Computer Vision dan Deep Learning dalam Revolusi Transportasi Cerdas

Disusun Oleh :

Imam Ahmad Ashari | Wahyul Amien Syafei | Adi Wibowo

Computer Vision dan Deep Learning dalam Revolusi Transportasi Cerdas

Penulis:

Imam Ahmad Ashari, S.Kom., M.Kom.
Dr. Ir. Wahyul Amien Syafei, S.T., M.T., IPM.
Dr. Eng. Adi Wibowo, S.Si., M.Kom

Penerbit Yayasan Peneliti Teknologi Teknik Indonesia

Computer Vision dan Deep Learning dalam Revolusi Transportasi Cerdas

Penulis:

Imam Ahmad Ashari, S.Kom., M.Kom.
Dr. Ir. Wahyul Amien Syafei, S.T., M.T., IPM.
Dr. Eng. Adi Wibowo, S.Si., M.Kom.

Editor:
Soiful Hadi, S.T, M.Kom.

Desain Cover:
Imam Ahmad Ashari, S.Kom., M.Kom.

Tata Letak:
Imam Ahmad Ashari, S.Kom., M.Kom.

Ukuran:
UNESCO 15x23 cm

ISBN:
Sedang Dalam Proses Penerbitan

Cetakan Pertama:
Oktober 2025

Penerbit Yayasan Peneliti Teknologi Teknik Indonesia
Jl. Empu Sedah No. 12, Pringwulung, Condongcatur, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta 55281, Indonesia

Computer Vision dan Deep Learning dalam Revolusi Transportasi Cerdas

Oleh:

Imam Ahmad Ashari, S.Kom., M.Kom.
Dr. Ir. Wahyul Amien Syafei, S.T., M.T., IPM
Dr.Eng. Adi Wibowo, S.Si., M.Kom.

Diterbitkan oleh
Penerbit Yayasan Peneliti Teknologi Teknik Indonesia
Jl. Empu Sedah No. 12, Pringwulung, Condongcatur, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta 55281, Indonesia

Imam Ahmad Ashari, S.Kom., M.Kom., Dr. Ir. Wahyul Amien Syafei, S.T., M.T., IPM., Dr.Eng. Adi Wibowo, S.Si., M.Kom.

Computer Vision dan Deep Learning dalam Revolusi Transportasi Cerdas

-edisi Pertama – Yogyakarta: Penerbit Yayasan Peneliti Teknologi Teknik Indonesia, 2025, 2025

Sinopsis

Buku "*Computer Vision* dan *Deep Learning* dalam Revolusi Transportasi Cerdas" membahas penerapan teknologi modern untuk menghadapi tantangan transportasi melalui pendekatan yang inovatif. Fokus utama buku ini adalah penggunaan algoritma You Only Look Once (YOLO) untuk mendeteksi kendaraan secara real-time dalam berbagai kondisi lalu lintas. Dengan kemampuan mendeteksi jenis kendaraan seperti motor, mobil, truk, dan bus secara cepat dan akurat, pembaca diajak memahami cara algoritma ini bekerja dalam meningkatkan efisiensi analisis lalu lintas serta mendukung pengembangan sistem transportasi cerdas.

Selain itu, buku ini mengeksplorasi optimasi model *Long Short-Term Memory* (LSTM) untuk memprediksi tingkat kemacetan lalu lintas. Dengan memanfaatkan metode metaheuristik untuk optimasi parameter LSTM serta data historis kendaraan, model deep learning ini mampu menghasilkan prediksi yang lebih akurat, mendukung pengambilan keputusan yang lebih baik dalam pengelolaan lalu lintas. Ditulis untuk akademisi, praktisi, dan mahasiswa, buku ini menjadi referensi komprehensif dalam memahami peran penting computer vision dan deep learning dalam menciptakan sistem transportasi yang lebih aman, efisien, dan berkelanjutan.

Kata Pengantar

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas terselesaikannya buku "Computer Vision dan Deep Learning dalam Revolusi Transportasi Cerdas". Buku ini hadir sebagai upaya untuk menjawab tantangan modern dalam sektor transportasi yang semakin kompleks dan dinamis. Kemajuan teknologi seperti computer vision dan deep learning telah membuka peluang baru dalam menciptakan solusi cerdas, baik untuk menganalisis lalu lintas secara real-time maupun memprediksi kondisi jalan untuk meningkatkan efisiensi dan keselamatan.

Buku ini dirancang untuk menjadi panduan praktis sekaligus teoretis bagi para akademisi, peneliti, mahasiswa, dan praktisi yang tertarik mengembangkan sistem transportasi cerdas. Kami membahas secara mendalam algoritma *You Only Look Onc*e (YOLO) untuk deteksi kendaraan secara real-time serta implementasi *Long Short-Term Memory* (LSTM) yang dioptimalkan untuk prediksi kemacetan lalu lintas. Pendekatan berbasis data dan penggunaan metode optimasi metaheuristik juga disertakan untuk memberikan gambaran lengkap tentang bagaimana teknologi ini dapat diterapkan dalam dunia nyata.

Kami menyadari bahwa buku ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran dari pembaca sangat kami harapkan untuk penyempurnaan di masa mendatang. Semoga buku ini bermanfaat bagi

perkembangan ilmu pengetahuan, khususnya di bidang transportasi cerdas, dan memberikan kontribusi nyata bagi masyarakat.

Banyumas, Oktober 2025

Penulis

Daftar Isi

Daftar Tabel

Daftar Gambar

BAB I Teknologi Computer Vision untuk Revolusi Transportasi Cerdas

1.1. Perkembangan Computer Vision dalam Sistem Transportasi Cerdas

Perkembangan teknologi computer vision telah membawa revolusi dalam berbagai aspek kehidupan, termasuk transportasi cerdas. Computer vision, yang memungkinkan komputer untuk memahami dan menganalisis gambar atau video secara otomatis, telah menjadi komponen kunci dalam menciptakan sistem transportasi yang lebih aman, efisien, dan terintegrasi. Teknologi ini digunakan untuk mengidentifikasi, melacak, dan menganalisis kendaraan, pejalan kaki, serta objek lain di jalan raya, sehingga memungkinkan pengambilan keputusan yang lebih baik dalam manajemen lalu lintas dan pengembangan kendaraan otonom.

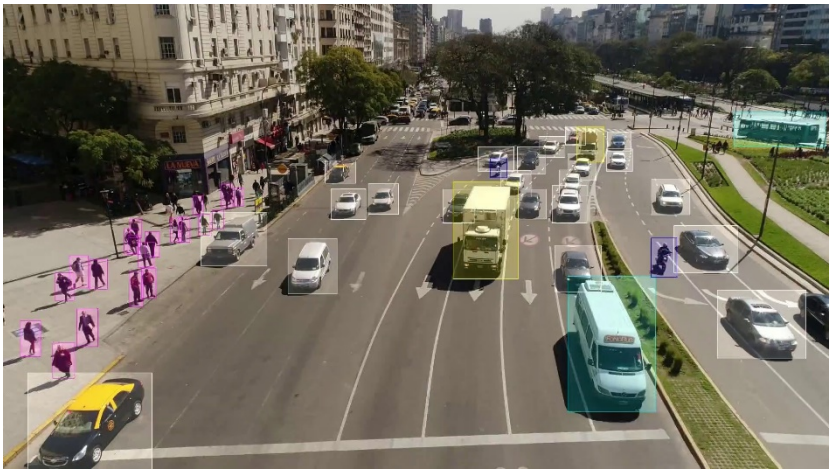
Dalam manajemen lalu lintas, computer vision memainkan peran penting dalam memantau kondisi jalan secara real-time. Kamera CCTV yang dilengkapi dengan algoritma deteksi objek, seperti YOLO dan Faster R-CNN, dapat digunakan untuk menghitung jumlah kendaraan, mendeteksi kemacetan, dan mengidentifikasi pelanggaran lalu lintas. Informasi yang dihasilkan membantu pihak berwenang dalam mengatur lalu lintas secara dinamis, seperti mengubah

durasi lampu lalu lintas atau memberikan rute alternatif untuk mengurangi kepadatan. Dengan demikian, teknologi ini memberikan solusi praktis untuk mengatasi permasalahan lalu lintas di kota-kota besar.

Tidak hanya itu, computer vision juga menjadi inti dari pengembangan kendaraan otonom, yang diharapkan menjadi solusi masa depan untuk transportasi. Melalui kombinasi kamera, sensor, dan algoritma pemrosesan gambar, kendaraan otonom mampu mengenali lingkungan sekitarnya, termasuk rambu lalu lintas, marka jalan, dan objek bergerak lainnya. Dengan kemampuan ini, kendaraan dapat mengambil keputusan secara mandiri, seperti berhenti ketika ada pejalan kaki atau menyesuaikan kecepatan sesuai dengan kondisi jalan. Hal ini tidak hanya meningkatkan efisiensi transportasi, tetapi juga mengurangi risiko kecelakaan akibat kesalahan manusia.

Kamera juga dapat ditempatkan untuk merekam gambar di persimpangan dan mengevaluasinya menggunakan algoritma computer vision. Algoritma ini dapat mengenali keberadaan kendaraan, membedakan kelasnya (misalnya, motor, mobil, truk, dan bus), dan memantau pergerakannya. Perangkat IoT, seperti mobil yang dilengkapi GPS atau ponsel, juga dapat menyediakan informasi tentang posisi dan pergerakan kendaraan di sekitar persimpangan. Semua data ini dikumpulkan dan dianalisis secara real-time menggunakan keputusan berbasis edge berdasarkan kondisi lalu lintas saat ini. Ilustrasi pemantauan arus

lalu lintas dengan teknologi computer vision dapat dilihat pada gambar 1.



Gambar 1. *Pemantauan arus lalu lintas dengan teknologi computer vision (Sumber: https://www.augmentedstartups.com/)*

Untuk analitik video berbasis computer vision terhadap objek, kejadian, dan status di jalan raya, teknologi ini sering digunakan untuk menghasilkan pesan V2X dengan latensi yang sangat rendah. Antarmuka udara PC5, yang juga dikenal sebagai sidelink, adalah media di mana Roadside Unit (RSU) mengomunikasikan pesan V2X dan informasi SPaT kepada pengguna jalan lainnya (kendaraan dan pejalan kaki) di sekitar lokasi tersebut.

Selain itu, teknologi computer vision juga mendukung inisiatif transportasi hijau dengan mengoptimalkan penggunaan sumber daya dan mengurangi emisi. Misalnya, melalui analisis video udara, pihak

berwenang dapat merancang jalur transportasi yang lebih efisien untuk kendaraan umum dan mengurangi konsumsi bahan bakar. Di sisi lain, implementasi computer vision dalam sistem berbagi kendaraan (ride-sharing) memungkinkan penyesuaian rute secara cerdas untuk mengurangi perjalanan kosong dan meningkatkan efisiensi penggunaan kendaraan. Dengan berbagai penerapannya, computer vision tidak hanya mengubah wajah transportasi modern tetapi juga mendukung tujuan keberlanjutan global.

1.2. Masa Depan Teknologi Computer Vision dalam Sistem Transportasi Cerdas

Di masa depan, teknologi ini diprediksi akan menjadi tulang punggung dalam menciptakan sistem transportasi yang lebih cerdas, efisien, dan aman. Salah satu arah pengembangan utama adalah integrasi yang lebih dalam antara computer vision dengan teknologi lain seperti Internet of Things (IoT), jaringan 5G, dan kecerdasan buatan (AI). Kombinasi ini memungkinkan pengumpulan, analisis, dan pemrosesan data secara real-time, yang akan mempercepat pengambilan keputusan dalam berbagai skenario transportasi.

Dalam konteks kendaraan otonom, teknologi computer vision akan semakin disempurnakan untuk menghadapi berbagai tantangan di dunia nyata. Sistem navigasi kendaraan otonom di masa depan diharapkan mampu beradaptasi dengan kondisi lingkungan yang kompleks, seperti jalanan dengan pencahayaan rendah, cuaca ekstrem, atau kepadatan lalu lintas yang tinggi. Model-

model deteksi objek dan segmentasi gambar akan dikembangkan dengan algoritma yang lebih canggih untuk meningkatkan akurasi dan kecepatan pemrosesan. Selain itu, penerapan teknik multi-augmentasi dan optimasi berbasis algoritma evolusi seperti Particle Swarm Optimization (PSO) dan Improved Dung Beetle Optimization (IDBO) akan menjadi kunci dalam meningkatkan performa model deep learning.

Selain kendaraan otonom, computer vision juga akan memainkan peran penting dalam manajemen lalu lintas kota pintar. Sistem berbasis computer vision akan memungkinkan pemantauan lalu lintas secara real-time untuk mengidentifikasi kemacetan, mengatur lampu lalu lintas secara dinamis, dan mendeteksi pelanggaran hukum lalu lintas dengan lebih akurat. Teknologi ini juga dapat digunakan untuk menganalisis pola perjalanan warga kota, yang akan membantu pemerintah merancang kebijakan transportasi yang lebih efektif dan ramah lingkungan.

Lebih jauh lagi, masa depan transportasi juga akan melibatkan integrasi teknologi computer vision dengan konsep mobilitas berbagi (shared mobility) dan transportasi multimoda. Misalnya, aplikasi berbasis computer vision dapat digunakan untuk memantau ketersediaan kendaraan umum atau layanan berbagi kendaraan di lokasi tertentu, serta membantu pengguna merencanakan perjalanan yang efisien dengan memanfaatkan berbagai moda transportasi.

Dengan terus berkembangnya teknologi sensor dan perangkat keras, biaya implementasi sistem berbasis computer vision diperkirakan akan semakin terjangkau. Hal ini akan membuka peluang bagi negara-negara berkembang untuk mengadopsi teknologi ini dalam upaya meningkatkan infrastruktur transportasi mereka. Selain itu, standar internasional terkait keamanan dan privasi data akan menjadi perhatian utama, mengingat volume data yang dihasilkan oleh sistem computer vision dalam transportasi sangat besar dan sering kali bersifat sensitif.

Pada akhirnya, masa depan teknologi computer vision dalam transportasi tidak hanya akan membawa efisiensi dan kenyamanan, tetapi juga memberikan kontribusi besar terhadap keselamatan dan keberlanjutan. Dengan inovasi yang terus berlanjut, transportasi cerdas berbasis computer vision akan menjadi fondasi bagi mobilitas global yang lebih baik di abad ke-21. Ilustrasi gambar peranan computer vision dalam transportasi cerdas dapat dilihat pada gambar 2.

Gambar 2. *Ilustrasi gambar peranan computer vision dalam transportasi cerdas (Sumber: https://www.unite.ai/)*

BAB II Deteksi Kendaraan Real-Time pada Lalu Lintas Perkotaan

2.1. Sistem Deteksi Kendaraan Real-Time untuk Manajemen Arus Lalu Lintas Kendaraan

Deteksi kendaraan secara real-time telah menjadi elemen penting dalam upaya meningkatkan efisiensi dan keselamatan lalu lintas perkotaan. Teknologi ini memungkinkan sistem transportasi untuk memantau arus lalu lintas, mengidentifikasi kendaraan, serta mendeteksi pola pergerakan secara langsung. Dengan kemajuan algoritma deteksi objek seperti YOLO (You Only Look Once), Faster R-CNN, dan SSD (Single Shot MultiBox Detector), sistem deteksi kendaraan kini mampu menghasilkan analisis yang cepat dan akurat meskipun dalam lingkungan yang kompleks. Teknologi ini memberikan solusi yang dibutuhkan untuk mengatasi berbagai tantangan lalu lintas yang semakin rumit di perkotaan.

Salah satu penerapan utama deteksi kendaraan real-time adalah dalam pengaturan lalu lintas berbasis data. Kamera yang ditempatkan di persimpangan jalan atau di sepanjang jalur utama dapat mendeteksi jumlah kendaraan yang melintas, jenis kendaraan, dan tingkat kepadatan lalu lintas. Data ini kemudian digunakan untuk mengatur sinyal lampu lalu lintas secara adaptif, mengurangi waktu tunggu, serta meningkatkan kelancaran arus kendaraan. Sistem seperti ini juga dapat

membantu pemerintah kota dalam merencanakan infrastruktur jalan yang lebih baik berdasarkan pola lalu lintas yang terpantau secara real-time. Ilustrasi pemantauan real-time arus lalu lintas dapat dilihat pada gambar 3.

Gambar 3. *Ilustrasi pemantauan real-time arus lalu lintas (Sumber: https://www.kusrini.com/)*

Teknologi deteksi real-time juga memainkan peran penting dalam pengawasan dan penegakan hukum di jalan raya. Sistem ini mampu mendeteksi pelanggaran lalu lintas, seperti kendaraan yang melanggar batas kecepatan, menerobos lampu merah, atau parkir sembarangan. Dengan integrasi ke dalam pusat pengendalian lalu lintas, pelanggaran tersebut dapat diidentifikasi secara otomatis dan disertai bukti visual untuk mempermudah proses penegakan hukum. Hal ini tidak hanya meningkatkan disiplin berkendara, tetapi juga memberikan efek jera bagi pelanggar, sehingga menciptakan lingkungan jalan yang lebih aman.

