisah, YOLO melakukan seluruh proses deteksi hanya dalam satu kali analisis terhadap gambar. Pendekatan ini membuat YOLO memiliki kecepatan inferensi yang tinggi sehingga sangat cocok digunakan untuk aplikasi berbasis *real-time* [66], [67].

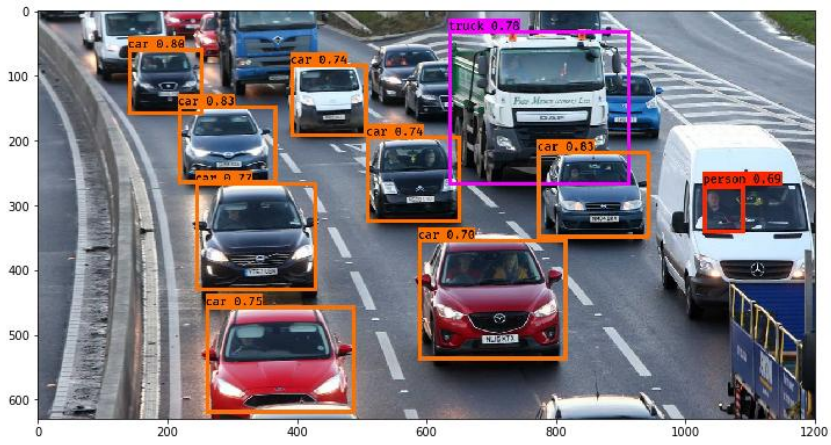


Gambar 2.2 Ilustrasi Arsitektur YOLO dalam Memproses Satu Frame Gambar [68]

Berdasarkan Gambar 2.2, algoritma YOLO menggunakan arsitektur jaringan saraf berbasis *Convolutional Neural Network* (CNN) yang terdiri dari beberapa *convolution layer*, *pooling layer*, dan *fully connected layer* [72]. Pada implementasi video, setiap video terlebih dahulu dipecah menjadi serangkaian frame, kemudian setiap frame diproses secara individual oleh arsitektur YOLO. Pada tahap awal, frame masukan diproses melalui

beberapa lapisan konvolusi untuk mengekstraksi fitur-fitur penting, seperti bentuk, warna, dan pola objek. Selanjutnya, fitur yang diperoleh digunakan untuk memprediksi lokasi objek dalam bentuk *bounding box*, nilai *confidence score*, serta kelas objek yang terdeteksi. Dengan pendekatan tersebut, YOLO mampu melakukan proses deteksi objek secara langsung dalam satu tahap (*one-stage detection*), sehingga memiliki kecepatan inferensi yang tinggi dibandingkan metode deteksi objek konvensional.

Pada proses kerjanya, YOLO membagi gambar input menjadi beberapa grid. Setiap grid bertugas untuk memprediksi lokasi objek dalam bentuk *bounding box*, nilai kepercayaan (*confidence score*), serta kelas objek yang terdeteksi. Hasil prediksi dari seluruh grid kemudian digabungkan untuk menentukan objek yang terdapat pada gambar atau video. Dengan pendekatan tersebut, YOLO mampu mendeteksi banyak objek secara simultan dalam satu frame dengan waktu pemrosesan yang relatif cepat.



Gambar 2.3 Contoh Hasil Deteksi Objek Menggunakan YOLO

Berdasarkan Gambar 2.3, algoritma YOLO mampu mendeteksi beberapa objek kendaraan seperti mobil dan truk secara bersamaan pada satu gambar. Setiap objek yang berhasil dikenali diberikan *bounding box*, label kelas objek, serta nilai *confidence score* yang menunjukkan tingkat keyakinan model terhadap hasil prediksi objek tersebut. Hasil deteksi tersebut menunjukkan bahwa YOLO memiliki kemampuan yang baik

dalam melakukan proses *object detection* pada kondisi lalu lintas dengan jumlah kendaraan yang cukup banyak.

