https://doi.org/10.32815/jitika.v19i2.1184

Volume 19, Nomor 2, Tahun 2025

Jurnal Ilmiah Teknologi Informasi Asia

Halaman Situs Jurnal: https://jurnal.asia.ac.id/index.php/jitika

Artikel

Implementasi algoritma YOLOv5 pada platform Android untuk penghitungan bibit ikan lele (*clarias sp.*)

Mohammad Zainuddin *, Muhammad Saifuddin Zuhri

Teknik Informatika, Institut Teknologi Dan Bisnis Asia Malang, Kota Malang, 65113, Indonesia

***Abstrak*—Budidaya ikan lele di Indonesia menghadapi tantangan efisiensi dalam proses penghitungan bibit, yang hingga kini masih bergantung pada metode manual. Penelitian ini bertujuan mengembangkan dan mengevaluasi aplikasi *mobile* berbasis Android yang mengimplementasikan algoritma YOLOv5 untuk otomasi deteksi dan penghitungan bibit ikan lele secara *real-time*. Model dilatih menggunakan dataset citra dari Roboflow dan diintegrasikan ke dalam aplikasi yang dikembangkan dengan *framework* Flutter. Kinerja model dievaluasi secara kuantitatif menggunakan metrik *Precision*, *Recall*, dan *F1-Score* pada tiga kondisi skenario: normal, bergerombol (*occlusion*), dan bayangan. Hasil pengujian menunjukkan performa terbaik dicapai pada kondisi normal dengan *F1-Score* sebesar 0.949. Kinerja menurun pada kondisi bibit bergerombol (*F1-Score* 0.874) yang disebabkan oleh oklusi objek, serta pada kondisi bayangan (*F1-Score* 0.786) yang disebabkan oleh deteksi positif palsu. Hasil ini mengonfirmasi kelayakan YOLOv5 untuk aplikasi penghitungan bibit di perangkat *mobile*, sekaligus menyoroti area krusial untuk perbaikan, khususnya pada penanganan variasi pencahayaan dan objek yang tumpang-tindih.**

Kata kunci*—*android; bibit ikan lele; deteksi objek; pengolahan citra; yolov5.

1. Pendahuluan

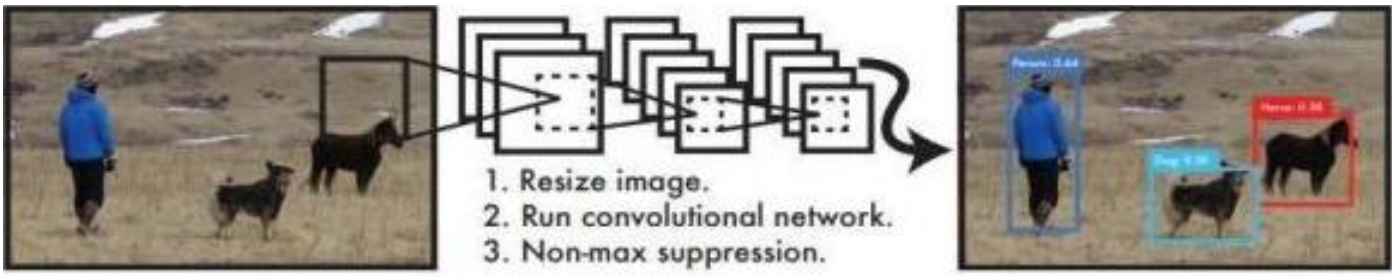
Ikan lele (*Clarias sp.*) adalah salah satu jenis ikan air tawar yang memiliki nilai ekonomi tinggi di Indonesia (Yumna et al., 2019). Selain itu, ikan lele juga memiliki beberapa keunggulan, seperti kemampuan tumbuh cepat, kemampuan beradaptasi dengan lingkungan yang buruk, serta kandungan gizi yang tinggi (Ciptawati et al., 2021).

Permintaan akan ikan lele terus meningkat seiring dengan pertumbuhan populasi dan perubahan pola konsumsi masyarakat. Permintaan ikan lele terus meningkat dari tahun ke tahun, yang berdampak pada peningkatan produksi ikan lele (Sitio et al., 2017). Di lain sisi, pertumbuhan produksi ikan lele bahkan tercatat mencapai rata-rata 11,77% per tahun antara 2013 hingga 2018, menunjukkan perannya yang signifikan dalam sektor budidaya (Fauziyah et al., 2019). Salah satu fase paling krusial dalam siklus budidaya adalah manajemen bibit, di mana proses penghitungan menjadi aktivitas fundamental untuk estimasi stok, perencanaan pakan, dan prediksi panen.

Dalam proses budidayanya, terutama saat masa pemijahan, para peternak sering dihadapkan pada permasalahan dalam menghitung jumlah benih ikan lele yang dihasilkan. Proses penghitungan jumlah benih ikan lele saat ini masih dilakukan secara manual, memakan waktu yang relatif lama. Proses perhitungan jumlah benih ikan lele memerlukan waktu 1,5 jam sampai 2 jam untuk 1 kantong plastik yang berisi sekitar 2.000

* Penulis korespondensi.
Alamat E-mail: mzein@asia.ac.id (M. Zainuddin)

Email para penulis: MZ (mzein@asia.ac.id), MSZ (sauzum73@gmail.com)
Digital Object Identifier 10.32815/jitika.v19i2.1184
Manuskrip dikirim 19 Juli 2025; direvisi 25 September 2025; diterima 26 September 2025.
ISSN: 2580-8397(O), 0852-730X(P).



Gambar 1. Contoh *bounding box*

ekor benih ikan lele dengan ukuran larva sekitar 6 mm sampai 8 mm. Ukuran bibit yang kecil dan gerakan lincah membuat penghitungan secara manual sulit dan memakan waktu, sehingga meningkatkan potensi kesalahan manusia (*human error*) (Dendi & Sunardi, 2021).

