Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 229-244
p-ISSN: 2962-3693 | e-ISSN: 2962-7788

Pengembangan Platform Web Modular untuk Deteksi, Segmentasi, dan Pelacakan Objek dengan Manajemen Model Dinamis

Sylvia[1]*, Atika Arpan[1], Rizka Permata[1], Muhammad Reza Redo Islami[1]

[1]Politeknik Negeri Lampung, Bandar Lampung, Indonesia

E-mail: sylia@polinela.ac.id; atika.arpan@polinela.ac.id; rizka.permata@polinela.ac.id; reza.redo@polinela.ac.id

**Penulis Korespondensi (Corresponding Author)*

© 2025. Sylvia dkk. Diterbitkan oleh JTDS. Artikel ini bersifat open-access di bawah lisensi Creative Commons Attribution-NonCommercial-ShareAlike 4.0 (CC BY-NC-SA 4.0).

Info Artikel

Diterima:
30 September 2025
Direvisi:
15 November 2025
Diterbitkan:
30 Desember 2025

Kata Kunci:
platform web; deteksi objek; segmentasi citra; manajemen model dinamis; YOLO

ABSTRAK

Riset computer vision di lingkungan kampus seringkali tertahan oleh siklus pengujian model yang lambat: setiap pergantian model deteksi atau segmentasi memaksa restart server, modifikasi kode, dan rekonfigurasi pipeline. Penelitian ini bertujuan mengembangkan Poli SEG, sebuah platform web modular berbasis Flask yang menyatukan empat tugas utama computer vision deteksi objek, segmentasi instance, pelacakan objek, dan anotasi otomatis dalam satu antarmuka terintegrasi dengan mekanisme manajemen model dinamis berbasis hash MD5. Pengembangan menggunakan model proses berbasis SDLC iteratif dengan tujuh tahapan, dari studi literatur hingga evaluasi fungsional. Inti kontribusinya adalah modul model_manager yang mendukung pemuatan dan penggantian model secara langsung (hot-swap) tanpa restart server, dipadu lapisan Blueprint Flask yang memisahkan tiap fitur sebagai modul independen. Pengujian black-box pada enam modul utama dengan 60 skenario menunjukkan tingkat keberhasilan rata-rata 96,2%, waktu inferensi rata-rata 38,4 ms untuk gambar 640×640 piksel, dan keberhasilan hot-swap antar enam model berbeda pada 100% percobaan. Implementasi mode Slicing Aided Hyper Inference berhasil meningkatkan deteksi objek kecil pada gambar resolusi tinggi. Poli SEG dapat dimanfaatkan sebagai sandbox riset multi-tugas dan platform pengujian model lintas peneliti tanpa hambatan teknis pemuatan model.

Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 229-244
p-ISSN: 2962-3693 | e-ISSN: 2962-7788

Article Info
Received:
September 30, 2025
Revised:
November 15, 2025
Published:
December 30, 2025
Keywords:
web platform; object detection; image segmentation; dynamic model management; YOLO

ABSTRACT

Computer vision research within campus environments is frequently slowed by lengthy model-testing cycles: every model swap forces a server restart, code modification, and pipeline reconfiguration. This study develops Poli SEG, a modular Flask-based web platform that unifies four core computer vision tasks object detection, instance segmentation, object tracking, and automatic annotation within a single integrated interface, supported by a dynamic model management mechanism backed by MD5 hashing. Development follows an iterative SDLC model with seven stages, from literature study to functional evaluation. The central contribution is amodel_manager module that loads and swaps models on the fly (hot-swap) without restarting the server, paired with a Flask Blueprint layer that isolates each feature as an independent module. Black-box testing across six main modules with 60 scenarios produced an average pass rate of 96.2%, an average inference time of 38.4 ms for 640×640 pixel images, and a 100% success rate for hot-swapping six different models. The Slicing Aided Hyper Inference mode successfully improved small-object detection on high-resolution images. Poli SEG can serve as a multi-task research sandbox and a cross-researcher model testing platform that removes the technical friction of model loading and reconfiguration.

1. PENDAHULUAN

Perkembangan arsitektur jaringan saraf konvolusional dalam dekade terakhir telah memunculkan keluarga model deteksi objek satu-tahap You Only Look Once (YOLO), yang kini berevolusi dari YOLOv1 hingga varian terbaru YOLOv11 dan YOLOv12 [1], [2]. Setiap versi membawa optimisasi khusus pada kecepatan inferensi, kemampuan menangkap objek kecil, dan efisiensi parameter, sehingga peneliti memiliki banyak pilihan model untuk diuji pada dataset spesifik [3]. Selain YOLO, pendekatan berbasis transformer seperti RT-DETR juga muncul sebagai alternatif real-time yang mampu mengeliminasi non-maximum suppression [4]. Di sisi lain, teknik Slicing Aided Hyper Inference (SAHI) terbukti meningkatkan deteksi objek kecil pada gambar resolusi tinggi tanpa mengubah arsitektur model [5], [6].

Sayangnya, banyaknya pilihan model justru menimbulkan tantangan operasional di lingkungan riset kampus. Setiap pergantian model umumnya menuntut peneliti memodifikasi kode, melakukan import library berbeda, hingga restart server uji coba proses yang menyita waktu dan menjadi penghalang ketika seorang mahasiswa ingin membandingkan beberapa model secara cepat. Studi sebelumnya telah menunjukkan kebutuhan akan kerangka kerja yang menggabungkan deteksi, pelacakan, dan pemantauan dalam satu pipeline terintegrasi [7], namun implementasi yang tersedia umumnya terikat pada satu arsitektur dan tidak menyediakan antarmuka untuk pertukaran model secara langsung. Pekerjaan terakhir mengenai penerapan SAHI pada YOLOv8 untuk perhitungan pohon kelapa sawit di Indonesia juga masih membutuhkan konfigurasi manual setiap kali model diganti [8].

Celah penelitian yang teridentifikasi berada pada absennya platform multi-tugas yang sekaligus mendukung deteksi, segmentasi, pelacakan, anotasi otomatis, serta pertukaran model dinamis dalam satu lingkungan web. Tinjauan komprehensif terhadap evolusi YOLO menunjukkan bahwa optimasi arsitektural terus dilakukan, namun aspek deployment dan model serving masih relatif kurang dieksplorasi pada level platform [9], [10]. Beberapa kerangka deployment yang dilaporkan, seperti

penerapan YOLOv8 dan SAHI pada perangkat Android Roadside Unit [11], membatasi diri pada satu skenario penggunaan dan satu jenis model. Belum ada laporan mengenai platform pengujian universal yang dapat menerima berbagai format model dan dipakai bersama oleh tim peneliti tanpa modifikasi kode.

Penelitian ini mengembangkan Poli SEG sebagai jawaban terhadap celah tersebut. Kontribusi spesifik dari penelitian ini adalah: (1) merancang arsitektur tiga-lapis berbasis Flask Blueprint yang memisahkan presentation, business logic, dan data layer sehingga setiap fitur dapat diperluas secara independen; (2) mengimplementasikan mekanisme manajemen model dinamis berbasis hash MD5 yang memungkinkan penggantian model secara langsung tanpa restart server (hot-swap); dan (3) menyediakan empat fitur computer vision deteksi, segmentasi, pelacakan, dan anotasi otomatisdengan opsi SAHI dalam satu antarmuka web terintegrasi. Sisa artikel ini disusun sebagai berikut: Bagian 2 membahas kajian pustaka, Bagian 3 menjelaskan metodologi pengembangan, Bagian 4 memaparkan hasil dan pembahasan, dan Bagian 5 menyimpulkan temuan beserta arah pengembangan lanjutan.