Dalam proses deteksi objek, YOLO menggunakan beberapa metrik evaluasi untuk mengukur performa model, seperti *Intersection over Union* (IoU), *Precision*, *Recall*, dan *Mean Average Precision* (mAP). IoU digunakan untuk mengukur tingkat kecocokan antara *bounding box* hasil prediksi dengan *ground truth*.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

Rumus 2.1 Rumus IoU

Semakin tinggi nilai IoU, maka semakin baik hasil deteksi objek yang dihasilkan model. Selain itu, *Precision* digunakan untuk mengukur tingkat ketepatan model dalam mendeteksi objek.

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

Rumus 2.2 Rumus Precision

Keterangan:

- TP (*True Positive*) = objek berhasil terdeteksi dengan benar
- FP (*False Positive*) = objek terdeteksi salah

Sementara itu, *Recall* digunakan untuk mengukur kemampuan model dalam menemukan seluruh objek yang sebenarnya ada pada gambar atau video.

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

Rumus 2.3 Rumus Recall

Keterangan:

- FN (*False Negative*) = objek yang gagal terdeteksi

Untuk mengukur performa keseluruhan model deteksi objek, digunakan metrik *Mean Average Precision* (mAP) yang merupakan rata-rata nilai *Average Precision* dari seluruh kelas objek yang dideteksi.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} A\,P_i$$

Rumus 2.4 Rumus mAP

Keterangan:

- AP_i= nilai *Average Precision* tiap kelas
- N= jumlah kelas objek

Perkembangan algoritma YOLO terus mengalami peningkatan dari berbagai versi, mulai dari YOLOv1 hingga versi terbaru seperti YOLOv8 dan YOLO11. Setiap versi menghadirkan peningkatan dari segi arsitektur model, efisiensi komputasi, kecepatan inferensi, serta akurasi deteksi objek[69]. Selain itu, perkembangan arsitektur YOLO juga menunjukkan adanya optimasi pada bagian *backbone*, *neck*, dan *head* untuk meningkatkan kemampuan model dalam mendeteksi objek pada berbagai kondisi lingkungan dan ukuran objek yang berbeda.

YOLO banyak digunakan dalam berbagai bidang, seperti sistem keamanan, kendaraan otonom, pengawasan video, hingga *monitoring* lalu lintas. Dalam konteks penelitian ini, YOLO digunakan untuk mendeteksi kendaraan seperti mobil, truk, dan bus pada video CCTV jalan tol. Hasil deteksi tersebut kemudian digunakan untuk menghitung jumlah kendaraan serta menyajikan informasi hasil deteksi secara *real-time* pada sistem *monitoring* berbasis web.

c) Perbandingan YOLO berdasarkan Versi

Algoritma YOLO (*You Only Look Once*) terus mengalami perkembangan dari berbagai versi yang dikembangkan untuk meningkatkan performa deteksi objek, baik dari segi akurasi, kecepatan inferensi, maupun efisiensi komputasi. Setiap versi YOLO memiliki karakteristik dan

keunggulan yang berbeda sesuai dengan kebutuhan implementasi sistem. Dalam penelitian ini, dilakukan perbandingan terhadap beberapa versi YOLO, yaitu YOLOv5m, YOLOv8m, dan YOLO11m untuk mengetahui perbedaan karakteristik masing-masing model dalam proses deteksi kendaraan pada video CCTV.

YOLOv5m merupakan salah satu versi YOLO yang cukup populer dan banyak digunakan dalam berbagai penelitian deteksi objek. Model ini dikenal memiliki performa yang stabil, mudah digunakan, serta mendukung proses pelatihan dan implementasi yang relatif sederhana. YOLOv5m juga memiliki kecepatan inferensi yang baik sehingga mampu digunakan pada aplikasi berbasis *real-time*. Namun, dibandingkan dengan versi terbaru, YOLOv5m masih memiliki keterbatasan dalam efisiensi arsitektur dan kemampuan deteksi objek kecil [32], [69], [70].

YOLOv8m merupakan pengembangan terbaru dari algoritma YOLO yang menghadirkan peningkatan dari sisi arsitektur model, efisiensi proses inferensi, serta akurasi deteksi objek. YOLOv8m (*medium*) memiliki ukuran model menengah yang memberikan keseimbangan antara akurasi dan kecepatan. Model ini dinilai lebih optimal dalam mendeteksi objek kendaraan pada video CCTV karena mampu memberikan performa yang baik tanpa mengorbankan kecepatan proses secara signifikan [32], [65], [69], [70].