Perkembangan pesat dalam teknologi *deep learning*, khususnya di bidang visi komputer, menawarkan solusi potensial untuk mengotomatisasi dan meningkatkan akurasi proses ini. Dalam hal ini, citra merupakan salah satu bentuk informasi yang dapat dianalisis dan digunakan manusia, selain teks, suara, dan video. Karena itu, diperlukan pengolahan citra agar dapat diperoleh informasi yang diinginkan (Sulistiyanti et al., 2016). Algoritma *You Only Look Once* (YOLO) adalah salah satu model deteksi objek canggih yang merevolusi deteksi *real-time* dengan memproses gambar secara keseluruhan dalam satu kali inferensi untuk menghasilkan koordinat *bounding box* dan probabilitas kelas secara bersamaan, seperti yang diilustrasikan oleh Gambar 1. Beberapa penelitian sebelumnya telah menunjukkan keberhasilan implementasi YOLO untuk deteksi objek agrikultur (Arganata et al., 2020; Badgujar et al., 2024), termasuk penghitungan bibit ikan gurami menggunakan *webcam*. Sistem yang dikembangkan mampu mendeteksi dan menghitung bibit ikan dengan akurasi deteksi mencapai 82–85% dan akurasi perhitungan hingga 100% ketika menggunakan pendekatan modus. Namun, keterbatasan utama dari penelitian tersebut adalah ketergantungan pada platform statis (komputer), yang mengurangi kepraktisan dan mobilitasnya untuk penggunaan langsung di lingkungan tambak.

Menjawab keterbatasan tersebut, penelitian ini berfokus pada pengembangan dan implementasi model YOLOv5 yang terintegrasi dalam sebuah aplikasi *mobile* berbasis Android. Tujuannya adalah untuk menyediakan alat bantu yang portabel, cepat, dan akurat bagi para peternak. Kontribusi utama penelitian ini terletak pada evaluasi kinerja model pada berbagai kondisi lingkungan yang realistis (normal, bergerombol, dan bayangan) untuk mengukur ketangguhan dan mengidentifikasi batasan sistem dalam skenario aplikasi praktis di lapangan.

Implementasi pengolahan citra menggunakan metode YOLOv5 dalam menghitung bibit ikan lele diharapkan tidak hanya meningkatkan efisiensi, tetapi juga memberikan data yang lebih akurat dan real-time. Data ini sangat berharga bagi para petani ikan dalam mengambil keputusan yang tepat terkait dengan manajemen stok, pemberian pakan, dan perawatan kolam. Dengan demikian, teknologi ini tidak hanya membantu meningkatkan produktivitas, tetapi juga mendukung praktik budidaya yang lebih berkelanjutan dan menguntungkan.

Hasil dari penelitian ini menunjukkan bahwa aplikasi yang dikembangkan mampu mencapai kinerja deteksi yang sangat baik pada kondisi ideal (F1-Score 0.949), namun mengalami penurunan performa yang signifikan pada kondisi yang lebih kompleks. Secara spesifik, performa menurun pada kondisi bibit bergerombol (F1-Score 0.874) akibat kegagalan mendeteksi objek yang tumpang-tindih, dan menurun lebih jauh pada kondisi bayangan (F1-Score 0.786) akibat kesalahan identifikasi objek. Temuan ini memberikan bukti kuantitatif mengenai kapabilitas dan keterbatasan sistem.

Bab selanjutnya pada naskah ini disusun sebagai berikut: Bab 2 menguraikan metodologi penelitian, mencakup landasan teori, proses pelatihan model, implementasi aplikasi, dan skenario pengujian. Bab 3 menyajikan hasil pengujian secara rinci dan pembahasannya. Terakhir, Bagian 4 merangkum kesimpulan dari penelitian ini dan memberikan saran untuk pengembangan di masa depan.

