Penerapan Algoritma *You Only Look Once V5 Large* (YOLOv5L) dan *Tesseract Optical Character Recognition* (*Tesseract* OCR) dalam Deteksi Plat Nomor Kendaraan untuk Cek Status Uji Emisi Kendaraan

Hashimatul Zaria*[1], Muh. Ihsan Sarita[2], Adha Mashur Sajiah[3]
Jurusan Informatika, Fakultas Teknik, Universitas Halu Oleo[1,2,3]
*e-mail: zariahashimatul@gmail.com

ABSTRAK

Peningkatan jumlah kendaraan bermotor di kawasan perkotaan, khususnya di Kota Kendari, berpotensi meningkatkan pencemaran udara akibat emisi gas buang kendaraan. Emisi ini mengandung senyawa berbahaya yang berdampak negatif terhadap kesehatan dan lingkungan. Sebagai upaya pengendalian, pemerintah telah menerapkan kebijakan uji emisi. Namun, pelaksanaannya masih terkendala oleh sistem pendataan manual, keterbatasan petugas, lambatnya proses identifikasi kendaraan, serta adanya ketidaksesuaian antara Peraturan Daerah Kota Kendari Nomor 8 Tahun 2015 dan Undang-Undang Nomor 22 Tahun 2009. Perbedaan regulasi ini menghambat optimalisasi pelaksanaan uji emisi oleh perangkat daerah. Penelitian ini bertujuan untuk merancang sistem otomatisasi deteksi status uji emisi kendaraan berbasis *computer vision*, dengan memanfaatkan algoritma YOLOv5L untuk mendeteksi plat nomor kendaraan, *Tesseract* OCR dan *EasyOCR* untuk mengenali karakter pada plat tersebut. Data karakter yang berhasil diekstraksi kemudian akan dicocokkan dengan *database* uji emisi guna mengetahui riwayat status uji emisi dari kendaraan yang bersangkutan. Pengujian sistem dilakukan dengan dua pendekatan, yaitu pengujian model deteksi dan pengujian algoritma OCR. Hasil pengujian model YOLOv5L menunjukkan bahwa sistem berhasil mendeteksi plat nomor kendaraan dengan nilai *Precision* sebesar 91%, *Recall* sebesar 98,1%, AP sebesar 91%, serta mAP (1 kelas), mAP50, dan mAP50-95 masing-masing sebesar 91%. Sementara itu, pengujian algoritma *Tesseract* OCR dan *EasyOCR* menghasilkan nilai *Accuracy* sebesar 0,93 (93%), *Precision* sebesar 0,95 (95%), *Recall* sebesar 0,98 (98%), dan *F1-Score* sebesar 0,96 (96%). Berdasarkan hasil tersebut, sistem dinyatakan layak digunakan karena mampu mendeteksi serta mengenali plat nomor kendaraan dengan baik untuk mendukung pengecekan status uji emisi secara otomatis.

Kata kunci : *Deteksi Objek; Plat Nomor Kendaraan; Tesseract OCR; Uji Emisi; You Only Look Once V5 Large.*

ABSTRACT

The increasing number of motor vehicles in urban areas, particularly in Kendari City, has the potential to worsen air pollution due to uncontrolled vehicle exhaust emissions. These emissions contain harmful compounds that negatively impact public health and the environment. As a control measure, the government has implemented an emission testing policy. However, its implementation still faces several obstacles, such as manual data recording systems, limited personnel, slow vehicle identification processes, and discrepancies between Kendari City Regional Regulation No. 8 of 2015 and Law No. 22 of 2009. These regulatory inconsistencies hinder the optimal implementation of vehicle emission testing by local government agencies. This study aims to design a computer vision-based vehicle emission test status detection automation system, utilizing the YOLOv5L algorithm to detect vehicle license plates, and Tesseract OCR and EasyOCR to recognize characters on the plates. The extracted character data will then be matched with the emission test database to determine the emission test status history of the vehicle in question. Detection and testing of OCR algorithms. The results of testing the YOLOv5L model show that the system successfully detected vehicle license plates with a Precision value of 91%, Recall of 98.1%, AP of 91%, and mAP (1 class), mAP50, and mAP50-95 of 91% each. Meanwhile, testing of the Tesseract OCR and EasyOCR algorithms produced an Accuracy value of 0.93 (93%), Precision of 0.95 (95%), Recall of 0.98 (98%), and F1-Score of 0.96 (96%). Based on these results, the system is deemed suitable for use as it is capable of detecting and recognizing vehicle license plates effectively to support automatic emission test status checks.

Keywords : *Emission Test; Object Detection; Tesseract OCR; Vehicle Number Plate; You Only Look Once V5 Large.*

PENDAHULUAN

Transportasi merupakan faktor yang berperan langsung dalam mendukung proses pembangunan, karena keberadaannya memudahkan masyarakat untuk berpindah dari satu tempat ke tempat lain (Saputro et al., 2022). Kehadiran transportasi menjadi bagian penting dalam kehidupan manusia untuk mendukung berbagai aktivitas harian, seperti bekerja, belajar, dan lainnya (Michelle et al., 2021). Jumlah kendaraan terus meningkat seiring waktu. Berdasarkan data Badan Pusat Statistik (BPS), jumlah kendaraan bermotor di Indonesia telah mencapai 157.080.504 unit pada tahun 2023, dengan wilayah Kota Kendari mencatatkan 47.163 unit kendaraan bermotor pada tahun yang sama, yang terdiri dari mobil penumpang 8.804 unit, truk 1.708 unit, dan sepeda motor 36.651 unit.

Berdasarkan Undang-Undang No. 22 Tahun 2009, kendaraan didefinisikan sebagai sarana transportasi yang digunakan di jalan, yang terbagi menjadi dua kategori utama. Pertama, kendaraan bermotor, yaitu kendaraan yang menggunakan mesin atau peralatan mekanik sebagai sumber tenaga penggeraknya, dengan pengecualian untuk kendaraan yang bergerak di atas rel. Kedua, kendaraan tidak bermotor, yaitu kendaraan yang digerakkan secara manual oleh tenaga manusia atau menggunakan tenaga hewan (Umriana et al., 2020).

Peningkatan jumlah kendaraan bermotor yang beroperasi di Kota Kendari turut memperburuk kualitas udara. Emisi gas buang, terutama dari kendaraan bermotor berukuran besar, menjadi salah satu penyebab utama menurunnya kualitas udara. Paparan asap hitam dari kendaraan dapat langsung berdampak pada pengendara lain, menyebabkan gangguan seperti batuk, sesak napas, dan iritasi mata. Oleh karena itu, diperlukan upaya untuk mengurangi emisi dari kendaraan bermotor, guna meningkatkan kualitas udara dan menjaga kesehatan masyarakat. Untuk mengatasi masalah pencemaran udara yang disebabkan oleh kendaraan bermotor, pemerintah Indonesia telah memberlakukan kebijakan uji emisi gas buang kendaraan bermotor secara berkala.

Uji emisi adalah regulasi pengendalian lingkungan yang bertujuan untuk mengurangi emisi gas buang yang dihasilkan oleh berbagai jenis kendaraan yang beroperasi di wilayah perkotaan (Suryotomo et al., 2024). Uji emisi memiliki tujuan utama untuk menekan tingkat polusi udara yang dihasilkan oleh kendaraan bermotor (Muzaky et al., 2024). Uji emisi dilakukan menggunakan alat khusus yang dirancang untuk mengukur emisi gas buang dari kendaraan bermotor. Gas buang ini mengandung sejumlah zat berbahaya, termasuk Karbon Monoksida (CO), Hidrokarbon (HC), Nitrogen Oksida (NOx), Sulfur Dioksida (SOx), dan Partikulat (PM10). Emisi yang melebihi ambang batas dapat berdampak negatif pada kesehatan manusia dan menyebabkan kerusakan lingkungan (Ishma Safira, 2023).

Pencegahan dan penanggulangan dampak lingkungan hidup akibat transportasi juga telah diatur dalam Pasal 209 ayat (1) Undang-Undang Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan. Uji emisi kendaraan bermotor merupakan kewenangan Pemerintah Daerah, sebagaimana diatur dalam Peraturan Daerah Kota Kendari Nomor 8 Tahun 2015 tentang Pengendalian Pencemaran Udara. Namun, implementasi kebijakan ini menghadapi tantangan signifikan, salah satunya adalah kurangnya harmonisasi antara Peraturan Daerah tersebut dan Undang-Undang Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan. Ketidaksesuaian aturan ini menghambat optimalisasi pelaksanaan uji emisi oleh organisasi perangkat daerah di Kota Kendari (Heryanti et al., 2024).