Sementara itu, YOLO11m merupakan generasi terbaru dari keluarga YOLO yang memiliki peningkatan pada kecepatan inferensi dan efisiensi komputasi. Model ini dirancang untuk memberikan performa deteksi yang lebih stabil dengan penggunaan sumber daya yang lebih optimal. Selain itu, YOLO11m juga mendukung implementasi sistem deteksi objek berbasis *real-time* dengan performa yang lebih modern dibandingkan versi sebelumnya [19], [69], [71], [72].

Perbandingan antara YOLOv5m, YOLOv8m, dan YOLO11m dapat dilihat pada Tabel 2.4 berikut.

Tabel 2.4 Perbandingan Versi YOLO

Aspek	YOLOv5	YOLOV8	YOLOv11	YOLOv12
Arsitektur	Anchor-based dengan backbone CSPDarknet.	Anchor-free dengan backbone C2f yang lebih modern.	Arsitektur yang dioptimalkan untuk efisiensi ekstraksi fitur dan komputasi.	Arsitektur terbaru dengan penyempurnaan mekanisme ekstraksi fitur dan efisiensi inferensi.
Fokus Pengembangan	Stabilitas model dan kemudahan implementasi.	Keseimbangan antara akurasi deteksi dan kecepatan inferensi.	Efisiensi komputasi dan peningkatan performa model.	Peningkatan akurasi, efisiensi, dan kemampuan generalisasi model.
Kecepatan Inferensi	Cepat dan stabil untuk berbagai kebutuhan deteksi.	Cepat dengan optimasi inferensi yang lebih baik.	Cepat dengan optimasi arsitektur terbaru.	Cepat dengan optimasi inferensi dan efisiensi komputasi yang lebih lanjut.
Akurasi Deteksi	Baik, namun lebih rendah dibandingkan versi terbaru.	Tinggi dengan kemampuan deteksi objek kecil yang lebih baik.	Tinggi dengan peningkatan performa pada berbagai kondisi deteksi.	Tinggi dengan peningkatan akurasi pada berbagai skenario deteksi.
Kebutuhan Komputasi	Relatif ringan dan dapat dijalankan pada GPU kelas menengah.	Membutuhkan sumber daya komputasi lebih besar dibanding model kecil.	Membutuhkan sumber daya komputasi lebih tinggi karena arsitektur lebih kompleks.	Membutuhkan sumber daya komputasi yang lebih tinggi sesuai kompleksitas arsitektur terbaru.

Berdasarkan perbandingan tersebut, setiap versi YOLO memiliki kelebihan dan kekurangan masing-masing sesuai dengan kebutuhan implementasi sistem. YOLOv5 lebih menekankan pada stabilitas dan kemudahan implementasi, sedangkan YOLOv11 dan YOLOv12 berfokus pada peningkatan efisiensi komputasi, akurasi, dan pengembangan arsitektur yang lebih modern. Di sisi lain, YOLOv8 dirancang untuk memberikan keseimbangan antara akurasi deteksi dan kecepatan inferensi melalui penggunaan arsitektur *anchor-free* dan *backbone* C2f. Karakteristik tersebut menjadikan YOLOv8 sebagai salah satu model yang banyak digunakan pada berbagai penelitian yang membutuhkan deteksi objek secara *real-time*, termasuk pada aplikasi berbasis *computer vision*.

d) YOLOv8

YOLOv8 merupakan salah satu versi terbaru dari algoritma *You Only Look Once* (YOLO) yang dikembangkan oleh *Ultralytics* untuk meningkatkan performa deteksi objek berbasis *Deep learning*[73]. YOLOv8 dirancang dengan arsitektur yang lebih modern dibandingkan versi sebelumnya, sehingga mampu menghasilkan proses deteksi objek yang lebih cepat, akurat, dan efisien. Algoritma ini banyak digunakan pada berbagai aplikasi *Computer vision*, seperti deteksi kendaraan, pengawasan CCTV, kendaraan otonom, hingga sistem *monitoring* lalu lintas secara *real-time* [23], [69], [74].