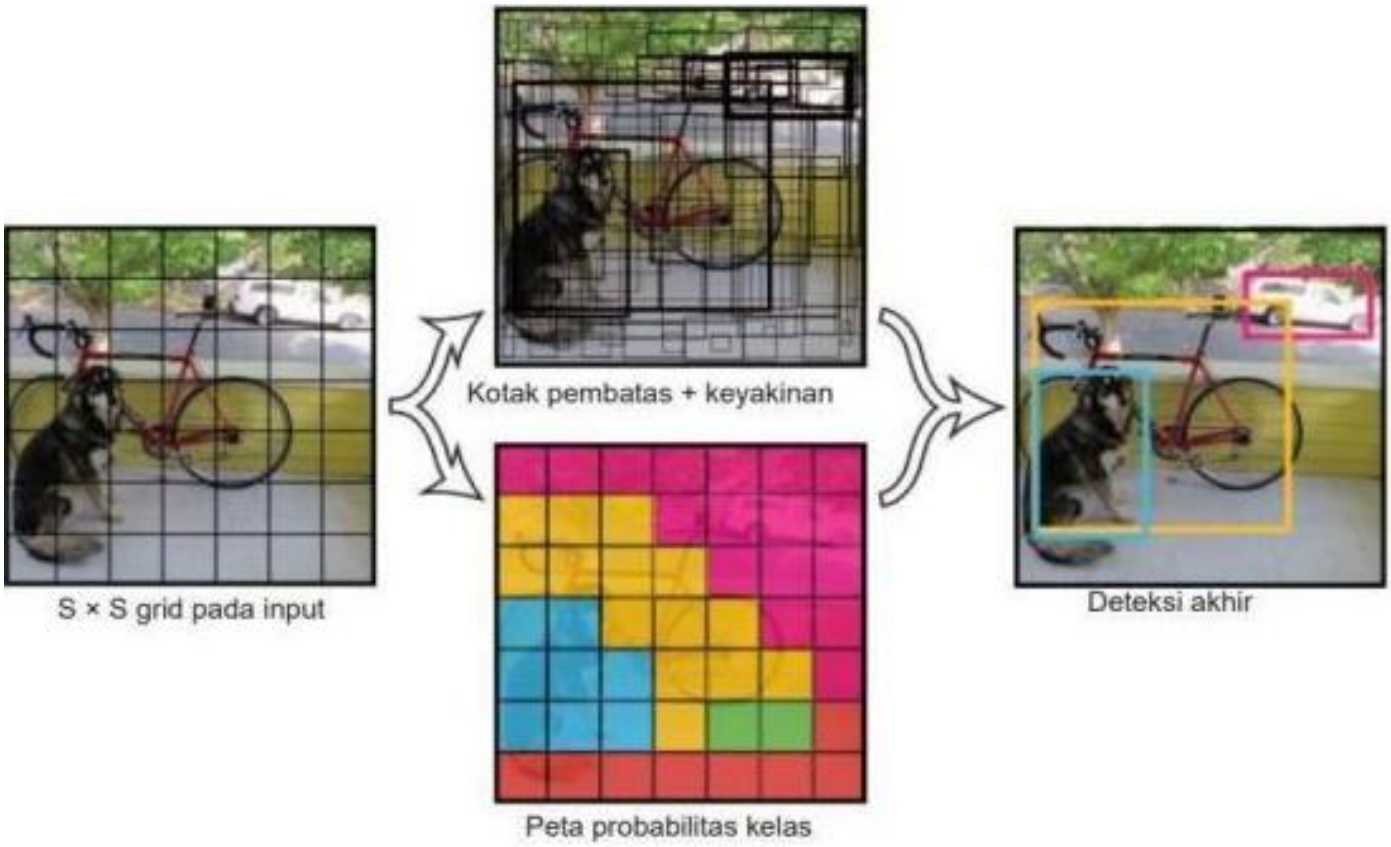
2. Metode

2.1. Landasan teori

2.1.1. YOLO

YOLO (*You Only Look Once*) adalah salah satu algoritma deteksi objek yang termasuk dalam kategori satu tahap (*one-stage*). Algoritma ini dikenal karena kemampuannya dalam memproses gambar secara *real-time* dengan kecepatan komputasi tinggi. Dalam *YOLO,* komponen-komponen deteksi objek ditempatkan dalam satu jaringan saraf tiruan. Hal ini memungkinkan pelatihan secara *end-to-end* dengan tetap menjaga tingkat presisi yang baik.

YOLO membagi gambar *input* menjadi *grid* berukuran S×S, di mana nilai S adalah 7 dan ukuran gambar *input* adalah 448 x 448, seperti pada Gambar 2. Proses pembentukan *bounding box* dilakukan melalui konvolusi terhadap gambar *input*. Hasilnya, ukuran *bounding box* menjadi S×S×(B×5 + C), di mana B adalah jumlah *bounding box* (biasanya 2) dalam satu *grid*, dan C adalah jumlah kelas yang dapat diklasifikasikan. Nilai B dikalikan dengan 5 karena setiap *bounding box* membutuhkan penyimpanan lima nilai, yaitu koordinat x, koordinat y, lebar, tinggi, dan nilai *confidence score* (probabilitas *bounding box* pada objek tertentu) (Armalivia et al., 2021). Selanjutnya, sistem menjalankan jaringan konvolusi tunggal pada gambar. Akhirnya, sistem memberikan deteksi objek berdasarkan tingkat



Gambar 2. Alur kerja YOLO untuk mendeteksi objek

Tabel 1. Spesifikasi Google Colab Pro+

No.	Komponen	Spesifikasi
1	RAM	334.6 GB
2	*Runtime*	TPU V2
3	Sistem Operasi	Linux
4	*Disk Space*	225.3 GB
5	Bahasa Pemrograman	Python 3.10.12
6	*Library*	PyTorch, OpenCV

kepercayaan sesuai dengan model yang telah dilatih.

