IMPLEMENTASI SISTEM DETEKSI KERUSAKAN ASET BERBASIS CITRA DENGAN CONVOLUTIONAL NEURAL NETWORK (CNN) PADA APLIKASI MOBILE

TUGAS AKHIR

BERNARDUS REYNALDI ANANDA PRIASMARA

NIM : 312110001

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

MALANG

2025

LEMBAR PENGESAHAN
TUGAS AKHIR

IMPLEMENTASI SISTEM DETEKSI KERUSAKAN ASET BERBASIS CITRA DENGAN CONVOLUTIONAL NEURAL NETWORK (CNN) PADA APLIKASI MOBILE

Oleh:

BERNARDUS REYNALDI ANANDA PRIASMARA

NIM. 312110001

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan dan berhak mendapatkan gelar Sarjana S.Kom.

Dosen Pembiming I,	Dosen Pembimbing II,
Paulus Lucky T. Irawan, S.Kom., MT.	**Yudhi Kurniawan, S.Kom., M.MT**
NIP. 20100005	**NIP. 20100032**

Dekan Fakultas Sains dan Teknologi,

Prof. Dr. Eng. Romy Budhi Widodo
NIP. 20070035

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan dari Tugas Akhir saya dengan judul "Implementasi Sistem Deteksi Kerusakan Aset Berbasis Citra dengan Convolutional Neural Network (CNN) pada Aplikasi Mobile" adalah benar benar hasil karya intelektual mandiri tanpa menggunakan bahan – bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar Pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Malang, 29 September 2025

Bernardus Reynaldi Ananda Priasmara
NIM. 312110001

IMPLEMENTASI SISTEM DETEKSI KERUSAKAN ASET BERBASIS CITRA DENGAN CONVOLUTIONAL NEURAL NETWORK (CNN) PADA APLIKASI MOBILE

Bernardus Reynaldi Ananda Priasmara[1] ,
Paulus Lucky Tirma Irawan[2], Yudhi Kurniawan[3]
Universitas Ma Chung

Abstrak

Inspeksi kerusakan aset kampus yang dilakukan secara manual sering terlambat dan rawan inkonsistensi. Untuk menjawab masalah ini, penelitian ini mengembangkan sistem deteksi kerusakan kursi berbasis visi komputer dengan memanfaatkan model *You Only Look Once* (YOLO). Empat jenis kerusakan yang menjadi fokus adalah jamur, cat mengelupas, patah, dan roda rusak. Seluruh citra dilatih menggunakan resolusi seragam 1024×1024 dengan pendekatan *transfer learning*, kemudian dievaluasi menggunakan metrik precision, recall, F1-Score, serta mAP (mAP@50 dan mAP@50–95). Selain akurasi, penelitian ini juga menilai kestabilan pelatihan melalui kurva loss dan konsistensi antar-epoch, serta mengkaji potensi integrasi hasil deteksi pada aplikasi mobile untuk mendukung manajemen aset secara real-time.

Hasil pengujian memperlihatkan bahwa YOLOv8s memberikan performa paling optimal dengan rata-rata precision 0,852, recall 0,703, F1-Score 0,770, mAP@50 sebesar 0,769, dan mAP@50–95 sebesar 0,697 berdasarkan validasi silang 5-fold. Model ini unggul tipis namun konsisten dibanding YOLOv5s yang mencatat precision 0,772, recall 0,730, F1-Score 0,751, mAP@50 sebesar 0,737, dan mAP@50–95 sebesar 0,690. Sementara itu, YOLOv11n menonjol dalam aspek sensitivitas dengan recall 0,737 dan mAP@50 tertinggi 0,772, meski presisinya lebih rendah (0,745) dan mAP@50–95 masih terbatas di 0,672. Perbandingan ini menegaskan bahwa YOLOv8s memiliki keseimbangan terbaik antara akurasi, generalisasi, dan efisiensi, sementara YOLOv5s relevan sebagai alternatif yang stabil, dan YOLOv11n menawarkan sensitivitas lebih tinggi meskipun dengan risiko *false positive*. Dengan demikian, YOLOv8s direkomendasikan sebagai model utama untuk mendukung otomasi deteksi kerusakan kursi berbasis citra, yang siap diintegrasikan ke pipeline pengawasan maupun aplikasi mobile.

Kata kunci : deteksi kerusakan aset, CNN, YOLOv8, Flutter, aplikasi mobile

IMPLEMENTATION OF AN IMAGE-BASED ASSET DAMAGE DETECTION SYSTEM USING CONVOLUTIONAL NEURALL NETWORK (CNN) ON A MOBILE APPLICATION

Bernardus Reynaldi Ananda Priasmara[1] ,
Paulus Lucky Tirma Irawan[2], Yudhi Kurniawan[3]
Universitas Ma Chung

Abstract

Manual inspection of campus assets is often delayed and inconsistent. To address this issue, this study developed a computer vision-based chair damage detection system using the *You Only Look Once* (YOLO) algorithm. Four damage categories were investigated: mold, paint peeling, fractures, and broken wheels. All images were normalized to a resolution of 1024×1024 and trained using a transfer learning approach, then evaluated with precision, recall, F1-Score, and mAP (mAP@50 and mAP@50–95). Beyond accuracy, training stability was examined through loss curves and epoch-to-epoch consistency, while the feasibility of integrating detection results into mobile applications was also assessed to support real-time asset management.

The experimental results show that YOLOv8s achieved the best balance, with average precision of 0.852, recall of 0.703, F1-Score of 0.770, mAP@50 of 0.769, and mAP@50–95 of 0.697 across 5-fold cross-validation. This model consistently outperformed YOLOv5s, which obtained precision of 0.772, recall of 0.730, F1-Score of 0.751, mAP@50 of 0.737, and mAP@50–95 of 0.690. Meanwhile, YOLOv11n demonstrated the highest sensitivity, with recall of 0.737 and the highest mAP@50 of 0.772, but lower precision (0.745) and a reduced mAP@50–95 of 0.672. These findings highlight YOLOv8s as the most reliable and well-balanced model in terms of accuracy, generalization, and efficiency. YOLOv5s remains a stable alternative with competitive performance, while YOLOv11n offers strong sensitivity at the cost of more false positives. Therefore, YOLOv8s is recommended as the primary model for automating chair damage detection, with practical readiness for integration into monitoring pipelines and mobile applications.

Keywords : *asset damage detection, CNN, YOLOv8, Flutter, mobile application*

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya, penulis dapat menyelesaikan penyusunan Tugas Akhir yang berjudul "Implementasi Sistem Deteksi Kerusakan Aset Berbasis Citra dengan Convolutional Neural Network (CNN) pada Aplikasi Mobile" dengan baik. Tugas Akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika, Fakultas Teknologi dan Desain, Universitas Ma Chung.

Perjalanan penulis dalam menempuh studi di Program Studi Teknik Informatika merupakan pengalaman yang penuh tantangan, pembelajaran, dan pertumbuhan. Sejak awal perkuliahan, penulis diperkenalkan pada berbagai bidang teknologi yang terus berkembang, mulai dari dasar-dasar pemrograman, struktur data, teknologi mobile dan kecerdasan buatan. Setiap semester memberikan tantangan baru bagi penulis yang memperkaya pengetahuan dan membentuk cara berpikir logis serta analitis. Tidak sedikit hambatan dan rintangan yang ditemui, baik secara akademik, maupun non-akademik, namun melalui semangat belajar dan dukungan dari berbagai pihak, penulis mampu melewatinya dan menumbuhkan rasa percaya diri dalam menyelesaikan studi ini dari awal hingga tahap akhir.

Dalam perjalanan menyelesaikan Tugas Akhir ini, penulis telah mendapatkan banyak dukungan, semangat, dan bantuan dari berbagai pihak. Oleh karena itu, dengan penuh rasa hormat dan terima kasih, penulis ingin menyampaikan penghargaan setinggi-tingginya kepada:

1. Bapak Prof. Dr. Eng. Romy Budhi Widodo, Selaku Dekan Fakultas Sains dan Teknologi. Penulis menyampaikan terima kasih atas dukungan dan kepemimpinan beliau yang menginspirasi serta menciptakan lingkungan akademik yang kondusif bagi pengembangan ilmu pengetahuan.
2. Bapak Hendry Setiawan, ST., M. Kom., Selaku Kepala Program Studi Teknik Informatika Universitas Ma Chung, atas arahannya selama perkuliahan dan atas bimbingannya yang sangat membantu dalam proses akademik penulis.
3. Bapak Paulus Lucky Tirma I., S.Kom., MT. selaku dosen pembimbing I dalam pengerjaan proyek Tugas Akhir yang telah memberikan arahan, saran, dan bimbingan selama proses pengerjaan Tugas Akhir ini. Terima kasih atas kesabaran dan waktu yang telah diberikan dalam membimbing penulis hingga tugas akhir bisa diselesaikan dengan baik.
4. Bapak Yudhi Kurniawan, S.Kom., M.MT. selaku dosen pembimbing dalam

pengerjaan proyek Tugas Akhir, sekaligus sebagai Kepala Unit Sistem Informasi dan Pusat Data (SIPUSDA), atas bimbingan teknis, masukan, dan dukungan data yang sangat membantu dalam proses penelitian ini.