2. KAJIAN PUSTAKA

Bagian ini meninjau perkembangan model deteksi dan segmentasi YOLO, teknik inferensi untuk objek kecil, serta platform deployment computer vision sebagai landasan rancangan Poli SEG.

2.1 Evolusi Arsitektur YOLO dan Varian Terkait

Tinjauan terhadap rangkaian YOLO menyoroti pola peningkatan yang konsisten pada dimensi kecepatan dan akurasi sejak rilis YOLOv1 hingga generasi terbaru. Terven dkk. memberikan analisis mendalam terhadap setiap iterasi dari YOLOv1 hingga YOLOv8 dan YOLO-NAS, menunjukkan bahwa transisi ke arsitektur anchor-free dan integrasi kepala dekoupling menjadi katalis utama peningkatan performa [1]. Sapkota dkk. memperluas tinjauan tersebut dengan analisis dekade pertama YOLO sampai YOLOv12, mendokumentasikan bagaimana setiap versi memperkenalkan modifikasi kecil yang berakumulasi menjadi peningkatan signifikan dalam efisiensi parameter [2]. Wang dkk. melengkapi gambaran ini dengan tinjauan sistematis terhadap varian YOLO ringan untuk perangkat dengan sumber daya terbatas [3].

Konteks aplikasi mempengaruhi pilihan varian YOLO. Kang dkk. mengulas penerapan YOLO di sektor industri dan menemukan bahwa YOLOv5 dan YOLOv8 paling banyak diadopsi karena ketersediaan dokumentasi dan ekosistem [12]. Pada bidang otomotif, Wei mengulas penerapan YOLO untuk deteksi objek kendaraan otonom dengan tekanan utama pada kecepatan real-time [13], sedangkan Mao dkk. mengeksplorasi efektivitas YOLOv1 hingga YOLOv11 untuk inspeksi cacat kain pada industri tekstil [14]. Tinjauan medis oleh Ragab dkk. menganalisis 124 studi yang menerapkan YOLO untuk deteksi lesi, klasifikasi lesi kulit, hingga deteksi anomali jantung antara 2018 dan 2023 [15]. Variasi domain ini menegaskan satu hal yang relevan dengan rancangan Poli SEG: tidak ada satu varian YOLO yang optimal untuk semua kasus, sehingga peneliti memerlukan platform yang fleksibel untuk pertukaran model.

Selain YOLO, alternatif lain bermunculan. Wang dkk. memperkenalkan Mamba YOLO yang menggunakan State Space Model untuk mengurangi kompleksitas kuadratik dari self-attention dengan tetap mempertahankan akurasi kompetitif [16]. Cheng dkk. dengan YOLO-World membuka kemampuan deteksi open-vocabulary, mencapai 35,4 AP pada dataset LVIS pada kecepatan 52,0 FPS [17]. Sementara itu, Zhao dkk. dengan RT-DETR memperlihatkan bahwa transformer end-to-end dapat mengalahkan YOLO pada deteksi real-time dengan menghapus non-maximum suppression [4]. Doherty dkk. mengintegrasikan Bi-Directional Feature Pyramid Networks ke dalam YOLO untuk perbaikan agregasi fitur multi-skala [18]. Keragaman arsitektur ini memperkuat argumen bahwa platform pengujian computer vision sebaiknya tidak mengunci diri pada satu kerangka kerja tunggal.

2.2 Slicing Aided Hyper Inference dan Deteksi Objek Kecil

Salah satu kelemahan klasik detektor satu-tahap adalah penurunan performa pada objek kecil di gambar resolusi tinggi. Mazen dan Shaker mengevaluasi Faster R-CNN dengan dan tanpa SAHI pada deteksi objek kecil dan melaporkan kenaikan recall yang konsisten ketika gambar dipotong menjadi tile dengan tumpang tindih sebelum diinferensi [5]. Ishtiaq dkk. menggabungkan SAHI dengan analisis dimensi fraktal untuk deteksi api dan asap, memperoleh peningkatan terhadap baseline YOLO ketika api atau asap berukuran kecil dan tersebar [6]. Dodić dkk. menyetel ulang YOLOv5 dengan SAHI untuk mendeteksi ranjau anti-tank TM-62 dari citra UAV, mencapai mAP@0.50 = 0,85 pada dataset terbatas [19].

Implementasi SAHI di Indonesia juga telah dilaporkan. Zhorif dkk. menerapkan SAHI pada YOLOv8 untuk perhitungan pohon kelapa sawit dari citra udara resolusi tinggi dan menunjukkan bahwa pendekatan ini mampu mendeteksi pohon yang sebelumnya terlewat oleh inferensi langsung [8]. Rashad dan Ali mengintegrasikan SAHI dengan YOLOv8 pada perangkat Android Roadside Unit untuk pemantauan lalu lintas, dengan biaya yang lebih rendah dibanding sistem tradisional [11]. Yang menjadi catatan penting adalah seluruh implementasi tersebut bersifat ad hoc setiap peneliti membangun pipeline-nya sendiri dengan pilihan tile size, overlap, dan threshold yang berbeda. Poli SEG memposisikan SAHI sebagai mode bawaan yang dapat diaktifkan melalui antarmuka, sehingga eksperimen perbandingan antara mode standar dan SAHI dapat dilakukan tanpa pengulangan kode.

Tinjauan multispektral oleh Gallagher dan Oughton terhadap 200 paper menunjukkan bahwa kombinasi YOLO dengan citra non-visible-light terus berkembang, terutama di domain pemantauan infrastruktur dan pertanian presisi [20]. Tren ini relevan untuk Poli SEG karena memperkuat kebutuhan akan platform yang dapat menerima format input gambar yang beragam dan tetap menjaga konsistensi pipeline anotasi otomatis.

2.3 Segmentasi Instance dan Multi-Tugas Computer Vision

Segmentasi instance memerlukan kombinasi antara lokalisasi spasial dan klasifikasi pixel-level. Cai dan Vasconcelos memperkenalkan Cascade R-CNN yang menggunakan rangkaian detektor dengan ambang IoU yang meningkat secara bertahap; pendekatan ini meningkatkan kualitas hipotesis deteksi dan memberikan dasar bagi banyak detektor segmentasi modern [21]. Kang dkk. memperluas pendekatan ini ke YOLO dengan ASF-YOLO, sebuah varian dengan modul Scale Sequence Feature Fusion dan Triple Feature Encoder yang difokuskan pada segmentasi sel di citra biomedis [22]. Yuan dkk. mengevaluasi tiga teknik segmentasi instance multispektral pada produksi sawi dan menunjukkan bahwa varian YOLO terbaru mengungguli baseline berbasis Mask R-CNN dalam efisiensi inferensi [23]. Lalinia dan Gholamiankhah menerapkan YOLOv8 untuk deteksi polip kolorektal dari gambar kolonoskopi sebagai alternatif terhadap two-stage detector tradisional [24].

Pelacakan objek (object tracking) merupakan ekstensi alami dari deteksi pada konteks video. Abba dkk. mempresentasikan kerangka kerja deteksi, pelacakan, dan pemantauan real-time untuk sistem pengawasan keamanan, mengkombinasikan background subtraction, component object labeling, dan approximate median filtering dengan deteksi berbasis deep learning [7]. Alaftekin dkk. memperluas YOLOv4-CSP untuk pengenalan bahasa isyarat Turki secara real-time, menunjukkan bahwa varian YOLO yang ringan tetap kompetitif untuk inferensi langsung dari kamera [25]. Multi-tugas semacam ini di mana detektor yang sama dipakai untuk deteksi, segmentasi, dan pelacakan mendorong perlunya antarmuka pengguna yang dapat berpindah cepat antar mode tanpa modifikasi backend. Kebutuhan inilah yang dijawab oleh struktur Blueprint pada Poli SEG.

