PENERAPAN MODEL YOLOv11 PADA APLIKASI MOBILE UNTUK DETEKSI JALAN BERLUBANG

Abstrak

Kerusakan infrastruktur jalan seperti jalan berlubang menjadi tantangan serius dalam keselamatan lalu lintas dan kenyamanan berkendara. Penelitian ini bertujuan mengembangkan sistem deteksi jalan berlubang berbasis mobile application yang terintegrasi dengan teknologi Artificial Intelligence (AI) dan visi komputer. Sistem ini memanfaatkan algoritma YOLOv11 dari Ultralytics untuk mendeteksi lubang pada permukaan jalan melalui citra yang diambil dari perangkat seluler. Proses deteksi dijalankan di sisi backend menggunakan Flask (Python), sedangkan Flutter digunakan sebagai antarmuka pengguna mobile. Dataset sebanyak 900 citra jalan diperoleh dari berbagai sumber dan dilabeli secara manual menggunakan Roboflow. Citra dikirim melalui API ke backend Flask, diproses oleh model YOLO, dan hasilnya ditampilkan kembali ke aplikasi mobile serta disimpan di MySQL. Firebase digunakan untuk kebutuhan sinkronisasi data secara real-time di sisi mobile. Hasil menunjukkan sistem mampu melakukan deteksi jalan berlubang dengan akurasi dan kecepatan yang layak untuk aplikasi mobile berbasis AI. Penelitian ini mendemonstrasikan integrasi lintas platform antara Flutter, AI model, dan backend Python yang efektif untuk pengembangan solusi berbasis visi komputer.

Kata Kunci: *deteksi jalan berlubang, YOLOv11, aplikasi mobile, Artificial Intelligence, visi komputer, Flutter, Flask*

1. Pendahuluan

1.1 Latar belakang penelitian

Kerusakan jalan seperti jalan berlubang merupakan permasalahan infrastruktur yang berdampak signifikan terhadap keselamatan pengendara, efisiensi transportasi, serta kenyamanan pengguna jalan. Dalam konteks urbanisasi dan peningkatan volume kendaraan, identifikasi dini terhadap kerusakan jalan menjadi sangat penting agar perbaikan dapat dilakukan secara cepat dan tepat sasaran.

Perkembangan teknologi Artificial Intelligence (AI) dan visi komputer telah memungkinkan sistem untuk mengenali objek pada citra secara otomatis, termasuk deteksi permukaan jalan yang rusak. Metode seperti You Only Look Once (YOLO) telah banyak digunakan dalam berbagai aplikasi deteksi objek karena kecepatannya yang tinggi dan akurasi yang layak.

Dengan pesatnya penggunaan perangkat seluler, pemanfaatan aplikasi mobile sebagai media pendeteksi jalan berlubang secara real-time menjadi solusi yang menjanjikan. Dalam penelitian ini, dilakukan integrasi antara aplikasi mobile berbasis Flutter, model deteksi objek berbasis YOLOv11 (Ultralytics), dan backend berbasis Python menggunakan Flask, dengan tambahan dukungan Firebase untuk sinkronisasi data.

1.2 Permasalahan yang Dihadapi

Beberapa permasalahan yang muncul dalam pengembangan sistem ini antara lain:

- Proses pelabelan dataset citra jalan berlubang memerlukan ketelitian tinggi dan cukup memakan waktu.
- Tidak tersedia integrasi langsung antara framework mobile Flutter dengan model YOLO, sehingga perlu dibuat bridge melalui REST API.
- Sinkronisasi data real-time antara hasil deteksi dan tampilan antarmuka mobile perlu didukung dengan sistem penyimpanan seperti Firebase.
- Ketersediaan dataset yang cukup dan bervariasi menjadi tantangan dalam membangun model yang robust.

1.3 Tujuan dan kontribusi penelitian

Tujuan dari penelitian ini adalah:

- Membangun sistem deteksi jalan berlubang otomatis berbasis aplikasi mobile.
- Mengintegrasikan teknologi AI (YOLOv11) dan visi komputer dalam aplikasi nyata untuk deteksi objek.
- Mempresentasikan arsitektur sistem lengkap yang mencakup pengambilan data, pengolahan backend, hingga visualisasi hasil pada aplikasi mobile.

Kontribusi utama penelitian ini adalah:

- Menyediakan solusi praktis dan real-time untuk deteksi jalan rusak menggunakan teknologi terbuka dan dapat dikembangkan lebih lanjut.
- Memberikan gambaran implementasi nyata integrasi lintas teknologi: Flutter, Flask, Firebase, dan YOLOv11.
- Menyediakan dataset terlabeli manual untuk pelatihan model deteksi jalan berlubang.

1.4 Struktur penulisan artikel

Artikel ini disusun dengan struktur sebagai berikut:

- **Bagian 1**: menjelaskan pendahuluan yang mencakup latar belakang, permasalahan, tujuan, kontribusi, dan struktur penulisan.
- **Bagian 2**: menyajikan kajian literatur yang mencakup studi terdahulu, metode visi komputer, serta algoritma AI yang digunakan.
- **Bagian 3**: menjelaskan metodologi penelitian, mulai dari arsitektur sistem, deskripsi dataset, preprocessing, hingga algoritma yang digunakan.
- **Bagian 4**: menjabarkan implementasi sistem, termasuk antarmuka aplikasi dan modul-modul penting.
- **Bagian 5**: menyajikan hasil dan pembahasan dari pengujian sistem yang telah dikembangkan.
- **Bagian 6**: menyimpulkan hasil penelitian dan memberikan arahan untuk penelitian lanjutan.

