Journal of Authentic Research
https://journal-center.litpam.com/index.php/jar/index

Mei 2026 Vol. 5, No. 2
e-ISSN: 2828-3724
pp. 2309-2321

Implementasi *YOLOv8* dan *TensorFlow Lite* pada Aplikasi Mobile untuk Deteksi Kerusakan Jalan secara Real-Time

[1]Syauqi Revo Mardian, [2]Khairi Budayawan, [3]Titi Sriwahyuni, [4]Hadi kurnia saputra

[1,2,3,4] Program Studi Informatika, Departemen Teknik Elektronika, Fakultas Teknik, Universitas Negeri Padang, Indonesia

*Corresponding Author e-mail: revolintau@gmail.com

Received: Month 2026; Revised: Month 2026; Published: Month 2026

Abstrak

Kerusakan jalan seperti lubang dan retak memerlukan mekanisme deteksi dan pelaporan yang cepat, objektif, serta berbasis lokasi agar pemeliharaan infrastruktur dapat dilakukan secara lebih responsif. Penelitian ini bertujuan mengembangkan purwarupa aplikasi mobile untuk mendeteksi dan melaporkan kerusakan jalan dengan mengintegrasikan YOLOv8s, TensorFlow Lite, kamera smartphone, GPS, SQLite, RESTful API Laravel, MySQL, dan dashboard Geographic Information System (GIS). Kebaruan penelitian diposisikan secara realistis sebagai kontribusi integratif pada alur kerja hybrid edge-cloud, bukan sebagai kontribusi algoritmik baru, karena sistem menghubungkan inferensi lokal pada perangkat, penyimpanan offline saat blank spot, sinkronisasi ke server, visualisasi geografis, dan validasi administrator dalam satu ekosistem pelaporan. Penelitian menggunakan pendekatan research and development dengan model Waterfall. Dataset disusun secara hibrida dari dataset publik dan data lokal, difokuskan pada kelas pothole dan crack, dipraproses pada resolusi 640 x 640 piksel, serta dibagi menjadi data latih, validasi, dan uji. Hasil pengujian fungsional menunjukkan bahwa fitur autentikasi, aktivasi kamera, deteksi objek, pengambilan koordinat, pengiriman laporan, penyimpanan offline, visualisasi GIS, dan validasi status berjalan sesuai skenario. Model YOLOv8s-TFLite menghasilkan precision rata-rata 69,4%, recall 66,6%, dan mAP50 67,7%; kelas pothole mencapai mAP50 76,6%, sedangkan kelas crack mencapai mAP50 58,8%. Performa yang lebih rendah pada crack menunjukkan bahwa retakan jalan masih sulit direpresentasikan dengan bounding box karena bentuknya tipis, memanjang, dan sering bercampur dengan tekstur aspal. Dengan demikian, sistem layak sebagai purwarupa pendukung pelaporan awal, tetapi klaim operasional real-time dan kesiapan lapangan tetap memerlukan pengukuran tambahan berupa FPS, latency, ukuran model, penggunaan memori, konsumsi baterai, dan pengujian lintas perangkat.

Kata kunci: Aplikasi Mobile; Deteksi Kerusakan Jalan; Edge Ai; Pelaporan Berbasis Lokasi; Tensorflow Lite; Yolov8

Implementation of YOLOv8 and TensorFlow Lite in Mobile Application for Real-Time Road Damage Detection

Abstract

Road damage, including potholes and cracks, requires a fast, objective, and location-based detection and reporting mechanism to support more responsive infrastructure maintenance. This study develops a mobile application prototype for road damage detection and reporting by integrating YOLOv8s, TensorFlow Lite, smartphone cameras, GPS, SQLite, a Laravel RESTful API, MySQL, and a Geographic Information System (GIS) dashboard. The novelty is positioned as an integrative contribution to a hybrid edge-cloud workflow rather than a new algorithmic contribution, because the system connects on-device inference, offline storage in blank-spot areas, server synchronization, geographic visualization, and administrator validation within a single reporting ecosystem. The study adopted a research-and-development approach using the Waterfall model. The dataset was prepared using a hybrid strategy that combined public datasets and local field data, focused on pothole and crack classes, was preprocessed at 640 x 640 pixels, and was split into training, validation, and testing subsets. Functional testing showed that authentication, camera activation, object detection, coordinate acquisition, report submission, offline storage, GIS visualization, and status validation operated according to the expected scenarios. The YOLOv8s-TFLite model achieved an average precision of 69.4%, recall of 66.6%, and mAP50 of 67.7%; pothole detection obtained an mAP50 of 76.6%, whereas crack detection obtained an mAP50 of 58.8%. The lower crack performance indicates that road cracks remain difficult to represent using bounding boxes because they are thin, elongated, irregular, and visually mixed with asphalt texture. Therefore, the system is suitable as an early reporting-support prototype, but operational real-time readiness still requires additional measurements of FPS, latency, model size, memory usage, battery consumption, and cross-device testing.

Keywords: *edge AI; location-based reporting; mobile application; road damage detection; TensorFlow Lite; YOLOv8*

How to Cite: Mardian, S. R., Budayawan, K. ., Sriwahyuni, T. ., & kurnia , H. . (2026). Implementasi YOLOv8 dan TensorFlow Lite pada Aplikasi Mobile untuk Deteksi Kerusakan Jalan secara Real-Time. *Journal of Authentic Research*, *5*(2), 2309-2321. https://doi.org/10.36312/qwq2bk22

https://doi.org/10.36312/qwq2bk22

PENDAHULUAN

Jalan merupakan infrastruktur dasar yang menentukan konektivitas sosial, distribusi barang, akses layanan publik, keselamatan transportasi, dan efisiensi mobilitas masyarakat. Dalam wilayah dengan intensitas lalu lintas tinggi dan variasi geografis yang besar, kualitas permukaan jalan menjadi faktor penting bagi kelancaran aktivitas ekonomi dan keselamatan pengguna. Kerusakan permukaan seperti lubang dan retak tidak hanya mengurangi kenyamanan berkendara, tetapi juga dapat meningkatkan risiko kecelakaan, mempercepat kerusakan kendaraan, dan menaikkan biaya pemeliharaan. Literatur terkini menegaskan bahwa pothole dan crack merupakan dua bentuk kerusakan jalan yang paling sering menjadi perhatian dalam sistem inspeksi berbasis visi komputer karena keduanya mudah diamati secara visual, tetapi memiliki kompleksitas bentuk yang berbeda (Ma et al., 2022; Safyari et al., 2024).