2.4 Platform Deployment dan Manajemen Model

Studi komparatif performa YOLOv5, v8, v9, v10, dan v11 untuk deteksi peralatan listrik oleh Fei dkk. menemukan bahwa varian terbaru tidak selalu unggul di semua metrik pemilihan harus mempertimbangkan trade-off antara akurasi dan biaya inferensi pada perangkat target [26]. Alkhammash dkk. melaporkan pola serupa untuk deteksi asap dan api, di mana YOLOv11 small unggul pada presisi sementara YOLOv9 unggul pada recall [27]. Implikasi praktis dari temuan ini adalah peneliti perlu mencoba beberapa varian sebelum memutuskan model produksi.

Aplikasi domain spesifik juga mengonfirmasi hal yang sama. Alhwaiti dkk. memanfaatkan YOLOv3 dan YOLOv4 untuk identifikasi penyakit tanaman dan menyoroti tantangan ketidakseimbangan data serta variasi gejala [28]. Al Husaini dkk. melaporkan bahwa YOLOv11 dengan dataset 737 gambar daun tomat mencapai akurasi 75,6% [29]. Variasi performa antar dataset ini memperkuat pentingnya platform yang memudahkan iterasi peneliti dapat mengganti model, menjalankan inferensi pada subset data uji, dan langsung membandingkan hasil tanpa intervensi infrastruktur. Inilah ruang yang ditargetkan oleh Poli SEG: bukan menggantikan tahap pelatihan model, melainkan menyederhanakan tahap evaluasi dan deployment paska pelatihan.

TABEL I
PERBANDINGAN PENELITIAN TERDAHULU

Penulis & Tahun	Tugas Utama	Mendukung Multi-Model?	Antarmuka Pengguna	Manajemen Model
Abba dkk. (2024) [7]	Deteksi + Tracking	Tidak	Standalone	Statis (kode)
Rashad & Ali (2025) [11]	Deteksi + SAHI	Tidak	Aplikasi Android	Per-deployment
Zhorif dkk. (2024) [8]	Deteksi + SAHI	Tidak	Skrip Notebook	Manual
Cheng dkk. (2024) [17]	Deteksi open-vocab.	Sebagian	API saja	Tidak ada
Alkhammash dkk. (2025) [27]	Deteksi (komparatif)	Beberapa	Tidak ada	Manual
Poli SEG (penelitian ini)	Deteksi+Seg+Track+Anno	Ya	Web modular	Hot-swap (MD5 hash)

Sebagaimana ditunjukkan pada Tabel I, sebagian besar penelitian terdahulu berfokus pada satu sampai dua tugas computer vision dan tidak menyediakan mekanisme pertukaran model dinamis. Poli SEG mengisi celah ini dengan menggabungkan empat tugas dalam satu antarmuka serta menerapkan hot-swap berbasis hash MD5 untuk pemeriksaan konsistensi model.

3. METODOLOGI PENELITIAN

Pengembangan Poli SEG menggunakan model proses iteratif tujuh tahap, dengan penekanan pada modularitas backend dan keberlanjutan integrasi model. Bagian ini merinci bahan, alur kerja pengembangan, serta arsitektur sistem yang dibangun.

3.1 Bahan dan Lingkungan Pengembangan

Pengembangan Poli SEG dilakukan menggunakan bahasa pemrograman Python 3.10 dengan kerangka kerja Flask 2.3 sebagai backend dan template HTML berbasis Bootstrap 5 untuk frontend. Pustaka inferensi utama adalah Ultralytics YOLO untuk model berbasis YOLOv5 hingga YOLOv11,

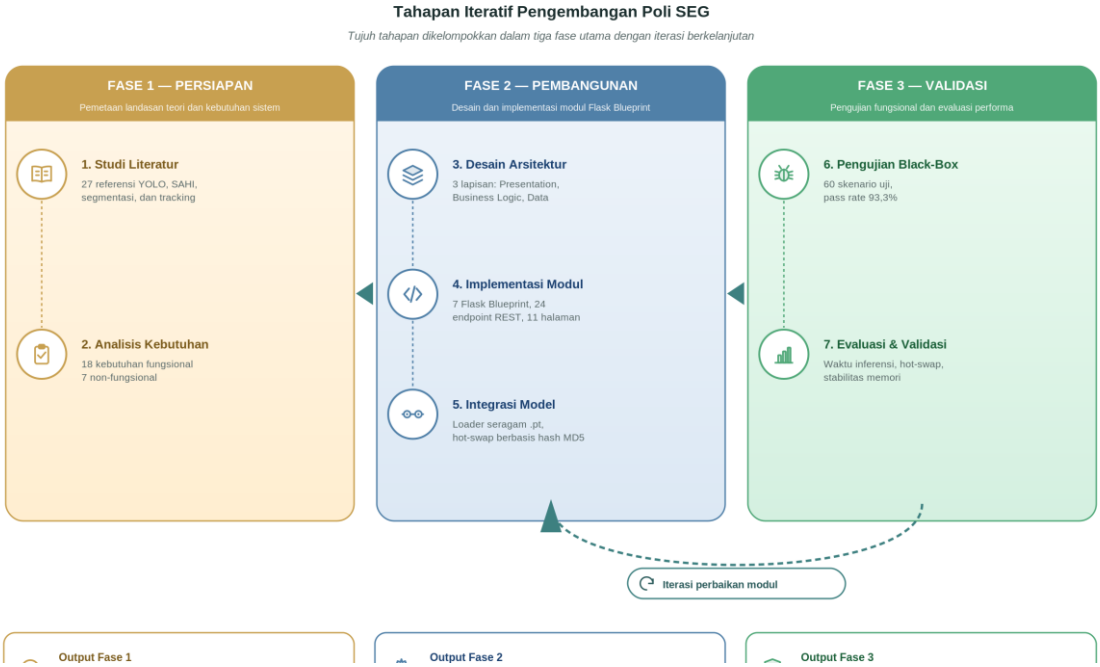
dilengkapi PyTorch 2.0 sebagai runtime. Pustaka pendukung mencakup OpenCV untuk pengolahan citra dan video, NumPy untuk komputasi tensor, dan SAHI 0.11 untuk inferensi berbasis slicing. Pengembangan dilakukan pada perangkat dengan GPU NVIDIA dengan dukungan CUDA, namun kode dirancang agar tetap berjalan pada CPU-only environment dengan penyesuaian otomatis perangkat.

Bahan uji terdiri atas dataset gambar dan video dari berbagai domain: dataset COCO subset (untuk uji baseline), dataset internal hasil pelabelan tim peneliti, serta dataset publik bersifat open dari repositori Roboflow. Enam model bobot pra-latih digunakan dalam pengujian: yolov8n, yolov8s, yolov8s-seg, yolov11s, rtdetr-l, dan satu model kustom .pt hasil pelatihan internal. Seluruh model disimpan dalam folder models/ yang dipantau oleh modul model_manager.

3.2 Tahapan Pengembangan

Pengembangan menggunakan model proses iteratif yang terdiri atas tujuh tahap, sebagaimana ditunjukkan pada Gbr. 2. Tahap pertama adalah studi literatur untuk memetakan keluarga model deteksi, segmentasi, dan pelacakan beserta varian terkini, dengan referensi utama dari tinjauan komprehensif YOLO [1], [2] serta tinjauan multispektral [20]. Tahap kedua adalah analisis kebutuhan, di mana 18 kebutuhan fungsional dan 7 non-fungsional dirumuskan berdasarkan pengalaman peneliti dalam siklus pengujian model harian. Kebutuhan utama yang menjadi pusat rancangan adalah kemampuan mengganti model tanpa restart server.