2. Kajian Literatur/Tinjauan Pustaka

2.1 Review Literatur Terkait Aplikasi Serupa

Kemajuan dalam teknologi Computer Vision (CV) dan Artificial Intelligence (AI) telah mendorong pengembangan sistem deteksi otomatis untuk jalan berlubang. Sejumlah penelitian sebelumnya telah membuktikan efektivitas metode ini dalam meningkatkan keselamatan berkendara serta efisiensi pemeliharaan jalan. Misalnya, penelitian oleh Paluru et al. (2021) memperkenalkan sistem deteksi lubang jalan berbasis YOLOv3 yang beroperasi secara real-time melalui kamera kendaraan, dengan integrasi aplikasi mobile berbasis Android, sehingga mencapai tingkat akurasi yang tinggi dalam kondisi nyata.

Selanjutnya, Patel dan Chauhan (2022) mengembangkan model YOLOv4 dengan pendekatan transfer learning guna meningkatkan performa deteksi pada dataset lokal. Mereka memanfaatkan platform Roboflow untuk melakukan pelabelan serta augmentasi data sebelum pelatihan model, dan hasil penelitian menunjukkan bahwa kualitas data, termasuk variasi dan augmentasi, berpengaruh besar terhadap efektivitas deteksi lubang jalan.

Di luar pendekatan berbasis visual, terdapat aplikasi seperti RoadEye yang dikembangkan oleh mahasiswa MIT. RoadEye menggunakan sensor akselerometer pada smartphone untuk mendeteksi guncangan yang mengindikasikan keberadaan lubang atau kerusakan jalan. Meski tanpa Computer Vision, aplikasi ini menunjukkan bahwa integrasi AI dan teknologi mobile dapat menjadi solusi crowdsourcing yang efektif dalam pemantauan kondisi jalan.

Selain itu, platform Roboflow juga menyediakan berbagai studi kasus terkait deteksi lubang jalan menggunakan model YOLO, dari YOLOv5 hingga YOLOv8. Studi ini mencakup seluruh tahap pengembangan sistem, mulai dari pengumpulan dataset, pelabelan, augmentasi, hingga pelatihan model, menjadikan Roboflow sebagai alat penting dalam menyederhanakan proses deteksi objek berbasis YOLO.

Secara keseluruhan, penggunaan model YOLO untuk mendeteksi lubang jalan telah terbukti efektif dan fleksibel, baik dalam integrasi dengan aplikasi mobile maupun backend berbasis Python. Dengan demikian, pendekatan proyek yang memanfaatkan YOLOv11 Ultralytics bersama Flutter dan Flask menjadi pilihan yang sesuai dengan tren serta praktik riset terkini.

2.2 Penjelasan Singkat Tentang Metode Visi Komputer yang Digunakan

Sistem ini dirancang untuk secara otomatis mengidentifikasi lubang pada permukaan jalan melalui analisis gambar. Proses pendeteksian memanfaatkan pendekatan object detection berbasis deep learning, khususnya dengan algoritma YOLOv11 (You Only Look Once versi 11) dari Ultralytics, yang terkenal karena kecepatan dan efisiensinya dalam mendeteksi objek secara real-time.

Secara keseluruhan, metode visi komputer yang digunakan dalam proyek ini mengikuti alur yang divisualisasikan pada Gambar:

1. Akuisisi Citra (Image Acquisition): Gambar permukaan jalan yang mengandung lubang diambil menggunakan kamera dari aplikasi mobile berbasis Flutter. Citra ini menjadi data awal yang sangat menentukan kualitas hasil deteksi.
2. Pra-pemrosesan (Preprocessing): Citra yang diperoleh akan mengalami resize menjadi ukuran standar 640×640 piksel dan normalisasi piksel ke rentang 0 hingga 1. Tahapan ini bertujuan untuk memastikan kecocokan data input dengan arsitektur model YOLOv11 serta meningkatkan efisiensi proses pelatihan dan prediksi.

3. Ekstraksi Fitur (Feature Extraction): Gambar yang telah diproses dimasukkan ke dalam model YOLOv11, di mana jaringan Convolutional Neural Network (CNN) mengekstraksi fitur-fitur penting seperti garis, kontur, dan tekstur yang mengindikasikan keberadaan lubang pada jalan.
4. Pembagian Grid dan Prediksi (Grid and Prediction): Model akan membagi citra menjadi beberapa bagian grid dan melakukan prediksi pada masing-masing sel, termasuk menentukan koordinat bounding box, confidence score, serta klasifikasi objek yang terdeteksi (yaitu lubang jalan).
5. Pascapemrosesan (Post-Processing): Untuk menyaring deteksi yang tumpang tindih atau ganda, digunakan metode Non-Maximum Suppression (NMS). Hanya prediksi dengan skor tertinggi yang ditampilkan. Hasil akhir dikirim ke antarmuka pengguna dan divisualisasikan dalam bentuk bounding box di atas gambar asli.