Selain itu, kegiatan uji emisi kendaraan mengalami kendala akibat tingginya jumlah kendaraan yang harus diperiksa secara bersamaan, sehingga menghambat kelancaran proses pelayanan. Metode manual dalam memeriksa plat nomor kendaraan kerap kali kurang efisien, karena membutuhkan waktu lebih lama untuk mencari dan mencocokkan data kendaraan secara manual. Hal ini menyebabkan antrean panjang dan waktu tunggu yang tidak optimal bagi pemilik kendaraan (Kharisma, 2021). Selain waktu yang tidak efisien, deteksi plat nomor secara manual juga rentan terhadap kesalahan penulisan. Hal ini biasanya disebabkan oleh faktor manusia, seperti kelelahan, kurangnya fokus, atau kondisi lingkungan yang kurang mendukung (Fahmi Chairulloh Widia Sumantri & Sutisna, 2022). Untuk mengatasi permasalahan ini, penerapan

teknologi deteksi plat nomor otomatis menjadi solusi inovatif. Dibandingkan sistem berbasis *classifier*, YOLO lebih unggul karena dapat menganalisis seluruh gambar sekaligus selama pengujian, menghasilkan prediksi yang lebih cepat dan efisien (Evita et al., 2022). YOLOv5L adalah salah satu varian dari algoritma *You Only Look Once* (YOLO) yang dirancang untuk deteksi objek. Dengan jaringan yang lebih luas, model ini mampu menganalisis fitur lebih mendalam, meningkatkan akurasi deteksi dalam berbagai aplikasi seperti pengenalan wajah, deteksi kendaraan, dan pemantauan keamanan (Naftali et al., 2022). Dengan memanfaatkan algoritma *You Only Look Once Versi 5 Large* (YOLOv5L), yang dirancang untuk mendeteksi objek secara efisien, sistem otomatis ini dapat memangkas waktu yang dibutuhkan, memungkinkan lebih banyak kendaraan untuk diuji dalam waktu yang lebih singkat, sekaligus meningkatkan efisiensi di lokasi uji emisi.

YOLOv5L adalah varian YOLOv5 dengan ukuran model lebih besar, mampu menangkap fitur dan detail yang kompleks untuk meningkatkan akurasi deteksi objek dibandingkan varian lebih kecil seperti YOLOv5S dan YOLOv5M. Penelitian oleh Jia (2023) berjudul "*Application Effect Analysis for Helmet Detection Algorithm based on* YOLOv5" menunjukkan bahwa model YOLOv5 yang digunakan, termasuk YOLOv5S, YOLOv5M, dan YOLOv5L, menghasilkan performa terbaik pada YOLOv5L dengan *mean Average Precision* (mAP) tertinggi sebesar 91,9%. Ketika dikombinasikan dengan teknologi *Tesseract Optical Character Recognition* (*Tesseract* OCR), sistem ini mampu mendeteksi dan membaca plat nomor kendaraan secara otomatis, cepat, dan akurat. *Tesseract* OCR adalah algoritma *open source* yang mampu mengenali lebih dari 100 bahasa dan menghasilkan teks dari gambar secara *offline*.

Penelitian oleh Reezky Illmawati & Hustinawati (2023) mengintegrasikan algoritma YOLOv5 dan *Tesseract* OCR untuk mendeteksi serta mengenali plat nomor kendaraan. Hasilnya menunjukkan bahwa YOLOv5 mampu mendeteksi plat nomor dengan rata-rata deteksi objek sebesar 92,38% dan nilai kepercayaan 75,55%. Proses ekstraksi karakter pada plat nomor mencapai tingkat keberhasilan sebesar 95,45%, sementara rata-rata proporsi deteksi karakter sesuai kategori plat nomor yang diidentifikasi adalah sebesar 97,2%.

Berdasarkan latar belakang dan penelitian terdahulu yang telah diuraikan, maka diambil topik penelitian dengan judul Penerapan Algoritma *You Only Look Once V5 Large* (YOLOv5L) dan *Tesseract Optical Character Recognition* (*Tesseract* OCR) dalam Deteksi Plat Nomor Kendaraan untuk Cek Status Uji Emisi Kendaraan. Dengan menggunakan kombinasi algoritma YOLOv5L untuk deteksi objek dan *Tesseract* OCR untuk membaca plat nomor kendaraan, diharapkan sistem ini dapat mengetahui status uji emisi kendaraan melalui pencocokan data plat nomor dengan *database* uji emisi yang tersedia.

METODE PENELITIAN

Metode Pengumpulan Data

Beberapa metode pengumpulan data yang digunakan dalam penelitian ini adalah, sebagai berikut:

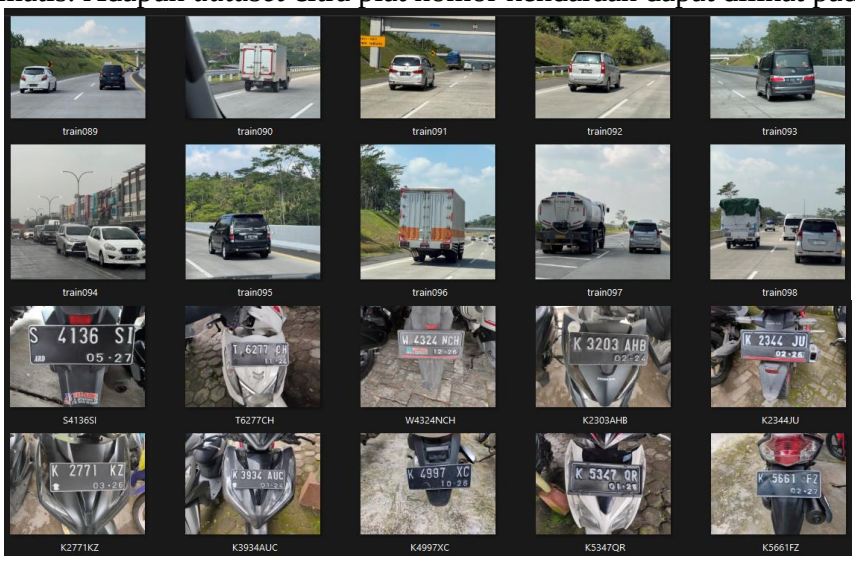
1. Studi Literatur

Langkah ini melibatkan proses pencarian, pengumpulan, dan analisis berbagai sumber informasi yang telah dipublikasikan, seperti jurnal ilmiah, buku, artikel, dan referensi terpercaya lainnya yang berkaitan dengan metode YOLOv5L, *Tesseract* OCR, *EasyOCR*, serta topik uji emisi kendaraan. Studi literatur ini bertujuan untuk memperoleh pemahaman yang mendalam mengenai konsep dasar, metode-metode yang telah diterapkan pada penelitian sebelumnya, serta mengevaluasi hasil-hasil penelitian terdahulu yang relevan sebagai dasar teori dan acuan penting dalam mendukung pelaksanaan dan pengembangan penelitian ini.

2. Dataset

Dalam rangka mendukung pelaksanaan penelitian ini, dilakukan proses pengumpulan *dataset* yang akan digunakan dalam pelatihan model menggunakan metode *You Only Look Once V5 Large* (YOLOv5L) dan *Tesseract Optical Character Recognition* (*Tesseract* OCR). Jumlah *dataset* mencapai 1158 citra plat nomor kendaraan yang diperoleh dari Kaggle. *Dataset* tersebut diperoleh dari dua sumber, yaitu pada *dataset Indonesian License Plate Dataset* (URL *dataset* dapat diakses pada *link*: https://www.kaggle.com/datasets/juanthomaswijaya/indonesian license-plate-dataset) yang diperoleh sebanyak 800 data, dan *Dataset* Plat Nomor Motor Indonesia (URL *dataset* dapat diakses pada *link*:

https://www.kaggle.com/datasets/caasperart/haarcascadeplatenumber) dengan jumlah 358 data. *Dataset* ini digunakan sebagai bahan pelatihan dan validasi model dalam mendeteksi serta mengenali plat nomor kendaraan secara otomatis. Adapun *dataset* citra plat nomor kendaraan dapat dilihat pada Gambar 1.



Gambar 1. *Dataset* Plat Nomor Kendaraan

3. Wawancara

Pengumpulan informasi dilakukan untuk memperoleh gambaran umum, keterangan, alur sistem, serta aspek-aspek lain yang diperlukan dalam memahami cara kerja sistem. Data diperoleh melalui wawancara dengan salah satu pegawai UPTD Pengujian Bermotor Dinas Perhubungan Kota Kendari. Wawancara tersebut membahas mengenai proses dan prosedur uji emisi kendaraan, waktu pelaksanaan uji emisi ulang, serta tahapan-tahapan yang dilakukan dalam pelaksanaan uji emisi, sehingga informasi ini menjadi dasar dalam perancangan sistem deteksi plat nomor untuk mendukung pengecekan status uji emisi kendaraan.