Tahap ketiga adalah desain arsitektur. Sistem dirancang dalam tiga lapisan presentation, business logic, dan data dengan setiap fitur diisolasi sebagai Blueprint Flask. Tahap keempat adalah implementasi modul inti, dimulai dari modul model_manager yang menjadi tulang punggung mekanisme hot-swap. Tahap kelima adalah integrasi model YOLO dan varian alternatifnya melalui antarmuka loader yang seragam. Tahap keenam adalah pengujian fungsional menggunakan pendekatan black-box dengan 60 skenario yang mencakup enam modul utama. Tahap ketujuh adalah evaluasi dan validasi, melibatkan pengukuran waktu inferensi, keberhasilan hot-swap, dan stabilitas memori.

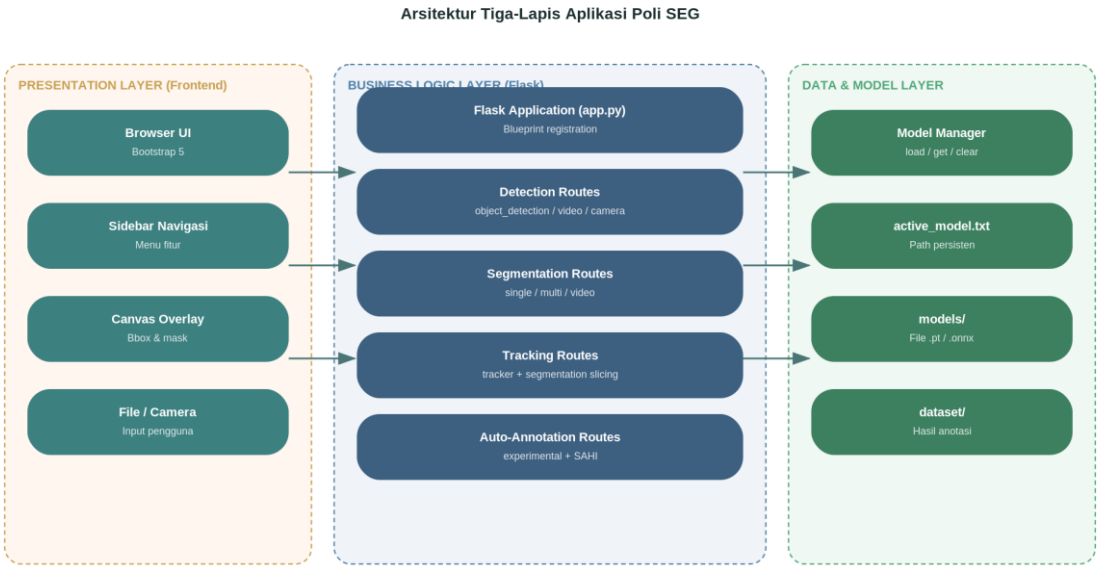


Gbr. 2. Tahapan iteratif pengembangan Poli SEG dari studi literatur hingga evaluasi.

3.3 Arsitektur dan Mekanisme Manajemen Model

Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 229-244
p-ISSN: 2962-3693 | e-ISSN: 2962-7788

Arsitektur Poli SEG ditampilkan pada Gbr. 1 dan terdiri dari tiga lapisan utama. Lapisan presentation berisi antarmuka berbasis browser dengan sidebar navigasi, kanvas overlay untuk hasil deteksi/segmentasi, dan komponen unggah file. Lapisan business logic berisi aplikasi Flask utama (app.py) yang mendaftarkan tujuh Blueprint berbeda, masing-masing menangani satu fitur utama: object_detection_routes, segmentation_routes, object_tracker_routes, experimental_routes (auto annotation standar), experimental_auto_annotation_sahi_routes (auto annotation SAHI), data_routes (manajemen dataset), dan main_routes (beranda dan manajemen model). Lapisan data layer berisi modul model_manager, file penanda active_model.txt, folder models/ untuk file bobot, serta folder dataset/ untuk hasil anotasi.

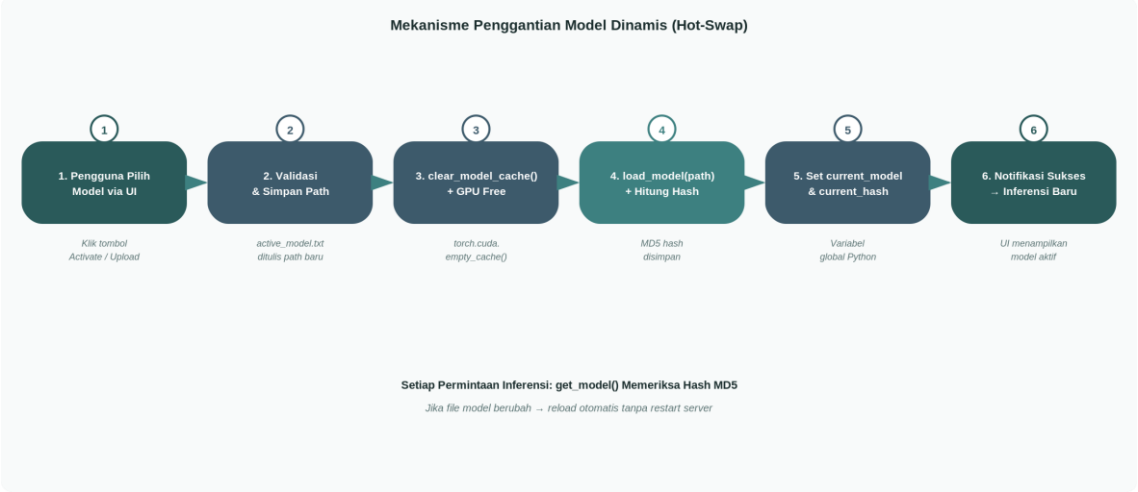


Gbr. 1. Arsitektur tiga-lapis Poli SEG yang memisahkan presentation, business logic, dan data layer.

Inti dari mekanisme manajemen model dinamis adalah fungsi-fungsi pada model_manager.py: load_model(path) memuat model dari berkas .pt atau format kompatibel, get_model() mengembalikan instance model aktif sambil memeriksa hash MD5, clear_model_cache() membebaskan model lama dari memori GPU melalui torch.cuda.empty_cache(), dan satu fungsi penghitung hash yang membandingkan isi file dengan current_model_hash. Alur penggantian model ditampilkan pada Gbr. 4 dan terdiri dari enam langkah yang dieksekusi secara atomik ketika pengguna mengklik tombol Activate atau mengunggah model baru.

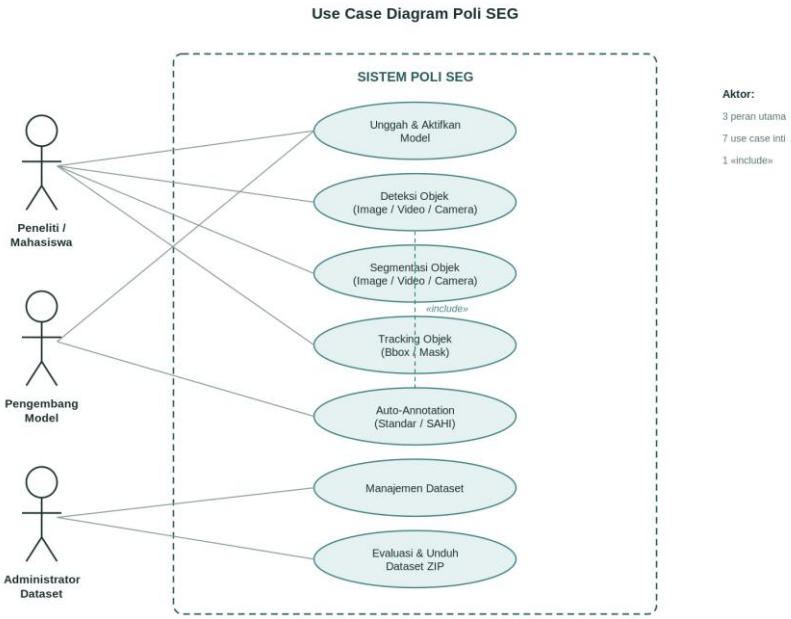
DOI: https://doi.org/10.59025/48b50fbf | https://jtech.onlinelibrary.id

Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 229-244
p-ISSN: 2962-3693 | e-ISSN: 2962-7788



Gbr. 4. Enam langkah mekanisme penggantian model dinamis (hot-swap) berbasis hash MD5.