Deteksi kendaraan merupakan teknologi penting untuk sistem transportasi cerdas dan kendaraan otonom. Deteksi real-time yang andal memungkinkan pemantauan lalu lintas, peningkatan keamanan, dan bantuan navigasi. Namun, tugas ini merupakan tantangan besar dalam visi komputer, terutama di lingkungan perkotaan yang kompleks. Metode tradisional yang menggunakan fitur buatan seperti HAAR cascade memiliki keterbatasan. Kemajuan pembelajaran mendalam (deep learning) telah membuka peluang baru dengan menerapkan jaringan saraf konvolusi (CNN) seperti Faster R-CNN, SSD, dan YOLO untuk deteksi kendaraan dengan akurasi yang jauh lebih baik. Namun, setiap teknik memiliki kompromi antara presisi dan kecepatan pemrosesan. Misalnya, detektor dua tahap seperti Faster R-CNN sangat akurat tetapi lambat dengan 7 FPS, sedangkan detektor satu tahap seperti SSD lebih cepat pada 22 FPS tetapi kurang presisi. YOLO, yang sangat cepat pada 45 FPS, juga menghadapi tantangan dengan akurasinya yang lebih rendah.

Selain itu, deteksi kendaraan real-time memiliki potensi besar untuk mendukung sistem transportasi cerdas di masa depan. Dengan integrasi teknologi ini ke dalam kendaraan otonom atau aplikasi berbasis Internet of Things (IoT), sistem transportasi dapat menjadi lebih terhubung dan responsif. Misalnya, kendaraan otonom dapat memanfaatkan data deteksi kendaraan untuk menghindari tabrakan atau memilih rute tercepat berdasarkan kondisi lalu lintas terkini. Melalui berbagai penerapannya, teknologi deteksi kendaraan real-time

tidak hanya memberikan manfaat praktis tetapi juga membangun fondasi menuju transportasi perkotaan yang lebih cerdas dan berkelanjutan.

Di antara berbagai teknik yang ada, YOLO (You Only Look Once) menjadi pilihan yang paling menarik untuk kasus ini. Dengan kecepatan pemrosesan hingga 45 FPS, YOLO memungkinkan deteksi kendaraan secara real-time, menjadikannya sangat cocok untuk aplikasi yang memerlukan respons cepat seperti kendaraan otonom dan sistem pemantauan lalu lintas. Meskipun YOLO memiliki kelemahan pada tingkat akurasi dibandingkan metode lain seperti Faster R-CNN, pendekatan berbasis YOLO yang dilengkapi dengan teknik tambahan, seperti pelacakan optical flow dan analisis trajektori, mampu meningkatkan presisi dan keandalan deteksi. Hal ini menjadikan YOLO sebagai solusi yang efisien untuk kebutuhan transportasi modern, yang menuntut keseimbangan optimal antara kecepatan dan akurasi.

2.2. Pengumpulan Data untuk Deteksi Kendaraan

Pada penelitin yang penulis lakukan proses pengumpulan data dilakukan menggunakan kamera pengawas yang terpasang di lampu lalu lintas Fatmawati, Kota Semarang. Kamera ini digunakan untuk merekam aktivitas lalu lintas di jalur sebelah kanan selama periode tertentu, yaitu dari tanggal 19 Desember 2023 hingga 15 Februari 2024. Waktu perekaman dibatasi antara pukul 06.00 hingga 07.00 WIB setiap hari, yang merupakan jam sibuk pagi hari,

sehingga data yang dihasilkan mencerminkan kondisi lalu lintas yang padat dan relevan untuk analisis.

Rekaman video dihasilkan dengan menggunakan codec H.264/MPEG-4 AVC, yang menawarkan efisiensi kompresi tinggi tanpa mengorbankan kualitas visual. Resolusi video ditetapkan pada 1280x960 piksel untuk memastikan setiap frame memiliki tingkat kejelasan yang cukup untuk menganalisis kendaraan. Dengan frame rate 25 FPS (frame per second), video ini menghasilkan 25 frame gambar setiap detik, memberikan visualisasi yang halus dan stabil.

Setelah video direkam, dilakukan proses ekstraksi gambar untuk mendapatkan dataset dalam format gambar statis. Interval waktu ekstraksi ditetapkan setiap lima menit. Interval ini dihitung berdasarkan jumlah frame yang dihasilkan dalam durasi lima menit, yaitu dengan mengalikan frame rate (25 FPS) dengan 60 detik dan durasi lima menit, menghasilkan total 7500 frame. Setiap 7500 frame, satu gambar diekstraksi dari video. Proses ini dilakukan secara otomatis menggunakan program komputer yang dirancang untuk mengakses dan mengekstrak frame berdasarkan interval yang telah ditentukan.

Dari seluruh periode perekaman, proses ini menghasilkan total 720 gambar. Setiap gambar merepresentasikan kondisi lalu lintas pada interval waktu tertentu, memberikan distribusi data yang merata selama waktu perekaman. Setelah gambar-gambar ini diekstraksi, langkah berikutnya adalah mengklasifikasikan kendaraan yang terdeteksi ke dalam

empat kategori utama, yaitu sepeda motor, mobil, truk, dan bus. Analisis terhadap gambar menghasilkan total 31,481 sepeda motor, 12,402 mobil, 1,184 truk, dan 280 bus.

Dataset ini kemudian dibagi menjadi dua bagian, yaitu untuk pelatihan model dan validasi. Sebanyak 80% dari dataset, atau 576 gambar, digunakan untuk pelatihan. Dalam dataset pelatihan ini, terdapat 22,136 sepeda motor, 8,804 mobil, 839 truk, dan 199 bus. Sementara itu, 20% sisanya, atau 144 gambar, digunakan untuk validasi. Dataset validasi ini mencakup 9,345 sepeda motor, 3,598 mobil, 345 truk, dan 81 bus.

Proses pengumpulan data ini dirancang untuk memastikan bahwa dataset yang dihasilkan memiliki distribusi waktu yang konsisten dan mencakup berbagai kondisi lalu lintas dalam rentang waktu tertentu. Dengan membagi dataset menjadi kategori yang jelas dan mendistribusikannya secara proporsional untuk pelatihan dan validasi, data ini menjadi landasan penting untuk membangun model deteksi kendaraan yang andal. Selain itu, pendekatan ini memberikan gambaran yang komprehensif tentang pola lalu lintas di lokasi dan waktu tertentu, sehingga sangat relevan untuk penelitian dalam pengembangan teknologi transportasi cerdas berbasis computer vision.

2.3. Ekstrak Frame video rekaman CCTV dengan PyThon

Program ini dirancang untuk memproses video dari rekaman CCTV dan mengekstrak frame setiap interval

waktu tertentu. Frame yang dihasilkan disimpan dalam folder output sebagai file gambar dengan format .jpg. Berikut adalah langkah-langkah program:

Import Pustaka

Program memanfaatkan pustaka cv2 (OpenCV) untuk membaca video, numpy untuk manipulasi array, os untuk manajemen file, dan PIL.Image untuk menyimpan gambar.

Fungsi process_video

Fungsi ini menerima tiga parameter:

- video_path: Lokasi file video yang akan diproses.
- output_dir: Folder tempat menyimpan frame hasil ekstraksi.
- interval_minutes: Interval waktu dalam menit untuk mengekstrak frame.

Fungsi ini:

- Membuka video menggunakan cv2.VideoCapture.
- Menghitung interval frame berdasarkan frame rate (FPS) video.
- Membaca frame video secara berurutan dan menyimpan frame pada interval tertentu ke dalam folder output dengan nama yang mencerminkan posisi frame dalam video.

Daftar Video

Video yang akan diproses disimpan dalam daftar video_files. Contoh:

```
Data Rekaman CCTV/18 Desember 2023/ch24_20231218060000.mp4
```

Pemrosesan Video

Untuk setiap video dalam daftar, fungsi process_video dipanggil dengan parameter yang sesuai.

Output Frame

Frame hasil ekstraksi disimpan dengan format nama:

```
[nama_file_video]_frame_[nomor_frame].jpg
```

Contoh: ch24_20231218060000_frame_0.jpg.

Contoh Folder Data yang Diproses

Misalnya, terdapat folder berikut:

```
Data Rekaman CCTV/
└── 18 Desember 2023/
    └── ch24_20231218060000.mp4
```

Untuk memproses video ch24_20231218060000.mp4 dan menyimpan hasilnya ke folder Output Frames/, program dipanggil sebagai berikut:

```
video_path = "Data Rekaman CCTV/18 Desember
2023/ch24_20231218060000.mp4"
```

```
output_dir = "Output Frames/"
interval_minutes = 5


process_video(video_path, output_dir,
interval_minutes)
```

Setelah dijalankan, folder Output Frames/ akan berisi frame yang diekstraksi setiap 5 menit:

```
Output Frames/
└── ch24_20231218060000_frame_0.jpg
└──
ch24_20231218060000_frame_7500.jpg
└──
ch24_20231218060000_frame_15000.jpg
...
```

Dapat juga menambahkan lebih banyak video ke daftar video_files untuk diproses secara otomatis dalam satu kali eksekusi.

2.4. Anotasi Data dengan Roboflow

Proses anotasi data dapat dilakukan dengan Roboflow untuk potongan gambar yang memuat berbagai jenis kendaraan seperti motor, mobil, bus, dan truk melibatkan beberapa langkah utama. Berikut adalah penjelasan detailnya:

1. Persiapan Data

Mengumpulkan Dataset

Pastikan dataset terdiri dari potongan gambar yang memiliki beragam kendaraan. Dataset ini bisa berupa data yang diambil dari kamera lalu lintas atau sumber lain.

Format File

Dataset harus dalam format gambar yang umum seperti .jpg, .png, atau lainnya yang didukung oleh Roboflow. Contoh gambar dataset dapat dilihat pada gambar 4.

Gambar 4. *Contoh gambar dataset*

2. Unggah Dataset ke Roboflow

Login ke Roboflow

Buat akun di platform Roboflow atau login menggunakan akun yang sudah ada.

Buat Project Baru

- Klik opsi "Create New Project".
- Beri nama proyek, misalnya, "Deteksi Kendaraan".
- Pilih jenis anotasi, seperti "Object Detection".

Unggah Data

- Klik tombol "Upload Images".
- Pilih semua potongan gambar yang akan dianotasi.
- Tunggu hingga proses unggah selesai.

3. Anotasi Gambar

Pilih Gambar untuk Anotasi

Setelah gambar berhasil diunggah, pilih salah satu gambar dari daftar.

Buat Bounding Box

Gunakan alat anotasi di Roboflow untuk menggambar kotak di sekitar objek kendaraan pada gambar.

Setiap kotak harus menutupi satu kendaraan dengan ukuran yang proporsional.

Pilih Label

Pilih label sesuai jenis kendaraan, seperti Motor, Mobil, Bus, atau Truk.

Label dapat diatur secara manual atau dipilih dari daftar yang sudah dibuat.

Simetri dan Konsistensi

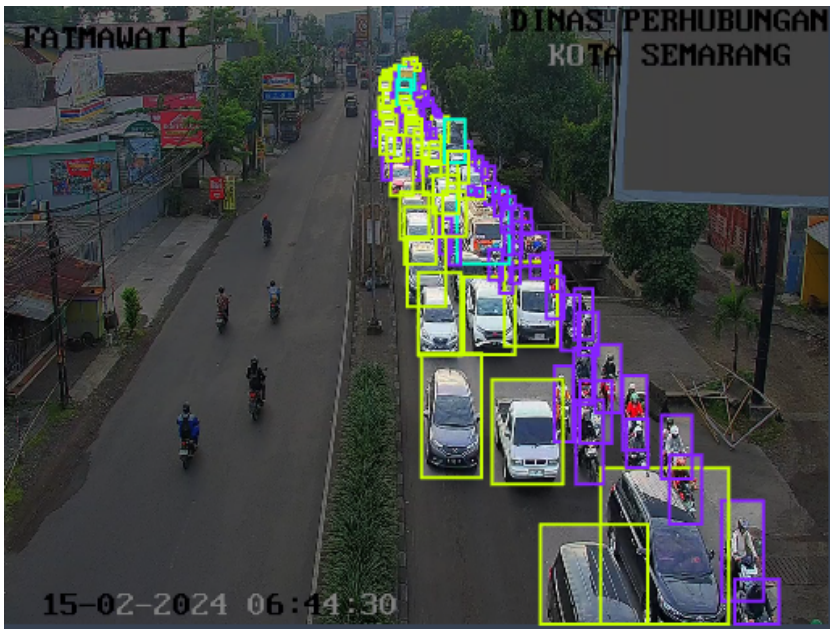
Pastikan kotak anotasi tidak terlalu besar atau kecil.

Usahakan untuk membuat anotasi dengan akurasi tinggi sehingga mencakup seluruh objek kendaraan tanpa latar belakang yang berlebihan.

Simpan Anotasi

Setelah selesai, simpan anotasi untuk gambar tersebut.

Lanjutkan proses ini untuk semua gambar dalam dataset. Contoh gambar yang telah dilakukan anotasi dapat dilihat pada gambar 5.



Gambar 5. *Contoh hasil anotasi gambar*

4. Validasi Anotasi

Periksa ulang semua anotasi untuk memastikan label dan bounding box sudah sesuai.

Lakukan revisi jika ada kesalahan atau inkonsistensi.

5. Ekspor Dataset

Pilih Format Ekspor

Setelah proses anotasi selesai, pilih format anotasi yang diinginkan, seperti YOLO, Pascal VOC, atau COCO, tergantung pada framework yang digunakan dalam model deteksi.

Unduh Dataset

Klik tombol ekspor untuk mengunduh dataset lengkap yang sudah dianotasi.

6. Augmentasi Data (Opsional)

Roboflow juga menyediakan fitur augmentasi untuk meningkatkan variasi dataset, seperti:

- Rotasi gambar.
- Flip horizontal atau vertikal.
- Penyesuaian pencahayaan.

Augmentasi ini dapat diatur sebelum ekspor dataset.

7. Integrasi dengan Framework

Dataset yang sudah dianotasi dapat langsung digunakan untuk pelatihan model menggunakan framework seperti YOLO, TensorFlow, atau PyTorch.

Dengan langkah-langkah ini, data akan siap untuk digunakan dalam proyek deteksi kendaraan, baik untuk pelatihan maupun pengujian model. Proses ini sangat penting untuk memastikan model dapat mengenali kendaraan secara akurat dalam berbagai kondisi.

BAB III Evolusi Model YOLO: Dari Konsep hingga Implementasi

Model YOLO (You Only Look Once) telah mengalami evolusi yang signifikan sejak pertama kali diperkenalkan sebagai salah satu algoritma deteksi objek paling efisien. Konsep utama YOLO adalah membagi gambar input menjadi grid dan memproses seluruh gambar sekaligus dalam satu langkah, berbeda dengan pendekatan tradisional yang memerlukan proses berbasis region proposal. Pendekatan ini memungkinkan YOLO untuk melakukan deteksi dengan kecepatan yang jauh lebih tinggi tanpa mengorbankan akurasi secara signifikan. Sebagai pelopor dalam deteksi objek real-time, YOLO telah menjadi fondasi penting bagi berbagai pengembangan di bidang computer vision.

3.1.YOLO: Sejarah Singkat

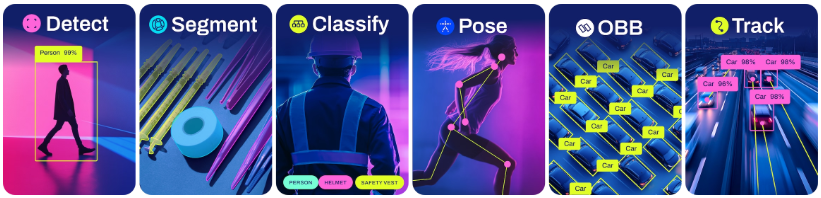
YOLO (You Only Look Once) adalah model deteksi objek dan segmentasi gambar yang populer, dikembangkan oleh Joseph Redmon dan Ali Farhadi di University of Washington. Diluncurkan pada tahun 2015, YOLO dengan cepat mendapatkan popularitas karena kecepatan dan akurasinya yang tinggi. Setelah peluncuran awal, YOLO terus mengalami pengembangan melalui berbagai versi sebagai berikut:

- YOLOv2 dirilis pada tahun 2016, meningkatkan model asli dengan menambahkan normalisasi batch, kotak jangkar (anchor boxes), dan klaster dimensi.
- YOLOv3, diluncurkan pada tahun 2018, lebih meningkatkan performa model dengan jaringan backbone yang lebih efisien, beberapa jangkar (multiple anchors), dan penggabungan piramida spasial.
- YOLOv4, dirilis pada tahun 2020, memperkenalkan inovasi seperti augmentasi data Mosaic, head deteksi tanpa jangkar (anchor-free), dan fungsi loss yang baru.
- YOLOv5 meningkatkan performa model lebih lanjut dengan menambahkan fitur seperti optimisasi hiperparameter, pelacakan eksperimen terintegrasi, dan ekspor otomatis ke berbagai format populer.
- YOLOv6 dibuat open-source oleh Meituan pada tahun 2022 dan banyak digunakan dalam robot pengiriman otonom perusahaan tersebut.
- YOLOv7 menambahkan tugas tambahan seperti estimasi pose pada dataset COCO keypoints.
- YOLOv8, dirilis pada tahun 2023 oleh Ultralytics, memperkenalkan fitur dan peningkatan baru untuk kinerja, fleksibilitas, dan efisiensi yang lebih baik, mendukung berbagai tugas AI penglihatan secara lengkap.
- YOLOv9 menghadirkan metode inovatif seperti Informasi Gradien yang Dapat Diprogram (Programmable Gradient Information/PGI) dan Generalized Efficient Layer Aggregation Network (GELAN).