5. Ibu dan Kakak penulis yang selalu memberikan kasih sayang, doa, dan dukungan moral maupun material secara tanpa henti sepanjang proses studi hingga selesainya Tugas Akhir ini.
6. Seluruh dosen dan staf di lingkungan Universitas Ma Chung yang telah membekali penulis dengan ilmu dan pengalaman berharga selama masa studi.
7. Teman-teman dekat penulis yang telah menjadi tempat berbagi semangat, motivasi, dan kebersamaan selama masa studi, baik yang berada di Malang, seperti Davin, Jonathan, Evan, Kenrick, Fito, Richi, Jerry, Jason, Gerry, Rafa, maupun teman-teman yang tidak dapat disebutkan satu per satu. Terima kasih juga kepada teman-teman di Universitas Ma Chung seperti Kevin, Vincent, David, Marvin, Louis, Natan, Cevin, serta seluruh rekan seperjuangan di Teknik Informatika angkatan 2021.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki kekurangan. Oleh karena itu, penulis sangat terbuka terhadap kritik dan saran yang membangun demi perbaikan di masa mendatang. Semoga karya ini dapat memberikan manfaat bagi pengembangan ilmu pengetahuan, khususnya dalam bidang teknologi informasi dan pengelolaan aset berbasis AI.

Malang, 15 April 2025

Bernardus Reynaldi Ananda Priasmara
312110001

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Pengelolaan aset di lingkungan universitas merupakan salah satu aspek penting dalam menunjang efektivitas operasional. Universitas Ma Chung, sebagai institusi pendidikan yang terus berkembang, memiliki banyak aset fisik seperti kursi, meja, dan peralatan lain yang memerlukan pemantauan berkala untuk memastikan kondisinya baik atau mendeteksi apabila terjadi kerusakan. Pengelolaan data aset ini berada di bawah tanggung jawab Sistem Informasi dan Pusat Data (SIPUSDA) Universitas Ma Chung. Namun, proses pendataan dan pemeriksaan aset yang dilakukan oleh SIPUSDA saat ini masih bersifat manual, membutuhkan waktu lama dan rentan terhadap kesalahan manusia *(human error)*. Keterlambatan pembaruan data dan terbatasnya tenaga untuk inspeksi rutin menyebabkan kerusakan aset kadang luput dari perhatian hingga sudah parah atau mengakibatkan kerugian operasional.

Perkembangan teknologi kecerdasan buatan, khususnya penglihatan komputer (*computer vision*), menawarkan solusi inovatif untuk otomatisasi deteksi objek. Convolutional Neural Network (CNN) merupakan salah satu arsitektur deep learning yang terbukti efektif mengklasifikasikan gambar dengan akurasi tinggi. Platform pendukung seperti Roboflow dapat mempermudah proses pelabelan dataset dan pelatihan model sehingga mempercepat pengembangan sistem deteksi. Selain itu, aplikasi mobile berbasis Flutter dipilih sebagai platform pengembangan karena kemampuannya membangun aplikasi lintas platform secara efisien dan mudah diintegrasikan dengan model pembelajaran mesin. Dengan menggabungkan CNN, Roboflow, dan Flutter, diharapkan tercipta solusi yang mampu mendeteksi kondisi aset melalui kamera smartphone, sehingga SIPUSDA selaku pengelola aset dapat meningkatkan efisiensi manajemen aset di Universitas Ma Chung.

Penelitian serupa telah dilakukan oleh beberapa peneliti sebelumnya. Zhang et al. (2019) mengembangkan metode berbasis CNN untuk mendeteksi kerusakan pada furnitur dan memperoleh akurasi tinggi dalam mengidentifikasi kerusakan fisik pada kursi dan meja. Lee dan Park (2021) membangun sistem manajemen aset

berbasis mobile yang terintegrasi dengan deteksi objek real-time; aplikasi mereka memungkinkan petugas memindai aset dan secara otomatis mengenali objek tersebut menggunakan model deteksi berbasis CNN tertanam di smartphone. Hasil-hasil penelitian terdahulu ini menunjukkan potensi besar penerapan deep learning dan mobile computing dalam inspeksi aset. Oleh karena itu, proyek Tugas Akhir ini bertujuan mengembangkan sistem manajemen aset yang lebih terintegrasi dengan memanfaatkan teknologi terkini, khususnya untuk deteksi kerusakan aset secara otomatis melalui perangkat mobile. Proyek ini dilaksanakan bekerja sama dengan SIPUSDA Universitas Ma Chung untuk mewujudkan sistem deteksi kerusakan aset kampus menggunakan data aset yang disediakan oleh SIPUSDA.

Salah satu model terbaru dalam deteksi objek adalah YOLOv8, yang digunakan untuk identifikasi kerusakan fisik secara otomatis melalui analisis citra. Dalam konteks manajemen aset universitas, YOLOv8 dapat diterapkan untuk mendeteksi kerusakan pada furnitur dan peralatan kampus dengan akurasi tinggi, sekaligus mengklasifikasikan tingkat kerusakan berdasarkan pola visual seperti retak, patah, atau aus. *You Only Look Once* (YOLO) adalah algoritma berbasis CNN yang menggunakan pendekatan *single neural network* untuk melakukan deteksi objek secara *real-time* (Telaumbanua, 2024). Data hasil deteksi ini kemudian dapat diintegrasikan dengan sistem pelaporan berbasis *mobile* untuk otomatisasi pendataan aset.

Aset fasilitas kampus seperti kursi kelas dan perabot lainnya memerlukan pemeliharaan rutin. Kerusakan pada aset, misalnya kursi rusak, perlu terdeteksi dan didata dengan cepat agar penanganan dapat segera dilakukan. Di Universitas Ma Chung, proses pendataan kerusakan aset selama ini ditangani oleh Bagian Aset kampus secara manual. Bagian Aset sebelumnya menggunakan sebuah aplikasi sederhana untuk pendataan: petugas mengambil foto kursi yang rusak dan mengunggahnya ke sistem. Namun, aplikasi tersebut hanya berfungsi sebagai media unggah gambar dan pencatatan; klasifikasi atau identifikasi tingkat kerusakan *tidak* dilakukan secara otomatis. Akibatnya, proses analisis kerusakan masih mengandalkan inspeksi visual manual oleh petugas, yang rawan tidak konsisten dan memakan waktu.

Unit Sistem Informasi dan Pusat Data (SIPUSDA) universitas membantu dari sisi infrastruktur TI dan dukungan aplikasi. Namun, tanggung jawab utama pencatatan kerusakan tetap berada di Bagian Aset. SIPUSDA menyediakan platform dan dukungan teknis untuk aplikasi, sedangkan petugas aset yang melakukan input data kerusakan. Dengan kata lain, SIPUSDA mendukung kebutuhan perangkat lunak, tetapi aktivitas pencatatan dan penilaian kondisi aset dikerjakan manual oleh Bagian Aset. Kondisi ini menimbulkan kesenjangan (gap) dalam hal efisiensi dan akurasi: aplikasi lama belum memanfaatkan kecerdasan buatan untuk mengotomatisasi deteksi kerusakan.

Berdasarkan uraian di atas, penelitian ini berfokus pada pengembangan aplikasi mobile untuk deteksi otomatis kerusakan kursi di lingkungan kampus Universitas Ma Chung menggunakan metode CNN. Ruang lingkupnya terbatas pada aset di area kampus Universitas Ma Chung sebagai studi kasus, meskipun secara teknis pendekatan serupa dapat diterapkan di luar kampus. Dengan solusi ini, diharapkan gap pada sistem sebelumnya dapat teratasi: proses pendataan kerusakan menjadi lebih cepat, akurasi identifikasi jenis kerusakan meningkat, dan beban kerja manual petugas berkurang.

1.2 Identifikasi Masalah

Berdasarkan latar belakang yang telah dijelaskan, dapat diidentifikasi bahwa proses pemantauan kondisi aset kampus Universitas Ma Chung masih dilakukan secara manual, tidak efisien, dan memiliki risiko kesalahan manusia yang tinggi. SIPUSDA belum memiliki sistem berbasis kecerdasan buatan yang dapat secara otomatis mendeteksi kerusakan fisik aset melalui visualisasi citra. Kondisi ini memperlambat proses perbaikan aset yang rusak karena deteksi kerusakan bergantung sepenuhnya pada observasi manual. Oleh karena itu, dibutuhkan solusi berupa sistem deteksi berbasis CNN yang dapat membantu petugas SIPUSDA dalam mengidentifikasi kerusakan aset secara cepat, akurat, dan terdokumentasi. Sistem ini tidak dimaksudkan untuk memantau aset 24 jam secara otomatis, tetapi sebagai alat bantu saat inspeksi langsung di lapangan.

1.3 Batasan Masalah

Adapun batasan masalah dalam pengerjaan proyek ini adalah sebagai berikut:

a. Sistem hanya difokuskan untuk mendeteksi kerusakan pada aset kursi dan di lingkungan Universitas Ma Chung.
b. Model deteksi kerusakan menggunakan algoritma CNN dengan pendekatan YOLOv8.
c. Aplikasi yang dikembangkan hanya ditargetkan untuk platform Android menggunakan Flutter.
d. Dataset citra aset yang digunakan bersumber dari SIPUSDA Universitas Ma Chung dan tidak mencakup aset dari institusi lain.