Metode Pengembangan Sistem

Dalam penelitian ini, digunakan metode pengembangan sistem yaitu *Rational Unified Process* (RUP), yang terdiri dari tahapan-tahapan utama yang saling terkait dan berkesinambungan, sehingga setiap fase dapat mendukung fase berikutnya untuk menghasilkan sistem yang lebih terstruktur, sebagai berikut:

1. *Inception* (Permulaan)

Pada fase *inception*, dilakukan analisis mendalam untuk memahami kebutuhan sistem secara menyeluruh. Selain itu, dilakukan peninjauan terhadap kebutuhan dasar pengguna serta pemahaman konsep utama penelitian, termasuk identifikasi pada proses deteksi plat nomor kendaraan dan mengenai uji emisi kendaraan. Pada tahap ini juga dilakukan analisis teori mengenai metode yang digunakan, yaitu model YOLOv5L untuk deteksi plat nomor, *Tesseract* OCR dan *EasyOCR* untuk ekstraksi teks, yang diperkuat dengan referensi penelitian terdahulu guna memastikan pendekatan yang dipilih relevan dengan tujuan penelitian.

2. *Elaboration* (Perluasan/Perencanaan)

Pada fase ini dilakukan perencanaan pembangunan model dengan menggunakan metode YOLOv5L untuk deteksi plat nomor kendaraan, serta metode *Tesseract* OCR dan *EasyOCR* untuk proses ekstraksi teks. Tahap ini juga mencakup perancangan arsitektur sistem yang menjadi dasar integrasi antara deteksi objek, pengolahan citra, dan pengenalan karakter. Selain itu, dibuat alur kerja sistem dalam bentuk *flowchart* untuk memberikan gambaran umum mengenai proses yang dijalankan mulai dari *input* citra hingga keluaran berupa informasi status uji emisi kendaraan.

Perancangan sistem kemudian dilanjutkan dengan pemodelan menggunakan *Unified Modeling Language* (UML), yang meliputi beberapa diagram penting. *Use case diagram* digunakan untuk menggambarkan kebutuhan fungsional antara pengguna dan sistem, *activity diagram* menjelaskan alur aktivitas proses deteksi plat nomor serta pengecekan status uji emisi, termasuk interaksi pengguna dalam

menambah, mengedit, melihat, dan menghapus data, sedangkan *class diagram* memodelkan struktur data dan hubungan antar kelas dalam sistem. Selanjutnya, dilakukan pembuatan desain antarmuka pengguna atau *Graphical User Interface* (GUI) yang dirancang sederhana, interaktif, dan mudah digunakan, sehingga memudahkan pengguna dalam mengoperasikan sistem secara praktis dan efisien.

3. *Construction* (Konstruksi)

Pada fase *construction*, proses pengembangan sistem dilakukan secara menyeluruh dengan tujuan menghasilkan aplikasi yang dapat berfungsi secara optimal. Tahap ini melibatkan pembuatan antarmuka sistem yang *user-friendly*, penulisan kode program sesuai kebutuhan sistem, serta perancangan alur kerja agar lebih terstruktur. Pembuatan kode program dilakukan menggunakan bahasa pemrograman Python dengan *framework* Flask untuk pengelolaan *backend*, sementara antarmuka pengguna dikembangkan berbasis web menggunakan HTML, CSS, dan JavaScript agar sistem mudah digunakan oleh pengguna.

Selain itu, implementasi berbagai langkah pengolahan citra juga dilakukan sesuai dengan kebutuhan penelitian. Beberapa tahapan pengolahan citra tersebut mencakup *resizing* menggunakan interpolasi *bilinear* untuk menyesuaikan ukuran citra, *cropping* area plat nomor agar fokus pada objek utama, normalisasi piksel untuk menyeragamkan intensitas cahaya, dan segmentasi karakter menggunakan metode *otsu thresholding* untuk memisahkan huruf serta angka dari latar belakang. Pada tahap ini juga diintegrasikan model YOLOv5L untuk deteksi plat nomor kendaraan, serta penerapan *Tesseract* OCR dan *EasyOCR* untuk ekstraksi teks, sehingga sistem mampu mengenali plat nomor kendaraan secara otomatis dan mendukung pengecekan status uji emisi kendaraan.

4. *Transition* (Transisi)

Fase *transition* merupakan tahap akhir dalam proses pengembangan sistem yang berfokus pada implementasi ke lingkungan pengguna. Pada tahap ini dilakukan pengujian untuk mengevaluasi kinerja sistem secara menyeluruh, dengan mendeteksi plat nomor kendaraan yang belum pernah digunakan pada proses pelatihan sebelumnya guna memeriksa status uji emisi. Pengujian difokuskan pada algoritma YOLOv5L untuk memastikan kemampuannya dalam mendeteksi area plat nomor kendaraan sebagai objek utama, serta pada *Tesseract* OCR dan *EasyOCR* untuk menilai sejauh mana algoritma tersebut mampu mengekstraksi karakter dari citra menjadi teks dengan akurat. Hal ini bertujuan untuk menilai kemampuan sistem dalam mengenali data baru secara akurat serta membuktikan keandalannya dalam proses deteksi plat nomor kendaraan.

Waktu dan Tempat Penelitian

Waktu pelaksanaan penelitian berlangsung dari bulan April 2025 sampai Oktober 2025. Penelitian ini dilakukan di Dinas Lingkungan Hidup Kota Kendari dan Laboratorium *Computer Science and Artificial Intelligence*, Jurusan Informatika, Fakultas Teknik, Universitas Halu Oleo, Kota Kendari, Sulawesi Tenggara.

HASIL DAN PEMBAHASAN

Implementasi Sistem

Implementasi adalah tahap dimana kode program diterapkan untuk membuat sistem berdasarkan rancangan dan desain sistem yang telah dibuat sebelumnya. Pada konteks deteksi plat dengan menggunakan algoritma *You Only Look Once V5 Large* (YOLOv5L), *Tesseract* OCR, dan *EasyOCR* berbasis citra digital plat kendaraan, implementasi melibatkan proses penerapan algoritma ke dalam kode program yang dapat memproses citra digital plat kendaraan untuk menentukan status uji emisi kendaraan. Implementasi sistem mencakup penjelasan tentang penggunaan data, penerapan antarmuka (*interface*), implementasi algoritma YOLOv5L, *Tesseract* OCR, dan *EasyOCR*, serta pengujian sistem untuk memastikan bahwa aplikasi dapat berfungsi dengan baik dalam menentukan status uji emisi kendaraan secara akurat dan efisien.

Data

Data yang digunakan dalam penelitian ini terdiri dari dua kategori utama, yaitu data citra untuk plat nomor kendaraan dan data uji emisi kendaraan. Penjelasan lebih rinci dijabarkan sebagai berikut:

1. Data Citra dan Anotasi Plat Nomor

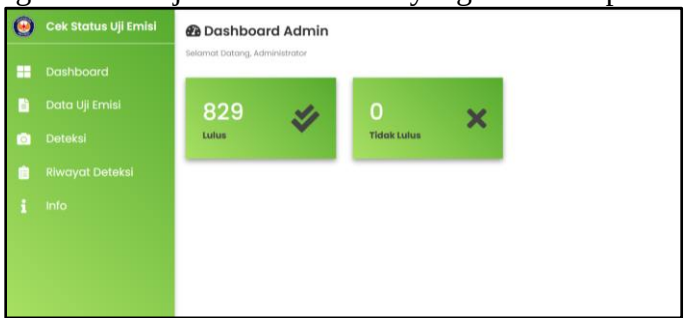
Penelitian ini menggunakan data citra plat nomor kendaraan dalam format .jpg sebanyak 4.037 gambar, yang telah dilabeli menggunakan platform Roboflow. Proses pelabelan menghasilkan *file* anotasi dalam format .txt, berisi informasi posisi *bounding box* dari setiap plat nomor yang terdeteksi. *Dataset* ini kemudian dibagi ke dalam tiga bagian, yaitu 70% untuk pelatihan (2.826 gambar), 20% untuk validasi (807 gambar), dan 10% untuk pengujian (404 gambar). Pembagian ini dilakukan untuk memastikan bahwa model YOLOv5L dapat dilatih secara optimal, divalidasi selama proses pelatihan, serta dievaluasi kinerjanya terhadap data yang belum pernah dilihat sebelumnya.

2. Data Uji Emisi Kendaraan

Selain data citra, penelitian ini juga menggunakan data uji emisi kendaraan yang diperoleh dari UPTD Pengujian Bermotor Dinas Perhubungan Kota Kendari. Data ini mencakup informasi kendaraan seperti nomor plat, tanggal uji, jenis kendaraan, status kelulusan, dan lainnya, yang dikumpulkan selama periode Januari hingga Maret 2025 sebanyak 829 data. Data ini digunakan untuk mencocokkan hasil deteksi plat nomor dengan status uji emisi kendaraan dalam sistem.