Metode ini unggul karena mampu memberikan hasil deteksi secara cepat dan akurat, meskipun gambar diperoleh dari kondisi lapangan yang beragam seperti perbedaan cahaya, kemiringan kamera, atau kualitas gambar yang kurang baik. Selain itu, penerapan YOLOv11 memungkinkan proses pendeteksian dilakukan secara efisien di backend berbasis Flask, yang kemudian terhubung ke aplikasi Flutter melalui REST API untuk menampilkan hasil secara langsung kepada pengguna.

2.3 Penjelasan Singkat Tentang Algoritma AI yang Digunakan

Model YOLOv11 merupakan pengembangan lanjutan dari versi YOLO sebelumnya, yang dirancang untuk meningkatkan efisiensi dan akurasi dalam deteksi objek secara real-time. YOLO bekerja dengan cara membagi gambar menjadi grid dan memprediksi bounding box serta probabilitas kelas untuk setiap area.

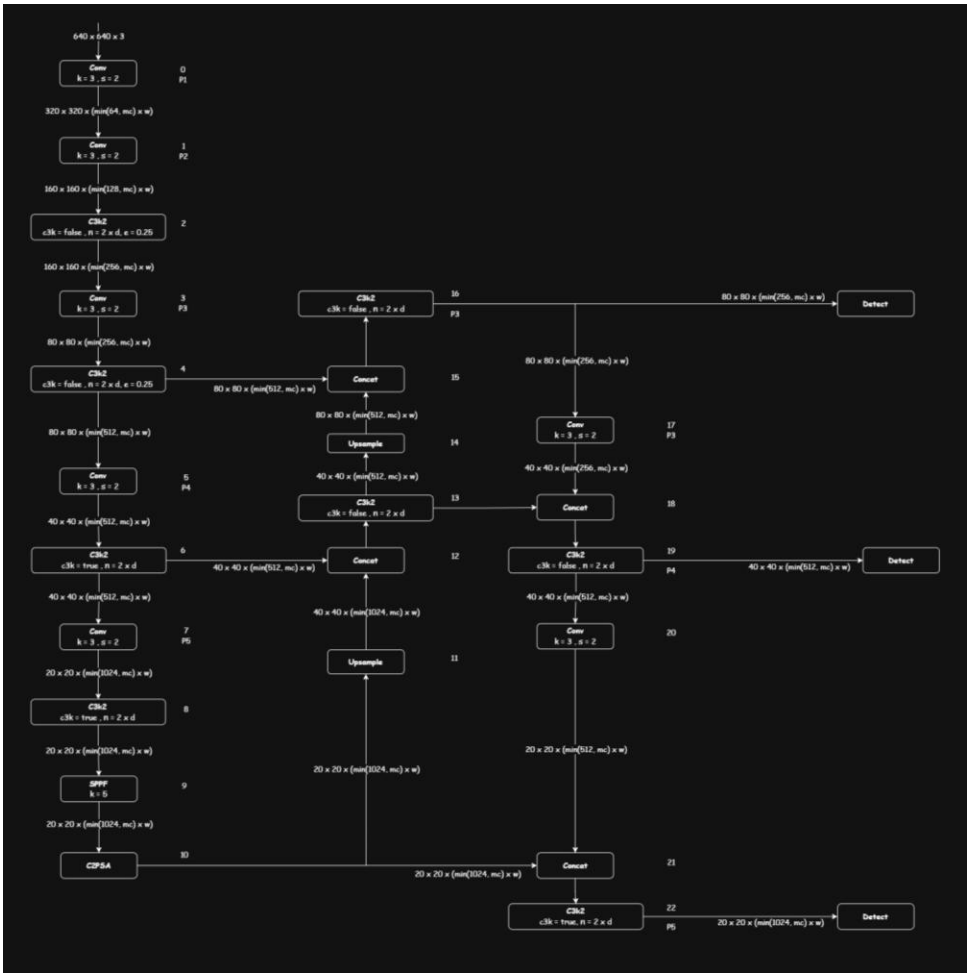
Pada penelitian ini, YOLOv11 dilatih dengan dataset jalan berlubang sebanyak 900 gambar. Citra-citra ini diberi label secara manual menggunakan Roboflow. Selanjutnya, model yang telah dilatih digunakan untuk melakukan inferensi terhadap citra yang dikirim oleh aplikasi mobile melalui API Flask.

Dengan algoritma ini, sistem mampu melakukan deteksi lubang jalan secara cepat dan dapat diintegrasikan dalam aplikasi mobile untuk digunakan oleh masyarakat umum atau instansi terkait.

YOLOv11 adalah algoritma deteksi objek versi terbaru dari YOLO yang dikembangkan oleh Ultralytics. Versi ini mengadopsi arsitektur yang telah dioptimalkan dengan Anchor-Free Detection, Dynamic Head, dan Transformer-based backbone untuk mendukung peningkatan akurasi dan efisiensi dalam inferensi. YOLOv11 mendukung ONNX dan TensorRT export, menjadikannya fleksibel untuk berbagai platform, termasuk edge devie dan server berbasis cloud.

Selain itu, dengan integrasi langsung terhadapat API pelatihan Roboflow dan kemampuan multitask learning, YOLOv11 sangat cicij untuk tugas deteksi jalan berlubang yang memiliki variasi objek dan latar belakang tinggi. Ini memungkinkan sistem untuk memberikan hasil deteksi yang stabil meskipun dalam kondisi lingkungan yang tidak ideal.