- YOLOv10 dikembangkan oleh peneliti dari Universitas Tsinghua menggunakan paket Python dari Ultralytics. Versi ini menyediakan kemajuan deteksi objek real-time dengan memperkenalkan head End-to-End yang menghilangkan kebutuhan untuk Non-Maximum Suppression (NMS).
- YOLOv11 BARU: Model YOLO terbaru dari Ultralytics menawarkan performa mutakhir (state-of-the-art/SOTA) untuk berbagai tugas, termasuk deteksi, segmentasi, estimasi pose, pelacakan, dan klasifikasi. Model ini memanfaatkan kemampuan untuk berbagai aplikasi dan domain AI yang beragam.

3.2.Ultralytics YOLO

Ultralytics YOLO merupakan pengembangan terbaru dalam seri YOLO (You Only Look Once) yang diakui untuk deteksi objek dan segmentasi gambar secara real-time. Seri ini dibangun berdasarkan versi sebelumnya dengan memperkenalkan fitur dan penyempurnaan baru untuk meningkatkan kinerja, fleksibilitas, dan efisiensi. YOLO mendukung berbagai tugas AI penglihatan seperti deteksi, segmentasi, estimasi pose, pelacakan, dan klasifikasi. Ilutrasi model YOLO dapat dilihat pada gambar 6. Arsitekturnya yang canggih memastikan kecepatan dan akurasi yang unggul, sehingga cocok untuk berbagai aplikasi, termasuk perangkat edge dan API cloud.



Gambar 6. *Ilustrasi model YOLO*

Deteksi objek telah mengalami perkembangan yang pesat selama bertahun-tahun, mulai dari teknik visi komputer tradisional hingga model deep learning yang canggih. Keluarga model YOLO (You Only Look Once) menjadi salah satu pelopor dalam perkembangan ini, terus mendorong batas kemampuan dalam deteksi objek secara real-time.

Pendekatan unik YOLO memperlakukan deteksi objek sebagai masalah regresi tunggal, yaitu dengan memprediksi posisi kotak pembatas (bounding box) dan probabilitas kelas langsung dari seluruh gambar dalam satu proses evaluasi. Metode revolusioner ini membuat model YOLO jauh lebih cepat dibandingkan detektor dua tahap sebelumnya, sambil tetap menjaga tingkat akurasi yang tinggi.

Setiap versi baru YOLO selalu menghadirkan peningkatan arsitektur dan teknik inovatif yang meningkatkan kinerja dalam berbagai aspek. YOLO11 meneruskan tradisi ini dengan mengadopsi berbagai kemajuan terbaru dalam riset visi komputer, sehingga mampu memberikan keseimbangan yang lebih baik antara kecepatan dan akurasi untuk berbagai kebutuhan di dunia nyata.

3.3.Cara Memulai Instalasi dan Pengaturan YOLO

Untuk menginstal YOLO menggunakan Ultralytics, jalankan perintah berikut di terminal atau command prompt:

```
pip install ultralytics
```

Setelah instalasi, verifikasi penginstalan dengan menjalankan:

```
yolo help
```

Perintah tersebut akan menampilkan daftar opsi yang tersedia untuk CLI YOLO.

Antarmuka baris perintah (CLI) Ultralytics memungkinkan perintah satu baris sederhana tanpa memerlukan lingkungan Python. CLI tidak memerlukan kustomisasi atau kode Python, cukup menjalankan semua tugas dari terminal dengan perintah yolo. Panduan CLI untuk mempelajari lebih lanjut tentang penggunaan YOLO adalah sebagai berikut:

Syntax

```
yolo TASK MODE ARGS
```

TASK (opsional): Salah satu dari (detect, segment, classify, pose, obb).

MODE (wajib): Salah satu dari (train, val, predict, export, track, benchmark).

ARGS (opsional): Pasangan argumen dalam format arg=value, seperti imgsz=640, untuk mengganti nilai default.

Train

Untuk melatih model deteksi selama 10 epoch dengan nilai awal learning_rate sebesar 0.01, gunakan perintah berikut:

```
yolo train data=coco8.yaml
model=yolo11n.pt epochs=10 lr0=0.01
```

Penjelasan:

- data=coco8.yaml: Dataset yang digunakan adalah coco8.
- model=yolo11n.pt: Model yang digunakan adalah yolo11n.pt.
- epochs=10: Model akan dilatih selama 10 epoch.
- lr0=0.01: Nilai awal untuk learning_rate ditetapkan ke 0.01.

Predict

Untuk melakukan prediksi pada sebuah video YouTube menggunakan model segmentasi yang telah dilatih sebelumnya dengan ukuran gambar 320, gunakan perintah berikut:

```
yolo predict model=yolo11n-seg.pt
source='https://youtu.be/LNwODJXcvt4
' imgsz=320
```

Penjelasan:

- model=yolo11n-seg.pt: Model segmentasi pretrained yang digunakan adalah yolo11n-seg.pt.
- source='https://youtu.be/LNwODJXcvt4': Sumber input adalah URL video YouTube.
- imgsz=320: Ukuran gambar input yang digunakan adalah 320.

Val

Gunakan perintah berikut untuk melakukan validasi model deteksi pretrained:

```
yolo val model=yolo11n.pt data=coco8.yaml batch=1 imgsz=640
```

Penjelasan:

- model=yolo11n.pt: Model deteksi pretrained yang divalidasi adalah yolo11n.pt.
- data=coco8.yaml: Dataset yang digunakan untuk validasi adalah coco8.
- batch=1: Ukuran batch untuk validasi adalah 1.
- imgsz=640: Ukuran gambar input untuk validasi adalah 640.

Ekspor

Gunakan perintah berikut untuk mengekspor model klasifikasi ke format ONNX:

```
yolo export model=yolo11n-cls.pt format=onnx imgsz=224,128
```

Penjelasan:

- model=yolo11n-cls.pt: Model klasifikasi pretrained yang akan diekspor adalah yolo11n-cls.pt.
- format=onnx: Format ekspor yang digunakan adalah ONNX.
- imgsz=224,128: Ukuran gambar input adalah 224x128.

Perintah khusus

Berikut adalah beberapa perintah khusus untuk memeriksa versi, melihat pengaturan, menjalankan pemeriksaan, dan lainnya:

```
yolo help
yolo checks
yolo version
yolo settings
yolo copy-cfg
yolo cfg
```

3.4.Cara Melatih Model YOLO Kustom pada Dataset

Mengapa Memilih Ultralytics YOLO untuk Training:

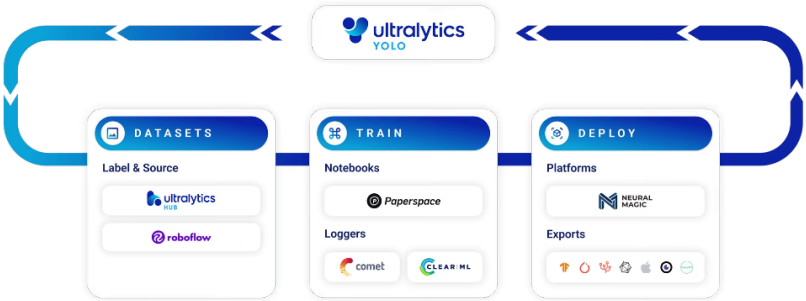
1. **Efisiensi**
   Memanfaatkan kemampuan perangkat keras modern secara optimal, baik untuk GPU tunggal maupun multi-GPU.
2. **Fleksibilitas**
   Mendukung pelatihan pada dataset kustom selain dataset populer seperti COCO, VOC, dan ImageNet.

3. **Kemudahan Penggunaan**
   Antarmuka CLI dan Python yang sederhana namun kuat memudahkan proses pelatihan.
4. **Hyperparameter yang Dapat Dikustomisasi**
   Mendukung pengaturan hyperparameter yang luas untuk meningkatkan kinerja model.

Melatih model YOLO kustom pada dataset melibatkan beberapa langkah:

1. Mempersiapkan dataset yang telah dianotasi.
2. Mengonfigurasi parameter pelatihan dalam file YAML.
3. Menggunakan perintah yolo TASK train untuk memulai pelatihan (setiap TASK memiliki argumen tersendiri).

Model Training dengan Ultralytics YOLO daspat dilihat pada gambar 7.



Gambar 7. *Model Training dengan Ultralytics YOLO*

Fitur Utama Train Mode

1. **Pengunduhan Dataset Otomatis**

Dataset seperti COCO, VOC, dan ImageNet diunduh secara otomatis saat pertama kali digunakan.
Contoh:

```
yolo train data=coco.yaml
```

2. **Dukungan Multi-GPU**
   Pelatihan dapat ditingkatkan dengan mulus menggunakan beberapa GPU untuk mempercepat proses.
3. **Konfigurasi Hyperparameter**
   Hyperparameter dapat disesuaikan melalui file konfigurasi YAML atau argumen CLI.
4. **Visualisasi dan Monitoring**
   Pelacakan metrik pelatihan secara real-time untuk mendapatkan wawasan yang lebih baik tentang proses pembelajaran.

Contoh Pelatihan Model YOLO11n pada Dataset COCO8

Melatih model pada dataset COCO8 selama 100 epoch dengan ukuran gambar 640. Perangkat pelatihan dapat ditentukan dengan argumen device. Jika tidak disebutkan, perangkat secara otomatis akan menggunakan:

- GPU (device=0) jika tersedia.
- CPU jika GPU tidak ada.

Menggunakan Python

Berikut contoh pelatihan menggunakan Python:

```
from ultralytics import YOLO
```

```
# Memuat model

model = YOLO("yolo11n.yaml")  # Membuat model baru dari YAML

model = YOLO("yolo11n.pt")  # Memuat model pretrained (disarankan untuk pelatihan)

model = YOLO("yolo11n.yaml").load("yolo11n.pt")  # Membuat model dari YAML dan transfer bobot

# Melatih model

results = model.train(data="coco8.yaml", epochs=100, imgsz=640)
```

Menggunakan CLI

Gunakan perintah berikut untuk melatih model:

```
yolo train data=coco8.yaml model=yolo11n.pt epochs=100 imgsz=640
```

Training Settings

Pengaturan pelatihan untuk model YOLO mencakup berbagai hiperparameter dan konfigurasi yang digunakan selama proses pelatihan. Pengaturan ini memengaruhi performa, kecepatan, dan akurasi model. Pengaturan pelatihan utama meliputi ukuran batch size, learning rate, momentum, dan weight decay. Selain itu,

pilihan choice of optimizer, loss function, dan training dataset dapat memengaruhi proses pelatihan. Penyetelan dan eksperimen yang cermat dengan pengaturan ini sangat penting untuk mengoptimalkan performa.

3.5.Penggunaan YOLO Ultralytics untuk Pelacakan Objek Real-Time

Ultralytics YOLO mendukung pelacakan multi-objek yang efisien dan fleksibel. Pelacakan Multi-Objek dengan Ultralytics YOLO dapat dilihat pada gambar 8.

Gambar 8. *Pelacakan Multi-Objek dengan Ultralytics YOLO*

Pelacakan objek dalam analitik video adalah tugas penting yang tidak hanya mengidentifikasi lokasi dan kelas objek dalam sebuah frame, tetapi juga mempertahankan ID unik untuk setiap objek yang terdeteksi seiring berjalannya video. Aplikasinya sangat beragam mulai dari pengawasan dan keamanan hingga analitik olahraga secara real-time.

Keluaran dari pelacak Ultralytics konsisten dengan deteksi objek standar tetapi memiliki nilai tambah berupa ID objek. Hal ini mempermudah pelacakan

objek dalam aliran video dan analitik lanjutan. Berikut alasan menggunakan Ultralytics YOLO untuk kebutuhan pelacakan objek:

- Efisiensi: Memproses aliran video secara real-time tanpa mengorbankan akurasi.
- Fleksibilitas: Mendukung berbagai algoritma pelacakan dan konfigurasi.
- Kemudahan Penggunaan: API Python sederhana dan opsi CLI untuk integrasi serta penerapan cepat.
- Kustomisasi: Mudah digunakan dengan model YOLO yang telah dilatih secara khusus, memungkinkan integrasi ke dalam aplikasi spesifik domain.

Fitur Sekilas

Ultralytics YOLO memperluas fitur deteksi objeknya untuk menyediakan pelacakan objek yang kuat dan serbaguna:

Pelacakan Real-Time: Melacak objek dengan mulus dalam video dengan frame rate tinggi.

Dukungan Multi-Pelacak: Memilih dari berbagai algoritma pelacakan yang sudah ada.

Konfigurasi Pelacak yang Dapat Disesuaikan: Menyesuaikan algoritma pelacakan dengan kebutuhan tertentu melalui penyesuaian berbagai parameter.

Pelacak yang Tersedia

Ultralytics YOLO mendukung algoritma pelacakan berikut. Algoritma ini dapat diaktifkan dengan memberikan file konfigurasi YAML terkait, seperti tracker=tracker_type.yaml:

- BoT-SORT: Gunakan botsort.yaml untuk mengaktifkan pelacak ini.
- ByteTrack: Gunakan bytetrack.yaml untuk mengaktifkan pelacak ini.

Pelacak default adalah BoT-SORT.

Untuk menjalankan pelacak pada aliran video, gunakan model Detect, Segment, atau Pose yang telah dilatih, seperti YOLO11n, YOLO11n-seg, dan YOLO11n-pose.

Python:

```
from ultralytics import YOLO


# Memuat model resmi atau model khusus
model = YOLO("yolo11n.pt")   # Model Detect resmi
model = YOLO("yolo11n-seg.pt")    # Model Segment resmi
model = YOLO("yolo11n-pose.pt")    # Model Pose resmi
model = YOLO("path/to/best.pt")    # Model yang dilatih secara khusus
```

```
# Melakukan pelacakan dengan model
results = model.track("https://youtu.be/LNwODJXcvt4", show=True)  # Pelacakan dengan pelacak default
results = model.track("https://youtu.be/LNwODJXcvt4", show=True, tracker="bytetrack.yaml")  # Menggunakan ByteTrack
```

CLI:

```
# Melakukan pelacakan dengan berbagai model menggunakan antarmuka baris perintah
yolo track model=yolo11n.pt source="https://youtu.be/LNwODJXcvt4"  # Model Detect resmi
yolo track model=yolo11n-seg.pt source="https://youtu.be/LNwODJXcvt4"  # Model Segment resmi
yolo track model=yolo11n-pose.pt source="https://youtu.be/LNwODJXcvt4"  # Model Pose resmi
yolo track model=path/to/best.pt source="https://youtu.be/LNwODJXcvt4"  # Model yang dilatih secara khusus
```

```
# Melacak menggunakan pelacak ByteTrack
yolo track model=path/to/best.pt tracker="bytetrack.yaml"
```

Contoh Python: Melacak Objek di Video

1. Mempertahankan Lintasan Objek

```
import cv2
from ultralytics import YOLO

# Memuat model YOLO11
model = YOLO("yolo11n.pt")

# Membuka file video
video_path = "path/to/video.mp4"
cap = cv2.VideoCapture(video_path)

# Melakukan pelacakan objek pada frame video
while cap.isOpened():
    success, frame = cap.read()
    if success:
```

```
        results = model.track(frame,
persist=True)
        annotated_frame             =
results[0].plot()
        cv2.imshow("YOLO11 Tracking",
annotated_frame)
        if cv2.waitKey(1) & 0xFF ==
ord("q"):
            break
    else:
        break
cap.release()
cv2.destroyAllWindows()
```

2. Visualisasi Lintasan Objek

```
from collections import defaultdict
import cv2
import numpy as np
from ultralytics import YOLO

model = YOLO("yolo11n.pt")
video_path = "path/to/video.mp4"
cap = cv2.VideoCapture(video_path)
```

```
track_history = defaultdict(list)


while cap.isOpened():
    success, frame = cap.read()
    if success:
        results = model.track(frame,
persist=True)
        boxes                          =
results[0].boxes.xywh.cpu()
        track_ids                      =
results[0].boxes.id.int().cpu().toli
st()
        annotated_frame                =
results[0].plot()
        for    box,    track_id    in
zip(boxes, track_ids):
            x, y, w, h = box

track_history[track_id].append((floa
t(x), float(y)))
            if
len(track_history[track_id]) > 30:
```

```
track_history[track_id].pop(0)
            points                    =
np.array(track_history[track_id]).as
type(int).reshape((-1, 1, 2))


cv2.polylines(annotated_frame,
[points], isClosed=False, color=(230,
230, 230), thickness=2)
        cv2.imshow("YOLO11 Tracking",
annotated_frame)
        if  cv2.waitKey(1)  &  0xFF  ==
ord("q"):
            break
    else:
        break
cap.release()
cv2.destroyAllWindows()
```

BAB IV Penerapan Model YOLO untuk Deteksi Kendaraan

4.1. Model YOLO untuk Deteksi Kendaraan di Lalu Lintas

Model YOLO (You Only Look Once) telah menjadi salah satu algoritma terdepan dalam deteksi kendaraan pada berbagai aplikasi transportasi cerdas. Dengan kemampuan untuk mendeteksi objek secara real-time, YOLO sangat efektif dalam menangani kompleksitas lalu lintas perkotaan, di mana berbagai jenis kendaraan bergerak secara dinamis. Pendekatan YOLO yang memproses gambar secara keseluruhan memungkinkan identifikasi cepat kendaraan seperti mobil, sepeda motor, bus, dan truk, bahkan dalam kondisi lalu lintas padat atau pencahayaan yang tidak ideal.