Permasalahan utama dalam pengelolaan kerusakan jalan bukan hanya keberadaan kerusakan fisik, melainkan juga lambatnya aliran informasi dari lokasi kerusakan menuju unit pengambil keputusan. Inspeksi manual masih memerlukan petugas lapangan, waktu survei, biaya operasional, dan penilaian visual yang dapat berbeda antar-surveyor. Laporan masyarakat sering belum dilengkapi bukti visual, koordinat geografis yang akurat, status waktu, dan kategori kerusakan yang jelas. Akibatnya, proses verifikasi, pemetaan prioritas, dan perencanaan perbaikan dapat berjalan lambat. Sistem pengaduan berbasis Android dan GIS telah menunjukkan potensi untuk memperbaiki kualitas informasi laporan, tetapi pendekatan tersebut masih perlu diperkuat dengan deteksi otomatis agar data yang dikirim lebih objektif dan konsisten (Falasyfa & Avianto, 2024; Hafidatur et al., 2021).

Computer vision dan deep learning menawarkan pendekatan yang lebih objektif karena sistem dapat mempelajari pola visual kerusakan dari citra digital. Convolutional Neural Network (CNN) mampu mengekstraksi fitur visual secara berlapis, mulai dari tepi, tekstur, warna, hingga pola objek yang lebih kompleks, sehingga banyak digunakan pada klasifikasi dan deteksi objek (Alzubaidi et al., 2021; Bhatt et al., 2021; Chai et al., 2021). Dalam deteksi kerusakan jalan, keluarga You Only Look Once (YOLO) banyak dipilih karena menggabungkan lokalisasi dan klasifikasi dalam satu tahap inferensi. Karakter ini membuat YOLO lebih sesuai untuk kebutuhan aplikasi responsif dibandingkan pendekatan dua tahap yang umumnya lebih berat secara komputasi (Jiang et al., 2022; Terven & Cordova-Esparza, 2023).

Namun, sebagian besar penelitian deteksi kerusakan jalan masih berhenti pada evaluasi model di lingkungan komputasi eksperimen. Banyak studi melaporkan precision, recall, atau mAP, tetapi belum menguji bagaimana model dijalankan pada smartphone, bagaimana data lapangan dikirim ketika jaringan tidak stabil, bagaimana laporan divisualisasikan pada peta, dan bagaimana validasi manusia dilakukan sebelum laporan digunakan sebagai dasar tindakan. Kesenjangan ini penting karena sistem yang akurat di komputer pengembangan belum tentu layak digunakan pada perangkat mobile yang memiliki batasan memori, CPU, GPU, konsumsi daya, suhu, dan latensi.

Penerapan deep learning pada smartphone menuntut kompromi antara akurasi dan efisiensi. Model dengan akurasi tinggi tidak selalu layak dijalankan pada perangkat terbatas karena ukuran parameter, kebutuhan memori, konsumsi daya, dan waktu inferensi. Penelitian berbasis ResNet-18, misalnya, dapat memberikan performa klasifikasi yang baik, tetapi kompleksitas arsitektur menjadi hambatan untuk skenario perangkat mobile (Saparudin et al., 2025). Sebaliknya, model ringan yang dikonversi ke TensorFlow Lite memungkinkan inferensi lokal pada perangkat, mengurangi ketergantungan terhadap server, dan membuka peluang penggunaan di wilayah dengan konektivitas tidak stabil (Nova & Rianto, 2025).

Kesenjangan riset yang ditangani dalam penelitian ini terletak pada minimnya studi yang menghubungkan object detection, edge inference, pelaporan berbasis lokasi, penyimpanan offline, sinkronisasi server, visualisasi GIS, dan validasi administrator dalam satu alur sistem. Dengan kata lain, persoalan yang belum cukup terjawab bukan hanya 'apakah model dapat mendeteksi pothole dan crack', tetapi 'apakah model tersebut dapat ditempatkan dalam alur pelaporan mobile yang dapat digunakan pada kondisi lapangan yang bervariasi'. Oleh sebab itu, novelty penelitian ini diposisikan secara hati-hati sebagai novelty integratif dan implementatif, bukan novelty algoritmik.

Penelitian ini merancang aplikasi mobile Android yang mengintegrasikan YOLOv8s-TFLite, kamera smartphone, GPS, SQLite, Laravel RESTful API, MySQL, dan dashboard GIS. Arsitektur yang diusulkan mengandalkan hybrid edge-cloud workflow: deteksi awal dilakukan pada perangkat pengguna agar respons lebih cepat, sementara pengelolaan data, pemetaan, dan validasi dilakukan pada server. Mekanisme offline synchronization ditambahkan agar laporan tetap dapat disimpan ketika pengguna berada di area blank spot dan dikirim ulang saat jaringan tersedia. Pendekatan human-in-the-loop diterapkan agar hasil prediksi AI tidak langsung menjadi keputusan akhir, tetapi diverifikasi oleh pengguna dan administrator.