1.4 Rumusan Masalah

Berdasarkan identifikasi masalah yang telah disampaikan, maka berikut adalah rumusan masalah dari proyek ini.

a. Bagaimana membangun model deteksi kerusakan aset berbasis YOLOv8 menggunakan dataset citra aset kampus dari SIPUSDA?
b. Bagaimana mengintegrasikan model YOLOv8 ke dalam aplikasi mobile berbasis Flutter untuk mendeteksi kerusakan aset secara real-time?
c. Bagaimana mengukur kinerja solusi yang diusulkan, baik dari sisi performa model deteksi (akurasi, mean Average Precision, dsb) maupun dari sisi usability aplikasi bagi pengguna (petugas aset)?

1.5 Tujuan

Tujuan dari pengerjaan proyek ini adalah sebagai berikut.

a. Mengembangkan model CNN berbasis YOLOv8 yang mampu mendeteksi kerusakan fisik pada aset kampus dengan akurasi tinggi dan kecepatan inferensi real-time.
b. Membangun aplikasi mobile berbasis Flutter yang terintegrasi dengan model Yolov8 untuk mendukung deteksi kerusakan aset secara efisien.
c. Mengevaluasi kinerja model yang digunakan (tingkat akurasi deteksi kerusakan, precision, recall, mAP) serta kegunaan aplikasi bagi pengguna akhir. Evaluasi aplikasi akan mencakup pengujian fungsional (black-box testing).

1.6 Manfaat

Manfaat dari pengerjaan proyek ini adalah sebagai berikut.

a. Bagi penulis, Meningkatkan pemahaman dan keterampilan dalam pengembangan aplikasi mobile berbasis *Artificial Intelligence* (AI), khususnya dalam implementasi CNN dan integrasinya dengan platform seperti Roboflow.

b. Bagi Universitas, Khususnya bagi Program Studi Teknik Informatika dan SIPUSDA Universitas Ma Chung, hasil proyek ini diharapkan dapat mendukung peningkatan kompetensi mahasiswa melalui pengalaman riset praktis. Di sisi operasional kampus, sistem deteksi dini kerusakan aset ini dapat membantu mengurangi biaya perawatan aset melalui identifikasi masalah sedini mungkin, sehingga memperpanjang usia pakai aset universitas.

c. Bagi Praktisi, Menjadi referensi dalam pengembangan aplikasi serupa untuk manajemen aset di institusi pendidikan atau organisasi lain, khususnya yang ingin menerapkan teknologi pada pemeliharaan aset fisik.

1.7 Luaran Tugas Akhir

Luaran dari penelitian ini berupa aplikasi mobile berbasis Flutter yang terintegrasi dengan model Convolutional Neural Network (CNN) menggunakan pendekatan Yolov8 untuk mendeteksi kerusakan aset di Universitas Ma Chung secara real-time melalui kamera smartphone. Aplikasi ini dilengkapi dengan fitur pelaporan kerusakan yang memungkinkan pengguna menyimpan data hasil deteksi secara lokal, termasuk foto aset, informasi kondisi, dan catatan tambahan. Selain itu, luaran penelitian mencakup model Yolov8 yang telah dilatih dan dievaluasi menggunakan dataset citra aset kampus dari SIPUSDA dengan akurasi tinggi, serta dataset yang telah dianotasi menggunakan Roboflow. Hasil penelitian ini juga didokumentasikan dalam bentuk laporan tugas akhir yang lengkap, termasuk panduan penggunaan aplikasi, serta kode sumber yang terdokumentasi dengan baik dan siap dikembangkan lebih lanjut.

1.8 Sistematika Penulisan

Sistematika penulisan proposal Tugas Akhir ini dibagi dalam tiga bab yaitu sebagai berikut.

BAB I. Pendahuluan

Bab ini terdiri dari latar belakang, identifikasi masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, luaran tugas akhir, serta sistematika penulisan laporan.

BAB II. Tinjauan Pustakan

Bab ini berisi uraian sistematis mengenai literatur yang digunakan dalam penyusunan laporan Tugas Akhir sehingga diperoleh landasan teori tentang android studio, Flutter, CNN dan Machine Learning.

BAB III. Metodologi Penelitian

Bab ini akan menjelaskan tahapan pengerjaan berserta analisis perancangan awal sistem yang akan dibuat. Tahapan pengerjaan terdiri dari identifikasi masalah, studi pustaka, pengumpulan data, desain sistem, dan pengujian arsitektur.

BAB IV. Hasil dan Pembahasan

Bab ini akan membahas mengenai hasil penelitian yang sudah dilakukan untuk metode CNN, dan Machine Learning yang dirancang pada bab sebelumnya.

BAB V. Kesimpulan dan Saran

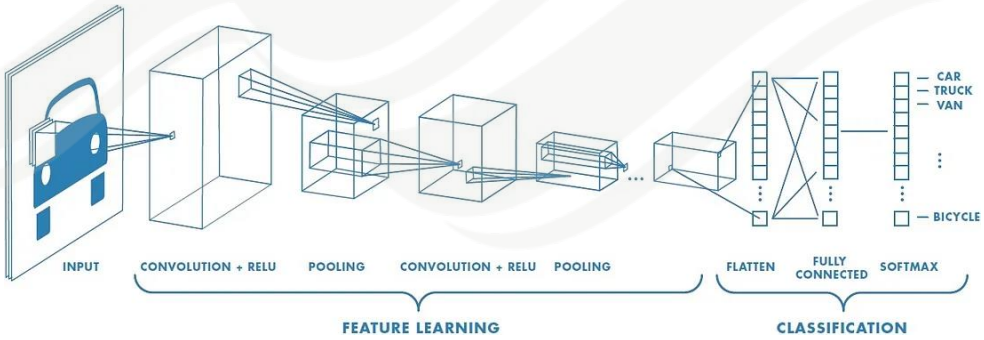
Bab terakhir berisi kesimpulan yang merangkum jawaban atas rumusan masalah berdasarkan hasil penelitian. Kesimpulan disajikan secara singkat dan jelas. Selain itu, disertakan saran-saran untuk pengembangan lebih lanjut, misalnya peningkatan dataset, penyempurnaan model, perluasan cakupan ke jenis aset lain, atau integrasi penuh dynamic model update melalui API ke depannya. Saran juga dapat ditujukan bagi institusi (Bagian Aset dan SIPUSDA) dalam mengadopsi hasil penelitian iniJadwal Penelitian

No.	Kegiatan	Bulan 2025					
		Mar	Apr	Mei	Jun	Jul	Agu
1.	Tahapan Penelitian						
	Penyusunan Laporan dan Judul						
	Pengajuan Proposal						
	Perizinan Penelitian						
2.	Tahapan Pelaksanaan						
	Desain Aplikasi						
	Implementasi Aplikasi						
	Pengujian dan Evaluasi						
3.	Tahap Penyusunan Laporan						

BAB II
TINJAUAN PUSTAKA

2.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) merupakan arsitektur deep learning yang mendominasi bidang visi komputer, khususnya untuk tugas klasifikasi dan deteksi objek. CNN dirancang dengan lapisan konvolusi yang meniru mekanisme penglihatan biologis, di mana setiap neuron hanya merespons stimulus visual pada area terbatas (receptive field), memungkinkan ekstraksi fitur hierarkis dari citra - mulai dari tepi dan tekstur hingga pola objek yang kompleks. Arsitektur modern CNN seperti yang digunakan dalam Yolov8 telah berevolusi secara signifikan sejak kemunculan AlexNet (Krizhevsky et al., 2017), dengan peningkatan utama pada kedalaman jaringan dan efisiensi komputasi.



Gambar 2. 1 Arsitektur Convolusional Neural Network (CNN)

Arsitektur CNN biasanya dimulai dengan input layer yang menerima data masukan, seperti citra hasil penginderaan jauh (remote sensing), dan diakhiri dengan output layer yang menghasilkan prediksi, misalnya klasifikasi spesies tanaman atau hasil panen. Di antara lapisan input dan output terdapat beberapa hidden layers yang bertugas mengolah data masukan dan mengubahnya menjadi representasi yang lebih sesuai untuk output yang diinginkan. Jaringan ini terdiri dari beberapa lapisan yang masing-masing berfungsi untuk mengekstraksi dan mempelajari fitur dari data masukan, hingga akhirnya menghasilkan output berupa klasifikasi atau prediksi. (Kattenborn, et al., 2021).

CNN tidak hanya terbatas pada klasifikasi citra, tetapi juga menjadi tulang punggung dalam deteksi objek (*object detection*). Deteksi objek merupakan tugas

untuk menemukan lokasi (berupa koordinat *bounding box*) sekaligus mengidentifikasi kelas objek yang muncul di dalam sebuah citra. Berbeda dengan klasifikasi yang hanya menentukan satu kategori untuk satu citra secara keseluruhan, deteksi objek harus menangani kemungkinan adanya berbagai objek di lokasi berbeda dalam satu gambar. CNN berperan krusial dalam deteksi objek karena kemampuannya mengekstraksi fitur secara otomatis yang dapat membedakan objek satu dengan lainnya. Sejumlah metode deteksi objek berbasis CNN telah dikembangkan dalam dua pendekatan utama, yakni *two-stage detectors* dan *one-stage detectors*. Pada pendekatan dua tahap (contohnya R-CNN dan turunannya), model akan menghasilkan sejumlah *region proposals* terlebih dahulu yang diduga berisi objek, kemudian CNN digunakan untuk mengklasifikasikan tiap region tersebut ke dalam kelas yang sesuai.