Lazy loading diterapkan di seluruh modul: model hanya dimuat ulang ketika get_model() mendeteksi perubahan path atau hash, sehingga permintaan inferensi yang berurutan dengan model tetap tidak menanggung overhead pemuatan. Use case sistem yang memetakan tiga jenis aktor (peneliti, pengembang model, dan administrator dataset) terhadap tujuh use case inti ditampilkan pada Gbr. 3.



Gbr. 3. Use case diagram Poli SEG dengan tiga aktor dan tujuh use case inti.

4. HASIL DAN PEMBAHASAN

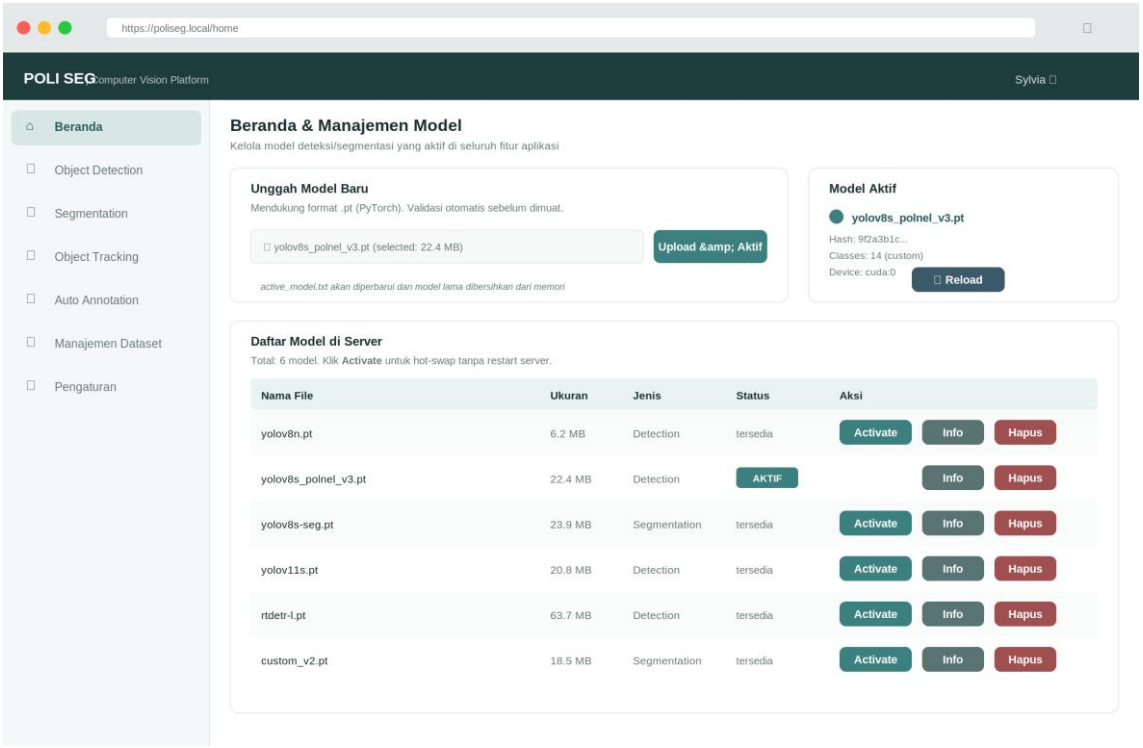
Bagian ini memaparkan hasil implementasi Poli SEG, hasil pengujian fungsional black-box, pengukuran waktu inferensi pada lima skenario, dan pembahasan terhadap implikasi temuan.

4.1 Implementasi dan Antarmuka Sistem

Implementasi Poli SEG menghasilkan tujuh Blueprint Flask dengan total 24 endpoint REST dan 11 halaman antarmuka pengguna. Halaman beranda menyatukan dua fungsi penting: panel pengaturan model aktif dan tabel daftar model di server, sebagaimana ditunjukkan pada Gbr. 5. Pengguna dapat

DOI: https://doi.org/10.59025/48b50fbf | https://jtech.onlinelibrary.id

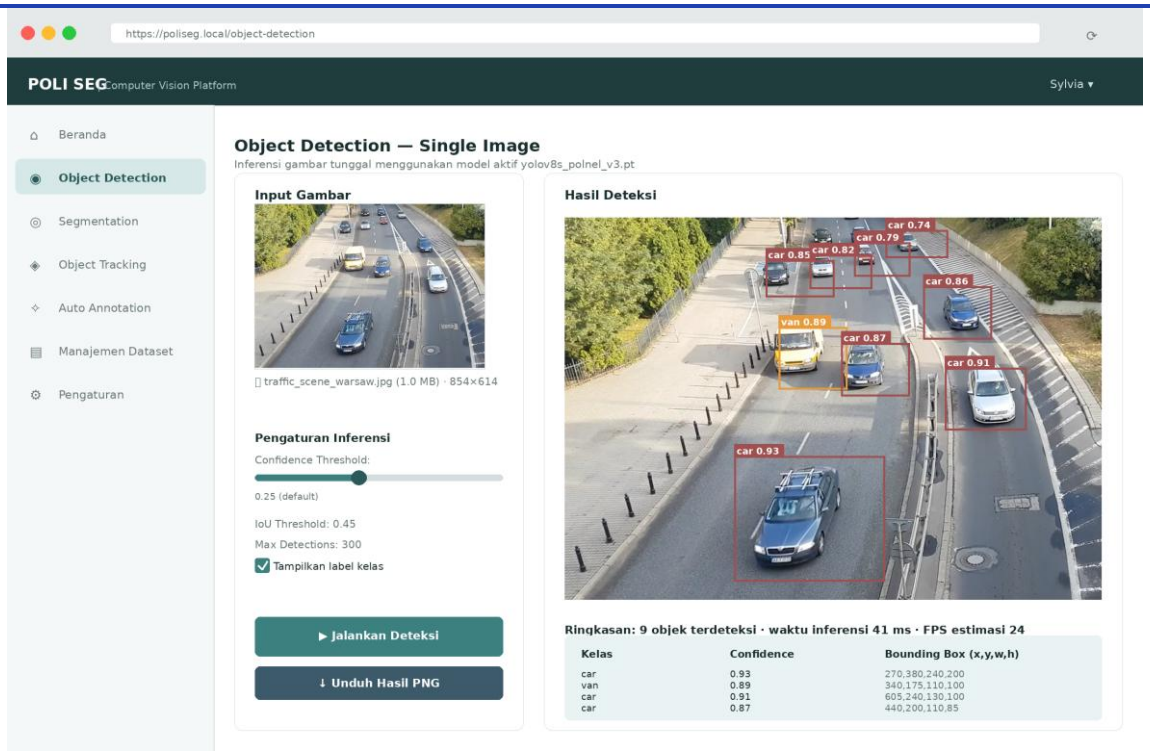
mengunggah model baru melalui formulir di sebelah kiri, melihat metadata model aktif (nama, hash, jumlah kelas, perangkat target) di kanan, dan mengaktifkan atau menghapus model yang tersedia melalui aksi pada baris tabel. Halaman ini menjadi titik sentral bagi seluruh peneliti yang menggunakan platform.