Tujuan penelitian ini adalah merancang, membangun, dan mengevaluasi purwarupa aplikasi mobile untuk deteksi serta pelaporan kerusakan jalan berbasis lokasi. Objek deteksi dibatasi pada dua kelas, yaitu pothole dan crack. Evaluasi model menggunakan precision, recall, dan mAP50, sedangkan evaluasi sistem menggunakan black box testing pada fitur utama serta rancangan pengukuran deployment mobile yang mencakup latency, FPS, waktu pemuatan model, ukuran model, penggunaan memori, dan konsumsi baterai. Dengan batasan tersebut, artikel ini memberikan kontribusi praktis berupa purwarupa pelaporan jalan rusak berbasis edge AI dan kontribusi teknis berupa dokumentasi integrasi YOLOv8s-TFLite dalam arsitektur mobile-to-GIS yang tetap disertai pembatasan klaim real-time secara ilmiah.

METODE

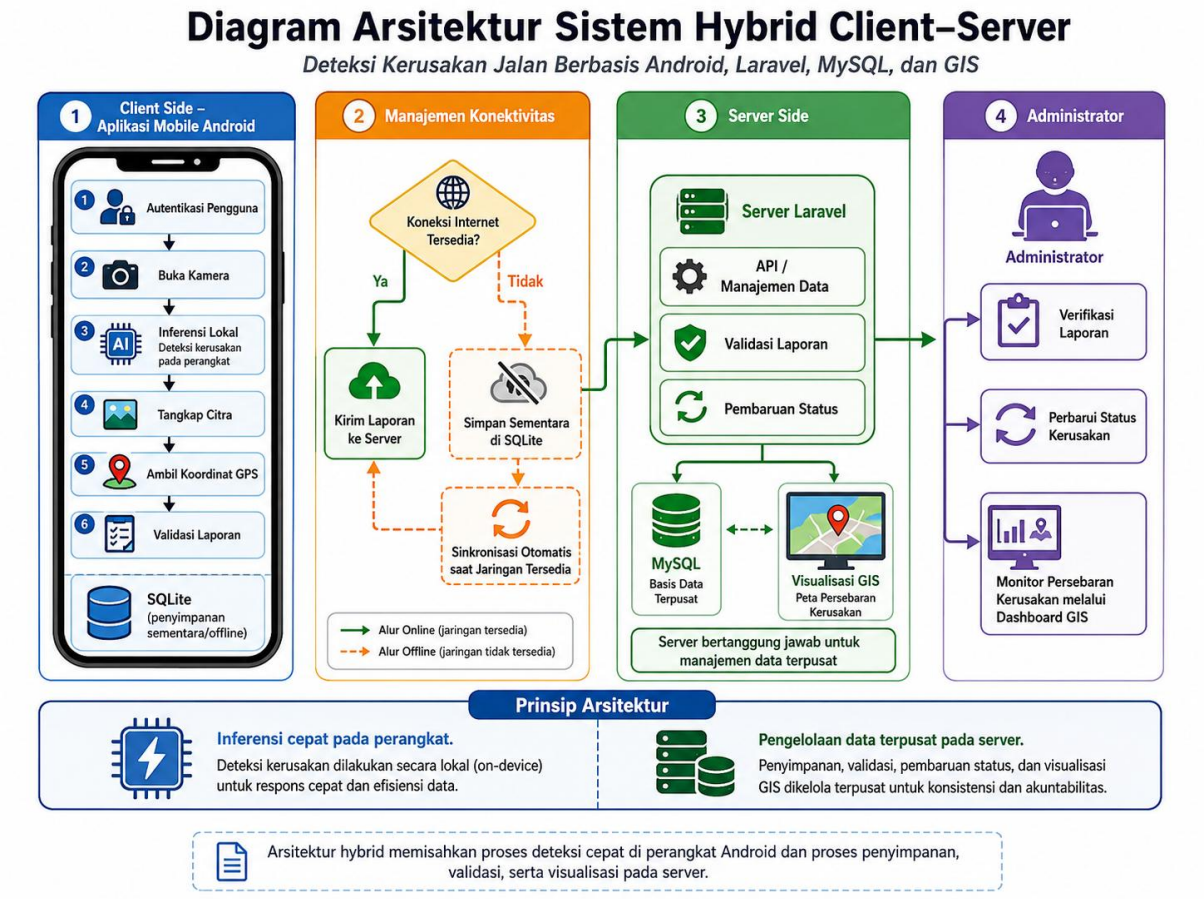
Jenis Penelitian dan Posisi Kontribusi

Penelitian ini menggunakan pendekatan penelitian dan pengembangan perangkat lunak untuk menghasilkan purwarupa aplikasi deteksi dan pelaporan kerusakan jalan. Model pengembangan yang digunakan adalah System Development Life Cycle dengan pendekatan Waterfall karena tahapan kebutuhan, desain, implementasi, pengujian, dan evaluasi dapat disusun secara berurutan serta terdokumentasi. Pendekatan ini relevan untuk sistem dengan kebutuhan utama yang telah dapat diidentifikasi sejak awal, yaitu deteksi objek kerusakan, pengambilan koordinat lokasi, pengiriman laporan, penyimpanan offline, visualisasi sebaran kerusakan, dan validasi administrator (Fachri & Rizal, 2024; Wahid, 2020). Secara ilmiah, penelitian ini tidak mengusulkan arsitektur deteksi baru, melainkan mengevaluasi integrasi object detection berbasis YOLOv8s-TFLite ke dalam ekosistem mobile reporting berbasis edge-cloud.

Arsitektur Sistem dan Alur Kerja

Sistem dirancang sebagai arsitektur hybrid client-server. Aplikasi mobile Android berfungsi sebagai lapisan akuisisi data dan inferensi lokal, sedangkan server Laravel dan MySQL berfungsi sebagai lapisan manajemen data, visualisasi GIS, dan validasi laporan. Alur sistem dimulai ketika pengguna melakukan autentikasi, membuka kamera, menjalankan deteksi kerusakan, menangkap citra, mengambil koordinat GPS, memvalidasi laporan, dan mengirim data ke server. Apabila koneksi tidak tersedia, laporan disimpan sementara pada SQLite dan disinkronkan ketika jaringan kembali tersedia. Administrator kemudian memverifikasi

laporan, memperbarui status, dan memonitor persebaran kerusakan melalui dashboard GIS. Arsitektur ini sengaja dirancang untuk memisahkan inferensi cepat pada perangkat dan pengelolaan data terpusat pada server.