Dalam penerapannya, YOLO digunakan secara luas dalam sistem pemantauan lalu lintas berbasis kamera CCTV. Kamera yang dipasang di persimpangan atau jalan utama dilengkapi dengan model YOLO untuk mendeteksi jumlah kendaraan, jenis kendaraan, serta kecepatan mereka. Data yang dihasilkan tidak hanya digunakan untuk menganalisis pola lalu lintas, tetapi juga untuk mengatur sinyal lampu lalu lintas secara adaptif guna mengurangi kemacetan. Teknologi ini juga mendukung inisiatif smart city dengan memberikan wawasan real-time bagi otoritas transportasi dalam membuat keputusan yang lebih baik.

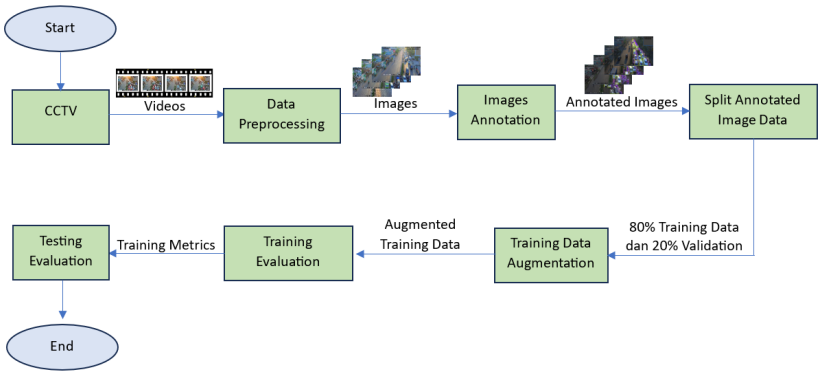
Selain itu, YOLO sering diintegrasikan dalam sistem kendaraan otonom untuk mendeteksi dan menghindari tabrakan. Dengan algoritma YOLO, kendaraan dapat mengenali objek di sekitar seperti kendaraan lain, pejalan kaki, atau rambu lalu lintas dalam hitungan milidetik. Informasi ini digunakan oleh sistem kendali untuk menentukan tindakan yang sesuai, seperti memperlambat laju kendaraan, berhenti, atau mengubah jalur. Efisiensi dan akurasi YOLO membuatnya menjadi komponen penting dalam pengembangan teknologi kendaraan otonom yang aman dan handal.

Penerapan YOLO juga mencakup analisis video untuk penelitian dan perencanaan infrastruktur transportasi. Misalnya, dengan menggunakan YOLO pada data video udara dari drone, pemerintah dapat mengidentifikasi pola kemacetan dan merancang jalur baru yang lebih efisien. YOLO juga dapat digunakan untuk sistem penegakan hukum, seperti mendeteksi pelanggaran lalu lintas secara otomatis, termasuk kendaraan yang melanggar batas kecepatan atau melanggar aturan parkir. Dengan fleksibilitas dan efisiensinya, penerapan model YOLO telah membantu membentuk solusi transportasi yang lebih cerdas, aman, dan berkelanjutan.

4.2. Model YOLOv8n untuk Deteksi Real-Time Kendaraan di Lalu Lintas

Alur kerja model

Diagram di bawah ini menggambarkan setiap tahap, mulai dari akuisisi data mentah hingga evaluasi model, dengan penekanan pada penggunaan teknik multi-augmentasi untuk meningkatkan kinerja model YOLOv8n. Diagram alur kerja ditunjukkan pada Gambar 9.



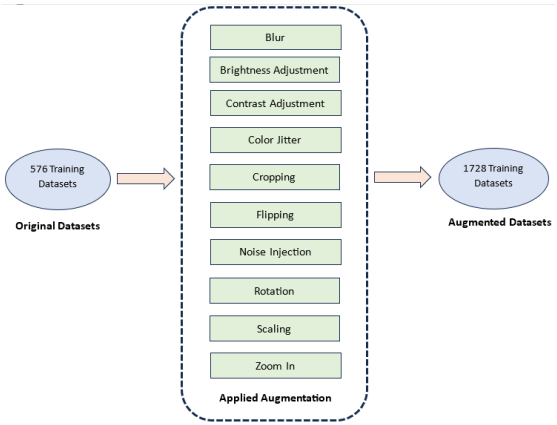
Gambar 9. *Alur kerja deteksi kendaraan menggunakan YOLO8n dan Teknik Multi Augmentasi*

Gambar 9 menunjukkan alur kerja untuk deteksi kendaraan di lalu lintas menggunakan model YOLO yang ditingkatkan dengan teknik multi-augmentasi. Proses dimulai dengan pengambilan video dari kamera CCTV yang merekam lalu lintas kendaraan untuk analisis. Video kemudian diproses pada tahap Prapemrosesan Data untuk persiapan anotasi. Frame diperoleh dengan memisahkan rekaman video menjadi frame setiap interval 5 menit selama tahap Prapemrosesan Data. Setiap kendaraan dalam frame kemudian dianotasi berdasarkan jenisnya, termasuk informasi detail seperti lokasi dan koordinat bounding box. Hal ini memastikan data siap untuk analisis lebih

lanjut atau pelatihan model. Setelah anotasi, data dibagi menjadi 80% untuk pelatihan dan 20% untuk validasi, memastikan model dilatih dan dievaluasi dengan data yang belum pernah dilihat sebelumnya.

Teknik Augmentasi

Selanjutnya, berbagai teknik augmentasi diterapkan pada dataset untuk meningkatkan variasi data dan kinerja model. Teknik augmentasi yang diterapkan meliputi blur, penyesuaian kecerahan, penyesuaian kontras, perubahan warna, pemotongan, flipping, injeksi noise, rotasi, scaling, dan zoom-in. Teknik augmentasi yang diterapkan dijelaskan pada Gambar 10.



Gambar 10. *Teknik Augmentasi yang Diterapkan*

Gambar 10 menunjukkan bahwa setiap teknik augmentasi melipatgandakan ukuran dataset asli. Sebagai hasilnya, dari 576 frame gambar asli, dihasilkan 1.728 frame hasil augmentasi. Setiap teknik augmentasi berkontribusi untuk memperluas dataset

sambil mensimulasikan kondisi dunia nyata seperti perubahan pencahayaan, variasi warna, sudut pandang berbeda, dan gangguan visual. Hal ini memungkinkan model belajar mengenali kendaraan dalam berbagai skenario dan kondisi pencahayaan, meningkatkan akurasi dan ketahanan dalam mendeteksi kendaraan di lingkungan nyata. Augmentasi sangat penting untuk membuat model lebih tahan terhadap variasi data di dunia nyata. Nilai-nilai teknik augmentasi ditampilkan pada Tabel 1.

Tabel 1. *Nilai-nilai teknik augmentasi yang digunakan*

No	**Aug**	**Value**	**Augmentation Factor (Image)**		
			1	**2**	**3**
1	Blur	Kernel Size	-	1	2
2	Brightness Adjustment	Brightness Factor	-	0.8	1.2
3	Contrast Adjustment	Alpha	-	1.5	2.0
4	Color Jitter	(Brightness, contrast, saturation) and hue	-	Rand(0.6, 1.4) and Rand(-0.1,0.1)	Rand(0.6, 1.4) and Rand(-0.1,0.1)

5	Croppin g	Crop height fraction	-	Top Crop (0,0) to (width, height * 0.5)	Bottom Crop (0, height * 0.5) to (width, height)
6	Flipping	Horizont al and Vertical Flip	-	Horizonta l Flip x_center = 1.0 - x_center	Vertical Flip y_center = 1.0 - y_center
7	Noise Injection	Gaussian Noise	-	Rand(0,0. 1)	Rand(0,0. 1)
8	Rotation	Rotation	-	90’	270’
9	Scaling	Scale Image	-	Rand(0.8, 1.2)	Rand(0.8, 1.2)
10	Zoom In	Zoom In	-	1.2	1.5

Setiap teknik augmentasi dari Tabel 1 menggunakan nilai spesifik untuk menghasilkan variasi gambar. Teknik blur menggunakan ukuran kernel sebesar 1 untuk Gambar 2 dan 2 untuk Gambar 3. Untuk penyesuaian kecerahan, faktor kecerahan yang digunakan adalah 0,8 untuk Gambar 2 dan 1,2 untuk Gambar 3. Penyesuaian kontras menggunakan nilai alpha sebesar 1,5 untuk Gambar 2 dan 2,0 untuk Gambar 3. Dalam penyesuaian Color Jitter, kombinasi faktor kecerahan, kontras, dan saturasi diacak dalam rentang 0,6 hingga 1,4, sedangkan faktor hue diacak antara -0,1 dan 0,1. Pemotongan gambar dilakukan

dengan dua jenis pemotongan: pemotongan atas dari (0,0) hingga (lebar, tinggi * 0,5) pada Gambar 2, dan pemotongan bawah dari (0, tinggi * 0,5) hingga (lebar, tinggi) pada Gambar 3. Teknik flipping mencakup flipping horizontal dan vertikal, dengan titik pusat flipping horizontal di x_center = 1,0 untuk Gambar 2 dan -x_center untuk flipping vertikal pada Gambar 3.

Penyuntikan noise menambahkan noise Gaussian dengan nilai yang diacak antara 0 dan 0,1 untuk Gambar 2 dan Gambar 3. Rotasi diterapkan sebesar 90 derajat untuk Gambar 2 dan 270 derajat untuk Gambar 3. Scaling diterapkan dengan faktor antara 0,8 dan 1,2 untuk Gambar 2 dan Gambar 3, sementara zoom-in menggunakan faktor 1,2 untuk Gambar 2 dan 1,5 untuk Gambar 3. Setiap nilai dirancang dengan cermat untuk menciptakan variasi gambar yang signifikan, sehingga meningkatkan kinerja model dalam berbagai kondisi pengenalan kendaraan. Contoh keluaran dari setiap teknik augmentasi data menggunakan nilai kedua dapat dilihat pada Gambar 11.

(1) Without Augmentation

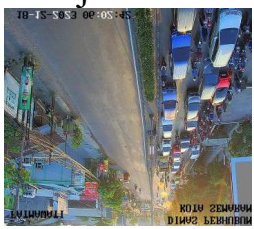
(2) Blur

(3) Brightness Adjustment

(4) Contrast Adjustment

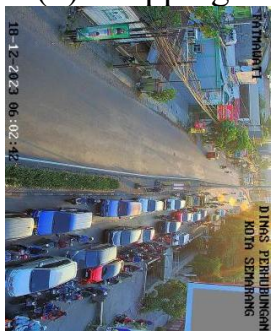
(5) Color Jitter

(6) Cropping



(7) Flipping

(8) Gaussian Noise



(9) Rotation

(10) Scaling

(11) Zoom In

Gambar 11. *Contoh Output Augmentasi Data*

Teknk Multi Augmentasi

Penggunaan multi-augmentasi dalam pemrosesan dataset citra sangat penting untuk memastikan model pembelajaran mesin dapat beradaptasi dengan berbagai situasi dan kondisi dalam aplikasi dunia nyata. Augmentasi citra bertujuan untuk meningkatkan keragaman dataset tanpa perlu mengumpulkan data asli tambahan. Dengan menerapkan berbagai teknik

augmentasi secara bersamaan, model dapat dilatih untuk mengenali objek dalam kondisi pencahayaan, ukuran, orientasi, dan kualitas gambar yang bervariasi. Hal ini meningkatkan ketahanan model dalam mendeteksi objek yang mungkin sulit dikenali dalam kondisi normal. Kombinasi teknik augmentasi yang digunakan dalam penelitian ini dirancang untuk mengatasi berbagai aspek penting dalam pengenalan objek. Teknik Multi-Augmentasi yang diterapkan dalam penelitian ini ditampilkan pada Tabel 2.

Tabel 2. *Multi Augmentasi*

No	Augmentation Technique Combination	Focus
1	Scaling + Cropping + Brightness Adjustment + Noise Injection + Blur	Small Objects and Lighting
2	Rotation + Flipping + Brightness Adjustment + Contrast Adjustment + Color Jitter	Angle and Lighting Variation
3	Scaling + Zoom In + Brightness Adjustment + Color Jitter + Noise Injection	Size and Color Variation
4	Cropping + Zoom In + Contrast Adjustment + Noise Injection + Blur	Detail Enhancement
5	Scaling + Rotation + Brightness Adjustment + Contrast Adjustment + Noise Injection	Complete Combination for Robustness

Tabel 2 menunjukkan bahwa kombinasi pertama berfokus pada objek kecil dan pencahayaan, sementara kombinasi kedua menangani variasi sudut dan pencahayaan. Kombinasi ketiga menargetkan variasi ukuran dan warna, sedangkan kombinasi keempat berfokus pada peningkatan detail. Kombinasi terakhir mencakup semua teknik augmentasi untuk memastikan ketahanan secara keseluruhan. Secara matematis, jika X adalah dataset asli, maka setelah menerapkan kombinasi augmentasi, dataset baru Y dapat direpresentasikan sebagai $Y = \sum_{i-1}^{n} A_i(X)$, di mana A_i adalah teknik augmentasi ke-i yang diterapkan pada dataset X. Hasilnya adalah dataset yang lebih besar dan lebih bervariasi, sehingga meningkatkan kemampuan model untuk melakukan generalisasi.

Model YOLO

YOLO banyak digunakan dalam sistem transportasi karena akurasi deteksi yang tinggi dan kinerja real-time yang andal. Dua versi terbaru YOLO, yaitu YOLOv8 dan YOLOv9, dirancang untuk memenuhi berbagai kebutuhan aplikasi, terutama dalam deteksi objek secara real-time. YOLOv8 memperkenalkan lima versi berskala yang disesuaikan dengan kebutuhan aplikasi yang berbeda: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, dan YOLOv8x. Model YOLOv8n menunjukkan kecepatan pemrosesan yang mengesankan, dengan waktu eksekusi 80,4 ms pada CPU menggunakan ONNX dan 0,99 ms pada GPU A100 menggunakan TensorRT. Model ini dirancang untuk mendeteksi objek dari dataset COCO yang

mencakup 80 kelas, seperti mobil, sepeda motor, bus, dan truk. Berkat kecepatan dan efisiensinya, YOLOv8n sangat cocok untuk aplikasi yang membutuhkan deteksi objek secara real-time, seperti pemantauan lalu lintas atau sistem pengawasan dengan sumber daya komputasi terbatas. Hal ini menjadikannya ideal untuk implementasi real-time pada perangkat dengan daya komputasi terbatas.

Serupa dengan YOLOv8, YOLOv9 juga memperkenalkan lima versi berskala untuk memenuhi berbagai kebutuhan aplikasi: YOLOv9t, YOLOv9s, YOLOv9m, YOLOv9c, dan YOLOv9e. Model YOLOv9t menawarkan kecepatan pemrosesan tinggi dengan 2 juta parameter (M) dan 7,7 miliar FLOPs. Model ini juga dirancang untuk mendeteksi objek dari dataset COCO yang mencakup 80 kelas objek seperti mobil, sepeda motor, bus, dan truk. Dengan skor mAPval 50-95 sebesar 38,3%, YOLOv9t sangat cocok untuk deteksi objek secara real-time di lingkungan dengan sumber daya terbatas, meskipun akurasinya sedikit lebih rendah dibandingkan model yang lebih besar. Namun, kecepatan dan efisiensi model ini tetap menjadikannya pilihan ideal untuk aplikasi dengan kebutuhan pemrosesan tinggi.

Di antara versi berskala YOLOv8 dan YOLOv9, penelitian ini menggunakan YOLOv8n dan YOLOv9t, dengan fokus pada model paling ringan yang optimal untuk implementasi real-time, namun tetap memberikan akurasi yang memadai. Parameter default

YOLO yang disesuaikan selama pengujian model dapat dilihat pada Tabel 3.

Tabel 3. *Parameter yang Disesuaikan*

Parameter	**Nilai**
Batch	8
Patience	10
Dropout	0.2
Image Size	640

Ukuran batch default pada YOLO dikurangi dari 64 menjadi 8, karena ukuran batch yang lebih besar dapat meningkatkan risiko overfitting, terutama pada gambar yang buram atau terlalu terang. Ukuran patience diubah dari 5 menjadi 10 untuk memastikan pelatihan tidak berhenti terlalu cepat. Nilai dropout yang awalnya 0,0 diubah menjadi 0,2 untuk mengurangi overfitting dan meningkatkan ketahanan model. Ukuran gambar diatur menjadi 640x640 piksel, karena ukuran ini dianggap seimbang antara kemampuan mendeteksi objek kecil secara efektif dan mempertahankan kecepatan inferensi tinggi. Tahapan analisis menggunakan YOLO dapat dilihat pada Gambar 12.