Berbeda dengan beberapa versi sebelumnya, YOLOv8 menggunakan pendekatan *anchor-free detection* yang memungkinkan proses prediksi objek menjadi lebih sederhana dan efisien. Selain itu, YOLOv8 juga menggunakan struktur *decoupled head*, yaitu pemisahan antara proses klasifikasi objek dan regresi *bounding box*, sehingga mampu meningkatkan performa deteksi objek secara lebih optimal [74]. Arsitektur YOLOv8 secara umum terdiri dari tiga bagian utama, yaitu *backbone*, *neck*, dan *head*.

Bagian *backbone* berfungsi untuk mengekstraksi fitur penting dari gambar *input*. YOLOv8 menggunakan arsitektur C2f (*Cross Stage Partial with Feature Fusion*) yang mampu meningkatkan efisiensi proses ekstraksi fitur dibandingkan arsitektur sebelumnya[74]. Selanjutnya, bagian *neck* digunakan untuk menggabungkan fitur dari berbagai skala menggunakan metode *Feature Pyramid Network* (FPN) dan *Path Aggregation Network* (PAN). Proses ini bertujuan untuk meningkatkan kemampuan model dalam mendeteksi objek dengan ukuran yang berbeda-beda.

Sementara itu, bagian *head* pada YOLOv8 bertugas untuk melakukan prediksi lokasi objek (*bounding box*), nilai kepercayaan (*confidence score*), serta klasifikasi objek. YOLOv8 mampu melakukan proses deteksi objek secara simultan dalam satu tahap (*one-stage detection*) sehingga

menghasilkan kecepatan inferensi yang tinggi dan cocok digunakan untuk aplikasi berbasis *real-time*[75].

YOLOv8 memiliki beberapa varian model berdasarkan ukuran dan kompleksitas arsitektur, yaitu YOLOv8n (*nano*), YOLOv8s (*small*), YOLOv8m (*medium*), YOLOv8l (*large*), dan YOLOv8x (*extra large*). Setiap varian memiliki perbedaan pada jumlah parameter, kebutuhan komputasi, serta tingkat akurasi deteksi objek. Semakin besar ukuran model, maka umumnya akurasi yang dihasilkan akan semakin tinggi, namun membutuhkan sumber daya komputasi yang lebih besar.

Tabel 2.5 Varian Model YOLOv8

Varian	Ukuran Model	Kecepatan	Akurasi	Kebutuhan Komputasi
YOLOv8n	Sangat kecil (*nano*)	Sangat Cepat	Relatif Lebih Rendah	Rendah
YOLOv8s	Kecil (*small*)	Cepat	Baik	Rendah – Sedang
YOLOv8m	Menengah (*medium*)	Cepat	Tinggi	Sedang
YOLOv8l	Besar (*large*)	Lebih Lambat	Sangat Tinggi	Tinggi
YOLOv8x	Sangat besar (*extra large*)	Paling Lambat	Paling Tinggi	Sangat Tinggi

2.4 Tools/*software* yang digunakan

2.4.1 Visual Studio Code

Visual Studio Code (VS Code) merupakan *source code editor* yang dikembangkan oleh Microsoft dan banyak digunakan dalam pengembangan aplikasi berbasis *Python*, *machine learning*, *deep learning*, maupun *web development* [76], [77]. VS Code memiliki berbagai fitur pendukung seperti *syntax highlighting*, terminal terintegrasi, fitur *debugging*, pengelolaan file proyek, serta dukungan ekstensi yang lengkap. Selain itu, VS Code juga mendukung integrasi dengan berbagai framework dan *library Python* seperti *Ultralytics* YOLOv8, *OpenCV*, dan *Streamlit* sehingga memudahkan proses pengembangan sistem secara menyeluruh[78].

Dalam penelitian ini, *Visual Studio Code* digunakan sebagai *Integrated Development Environment* (IDE) utama untuk proses pengembangan sistem *monitoring* lalu lintas berbasis CCTV secara *real-time*. VS Code digunakan dalam proses penulisan kode program *Python*, implementasi model YOLOv8m, pengembangan aplikasi berbasis *Streamlit*, serta pengujian sistem secara keseluruhan.