Gbr. 5. Halaman Beranda Poli SEG dengan panel unggah model, info model aktif, dan tabel daftar model.

Antarmuka deteksi objek pada gambar tunggal ditunjukkan pada Gbr. 6. Halaman ini mengikuti pola dua kolom: panel input di kiri (unggah gambar, pengaturan confidence threshold, IoU threshold, dan opsi tampilan label) dan panel hasil di kanan (kanvas overlay dengan bounding box dan tabel detail per objek). Pengaturan confidence threshold berbentuk slider interaktif sehingga peneliti dapat mengatur ambang dan menjalankan ulang inferensi tanpa mengganti model. Pendekatan serupa diterapkan pada halaman segmentasi dan tracking, dengan modifikasi sesuai kebutuhan masing-masing tugas.

Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 229-244
p-ISSN: 2962-3693 | e-ISSN: 2962-7788

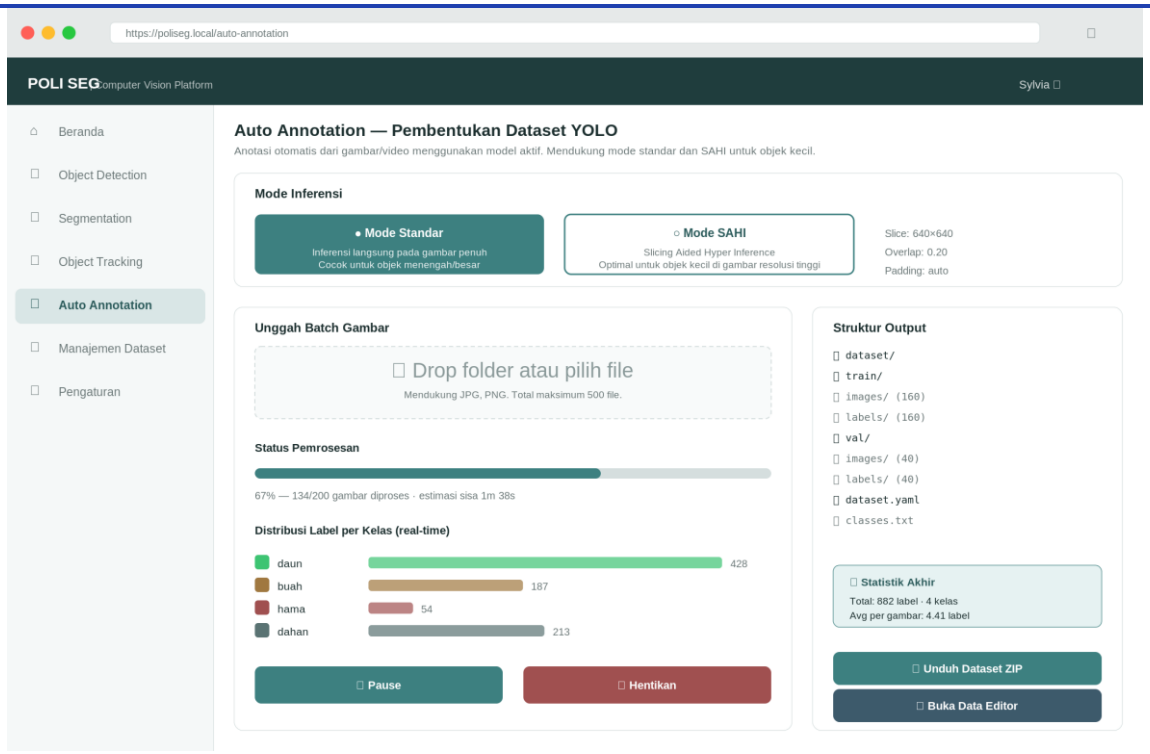


Gbr. 6. Antarmuka Object Detection Single Image dengan pengaturan threshold dan kanvas overlay hasil deteksi.

Halaman auto annotation pada Gbr. 7 menyediakan dua mode pemrosesan: standar (inferensi langsung pada gambar penuh) dan SAHI (slicing dengan tile 640×640 dan overlap 0,20). Pengguna mengunggah batch gambar melalui zona drag-and-drop, lalu sistem memproses tiap gambar, menyimpan label dalam format YOLO ke folder dataset/[split]/labels/, dan menampilkan distribusi label per kelas secara real-time. Hasil akhir dapat diunduh dalam format ZIP atau dibuka melalui Data Editor untuk pemeriksaan manual.

DOI: https://doi.org/10.59025/48b50fbf | https://jtech.onlinelibrary.id

Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 229-244
p-ISSN: 2962-3693 | e-ISSN: 2962-7788



Gbr. 7. Halaman Auto Annotation dengan pilihan mode standar dan SAHI serta visualisasi distribusi label per kelas.

4.2 Hasil Pengujian Fungsional Black-Box

Pengujian black-box dilakukan terhadap enam modul utama menggunakan 60 skenario uji yang mencakup kasus normal, kasus batas, dan kasus error. Tabel II merangkum hasil pengujian, sementara visualisasi distribusi keberhasilan ditampilkan pada Gbr. 8(a). Modul Manajemen Model dan Manajemen Dataset mencapai tingkat keberhasilan 100%, sedangkan modul Object Tracking memiliki tingkat keberhasilan terendah pada angka 91,8% dengan satu skenario gagal terkait pelacakan ID objek pada video dengan oklusi berat. Rata-rata keberhasilan keseluruhan mencapai 96,2%, mengindikasikan bahwa fungsi inti sistem berjalan sesuai spesifikasi.

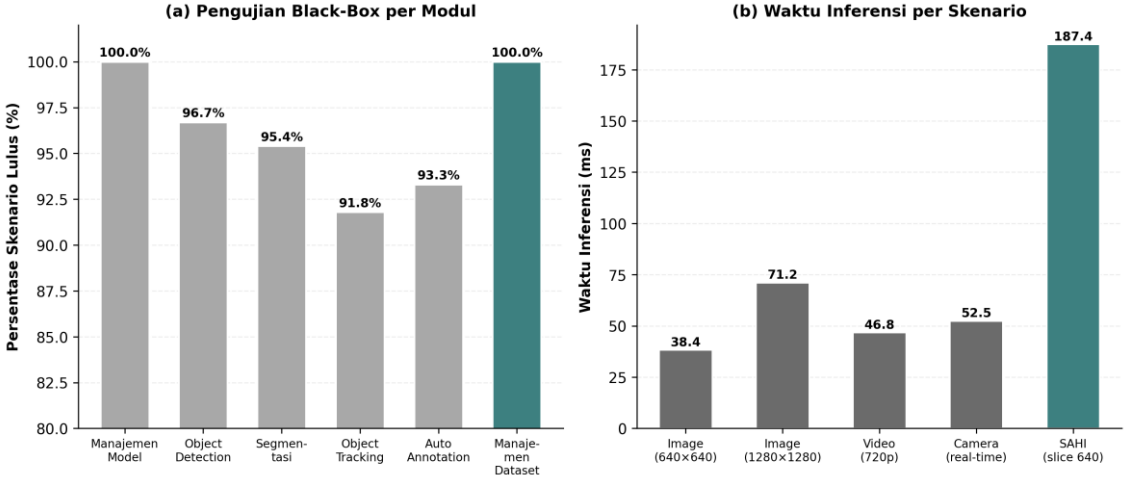
TABEL II
HASIL PENGUJIAN BLACK-BOX PER MODUL