Gambar 12. *Tahapan Analisis Model YOLO*

Dataset akhir, seperti yang ditunjukkan pada Gambar 5, telah melalui beberapa tahapan penting sebelum

digunakan untuk pelatihan model. Tahapan ini mencakup pra-pemrosesan, pelabelan, anotasi, augmentasi data, dan pembagian data menjadi set pelatihan dan validasi. Tahap pelatihan model YOLO menggunakan 80% dari dataset akhir, dengan proses pelatihan dilakukan selama 64 epoch menggunakan parameter yang telah ditentukan dalam Tabel 2. Selanjutnya, tahap validasi dilakukan dengan menggunakan 20% dataset untuk mengevaluasi kinerja model berdasarkan metrik seperti presisi, recall, dan mAP. Proses ini bertujuan untuk mengukur kemampuan model mendeteksi objek dengan akurat. Akhirnya, pengujian dilakukan menggunakan 10 frame yang dipilih berdasarkan kondisi tertentu untuk menilai keandalan model terbaik yang diperoleh selama tahap validasi. Pengujian ini memastikan bahwa model dapat bekerja dengan baik dalam skenario dunia nyata dan memberikan hasil yang konsisten.

Pengujian model YOLO

Tahap berikutnya melibatkan pengujian model YOLO, menggunakan versi YOLOv8n dan YOLOv9t yang dikombinasikan dengan teknik augmentasi yang diterapkan. Pengujian ini mengevaluasi kinerja model dalam mendeteksi kendaraan pada video lalu lintas. Proses dimulai dengan mengevaluasi hasil pelatihan menggunakan metrik pelatihan untuk menganalisis seberapa baik model telah dilatih. Selanjutnya, dilakukan evaluasi pelatihan untuk menilai efektivitas model berdasarkan data pelatihan. Setelah evaluasi pelatihan, model diuji pada data dengan berbagai

kondisi jalan, seperti yang ditunjukkan pada Tabel 5. Hasil diukur menggunakan metrik pengujian untuk menentukan kemampuan model dalam mendeteksi kendaraan pada video lalu lintas. Tahap akhir adalah evaluasi pengujian, di mana hasil pengujian dinilai untuk menentukan akurasi dan kinerja keseluruhan model, termasuk mengidentifikasi area yang perlu ditingkatkan. Diagram ini menggambarkan seluruh proses, mulai dari akuisisi data mentah hingga evaluasi model dalam deteksi kendaraan, dengan penekanan pada penggunaan teknik augmentasi untuk meningkatkan kinerja model YOLO.

Dalam penelitian ini, pelatihan dan validasi dilakukan menggunakan 32 skenario yang melibatkan model YOLOv8n dan YOLOv9t. Data pelatihan mencakup 1 dataset tanpa augmentasi, 10 dataset dengan augmentasi tunggal, dan 5 dataset dengan augmentasi ganda. Jumlah sampel pelatihan yang digunakan adalah 576 untuk dataset tanpa augmentasi, 1728 untuk dataset dengan augmentasi tunggal, dan 6336 untuk dataset dengan augmentasi ganda. Untuk validasi, digunakan 144 sampel, yang merupakan 20% dari dataset awal. Validasi dilakukan dengan mengukur beberapa metrik, termasuk Precision, Recall, mAP50, dan mAP50-95. Di antara metrik-metrik ini, mAP50-95 dianggap sebagai metrik utama untuk menentukan performa terbaik karena mencakup evaluasi pada berbagai nilai IoU, sehingga memberikan penilaian yang lebih komprehensif terhadap kinerja model dalam berbagai skenario.

Metrik kinerja model yang diuji disajikan dalam Tabel 4, yang menggambarkan kinerja setiap model beserta augmentasi yang diterapkan.

Tabel 4. *Perbandingan Metrik Kinerja Model yang Diuji*

No	Augmentation	Models	Best Epoch	mAP50	mAP50-95
1	Without Augmentation	YOLOv8n	64	0.673	0.390
2	Without Augmentation	YOLOv9t	64	0.669	0.378
3	Blur	YOLOv8n	64	0.767	0.465
4	Blur	YOLOv9t	64	0.746	0.443
5	Brightness Adjustment	YOLOv8n	64	0.757	0.462
6	Brightness Adjustment	YOLOv9t	62	0.749	0.449
7	Contrast Adjustment	YOLOv8n	49	0.325	0.203

8	Contrast Adjustment	YOLOv9t	10	0.280	0.167
9	Color Jitter	YOLOv8n	61	0.755	0.461
10	Color Jitter	YOLOv9t	46	0.713	0.411
11	Cropping	YOLOv8n	60	0.646	0.356
12	Cropping	YOLOv9t	64	0.635	0.349
13	Flipping	YOLOv8n	64	0.732	0.436
14	Flipping	YOLOv9t	62	0.719	0.411
15	Noise Injection	YOLOv8n	51	0.739	0.433
16	Noise Injection	YOLOv9t	64	0.715	0.422
17	Rotation	YOLOv8n	62	0.663	0.371
18	Rotation	YOLOv9t	63	0.653	0.366
19	Scaling	YOLOv8n	61	0.753	0.464
20	Scaling	YOLOv9t	64	0.741	0.459
21	Zoom-In	YOLOv8n	64	0.722	0.426
22	Zoom-In	YOLOv9t	64	0.716	0.421
23	Scaling + Cropping + Brightness Adjustment + Noise	YOLOv8n	63	0.777	0.511

	Injection + Blur				
24	Scaling + Cropping + Brightness Adjustment + Noise Injection + Blur	YOLOv9t	64	0.784	0.503
25	Rotation + Flipping + Brightness Adjustment + Contrast Adjustment + Color Jitter	YOLOv8n	64	0.762	0.491
26	Rotation + Flipping + Brightness Adjustment + Contrast Adjustment + Color Jitter	YOLOv9t	63	0.768	0.480
27	Scaling + Zoom In + Brightness Adjustment + Color Jitter +	YOLOv8n	61	0.792	0.526

	Noise Injection				
28	Scaling + Zoom In + Brightness Adjustment + Color Jitter + Noise Injection	YOLOv9t	64	0.776	0.506
29	Cropping + Zoom In + Contrast Adjustment + Noise Injection + Blur	YOLOv8n	63	0.774	0.498
30	Cropping + Zoom In + Contrast Adjustment + Noise Injection + Blur	YOLOv9t	62	0.747	0.469
31	Scaling + Rotation + Brightness Adjustment + Contrast Adjustment +	YOLOv8n	64	0.784	0.511

	Noise Injection				
32	Scaling + Rotation + Brightness Adjustment + Contrast Adjustment + Noise Injection	YOLOv9t	63	0.777	0.493

Berdasarkan nilai mAP50-95 yang disajikan dalam Tabel 4, model YOLO tanpa augmentasi mencapai nilai mAP50-95 tertinggi pada pengujian YOLOv8n, dengan skor 0.390. Sebaliknya, model YOLO dengan augmentasi tunggal mencapai nilai mAP50-95 tertinggi pada pengujian YOLOv8n menggunakan teknik augmentasi Blur, dengan skor 0.465. Untuk model YOLO dengan augmentasi ganda, nilai mAP50-95 tertinggi diperoleh menggunakan kombinasi teknik Scaling, Zoom In, Brightness Adjustment, Color Jitter, dan Noise Injection, dengan skor mencapai 0.526.

Hasil pengujian ini menunjukkan bahwa penerapan teknik augmentasi, baik tunggal maupun ganda, secara signifikan meningkatkan kinerja model YOLO dibandingkan dengan model tanpa augmentasi. Model dengan augmentasi ganda memberikan hasil terbaik, yang mengindikasikan bahwa kombinasi beberapa

teknik augmentasi dapat secara efektif meningkatkan kemampuan deteksi model.

Untuk deteksi lalu lintas perkotaan selama jam sibuk, di mana objek kendaraan seringkali berukuran kecil dan terpengaruh oleh pantulan sinar matahari, model YOLOv8n menunjukkan kinerja yang lebih unggul dalam menghadapi tantangan tersebut. Pengujian menunjukkan bahwa model YOLOv8n mencapai akurasi yang lebih tinggi dibandingkan dengan model YOLOv9t dalam skenario ini. Oleh karena itu, dapat disimpulkan bahwa model YOLOv8n lebih unggul dalam hal akurasi untuk mendeteksi objek dalam kondisi lalu lintas yang kompleks dan menantang.

4.3. Program YOLOv8n untuk Deteksi Real-Time Kendaraan di Lalu Lintas

Program ini dirancang untuk memanfaatkan model YOLOv8 dalam mendeteksi objek seperti bus, mobil, motor, dan truk pada dataset tertentu. Program juga mencakup fungsi untuk memvisualisasikan hasil deteksi dengan anotasi pada gambar. Program ini melibatkan penggunaan pustaka Python seperti OpenCV, NumPy, dan Matplotlib.

1. Import Pustaka dan Inisialisasi

Program diawali dengan mengimpor pustaka-pustaka yang diperlukan:

os: Untuk operasi sistem file.

glob: Untuk pencarian file dengan pola tertentu.

matplotlib.pyplot: Untuk membuat visualisasi gambar.

cv2 (OpenCV): Untuk manipulasi gambar.

random: Untuk memilih data acak.

numpy: Untuk komputasi numerik.

requests: Untuk mengirim permintaan HTTP (meskipun tidak digunakan di kode ini).

Inisialisasi Variabel

- class_names: Daftar kelas objek yang akan dideteksi ('Bus', 'Car', 'Motorcycle', 'Truck').
- colors: Array warna untuk anotasi tiap kelas.

2. Fungsi-Fungsi Utama

Fungsi yolo2standard(bboxes)

Fungsi ini mengubah format bounding box dari format YOLO (dengan pusat dan ukuran) menjadi format standar (koordinat sudut kiri atas dan kanan bawah).

- Input: Bounding box dalam format [x_center, y_center, width, height].
- Output: Koordinat [xmin, ymin, xmax, ymax].

Fungsi plot_box(image, bboxes, labels)

Fungsi ini menambahkan bounding box ke gambar berdasarkan hasil deteksi YOLO.

Input:

- image: Gambar input.

- bboxes: Daftar bounding box dalam format YOLO.
- labels: Daftar label untuk tiap bounding box.

Proses:

- Mengonversi bounding box ke format standar.
- Menggambar kotak pada gambar menggunakan warna sesuai kelas.
- Menambahkan teks label ke bounding box.

Output: Gambar dengan bounding box dan label.

Fungsi plot(image_paths, label_paths, num_samples)

Fungsi ini digunakan untuk menampilkan beberapa sampel gambar dengan bounding box yang divisualisasikan.

Input:

- image_paths: Path ke gambar.
- label_paths: Path ke file label YOLO.
- num_samples: Jumlah sampel gambar yang ingin divisualisasikan.

Proses:

- Memuat dan mengurutkan file gambar dan label.
- Memilih sampel secara acak.
- Membaca dan memproses file label.
- Memanggil fungsi plot_box untuk menambahkan anotasi pada gambar.

Output: Visualisasi gambar dengan anotasi bounding box.

3. Integrasi Google Drive

Untuk menyimpan dan mengakses dataset, Google Drive dihubungkan menggunakan:

```
from google.colab import drive
drive.mount('/content/drive')
```

Hal ini memungkinkan pengguna untuk memuat dataset dari direktori Google Drive.

4. Instalasi dan Konfigurasi YOLOv8

Instalasi YOLOv8

YOLOv8 diinstal melalui perintah:

```
pip install ultralytics
```

Memuat Model YOLO

Model YOLOv8 diinisialisasi menggunakan:

```
from ultralytics import YOLO
model = YOLO("yolov8n.pt")
```

5. Pelatihan Model

Fungsi model.train() digunakan untuk melatih model YOLOv8 dengan parameter yang telah disesuaikan:

Parameter:

- data: Path ke file konfigurasi data.yaml yang mendeskripsikan dataset.
- epochs: Jumlah epoch pelatihan (1 epoch pada kode ini).
- batch: Ukuran batch (8 pada kode ini).
- patience: Jumlah epoch tanpa peningkatan sebelum berhenti lebih awal (early stopping).
- dropout: Dropout rate untuk mengurangi overfitting.

python

```
results = model.train(
    data="/content/drive/My Drive/datasets_imam/datasets_blur/data.yaml",
    epochs=1,
    batch=8,
    patience=10,
    dropout=0.2
)
```

BAB V Integrasi Computer Vision dan Deep Learning untuk Prediksi Lalu Lintas

Computer Vision (CV) adalah cabang ilmu komputer yang memungkinkan komputer untuk memahami dan menafsirkan dunia visual melalui gambar dan video. Sementara itu, Deep Learning (DL) adalah subset dari machine learning yang menggunakan jaringan saraf tiruan dengan banyak lapisan untuk mempelajari representasi data yang kompleks. Integrasi kedua teknologi ini memiliki relevansi yang sangat besar dalam pengelolaan lalu lintas, seperti meningkatkan efisiensi pengelolaan lalu lintas, mengurangi kemacetan dan risiko kecelakaan, serta mendukung implementasi kota pintar.

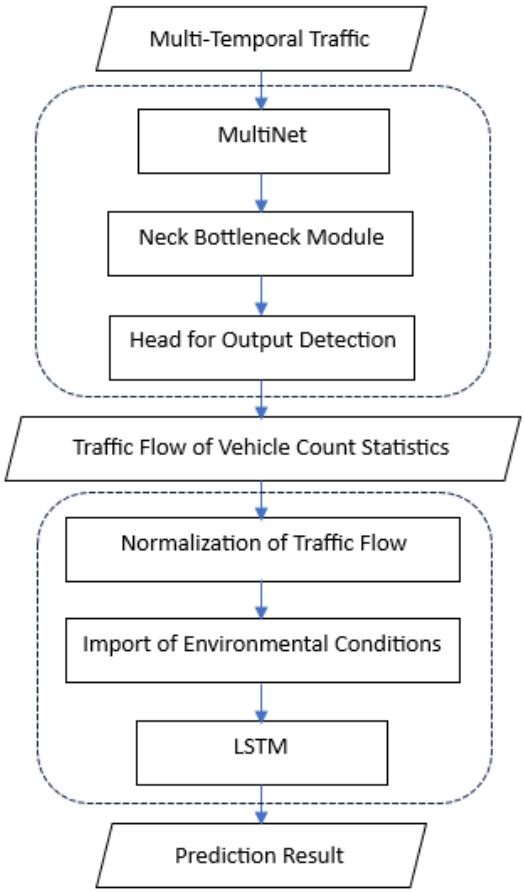
Computer Vision memiliki beberapa teknologi utama, seperti deteksi objek untuk mengidentifikasi kendaraan, pejalan kaki, dan elemen jalan lainnya; segmentasi gambar untuk memisahkan berbagai elemen dalam gambar, seperti jalan, kendaraan, dan marka jalan; serta pelacakan objek untuk memantau pergerakan kendaraan dari frame ke frame dalam video. Framework populer yang sering digunakan dalam Computer Vision adalah OpenCV, MediaPipe, dan Detectron2. Sementara itu, Deep Learning menawarkan model seperti Convolutional Neural Networks (CNN) untuk ekstraksi fitur dari gambar, Recurrent Neural Networks (RNN) dan Long Short-Term Memory (LSTM) untuk

analisis data temporal, serta YOLO (You Only Look Once) untuk deteksi objek secara real-time. Framework yang sering digunakan untuk Deep Learning meliputi TensorFlow, PyTorch, dan Keras.

Integrasi antara Computer Vision dan Deep Learning biasanya dilakukan melalui pipeline sistem yang dimulai dari pengumpulan data menggunakan kamera CCTV atau drone untuk merekam video lalu lintas. Data yang diperoleh kemudian diproses melalui langkah-langkah seperti resizing, normalisasi, dan augmentasi data. Setelah itu, deteksi dan klasifikasi kendaraan dilakukan menggunakan model seperti YOLO atau Faster R-CNN untuk mendeteksi kendaraan dan mengklasifikasikannya ke dalam kategori seperti motor, mobil, truk, dan bus. Langkah selanjutnya adalah pelacakan kendaraan menggunakan algoritma seperti SORT (Simple Online and Realtime Tracking) atau DeepSORT. Data hasil pelacakan ini kemudian digunakan untuk memprediksi pola lalu lintas menggunakan model seperti LSTM.

Sebagai contoh, pada penelitian yang berjudul '*A Deep Learning–Based Approach for Moving Vehicle Counting and Short-Term Traffic Prediction From Video Images*'. Penelitian ini bertujuan untuk mengusulkan sistem manajemen arus lalu lintas dua tingkat untuk mengatasi tantangan yang disebutkan sebelumnya. Sistem ini didukung oleh teknik pembelajaran mendalam (deep learning) dan divalidasi menggunakan rangkaian video selama dua bulan di sebuah persimpangan jalan populer di pusat kota

Shenzhen. Secara khusus, sistem ini terdiri dari dua lapisan, yaitu YOLOv4 untuk deteksi dan pelacakan objek kendaraan, serta jaringan Long Short-Term Memory (LSTM) yang telah dimodifikasi dengan karakteristik spasial-temporal dari catatan lalu lintas historis dan informasi cuaca terkait untuk membangun model prediksi arus lalu lintas jangka pendek. Dalam hal deteksi objek bergerak, penelitian ini mengusulkan jaringan DCN-MultiNet-YOLO yang ringan untuk pelacakan multi-target berbasis video guna mengumpulkan statistik volume lalu lintas pada tingkat persimpangan jalan perkotaan. Sedangkan dalam hal prediksi arus lalu lintas, penelitian ini mengembangkan jaringan LSTM yang ditingkatkan untuk mendekati skenario realistis dengan mempertimbangkan berbagai kondisi cuaca yang terkait dengan perubahan arus lalu lintas sebenarnya. Model Prediksi jumlah kendaraan menggunakan Integrasi YOLO dan LSTM yang diusulkan pada penelitian ini dapat dilihat pada gambar 13.