Data mentah biasanya perlu dipersiapkan terlebih dahulu sebelum digunakan untuk pelatihan model. Proses *preprocessing* meliputi pembersihan data, normalisasi fitur, pengubahan ukuran citra, konversi format, dan pembagian data menjadi set pelatihan, validasi, dan pengujian. *Preprocessing* yang tepat sangat penting agar model dapat belajar pola dengan efektif dan menghindari bias dari data yang kotor atau tidak konsisten. Dengan melakukan praproses data (misalnya normalisasi pixel citra), pelatihan CNN menjadi lebih stabil dan konvergen. Sebuah tinjauan menyimpulkan bahwa *data pre-processing* memiliki peran efektif dalam meningkatkan akurasi model AI dengan memastikan data dalam kondisi optimal untuk dipelajari.

Class imbalance dapat berdampak signifikan pada kinerja model, di mana model lebih sering mengabaikan kelas minoritas. Oleh karena itu, perlu diterapkan teknik penanganan ketidakseimbangan data, seperti *oversampling* (memperbanyak sampel kelas minoritas), *undersampling* (mengurangi sampel kelas mayoritas), pemberian *class weight* yang lebih tinggi untuk kelas minoritas, atau pembuatan data sintetis untuk kelas minoritas. Teknik-teknik tradisional seperti penyeimbangan data melalui sampling dan *cost-sensitive learning* terbukti masih relevan dalam konteks deep learning untuk mengatasi ketimpangan kelas

Data augmentation adalah teknik memperbanyak dan memperkaya dataset pelatihan secara artifisial tanpa perlu mengumpulkan data baru. Contohnya pada

citra, augmentasi dapat berupa rotasi, flipping, zooming, perubahan pencahayaan, dan sebagainya. Tujuan augmentasi adalah meningkatkan keragaman data sehingga model lebih kuat terhadap variasi dan tidak overfitting pada data pelatihan. Teknik Data Augmentation dapat meningkatkan kinerja model dan memperluas dataset terbatas sehingga model CNN dapat memanfaatkan kemampuan belajar pada data yang lebih. Bahkan metode augmentasi tingkat lanjut (misal menggunakan GAN atau mixup) telah menunjukkan hasil menjanjikan dalam meningkatkan generalisasi. Dalam konteks deteksi kerusakan, augmentasi citra (seperti memotong dan menempel kerusakan pada latar berbeda sebagaimana diusulkan Dwibedi et al.) juga dapat membantu model belajar pola kerusakan dengan lebih robust (Dwibedi, 2021).

2.2 Deteksi Objek dan Algoritma YOLO

Tugas object detection berbeda dengan image classification biasa, karena model harus menentukan lokasi objek dalam gambar (melalui koordinat bounding box) sekaligus kelas objek tersebut. YOLO (You Only Look Once) adalah salah satu algoritma object detection yang sangat populer. Ide kunci YOLO adalah menggunakan single-stage detection, di mana sebuah CNN end-to-end langsung memprediksi kelas dan koordinat bounding box beberapa objek sekaligus dalam satu kali forward pass . Pendekatan ini berbeda dari metode dua tahap seperti Faster R-CNN yang terlebih dahulu menghasilkan region proposals kemudian mengklasifikasinya. Dengan YOLO, deteksi objek bisa dilakukan real-time karena seluruh perhitungan disatukan. Versi YOLO orisinal diperkenalkan oleh Redmon et al. (2016) dengan nama YOLOv1 , dan sejak itu algoritma ini terus dikembangkan

YOLO (You Only Look Once) adalah keluarga arsitektur CNN untuk deteksi objek real-time. Salah satu versi yang banyak digunakan saat ini adalah YOLOv8, yang diluncurkan oleh Ultralytics pada awal 2023 sebagai penerus dari YOLOv5. YOLOv8 menghadirkan berbagai peningkatan signifikan dalam arsitektur dan metodologi pelatihan dibanding versi sebelumnya. Secara umum, YOLOv8 tetap mempertahankan prinsip single-stage detector (model tunggal yang langsung memprediksi kotak dan kelas objek), namun dengan backbone dan neck yang disempurnakan untuk memperkuat kemampuan ekstraksi fitur. Peningkatan ini

membuat YOLOv8 mampu mendeteksi objek dengan lebih akurat sekaligus tetap efisien. Desain YOLOv8 dioptimalkan untuk kecepatan dan fleksibilitas, sehingga cocok digunakan untuk berbagai kebutuhan implementasi real-time, termasuk pada perangkat dengan sumber daya terbatas.

2.2.1 Perkembangan Versi YOLO

YOLOv1 (2016) memperkenalkan konsep single-stage detection. YOLOv2 (2017, dikenal sebagai YOLO9000) membawa perbaikan seperti penggunaan anchor boxes untuk menangani berbagai skala objek dan batch normalization untuk stabilitas pelatihan. YOLOv3 (2018) menyempurnakan arsitektur dengan menambah kedalaman jaringan dan feature pyramid sehingga lebih baik dalam mendeteksi objek kecil, serta menggunakan sigmoid untuk prediksi bounding box. YOLOv4 (2020) dikembangkan oleh Bochkovskiy et al., menggabungkan banyak "tricks" training (bag-of-freebies) dan optimasi arsitektur (penggunaan CSPDarknet53 backbone, PANet, dll.) sehingga meningkatkan akurasi tanpa mengorbankan kecepatan.

YOLOv4 (2020) dikembangkan oleh Bochkovskiy et al., menggabungkan banyak "tricks" training (bag-of-freebies) dan optimasi arsitektur (penggunaan CSPDarknet53 backbone, PANet, dll.) sehingga meningkatkan akurasi tanpa mengorbankan kecepatan. YOLOv5 (2020, dirilis oleh Ultralytics) menjadi versi yang de facto banyak dipakai industri, meskipun tanpa makalah resmi. YOLOv5 ditulis sepenuhnya di PyTorch dan terkenal akan kemudahan penggunaannya. Arsitektur YOLOv5 memanfaatkan CSP-Darknet backbone dan PANet neck, dengan head anchor-based . YOLOv5 mendukung ukuran model beragam (n, s, m, l, x) untuk menyesuaikan trade-off akurasi dan kecepatan . Keunggulan YOLOv5 antara lain inference yang sangat cepat dan efisien pada perangkat terbatas (Jetson, Raspberry Pi) serta ekosistem yang matang (pre-trained weights, dukungan komunitas).

YOLOv6dan YOLOv7 merupakan versi yang dikembangkan komunitas terpisah. YOLOv7 dilaporkan mencapai state-of-the-art pada tahun 2022 dengan berbagai optimasi head dan label assignment. Namun, fokus penelitian ini adalah pada lini Ultralytics YOLO yang terus berlanjut ke YOLOv8+.

Diluncurkan Ultralytics awal 2023, YOLOv8 menghadirkan kerangka unified

untuk berbagai tugas (deteksi, segmentasi, pose) dan memperkenalkan arsitektur modern seperti anchor-free detection head serta modul backbone C2f . YOLOv8 meningkatkan akurasi mAP secara signifikan dibanding YOLOv5 untuk ukuran model sebanding . Sebagai contoh, YOLOv8s (small) mencapai mAP 44.9% pada COCO dengan ~11.2M parameter, lebih tinggi dari YOLOv5s yang mAP-nya 37.4% . Kelebihan lain YOLOv8 adalah sifatnya yang lebih versatile – satu model dapat dilatih untuk deteksi maupun segmentasi – dan performa inference tetap cepat berkat optimasi memori.

Versi terbaru dari keluarga YOLO adalah YOLOv11, yang dirilis resmi oleh Ultralytics pada September 2024. YOLOv11 membawa peningkatan arsitektur yang signifikan dibanding YOLOv8, termasuk backbone dan neck yang dioptimalkan, detection head baru untuk mengurangi latensi, serta efisiensi parameter yang lebih baik. Pada benchmark COCO, YOLOv11s (varian small) mampu mencapai mAP 47,0% dengan hanya 9,4 juta parameter, lebih akurat sekaligus lebih ringan daripada YOLOv8s (44,9% mAP, 11,2M parameter). Hal ini menjadikan YOLOv11s sangat menarik untuk implementasi real-time, terutama pada perangkat mobile atau edge yang memiliki keterbatasan sumber daya.

Walaupun YOLOv11 menawarkan peningkatan akurasi dan efisiensi, penelitian ini tidak menggunakan YOLOv11. Alasan utamanya adalah faktor konsistensi dengan penelitian terdahulu serta stabilitas ekosistem. YOLOv8s telah menjadi standar de facto sejak awal 2023, dengan dokumentasi yang lengkap, komunitas yang luas, serta dukungan tool yang matang, sehingga lebih sesuai dengan kebutuhan penelitian ini. Sebaliknya, YOLOv11 yang baru diluncurkan pada akhir 2024 masih relatif baru, dengan dokumentasi dan studi aplikasi yang belum sebanyak YOLOv8, sehingga berpotensi menimbulkan ketidakpastian pada tahap implementasi. Dengan demikian, YOLOv8s dipilih karena dianggap lebih konsisten, stabil, dan relevan untuk penelitian yang berfokus pada implementasi mobile object detection.