Gambar 1. Arsitektur hybrid edge-cloud untuk deteksi dan pelaporan kerusakan jalan berbasis lokasi.

Subjek, Aktor, dan Lingkungan Sistem

Aktor sistem terdiri atas pengguna aplikasi dan administrator. Pengguna aplikasi dapat berupa masyarakat atau petugas lapangan yang melakukan autentikasi, membuka kamera, menangkap citra kerusakan, memperoleh hasil deteksi, mengirim laporan, dan melihat riwayat laporan. Administrator berperan memonitor data, memverifikasi laporan, mengelola kategori kerusakan, dan melihat persebaran titik kerusakan melalui dashboard GIS. Lingkungan perangkat keras mencakup komputer pengembangan, smartphone Android, dan server aplikasi. Perangkat lunak yang digunakan meliputi Flutter dan Dart untuk aplikasi mobile, Laravel dan PHP untuk backend, MySQL melalui XAMPP sebagai basis data, Google Colab untuk pelatihan model, Python untuk pengolahan model, serta browser untuk mengakses dashboard administrator.

Subjek, Aktor, dan Lingkungan Sistem

Aktor sistem terdiri atas pengguna aplikasi dan administrator. Pengguna aplikasi dapat berupa masyarakat atau petugas lapangan yang melakukan autentikasi, membuka kamera, menangkap citra kerusakan, memperoleh hasil deteksi, mengirim laporan, dan melihat riwayat laporan. Administrator berperan memonitor data, memverifikasi laporan, mengelola kategori kerusakan, dan melihat persebaran titik kerusakan melalui dashboard GIS. Lingkungan perangkat keras mencakup komputer pengembangan, smartphone Android, dan server aplikasi. Perangkat lunak yang digunakan meliputi Flutter dan Dart untuk aplikasi mobile, Laravel dan PHP untuk backend, MySQL melalui XAMPP sebagai basis data, Google Colab untuk pelatihan model, Python untuk pengolahan model, serta browser untuk mengakses dashboard administrator.

Tabel 1. Aktor dan peran sistem

Kategori Pengguna	Peran Utama	Keluaran Sistem
Pengguna aplikasi	Autentikasi, deteksi kerusakan, capture citra, pengambilan koordinat, pengiriman laporan, pemantauan riwayat	Foto kerusakan, koordinat GPS, label deteksi, confidence score, status sinkronisasi, status laporan
Administrator	Monitoring laporan, validasi laporan, manajemen kategori, visualisasi GIS, pembaruan status laporan	Peta sebaran, status validasi, statistik laporan, data master kerusakan, data prioritas tindak lanjut

Dataset, Anotasi, dan Pra-pemrosesan

Dataset citra disusun dengan pendekatan hibrida, yaitu menggabungkan dataset publik dari repositori terbuka dengan data lokal yang menggambarkan kondisi jalan pada wilayah pengujian. Pendekatan hibrida dipilih untuk memperluas variasi visual sekaligus menjaga relevansi terhadap karakteristik lapangan lokal. Seluruh citra difokuskan pada dua kelas, yaitu pothole dan crack. Untuk memperkuat validitas eksperimen, naskah revisi ini menegaskan bahwa sumber dataset, jumlah total citra, jumlah citra per kelas, proporsi data publik dan lokal, perangkat pengambilan citra, kondisi pencahayaan, variasi cuaca, sudut kamera, dan prosedur anotasi harus dilaporkan secara eksplisit. Informasi tersebut penting karena performa object detection sangat dipengaruhi oleh kualitas anotasi, keragaman lingkungan, dan distribusi kelas.

Anotasi dilakukan dalam format bounding box dengan label pothole dan crack. Setiap anotasi idealnya diperiksa minimal oleh dua anotator atau divalidasi oleh peneliti utama untuk mengurangi kesalahan label. Pra-pemrosesan mencakup penyeragaman resolusi menjadi 640 x 640 piksel, konversi format dataset agar kompatibel dengan pipeline YOLO, pemeriksaan duplikasi citra, dan pengecekan konsistensi label. Augmentasi hanya diterapkan pada data latih untuk menambah

variasi visual dan mengurangi risiko overfitting. Transformasi yang dapat digunakan meliputi perubahan brightness, contrast, blur ringan, rotasi terbatas, horizontal flip, dan skala, tetapi harus dijaga agar tidak mengubah makna visual kerusakan.

Tabel 2. Pembagian dataset penelitian

Subset	Proporsi	Sumber Dominan	Fungsi Teknis
Training	70%	Gabungan data publik dan lokal	Pembelajaran fitur visual dasar dan karakteristik lokal; augmentasi diterapkan hanya pada subset ini
Validation	20%	Gabungan data publik dan lokal	Pemantauan performa per epoch, pemilihan konfigurasi model, dan pencegahan overfitting
Testing	10%	Data lokal dan/atau skenario lapangan	Pengujian akhir model pada data yang tidak digunakan dalam pelatihan maupun validasi

Pelatihan, Konversi Model, dan Lingkungan Komputasi

Model yang digunakan adalah YOLOv8s karena varian ini menawarkan kompromi antara akurasi dan efisiensi dibandingkan model yang lebih besar. Model dilatih selama 100 epoch dengan batch size 16, optimizer Adam, dan learning rate awal 0,001. Setelah pelatihan selesai, bobot model dikonversi dari format PyTorch ke TensorFlow Lite agar dapat dijalankan pada perangkat Android. Proses konversi perlu dilaporkan secara rinci, termasuk apakah model menggunakan float16, integer quantization, atau dynamic range quantization. Informasi ini penting karena ukuran file, latency, penggunaan memori, dan akurasi model dapat berubah setelah konversi.