Gambar 13. *Prediksi jumlah kendaraan menggunakan Integrasi YOLO dan LSTM*

Gambar 13 menunjukkan gambaran umum pendekatan model yang terdiri dari dua bagian. Bagian pertama adalah deteksi kendaraan dan ekstraksi arus lalu lintas dari data video lalu lintas multi-temporal, di mana jaringan saraf inti mencakup Multi-Net yang terdiri dari backbone, modul leher untuk meningkatkan ekstraksi fitur, dan modul kepala untuk mendeteksi keluaran.

Pada bagian kedua, setelah normalisasi data lalu lintas, selanjutnya mengekstrak vektor fitur dari faktor lingkungan dan memasukkannya ke dalam jaringan LSTM yang telah ditingkatkan untuk memprediksi data arus lalu lintas di bawah berbagai kondisi cuaca.

Penelitian ini melakukan serangkaian eksperimen menggunakan data video lalu lintas nyata dengan rentang waktu dua bulan di salah satu persimpangan jalan populer di pusat kota Shenzhen, Tiongkok. Hasil penelitian menunjukkan bahwa algoritma yang diusulkan memiliki kinerja lebih baik dibandingkan metode sebelumnya, dengan peningkatan akurasi pelacakan deteksi target sebesar 10% dan pengurangan kesalahan prediksi arus lalu lintas hingga setengahnya ketika mempertimbangkan kondisi cuaca.

Meskipun algoritma yang diusulkan dalam penelitian ini menunjukkan keunggulan, masih ada ruang untuk perbaikan. Dalam hal deteksi objek, diperlukan untuk menyematkan konvolusi terpisah berbasis kedalaman (depth-wise separable convolution) untuk mengurangi jumlah CSPDarknet53 pada jaringan YOLOv4 agar sesuai untuk operasi waktu nyata pada perangkat seluler. Dalam hal prediksi arus lalu lintas jangka pendek, kondisi cuaca saat ini hanya dapat dijelaskan sebagai variabel kualitatif, seperti hari cerah dan hujan, yang membatasi akurasi prediksi hingga batas tertentu. Penelitian di masa depan dapat mencakup faktor kuantitatif seperti curah hujan dan tekanan udara. Selain faktor lingkungan fisik, faktor manusia, seperti perilaku pengemudi dalam situasi darurat, juga dapat

dipertimbangkan untuk membuat model lebih mendekati kenyataan.

BAB VI Prediksi Arus Lalu Lintas dengan Model LSTM

6.1. Model Long Short-Term Memory (LSTM) untuk Prediksi Arus Lalu Lintas

Model Long Short-Term Memory (LSTM) merupakan salah satu arsitektur deep learning yang sangat efektif untuk menangani prediksi arus lalu lintas. LSTM dirancang untuk memahami dan memproses data berurutan, sehingga ideal untuk analisis lalu lintas yang bergantung pada pola temporal. Dalam konteks ini, LSTM mampu mengidentifikasi hubungan antara data historis dan data real-time untuk memprediksi kondisi lalu lintas di masa mendatang, seperti tingkat kepadatan, kecepatan kendaraan, atau waktu perjalanan.

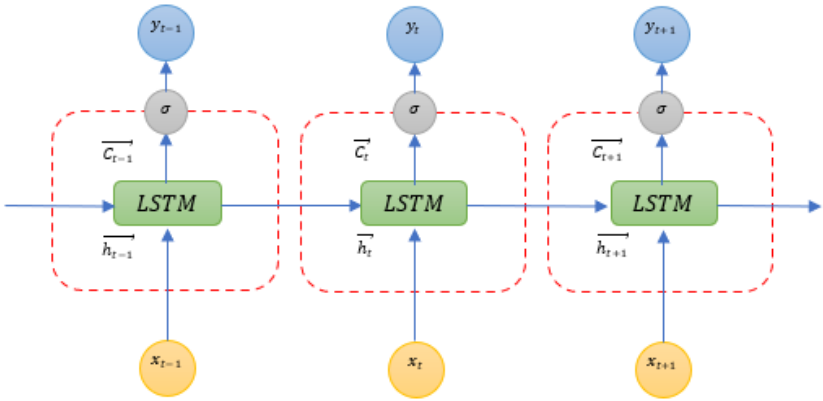
Salah satu keunggulan utama LSTM adalah kemampuannya untuk mengatasi masalah vanishing gradient yang sering ditemukan pada model jaringan saraf tradisional. Dengan mekanisme memori selnya, LSTM dapat mempertahankan informasi penting dari data sekuensial untuk periode waktu yang lebih panjang. Hal ini memungkinkan model untuk memahami pola yang kompleks dalam arus lalu lintas, termasuk efek dari variabel eksternal seperti cuaca, waktu, dan hari dalam seminggu.

Implementasi LSTM untuk prediksi arus lalu lintas biasanya melibatkan penggunaan data lalu lintas yang

dikumpulkan dari sensor jalan, kamera CCTV, atau perangkat IoT lainnya. Data ini mencakup parameter seperti jumlah kendaraan, kecepatan rata-rata, dan jarak antar kendaraan. Setelah data diolah dan disiapkan, model LSTM dilatih untuk mempelajari pola pergerakan lalu lintas. Hasil prediksi dari model ini dapat digunakan untuk merencanakan strategi pengelolaan lalu lintas yang lebih efektif, seperti pengaturan lampu lalu lintas adaptif atau rute alternatif untuk mengurangi kemacetan.

Selain manfaatnya dalam manajemen lalu lintas, LSTM juga memberikan kontribusi signifikan pada pengembangan sistem transportasi pintar. Dengan integrasi model ini ke dalam aplikasi real-time, seperti navigasi berbasis peta digital atau kendaraan otonom, pengguna jalan dapat menerima informasi akurat mengenai kondisi lalu lintas terkini dan prediksi masa depan. Dengan demikian, model LSTM tidak hanya membantu mengurangi kemacetan, tetapi juga meningkatkan pengalaman berkendara dan efisiensi transportasi secara keseluruhan.

Arsitektur atau proses dalam jaringan LSTM yang menggunakan mekanisme Attention untuk pemrosesan data arus lalu lintas dapat dilihat pada gambar 14.



Gambar 14. *Arsitektur LSTM untuk pemrosesan data arus lalu lintas*

Gambar tersebut menunjukkan bagaimana mekanisme perhatian (Attention) digunakan dalam jaringan LSTM untuk memproses data historis arus lalu lintas. Proses dimulai dari lapisan input, di mana data historis dari berbagai sumber atau waktu yang berbeda dimasukkan. Setiap data historis ini direpresentasikan sebagai D_1, D_2, ... D_n, dengan masing-masing data mencakup urutan waktu tertentu dalam jendela waktu $(2m + 1)$.

Setelah data masuk ke lapisan input, mereka diproses oleh lapisan perhatian (Attention Layer). Lapisan ini bertugas menghitung bobot perhatian (w^i) untuk setiap data input (x^i). Bobot perhatian ini menentukan seberapa penting setiap elemen data dalam memengaruhi hasil akhir. Dengan kata lain, mekanisme ini memungkinkan model untuk fokus pada bagian data yang lebih relevan, sehingga informasi yang kurang penting dapat diabaikan.

Bobot perhatian kemudian dinormalisasi menggunakan fungsi softmax. Fungsi softmax ini mengubah bobot mentah menjadi probabilitas $(\widetilde{w}^i)$ yang jumlah totalnya sama dengan satu. Hal ini memastikan bahwa perhatian yang diberikan pada setiap elemen input proporsional terhadap relevansinya.

Setelah normalisasi, bobot perhatian $(\widetilde{w}^i)$ digunakan untuk mengalikan setiap nilai input (x^i). Hasil perkalian ini $(\widetilde{w}^i x^i)$ menghasilkan nilai tertimbang untuk masing-masing elemen input. Proses ini memberikan kontribusi dari setiap elemen data sesuai dengan tingkat perhatian yang diberikan.

Akhirnya, semua nilai tertimbang ini dijumlahkan untuk menghasilkan input baru pada waktu t, yang direpresentasikan sebagai x_t. Input baru ini adalah representasi gabungan dari data historis yang telah disesuaikan dengan bobot perhatian, sehingga hanya informasi yang relevan yang berperan dalam prediksi.

Mekanisme perhatian ini sangat bermanfaat dalam membantu jaringan LSTM menangkap pola-pola penting dalam data historis, terutama pada dataset kompleks seperti prediksi arus lalu lintas. Dengan berfokus pada elemen data yang relevan, model dapat menghasilkan prediksi yang lebih akurat dan efisien.

6.2. Program LSTM untuk Prediksi Arus Lalu Lintas Menggunakan Python

Berikut adalah langkah-langkah detail untuk membuat program prediksi arus lalu lintas menggunakan LSTM:

1. **Persiapan Lingkungan**
   Install library yang diperlukan:
   Pastikan library berikut sudah terpasang:

```
pip install pandas numpy scikit-learn tensorflow
```

   Dataset:
   Dataset harus dalam format CSV dengan kolom berikut: Date, Time, CarCount, MotorcycleCount, BusCount, TruckCount, dan Total.

2. **Penjelasan dan Implementasi Program**
   Langkah 1: Membaca Dataset
   Baca file CSV menggunakan pandas:

```
df = pd.read_csv('/content/drive/My Drive/datasets_imam/traffic_hasil.csv')
```

   Langkah 2: Gabungkan Kolom Date dan Time
   Gabungkan kedua kolom untuk membuat kolom Datetime:

```
df['Datetime'] = pd.to_datetime(df['Date'] + ' ' + df['Time'], format='%d/%m/%Y %H:%M')
```

   Langkah 3: Atur Datetime sebagai Indeks

Atur kolom Datetime sebagai indeks untuk analisis berbasis waktu:

```
df.set_index('Datetime',
inplace=True)
```

Langkah 4: Pilih Kolom Relevan
Pilih kolom yang relevan untuk prediksi:

```
data = df[['CarCount',
'MotorcycleCount', 'BusCount',
'TruckCount', 'Total']]
```

Langkah 5: Scaling Data
Gunakan MinMaxScaler untuk normalisasi data:

```
from sklearn.preprocessing import
MinMaxScaler
scaler = MinMaxScaler()
data_scaled =
scaler.fit_transform(data)
```

Langkah 6: Membuat Input dan Output
Buat input dan output dengan interval 5 menit:

```
sequence_length = 6  # Menggunakan
data 30 menit terakhir (6 interval
5 menit)
X, y = [], []
```

```
for i in range(sequence_length,
len(data_scaled)):
    X.append(data_scaled[i-
sequence_length:i])
    y.append(data_scaled[i])

X = np.array(X)
y = np.array(y)
```

Langkah 7: Membagi Data
Pisahkan data menjadi training dan testing:

```
from sklearn.model_selection
import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(X, y,
test_size=0.2, shuffle=False)
```

Langkah 8: Membangun Model LSTM
Bangun model LSTM dengan TensorFlow:

```
from tensorflow.keras.models
import Sequential
from tensorflow.keras.layers
import LSTM, Dense

model = Sequential([
    LSTM(50,
return_sequences=False,
input_shape=(X_train.shape[1],
X_train.shape[2])),
```

```
    Dense(5)   # Output: CarCount,
MotorcycleCount,          BusCount,
TruckCount, Total
])

model.compile(optimizer='adam',
loss='mean_squared_error')
```

Langkah 9: Melatih Model
Latih model dengan data training:

```
history    =    model.fit(X_train,
y_train,                epochs=300,
batch_size=32,
validation_data=(X_test, y_test))
```

Langkah 10: Prediksi Data Uji
Lakukan prediksi:

```
y_pred = model.predict(X_test)
```

Langkah 11: Mengembalikan ke Skala Asli
Gunakan scaler untuk mengembalikan prediksi ke skala asli:

```
y_pred_rescaled                    =
scaler.inverse_transform(y_pred)
y_test_rescaled                    =
scaler.inverse_transform(y_test)
```

Langkah 12: Membulatkan Hasil Prediksi

Bulatkan hasil prediksi:

```
y_pred_rounded = np.round(y_pred_rescaled)
```

Langkah 13: Evaluasi Model
Hitung Mean Absolute Error (MAE):

```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test_rescaled, y_pred_rescaled)
print(f"Mean Absolute Error (MAE): {mae}")
```

Langkah 14: Simpan Hasil Prediksi
Gabungkan hasil prediksi dengan data asli dan simpan ke CSV:

```
predictions_df = pd.DataFrame({
    'Datetime': df.index[-len(y_test):],
    'CarCount_Pred': y_pred_rounded[:, 0],
    'MotorcycleCount_Pred': y_pred_rounded[:, 1],
    'BusCount_Pred': y_pred_rounded[:, 2],
    'TruckCount_Pred': y_pred_rounded[:, 3],
```

```
    'Total_Pred':
y_pred_rounded[:, 4],
    'CarCount_True':
np.round(y_test_rescaled[:, 0]),
    'MotorcycleCount_True':
np.round(y_test_rescaled[:, 1]),
    'BusCount_True':
np.round(y_test_rescaled[:, 2]),
    'TruckCount_True':
np.round(y_test_rescaled[:, 3]),
    'Total_True':
np.round(y_test_rescaled[:, 4])
})

predictions_df.to_csv('/content/dr
ive/My
Drive/hasil_yolo_sw_lstm/LSTM.csv'
, index=False)
```

Langkah 15: Total Runtime
Hitung waktu eksekusi program:

```
import time
start_time = time.time()

# Semua proses di atas

end_time = time.time()
total_runtime   =   (end_time   -
start_time) / 60
```

```
print(f"Total                Runtime:
{total_runtime:.2f} menit")
```

Hasil

- Evaluasi: Nilai MAE akan ditampilkan untuk mengevaluasi performa model.
- File CSV: File hasil prediksi disimpan dalam file LSTM.csv.

BAB VII Optimasi Model LSTM menggunakan Algoritma Metaheuristik

7.1. Optimasi Model LSTM menggunakan Algoritma Metaheuristik

Optimasi model Long Short-Term Memory (LSTM) menjadi salah satu langkah penting dalam meningkatkan akurasi prediksi arus lalu lintas. Model LSTM memiliki sejumlah parameter hiper seperti jumlah neuron, jumlah lapisan tersembunyi, learning rate, batch size, dan jumlah epoch, yang perlu diatur dengan tepat untuk mencapai performa terbaik. Namun, pencarian kombinasi parameter yang optimal sering kali menjadi tantangan karena kompleksitas ruang pencarian dan interaksi antarparameter. Di sinilah algoritma metaheuristik memainkan peran penting dalam mengatasi tantangan ini.

Algoritma metaheuristik, seperti Particle Swarm Optimization (PSO), Grey Wolf Optimization (GWO), dan Komodo Mlipir Optimization (KMA), dirancang untuk mencari solusi optimal dalam ruang pencarian yang luas dan tidak terstruktur. Dengan menggunakan pendekatan berbasis populasi, algoritma ini mengevaluasi banyak kandidat solusi secara paralel dan memperbarui solusi berdasarkan mekanisme iteratif. Dalam konteks optimasi LSTM, algoritma ini dapat digunakan untuk menemukan kombinasi parameter

yang menghasilkan kesalahan prediksi terkecil, diukur dengan metrik seperti Root Mean Square Error (RMSE) atau Mean Absolute Error (MAE).

Implementasi optimasi menggunakan algoritma metaheuristik biasanya melibatkan proses iteratif yang mencakup pelatihan LSTM dengan parameter yang dihasilkan oleh algoritma, evaluasi performa model, dan pembaruan parameter berdasarkan aturan optimasi. Misalnya, algoritma PSO akan memanfaatkan posisi dan kecepatan partikel untuk mencari nilai parameter optimal, sementara GWO meniru perilaku berburu serigala abu-abu untuk mencapai solusi terbaik. Proses ini memastikan bahwa model LSTM tidak hanya lebih akurat tetapi juga lebih efisien dalam memproses data arus lalu lintas yang kompleks.

Dengan mengintegrasikan algoritma metaheuristik, model LSTM mampu memberikan prediksi yang lebih presisi dan adaptif terhadap berbagai kondisi lalu lintas. Hal ini sangat penting dalam aplikasi transportasi cerdas, seperti pengelolaan lalu lintas real-time, sistem navigasi, dan kendaraan otonom. Optimasi berbasis metaheuristik juga membuka peluang untuk mengatasi tantangan di masa depan, termasuk peningkatan skala model dan integrasi variabel eksternal seperti cuaca atau peristiwa khusus. Dengan pendekatan ini, kombinasi antara LSTM dan algoritma metaheuristik menawarkan solusi yang canggih dan efektif untuk prediksi lalu lintas yang lebih baik. Beberapa review penelitian yang membahas tentang Optimasi Model

LSTM menggunakan Algoritma Metaheuristik dapat dilihat padsa tabel 5.