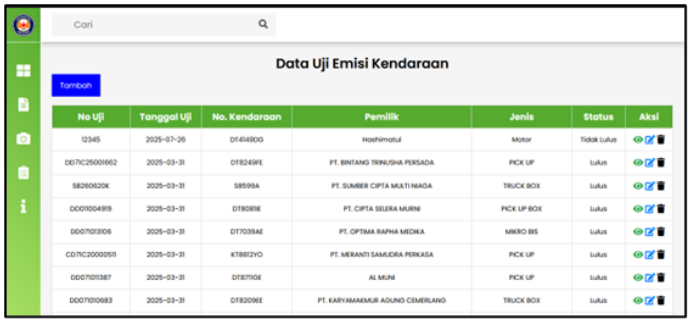
Implementasi Antarmuka *(Interface)*

Pada gambar berikut menampilkan halaman utama atau *dashboard* dari sistem pemeriksaan status uji emisi kendaraan, yang berisi ringkasan data jumlah kendaraan yang lulus maupun tidak lulus uji emisi.

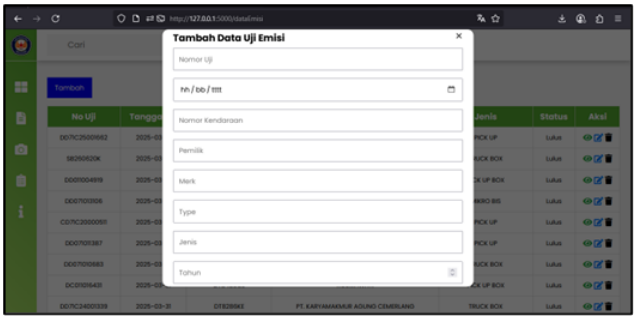


Gambar 2. Halaman *Dashboard* Cek Status Uji Emisi

Pada Gambar 3 menampilkan halaman data uji emisi kendaraan. Pada halaman ini admin dapat mencari, menambah, mengedit, melihat, dan menghapus data. Adapun Gambar 4 menunjukkan tampilan formulir tambah data, Gambar 5 menampilkan tampilan edit data, dan Gambar 6 memperlihatkan tampilan detail atau lihat data.



Gambar 3. Tampilan Halaman Data Uji Emisi

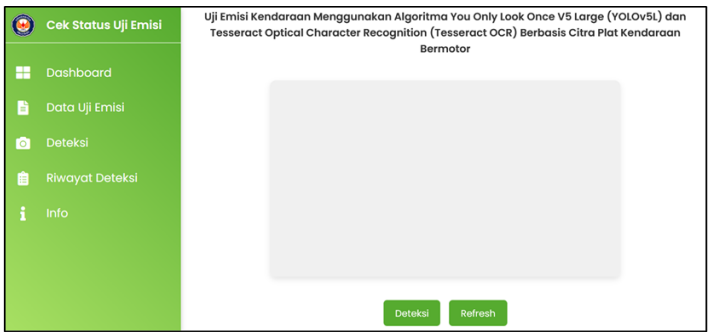


Gambar 4. Tampilan Tambah Data

Gambar 5. Tampilan Edit Data

Gambar 6. Tampilan Lihat Data

Pada Gambar 7 menampilkan halaman deteksi status uji emisi kendaraan berbasis citra plat nomor, dimana proses deteksi dan pengambilan citra dilakukan dengan menekan tombol Deteksi. Pada halaman ini juga terdapat *canvas* yang menampilkan tampilan kamera untuk memantau gambar *input* secara langsung sebelum diproses.



Gambar 7. Tampilan Halaman Deteksi

Gambar 8 menampilkan halaman rekap data hasil deteksi yang berfungsi sebagai arsip riwayat kendaraan yang telah melalui proses deteksi. Halaman ini memuat informasi berupa tanggal, gambar, nomor plat, status uji emisi, status berlaku, status denda, serta opsi untuk menghapus data.

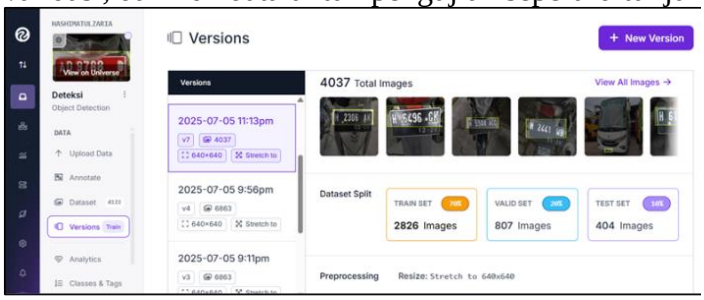
Gambar 8. Tampilan Halaman Riwayat Deteksi

Pembuatan Model YOLOv5L

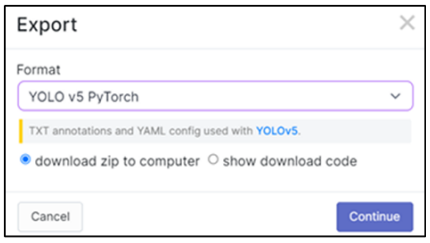
Pre-processing Data

Data gambar yang telah dikumpulkan sebelumnya selanjutnya akan diolah terlebih dahulu sebelum dimasukkan ke dalam sistem deteksi status uji emisi kendaraan. Tahap *pre-processing* yang dilakukan adalah pelabelan (*labelling*), yaitu proses anotasi citra plat kendaraan dengan memberi tanda objek menggunakan kotak pembatas (*bounding box*) di platform Roboflow. Anotasi ini memberikan informasi berupa label dan posisi objek yang digunakan dalam pelatihan model YOLO agar dapat mengenali objek pada citra.

Setelah proses pelabelan selesai, *dataset* dibagi menjadi tiga bagian, yaitu data pelatihan (*training*), validasi (*validation*), dan pengujian (*testing*), dengan proporsi masing-masing 70%, 20%, dan 10%. Dari pembagian tersebut, diperoleh total 4037 data, yang terdiri atas 2826 data untuk pelatihan, 807 data untuk validasi, dan 404 data untuk pengujian seperti ditunjukkan pada Gambar 9.



Gambar 9. Pembagian *Dataset*



Gambar 10. Proses Ekspor *Dataset*

Dataset yang telah dibagi dan dilabeli kemudian diekspor ke dalam format YOLOv5 untuk digunakan dalam proses pelatihan model, seperti ditunjukkan pada Gambar 10.

File yang diunduh terdiri dari gambar, *file* label dalam format .txt, serta *file* data.yaml yang akan digunakan dalam proses pelatihan model YOLOv5L. Setiap gambar yang telah diberi label akan menghasilkan *file* .txt sebagai anotasi yang sesuai dengan format YOLO, sehingga memudahkan sistem dalam membaca dan memahami data selama proses pelatihan model.

File label dengan format YOLO .txt berbentuk *array* atau tabel yang terdiri dari baris dan kolom. Setiap objek yang diberi *bounding box* akan direpresentasikan dalam satu baris dengan lima kolom, yaitu kolom pertama menunjukkan kelas objek, diikuti oleh koordinat pusat (x dan y), lebar, dan tinggi kotak pembatas (*bounding box*). Kelas 0 merepresentasikan objek berupa plat nomor kendaraan.

Proses *Training*

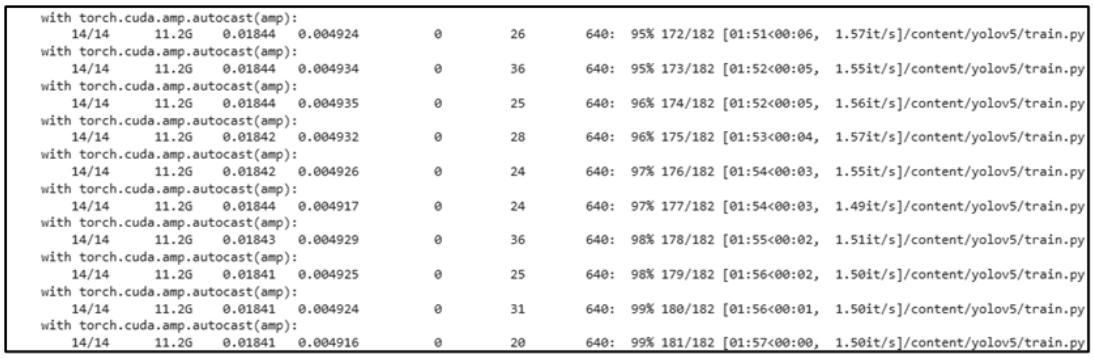
Tahap pertama dalam proses pelatihan (*training*) adalah menghubungkan YOLOv5 dengan Google Colab melalui GitHub, dengan menjalankan perintah untuk meng-*clone repository* Ultralytics YOLOv5 serta meng-*install* dependensi yang diperlukan. Langkah selanjutnya adalah mengunggah folder *dataset* yang telah diberi label, yang akan digunakan dalam pelatihan model YOLOv5L. *Dataset* yang diunggah ke Google Colab memiliki ekstensi .zip. Agar dapat digunakan dalam proses *training* dengan YOLOv5, *file* .zip tersebut perlu diekstrak terlebih dahulu. Data yang telah diekstrak selanjutnya akan melalui proses *resizing* dan normalisasi piksel pada *dataset train*, *valid*, dan *test*. Tahapan ini dilakukan sebelum pelatihan model agar mempercepat dan mengoptimalkan proses *training.* Langkah selanjutnya adalah mengubah *file* data.yaml yang berada pada direktori yolov5/data/data.yaml. Perubahan yang dilakukan meliputi penyesuaian *path* untuk folder *train*, *valid*, dan *test*, serta penyesuaian jumlah kelas dan nama kelas. *Path* tersebut berfungsi untuk menunjukkan lokasi masing-masing folder *dataset* yang akan digunakan dalam proses pelatihan model.