Pemilihan *Visual Studio Code* pada penelitian ini dilakukan dengan membandingkannya terhadap *Jupyter Notebook*. Perbandingan tersebut dilakukan karena kedua *software* tersebut sama-sama sering digunakan dalam bidang *machine learning*, analisis data, dan pengembangan program berbasis *Python*. Namun, keduanya memiliki fungsi dan karakteristik penggunaan yang berbeda. *Visual Studio Code* lebih berfokus pada pengembangan aplikasi dan pengelolaan proyek secara menyeluruh, sedangkan *Jupyter Notebook* lebih banyak digunakan untuk eksplorasi data, eksperimen model, dan visualisasi data dalam bentuk *notebook* interaktif[79].

Jupyter Notebook memiliki keunggulan dalam proses eksplorasi data karena memungkinkan pengguna menjalankan kode secara bertahap (*cell by cell*) serta menampilkan visualisasi dan dokumentasi dalam satu halaman. *Software* ini sangat membantu pada tahap awal pengembangan model *machine learning* dan analisis data. Akan tetapi, *Jupyter Notebook* kurang optimal untuk pengembangan aplikasi berbasis *real-time* yang memiliki banyak file program dan membutuhkan integrasi sistem secara menyeluruh [80].

Sementara itu, *Visual Studio Code* memiliki keunggulan dalam pengelolaan proyek yang lebih terstruktur, fitur *debugging* yang lebih lengkap, serta integrasi yang lebih baik dengan framework pengembangan aplikasi seperti *Streamlit*. Selain itu, VS Code juga mendukung pengembangan aplikasi berbasis web dan implementasi sistem secara langsung sehingga lebih sesuai digunakan pada penelitian ini. Kemampuan *Visual Studio Code* dalam mendukung pengembangan aplikasi *monitoring* berbasis *real-time* menjadi alasan utama pemilihan *software* tersebut dibandingkan *Jupyter Notebook*.

Tabel 2.6 Perbandingan *Visual Studio Code* dan *Jupyter*

Indikator	*Visual Studio Code*	*Jupyter*
Kinerja	Lebih optimal untuk pengembangan aplikasi dan proyek skala besar	Lebih optimal untuk eksplorasi data dan eksperimen model
Kemudahan Pengunaan	Memerlukan konfigurasi awal namun fleksibel untuk berbagai kebutuhan	Mudah digunakan karena berbasis notebook interaktif
Integrasi	Mendukung integrasi dengan berbagai framework seperti *Streamlit* dan YOLOv8m	Integrasi lebih terbatas untuk pengembangan aplikasi secara menyeluruh
Library	Mendukung banyak *library* dan ekstensi pengembangan *Python*	Mendukung *library* analisis data dan visualisasi
Kelebihan	Memiliki fitur debugging lengkap, pengelolaan proyek lebih terstruktur, dan cocok untuk pengembangan sistem *real-time*	Memudahkan eksplorasi data, visualisasi, dan pengujian kode secara bertahap
Kekurangan	Membutuhkan konfigurasi tambahan pada beberapa *library* dan ekstensi	Kurang optimal untuk pengembangan aplikasi berbasis web dan proyek kompleks

Berdasarkan perbandingan tersebut, *Visual Studio Code* dipilih sebagai *software* utama dalam penelitian ini karena memiliki kemampuan yang lebih baik dalam pengembangan sistem secara terintegrasi dan mendukung implementasi aplikasi berbasis *real-time*. Selain itu, *Visual Studio Code* juga lebih sesuai untuk pengembangan aplikasi *monitoring* lalu lintas berbasis *Streamlit* dibandingkan *Jupyter Notebook* yang lebih berfokus pada eksplorasi data dan eksperimen model.

2.4.2 *Python*

Python merupakan bahasa pemrograman tingkat tinggi (*high-level programming language*) yang banyak digunakan dalam bidang *machine learning*, *deep learning*, *computer vision*, analisis data, serta pengembangan aplikasi berbasis web [76], [79]. *Python* memiliki sintaks yang sederhana dan mudah dipahami sehingga banyak digunakan dalam penelitian maupun pengembangan sistem berbasis kecerdasan buatan (*Artificial Intelligence*). Selain itu, *Python* juga didukung oleh komunitas yang besar dan memiliki banyak *library* yang dapat digunakan untuk mendukung berbagai kebutuhan pengembangan sistem [76].