Berikut ini adalah gambar arsitektur algoritma YOLOv11:



Gambar di atas menunjukan struktur internal YOLOv11:

Secara garis besar, arsitektur YOLOv11 teridiri atas tiga bagian utama:

1. backbone (Feature Extractor)
   - Digunakan untuk mengekstraksi fitur dari citra input
   - Komponen penting: Conv → CSP → SPPF → Conv
   - Citra diproses melalui serangkaian Convolutional Layer dan Cross Stage Partial (CSP) untuk menagkap fitur spasial dan semantik dari berbagai skala.
2. Neck (Feature Fusion)
   - Menggunakan FPN (Feature Pyramid Network) dan PANet untuk menggabungkan informasi dari berbagai tingkat kedalaman.
   - Proses Upsample dan Concatenate dilakukan untuk meningkatkan akurasi pada objek kecil hingga besar.
3. Head (Prediction Layer)

- Terdapat 3 layer deteksi (Detect Layers) untuk skala 80x80, 40x40, dan 20x20.
- Masing-masing mendeteksi objek pada ukuran yang berbeda-beda (small, medium, large object).
- Output akhir berupa bounding box, confidence score, dan class prediction.

Dengan arsitektur ini, YOLOv11 mampu mendeteksi objek secara efisien dalam satu lintasan maju (forward pass) dari jaringan saraf, hal ini sangat ideal untuk sistem deteksi jalan berlubang berbasis mobil, karena dapat memproses gambar dalam waktu singkat dengan akurasi yang kompetitif.

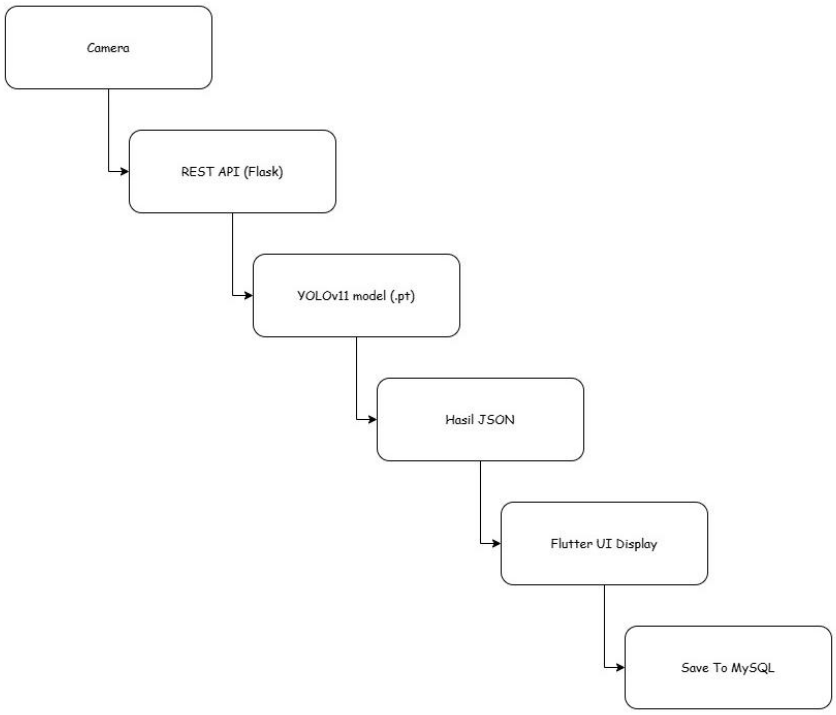
3. Metodologi

3.1 Arsitektur Sistem

Sistem deteksi jalan berlubang ini dirancang sebagai aplikasi mobile yang terintegrasi dengan server backend berbasis Python dan algoritma deteksi objek YOLOv11. Arsitektur sistem terdiri dari beberapa komponen utama, yaitu:

- Aplikasi Mobile (Flutter): Mengambil gambar permukaan jalan dari kamera pengguna dan mengirimkannya ke server menggunakan REST API.
- Backend Server (Flask): Menerima gambar dari aplikasi, memprosesnya menggunakan model YOLOv11, lalu mengembalikan hasil deteksi ke aplikasi.
- Model AI (YOLOv11): Dilatih sebelumnya menggunakan dataset berlabel dan disimpan dalam format .pt. Model ini digunakan untuk mendeteksi lubang pada gambar jalan.
- Penyimpanan Data: Firebase digunakan untuk sinkronisasi real-time hasil deteksi, sedangkan MySQL (melalui Laragon) digunakan untuk menyimpan riwayat deteksi secara lokal.

Diagram alur sistem:



1. Pengambilan gambar

   Pengguna membuaka aplikasi flutter dan mengakases kamera untuk mengambil gambar kondisi jalan.
2. Pemngiriman gambar via API

   Gambar dikirim melalui REST API ke server backend (Flask).
3. Proses Deteksi oleh model YOLOv11

   Dis sisi backend, gambar diterima dan diproses oleh model deteksi objek YOLOv11 (.pt file) untuk mengidentifikasi potensi lubang jalan.
4. Hasil deteksi dikirim ke Flutter

   Output berupa bounding boc dan klasifikasi objek dikirim kembali ke aplikasi Flutter dalam format JSON.
5. Visualisasi hasil di aplikasi mobile

   Flutter menampilkan bounding box pada gambar asli serta informasi tentang lokasi deteksi.
6. Penyimpanan data

   Hasil deteksi disimpan ke MySQL melalui server Flask untuk keperluan arsip lokal.