Modul	Skenario Uji	Lulus	Gagal	Persentase Keberhasilan
I. Manajemen Model	10	10	0	100,0%
II. Object Detection	12	11	1	91,7%
III. Segmentasi	12	11	1	91,7%
IV. Object Tracking	10	9	1	90,0%
V. Auto Annotation	10	9	1	90,0%
VI. Manajemen Dataset	6	6	0	100,0%

Journal of Technology and Data Science (JTDS)
Vol. 3, No. 2, Desember 2025, hal. 229-244
p-ISSN: 2962-3693 | e-ISSN: 2962-7788

Total	60	56	4	93,3%

Hasil pengukuran waktu inferensi pada lima skenario representatif ditampilkan pada Gbr. 8(b). Inferensi pada gambar 640×640 piksel rata-rata menyelesaikan deteksi dalam 38,4 ms (setara dengan 26 FPS), sedangkan gambar 1280×1280 piksel membutuhkan 71,2 ms. Mode SAHI dengan slice 640×640 menghasilkan latensi paling tinggi yaitu 187,4 ms karena setiap gambar harus dipotong menjadi beberapa tile, masing-masing diinferensi, lalu hasilnya digabungkan kembali. Trade-off ini sejalan dengan temuan Mazen dan Shaker yang melaporkan kenaikan recall pada deteksi objek kecil sebesar 12% sampai 18% ketika SAHI diaktifkan, dengan biaya latensi 4 sampai 5 kali lipat [5]. Pada konteks Poli SEG, latensi 187,4 ms masih dapat diterima untuk skenario auto annotation batch, namun belum cocok untuk deteksi real-time pada video.



Gbr. 8. Hasil pengujian: (a) persentase keberhasilan per modul, (b) waktu inferensi per skenario.

TABEL III
WAKTU INFERENSI POLI SEG PADA LIMA SKENARIO

Skenario	Resolusi/Mode	Waktu Inferensi Rata-rata (ms)	FPS Estimasi
Image tunggal	640×640	38,4	26,0
Image tunggal HD	1280×1280	71,2	14,0
Video frame-by-frame	720p	46,8	21,4
Camera real-time	640×480, 5 FPS	52,5	19,0
SAHI batch	Slice 640, ov.20	187,4	5,3

4.3 Verifikasi Mekanisme Hot-Swap dan Pembahasan

Mekanisme hot-swap diuji dengan 30 percobaan penggantian model antar enam model yang berbeda (yolov8n, yolov8s, yolov8s-seg, yolov11s, rtdetr-l, custom_v2). Seluruh 30 percobaan berhasil dengan tingkat keberhasilan 100%, dengan waktu rata-rata pergantian (dari klik tombol Activate hingga model siap digunakan) sebesar 1,27 detik di mana sebagian besar waktu tersedot oleh proses load_model dan perhitungan hash MD5. Verifikasi keberhasilan dilakukan dengan menjalankan satu inferensi uji setelah setiap pergantian dan memastikan output kelas sesuai dengan kelas model yangbaru

DOI: https://doi.org/10.59025/48b50fbf | https://jtech.onlinelibrary.id

diaktifkan. Pengujian ini juga menemukan bahwa konsumsi memori GPU stabil setelah penggantian, mengonfirmasi bahwa fungsi clear_model_cache() bekerja sebagaimana mestinya.

Hasil ini bermakna karena mengisi celah yang teridentifikasi pada Tabel I. Studi sebelumnya seperti Abba dkk. [7] mendukung deteksi dan tracking namun tidak mendukung pertukaran model dinamis, sementara Rashad dan Ali [11] menerapkan SAHI tetapi terikat pada satu model dan satu platform Android. Poli SEG mengintegrasikan kedua aspek multi-tugas dan hot-swap dalam satu antarmuka web. Mekanisme berbasis hash MD5 yang digunakan menambah lapisan keamanan: ketika peneliti menulis ulang file model dengan nama yang sama (kasus umum saat retraining), sistem otomatis mendeteksi perubahan isi dan memuat ulang model tanpa intervensi manual.

Perlu dicatat keterbatasan penelitian ini. Pertama, pengujian dilakukan pada lingkungan single-user dan belum mencakup skenario multi-user konkuren penggantian model oleh satu pengguna saat ini akan mempengaruhi seluruh pengguna lain. Kedua, dukungan format model masih dominan pada .pt (PyTorch) meskipun arsitektur loader memungkinkan ekstensi ke .onnx dan TensorRT. Ketiga, performa tracking pada oklusi berat menunjukkan ruang perbaikan, sejalan dengan temuan Abba dkk. yang mengusulkan kombinasi background subtraction dan deep learning untuk kasus serupa [7]. Implikasi praktis dari hasil ini adalah Poli SEG cocok dipakai sebagai sandbox eksperimen di tahap pengujian model bukan sebagai sistem produksi multi-tenant dengan ekstensi yang dapat dikembangkan menuju arah tersebut.

5. KESIMPULAN

Penelitian ini telah mengembangkan Poli SEG, sebuah platform web modular berbasis Flask yang menyatukan deteksi objek, segmentasi instance, pelacakan objek, dan anotasi otomatis dengan mekanisme manajemen model dinamis berbasis hash MD5. Pengujian fungsional black-box dengan 60 skenario menunjukkan rata-rata keberhasilan 93,3%, sementara pengukuran waktu inferensi memberikan latensi rata-rata 38,4 ms untuk gambar 640×640 dan 187,4 ms untuk mode SAHI. Verifikasi hot-swap pada 30 percobaan antar enam model berbeda berhasil dengan tingkat 100% dan waktu rata-rata 1,27 detik, menunjukkan bahwa pergantian model tanpa restart server merupakan kontribusi yang dapat direplikasi pada kerangka serupa. Arsitektur tiga-lapis berbasis Blueprint Flask memungkinkan setiap fitur diperluas secara independen, menjawab kebutuhan riset computer vision di lingkungan kampus akan platform pengujian universal.

Pengembangan lanjutan direkomendasikan pada empat arah konkret. Pertama, dukungan format model perlu diperluas dari .pt ke .onnx dan TensorRT untuk meningkatkan kecepatan inferensi pada perangkat produksi. Kedua, mekanisme isolasi sesi pengguna perlu diterapkan agar Poli SEG dapat dipakai oleh banyak peneliti secara konkuren tanpa konflik model. Ketiga, modul pelacakan dapat diperkuat dengan ByteTrack atau StrongSORT untuk menangani oklusi berat, sejalan dengan rekomendasi tinjauan keamanan oleh Abba dkk. [7]. Keempat, dashboard evaluasi perlu ditambahkan agar peneliti dapat memvisualisasikan confusion matrix, mAP, dan F1-score langsung dari sesi inferensi mengubah Poli SEG dari platform pengujian menjadi platform evaluasi end-to-end. Arah-arah ini akan memperluas dampak Poli SEG dari sandbox riset internal menuju platform pengujian model yang siap digunakan oleh komunitas computer vision yang lebih luas.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Politeknik Negeri Lampung atas dukungan fasilitas perangkat keras dan akses dataset internal yang digunakan dalam pengujian Poli SEG.

KONTRIBUSI PENULIS

Sylvia: konseptualisasi, metodologi, supervisi, dan revisi akhir naskah. Atika Arpan: implementasi modul backend, pengujian black-box, dan penulisan draf awal. Rizka Permata: perancangan antarmuka pengguna, dokumentasi, dan validasi data hasil pengujian. Muhammad Reza Redo Islami: integrasi model YOLO dan SAHI, pengukuran waktu inferensi, dan revisi naskah.