Tabel 3. Lingkungan pelatihan dan deployment yang perlu dicantumkan

Komponen	Spesifikasi
Platform pelatihan	Google Colab; jenis GPU, RAM, dan versi runtime
Framework pelatihan	Python versi Ultralytics/YOLOv8, PyTorch, CUDA
Parameter pelatihan	YOLOv8s, 100 epoch, batch size 16, Adam, learning rate 0,001, input 640 x 640 piksel
Format model awal	PyTorch weights
Format deployment	TensorFlow Lite
Server/backend	Laravel, PHP, MySQL/XAMPP

Teknik Evaluasi dan Analisis Data

Evaluasi dilakukan dalam tiga kelompok. Pertama, evaluasi model menghitung precision, recall, dan mAP50 berdasarkan prinsip evaluasi object detection yang menggunakan hubungan antara bounding box prediksi dan ground truth melalui Intersection over Union (Padilla et al., 2021). Precision menunjukkan ketepatan prediksi positif, recall menunjukkan kemampuan menemukan objek yang benar-benar ada, sedangkan mAP50 merangkum kualitas deteksi pada ambang IoU 0,50. Untuk memperkuat rigor evaluasi, naskah final sebaiknya menambahkan confusion matrix berbasis kelas objek, PR curve, mAP50-95, dan analisis false positive serta false negative.

Kedua, evaluasi deployment mobile perlu mengukur metrik runtime karena klaim aplikasi mobile responsif tidak cukup dibuktikan dengan mAP. Metrik yang perlu dicatat meliputi waktu pemuatan model, latency inferensi per frame, FPS, penggunaan RAM, ukuran model, suhu perangkat, dan konsumsi baterai selama skenario penggunaan tertentu. Ketiga, black box testing digunakan untuk memeriksa kesesuaian keluaran sistem terhadap skenario penggunaan tanpa meninjau kode internal. Pengujian fungsional diperluas dengan skenario error dan robustness, seperti password salah, file kosong, GPS tidak aktif, koneksi terputus saat upload, duplikasi laporan, sinkronisasi gagal, dan pemulihan sinkronisasi.

HASIL DAN PEMBAHASAN

Implementasi Arsitektur Sistem

Hasil implementasi menunjukkan bahwa sistem berhasil dibangun sebagai arsitektur hybrid edge-cloud. Lapisan client berupa aplikasi Android menjalankan akuisisi citra, inferensi YOLOv8s-TFLite, pengambilan koordinat GPS, validasi pengguna, penyimpanan lokal, dan pengiriman laporan. Lapisan server dibangun dengan Laravel dan MySQL untuk menerima laporan, menyimpan metadata, menyediakan layanan peta, menghitung statistik, dan mendukung validasi administrator. Pemisahan tugas ini membuat inferensi AI tetap dapat dilakukan pada perangkat pengguna, sedangkan pengelolaan data tetap terpusat dan dapat dipantau. Dari sisi kontribusi sistem, arsitektur ini lebih kuat dibandingkan aplikasi pengaduan manual karena setiap laporan membawa bukti visual, koordinat, status sinkronisasi, dan hasil deteksi awal.

Kekuatan utama arsitektur bukan terletak pada penciptaan algoritma deteksi baru, melainkan pada integrasi deteksi lokal dengan alur pelaporan operasional. Inferensi lokal membantu mengurangi kebutuhan pengiriman video atau citra mentah secara terus-menerus ke server. SQLite berperan sebagai mekanisme toleransi jaringan ketika pengguna berada pada area blank spot. Dashboard GIS berperan mengubah laporan individual menjadi informasi spasial yang dapat dipantau oleh administrator. Dengan demikian, sistem berfungsi sebagai prototipe decision-support workflow untuk pelaporan awal, bukan sebagai instrumen tunggal untuk penilaian teknis jalan.

Implementasi Fitur Aplikasi Mobile dan Dashboard

Pada sisi pengguna, aplikasi menyediakan halaman login, registrasi, dashboard, deteksi kamera, validasi laporan, riwayat laporan, pengaturan akun, pusat bantuan, dan informasi aplikasi. Halaman deteksi menjadi fitur utama karena sistem memuat model ke memori smartphone dan memindai frame kamera untuk mendeteksi pothole atau crack. Setelah pengguna menekan tombol capture, halaman validasi menampilkan foto bukti, bounding box, label kerusakan, confidence score, serta koordinat latitude dan longitude. Pada sisi administrator, dashboard menyediakan ringkasan jumlah laporan, status pending, laporan tervalidasi, statistik, tren pelaporan, peta GIS, halaman validasi, dan pengelolaan data master. Desain human-in-the-loop ini penting karena prediksi AI tidak langsung dianggap sebagai kebenaran final, tetapi tetap membutuhkan validasi manusia.

Hasil Pengujian Fungsional dan Robustness

Black box testing menunjukkan bahwa skenario utama berjalan sesuai harapan. Sistem mampu memverifikasi kredensial pengguna, mengaktifkan kamera dan model AI, menangkap citra beserta koordinat GPS, mengirim laporan saat jaringan tersedia, menyimpan laporan pada basis data lokal saat offline, menampilkan titik kerusakan pada peta GIS, dan memperbarui status laporan setelah validasi administrator. Hasil ini menunjukkan bahwa sistem tidak hanya bergantung pada kemampuan model deteksi, tetapi juga pada keandalan alur data dari pengguna ke server.