2.2.2 Transfer Learning Model YOLO

Mengingat dataset kerusakan kursi yang dimiliki kampus relatif terbatas (hanya ratusan foto yang berhasil dikumpulkan), metode transfer learning akan digunakan untuk melatih model. Transfer learning memanfaatkan model CNN yang

telah dilatih pada dataset besar (misal ImageNet atau COCO) sebagai titik awal, kemudian di-fine-tune pada data spesifik kita. Keuntungan pendekatan ini adalah model sudah memiliki pengetahuan awal tentang fitur-fitur umum (tepi, tekstur, bentuk kursi secara umum) sehingga konvergensi lebih cepat dan akurasi lebih baik meski data sedikit . Model YOLOv8s yang digunakan akan diawali dari pre-trained weights pada COCO (yang sudah mengenali kategori umum "chair" dan objek lainnya). Selanjutnya, lapisan output (deteksi) disesuaikan untuk mendeteksi kelas-kelas kerusakan yang didefinisikan (misal 3-5 kelas kerusakan). Optimasi fine-tuning akan dilakukan dengan meminimalkan loss gabungan: Binary Cross-Entropy (BCE) untuk klasifikasi dan CIoU (Complete IoU) untuk lokalisasi box – sesuai mekanisme loss pada arsitektur YOLO. Optimizer yang dipakai mengikuti praktik umum Ultralytics, yaitu Stochastic Gradient Descent (SGD) momentum (default) dengan learning rate awal sekitar 0.01 . SGD dipilih karena stabil untuk deteksi objek; eksperimen dengan Adam juga mungkin, namun Adam sering membutuhkan LR lebih rendah (sekitar 0.001) . Dalam implementasi, Ultralytics YOLO menyediakan opsi optimizer=auto yang akan memilih pengaturan optimal secara otomatis – fitur ini akan dimanfaatkan jika tersedia.

Pada model YOLOv5/v8, fungsi aktivasi utama yang digunakan pada convolution layers adalah SiLU (Sigmoid Linear Unit) – alias Swish activation . SiLU diyakini membantu aliran gradien lebih baik daripada Leaky ReLU klasik, sehingga meningkatkan ekspresivitas fitur . YOLOv8 default menggunakan SiLU sebagai aktivasi di semua layer Convolution. Sementara YOLOv5 pada implementasi Ultralytics juga diupdate menggunakan SiLU (awal YOLOv5 mungkin Leaky ReLU, namun di versi terbaru memakai activation serupa YOLOv8). Untuk spesifikasi pastinya mengikuti YOLOv8, yaitu arsitektur anchor-free dengan aktivasi non-linier SiLU dan komponen attention opsional. Pemahaman ini penting saat melakukan penyesuaian arsitektur atau saat mencoba teknik knowledge distillation antar model.

2.2.3 Data Augmentation

Overfitting adalah kekhawatiran utama saat data training sedikit. Oleh karena itu, teknik augmentasi data diterapkan selama pelatihan model YOLO. Augmentasi yang digunakan mengacu pada pengaturan default Ultralytics YOLO, antara lain:

random scaling, random rotation, shear, perubahan brightness/contrast, flip, mosaic augmentation, dan mixup. Augmentasi membantu memperkaya variasi data sehingga model lebih robust terhadap kondisi nyata. Misalnya, mosaic augmentation menggabungkan 4 gambar acak, efektif untuk melatih deteksi multiobjek dan memperkuat deteksi objek kecil. Random rotation ±15° melatih model agar tahan terhadap sudut foto berbeda. Brightness/hue variation ±15% menyiapkan model menghadapi pencahayaan berbeda. Penelitian terkini menegaskan bahwa augmentasi mampu mengurangi overfitting dengan menambah variability data sehingga model tidak sekadar menghafal detail khusus gambar latih. Meskipun demikian, augmentasi perlu diterapkan secara masuk akal. Terlalu banyak gambar hasil augmentasi yang mirip justru tidak menambah informasi baru dan berpotensi memperkenalkan artefak. Oleh karena itu, dataset disiapkan dengan memperhatikan tips: hindari menyertakan gambar yang hampir duplikat (hanya beda sedikit) karena tidak menambah nilai. Juga, mulai pelatihan dari contoh-contoh yang jelas terlebih dahulu sebelum memasukkan contoh yang sulit (dengan occlusion, blur) , agar model dapat belajar pola dasar dengan benar.

Selain augmentasi algoritmis, panduan pengambilan gambar di lapangan diberikan kepada petugas agar kualitas input ke model tetap optimal. Panduan ini merupakan bagian dari proses *data preparation* pada tahap awal deployment aplikasi. Beberapa hal yang ditekankan dalam panduan tersebut antara lain: petugas harus memastikan pencahayaan cukup saat memotret kerusakan, karena gambar yang terlalu gelap atau terlalu silau dapat menyulitkan model dalam mengenali fitur visual penting. Selain itu, pengambilan gambar sebaiknya dilakukan dari jarak yang proporsional agar objek kursi tampak memenuhi frame; foto yang terlalu jauh dapat menyebabkan detail kerusakan tidak terlihat, sementara foto yang terlalu dekat bisa kehilangan konteks bentuk kursi secara keseluruhan.

Selain itu, latar belakang gambar sebaiknya tidak terlalu ramai. Kursi sebaiknya diposisikan dengan latar yang bersih atau minimal tidak tumpang tindih dengan objek lain, karena latar belakang yang terlalu padat dapat membingungkan model dan menyulitkan proses pelabelan. Jika terdapat banyak kursi dalam satu ruangan, fokus pengambilan foto sebaiknya diarahkan pada satu kursi target saja.

Dengan panduan ini, diharapkan data input ke model akan lebih sesuai

dengan asumsi pelatihan, sehingga akurasi deteksi di lapangan dapat meningkat. Selain itu, panduan ini juga bertujuan untuk meminimalkan potensi kesalahan pengguna (user error) dalam mengambil gambar, yang jika tidak dikontrol dapat menyebabkan model gagal mendeteksi meskipun pola kerusakan telah dipelajari dengan baik.

2.3 Roboflow

Gambar 2. 2 Logo Roboflow

Kinerja model deep learning seperti CNN sangat bergantung pada kualitas dan kuantitas dataset yang digunakan untuk pelatihan. Untuk kasus deteksi objek atau klasifikasi, proses penandaan data (pelabelan) secara akurat merupakan tahap krusial yang dapat menyita waktu dan rawan kesalahan jika dilakukan manual. Inilah area di mana platform Roboflow berperan penting dalam mempercepat dan mempermudah pengolahan dataset citra. Roboflow adalah platform berbasis web yang menyediakan alat untuk anotasi data citra, augmentasi, dan manajemen dataset secara terpadu. Melalui Roboflow, pengguna dapat mengunggah sekumpulan gambar dan melakukan *labeling* objek pada gambar-gambar tersebut menggunakan antarmuka grafis yang intuitif. Platform ini mendukung pembuatan *bounding box* secara manual untuk menandai objek atau area tertentu pada citra, serta penetapan kelas label untuk setiap objek tersebut.

Selain anotasi manual, Roboflow menyediakan berbagai opsi augmentasi data secara otomatis, seperti rotasi, flipping, penyesuaian kecerahan/kontras, penambahan *noise*, dan lain-lain. Augmentasi ini efektif untuk memperbanyak data

secara sintetis guna mengurangi *overfitting* dan membuat model lebih tangguh terhadap variasi kondisi di lapangan. Penelitian oleh *Dwibedi et al.* (2021) menunjukkan bahwa pendekatan sintesis data otomatis *cut-and-paste* dapat menghasilkan dataset instansi obyek secara cepat, dengan kinerja model yang mendekati pelatihan menggunakan data nyata sepenuhnya. Hal tersebut menggarisbawahi pentingnya teknik augmentasi dan otomasi dalam persiapan data deteksi objek. Roboflow mengadopsi prinsip serupa dengan menyediakan berbagai fitur augmentasi sekali klik, sehingga peneliti dapat bereksperimen dengan beragam transformasi data tanpa perlu menulis kode khusus.

Keunggulan lain dari Roboflow adalah kemampuannya mengonversi format anotasi dataset ke berbagai skema yang dibutuhkan oleh model atau *framework* berbeda. Misalnya, peneliti dapat mengunggah data berformat Pascal VOC lalu mengunduhnya kembali dalam format YOLO, COCO JSON, TensorFlow TFRecord, dan lain sebagainya. Roboflow bertindak sebagai *universal conversion tool* untuk dataset visi komputer, memungkinkan impor dan ekspor dalam lebih dari 30 format populer. Dengan demikian, integrasi antara dataset dan kerangka kerja/model yang dipilih menjadi lebih mudah; peneliti tidak perlu membuat skrip konversi untuk setiap format secara manual. Kemudahan ini menghemat waktu pengembangan sehingga perhatian dapat difokuskan pada eksperimen model daripada terjerat kerumitan praproses data. Secara keseluruhan, penggunaan Roboflow dalam penelitian ini dimaksudkan untuk mendapatkan *pipeline* pengolahan data yang efisien dan terorganisir. Dataset citra kerusakan aset Universitas Ma Chung dikelola melalui Roboflow – mulai dari tahap unggah data mentah, pemberian label area kerusakan, augmentasi citra untuk memperkaya variasi, hingga mengekspor data akhir dalam format yang kompatibel dengan model YOLOv8 yang akan dilatih. Platform ini memastikan konsistensi anotasi dan kualitas dataset, yang pada akhirnya diharapkan meningkatkan performa model deteksi kerusakan aset yang dikembangkan.