Pada penelitian ini, digunakan model YOLOv5L. Gambar 11 menampilkan spesifikasi arsitektur model YOLOv5L yang digunakan dalam proses pelatihan. *File* model.yaml telah disesuaikan dengan parameter nc=1, yang menunjukkan bahwa model ini dilatih untuk mendeteksi satu kelas objek, yaitu plat nomor kendaraan. Arsitektur model terdiri dari beberapa komponen utama seperti *Conv*, C3, SPPF, *Concat*, dan *Upsample*, yang masing-masing memiliki konfigurasi tertentu, seperti jumlah kanal, ukuran *kernel*, dan nilai *stride*. Pada bagian akhir, modul *Detect* berfungsi untuk melakukan prediksi terhadap objek berdasarkan fitur yang telah diproses oleh lapisan sebelumnya. Selanjutnya, Gambar 12 menampilkan proses pelatihan (*training*) model yang sedang berlangsung.

```
Overriding model.yaml nc=80 with nc=1

                 from  n    params  module                                  arguments
  0                -1  1      7040  models.common.Conv                      [3, 64, 6, 2, 2]
  1                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  2                -1  3    156928  models.common.C3                        [128, 128, 3]
  3                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  4                -1  6   1118208  models.common.C3                        [256, 256, 6]
  5                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  6                -1  9   6433792  models.common.C3                        [512, 512, 9]
  7                -1  1   4720640  models.common.Conv                      [512, 1024, 3, 2]
  8                -1  3   9971712  models.common.C3                        [1024, 1024, 3]
  9                -1  1   2624512  models.common.SPPF                      [1024, 1024, 5]
 10                -1  1    525312  models.common.Conv                      [1024, 512, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  3   2757632  models.common.C3                        [1024, 512, 3, False]
 14                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  3    690688  models.common.C3                        [512, 256, 3, False]
 18                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  3   2495488  models.common.C3                        [512, 512, 3, False]
 21                -1  1   2360320  models.common.Conv                      [512, 512, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  3   9971712  models.common.C3                        [1024, 1024, 3, False]
 24      [17, 20, 23]  1     32310  models.yolo.Detect                      [1, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119
Model summary: 368 layers, 46138294 parameters, 46138294 gradients, 108.2 GFLOPs
```

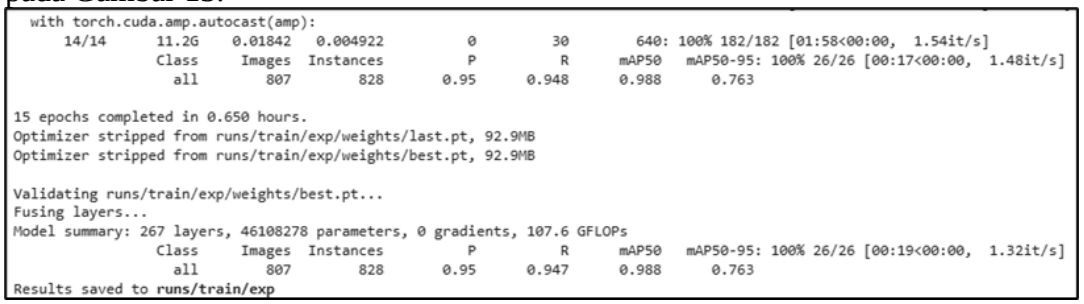
Gambar 11. Spesifikasi Arsitektur Model YOLOv5L



Gambar 12. Proses *Training* Model

Hasil *Training*

Proses pelatihan model akan menghasilkan *file* berformat *weights* yang disimpan secara otomatis oleh YOLOv5 di dalam direktori /yolov5/*runs*/*train*, dengan nama *file best.pt*. *File* tersebut tersimpan di dalam *subfolder* bernama *exp*, yang dibuat secara otomatis setelah pelatihan selesai. Hasil dari proses pelatihan ini ditampilkan pada Gambar 13.



Gambar 13. Hasil *Training Dataset* Model YOLOv5L

Berdasarkan hasil *training* yang ditampilkan pada Gambar 13, diketahui bahwa jumlah data validasi yang digunakan adalah sebanyak 807 gambar dengan total 828 objek (*instances*) yang berhasil diberi label atau *bounding box*. Proses pelatihan dilakukan selama 15 *epoch* dalam waktu sekitar 0.650 jam (39 menit). Hasil evaluasi model menunjukkan performa yang sangat baik, dengan nilai *Precision* (P) sebesar 95%, *Recall* (R) sebesar 94,7%, dan mAP50 sebesar 98.8%. Sementara itu, nilai mAP50-95 berada pada angka 76,3%. Setelah pelatihan, model *best.pt* dievaluasi pada data uji menggunakan metrik seperti *precision*, *recall*, AP, mAP50, dan mAP50-95, diawali dengan mengimpor *library* evaluasi YOLOv5L. Berikut, adalah hasil akhir dari perhitungan evaluasi performa model YOLOv5L yang telah diuji:

Tabel 1. Hasil Evaluasi Model YOLOv5L

Metrik	**Nilai**
Precision	91%
Recall	98,1%
AP	91%
mAP (1 kelas)	91%
mAP50	91%
mAP50-95	91%

Implementasi Algoritma *You Only Look Once V5 Large* (YOLOv5L)

Implementasi algoritma YOLOv5L digunakan untuk mendeteksi area plat nomor kendaraan pada citra. Proses deteksi dijalankan menggunakan bahasa pemrograman Python dengan pustaka PyTorch untuk memuat model hasil pelatihan (*best.pt*), serta OpenCV dan NumPy untuk pengolahan citra. Pada fungsi detect_plate_yolo, sistem terlebih dahulu membaca citra *input* dan melakukan proses *letterbox* agar ukuran gambar sesuai dengan resolusi yang dibutuhkan model. Selanjutnya, citra dikonversi ke format RGB, diubah menjadi *tensor*, dan dinormalisasi. Model kemudian dijalankan dalam mode evaluasi untuk menghasilkan prediksi berupa koordinat ***bounding box***, tingkat kepercayaan (*confidence score*), dan kelas objek terdeteksi. Hasil prediksi selanjutnya diproses menggunakan fungsi *Non-Maximum Suppression* untuk menghilangkan deteksi ganda, dan fungsi *scale_coords* digunakan untuk menyesuaikan koordinat *bounding box* ke ukuran asli citra.

Setiap *bounding box* yang terdeteksi kemudian dipotong (*cropping*) dari citra asli, dengan tambahan jarak tepi (*padding*) untuk memastikan area plat nomor terambil secara utuh. Potongan citra ini disimpan dalam bentuk daftar yang berisi gambar potongan plat dan koordinatnya. Kode program untuk implementasi deteksi plat nomor dengan YOLOv5 dapat dilihat pada Gambar 14.

```
#Import Library
import torch
import cv2
import numpy as np
from utils.general import non_max_suppression, scale_coords
from utils.datasets import letterbox

# Load YOLOv5 model manual tanpa attempt_load
device = torch.device('cpu')
checkpoint = torch.load('best.pt', map_location=device)
model = checkpoint['model'].float().eval()
stride = int(model.stride.max())
img_size = 640
conf_thres = 0.5
iou_thres = 0.5

# ==== Fungsi Utama ====
def detect_and_ocr(image_path):
    start_time = time.time()
    debug_img = cv2.imread(image_path)
    os.makedirs('static/hasil_debug', exist_ok=True)

    # Resize dan Letterbox
    image_resized = letterbox(debug_img, img_size, stride=stride)[0]
    img_rgb = cv2.cvtColor(image_resized, cv2.COLOR_BGR2RGB)
    img_tensor = torch.from_numpy(img_rgb).to(device)
    img_tensor = img_tensor.permute(2, 0, 1).float() / 255.0
    if img_tensor.ndimension() == 3:
        img_tensor = img_tensor.unsqueeze(0)

#Proses Deteksi YOLOv5L
with torch.no_grad():
        pred = model(img_tensor)[0]
        detections = non_max_suppression(pred, conf_thres,
        iou_thres)[0]

    #Validasi hasil deteksi
    if detections is None or len(detections) == 0:
        print("⚠️Tidak ada deteksi dari YOLO - proses OCR dibatalkan.")
        return "", None

    #Proses bounding box
    detections = detections.cpu().numpy()
    all_texts = []

    for idx, det in enumerate(detections):
        x1, y1, x2, y2, conf, cls = det
        x1, y1, x2, y2 = map(int, scale_coords(img_tensor.shape[2:],
        np.array([[x1, y1, x2, y2]]), debug_img.shape[:2]).squeeze())

        #Cropping plat nomor kendaraan hasil deteksi
        pad = 10
        x1, y1 = max(0, x1 - pad), max(0, y1 - pad)
        x2, y2 = min(debug_img.shape[1], x2 + pad),
        min(debug_img.shape[0], y2 + pad)
        cropped = debug_img[y1:y2, x1:x2]
```

Gambar 14. Implementasi *Code* Algoritma YOLOv5L

Implementasi Algoritma *Tesseract OCR* dan *EasyOCR*

Implementasi Algoritma *Tesseract OCR* dan *EasyOCR* digunakan untuk mengubah citra menjadi teks. Setelah plat nomor terdeteksi dan di-*crop*, dilakukan *pre-processing* untuk meningkatkan kualitas citra sebelum pendeteksian karakter dengan OCR.