Tabel 4. Ringkasan black box testing dan skenario robustness

No.	Skenario	Hasil yang Diharapkan	Status
1	Login dengan email dan password valid	Pengguna masuk ke dashboard	Berhasil
2	Login dengan password salah	Sistem menolak akses dan menampilkan pesan kesalahan	Berhasil
3	Membuka kamera deteksi	Kamera aktif dan model memindai frame	Berhasil
4	Capture objek kerusakan	Foto, bounding box, confidence score, dan koordinat GPS terkunci	Berhasil
5	GPS tidak aktif	Sistem meminta pengguna mengaktifkan lokasi atau menandai laporan belum lengkap	Berhasil
6	Kirim laporan online	Data tersimpan pada server Laravel dan MySQL	Berhasil
7	Koneksi terputus saat upload	Data tidak hilang dan masuk antrean SQLite	Berhasil
8	Kirim laporan saat blank spot	Data tersimpan lokal pada SQLite	Berhasil
9	Sinkronisasi ulang setelah online	Data lokal terkirim ke server dan status sinkron berubah	Berhasil
10	Administrator membuka peta GIS	Marker kerusakan tampil pada peta	Berhasil
11	Administrator mengubah status laporan	Status laporan diperbarui pada basis data	Berhasil
12	Duplikasi laporan pada lokasi sama	Sistem menandai potensi duplikasi atau tetap menyimpan untuk validasi admin	Berhasil

Performa Model Deteksi YOLOv8s-TFLite

Evaluasi model menunjukkan bahwa YOLOv8s-TFLite menghasilkan rata-rata precision 69,4%, recall 66,6%, dan mAP50 67,7%. Kelas pothole memperoleh performa lebih tinggi dengan precision 75,0%, recall 75,8%, dan mAP50 76,6%. Performa ini dapat dijelaskan oleh karakteristik pothole yang relatif membentuk area terpusat, memiliki kontras visual lebih jelas, dan lebih sesuai dengan representasi bounding box. Sebaliknya, kelas crack memperoleh precision 63,8%, recall 57,4%, dan mAP50 58,8%. Nilai yang lebih rendah pada retak menunjukkan bahwa model masih

kesulitan mendeteksi objek yang tipis, memanjang, tidak beraturan, serta sering bercampur dengan tekstur aspal, bayangan, tambalan jalan, atau garis permukaan.

Tabel 5. Performa model deteksi YOLOv8s-TFLite

Kelas Objek	Precision	Recall	mAP50	Interpretasi
Pothole (Lubang)	0,750	0,758	0,766	Objek relatif terpusat dan batas visual lebih jelas, sehingga lebih mudah dilokalisasi.
Crack (Retak)	0,638	0,574	0,588	Objek tipis, panjang, dan tidak beraturan; bounding box menangkap banyak piksel latar belakang.
Rata-rata	0,694	0,666	0,677	Performa cukup untuk purwarupa pelaporan awal, tetapi belum cukup kuat untuk audit teknis otomatis.

Analisis error perlu diarahkan pada dua kelompok kesalahan. False positive dapat muncul ketika tekstur aspal, bayangan pohon, bekas tambalan, marka jalan aus, atau noda permukaan menyerupai retakan. False negative dapat muncul ketika retakan terlalu tipis, kontras rendah, tertutup bayangan, terekam dengan motion blur, atau hanya terlihat sebagian. Pada pothole, kesalahan dapat terjadi ketika lubang sangat dangkal, tertutup air, atau batasnya menyatu dengan warna aspal. Dengan demikian, peningkatan model tidak cukup dilakukan hanya dengan menambah epoch, tetapi perlu memperluas data lokal pada kondisi ekstrem, memperbaiki anotasi, menambahkan contoh negatif yang menyerupai kerusakan, dan mempertimbangkan pendekatan instance segmentation untuk crack.

Evaluasi Deployment Mobile dan Pembatasan Klaim Real-Time

Karena judul dan tujuan penelitian menekankan implementasi pada aplikasi mobile, evaluasi deployment perlu dibedakan dari evaluasi akurasi model. Precision, recall, dan mAP50 menjelaskan kualitas deteksi, sedangkan latency, FPS, waktu pemuatan model, penggunaan memori, konsumsi baterai, dan suhu perangkat menjelaskan kelayakan implementasi mobile. Pada naskah awal, metrik runtime belum dilaporkan sehingga klaim real-time harus diperhalus. Oleh karena itu, sistem lebih tepat disebut sebagai purwarupa deteksi dan pelaporan berbasis mobile yang menjalankan inferensi on-device; klaim real-time secara operasional baru dapat dinyatakan setelah pengukuran FPS dan latency dilakukan pada perangkat target. Jika hasil pengukuran menunjukkan FPS rendah atau latency tinggi, sistem tetap dapat digunakan sebagai deteksi berbasis capture, bukan deteksi video real-time berkelanjutan. Pembedaan ini penting agar kontribusi penelitian tidak overclaim. Dalam aplikasi lapangan, mode capture-based detection sering lebih realistis karena pengguna dapat mengarahkan kamera, menstabilkan perangkat, mengambil foto, dan memvalidasi hasil sebelum mengirim laporan. Dengan demikian, manfaat sistem tetap kuat walaupun klaim real-time penuh belum dibuktikan.

Perbandingan dengan Model dan Pendekatan Pembanding

Tanpa baseline comparison, efektivitas YOLOv8s sulit dinilai secara relatif. Karena itu, naskah revisi menambahkan posisi pembanding yang perlu diuji pada pengembangan berikutnya. YOLOv8s dipilih karena menawarkan keseimbangan antara akurasi dan efisiensi, tetapi model lain seperti YOLOv5n, YOLOv8n, SSD-MobileNet, EfficientDet-Lite, dan YOLOv8-seg relevan sebagai baseline. SSD-MobileNet dapat menjadi pembanding ringan untuk mobile, YOLOv8n dapat menjadi pembanding yang lebih kecil, sedangkan YOLOv8-seg atau pendekatan segmentasi lebih sesuai untuk crack karena retakan tidak selalu cocok direpresentasikan sebagai kotak persegi panjang.