2.4 Android Studio

Android Studio merupakan Integrated Development Environment (IDE) resmi yang dikembangkan oleh Google untuk membangun aplikasi Android. IDE ini berbasis IntelliJ IDEA dan dilengkapi dengan berbagai fitur yang mendukung proses pengembangan aplikasi secara komprehensif, mulai dari penulisan kode, debugging, hingga proses kompilasi dan distribusi aplikasi ke Google Play Store. Android Studio menyediakan emulator, visual layout editor, integrasi dengan Firebase, serta dukungan penuh untuk Android SDK dan Android Jetpack, yang mempermudah pengembangan aplikasi Android secara modern dan efisien (Google Developers, 2023).

Dalam konteks penelitian ini, meskipun pengembangan utama aplikasi dilakukan menggunakan Flutter yang lintas platform, Android Studio tetap digunakan sebagai lingkungan pengembangan utama untuk keperluan pengujian dan debugging di perangkat Android. Android Studio kompatibel dengan plugin Flutter dan Dart, sehingga memungkinkan pengembang menjalankan simulasi aplikasi langsung di emulator Android atau perangkat fisik. Selain itu, Android Studio menyediakan tools analitik dan pelacakan performa aplikasi (Android Profiler) yang membantu dalam optimalisasi runtime, penggunaan memori, serta mendeteksi bottleneck performa yang mungkin timbul saat aplikasi menjalankan proses inferensi model CNN. Keunggulan ini menjadikan Android Studio sebagai alat pendukung penting dalam pengembangan aplikasi mobile pendeteksi kerusakan aset yang efisien dan stabil.

2.5 Flutter

Flutter adalah framework open-source yang dikembangkan oleh Google untuk membangun aplikasi mobile secara lintas platform (cross-platform). Berbeda dari pendekatan natif tradisional yang mengharuskan pengembangan terpisah untuk Android (Java/Kotlin) dan iOS (Swift/Objective-C), Flutter memungkinkan penulisan satu basis kode (menggunakan bahasa Dart) yang dapat dikompilasi langsung menjadi aplikasi Android maupun iOS. Hal ini tentunya sangat menguntungkan dalam hal produktivitas dan konsistensi fitur antar platform. Flutter menyediakan serangkaian widget UI bawaan yang kaya dan fleksibel, sehingga

pengembang dapat merancang antarmuka pengguna yang elegan dengan performa mendekati native. Mesin rendering Flutter (berbasis Skia) menggambar UI langsung ke kanvas tanpa memerlukan bridge ke kode native, yang membuat animasi dan respons aplikasi menjadi halus. Salah satu alasan pemilihan Flutter dalam penelitian ini adalah efisiensi pengembangan lintas platform dan kematangan ekosistemnya. Dengan Flutter, aplikasi pendeteksi kerusakan aset dapat dikembangkan dan diuji dengan cepat pada berbagai perangkat tanpa perlu menulis ulang kode untuk tiap sistem operasi. Selain itu, Flutter dikenal mudah diintegrasikan dengan pustaka-pustaka pihak ketiga, termasuk yang mendukung fungsi machine learning. Menurut dokumentasi resmi Google (2022), Flutter memiliki kompatibilitas yang baik untuk integrasi model pembelajaran mesin.

Integrasi ini biasanya dilakukan melalui mekanisme plugin atau packages. Contohnya, terdapat plugin seperti tflite (TensorFlow Lite plugin) yang memungkinkan aplikasi Flutter menjalankan model TensorFlow Lite (format terkompresi dari model deep learning yang dioptimalkan untuk mobile) secara lokal di perangkat. Dengan plugin tersebut, model CNN yang telah dilatih untuk mendeteksi kerusakan aset dapat dimuat ke dalam aplikasi dan melakukan inferensi langsung menggunakan kamera smartphone. Proses inferensi lokal ini penting untuk mencapai deteksi secara real-time tanpa ketergantungan koneksi internet, sekaligus mengamankan data karena citra tidak perlu diunggah ke server eksternal.

Keunggulan Flutter lainnya adalah dukungan komunitas dan dokumentasi yang luas. Banyak contoh implementasi dan best practices yang tersedia secara daring, termasuk untuk kasus penggunaan yang relevan seperti menampilkan *live camera feed*, melakukan pemrosesan gambar di background, hingga optimasi performa aplikasi agar tetap interaktif saat menjalankan beban komputasi (misalnya inferensi CNN). Flutter juga terus berkembang menambahkan dukungan ke platform lain (seperti Web dan desktop), sehingga aplikasi yang dibangun memiliki potensi perluasan di masa depan . Dalam konteks penelitian ini, Flutter memberikan landasan yang baik untuk mengembangkan aplikasi mobile pendeteksi kerusakan aset yang andal: antarmuka pengguna dapat dibuat intuitif bagi petugas inventaris, akses kamera dapat diintegrasikan dengan mudah, dan model AI (YOLOv8) hasil pelatihan dapat disematkan dengan mulus ke dalam aplikasi.

2.6 Penelitian Terdahulu

Pengembangan sistem pendeteksi kerusakan aset berbasis AI dan mobile merupakan topik yang memadukan dua disiplin: visi komputer (deep learning untuk deteksi objek) dan rekayasa perangkat lunak mobile. Beberapa penelitian terdahulu telah dilakukan di masing-masing bidang tersebut maupun kombinasi keduanya, yang menjadi acuan dan pembanding bagi penelitian ini. Deteksi kerusakan objek dengan Deep Learning. Zhang et al. (2019) melakukan studi mengenai deteksi kerusakan pada furnitur menggunakan metode deep learning. Dalam penelitian tersebut, mereka memanfaatkan model CNN untuk mengidentifikasi kerusakan fisik pada furnitur (misalnya retak, atau patah pada kursi). Model dilatih menggunakan dataset citra furnitur yang mencakup kondisi rusak dan tidak rusak, kemudian dievaluasi kemampuannya dalam membedakan kedua kondisi tersebut. Hasil penelitian Zhang dkk. menunjukkan bahwa pendekatan CNN mampu mendeteksi kerusakan furnitur dengan tingkat akurasi tinggi, membuktikan efektivitas deep learning dalam tugas inspeksi visual aset fisik (Zhang et al., 2019). Penelitian ini memberikan dasar bahwa fitur-fitur visual kerusakan (seperti tekstur retakan atau deformasi bentuk) dapat ditangkap oleh model CNN secara otomatis, sehingga inspeksi kerusakan dapat diotomasi dengan reliabilitas yang baik. Keterbatasan dari studi Zhang et al. adalah fokus obyek yang masih terbatas (hanya jenis furnitur tertentu) dan tidak dibahasnya implementasi pada perangkat bergerak; namun demikian, temuan mereka menjadi landasan penting bahwa deteksi kondisi aset dengan CNN adalah mungkin dan akurat. Sistem manajemen aset berbasis mobile dengan deteksi objek.

Lee & Park (2021) mengembangkan sebuah sistem manajemen aset yang terintegrasi dengan object detection secara real-time di perangkat mobile. Riset mereka diarahkan untuk kebutuhan inventaris aset di lingkungan perusahaan, di mana aplikasi mobile digunakan petugas untuk memindai aset (misalnya peralatan kantor) dan secara otomatis mendeteksi serta mengenali objek tersebut. Dalam implementasinya, Lee & Park memanfaatkan model deteksi objek berbasis CNN yang ditanamkan ke dalam smartphone, sehingga kamera ponsel dapat mengenali item inventaris hanya dengan mengarahkannya ke objek tersebut. Sistem ini secara otomatis mencatat informasi aset yang terdeteksi beserta kondisinya ke dalam basis

data manajemen inventaris. Hasil pengujian menunjukkan bahwa integrasi teknologi deteksi objek pada aplikasi mobile mampu meningkatkan efisiensi pendataan aset dibandingkan cara manual (Lee & Park, 2021). Petugas tidak lagi harus mengidentifikasi dan memasukkan data setiap item secara manual, karena aplikasi dapat mengenali objek dan memasok detailnya secara instan. Studi ini relevan dengan penelitian Tugas Akhir karena membuktikan bahwa kombinasi mobile app dan on-device CNN dapat bekerja dengan baik untuk tugas pengelolaan aset. Perbedaannya, fokus Lee & Park adalah pada identifikasi jenis aset dan keberadaannya, sedangkan penelitian ini lebih fokus pada pendeteksian kondisi/kerusakan aset. Meskipun begitu, pendekatan teknis mereka dalam mengintegrasikan model deteksi di mobile menjadi referensi berharga bagi implementasi sistem kita. Otomatisasi pelacakan aset di institusi pendidikan. Smith et al. (2020) membahas tantangan dan solusi dalam penerapan pelacakan aset secara otomatis di lingkungan institusi pendidikan. Walaupun pendekatan yang dibahas tidak semuanya berbasis visi komputer (beberapa mencakup penggunaan RFID, barcode, dan IoT), studi tersebut menekankan bahwa proses tradisional pendataan aset kampus sering terkendala oleh human error dan kurang real-time. Mereka mengidentifikasi bahwa kesalahan pencatatan, keterlambatan pembaruan data, dan keterbatasan tenaga untuk inspeksi rutin merupakan masalah umum dalam manajemen aset universitas (Smith et al., 2020). Smith dkk. kemudian menganalisis berbagai solusi, termasuk kemungkinan memanfaatkan kamera dan AI untuk memonitor aset. Rekomendasi mereka mengarah pada perlunya sistem terotomasi yang dapat mengurangi intervensi manual dalam pengecekan kondisi dan lokasi aset. Temuan dari Smith et al. ini memberikan justifikasi kuat bagi penelitian kita: sebuah aplikasi mobile dengan dukungan CNN untuk deteksi kerusakan aset secara real-time akan sangat sejalan dengan kebutuhan mengurangi beban manual dan meningkatkan akurasi pendataan aset di institusi pendidikan seperti Universitas Ma Chung. Integrasi deteksi kerusakan berbasis AI pada domain lain. Di luar konteks furnitur dan inventaris kampus, terdapat pula penelitian lain yang mengintegrasikan deteksi kerusakan berbasis CNN ke perangkat mobile. Misalnya, Maeda et al. (2018) mengembangkan aplikasi smartphone untuk mendeteksi kerusakan pada jalan raya (seperti lubang atau retakan di aspal) menggunakan citra yang diambil