2.1.2. Convolutional neural network (CNN)

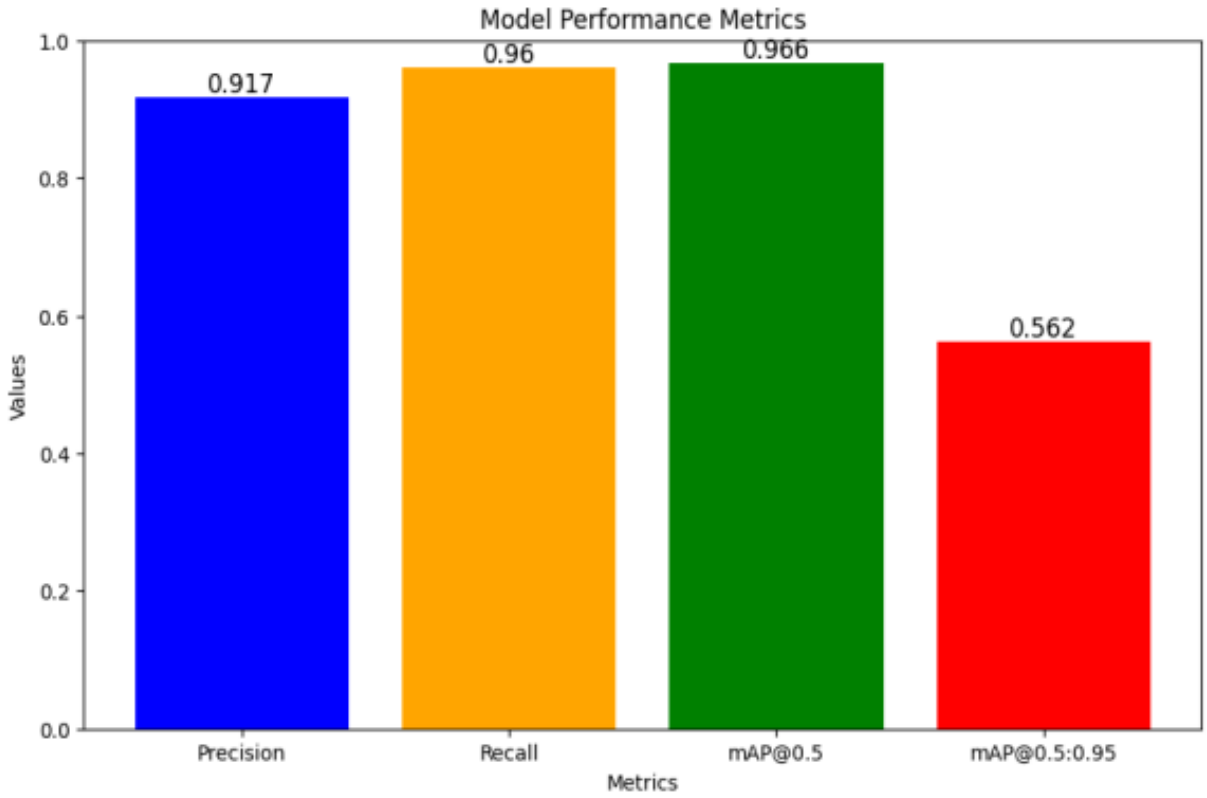
CNN adalah pengembangan lebih lanjut dari *Multilayer Perceptron* (MLP) yang dirancang khusus untuk memproses data dua dimensi. CNN termasuk dalam kategori *Deep Neural Network* karena memiliki jaringan yang sangat dalam, dan sering diterapkan pada data citra. Dalam kasus klasifikasi gambar, MLP kurang efektif karena tidak mempertahankan informasi spasial dari data gambar dan memperlakukan setiap piksel sebagai fitur independen, yang mengakibatkan hasil yang kurang memuaskan (Putra, 2016).

CNN tersusun atas susunan neuron 3 dimensi, yaitu lebar, tinggi, dan kedalaman. Lebar dan tinggi mewakili ukuran lapisan, sedangkan kedalaman menunjukkan jumlah lapisan dalam jaringan. Sebuah CNN dapat memiliki puluhan hingga ratusan lapisan, di mana masing-masing lapisan terlatih untuk mengenali fitur-fitur tertentu pada gambar. Proses pengolahan gambar diterapkan pada setiap gambar latih dengan resolusi yang berbeda. Hasil dari pengolahan setiap gambar kemudian digabungkan dan menjadi *input* untuk lapisan berikutnya. Pengolahan gambar ini dapat dimulai dari fitur yang sangat sederhana, seperti deteksi garis tepi, hingga fitur yang lebih kompleks, seperti mengenali wajah atau objek tertentu (Ilahiyah & Nilogiri, 2018).

2.2. Pelatihan dan validasi model

Implementasi sistem adalah tahap penting dalam pengembangan proyek yang memastikan semua komponen yang telah dirancang dapat berfungsi dengan baik secara keseluruhan. Proses ini mencakup beberapa langkah penting mulai dari penyiapan lingkungan pengembangan, integrasi dan pelatihan model, hingga *deployment* sistem ke lingkungan produksi. Setiap langkah memerlukan perhatian khusus untuk memastikan keberhasilan implementasi dan kelancaran operasional sistem yang dibangun.

Penyiapan lingkungan pengembangan yang optimal adalah langkah pertama dalam implementasi sistem yang sukses. Pada tahap ini, perangkat keras dan perangkat lunak yang diperlukan harus diatur dengan benar untuk mendukung proses pengembangan dan pengujian model. Untuk penelitian ini, Google Colab Pro+ digunakan sebagai lingkungan pengembangan utama, dengan spesifikasi seperti yang



Gambar 3. Hasil validasi model terbaik setelah 50 *epoch*

didetailkan pada Tabel 1. Google Colab Pro+ menawarkan sejumlah keuntungan, termasuk akses ke sumber daya komputasi yang lebih kuat, waktu penggunaan yang lebih lama, dan performa yang lebih stabil.

Proses *training* model merupakan tahap penting dalam pengembangan sistem berbasis *machine learning*. Pada tahap ini, model dilatih menggunakan data yang telah disiapkan untuk mengenali pola dan membuat prediksi yang akurat. Proses ini mencakup beberapa sub-tahap penting seperti penyiapan data *training*, pengaturan parameter *training*, dan pelaksanaan *training* itu sendiri. Selain itu, penggunaan metrik evaluasi yang tepat sangat penting dalam validasi model. Metrik seperti *mean Average Precision* (mAP), *precision*, dan *recall* yang akan digunakan untuk mengukur kinerja model deteksi objek YOLOv5 pada penelitian ini. Metrik mAP mengukur rata-rata presisi untuk berbagai tingkat *recall*, sementara *precision* dan *recall* mengukur seberapa baik model dalam mendeteksi objek yang benar tanpa menghasilkan terlalu banyak kesalahan.