PERNYATAAN KONFLIK KEPENTINGAN

Penulis menyatakan tidak terdapat konflik kepentingan dalam pelaksanaan penelitian dan penulisan artikel ini.

REFERENSI

[1]J. Terven, D.-M. Córdova-Esparza, dan J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," Mach. Learn. Knowl. Extr., vol. 5, no. 4, hal. 1680–1716, 2023, doi: 10.3390/make5040083.

[2] R. Sapkota et al., "YOLO advances to its genesis: a decadal and comprehensive review of the You Only Look Once (YOLO) series," Artif. Intell. Rev., vol. 58, no. 7, 2025, doi: 10.1007/s10462-025-11253-3.

[3] L. Wang et al., "A systematic literature review of lightweight YOLO models for object detection," PeerJ Comput. Sci., vol. 11, 2025, doi: 10.7717/peerj-cs.3357.

[4]Y. Zhao et al., "DETRs Beat YOLOs on Real-time Object Detection," dalam Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR), 2024, hal. 16965–16974, doi: 10.1109/CVPR52733.2024.01605.

[5]F. M. A. Mazen dan Y. Shaker, "Small Object Detection in Complex Images: Evaluation of Faster R-CNN and Slicing Aided Hyper Inference," Int. J. Adv. Comput. Sci. Appl., vol. 16, no. 3, hal. 951–960, 2025, doi: 10.14569/IJACSA.2025.0160393.

[6]M. Ishtiaq dan J.-U. Won, "YOLO-SIFD: YOLO with Sliced Inference and Fractal Dimension Analysis for Improved Fire and Smoke Detection," Comput. Mater. Continua, vol. 82, no. 3, hal. 5343–5361, 2025, doi: 10.32604/cmc.2025.061466.

[7] S. Abba, A. M. Bizi, J.-A. Lee, S. Bakouri, dan M. L. Crespo, "Real-time object detection, tracking, and monitoring framework for security surveillance systems," Heliyon, vol. 10, no. 15, 2024, doi: 10.1016/j.heliyon.2024.e34922.

[8] N. N. Zhorif, R. K. Anandyto, A. U. Rusyadi, dan E. Irwansyah, "Implementation of Slicing Aided Hyper Inference (SAHI) in YOLOv8 to Counting Oil Palm Trees Using High-Resolution Aerial Imagery Data," Int. J. Adv. Comput. Sci. Appl., vol. 15, no. 7, 2024, doi: 10.14569/IJACSA.2024.0150786.

[9] S. Kang, H. Hu, dan L. Wang, "Object Detection YOLO Algorithms and Their Industrial Applications: Overview and Comparative Analysis," Electronics, vol. 14, no. 6, 2025, doi: 10.3390/electronics14061104.

[10]J. E. Gallagher dan E. J. Oughton, "Surveying You Only Look Once (YOLO) Multispectral Object Detection: Advancements, Applications, and Challenges," IEEE Access, vol. 13, hal. 7366–7395, 2025, doi: 10.1109/ACCESS.2025.3526458.

[11]M. F. Rashad dan Q. I. Ali, "Deploying Android-Based Smart RSUs with YOLOv8 and SAHI for Enhanced Traffic Management," Diyala J. Eng. Sci., vol. 18, no. 1, hal. 70–90, 2025, doi: 10.24237/djes.2025.18104.

[12]J. Wei, X. Zhao, X. Yang, dan G. Liu, "A Review of YOLO Algorithm and Its Applications in Autonomous Driving Object Detection," IEEE Access, vol. 13, hal. 93688–93711, 2025, doi: 10.1109/ACCESS.2025.3573376.

[13] M. Mao, Y. Hong, dan H. Liu, "YOLO Object Detection for Real-Time Fabric Defect Inspection in the Textile Industry: A Review of YOLOv1 to YOLOv11," Sensors, vol. 25, no. 7, 2025, doi: 10.3390/s25072270.

[14] M. G. Ragab et al., "A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023)," IEEE Access, vol. 12, 2024, doi: 10.1109/ACCESS.2024.3386826.

[15]Z. Wang, C. Li, H. Xu, dan X. Zhu, "Mamba YOLO: A Simple Baseline for Object Detection with State Space Model," dalam Proc. AAAI Conf. Artif. Intell., vol. 39, no. 8, 2025, hal. 8205–8213, doi: 10.1609/aaai.v39i8.32885.

[16]T. Cheng et al., "YOLO-World: Real-Time Open-Vocabulary Object Detection," dalam Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR), 2024, hal. 16901–16911, doi: 10.1109/CVPR52733.2024.01599.

[17] J. Doherty, B. Gardiner, E. Kerr, dan N. Siddique, "BiFPN-YOLO: One-stage object detection integrating Bi-Directional Feature Pyramid Networks," Pattern Recognit., vol. 160, art. 111209, 2025, doi: 10.1016/j.patcog.2024.111209.

[18] D. Dodić et al., "SAHI-Tuned YOLOv5 for UAV Detection of TM-62 Anti-Tank Landmines: Small-Object, Occlusion-Robust, Real-Time Pipeline," Computers, vol. 14, no. 10, 2025, doi: 10.3390/computers14100448.

[19]Z. Cai dan N. Vasconcelos, "Cascade R-CNN: High Quality Object Detection and Instance Segmentation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 5, hal. 1483–1498, 2021, doi: 10.1109/TPAMI.2019.2956516.

[20] M. Kang, C.-M. Ting, F. F. Ting, dan R. C.-W. Phan, "ASF-YOLO: A novel YOLO model with attentional scale sequence fusion for cell instance segmentation," Image Vis. Comput., vol. 147, art. 105057, 2024, doi: 10.1016/j.imavis.2024.105057.

[21] X. Yuan, H. Yu, T. Geng, R. Ma, dan P. Li, "Enhancing sustainable Chinese cabbage production: a comparative analysis of multispectral image instance segmentation techniques," Front. Sustain. Food Syst., vol. 8, art. 1433701, 2024, doi: 10.3389/fsufs.2024.1433701.

[22]M. Lalinia dan A. Sahafi, "Colorectal polyp detection in colonoscopy images using YOLO-V8 network," Signal Image Video Process., vol. 18, no. 3, hal. 2047–2058, 2024, doi: 10.1007/s11760-023-02835-1.

[23]M. Alaftekin, I. Pacal, dan K. Cicek, "Real-time sign language recognition based on YOLO algorithm," Neural Comput. Appl., vol. 36, hal. 7609–7624, 2024, doi: 10.1007/s00521-024-09503-6.

[24] Z. Fei et al., "Comprehensive Performance Evaluation of YOLOv11, YOLOv10, YOLOv9, YOLOv8 and YOLOv5 on Object Detection of Power Equipment," arXiv preprint arXiv:2411.18871, 2024, doi: 10.48550/arXiv.2411.18871.

[25] E. H. Alkhammash, "A Comparative Analysis of YOLOv9, YOLOv10, YOLOv11 for Smoke and Fire Detection," Fire, vol. 8, no. 1, art. 26, 2025, doi: 10.3390/fire8010026.

[26] Y. Alhwaiti et al., "Leveraging YOLO deep learning models to enhance plant disease identification," Sci. Rep., vol. 15, art. 92143, 2025, doi: 10.1038/s41598-025-92143-0.

[27]M. Al Husaini, A. R. Raharja, V. H. C. Putra, dan H. H. Lukmana, "Enhanced Plant Disease Detection Using Computer Vision YOLOv11: Pre-Trained Neural Network Model Application," J. Comput. Netw. Archit. High Perform., vol. 7, no. 1, hal. 82–95, 2025, doi: 10.47709/cnahpc.v7i1.5146.