Tabel 5. *Hasil review optimasi LSTM dengan Algoritma Metaheuristik untuk Prediksi Arus Lalu Lintas*

No	Judul	Optimasi Parameter LSTM dengan Metheuristik
1	Short-Term Traffic Flow Prediction Based on VMD and IDBO-LSTM	IDBO (Improved Dragonfly Algorithm): IDBO memanfaatkan prinsip dari Dragonfly Algorithm yang terinspirasi dari perilaku kawanan capung. Metode ini mencari parameter optimal LSTM dengan menyeimbangkan eksplorasi (mencari solusi baru) dan eksploitasi (menggunakan solusi terbaik). IDBO meningkatkan efisiensi algoritma standar dengan mekanisme adaptif untuk memperbarui posisi dan mempercepat konvergensi menuju solusi terbaik.
2	MFOA-Bi-LSTM: An optimized	MFOA (Modified Fruit Fly Optimization Algorithm):

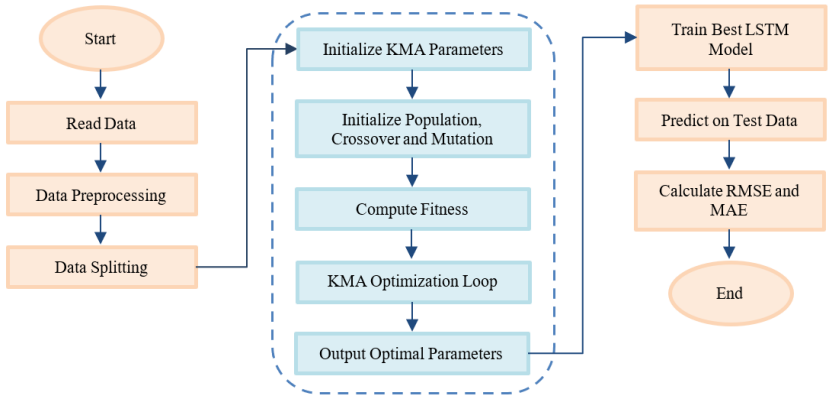
	bidirectional long short-term memory model for short-term traffic flow prediction	MFOA adalah versi yang dimodifikasi dari FOA, terinspirasi oleh kemampuan lalat buah untuk menemukan makanan menggunakan penciuman. MFOA mengoptimalkan parameter LSTM dengan iterasi pencarian global di awal untuk eksplorasi yang lebih luas, kemudian memperbaiki pencarian di sekitar solusi terbaik pada tahap akhir. Ini membantu menghindari perangkap local optima.
3	Short-term traffic flow prediction based on optimized deep learning neural network: PSO-Bi-LSTM	PSO (Particle Swarm Optimization): PSO bekerja dengan mensimulasikan perilaku kawanan burung atau ikan. Setiap partikel dalam kawanan merepresentasikan sekumpulan parameter LSTM. Partikel bergerak di ruang pencarian berdasarkan pengalaman terbaiknya (personal best) dan pengalaman kawanan

		(global best), sehingga menghasilkan parameter LSTM yang optimal secara iteratif.
4	Traffic Flow Prediction at Intersections: Enhancing with a Hybrid LSTM-PSO Approach	PSO (Particle Swarm Optimization): Dalam pendekatan hybrid ini, PSO digunakan untuk mengoptimalkan parameter utama LSTM, seperti jumlah neuron, learning rate, dan ukuran batch. Hybridisasi dilakukan dengan mengintegrasikan PSO dengan mekanisme LSTM sehingga menghasilkan model yang lebih akurat dan efisien.

7.2. Optimasi Model LSTM menggunakan Algoritma KMA untuk Prediksi Kemacetan Lalu Lintas

Integrasi antara Long Short-Term Memory (LSTM) dan Komodo Mlipir Algorithm (KMA) memanfaatkan keunggulan masing-masing metode dalam analisis data dan optimasi. LSTM sangat efektif dalam menangkap pola temporal pada data deret waktu, menjadikannya unggul untuk tugas prediksi baik jangka pendek maupun panjang. Studi terdahulu menunjukkan bahwa optimasi hiperparameter menggunakan pendekatan

metaheuristik sering kali memberikan hasil lebih baik dibandingkan metode konvensional, memperkuat keunggulan kombinasi teknik ini dalam meningkatkan kinerja model prediksi. Penggunaan KMA dalam pendekatan ini diharapkan mampu melampaui performa algoritma metaheuristik lainnya. Kombinasi LSTM dan KMA tidak hanya mempercepat proses optimasi, tetapi juga meningkatkan peluang menemukan konfigurasi hiperparameter yang optimal, yang secara signifikan dapat meningkatkan performa model LSTM dalam aplikasi prediksi aliran lalu lintas. Pendekatan yang diusulkan dapat dilihat pada Gambar 15.



Gambar 15. *Arsitektur LSTM-KMA dalam prediksi arus lalu lintas*

Gambar 14 menunjukkan proses komputasi LSTM-KMA dimulai dengan membaca dan melakukan praproses data, dilanjutkan dengan membagi dataset menjadi data latih dan data uji, Training 80%, Testing 20%. Algoritma Komodo Mlipir (KMA) kemudian diinisialisasi dengan parameter tertentu. Proses ini

mencakup inisialisasi populasi solusi kandidat, serta penerapan operasi crossover dan mutasi. Fitness dari setiap kandidat solusi dihitung untuk mengevaluasi kecocokan parameter terhadap model LSTM menggunakan MAE. Parameter LSTM yang dioptimalkan adalah neurons, learning rate, dan epoch. KMA secara iteratif memperbarui solusi kandidat melalui loop optimisasi hingga parameter optimal ditemukan. Parameter yang dioptimalkan ini digunakan untuk melatih model LSTM terbaik. Model yang telah dilatih kemudian dievaluasi menggunakan data uji dengan menghitung RMSE dan MAE, yang memberikan ukuran akurasi prediksi. Konsep KMA dalam optimasi parameter LSTM dijelaskan menggunakan pseudocode pada algoritma 1.

Algorithm 1: Komodo MLIPIR for Optimizing LSTM Parameters

Input:

Jumlah iterasi maksimum (T), ukuran populasi (n).
Rentang parameter LSTM yang akan dioptimasi (jumlah neuron, learning rate, dan jumlah epoch).

Step 1: Initialization

Inisialisasi populasi sebanyak n individu (komodo) dengan kombinasi acak dari parameter LSTM.
Setiap individu q dalam populasi direpresentasikan sebagai: $Pq = [Xq, Yq, Zq]$, di mana $Xq, Yq,$ dan Zq masing-masing menunjukkan jumlah neuron, learning rate, dan epoch.

Step 2: Fitness Evaluation

Evaluasi nilai fitness awal dari setiap kandidat solusi dengan mengukur kinerja LSTM pada data validasi.
Gunakan fungsi objektif: Minimalkan F = MAE (Mean Absolute Error).
Urutkan individu berdasarkan skor fitness-nya, lalu kategorikan ke dalam tiga kelompok:

- Jantan besar (elite, performa terbaik)
- Betina (performa sedang)
- Jantan kecil (performa rendah)

Step 3: Main Loop

While $(t \leq T)$:

1. Evaluasi ulang skor fitness setiap individu.
2. Perbarui posisi mereka sebagai berikut:
   - Jantan besar: Sesuaikan posisi menggunakan strategi eksploitasi.
   - Betina:
     - Kawin dengan jantan besar terbaik menggunakan metode eksploitasi.
     - Bereproduksi secara aseksual (parthenogenesis) menggunakan strategi eksplorasi.
   - Jantan kecil: Menjelajahi ruang solusi secara acak menggunakan strategi eksplorasi.
3. Terapkan proses seleksi:
   - Pertahankan individu dengan performa terbaik (elitisme).
   - Perbaiki individu yang lemah menggunakan strategi pembaruan pada persamaan.

4. Tingkatkan jumlah iterasi ($t = t + 1$).

End While

Step 4: Output the Best Solution

Keluarkan parameter LSTM terbaik (P_best) dan nilai fitness terbaik (F_best).

Output:

Parameter LSTM yang optimal.

Program LSTM-KMA untuk Prediksi Arus Lalu Lintas

Program dimulai dengan mengimpor pustaka yang diperlukan, seperti pandas untuk manipulasi data, numpy untuk operasi numerik, dan pustaka dari Keras (Sequential, LSTM, Dense) untuk membangun dan melatih model LSTM. Selain itu, pustaka seperti time digunakan untuk mencatat waktu eksekusi program, dan random untuk proses inisialisasi parameter secara acak.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

```
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
import random
import time
```

Tahap berikutnya adalah memulai pencatatan waktu eksekusi program. Program kemudian memuat data dari file CSV menggunakan fungsi pd.read_csv, di mana data tersebut berisi informasi jumlah kendaraan berdasarkan waktu.

```
# Start timing
start_time = time.time()


# Load data from CSV
file_path = '/content/drive/MyDrive/datasets_imam/traffic_hasil.csv'
df = pd.read_csv(file_path)
```

Setelah memuat data, kolom Date dan Time digabung menjadi satu kolom Datetime menggunakan fungsi pd.to_datetime. Data ini kemudian diset sebagai indeks

agar dapat diakses lebih mudah dalam analisis berbasis waktu.

```
# Combine Date and Time columns into a datetime column
df['Datetime']                          = pd.to_datetime(df['Date']  +  '  '  + df['Time'], format='%d/%m/%Y %H:%M')
df.set_index('Datetime', inplace=True)
# Combine Date and Time columns into a datetime column
df['Datetime']                          = pd.to_datetime(df['Date']  +  '  '  + df['Time'], format='%d/%m/%Y %H:%M')
df.set_index('Datetime', inplace=True)
```

Kolom yang relevan untuk model, yaitu CarCount, MotorcycleCount, BusCount, TruckCount, dan Total, dipilih dari data untuk digunakan sebagai input.

```
# Select relevant columns for the model
data        =        df[['CarCount', 'MotorcycleCount',        'BusCount', 'TruckCount', 'Total']]
```

Data yang telah dipilih dinormalisasi menggunakan MinMaxScaler untuk mengubah nilai menjadi dalam

rentang [0, 1]. Hal ini memastikan model tidak terpengaruh oleh perbedaan skala antara variabel.

```
# Normalize data using MinMaxScaler

scaler = MinMaxScaler()

data_scaled                              =
scaler.fit_transform(data)
```

Input (X) dan output (y) dipersiapkan dengan menggeser data satu langkah. Data input adalah semua baris kecuali yang terakhir, sedangkan data output adalah semua baris kecuali yang pertama. Data input kemudian diubah menjadi bentuk 3D, yang merupakan format masukan untuk model LSTM.

```
# Prepare input (X) and output (y)
without sliding window

X = data_scaled[:-1]

y = data_scaled[1:]

# Reshape input to 3D for LSTM

X    =    X.reshape((X.shape[0],    1,
X.shape[1]))
```

Data dibagi menjadi data latih dan uji menggunakan fungsi train_test_split, dengan pembagian 80% untuk pelatihan dan 20% untuk pengujian. Urutan data dipertahankan dengan mengatur shuffle=False.

```
# Split data into training and testing
sets
```

```
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
shuffle=False)
```

Parameter yang akan dioptimalkan oleh KMA didefinisikan, mencakup learning_rate, jumlah neurons, epochs, dan jumlah hidden_layers.

```
# Define KMA parameters
population_size = 30
max_iterations = 1
param_bounds = {
    'learning_rate': (0.001, 0.01),
    'neurons': (300, 500),
    'epochs': (1, 150),
    'hidden_layers': (1, 5)
}
```

Fungsi fitness didefinisikan untuk mengevaluasi setiap individu dalam populasi. Fungsi ini menggunakan parameter untuk membangun model LSTM, melatih model pada data latih, dan menghitung nilai Mean Absolute Error (MAE) serta Root Mean Squared Error (RMSE).

```
# Fitness function
def fitness(individual):
    model = build_model(individual)
    model.fit(X_train,       y_train,
epochs=individual['epochs'],
verbose=0)
    y_pred = model.predict(X_test)
```

```
    # Rescale predictions and ground
truth
    y_pred_rescaled                 =
scaler.inverse_transform(y_pred)
    y_test_rescaled                 =
scaler.inverse_transform(y_test)

    # Calculate MAE and RMSE
    mae                             =
mean_absolute_error(y_test_rescaled,
y_pred_rescaled)
    rmse                            =
np.sqrt(np.mean((y_test_rescaled    -
y_pred_rescaled) ** 2))
    return mae, rmse
```

Fungsi untuk membangun model LSTM dengan parameter yang fleksibel juga didefinisikan. Model dapat memiliki jumlah lapisan tersembunyi yang bervariasi tergantung pada parameter yang dioptimalkan.

```
# Build LSTM model with flexible
hidden layers
def build_model(params):
    model = Sequential()
    for                 _               in
range(params['hidden_layers']):

model.add(LSTM(params['neurons'],
return_sequences=True,
input_shape=(X_train.shape[1],
X_train.shape[2])))
```

```
model.add(LSTM(params['neurons']))

model.add(Dense(y_train.shape[1]))
    optimizer                          =
Adam(learning_rate=params['learning_
rate'])

model.compile(optimizer=optimizer,
loss='mean_squared_error')
    return model
```

Populasi awal dihasilkan secara acak berdasarkan batas parameter yang telah didefinisikan sebelumnya.

```
# Initialize population
def initialize_population():
    population = []
    for _ in range(population_size):
        individual = {
            'learning_rate':
random.uniform(*param_bounds['learni
ng_rate']),
            'neurons':
random.randint(*param_bounds['neuron
s']),
            'epochs':
random.randint(*param_bounds['epochs
']),
            'hidden_layers':
random.randint(*param_bounds['hidden
_layers'])
        }
```

```
population.append(individual)
    return population
```

Proses optimasi dilakukan menggunakan KMA. Dalam setiap iterasi, nilai fitness dihitung untuk setiap individu, dan individu terbaik diperbarui jika performanya lebih baik dari sebelumnya.

```
# KMA optimization
population = initialize_population()
best_individual = None
best_fitness_mae = float('inf')
best_fitness_rmse = float('inf')
for _ in range(max_iterations):
    fitness_scores = [fitness(ind)
for ind in population]
    maes, rmses =
zip(*fitness_scores)
    best_idx = np.argmin(maes)
    if maes[best_idx] <
best_fitness_mae:
        best_fitness_mae =
maes[best_idx]
        best_fitness_rmse =
rmses[best_idx]
        best_individual =
population[best_idx]
```

Setelah parameter terbaik ditemukan, model dilatih ulang menggunakan parameter tersebut. Hasil prediksi

dan evaluasi model disimpan dalam file CSV dan ditampilkan.

```
# Final training with best parameters
final_model                              =
build_model(best_individual)
final_model.fit(X_train,     y_train,
epochs=best_individual['epochs'],
verbose=1)
# Predictions
y_pred = final_model.predict(X_test)
y_pred_rescaled                          =
scaler.inverse_transform(y_pred)
y_test_rescaled                          =
scaler.inverse_transform(y_test)
# Evaluations
mae                                      =
mean_absolute_error(y_test_rescaled,
y_pred_rescaled)
rmse                                     =
np.sqrt(np.mean((y_test_rescaled    -
y_pred_rescaled) ** 2))
# Save predictions
predictions_df = pd.DataFrame({
    'Datetime':            df.index[-
len(y_test):],
    'CarCount_Pred':
np.round(y_pred_rescaled[:, 0]),
    'MotorcycleCount_Pred':
np.round(y_pred_rescaled[:, 1]),
    'BusCount_Pred':
np.round(y_pred_rescaled[:, 2]),
```

```
    'TruckCount_Pred':
np.round(y_pred_rescaled[:, 3]),
    'Total_Pred':
np.round(y_pred_rescaled[:, 4]),
    'CarCount_True':
np.round(y_test_rescaled[:, 0]),
    'MotorcycleCount_True':
np.round(y_test_rescaled[:, 1]),
    'BusCount_True':
np.round(y_test_rescaled[:, 2]),
    'TruckCount_True':
np.round(y_test_rescaled[:, 3]),
    'Total_True':
np.round(y_test_rescaled[:, 4])
})
predictions_df.to_csv('/content/driv
e/My   Drive/hasil_yolo_sw_lstm/LSTM-
KMA1.csv', index=False)
# Print results
print(f"Best               Parameters:
{best_individual}")
print(f"Best      Fitness      (MAE):
{best_fitness_mae}")
print(f"Best      Fitness      (RMSE):
{best_fitness_rmse}")
print(predictions_df.head())
# Print execution time
end_time = time.time()
total_runtime    =    (end_time    -
start_time) / 60
print(f"Total                Runtime:
{total_runtime:.2f} menit")
```

BAB VIII Kontribusi dan Implikasi Teknologi Computer Vision untuk Transportasi Cerdas

Teknologi computer vision telah memberikan kontribusi yang signifikan dalam pengembangan sistem transportasi cerdas (Intelligent Transportation Systems/ITS). Dengan kemampuan untuk menganalisis dan memahami data visual dari berbagai sumber seperti kamera pengawas, drone, dan perangkat mobile, teknologi ini mampu mendukung berbagai aspek pengelolaan transportasi modern. Kontribusi utama computer vision dalam transportasi cerdas mencakup peningkatan efisiensi operasional, pengurangan kemacetan, peningkatan keselamatan, dan perencanaan infrastruktur yang lebih baik.