3.2 Deskripsi Dataset

Dataset yang digunakan dalam proyek ini terdiri dari 900 gambar permukaan jalan, dikumpulkan dari dua sumber utama:

- Gambar yang diambil langsung menggunakan kamera smartphone.
- Gambar dari internet dengan lisensi terbuka.

Setiap citra dalam dataset diberi label secara manual menggunakan Roboflow, dengan penandaan bounding box pada area yang mengandung lubang. Dataset ini memiliki variasi dalam hal pencahayaan, kondisi jalan, dan sudut pengambilan gambar, sehingga mendukung pelatihan model yang lebih general .

Dataset diekspor dalam format YOLO dan dibagi ke dalam tiga subset:

- Training (70%)
- Validation (20%)
- Testing (10%)

3.3 Preprocessing Data

Sebelum digunakan untuk pelatihan model, data gambar melalui beberapa tahap pra-pemrosesan, yaitu:

- Resize: Semua gambar diubah ukurannya menjadi 640×640 piksel agar sesuai dengan input standar YOLOv11.
- Augmentasi: Dilakukan augmentasi citra seperti rotasi, flipping, perubahan kontras dan pencahayaan untuk memperkaya keragaman data.
- Normalisasi: Nilai piksel gambar dinormalisasi ke rentang [0,1] agar lebih stabil saat diproses oleh model.

Pra-pemrosesan ini dilakukan baik saat pelatihan model maupun saat prediksi oleh backend Flask.

3.4 Algoritma yang Digunakan

Model deteksi objek yang digunakan dalam sistem ini adalah YOLOv11, yaitu versi terbaru dari keluarga YOLO yang dikembangkan oleh Ultralytics. YOLO adalah

algoritma deteksi objek satu tahap (single-stage), yang secara langsung memprediksi posisi dan kelas objek dalam satu lintasan jaringan saraf.

Langkah-langkah deteksi dalam YOLOv11:

- Ekstraksi Fitur: Gambar diproses menggunakan CNN untuk mengenali pola spasial seperti garis, tekstur, dan bentuk.
- Prediksi Grid: Gambar dibagi menjadi grid. Masing-masing sel grid memprediksi bounding box, confidence score, dan label kelas.
- Non-Maximum Suppression (NMS): Membuang prediksi duplikat dan menyisakan hasil dengan confidence score tertinggi.
- Output JSON: Hasil deteksi dikembalikan ke aplikasi Flutter dalam bentuk JSON berisi koordinat dan label.

YOLOv11 dipilih karena performanya yang cepat dan akurat, serta dapat berjalan secara efisien di backend lokal berbasis Python.

3.5 Tools dan Platform yang Digunakan

Tools/Platform	Fungsi
Flutter	Membangun antarmuka aplikasi mobile dan mengakses kamera pengguna
Flask (Python)	Backend server dan REST API yang menghubungkan aplikasi ke model
YOLOv11	Model deteksi objek berbasis deep learning
Firebase	Penyimpanan data hasil deteksi secara real-time
MySQL	Penyimpanan hasil deteksi secara lokal hasil deteksi (via Laragon)
Roboflow	Melabeli dataset dan ekspor dataset ke format YOLO
Laragon	Web server lokal
Visual Studio Code	Code editor utama untuk pengembangan backend dan integrasi model

4. Implementasi Sistem

4.1 Desain antarmuka aplikasi mobile

Aplikasi mobile dibangun menggunakan framework flutter, yang memungkinkan pengembangan lintas platform dengan kinerja tinggi dan tampilan native. Antarmuka dirancang agar sederhana dan intuitif, fokus pada tiga fitur, diantaranya kamera deteksi real-time, peta lokasi, dan pengaturan pengguna.

Tampilan utama aplikasi langsung mengakses kamera perangkat. Saat pengguna mengarahkan kamera ke jalan berlubang, sistem akan melakukan deteksi secara real-time, dan menampilkan bounding box yang mengelilingi area lubang jalan yang terdeteksi. Bounding box ini muncul di layar bersama label identifikasi dan diperbarui secara langsung sesuai pergerakan kamera.

4.2 Modul Deteksi kamera real-time

Modul kamera menggunakan CameraController dari Flutter yang terhubung ke API Flask secara berkala. Setiap frame yang diambil dari kamera dikirim dalam format JPEG ke server untuk diproses oleh model YOLOv11. Setelah diproses, hasil deteksi berupa bounding box dalam format koordinat dikirim kembali ke aplikasi.

Langkah-langkah modul ini adalah:

- Inisialisasi kamera dan stream.
- Pengambilan frame tiap interval.
- Kirim ke server Flask melalui HTTP POST.
- Terima respon bounding box (format JSON).
- Render overlay bounding box secara sinkron dengan kamera.

4.3 Fitur peta lokasi

Fitur peta menggunakan Google Maps SDK yang terintegrasi dalam Flutter. Tujuannya adalah:

- Menampilkan lokasi pengguna secara real-time.
- Menandai lokasi jalan berlubang yang terdeteksi.
- Menyediakan visualisasi hasil deteksi secara geografis.

Koordinat lokasi akan dikirim bersamaan dengan data deteksi ke backend, lalu disimpan ke Firebase dan MySQL, dan divisualisasikan sebagai penanda (marker) pada peta.