Dalam penelitian ini, *Python* digunakan sebagai bahasa pemrograman utama dalam proses implementasi sistem *monitoring* lalu lintas berbasis CCTV secara *real-time*. *Python* digunakan untuk proses pengolahan video CCTV, implementasi model YOLOv8m, deteksi kendaraan, perhitungan jumlah kendaraan, serta integrasi dengan aplikasi berbasis web menggunakan *Streamlit*.

Pemilihan *Python* pada penelitian ini dilakukan dengan membandingkannya terhadap bahasa pemrograman *R*. Perbandingan tersebut dilakukan karena *Python* dan *R* merupakan bahasa pemrograman yang cukup populer dalam bidang analisis data dan *machine learning*[76], [79]. Namun, kedua bahasa pemrograman tersebut memiliki fokus penggunaan yang berbeda. *Python* lebih banyak digunakan untuk pengembangan sistem berbasis *Artificial Intelligence*, *Deep learning*, dan *Computer vision*, sedangkan *R* lebih berfokus pada pengolahan statistik dan visualisasi data.

R memiliki kemampuan yang sangat baik dalam analisis statistik, pengolahan data akademik, dan visualisasi data. Selain itu, *R* juga banyak digunakan pada penelitian berbasis statistik karena memiliki berbagai package yang mendukung proses analisis data secara mendalam. Akan tetapi, implementasi *Computer vision*, pengolahan video secara *real-time*, serta integrasi dengan framework *Deep learning* pada *R* masih lebih terbatas dibandingkan *Python*.

Sementara itu, *Python* memiliki dukungan *library* yang sangat lengkap untuk kebutuhan *Computer vision* dan *Deep learning*, seperti *OpenCV*, *TensorFlow, PyTorch, NumPy, Pandas,* serta *Ultralytics* YOLOv8. Selain itu, *Python* juga mendukung integrasi dengan framework pengembangan aplikasi berbasis web seperti *Streamlit* sehingga lebih sesuai digunakan dalam penelitian ini. Kemampuan *Python* dalam mengolah video secara *real-time* dan implementasi model deteksi objek menjadi alasan utama pemilihan *Python* dibandingkan *R*.

Tabel 2.7 Perbandingan *Python* dan R Studio

Indikator	*Python*	*R Studio*
Kinerja	Optimal untuk pengembangan AI, *Deep learning*, *Computer vision*, dan aplikasi *real-time*	Optimal untuk analisis statistik dan visualisasi data
Kemudahan Penggunaan	Sintaks sederhana dan mudah dipahami	Lebih fokus pada kebutuhan statistik
Integrasi	Mendukung integrasi dengan berbagai framework seperti YOLOv8, *OpenCV*, dan *Streamlit*	Integrasi framework *Computer vision* lebih terbatas
Library	Memiliki *library* AI dan *Computer vision* yang sangat lengkap	Kuat pada *library* statistik dan analisis data
Kelebihan	Fleksibel, mudah dikembangkan, dan mendukung implementasi sistem secara menyeluruh	Sangat baik dalam analisis statistik dan visualisasi data
Kekurangan	Membutuhkan optimasi pada komputasi skala besar	Kurang optimal untuk implementasi *Deep learning* dan *Computer vision*

Berdasarkan perbandingan tersebut, *Python* dipilih sebagai bahasa pemrograman utama dalam penelitian ini karena memiliki fleksibilitas tinggi, dukungan *library* yang lengkap, serta kemampuan integrasi yang baik untuk pengembangan sistem *monitoring* lalu lintas berbasis CCTV secara *real-time*. Selain itu, *Python* juga lebih sesuai untuk implementasi algoritma YOLOv8m dan pengembangan aplikasi berbasis web menggunakan *Streamlit* dibandingkan bahasa pemrograman *R*.