Salah satu kontribusi utama computer vision adalah dalam mendeteksi dan mengklasifikasikan kendaraan di jalan raya. Algoritma deteksi objek, seperti yang diterapkan dalam model-model deep learning seperti YOLO (You Only Look Once) dan Faster R-CNN, memungkinkan pengenalan kendaraan dengan tingkat akurasi yang tinggi. Teknologi ini tidak hanya mampu membedakan jenis kendaraan seperti mobil, sepeda motor, bus, dan truk, tetapi juga dapat menghitung jumlah kendaraan secara real-time. Informasi ini sangat penting untuk mengelola lalu lintas di persimpangan yang padat, memberikan data yang akurat untuk sistem pengendalian lalu lintas adaptif, dan membantu

pemerintah dalam merancang kebijakan transportasi berbasis data.

Selain itu, computer vision juga berperan dalam analisis perilaku pengemudi. Teknologi ini memungkinkan deteksi perilaku berisiko seperti penggunaan ponsel saat mengemudi, mengantuk, atau tidak menggunakan sabuk pengaman. Dengan menggunakan kamera yang terpasang di dalam kendaraan atau di jalan raya, sistem dapat memberikan peringatan kepada pengemudi secara real-time atau mengumpulkan data untuk analisis lebih lanjut. Hal ini berkontribusi pada peningkatan keselamatan di jalan raya, yang merupakan salah satu tujuan utama transportasi cerdas.

Implikasi lain dari computer vision adalah dalam sistem navigasi dan pengelolaan transportasi umum. Misalnya, teknologi pengenalan wajah dapat digunakan untuk sistem pembayaran otomatis di transportasi umum, sehingga mempercepat proses masuk dan keluar penumpang. Di sisi lain, analisis citra dapat digunakan untuk memantau kepadatan penumpang di stasiun atau halte, memungkinkan operator untuk menyesuaikan jadwal layanan sesuai kebutuhan. Dengan demikian, pengalaman pengguna dalam menggunakan transportasi umum dapat ditingkatkan secara signifikan.

Penggunaan computer vision juga telah merambah ke pengelolaan parkir. Sistem berbasis kamera dapat mendeteksi ketersediaan tempat parkir secara real-time dan mengarahkan pengemudi ke lokasi yang tersedia. Hal ini tidak hanya mengurangi waktu pencarian parkir, tetapi juga mengurangi emisi kendaraan yang

dihasilkan selama proses pencarian tersebut. Selain itu, teknologi ini dapat digunakan untuk mendeteksi pelanggaran parkir, membantu otoritas dalam menegakkan aturan dengan lebih efisien.

Dalam konteks perencanaan infrastruktur, computer vision dapat membantu dalam analisis data lalu lintas jangka panjang. Dengan menganalisis pola pergerakan kendaraan dan pejalan kaki, pemerintah dapat merancang jalan, jembatan, dan fasilitas transportasi lainnya yang lebih sesuai dengan kebutuhan masyarakat. Teknologi ini juga dapat digunakan untuk memantau kondisi infrastruktur secara otomatis, seperti mendeteksi kerusakan jalan atau jembatan melalui citra yang diambil oleh drone atau kendaraan inspeksi.

Namun, meskipun memiliki banyak manfaat, implementasi computer vision dalam transportasi cerdas juga menghadapi berbagai tantangan. Salah satu tantangan utama adalah kebutuhan akan data yang besar dan berkualitas tinggi untuk melatih model AI. Selain itu, masalah privasi dan keamanan data juga menjadi perhatian, terutama ketika menggunakan teknologi seperti pengenalan wajah. Oleh karena itu, diperlukan kerangka kerja yang kuat untuk melindungi data pengguna dan memastikan penggunaan teknologi ini secara etis.

Secara keseluruhan, kontribusi dan implikasi teknologi computer vision dalam transportasi cerdas sangatlah luas. Dengan terus berkembangnya teknologi ini, diharapkan berbagai tantangan transportasi modern dapat diatasi, menciptakan sistem transportasi yang

lebih efisien, aman, dan ramah lingkungan. Hal ini tidak hanya memberikan manfaat langsung bagi pengguna jalan, tetapi juga mendukung pembangunan kota pintar yang berkelanjutan.

Daftar Pustaka

Abdel-aty, M., Wang, Z., Zheng, O., & Abdelraouf, A. (2023). Advances and applications of computer vision techniques in vehicle trajectory generation and surrogate traffic safety indicators. Accident Analysis and Prevention, 191(July), 107191. https://doi.org/10.1016/j.aap.2023.107191

Abirami, A., & Kavitha, R. (2023). A novel automated komodo Mlipir optimization-based attention BiLSTM for early detection of diabetic retinopathy. Signal, Image and Video Processing, 17(5), 1945–1953. https://doi.org/10.1007/s11760-022-02407-9

Almatar, K. M. (2024). Smart transportation planning and its challenges in the Kingdom of Saudi Arabia. Sustainable Futures, 8(December 2023), 100238. https://doi.org/10.1016/j.sftr.2024.100238

ARSITO ARI KUNCORO S.Kom., M. K. (2022). Deep learning 简介 一 、 什么是 Deep Learning ？. Nature, 29(7553), 1–73. http://deeplearning.net/

Awaluddin, B., & Chao, C. (2023). Investigating Effective Geometric Transformation for Image Augmentation to Improve Static Hand Gestures with a Pre-Trained Convolutional Neural Network.

Bharti, Redhu, P., & Kumar, K. (2023). Short-term traffic flow prediction based on optimized deep learning neural network: PSO-Bi-LSTM. Physica A: Statistical Mechanics and Its Applications, 625,

129001. https://doi.org/10.1016/j.physa.2023.129001
Changxi, M., Yanming, H., & Xuecai, X. (2024). na l P re r f. Data Science and Management. https://doi.org/10.1016/j.dsm.2024.10.004
CHAOURA, C., LAZAR, H., & JARIR, Z. (2024). Traffic Flow Prediction at Intersections: Enhancing with a Hybrid LSTM-PSO Approach. International Journal of Advanced Computer Science and Applications, 15(5), 494–501. https://doi.org/10.14569/IJACSA.2024.0150549
Chauhan, N. S., & Kumar, N. (2024). Confined attention mechanism enabled Recurrent Neural Network framework to improve traffic flow prediction. Engineering Applications of Artificial Intelligence, 136(December 2022). https://doi.org/10.1016/j.engappai.2024.108791
Chi, H., Lu, Y., Xie, C., Ke, W., & Chen, B. (2025). Engineering Applications of Artificial Intelligence Spatio-temporal attention based collaborative local – global learning for traffic flow prediction. Engineering Applications of Artificial Intelligence, 139(PB), 109575. https://doi.org/10.1016/j.engappai.2024.109575
Ćorović, A., Ilić, V., Marijan, M., & Pavkovi, B. (2018). The Real-Time Detection of Traffic Participants Using YOLO Algorithm. November. https://doi.org/10.1109/TELFOR.2018.8611986
Detection, O. (2023). YOLO-Drone : An Optimized YOLOv8 Network for Tiny UAV Object Detection.

Dewi, C., Chen, R., Jiang, X., & Yu, H. (n.d.). Deep Convolutional Neural Network for Enhancing Traffic Sign Recognition developed on Yolo V4.

Dilek, E., & Dener, M. (2023). Computer Vision Applications in Intelligent Transportation Systems: A Survey. Sensors, 23(6). https://doi.org/10.3390/s23062938

Elassy, M., Al-Hattab, M., Takruri, M., & Badawi, S. (2024). Intelligent transportation systems for sustainable smart cities. Transportation Engineering, 100252.

Eldeen, N., Mohamed, K., & Seyedali, L. (2022). A comprehensive survey of recent trends in deep learning for digital images augmentation. Artificial Intelligence Review, 55(3), 2351–2377. https://doi.org/10.1007/s10462-021-10066-4

Firdaus, M., Arief, M. R., & Yogyakarta, U. A. (n.d.). Impact of Data Augmentation Techniques on the Implementation of a Combination Model of Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP) for the Detection of Diseases in Rice Plants. 2(2), 453–465.

Flores-Calero, M., Astudillo, C. A., Guevara, D., Maza, J., Lita, B. S., Defaz, B., Ante, J. S., Zabala-Blanco, D., & Armingol Moreno, J. M. (2024). Traffic Sign Detection and Recognition Using YOLO Object Detection Algorithm: A Systematic Review. Mathematics, 12(2), 1–31. https://doi.org/10.3390/math12020297

Gao, H., Jia, H., Huang, Q., Wu, R., Tian, J., Wang, G., & Liu, C. (2024). A hybrid deep learning model for

urban expressway lane-level mixed traffic flow prediction. Engineering Applications of Artificial Intelligence, 133(PB), 108242. https://doi.org/10.1016/j.engappai.2024.108242

Halpern-wight, N., Konstantinou, M., & Charalambides, A. G. (n.d.). applied sciences Training and Testing of a Single-Layer LSTM Network for Near-Future Solar Forecasting. 1–9. https://doi.org/10.3390/app10175873

Hamiane, S., Ghanou, Y., Khalifi, H., & Telmem, M. (2024). Comparative Analysis of LSTM, ARIMA, and Hybrid Models for Forecasting Future GDP. Ingenierie Des Systemes d'Information, 29(3), 853–861. https://doi.org/10.18280/isi.290306

Jalil, K., Xia, Y., Chen, Q., Noaman, M., Manzoor, T., & Zhao, J. (2024). Integrative review of data sciences for driving smart mobility in intelligent transportation systems. Computers and Electrical Engineering, 119(PB), 109624. https://doi.org/10.1016/j.compeleceng.2024.109624

Kablaoui, R., Ahmad, I., Abed, S., & Awad, M. (2024). Network traffic prediction by learning time series as images. Engineering Science and Technology, an International Journal, 55(May). https://doi.org/10.1016/j.jestch.2024.101754

Khan, S., Khan, S., Sulaiman, A., Saleh, M., Reshan, A., & Alshahrani, H. (2024). Deep neural network and trust management approach to secure smart transportation data in sustainable smart cities. 10, 1059–1065.

Kusuma, G., Kuwanto, G., Hashmi, T., & Widjaja, H. (2023). Discrete komodo algorithm for traveling salesman problem. 139.

Lawrence, B. L., & Lemmus, E. De. (2024). Science of Remote Sensing Using computer vision to classify , locate and segment fire behavior in UAS-captured images. Science of Remote Sensing, 10(July), 100167. https://doi.org/10.1016/j.srs.2024.100167

Lu, J. (2023). An efficient and intelligent traffic flow prediction method based on LSTM and variational modal decomposition. Measurement: Sensors, 28(June), 100843. https://doi.org/10.1016/j.measen.2023.100843

Luo, Y., Zheng, J., Wang, X., Tao, Y., & Jiang, X. (2024). GT-LSTM: A spatio-temporal ensemble network for traffic flow prediction. Neural Networks, 171(August 2023), 251–262. https://doi.org/10.1016/j.neunet.2023.12.016

Naheliya, B., Redhu, P., & Kumar, K. (2024). MFOA-Bi-LSTM: An optimized bidirectional long short-term memory model for short-term traffic flow prediction. Physica A: Statistical Mechanics and Its Applications, 634(November 2023), 129448. https://doi.org/10.1016/j.physa.2023.129448

Oladimeji, D., Gupta, K., Kose, N. A., Gundogan, K., Ge, L., & Liang, F. (2023). Smart Transportation: An Overview of Technologies and Applications. Sensors, 23(8), 1–32. https://doi.org/10.3390/s23083880

Pei, M., Liu, N., Zhao, B., & Sun, H. (2023). Self-Supervised Learning for Industrial Image Anomaly Detection by Simulating Anomalous Samples. International Journal of Computational Intelligence Systems, 16(1). https://doi.org/10.1007/s44196-023-00328-0

Ronariv, R., Antonio, R., Jorgensen, S. F., Achmad, S., Ronariv, R., & Antonio, R. (2024). ScienceDirect Object detection algorithms for car tracking with euclidean Object detection algorithms for car tracking distance tracking and YOLO with euclidean distance a tracking and YOLO. Procedia Computer Science, 245, 627–636. https://doi.org/10.1016/j.procs.2024.10.289

Saputri, H. A., Avrillio, M., Christofer, L., Simanjaya, V., & Alam, I. N. (2024). ScienceDirect ScienceDirect Implementation of YOLO v7 algorithm in estimating traffic Implementation of YOLO flow v7 in algorithm Malang in estimating traffic flow in Malang. Procedia Computer Science, 245(2022), 117–126. https://doi.org/10.1016/j.procs.2024.10.235

Sattarzadeh, A. R., Kutadinata, R. J., Pathirana, P. N., & Huynh, V. T. (2023). A novel hybrid deep learning model with ARIMA Conv-LSTM networks and shuffle attention layer for short-term traffic flow prediction. Transportmetrica A: Transport Science. https://doi.org/10.1080/23249935.2023.2236724

Shamta, I., Demir, F., & Demir, B. E. (2024). Predictive fault detection and resolution using YOLOv8 segmentation model: A comprehensive study on

hotspot faults and generalization challenges in computer vision. Ain Shams Engineering Journal, October. https://doi.org/10.1016/j.asej.2024.103148

Suyanto, S., Ariyanto, A. A., & Ariyanto, A. F. (2022). Komodo Mlipir Algorithm ☆. Applied Soft Computing, 114, 108043. https://doi.org/10.1016/j.asoc.2021.108043

Swathi, P., Tejaswi, D. S., Khan, M. A., Saishree, M., Rachapudi, V. B., & Anguraj, D. K. (2024). Real-Time Vehicle Detection for Traffic Monitoring: A Deep Learning Approach. Data and Metadata, 3, 295-295.

Tan, K. L., Lee, C. P. O. O., Sonai, K., & Anbananthen, M. (2022). RoBERTa-LSTM : A Hybrid Model for Sentiment Analysis With Transformer and Recurrent Neural Network. IEEE Access, 10, 21517–21525. https://doi.org/10.1109/ACCESS.2022.3152828

Tang, J., Ye, C., Zhou, X., & Xu, L. (2024). YOLO-Fusion and Internet of Things: Advancing object detection in smart transportation. Alexandria Engineering Journal, 107(August), 1–12. https://doi.org/10.1016/j.aej.2024.09.012

https://docs.ultralytics.com/

Wang, J. D., & Susanto, C. O. N. (2023). Traffic Flow Prediction with Heterogenous Data Using a Hybrid CNN-LSTM Model. Computers, Materials and Continua, 76(3), 3097–3112. https://doi.org/10.32604/cmc.2023.040914

Wang, K., Ma, C., Qiao, Y., Lu, X., Hao, W., & Dong, S. (2021). A hybrid deep learning model with 1DCNN-LSTM-Attention networks for short-term traffic flow prediction. Physica A: Statistical Mechanics and Its Applications, 583, 126293. https://doi.org/10.1016/j.physa.2021.126293

Xing, H., Chen, A., & Zhang, X. (2023). RL-GCN: Traffic flow prediction based on graph convolution and reinforcement learning for smart cities. Displays, 80(September). https://doi.org/10.1016/j.displa.2023.102513

Xu, S., Zhang, M., Chen, J., & Zhong, Y. (2024). YOLO-HyperVision: A vision transformer backbone-based enhancement of YOLOv5 for detection of dynamic traffic information. Egyptian Informatics Journal, 27(September 2023). https://doi.org/10.1016/j.eij.2024.100523.

Yang, B., Sun, S., Li, J., Lin, X., & Tian, Y. (2019). Traffic flow prediction using LSTM with feature enhancement. Neurocomputing, 332, 320-327.

Yang, C., Atinafu, A., Sutrisno, H., Haile, B., Phuong, T., & Nguyen, Q. (2023). LSTM-based framework with metaheuristic optimizer for manufacturing process monitoring. 83(October), 43–52.

Zhang, W., Yao, R., Yuan, Y., Du, X., Wang, L., & Sun, F. (2024). Engineering Applications of Artificial Intelligence A traffic-weather generative adversarial network for traffic flow prediction for road networks under bad weather. Engineering Applications of Artificial Intelligence, 137(PA),

109125. https://doi.org/10.1016/j.engappai.2024.109125

Zhao, K., Guo, D., Sun, M., Zhao, C., & Shuai, H. (2023). Short-Term Traffic Flow Prediction Based on VMD and IDBO-LSTM. IEEE Access, 11(August), 97072–97088. https://doi.org/10.1109/ACCESS.2023.3312711

Zhao, R., Tang, S. H., Shen, J., Supeni, E. E. Bin, & Rahim, S. A. (2024). Enhancing autonomous driving safety: A robust traffic sign detection and recognition model TSD-YOLO. Signal Processing, 225(July), 109619. https://doi.org/10.1016/j.sigpro.2024.109619

Zheng, J., Wang, M., & Huang, M. (2024). Exploring the relationship between data sample size and traffic flow prediction accuracy. Transportation Engineering, 18(September), 100279. https://doi.org/10.1016/j.treng.2024.100279

Zheng, Y., Li, X., Xu, L., & Wen, N. (2022). A deep learning–based approach for moving vehicle counting and short-term traffic prediction from video images. Frontiers in Environmental Science, 10, 905443.