Tabel 6. Baseline comparison yang direkomendasikan

Model/Pendekatan	Alasan Pembanding	Metrik yang Perlu Dibandingkan
YOLOv8s-TFLite	Model utama penelitian; kompromi akurasi dan efisiensi	mAP50, mAP50-95, latency, FPS, ukuran model
YOLOv8n-TFLite	Varian lebih ringan untuk perangkat terbatas	Penurunan/kenaikan FPS dan dampak terhadap mAP
YOLOv5n/YOLOv5s	Baseline YOLO yang banyak digunakan	Perbandingan performa terhadap YOLOv8 pada dataset sama
SSD-MobileNet	Baseline mobile object detection yang ringan	Ukuran model, latency, dan akurasi pada pothole/crack
YOLOv8-seg atau segmentation model	Lebih sesuai untuk objek crack yang tipis dan tidak beraturan	IoU area crack, precision, recall, dan error boundary

Pembahasan Kritis dan Implikasi

Secara substantif, penelitian ini menunjukkan bahwa integrasi object detection dan pelaporan berbasis lokasi dapat mengurangi kelemahan inspeksi manual. Sistem tidak hanya menghasilkan label kerusakan, tetapi juga menyediakan foto, koordinat, waktu pelaporan, status sinkronisasi, dan status validasi. Dibandingkan sistem pengaduan berbasis input manual, integrasi AI membuat laporan lebih objektif karena jenis kerusakan diidentifikasi oleh model sebelum diverifikasi pengguna dan administrator. Dibandingkan model CNN yang berat, penggunaan YOLOv8s-TFLite memberikan kompromi yang lebih realistis untuk perangkat mobile karena inferensi dapat dilakukan secara lokal dan data tetap dapat disinkronkan ke server.

Namun, nilai mAP50 rata-rata 67,7% menunjukkan bahwa sistem masih berada pada tahap purwarupa. Kinerja tersebut cukup untuk mendukung pelaporan awal, tetapi belum cukup untuk digunakan sebagai dasar tunggal audit teknis atau prioritas anggaran. Performa crack yang lebih rendah menegaskan bahwa pendekatan object detection berbasis bounding box memiliki keterbatasan untuk objek tipis dan tidak beraturan. Dari perspektif computer vision, masalah crack detection bukan hanya klasifikasi objek, tetapi juga lokalisasi area kerusakan yang sering membutuhkan segmentasi piksel, peningkatan kontras, atau strategi anotasi lebih detail. Oleh karena

itu, hasil penelitian harus dibaca sebagai bukti kelayakan integrasi sistem, bukan bukti bahwa model telah siap untuk seluruh kondisi jalan.

Keterbatasan utama penelitian berada pada detail dataset, variasi uji lapangan, dan evaluasi mobile deployment. Dataset hibrida memang meningkatkan variasi, tetapi jumlah dan keragaman data primer lokal perlu diperluas agar model lebih tahan terhadap aspal basah, pencahayaan rendah, bayangan kendaraan, jalan berdebu, kamera bergerak, sudut ekstrem, dan kualitas kamera berbeda. Evaluasi runtime juga perlu dilakukan pada beberapa kelas smartphone karena performa TensorFlow Lite sangat dipengaruhi oleh chipset, memori, akselerator perangkat, dan optimasi model. Dengan memperbaiki aspek tersebut, penelitian dapat bergerak dari purwarupa aplikatif menuju sistem pendukung keputusan yang lebih andal.

KESIMPULAN

Penelitian ini berhasil merancang dan mengimplementasikan purwarupa aplikasi mobile untuk deteksi dan pelaporan kerusakan jalan berbasis lokasi dengan mengintegrasikan YOLOv8s, TensorFlow Lite, Flutter, GPS, SQLite, Laravel, MySQL, dan dashboard GIS. Secara fungsional, sistem mampu menjalankan alur utama mulai dari autentikasi, deteksi kamera, pengambilan koordinat, validasi pengguna, pengiriman laporan, penyimpanan offline pada kondisi blank spot, pemetaan geospasial, hingga validasi administrator. Evaluasi model menunjukkan rata-rata precision 69,4%, recall 66,6%, dan mAP50 67,7%, dengan performa pothole lebih tinggi daripada crack. Perbedaan ini menunjukkan bahwa lubang jalan lebih mudah direpresentasikan melalui bounding box, sedangkan retak memerlukan pendekatan yang lebih presisi karena bentuknya tipis dan tidak beraturan. Kontribusi utama penelitian ini adalah rancangan end-to-end yang menghubungkan edge detection pada smartphone, offline synchronization, server pelaporan, GIS, dan validasi manusia. Namun, sistem belum dapat diposisikan sebagai instrumen tunggal untuk audit teknis infrastruktur karena detail dataset, baseline comparison, robustness testing, dan evaluasi runtime mobile masih perlu diperkuat. Penelitian selanjutnya perlu memperluas dataset lokal, melaporkan spesifikasi hardware, mengukur latency dan FPS, membandingkan model ringan lain, serta mengevaluasi pendekatan instance segmentation untuk meningkatkan deteksi retakan.

REKOMENDASI

Pengembangan berikutnya direkomendasikan untuk memperbesar proporsi data primer lokal, terutama pada kondisi aspal basah, pencahayaan rendah, bayangan kendaraan, sudut kamera berbeda, variasi jenis permukaan jalan, dan skenario kendaraan bergerak. Model perlu dibandingkan dengan YOLOv8n, YOLOv5, SSD-MobileNet, EfficientDet-Lite, dan pendekatan segmentasi agar efektivitas YOLOv8s-TFLite dapat dinilai secara lebih objektif. Uji lapangan juga perlu memasukkan metrik performa perangkat seperti FPS, latensi inferensi, waktu loading model, ukuran model, penggunaan RAM, konsumsi baterai, suhu perangkat, dan keberhasilan sinkronisasi offline-online. Dari sisi sistem, pengujian perlu diperluas pada skenario error, pemulihan koneksi, keamanan data pengguna, duplikasi laporan, dan usability agar aplikasi tidak hanya berhasil secara fungsional, tetapi juga layak digunakan pada konteks operasional.