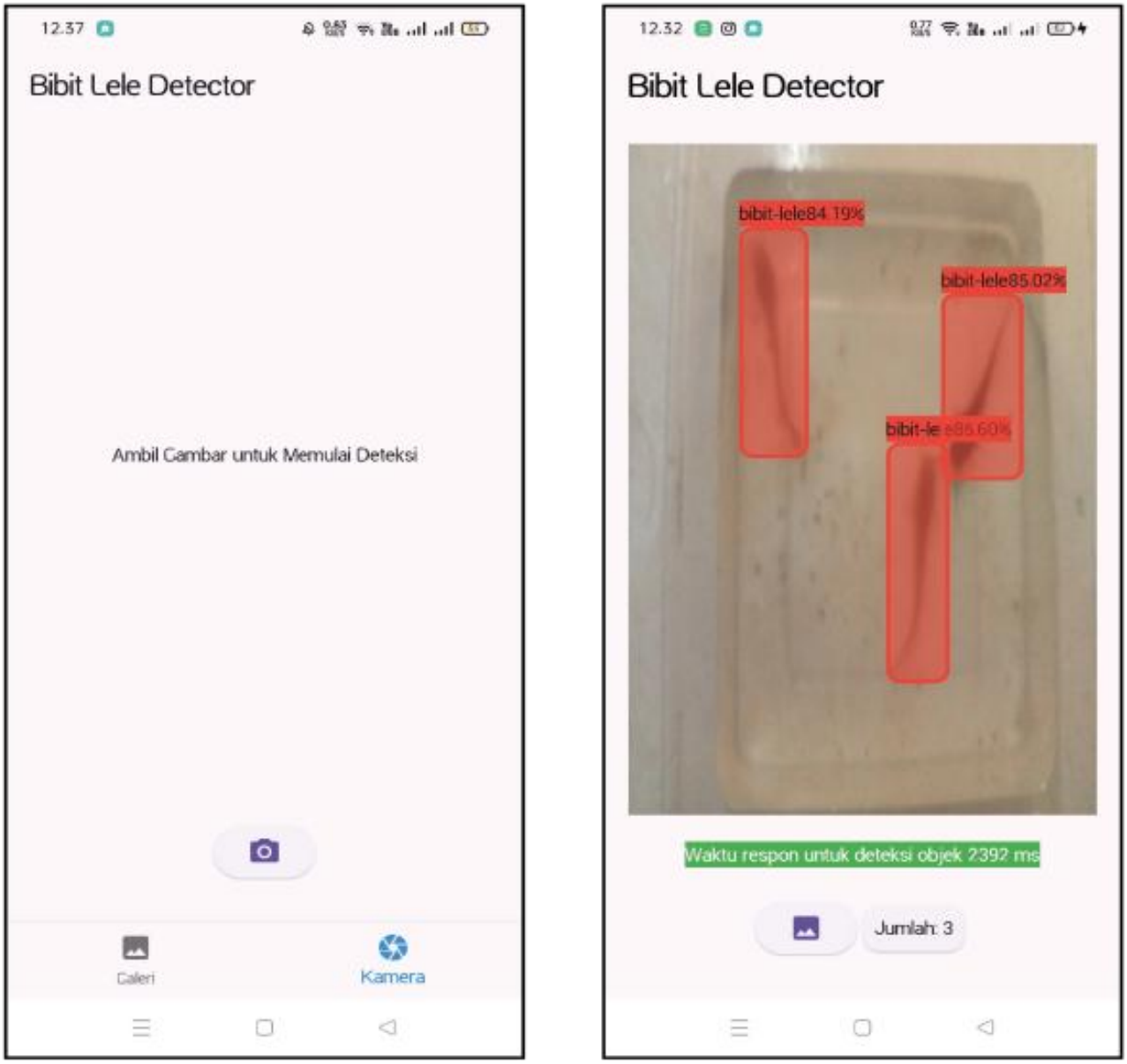
Dalam studi ini, model YOLOv5 dilatih menggunakan dataset citra bibit ikan lele yang diperoleh dari repositori publik Roboflow. Proses pelatihan dieksekusi di lingkungan komputasi awan Google Colab Pro+ selama 50 *epoch* menggunakan *library* PyTorch. Selama pelatihan, kinerja model pada set data validasi dipantau secara berkala. Gambar 3 menunjukkan metrik performa model terbaik yang dicapai, dengan nilai *precision* 0.917, *recall* 0.96, dan mAP@0.5 sebesar 0.966, mengindikasikan kemampuan generalisasi model yang sangat baik pada data yang belum pernah dilihat.

2.3. Implementasi pada platform Android

Setelah pelatihan selesai, model terbaik diekspor ke format TorchScript (.pt), sebuah format yang dioptimalkan untuk inferensi di berbagai platform, termasuk perangkat *mobile*. Aplikasi Android kemudian dikembangkan menggunakan *framework* Flutter karena kemampuannya dalam membangun antarmuka pengguna yang responsif dari satu basis kode. Integrasi model ke dalam aplikasi difasilitasi oleh *plugin* flutter_pytorch, yang menyediakan jembatan untuk memuat model dan menjalankan inferensi secara langsung pada perangkat keras perangkat. Arsitektur ini memungkinkan seluruh alur kerja—mulai dari pengambilan gambar (melalui kamera atau galeri), proses inferensi oleh model, hingga visualisasi hasil deteksi—berjalan sepenuhnya secara lokal (*on-device*), sehingga aplikasi dapat berfungsi tanpa koneksi internet dan menjaga privasi data pengguna. Gambar 4 menampilkan antarmuka pengguna aplikasi, dari layar utama hingga layar hasil deteksi.

File model ditempatkan dalam direktori assets/models/ proyek Flutter bersama *file* label pada assets/labels/. Pada proses inisiasi, model dimuat menggunakan fungsi FlutterPytorch.loadObjectDetectionModel() dengan parameter berupa path model, jumlah kelas, ukuran input (640×640), serta file label. Proses ini dilakukan sekali pada saat inisiasi aplikasi (main.dart). Selanjutnya, aplikasi menyediakan dua pilihan sumber data, yaitu kamera (camera_screen.dart) dan galeri (gallery_screen.dart). Jika pengguna memilih kamera, setiap *frame* ditangkap menggunakan *plugin* camera. Jika pengguna memilih galeri, gambar dipilih melalui *plugin* image_picker.