2.4.3 *Streamlit*

Streamlit merupakan framework berbasis *Python* yang digunakan untuk membangun aplikasi web interaktif secara sederhana dan cepat. *Streamlit* banyak digunakan dalam pengembangan aplikasi *data science*, *machine learning*, dan *deep learning* karena memungkinkan pengguna membuat antarmuka aplikasi tanpa memerlukan pengembangan *front-end* yang kompleks. Selain itu, *Streamlit* juga mendukung integrasi dengan berbagai *library Python* seperti *OpenCV, Pandas, Matplotlib*, dan *Ultralytics* YOLOv8 sehingga sangat sesuai digunakan dalam penelitian berbasis *Computer vision* [81].

Dalam penelitian ini, *Streamlit* digunakan untuk membangun aplikasi *monitoring* lalu lintas berbasis CCTV secara *real-time*. *Streamlit* digunakan

untuk menampilkan video CCTV, hasil deteksi kendaraan menggunakan YOLOv8, jumlah kendaraan yang terdeteksi, serta dashboard statistik secara interaktif. Penggunaan *Streamlit* memudahkan proses visualisasi hasil deteksi objek sehingga sistem dapat digunakan dengan lebih mudah oleh pengguna.

Pemilihan *Streamlit* pada penelitian ini dilakukan dengan membandingkannya terhadap *Spyder*. Perbandingan tersebut dilakukan karena keduanya sama-sama sering digunakan dalam pengembangan program berbasis *Python*, khususnya pada bidang analisis data dan *machine learning*. Namun, kedua *software* tersebut memiliki fungsi dan tujuan penggunaan yang berbeda. *Streamlit* lebih berfokus pada pengembangan aplikasi berbasis web interaktif, sedangkan *Spyder* lebih banyak digunakan sebagai *Integrated Development Environment* (IDE) untuk penulisan dan analisis kode program *Python*.

Spyder memiliki beberapa keunggulan seperti fitur debugging, *variable explorer*, dan tampilan yang mendukung proses analisis data secara detail [76]. *Software* ini sangat cocok digunakan dalam proses pengembangan kode program, pengujian algoritma, dan komputasi ilmiah. Akan tetapi, *Spyder* tidak dirancang untuk membangun aplikasi web interaktif yang dapat diakses secara langsung oleh pengguna. Oleh karena itu, implementasi sistem *monitoring* berbasis *real-time* menggunakan *Spyder* menjadi kurang optimal.

Sementara itu, *Streamlit* memungkinkan proses pengembangan aplikasi berbasis web dilakukan dengan lebih sederhana hanya menggunakan bahasa pemrograman *Python* tanpa memerlukan HTML, CSS, maupun *JavaScript* secara mendalam. Selain itu, *Streamlit* juga mendukung tampilan visualisasi secara langsung sehingga hasil deteksi kendaraan dan informasi jumlah kendaraan dapat ditampilkan secara *real-time* melalui antarmuka web yang interaktif. Hal tersebut menjadi alasan utama *Streamlit* lebih sesuai digunakan pada penelitian ini dibandingkan *Spyder*.

Tabel 2.8 Perbandingan *Streamlit* dan *Spyder*

Indikator	*Streamlit*	*Spyder*
Kinerja	Optimal untuk pengembangan aplikasi web interaktif	Optimal untuk analisis data dan pengembangan kode *Python*

Kemudahan Penggunaan	Mudah digunakan untuk membuat aplikasi web sederhana	Mudah digunakan untuk analisis dan debugging program
Integrasi	Mendukung integrasi dengan YOLOv8, *OpenCV*, dan *Pandas*	Lebih fokus pada pengembangan kode *Python*
Library	Mendukung *library* visualisasi dan *machine learning*	Mendukung *library Python* untuk komputasi ilmiah
Kelebihan	Cepat, ringan, interaktif, dan cocok untuk aplikasi *real-time*	Memiliki fitur analisis dan *debugging* yang baik
Kekurangan	Tampilan visualisasi lebih sederhana dibanding framework web lain	Tidak dirancang untuk pengembangan aplikasi web interaktif

Berdasarkan perbandingan tersebut, *Streamlit* dipilih dalam penelitian ini karena lebih sesuai untuk membangun aplikasi *monitoring* lalu lintas berbasis CCTV secara *real-time*. Selain mudah digunakan, *Streamlit* juga mendukung integrasi dengan model YOLOv8 dan memungkinkan hasil deteksi kendaraan ditampilkan secara interaktif melalui antarmuka web sederhana.