ACKNOWLEDGMENT

Penulis menyampaikan terima kasih kepada Program Studi Informatika, Departemen Teknik Elektronika, Fakultas Teknik, Universitas Negeri Padang, serta seluruh pihak yang mendukung proses penyusunan, pengembangan, dan pengujian sistem.

REFERENSI

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaria, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 8, 53. https://doi.org/10.1186/s40537-021-00444-8

Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., & Pandya, S. (2021). CNN variants for computer vision: History, architecture, application, challenges and future scope. Electronics, 10(20), 2470.

Cendekia, A. I., Kharisma, A. P., & Priyambadha, B. (2025). Analisis perbandingan kinerja antara native Android Kotlin dengan framework Flutter pada aplikasi informasi rumah sakit. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 9(5), 1-8.

Chai, J., Zeng, H., Li, A., & Ngai, E. W. T. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. Machine Learning with Applications, 6, 100134. https://doi.org/10.1016/j.mlwa.2021.100134

Dharma, A. S., Nadya, C., Pardosi, S., & Silaen, Z. P. (2025). Comparative performance of YOLOv8 and SSD-MobileNet algorithms for road damage detection in mobile applications. Jurnal Ilmiah Teknik Elektro Komputer dan Informatika, 9(3), 1159-1169.

Fachri, B., & Rizal, C. (2024). Penerapan metode Waterfall dalam perancangan sistem informasi Merdeka Belajar Kampus Merdeka berbasis web. Jurnal Komputer Teknologi Informasi dan Sistem Informasi, 2(3), 591-597.

Falasyfa, R. S., & Avianto, D. (2024). Perancangan aplikasi layanan pengaduan kerusakan jalan berbasis Android. Jurnal Indonesia: Manajemen Informatika dan Komunikasi, 5(1), 944-953.

Gabriel, S., Kenrick, V., & Irsyad, H. (2024). Implementasi deteksi objek pada jalan rusak menggunakan metode YOLO. Bulletin of Information Technology, 3(1), 1-9. https://doi.org/10.58369/biit.v2i3.76

Hafidatur, N., Faid, M., & Novia, C. (2021). Sistem informasi geografis pemetaan kerusakan jalan berbasis web dan Android. Jurnal Informatika dan Sistem Informasi, 8(4).

Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A review of YOLO algorithm developments. Procedia Computer Science, 199, 1066-1073. https://doi.org/10.1016/j.procs.2022.01.135

Ma, N., Fan, J., Wang, W., Wu, J., Jiang, Y., Xie, L., & Fan, R. (2022). Computer vision for road imaging and pothole detection: A state-of-the-art review of systems and

algorithms. Transportation Safety and Environment, 4(4), tdac026. https://doi.org/10.1093/tse/tdac026

Mumuni, A., & Mumuni, F. (2022). Data augmentation: A comprehensive survey of modern approaches. Array, 16, 100258. https://doi.org/10.1016/j.array.2022.100258

Nova, L., & Rianto, Y. (2025). Real-time road damage detection on mobile devices using TensorFlow Lite and Teachable Machine. MALCOM: Indonesian Journal of Machine Learning and Computer Science, 5(3), 788-796. https://doi.org/10.57152/malcom.v5i3.1933

Padilla, R., Passos, W. L., Netto, S. L., & da Silva, E. A. B. (2021). A comparative analysis of object detection metrics with a companion open-source toolkit. Electronics, 10(3), 279. https://doi.org/10.3390/electronics10030279

Pardede, J. P., & Putranto, L. S. (2024). Pengaruh kendaraan overload terhadap umur rencana jalan. JMTS: Jurnal Mitra Teknik Sipil, 7(3), 819-830.

Rozi, M. F., Siregar, H., & Hambali, Y. A. (2025). Rancang bangun sistem manajemen akademik mahasiswa berbasis mobile multiplatform menggunakan Flutter. Jurnal Teknologi dan Sistem Informasi, 4(2), 459-468.

Safyari, Y., Mahdianpari, M., & Shiri, H. (2024). A review of vision-based pothole detection methods using computer vision and machine learning. Sensors, 24(17), 5652. https://doi.org/10.3390/s24175652

Saparudin, H., Ramdi, M. A., & Alfarizi, F. S. (2025). Implementasi Convolutional Neural Network dan ResNet-18 untuk klasifikasi kerusakan jalan berbasis citra. Jurnal Aplikasi Informatika dan Multimedia, 1(1), 27-31.

Sarker, I. H. (2021). Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. SN Computer Science, 2, 420. https://doi.org/10.1007/s42979-021-00815-1

Sasmito, B., Setiadji, B. H., & Isnanto, R. (2023). Deteksi kerusakan jalan menggunakan pengolahan citra deep learning di Kota Semarang. Teknik, 44(1), 7-14. https://doi.org/10.14710/teknik.v44i1.51908

Seandrio, A. L., Pratomo, A. H., & Yanu, M. (2021). Implementation of Convolutional Neural Network in facial expression recognition. Telematika, 18(2), 211-221. https://doi.org/10.31515/telematika.v18i2.4823

Terven, J., & Cordova-Esparza, D. M. (2023). A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction, 5(4), 1680-1716.

Wahid, A. A. (2020). Analisis metode Waterfall untuk pengembangan sistem informasi. Jurnal Ilmu-Ilmu Informatika dan Manajemen STMIK, 1-5.

Yang, J., Tian, R., Zhou, Z., Tan, X., & He, P. (2025). Flexi-YOLO: A lightweight method for road crack detection in complex environments. PLOS ONE, 20(6), e0325993. https://doi.org/10.1371/journal.pone.0325993