Citra yang diperoleh dari kamera atau galeri dikirim ke model melalui fungsi getImagePrediction(). Hasil keluaran berupa daftar objek yang terdeteksi, berisi koordinat *bounding box* dan label kelas. Hasil deteksi divisualisasikan pada antarmuka aplikasi dengan menggambar *bounding box* pada



Gambar 4. *User interface* (UI) aplikasi deteksi ikan lele

citra, serta menampilkan jumlah bibit ikan yang berhasil teridentifikasi. Dengan pendekatan ini, aplikasi Android dapat menjalankan model YOLOv5 secara langsung menggunakan PyTorch di Flutter tanpa memerlukan server eksternal. Hal ini membuat aplikasi lebih praktis digunakan di lapangan karena proses deteksi dilakukan sepenuhnya di perangkat *mobile*.

2.4. Skenario pengujian dan metrik evaluasi

Untuk mengevaluasi kinerja aplikasi di lingkungan yang mendekati kondisi nyata, pengujian dilakukan pada 17 citra yang belum pernah digunakan dalam pelatihan maupun validasi. Citra-citra ini dipilih untuk merepresentasikan tiga skenario utama yang sering ditemui di lapangan:

a. Normal: Pencahayaan baik dan merata, bibit terdistribusi dengan baik, dan tidak ada tumpang-tindih.
b. Bergerombol (*Clustered*): Beberapa bibit saling berdekatan dan tumpang-tindih, menyebabkan oklusi sebagian.
c. Bayangan (*Shadow*): Terdapat bayangan signifikan yang jatuh di atas bibit atau area sekitarnya.

Kinerja kuantitatif diukur menggunakan metrik standar dalam evaluasi deteksi objek: *Precision*, *Recall*, dan *F1-Score*.

a. Precision = TP / (TP + FP), mengukur akurasi dari deteksi yang dibuat (seberapa banyak deteksi yang benar dari total deteksi).
b. Recall = TP / (TP + FN), mengukur kelengkapan deteksi (seberapa banyak objek asli yang berhasil dideteksi).
c. F1-Score = 2 * (Precision * Recall) / (Precision + Recall), adalah rata-rata harmonik dari *Precision* dan *Recall*, memberikan skor tunggal yang menyeimbangkan kedua metrik tersebut.

Di mana TP (*True Positive*) adalah jumlah bibit yang terdeteksi dengan benar, FP (*False Positive*) adalah deteksi keliru (misalnya, bayangan atau kotoran yang dianggap bibit), dan FN

Gambar 5. Citra *input* dengan 10 bibit ikan lele

Gambar 6. Hasil deteksi pada aplikasi (jumlah terdeteksi: 10)

Tabel 2. Hasil rinci pengujian deteksi bibit ikan lele

No.	Jml. bibit	Kondisi	Respon (ms)	TP	FN	FP
1	1	normal	2450	1	0	0
2	2	normal	2545	2	0	0
3	3	normal	2235	3	0	0
4	4	normal	1649	4	0	0
5	4	gerombol	1692	3	1	0
6	5	gerombol	1592	5	0	0
7	5	normal	1657	5	0	0
8	6	normal	2291	3	3	0
9	6	normal	1647	5	1	1
10	6	normal	1675	6	0	0
11	7	normal	1823	7	0	0
12	7	bayangan	1965	4	3	3
13	7	bayangan & bergerombol	2124	7	0	0
14	10	gerombol	1990	7	3	0
15	10	normal	1858	10	0	0
16	10	gerombol	2405	8	2	0
17	10	gerombol	2184	8	1	1

(*False Negative*) adalah jumlah bibit yang ada namun gagal terdeteksi.

3. Hasil

Bagian ini menyajikan temuan-temuan dari pengujian aplikasi secara objektif. Hasil disajikan dalam bentuk visualisasi deteksi dan data kuantitatif dalam tabel.

3.1. Contoh deteksi dalam aplikasi

Untuk memberikan gambaran fungsionalitas aplikasi, Gambar 5 menampilkan citra *input* asli yang berisi 10 ekor bibit ikan lele. Gambar 6 menunjukkan hasil *output* dari aplikasi setelah pemrosesan. Terlihat bahwa model berhasil mendeteksi ke-10 bibit dengan benar, melingkupinya dengan *bounding box* merah dan menampilkan jumlah total "10" di bagian bawah layar. Contoh ini mendemonstrasikan kinerja aplikasi dalam kondisi yang relatif ideal.

3.2. Evaluasi kinerja kuantitatif

Pengujian yang lebih komprehensif dilakukan pada 17 citra uji dengan berbagai kondisi. Tabel 2 menyajikan rincian hasil deteksi untuk setiap citra, mencakup jumlah bibit aktual, kondisi lingkungan, waktu respon, serta nilai TP, FN, dan FP.

Data dari Tabel 2 kemudian diagregasi berdasarkan kondisi

Tabel 3. Ringkasan kinerja model per kondisi

Kondisi	*Precision*	*Recall*	F1-Score	Catatan kualitatif singkat
Normal	0.979	0.920	0.949	Performa sangat baik; deteksi akurat pada citra jelas.
Bergerombol (*clustered*)	0.969	0.795	0.874	*Recall* menurun (*missed detections*) akibat oklusi/tumpang-tindih (FN tinggi).
Bayangan (*shadow*)	0.786	0.786	0.786	*Precision* menurun (*false positives tinggi*) kemungkinan karena bayangan menyerupai fitur bibit.

lingkungan untuk menghitung metrik kinerja keseluruhan. Selanjutnya, Tabel 3 merangkum hasil perhitungan *Precision*, *Recall*, dan *F1-Score* untuk setiap skenario pengujian.