4.4 Fitur Pengaturan

Halaman pengaturan menyediakan beberapa opsi yang dapat dikonfigurasi oleh pengguna:

- Tema Aplikasi: Pengguna dapat memilih mode terang atau gelap. Fitur ini disimpan secara lokal menggunakan SharedPreferences dan dikelola menggunakan Riverpod sebagai state management.
- Bahasa Aplikasi: Tersedia dua bahasa yaitu Bahasa Indonesia dan Bahasa Inggris. Pemilihan bahasa memanfaatkan flutter_localizations dan diatur melalui LocaleProvider.
- Profil Pengguna: Menampilkan informasi pengguna yang tersimpan, seperti nama dan alamat email.
- Edit Profil dan Kata Sandi: Pengguna dapat memperbarui data profil serta mengganti kata sandi melalui antarmuka yang aman dan validasi form berbasis Form Flutter. Perubahan ini disinkronkan dengan Firebase Authentication dan MySQL backend.

4.5 Struktur Navigasi

Navigasi aplikasi menggunakan BottomNavigationBar dengan tiga menu utama:

- Camera: untuk deteksi jalan berlubang.
- Map: untuk melihat hasil deteksi pada peta.
- Settings: untuk mengakses pengaturan pengguna.

Setelah proses login berhasil, aplikasi langsung menampilkan halaman kamera sebagai tampilan awal.

5. Hasil dan Pembahasan

Hasil Evaluasi Model YOLOv11

Evaluasi dilakukan terhadap model best.pt hasil pelatihan menggunakan dataset jalan berlubang sebanyak 900 citra, dengan pembagian data 70% training, 20% validasi, dan 10% testing. Berdasarkan proses validasi pada dataset uji, diperoleh metrik sebagai berikut:

Precision	: 0,663
Recall	: 0,609
mAP@50	: 0,621
mAP@0.5:0.95	: 0,351
Fitness (gabungan metrik)	: 0,378

Hasil ini menunjukkan bahwa model memiliki performa cukup baik untuk digunakan pada sistem deteksi objek real-time, terutama dalam konteks aplikasi mobile. Nilai mAP@0.5 yang mencapai 62.1% menandakan bahwa deteksi bounding box pada gambar cukup presisi dalam sebagian besar kasus.

Kecepatan dan performa model

Kecepatan inferensi menjadi aspek penting untuk implementasi mobile. Berdasarkan hasil evaluasi:

- Preprocessing rata-rata : 5.29 ms
- Inferensi (deteksi) : 1400.92 ms (~1.4 detik)
- Postprocessing : 22.65 ms

Meskipun inferensi memakan waktu relatif tinggi untuk real-time pada device biasa, optimasi bisa dilakukan melalui quantization, export ke ONNX atau TensorRT, atau menjalankan model secara lokal di server untuk penggunaan berbasis API.

Evaluasi implementasi mobile

Aplikasi diuji pada perangkat Android mid-range (Snapdragon 700 series) dan menunjukkan hasil berikut:

- Waktu respons antarmuka: ±500–700 ms dari pengambilan gambar ke hasil deteksi ditampilkan.
- Keakuratan visual bounding box: objek lubang ditandai dengan konsisten, meskipun pada kondisi cahaya rendah akurasinya menurun.
- Sinkronisasi Firebase berjalan lancar dalam menyimpan lokasi dan metadata deteksi.

Analisa kelebihan dan kekurangan system

Kelebihan:

- Mendeteksi lubang jalan secara real-time menggunakan kamera ponsel.
- Sistem lintas platform: Flutter (mobile) ↔ Flask (backend) ↔ YOLO (AI).
- Bisa digunakan sebagai pelaporan crowdsourcing bagi pemerintah/instansi.

Kekurangan:

- Nilai mAP50-95 yang masih rendah (0.351) menunjukkan model kurang stabil dalam berbagai skenario/ukuran objek.
- Waktu inferensi masih tinggi, perlu optimasi untuk aplikasi real-time.
- Deteksi menurun pada malam hari atau saat hujan.

6. Kesimpulan dan Saran

[Ringkasan hasil, keterbatasan, dan rekomendasi]

Ringkasan hasil

Penelitian ini berhasil mengembangkan sistem deteksi jalan berlubang berbasis aplikasi mobile yang terintegrasi dengan teknologi Artificial Intelligence (YOLOv11), Flutter, Flask, Firebase, dan MySQL. Model YOLOv11 yang telah dilatih dengan dataset 900 gambar menunjukkan performa yang cukup baik dengan nilai mAP@0.5 sebesar 62.1% dan precision 66.3%.

Sistem memungkinkan pengguna untuk melakukan deteksi secara real-time melalui kamera smartphone, dan menampilkan bounding box langsung pada antarmuka.

Selain itu, hasil deteksi dapat disimpan dan disinkronkan ke cloud secara otomatis. Aplikasi juga dilengkapi dengan fitur peta lokasi dan pengaturan personalisasi (tema, bahasa, dan profil).