1. *Resizing* (*def bl_resized*)

Resizing digunakan untuk mengubah ukuran citra menggunakan metode *bilinear interpolation*. Setiap piksel pada citra baru dihitung sebagai hasil interpolasi dari piksel-piksel tetangga pada citra asli, sehingga menghasilkan kualitas perbesaran atau pengecilan gambar. Proses ini dilakukan secara manual dengan menghitung skala tinggi dan lebar, kemudian mengkombinasikan nilai piksel secara proporsional. Kode program untuk implementasi *resizing* ditunjukkan pada Gambar 15, sedangkan hasil penerapannya dapat dilihat pada Gambar 16.

```
def bl_resize(original_img, new_h, new_w):
    old_h, old_w, c = original_img.shape
    resized = np.zeros((new_h, new_w, c))
    w_scale = old_w / new_w
    h_scale = old_h / new_h
    for i in range(new_h):
        for j in range(new_w):
            x = i * h_scale
            y = j * w_scale
            x_floor, x_ceil = math.floor(x), min(old_h - 1,
            math.ceil(x))
            y_floor, y_ceil = math.floor(y), min(old_w - 1,
            math.ceil(y))
            if x_ceil == x_floor and y_ceil == y_floor:
                q = original_img[int(x), int(y), :]
            elif x_ceil == x_floor:
                q = original_img[int(x), y_floor, :] * (y_ceil - y) +
                original_img[int(x), y_ceil, :] * (y - y_floor)
            elif y_ceil == y_floor:
                q = original_img[x_floor, int(y), :] * (x_ceil - x) +
                original_img[x_ceil, int(y), :] * (x - x_floor)
            else:
                v1 = original_img[x_floor, y_floor, :]
                v2 = original_img[x_ceil, y_floor, :]
                v3 = original_img[x_floor, y_ceil, :]
                v4 = original_img[x_ceil, y_ceil, :]
                q1 = v1 * (x_ceil - x) + v2 * (x - x_floor)
                q2 = v3 * (x_ceil - x) + v4 * (x - x_floor)
                q = q1 * (y_ceil - y) + q2 * (y - y_floor)
            resized[i, j, :] = q
    return resized.astype(np.uint8)
```

Gambar 15. Implementasi *Code Resizing*

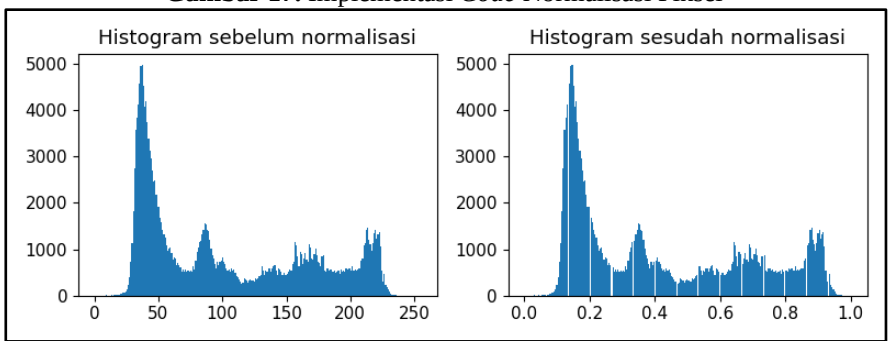
Gambar 16. Hasil *Resizing*

2. Normalisasi Piksel (*def normalize_image*)

Normalisasi digunakan untuk menyamakan distribusi nilai piksel sehingga perbedaan pencahayaan tidak terlalu memengaruhi hasil OCR. Jika citra memiliki nilai maksimum yang lebih besar dari nilai minimum, normalisasi dilakukan berdasarkan selisih keduanya jika tidak, piksel dibagi dengan 255 agar berada pada skala 0–1. Kode program untuk implementasi normalisasi piksel ditunjukkan pada Gambar 17, sedangkan hasil penerapannya dapat dilihat pada Gambar 18.

```
def normalize_image(img):
    img_min = img.min()
    img_max = img.max()
    return (img.astype(np.float32) - img_min) / (img_max - img_min) if
    img_max > img_min else img.astype(np.float32) / 255.0
```

Gambar 17. Implementasi *Code* Normalisasi Piksel



Gambar 18. Hasil Normalisasi Piksel

3. Konversi RGB ke *Greyscale* (*def rgb_to_greyscale*)

Konversi RGB digunakan untuk mengubah citra berwarna RGB menjadi citra *grayscale* (hitam-putih bertingkat). Konversi dilakukan menggunakan bobot tertentu pada setiap kanal warna (merah, hijau, biru). Kode program untuk implementasi *greyscale* ditunjukkan pada Gambar 19, sedangkan hasil penerapannya dapat dilihat pada Gambar 20.

```
def rgb_to_grayscale(img):
    return (0.289 * img[:, :, 2] + 0.587 * img[:, :, 1] + 0.114
    img[:, :, 0]).astype(np.uint8)
```

Gambar 19. Implementasi *Code* Konversi RGB ke *Greyscale*

Gambar 20. Hasil Konversi Citra RGB ke *Greyscale*

4. *Remove Noice* dengan Median *Blur* (*def remove_noise*)

Remove noise digunakan untuk mengurangi gangguan pada citra tanpa menghilangkan detail penting. Metode yang digunakan adalah median *blur*, yaitu mengganti nilai piksel pusat dengan nilai median dari piksel-piksel dalam jendela (*kernel*) sekitarnya. Kode program median *blur* ditunjukkan pada Gambar 21 sedangkan hasil penerapannya pada Gambar 22.

```
1.    def remove_noise(image):
2.        return cv2.medianBlur(image, 3)
```

Gambar 21. Implementasi *Code Remove Noise*

Gambar 22. Hasil *Remove Noise*

5. Operasi Morfologi *Opening*

Morfologi *opening* digunakan untuk menghilangkan *noise* kecil dan memperhalus bentuk objek pada citra. Metode ini dilakukan dengan menerapkan operasi erosi yang diikuti dengan dilasi menggunakan *kernel* tertentu. Kode program untuk morfologi *opening* ditunjukkan pada Gambar 23, sedangkan hasil penerapannya pada Gambar 24.

```
1.    def opening(image):
2.        kernel = np.ones((2, 2), np.uint8)
3.        return cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)
```

Gambar 23. Implementasi *Code* Morfologi *Opening*

Gambar 24. Hasil Morfologi *Opening*

6. *Otsu Thresholding* (*def otshu_threshold*)

Otsu thresholding digunakan untuk memisahkan objek dari latar belakang secara otomatis. Algoritma mencari nilai ambang (*threshold*) yang memaksimalkan perbedaan (variansi antar kelas) antara piksel objek dan piksel latar belakang. Kode program untuk implementasi *otsu thresholding* ditunjukkan pada Gambar 25, sedangkan hasil penerapannya dapat dilihat pada Gambar 26.

```
1.    def otsu_threshold(gray_img):
2.        gray_img = cv2.GaussianBlur(gray_img, (5, 5), 0)
3.        hist, bin_edges = np.histogram(gray_img.ravel(), bins=256,
4.        range=(0, 256))
5.        hist = hist.astype(np.float32) / hist.sum()
6.        bin_mids = (bin_edges[:-1] + bin_edges[1:]) / 2
7.        weight1 = np.cumsum(hist)
8.        weight2 = np.cumsum(hist[::-1])[::-1]
9.        mean1 = np.cumsum(hist * bin_mids) / (weight1 + 1e-8)
10.       mean2 = (np.cumsum((hist * bin_mids)[::-1]) /
11.       (weight2[::-1] +
12.       1e-8))[::-1]
13.       inter_class_variance = weight1[:-1] * weight2[1:] *
14.       (mean1[:-1] - mean2[1:]) ** 2
15.       threshold_manual = bin_mids[:1][np.argmax
16.       (inter_class_variance)]
17.       return np.where(gray_img >= threshold_manual, 255, 0).astype
18.       (np.uint8)
```