4. Pembahasan

Bagian ini menginterpretasikan temuan yang telah disajikan pada bab hasil, menganalisis faktor-faktor yang memengaruhi kinerja model, dan membahas implikasi dari hasil tersebut. Hasil pengujian secara jelas mengindikasikan bahwa kinerja model deteksi sangat bergantung pada kondisi visual dari citra *input*.

Pada kondisi normal, model menunjukkan kinerja yang sangat memuaskan dengan F1-Score mencapai 0.949. Sebagaimana terlihat pada Tabel 3, nilai *Precision* yang sangat tinggi (0.979) menandakan bahwa hampir semua objek yang dideteksi oleh model adalah benar-benar bibit lele. Sementara itu, nilai *Recall* yang juga tinggi (0.920) menunjukkan bahwa model berhasil menemukan sebagian besar bibit yang ada di dalam citra. Hal ini mengonfirmasi bahwa dalam kondisi ideal—dengan pencahayaan yang baik dan distribusi objek yang tidak tumpang-tindih—model dapat berfungsi sebagai alat hitung yang andal dan akurat.

Tantangan pertama yang signifikan muncul pada kondisi bergerombol. Di sini, F1-Score turun menjadi 0.874. Analisis lebih dalam pada metrik penyusunnya mengungkapkan bahwa penurunan ini secara spesifik disebabkan oleh anjloknya nilai *Recall* ke 0.795, meskipun *Precision* tetap tinggi (0.969). Secara kualitatif, ini berarti model masih sangat akurat dalam mengklasifikasikan objek yang berhasil dideteksinya, tetapi ia gagal menemukan sejumlah bibit yang seharusnya ada (*False Negative* meningkat). Fenomena ini terjadi karena beberapa bibit saling tumpang-tindih (oklusi), menyebabkan fitur visual unik dari masing-masing bibit menjadi ambigu atau tidak lengkap. Akibatnya, model kesulitan untuk memisahkan dan mengidentifikasi setiap individu bibit dalam gerombolan yang padat.

Kondisi yang paling menantang bagi model adalah kondisi bayangan, di mana F1-Score turun drastis ke 0.786. Berbeda dengan skenario bergerombol, penurunan ini utamanya dipicu oleh anjloknya nilai *Precision* ke 0.786. Ini menandakan bahwa model menghasilkan banyak deteksi yang salah (*False Positive* meningkat). Kemungkinan besar, area gelap atau kontras tinggi yang diciptakan oleh bayangan memiliki karakteristik visual (seperti bentuk memanjang atau gradien warna) yang secara keliru ditafsirkan oleh model sebagai fitur yang mirip dengan bibit lele. Hal ini menyebabkan kesalahan identifikasi, di mana bayangan atau artefak visual lainnya dideteksi sebagai objek target.

Perlu dicatat, terdapat sebuah hasil anomali pada pengujian dengan kondisi gabungan "bayangan dan bergerombol" (Tabel 2, baris 13), di mana model justru mencapai deteksi sempurna (TP=7, FN=0, FP=0). Hasil ini kontras dengan performa model yang lebih rendah pada kondisi tunggal yang lebih sederhana. Diduga kuat bahwa citra spesifik yang digunakan untuk pengujian ini secara kebetulan memiliki karakteristik pencahayaan atau penempatan objek yang unik, sehingga tidak merepresentasikan tantangan sesungguhnya dari gabungan kedua kondisi tersebut. Oleh karena itu, hasil ini dianggap sebagai *outlier* dan interpretasi umum mengenai kesulitan kondisi bayangan tetap didasarkan pada keseluruhan data yang menunjukkan adanya penurunan performa.

Dari segi efisiensi komputasi, waktu respon aplikasi di perangkat *mobile* berkisar antara 1.6 hingga 2.5 detik per inferensi. Mengingat proses manual dapat memakan waktu berjam-jam, waktu pemrosesan ini dianggap sangat efisien dan memadai untuk mendukung penggunaan praktis di lapangan oleh para peternak.

5. Kesimpulan

Penelitian ini berhasil merancang, mengimplementasikan, dan mengevaluasi sebuah aplikasi Android berbasis YOLOv5 untuk otomasi penghitungan bibit ikan lele. Aplikasi ini terbukti memiliki kinerja yang sangat baik dan akurat pada kondisi pencahayaan dan penempatan objek yang ideal, dengan F1-Score mencapai 0.949. Namun, studi ini juga secara kuantitatif mengidentifikasi dua kelemahan utama model. Pertama, kinerjanya terdegradasi pada kondisi bibit bergerombol akibat oklusi, yang menyebabkan penurunan *Recall*. Kedua, kinerjanya menurun drastis pada kondisi bayangan, yang menyebabkan penurunan *Precision* akibat deteksi positif palsu. Temuan ini menegaskan bahwa meskipun YOLOv5 merupakan fondasi teknologi yang kuat, ketangguhannya di lingkungan dunia nyata sangat bergantung pada kemampuannya dalam mengatasi variasi pencahayaan dan objek yang tumpang-tindih.