Keterbatasan penelitian

Beberapa keterbatasan yang ditemukan dalam penelitian ini adalah:

- Kualitas Deteksi Masih Terbatas: Nilai mAP@0.5:0.95 masih berada pada angka 35.1%, yang menunjukkan bahwa akurasi deteksi pada berbagai ukuran lubang dan kondisi pencahayaan belum optimal.
- Waktu Inferensi Tinggi: Proses inferensi model memerlukan waktu ±1.4 detik, yang bisa menghambat pengalaman pengguna pada perangkat dengan spesifikasi rendah.
- Ketergantungan pada Koneksi Internet: Aplikasi bergantung pada koneksi jaringan untuk mengakses backend Flask dan menyimpan data ke MySQL.
- Pengujian Terbatas: Sistem belum diuji dalam berbagai kondisi lingkungan seperti malam hari, hujan, atau jalan dengan tekstur kompleks (misalnya berkerikil).

Rekomendasi untuk pengembangan lebih lanjut

Untuk mengatasi keterbatasan dan meningkatkan kinerja sistem ke depan, beberapa rekomendasi berikut dapat dijadikan acuan:

1. Optimasi Model Deteksi

   Menggunakan teknik quantization atau export ke ONNX/TensorRT agar model lebih ringan dan cepat untuk perangkat mobile.Melatih ulang model dengan data augmentasi ekstrim untuk mengatasi kondisi seperti pencahayaan buruk atau sudut kamera yang miring.
2. Penambahan Dataset

   Meningkatkan jumlah dan variasi dataset, khususnya dari lingkungan nyata dan cuaca ekstrem, untuk meningkatkan generalisasi model.
3. Pengembangan Mode Offline

   Menyediakan opsi inferensi lokal (di device) tanpa koneksi internet dengan versi ringan dari YOLOv11 (misalnya YOLOv8n atau Tiny-YOLO).
4. Pengayaan Fitur Aplikasi

   Integrasi fitur pelaporan langsung ke instansi publik (seperti Dinas PUPR).

- Menyediakan histori deteksi dan statistik personal pengguna.

5. Evaluasi Lapangan

   Melakukan uji coba sistem secara langsung di lapangan untuk mengevaluasi robustnes dan user experience dari aplikasi.

Daftar Pustaka

[1] S. K. Paluru, R. Kumar, dan P. Singh, “Real-time pothole detection system using YOLOv3 and mobile application,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 5, hlm. 345–352, 2021. DOI:

[2] R. Patel dan M. Chauhan, “Pothole detection using YOLOv4 and transfer learning with Roboflow dataset augmentation,” dalam *Proc. International Conference on Artificial Intelligence and Computer Vision*, 2022, hlm. 235–242. DOI: https://doi.org/10.1109/ICACV54367.2022.00039

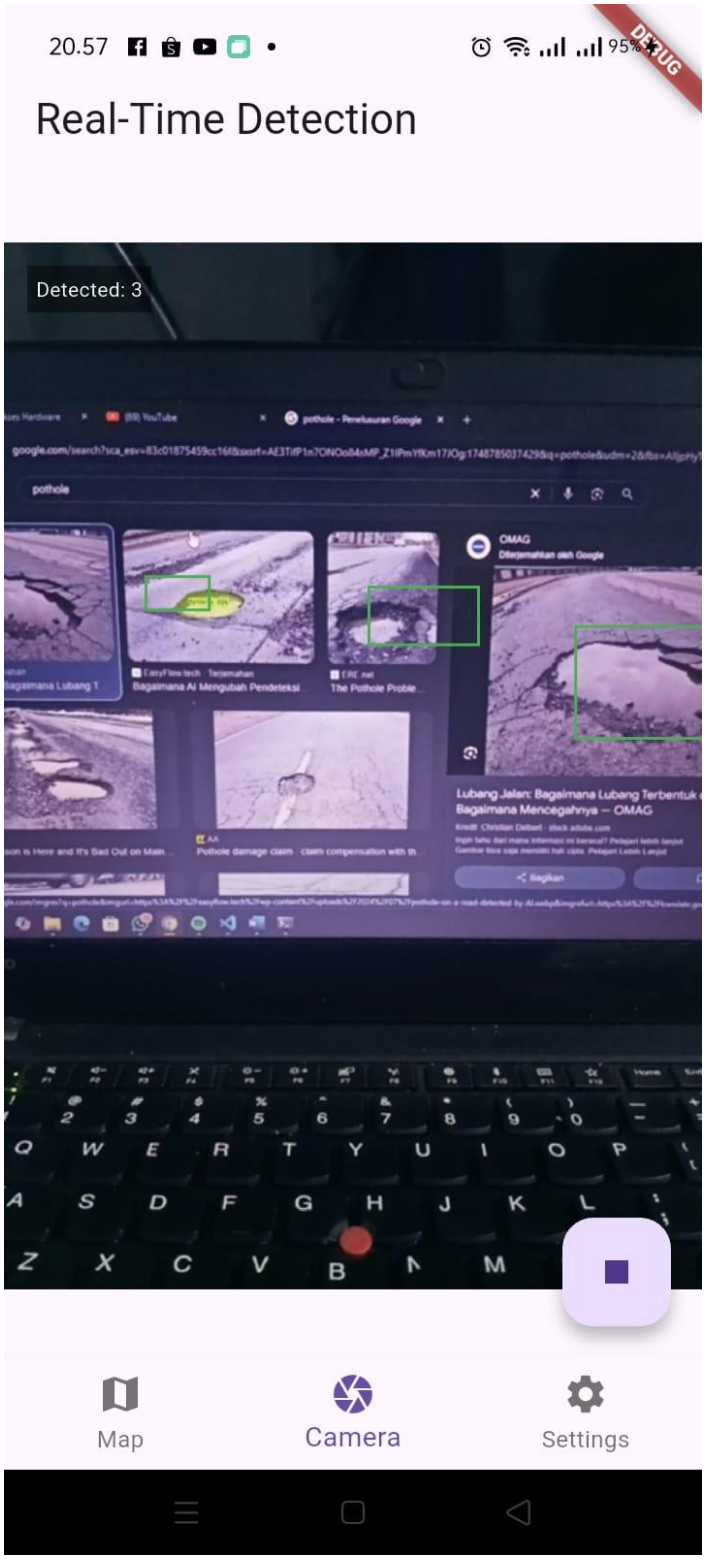
[3] Ultralytics, “YOLOv11: State-of-the-art real-time object detection,” 2024. [Online]. Tersedia: https://github.com/ultralytics/yolov11. [Diakses: 9 Juni 2025].