dari ponsel yang dipasang pada kendaraan. Model CNN dilatih untuk mengenali berbagai jenis kerusakan jalan dan diaplikasikan pada perangkat mobile sehingga dapat melakukan deteksi secara langsung selama kendaraan bergerak. Hasilnya menunjukkan sistem mampu mengenali kerusakan jalan secara real-time dengan akurasi tinggi, membantu otoritas dalam melakukan pemeliharaan jalan tepat waktu (Maeda et al., 2018). Walaupun domainnya berbeda, pendekatan Maeda dkk. memperlihatkan bahwa perangkat bergerak dapat diandalkan sebagai platform inference bagi model CNN untuk tugas deteksi kerusakan objek di lapangan. Ini mengindikasikan bahwa rancangan sistem dalam penelitian kita (menggunakan smartphone untuk deteksi kerusakan furnitur) berada dalam tren yang sama dengan berbagai aplikasi "AI on Edge" yang memanfaatkan kecerdasan buatan langsung di perangkat pengguna. Secara keseluruhan, studi pustaka di atas menunjukkan perkembangan di mana deep learning telah diimplementasikan untuk mendeteksi kerusakan atau melakukan tracking aset, dan bahwa integrasi ke dalam aplikasi mobile juga telah mulai dilakukan oleh peneliti terdahulu. Penelitian ini akan melanjutkan upaya tersebut dengan fokus yang lebih spesifik dan terintegrasi: mendeteksi kerusakan aset (furnitur) dalam skala lingkungan universitas dengan memanfaatkan platform mobile. Perbedaan utama terletak pada objek dan skenario penggunaan, namun prinsip dasarnya sejalan dengan tren bahwa computer vision berbasis CNN dapat dijalankan di perangkat mobile untuk solusi manajemen aset/infrastruktur yang lebih cerdas.

2.7 Keras

Keras adalah API tingkat tinggi untuk membangun dan melatih jaringan saraf tiruan, yang berjalan di atas framework TensorFlow. Keras dirancang dengan filosofi user-friendly, modular, dan mudah diperluas, yang memungkinkan peneliti dan pengembang untuk merancang model CNN secara cepat tanpa harus menulis kode numerik kompleks.

Dalam penelitian ini, Keras digunakan untuk membangun arsitektur CNN secara modular. Misalnya, peneliti dapat membuat model deteksi dengan mudah menggunakan baris kode seperti Conv2D, MaxPooling2D, dan Dense, sambil memantau metrik pelatihan seperti akurasi dan loss.

2.8 Python

Python merupakan bahasa pemrograman tingkat tinggi yang sangat populer di kalangan pengembang dan peneliti dalam bidang Artificial Intelligence (AI) dan Machine Learning. Keunggulan Python terletak pada sintaksnya yang sederhana dan mudah dipahami, yang membuatnya sangat cocok untuk kebutuhan riset, eksplorasi data, serta pengembangan sistem berbasis pembelajaran mesin. Selain itu, Python memiliki ekosistem pustaka yang sangat kaya untuk mendukung pengembangan AI, seperti TensorFlow, Keras, PyTorch, Scikit-Learn, NumPy, Pandas, dan OpenCV. Dengan berbagai pustaka tersebut, pengembang dapat membangun, melatih, dan mengevaluasi model pembelajaran mesin maupun deep learning dengan efisien dan fleksibel (Van Rossum & Drake, 2009).

Dalam konteks pengembangan model CNN untuk deteksi kerusakan aset, Python berperan sebagai bahasa utama yang digunakan dalam proses pelatihan model, pengolahan dataset, hingga konversi model ke format yang dapat dijalankan di perangkat mobile. Selain mendukung pengembangan teknis, Python juga kompatibel dengan berbagai platform seperti Google Colab, Jupyter Notebook, hingga sistem operasi Linux dan Windows. Dukungan komunitas Python yang sangat besar juga mempercepat penyelesaian masalah teknis yang mungkin muncul selama pengembangan, karena tersedia banyak dokumentasi, forum diskusi, dan tutorial daring (Lutz, 2013).

2.9 Google Colaboratory

Google Colaboratory, atau lebih dikenal dengan Google Colab, merupakan platform komputasi berbasis cloud yang memungkinkan pengguna untuk menjalankan kode Python secara langsung melalui browser tanpa memerlukan instalasi perangkat lunak tambahan. Dikembangkan oleh Google Research, Colab menyediakan lingkungan pemrograman interaktif dengan dukungan akses gratis ke perangkat keras komputasi seperti GPU (Graphics Processing Unit) dan TPU (Tensor Processing Unit), menjadikannya sangat ideal untuk pekerjaan yang memerlukan daya komputasi tinggi, termasuk pelatihan model deep learning (Google Colab, n.d.).

Salah satu keunggulan utama dari Google Colab terletak pada integrasinya

yang lancar dengan Google Drive, sehingga pengguna dapat menyimpan dan mengelola file notebook secara daring serta melakukan kolaborasi secara waktu nyata. Platform ini juga mendukung berbagai pustaka populer dalam pengembangan AI dan visi komputer seperti TensorFlow, Keras, PyTorch, dan OpenCV, yang semuanya dapat langsung digunakan di dalam notebook. Kemudahan ini menjadikan Google Colab sebagai pilihan tepat bagi mahasiswa, peneliti, dan praktisi yang memerlukan lingkungan komputasi yang mudah diakses, fleksibel, dan mendukung kolaborasi (Bisong, 2019).

2.10 Proses Bisnis

Sebelum solusi baru dikembangkan, perlu dipahami proses dan sistem eksisting yang digunakan oleh Bagian Aset Universitas Ma Chung. Berdasarkan wawancara informal dengan pihak Bagian Aset, diperoleh gambaran bahwa petugas lapangan melakukan inspeksi berkala ke ruangan-ruangan kampus. Jika ditemukan kursi rusak, petugas akan mengambil foto menggunakan ponsel dan mengunggahnya ke aplikasi pendataan aset. Aplikasi tersebut merupakan aplikasi web internal sederhana yang dibangun oleh SIPUSDA, berfungsi untuk mengunggah foto dan memasukkan deskripsi kerusakan secara manual.

Data yang disimpan dalam aplikasi mencakup identitas aset, lokasi, tanggal, foto kerusakan, dan keterangan teks yang dituliskan oleh petugas. Namun, aplikasi tersebut tidak melakukan analisis otomatis; seluruh identifikasi jenis kerusakan ditulis sendiri oleh petugas. Setelah laporan kerusakan dicatat dalam aplikasi, proses selanjutnya adalah peninjauan oleh atasan atau pihak yang berwenang. Berdasarkan laporan tersebut, diputuskan apakah kursi perlu diperbaiki atau diganti. Proses ini seluruhnya masih bergantung pada tenaga manusia, mulai dari deteksi, deskripsi, hingga pembaruan status.

Dari gambaran di atas, dapat diidentifikasi beberapa keterbatasan (gap) pada sistem eksisting. Pertama, sistem yang digunakan masih memiliki tingkat otomasi yang sangat minim. Aplikasi lama tidak dilengkapi dengan kemampuan kecerdasan buatan untuk mengklasifikasikan kerusakan; seluruh interpretasi bergantung pada pengamatan subjektif petugas. Hal ini berisiko menyebabkan keterlambatan (karena waktu yang dibutuhkan untuk menuliskan keterangan) dan perbedaan

persepsi antar petugas. Kedua, sistem tersebut memiliki ketergantungan tinggi terhadap input manual pengguna. Kualitas data sangat bergantung pada ketelitian petugas dalam menulis deskripsi. Apabila terjadi kesalahan atau kelalaian dalam pengisian, data kerusakan menjadi tidak akurat. Misalnya, meskipun foto jelas menunjukkan kaki kursi yang patah, deskripsi yang kurang tepat dapat menyebabkan kesalahan pemahaman oleh pihak manajemen.

Ketiga, sistem tidak menyediakan standarisasi dalam klasifikasi jenis kerusakan. Idealnya, tipe-tipe kerusakan seperti patah pada kaki kursi, sobek pada permukaan jok, atau kerusakan pada sambungan, harus distandarkan dalam bentuk pilihan kategori yang baku. Namun, pada sistem lama, semua data kerusakan dicatat sebagai teks bebas tanpa kontrol istilah. Kondisi ini menyulitkan proses analisis dan pengelompokan data kerusakan dalam skala besar. Keempat, dari sisi efisiensi dan skalabilitas, sistem manual ini sangat terbatas. Dalam situasi di mana jumlah aset meningkat atau frekuensi inspeksi lebih tinggi, sistem akan kewalahan. Sebagai contoh, dalam satu sesi inspeksi, petugas bisa menemukan puluhan kursi rusak dan harus mencatat masing-masing satu per satu secara manual. Pendekatan ini tidak efisien dan sulit diandalkan dalam jangka panjang.