Berdasarkan temuan tersebut, pengembangan di masa depan disarankan untuk fokus pada beberapa area strategis untuk meningkatkan ketangguhan model. Pertama, pengayaan dataset pelatihan menjadi prioritas utama. Ini dapat dilakukan dengan mengumpulkan lebih banyak data citra yang secara spesifik merepresentasikan kondisi sulit (oklusi berat dan berbagai jenis bayangan) dan menggunakan teknik augmentasi data (misalnya, rotasi, *mosaic*, simulasi bayangan) untuk meningkatkan variasi. Kedua, penelitian lebih lanjut dapat mengeksplorasi metode *pre-processing* citra yang dapat diintegrasikan ke dalam alur kerja aplikasi, seperti algoritma normalisasi pencahayaan atau peningkatan kontras (misalnya, *Contrast Limited Adaptive Histogram Equalization* - CLAHE) untuk meminimalisir dampak negatif dari bayangan sebelum citra diumpankan ke model. Terakhir, pengujian pada dataset yang lebih besar dan terstandarisasi, yang mencakup lebih banyak variasi dari berbagai lokasi tambak, diperlukan untuk memvalidasi generalisasi dan keandalan model sebelum

diimplementasikan dalam skala yang lebih luas oleh para peternak.

Ketersediaan data

Semua data yang dihasilkan atau dianalisis selama penelitian tersedia dalam artikel ini.

Deklarasi konflik kepentingan

Para penulis menyatakan bahwa mereka tidak memiliki konflik kepentingan atau hubungan pribadi yang diketahui yang dapat mempengaruhi pekerjaan yang dilaporkan dalam makalah ini.

Kontribusi penulis

Semua penulis mendesain artikel, berkontribusi dalam penulisan konten, dan merevisi naskah artikel. MZ adalah penulis utama draf pertama naskah dan MSZ merevisi naskah. Semua penulis membaca dan menyetujui naskah versi akhir.

Daftar rujukan

Arganata, A. R., Rasmana, S. T., & Kusumawati, W. I. (2020). *Analisis Perhitungan Bibit Ikan Gurame Menggunakan Webcam dengan Metode Yolo (You Only Look Once)*. https://repository.dinamika.ac.id/id/eprint/5195/

Armalivia, S., Zainuddin, M. S., & MT., P. D. I. A. (2021). *Penghitungan Otomatis Larva Udang Menggunakan Metode Yolo*. https://repository.unhas.ac.id/id/eprint/12479/

Badgujar, C. M., Poulose, A., & Gan, H. (2024). Agricultural Object Detection with You Look Only Once (YOLO) Algorithm: A Bibliometric and Systematic Literature Review. *ArXiv*. https://doi.org/10.48550/arXiv.2401.10379

Dendi, A. S., & Sunardi, S. (2021). *Alat Penghitung Benih Ikan Lele Menggunakan Pengolahan Citra*. http://repository.polman-babel.ac.id/id/eprint/350/

Putra, W. S. E. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101. *Jurnal Teknik ITS*, *5*(1). https://doi.org/10.12962/j23373539.v5i1.15696

Fauziyah, N., Nirmala, K., Supriyono, E., & Hadiroseyani, Y. (2019). Evaluasi Sistem Budidaya Lele: Aspek Produksi dan Strategi Pengembangannya (Studi Kasus: Pembudidaya Lele Kabupaten Tangerang). *Jurnal Kebijakan Sosial Ekonomi Kelautan Dan Perikanan*, *9*(2), 129. https://doi.org/10.15578/jksekp.v9i2.7764

Ilahiyah, S., & Nilogiri, A. (2018). Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. *JUSTINDO (Jurnal Sistem dan Teknologi Informasi Indonesia)*, *3*(2), 49–56. https://doi.org/10.32528/justindo.v3i2.2254

Ciptawati, E., Budi Rachman, I., Oktiyani Rusdi, H., & Alvionita, M. (2021). Analisis Perbandingan Proses Pengolahan Ikan Lele terhadap Kadar Nutrisinya. *IJCA (Indonesian Journal of Chemical Analysis)*, *4*(1), 40–46. https://doi.org/10.20885/ijca.vol4.iss1.art5

Sitio, M. H. F., Jubaedah, D., & Syaifudin, M. (2017). Kelangsungan Hidup dan Pertumbuhan Benih Ikan Lele (Clarias sp.) Pada Salinitas Media yang Berbeda. *Jurnal Akuakultur Rawa Indonesia*, *5*(1), 83–96. https://doi.org/10.36706/jari.v5i1.5810

Sulistiyanti, S., Setyawan, F. A., & Komarudin, M. (2016). *Pengolahan Citra Dasar dan Contoh Penerapannya*. Teknosain. https://www.researchgate.net/profile/Muhamad-Komarudin/publication/337928062_PENGOLAHAN_CITRA_DASAR_DAN_CONTOH_PENERAPANNYA/links/67f71aa495231d5ba5bd8e5f/PENGOLAHAN-CITRA-DASAR-DAN-CONTOH-PENERAPANNYA.pdf

Yumna, A. S., Rukmono, D., Panjaitan, A. S., & Mulyono, M. (2019). Penigkatan Produktifitas Ikan Lele (Clarias sp.) Sistem Bioflok Di Pesantren Modern Darul Ma'arif Legok, Indramayu. *Jurnal Kelautan Dan Perikanan Terapan (JKPT)*, *2*(2), 113. https://doi.org/10.15578/jkpt.v2i2.8080

Mohammad Zainuddin memperoleh gelar Magister Teknik Informatika dari Universitas Dian Nuswantoro, Semarang (2017). Saat ini, beliau menjabat sebagai Dosen pada Program Studi Teknik Informatika di Institut Teknologi dan Bisnis Asia Malang.

Muhammad Saifuddin Zuhri. memperoleh gelar Sarjana Komputer dari Institut Teknologi dan Bisnis Asia, Malang (2024). Saat ini, beliau bekerja sebagai *Full-stack Developer* di PT. Mulia Bagus Digital. Minat risetnya meliputi visi komputer dan pengembangan aplikasi *mobile*.