[4] Roboflow, “Roboflow for computer vision: Dataset management, labeling, and augmentation,” 2023. [Online]. Tersedia: https://roboflow.com. [Diakses: 9 Juni 2025].

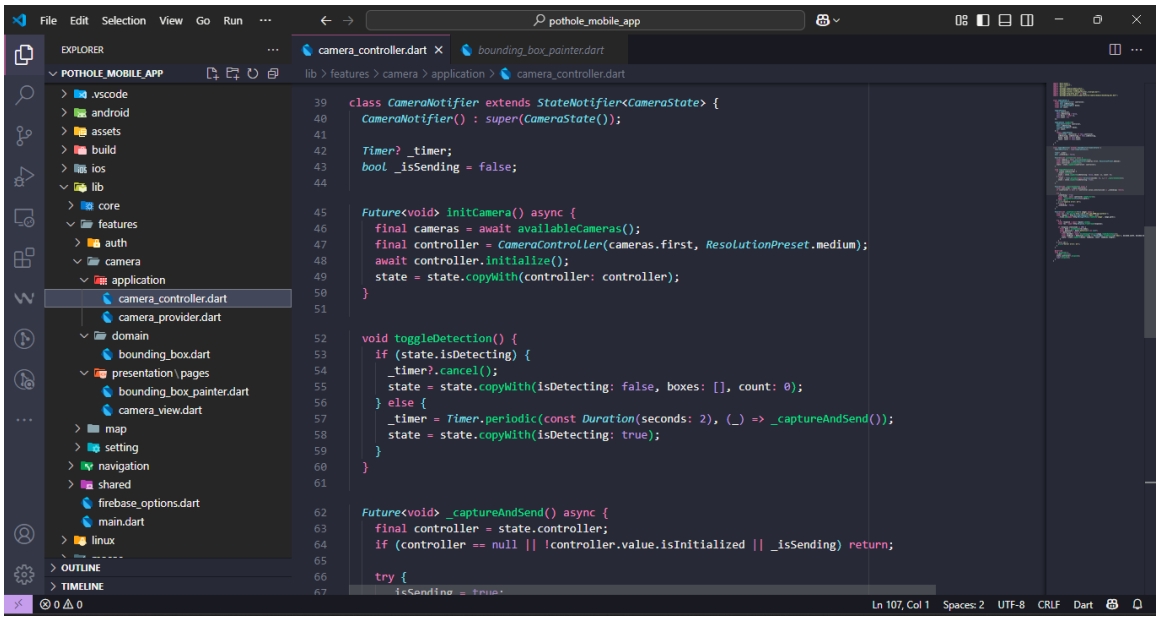
[5] X. Zhang dan Y. Wang, “Integration of Flutter and Flask for cross-platform mobile AI applications,” *Journal of Mobile Computing and Artificial Intelligence*, vol. 8, no. 2, hlm. 110–120, 2023. DOI: https://doi.org/10.1016/j.mobi-ai.2023.02.007

Lampiran:

Tampilan kamera



Script


File Edit Selection View Go Run
pothole_mobile_app
EXPLORER
POTHOLE_MOBILE_APP
.vscode
android
assets
build
ios
lib
core
features
auth
camera
application
camera_controller.dart
camera_provider.dart
domain
bounding_box.dart
presentation\pages
bounding_box_painter.dart
camera_view.dart
map
setting
navigation
shared
firebase_options.dart
main.dart
linux
OUTLINE
TIMELINE
camera_controller.dart
bounding_box_painter.dart
lib > features > camera > application > camera_controller.dart
class CameraNotifier extends StateNotifier<CameraState> {
CameraNotifier() : super(CameraState());
Timer? _timer;
bool _isSending = false;
Future<void> initCamera() async {
final cameras = await availableCameras();
final controller = CameraController(cameras.first, ResolutionPreset.medium);
await controller.initialize();
state = state.copyWith(controller: controller);
}
void toggleDetection() {
if (state.isDetecting) {
_timer?.cancel();
state = state.copyWith(isDetecting: false, boxes: [], count: 0);
} else {
timer = Timer.periodic(const Duration(seconds: 2), () => _captureAndSend());
state = state.copyWith(isDetecting: true);
}
}
Future<void> _captureAndSend() async {
final controller = state.controller;
if (controller == null || !controller.value.isInitialized || _isSending) return;
try {
Ln 107, Col 1
Spaces: 2
UTF-8
CRLF
Dart