Gambar 25. Implementasi *Code Otsu Thresholding*

Gambar 26. Hasil *Otsu Thresholding*

Setelah tahap *pre-processing* citra, langkah berikutnya adalah melakukan ekstraksi citra menjadi teks (*image-to-text*) menggunakan metode OCR, sehingga informasi pada plat nomor dapat dibaca dan diolah lebih lanjut untuk pengecekan status uji emisi kendaraan. Proses OCR menggabungkan *Tesseract* OCR dan *EasyOCR*, lalu memvalidasi hasilnya dengan pola plat nomor umum dan memformatnya sesuai standar. Kode program untuk implementasi algoritma OCR ditunjukkan pada Gambar 27, sedangkan hasil penerapannya pada Gambar 28.

```
1.  #Import Library
2.  import pytesseract
3.  import easyocr
4.  import re
5.
6.  # Konfigurasi Tesseract untuk Windows
7.  pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-
    OCR\tesseract.exe'
8.
9.  # Inisialisasi EasyOCR
10. reader = easyocr.Reader(['en'], gpu=False)
11.
12. # Validasi pola plat nomor
13. def is_probable_plate(text):
14.     return re.match(r'^[A-Z]{1,2}[0-9]{3,4}[AZ]{1,3}$', text)
15.     is not None
16. # Fungsi pemotong jadi 8 karakter
17. def extract_plate_custom(text):
18.     text = ''.join(re.findall(r'[A-Z0-9]', text.upper()))
19.     # ambil 2 huruf awal
20.     huruf = re.findall(r'[A-Z]', text)
21.     part1 = ''.join(huruf[:2])
22.     # ambil 4 angka
23.     angka = re.findall(r'\d', text)
24.     part2 = ''.join(angka[:4])
25.     # ambil 2 huruf pertama setelah angka ke-4
26.     after_nums = re.split(r'\d{4}', text, maxsplit=1)[-1]
27.     huruf_setelah = re.findall(r'[A-Z]', after_nums)
28.     part3 = ''.join(huruf_setelah[:2])
29.     return (part1 + part2 + part3)[:8]
30.
31. # OCR
32. def combined_ocr(thresh_img):
33.     t_text = pytesseract.image_to_string(
34.         thresh_img,
35.         config='--psm 7 -c
36.         tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
37.     ).strip().replace(" ","").replace("\n","")
38.     easy = reader.readtext(thresh_img)
39.     e_text = ''.join([re.sub('[^A-Z0-9]','', r[1].upper())
40.     for r in easy]).strip()
41.     raw = e_text if len(e_text)>0 else t_text
42.     final = extract_plate_custom(raw)
43.     return final, t_text, e_text
```

Gambar 27. Implementasi *Code* Algoritma OCR

Gambar 28. Hasil OCR

Implementasi Penggunaan HTML

Hypertext Markup Language (HTML) adalah bahasa utama yang digunakan dalam pembuatan *website*. HTML terdiri dari elemen-elemen penting seperti *head* dan *body*, serta berbagai *tag* dan atribut yang berfungsi untuk mengatur dan menampilkan konten pada halaman web. Meskipun sering disebut sebagai bahasa pemrograman, HTML tidak sepenuhnya memenuhi kriteria sebagai bahasa pemrograman karena tidak memiliki kemampuan logika seperti perulangan atau pengkondisian. Fungsi utamanya adalah untuk menyusun struktur dan tata letak halaman web (Chandra Christian & Apriade Voutama, 2024).

Kode HTML terdiri dari kombinasi teks dan simbol yang disimpan dalam sebuah *file* dengan format tertentu, sesuai dengan standar internasional *American Standard Code for Information Interchange* (ASCII),

yang memungkinkan komputer untuk membaca dan menafsirkan instruksi-instruksi dalam bentuk *tag* untuk ditampilkan sebagai halaman web yang terstruktur (Agung et al., 2022).

Implementasi penggunaan HTML pada penelitian ini berfungsi sebagai tampilan antarmuka (*frontend*) yang digunakan untuk memudahkan interaksi antara pengguna dan sistem. Antarmuka ini mencakup beberapa halaman utama, yaitu halaman *dashboard* untuk menampilkan ringkasan data uji emisi, halaman data uji emisi yang terdiri dari *form* tambah data, *form* edit data, serta *form* lihat detail data, halaman deteksi untuk mengunggah gambar kendaraan dan menampilkan hasil deteksi plat nomor beserta status uji emisinya, halaman riwayat deteksi untuk melihat riwayat hasil deteksi sebelumnya, serta halaman info yang berisi informasi terkait sistem. Implementasi penggunaan HTML ditunjukkan pada gambar berikut.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initialscale=1.0">
    <title>Deteksi Uji Emisi</title>
    <link rel="stylesheet" href="{{ url_for('static',
    filename='css/Detect.css') }}">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/
    libs/font-awesome/5.15.4/css/all.min.css">
</head>
<body>
    <!--Tampilan Deteksi -->
    <div class="main">
        <h4> Uji Emisi Kendaraan Menggunakan Algoritma You Only Look
            Once V5 Large (YOLOv5L) (Tesseract OCR) Berbasis Citra
            Plat Kendaraan Bermotor </h4>
        <div class="content">
            <div class="video-wrapper">
                <video id="video" autoplay muted playsinline></video>
                <canvas id="canvas" style="display: none;"></canvas>
            </div>
```

Gambar . Implementasi HTML *dalam User Interface*

Implementasi Penggunaan Flask

Implementasi penggunaan Flask pada penelitian ini berfungsi sebagai *backend* yang mengatur logika pemrosesan data, pengelolaan rute (*route*), serta komunikasi antara antarmuka HTML dan *database*. Flask digunakan untuk menampilkan berbagai halaman seperti *dashboard*, data uji emisi, deteksi, riwayat deteksi, dan info melalui fungsi *render_template*(). Selain itu, Flask menangani proses *input* gambar kendaraan pada halaman deteksi, memanggil fungsi pendeteksian dan OCR (*detect_and_ocr*()), mengolah hasilnya, serta mencocokkannya dengan data uji emisi yang tersimpan di *database* untuk menampilkan status emisi kendaraan. Flask juga mengelola proses penambahan, pengeditan, dan penghapusan data uji emisi maupun data riwayat deteksi melalui metode HTTP seperti GET dan POST, sehingga sistem dapat berjalan secara interaktif. Implementasi penggunaan Flask ditunjukkan pada gambar berikut.

```
import os
import cv2
from flask import render_template, request, jsonify, redirect, url_for
from werkzeug.utils import secure_filename
from model import DataEmisi, DataDeteksi
from extensions import db
from datetime import datetime, timedelta
from Deteksi import detect_and_ocr
from sqlalchemy import desc
import uuid

def register_routes(app):
    @app.route('/')
    def dashboard():
        total_lulus = DataEmisi.query.filter_by(status='Lulus').
        count()
        total_tidak_lulus = DataEmisi.query.filter_by(
        status='Tidak Lulus').count()
        return render_template('Dashboard.html', lulus=total_lulus,
        tidak_lulus=total_tidak_lulus)

    @app.route('/deteksi', methods=['GET'])
    def deteksi_page():
        return render_template('Detect.html')

    @app.route('/proses_detection', methods=['POST'])
    def proses_detection():
        if 'image' not in request.files:
            return jsonify({'error': 'No image uploaded'}), 400
        file = request.files['image']
        if file.filename == '':
            return jsonify({'error': 'No image uplouded'}), 400
```

Gambar . Implementasi Penggunaan Flask

Evaluasi Sistem

Pada tahap evaluasi sistem deteksi plat, dilakukan pengujian untuk mengetahui tingkat akurasi sistem yang dibangun. Pengujian sistem dalam penelitian ini menggunakan *confusion matrix*. Berdasarkan hasil

evaluasi terhadap 60 gambar diperoleh 56 TP, 3 FP, 1 FN, dan 0 TN. Kesalahan deteksi OCR ditemukan pada 4 gambar plat nomor. Pada gambar plat nomor DT8980IE menjadi DT8986IE, terjadi kesalahan pembacaan angka 0 bagian bawah lingkarannya tertutup *noise* sehingga menyerupai angka 6. Pada gambar plat nomor DT9769UE menjadi DT9789VE, sistem salah mengenali huruf U sebagai V karena setelah proses *Otsu* lengkung bawah huruf U hilang, membuat bentuknya lebih tajam dan sempit menyerupai huruf V. Pada gambar plat nomor DT1390 menjadi IDT1390, muncul karakter tambahan I di depan yang disebabkan oleh *bounding box* terlalu lebar sehingga bingkai plat terbaca sebagai huruf. Sedangkan pada gambar plat nomor DT1036FE menjadi DT1O36E, huruf F gagal dikenali dengan benar karena tampak kurang jelas dan dipengaruhi oleh pencahayaan rendah.

Pengujian *Black Box*

Pengujian *black box* dalam penelitian ini bertujuan memastikan setiap fungsi sistem berjalan sesuai harapan, tanpa melihat struktur internal program. Fokusnya pada input dan output berdasarkan spesifikasi kebutuhan pengguna, sehingga dapat diketahui apakah sistem sudah memenuhi fungsionalitas yang diharapkan. Hasil dari pengujian ini membantu mendeteksi adanya kesalahan (*error*) pada sistem dan memastikan kualitas perangkat lunak tetap terjaga. Skenario pengujian ditunjukkan pada tabel berikut.

Tabel 2. Pengujian *Black Box*

No.	*Input*	Aksi	*Output* yang diharapkan	Hasil
1.	Navigasi ke menu *home*	*Klik* menu *home*	Sistem menampilkan halaman utama aplikasi.	
2.	Navigasi ke menu deteksi	*Klik* menu deteksi	Sistem menampilkan halaman deteksi dengan *canvas* untuk *input* citra	
3.	Navigasi ke menu data uji emisi	*Klik* menu data uji emisi	Sistem menampilkan halaman data uji emisi kendaraan	
4.	Navigasi ke menu riwayat deteksi	*Klik* menu riwayat deteksi	Sistem menampilkan halaman riwayat deteksi	
5.	Navigasi ke menu info	*Klik* menu info	Sistem menampilkan halaman info	
6.	Mengambil gambar plat	*Capture* gambar kendaraan	Sistem menampilkan hasil tangkapan gambar kendaraan	

7.	Proses deteksi plat	*Klik* tombol Deteksi	Sistem menampilkan *output* gambar *input* beserta keterangan	

SIMPULAN

Berdasarkan hasil dan pembahasan yang telah diuraikan dalam penelitian ini, disimpulkan bahwa algoritma YOLOv5L digunakan untuk mendeteksi plat nomor kendaraan secara otomatis, kemudian teks diekstraksi menggunakan *Tesseract* OCR dan *EasyOCR* sehingga status uji emisi kendaraan dapat diperiksa berdasarkan data yang ada. Algoritma tersebut berhasil diimplementasikan pada sistem dengan baik. Hasil pengujian menunjukkan bahwa model YOLOv5L berhasil mendeteksi plat nomor kendaraan dengan nilai *Precision* sebesar 91%, *Recall* sebesar 98,1%, AP sebesar 91%, serta mAP (1 kelas), mAP50, dan mAP50-95 masing-masing sebesar 91%. Sementara itu, *Tesseract* OCR dan *EasyOCR* yang diintegrasikan ke dalam sistem menghasilkan nilai *Accuracy* sebesar 93%, *Precision* sebesar 95%, *Recall* sebesar 98%, dan F1-Score sebesar 96%. Hasil ini membuktikan bahwa sistem cukup andal dalam mengenali serta membaca plat nomor kendaraan secara otomatis untuk mendukung pengecekan status uji emisi kendaraan.

Adapun beberapa saran yang perlu diperhatikan guna pengembangan selanjutnya untuk sistem ini yaitu memperbanyak variasi data plat nomor kendaraan untuk proses *training*, baik dari segi jenis plat, warna, pencahayaan, sudut pengambilan gambar, maupun kualitas citra, sehingga model YOLOv5L dapat mendeteksi area plat dengan akurasi yang lebih tinggi pada berbagai kondisi. Mengintegrasikan model OCR berbasis *deep learning*, seperti *Convolutional Recurrent Neural Network* (CRNN), yang lebih stabil terhadap variasi bentuk huruf dan angka dibandingkan OCR konvensional (*Tesseract* OCR). Menambahkan tahapan *preprocessing* citra sebelum proses OCR, seperti peningkatan kontras untuk mempertegas perbedaan karakter dengan latar belakang, *adaptive thresholding* untuk mengatasi pencahayaan yang tidak merata, dan *edge enhancement* untuk memperjelas garis tepi huruf dan angka.

DAFTAR PUSTAKA

Agung, F. N., Junaedi, I., & Yulianto, A. B. (2022). Perancangan Sistem Informasi Pelayanan Customer Dengan Platform Web. *Jurnal Manajamen Informatika Jayakarta*, *2*(4), 320. https://doi.org/10.52362/jmijayakarta.v2i4.916

Chandra Christian, & Apriade Voutama. (2024). Implementasi Aplikasi Antrian Pencucian Mobil Berbasis Web Menggunakan Php, Javascript, Html, Css Dan Uml. *JATI (Jurnal Mahasiswa Teknik Informatika)* , *8*(2), 2243–2248.

Evita, C., Alfita, R., Haryanto, H., Vivin Nahari, R., Ulum, M., & Pramudia, M. (2022). Rancang Bangun Timbangan Buah Digital Menggunakan Metode YOLO. *Jurnal FORTECH*, *3*(1), 34–42. https://doi.org/10.56795/fortech.v3i1.105

Fahmi Chairulloh Widia Sumantri, & Sutisna. (2022). Pengolahan Citra Digital Plat Nomor Kendaraan Bermotor Menggunakan K-Nn. *Jurnal Informatika Teknologi Dan Sains*, *4*(2), 101–104. https://doi.org/10.51401/jinteks.v4i2.1999

Heryanti, Tatawu, G., Sensu, L., Haris, O. K., Sinapoy, M. S., & Ihlas, M. (2024). Analisis Hukum Kewenangan Pemerintah Kota Kendari dalam Pengujian Emisi Gas Buang Kendaraan Bermotor Terkait Perlindungan terhadap Pencemaran Udara. *Halu Oleo Legal Research*, *6*(1), 217–228. https://jial-apha.net/index.php/adat/article/view/pengelolaan-hutan-

Ishma Safira, S. W. A. A. W. (2023). Efektivitas Peraturan Gubernur Jakarta Tentang Uji Emisi Terhadap Pencemaran Udara Di Dki Jakarta. *Triwikrama: Jurnal Ilmu Sosial*, *01*(08), 14. https://jakarta.bps.go.id/indicator/17/786/1/jumlah-kendaraan-bermotor-menurut-jenis-kendaraan-unit-di-

Jia, C. (2023). Application Effect Analysis for Helmet Detection Algorithm based on YOLOV5. *Highlights in Science, Engineering and Technology*, *39*, 1214–1220. https://doi.org/10.54097/hset.v39i.6731

Kharisma, O. B. (2021). Sistem Identifikasi Plat Nomor Kendaraan Dalam Penerapan Regulasi Pajak Berbasis Citra Digital. *JST (Jurnal Sains Dan Teknologi)*, *10*(1), 117–127. https://doi.org/10.23887/jstundiksha.v10i1.32975

Michelle, E., Jusuf, M., & Julian, J. (2021). Efektivitas Pelaksanaan Kebijakan Berdasarkan Pergub No 66 Tahun 2020 Tentang Uji Emisi Kendaraan Bermotor Di Jakarta. *ADIL: Jurnal Hukum*, *12*(1). https://doi.org/10.33476/ajl.v12i1.1920

Muzaky, A., Arifianto, F., Hendrowati, R., & Darwis, M. (2024). Menerapkan Metode Klasifikasi pada Data Uji Emisi Kendaraan di Jakarta dengan Menggunakan Jupyter Notebook. *Jurnal Pengembangan Sistem Informasi Dan Informatika*, *5*(2), 74–84. https://doi.org/10.47747/jpsii.v5i2.1722

Naftali, M. G., Sulistyawan, J. S., & Julian, K. (2022). *Comparison of Object Detection Algorithms for Street-level Objects*. http://arxiv.org/abs/2208.11315

Reezky Illmawati, & Hustinawati. (2023). YOLO V5 for Vehicle Plate Detection in DKI Jakarta. *Jurnal Ilmu Komputer Dan Agri-Informatika*, *10*(1), 32–43. https://doi.org/10.29244/jika.10.1.32-43

Saputro, H. I., Martanto, E. A., & Yuminarti, U. (2022). Analisis emisi gas buang kendaraan bermotor (angkutan umum penumpang) di Kabupaten Manokwari. *Cassowary*, *5*(1), 35–47. https://doi.org/10.30862/casssowary.cs.v5.i1.100

Suryotomo, A. P., Akbar, B. M., & Husaini, R. (2024). Performance Analysis of FastAPI Framework on Lost Circulation Handling Management Application in Oil Well Drilling. *Telematika*, *21*(1), 110. https://doi.org/10.31315/telematika.v21i1.13259

Umriana, A. E., Insanial, M., Pribadi, M., Pada, M., & Pandemi, M. (2020). *ANALISA POLA PERILAKU PENGGUNA MOBIL PRIBADI DI KOTA MAKASSAR PADA MASA PENDEMI COVID-19 Adinda Erwita Umriana 31217031 ; Muhammad Insanial 31217033 , Analisa Pola Perilaku Pengguna Mobil Pribadi di Kota Makassar Pada Masa Pandemi Covid-19*.