Solusi yang diusulkan dalam penelitian ini dirancang untuk menutup berbagai gap tersebut. Dengan memanfaatkan aplikasi mobile berbasis visi komputer, petugas cukup mengambil gambar kursi yang dicurigai rusak, lalu aplikasi akan secara otomatis menampilkan prediksi kategori kerusakan – seperti kerusakan pada kaki, sandaran, atau sobekan pada jok – yang dihasilkan oleh model CNN. Prediksi tersebut akan langsung tersimpan dalam bentuk data terstruktur, bukan lagi teks bebas, sehingga meminimalkan variasi penulisan dan meningkatkan konsistensi data. Risiko kesalahan pengguna (user error) dalam pendataan pun berkurang, karena petugas tidak perlu lagi mengetik deskripsi panjang; mereka cukup memverifikasi apakah hasil prediksi model sudah sesuai. Jika tidak sesuai, sistem tetap memberikan opsi untuk memilih kategori kerusakan secara manual dari daftar yang telah ditentukan.

Lebih jauh, penggunaan model CNN juga memungkinkan adanya standarisasi istilah kerusakan. Kategori yang digunakan oleh model merupakan hasil pelatihan dari data yang telah dianotasi sebelumnya, sehingga output

klasifikasi selalu konsisten dengan label yang ditetapkan. Hal ini akan sangat membantu dalam pelacakan data historis, seperti menghitung jumlah kursi yang mengalami kerusakan tertentu selama satu semester. Selain itu, sistem juga mendukung integrasi lebih lanjut ke dalam infrastruktur yang sudah dimiliki SIPUSDA. Foto dan hasil deteksi dapat langsung dikirim ke server untuk dicatat, memungkinkan data kerusakan dapat diakses secara real-time oleh pihak manajemen, termasuk pimpinan universitas. Dengan adanya API, proses pelaporan menjadi lebih cepat, dan respons terhadap perbaikan dapat dilakukan dengan segera.

Meskipun belum banyak penelitian spesifik yang membahas sistem otomatisasi deteksi kerusakan aset furnitur kampus, pendekatan serupa telah diterapkan di berbagai bidang seperti smart campus dan facility management. Salah satu contohnya adalah studi oleh Prasetyo et al. (2020) mengenai pemeliharaan gedung pintar (smart building maintenance) yang menggunakan sensor dan pengenalan citra (image recognition). Studi tersebut menunjukkan peningkatan efisiensi hingga 30% dibandingkan metode inspeksi manual. Temuan ini memperkuat argumen bahwa penerapan teknologi otomasi berbasis visi komputer dalam deteksi kondisi aset sangat relevan dan menjanjikan, terutama di institusi pendidikan seperti kampus.

2.11 Aplikasi Sebelumnya

Pada tahap sebelumnya, Universitas Ma Chung telah menggunakan aplikasi Mobile Asset Information System (MAIS) sebagai sarana untuk melakukan pencatatan dan monitoring kondisi aset kampus. Aplikasi ini berfungsi sebagai pendukung kegiatan administrasi, khususnya dalam proses pencatatan manual hasil inspeksi aset yang dilakukan oleh petugas lapangan.

Setiap aset diberi identitas berupa kode atau QR code yang dapat dipindai melalui perangkat mobile, kemudian data kondisi aset, catatan, dan bukti foto dimasukkan secara manual ke dalam sistem. Meskipun aplikasi ini sudah membantu mengurangi ketergantungan pada pencatatan berbasis kertas, sistem lama masih memiliki sejumlah keterbatasan, terutama dalam hal validasi data, keakuratan informasi, serta efisiensi proses monitoring. Oleh karena itu, diperlukan pengembangan aplikasi baru dengan fitur otomatisasi verifikasi agar proses

pencatatan aset menjadi lebih akurat, terintegrasi, dan mampu mendukung pengambilan keputusan secara real-time.

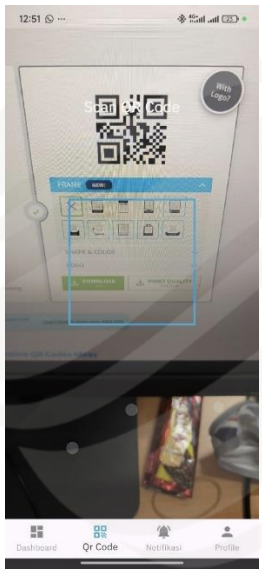
Gambar 2. 3 Layar Login Aplikasi MAIS

Pada tahap awal, pengguna harus melakukan login ke aplikasi Mobile Asset Information System (MAIS) Universitas Ma Chung. Layar login meminta username dan password sebagai autentikasi pengguna sebelum mengakses fitur-fitur monitoring aset. Proses ini memastikan hanya petugas atau user yang berwenang yang dapat masuk ke sistem untuk melakukan pemantauan aset. Setelah berhasil login, pengguna diarahkan ke menu utama aplikasi untuk memilih fungsi yang diinginkan, termasuk fungsi monitoring aset.

Gambar 2. 4 Layar Dashboard Aplikasi MAIS

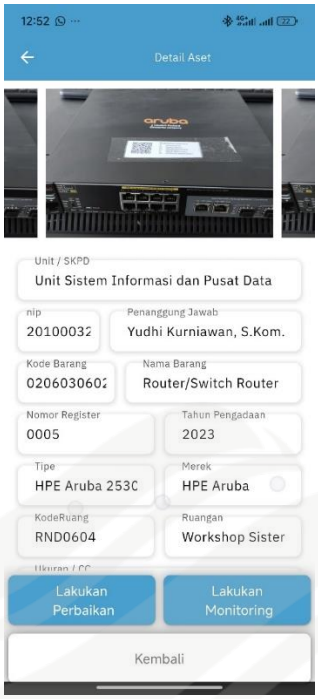
Pada menu dashboard, ditampilkan berbagai modul pengelolaan aset secara mobile. Terdapat ikon List Monitoring dan Jadwal Monitoring yang berkaitan

dengan aktivitas inspeksi rutin aset, serta fitur lain seperti Pengaduan Aset Rusak dan Perbaikan Aset. Untuk memulai proses monitoring aset, pengguna dapat memilih menu QR Code pada bottom navigation aplikasi. Fitur ini memungkinkan petugas memindai kode QR yang terpasang pada aset, sehingga identitas aset dapat diperoleh secara instan tanpa perlu input manual yang rawan kesalahan. Hal ini sejalan dengan praktik terbaik manajemen aset dimana pemanfaatan barcode atau QR code mampu mengotomatiskan identifikasi aset dan mengurangi human error.



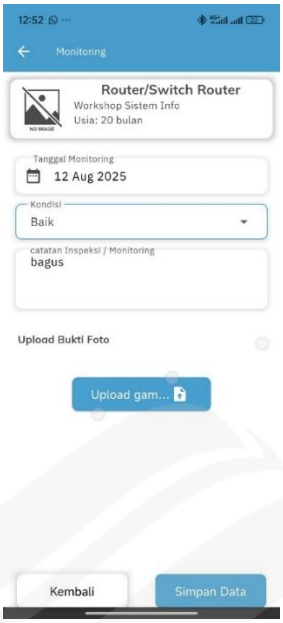
Gambar 2. 5 Layar Scan QR Code Aplikasi MAIS

Setelah memilih mode pemindaian, aplikasi mengaktifkan kamera untuk scan QR code. Petugas mengarahkan kamera ke label QR di aset; sistem kemudian membaca kode unik tersebut dan mencocokkannya dengan database aset. Jika sesuai, aplikasi akan menampilkan informasi aset yang bersangkutan secara otomatis. Metode ini mempercepat akses data dan memastikan aset yang diinspeksi benar dan sesuai catatan, menggantikan pencatatan manual yang lambat dan rentan salah identifikasi . Dengan scan QR code, informasi dasar aset dapat ditarik secara real-time, pemindaian QR mampu menampilkan data aset secara instan dengan akurasi sangat tinggi (99,9%) . Pada proses manual sebelumnya (misalnya menggunakan daftar aset cetak atau spreadsheet), petugas harus mencari data aset secara manual, sehingga penggunaan QR code pada aplikasi mobile ini adalah langkah maju untuk efisiensi. Meskipun demikian, langkah ini masih memerlukan interaksi manual oleh petugas (memindai satu per satu), dan sistem belum melakukan verifikasi lain selain pengecekan kode aset.



Gambar 2. 6 Layar Detail Aset Aplikasi MAIS

Setelah QR code berhasil dipindai, aplikasi menampilkan Detail Aset. Pada layar ini tertera informasi lengkap aset, antara lain: unit pemilik (Unit/SKPD), penanggung jawab, kode barang, nama barang, nomor register, tahun pengadaan, merek/tipe, lokasi (ruangan), hingga umur aset. Tampilan detail ini berfungsi untuk verifikasi visual oleh petugas bahwa aset yang akan dimonitor memang benar (misalnya memastikan nama dan lokasi sesuai dengan aset fisik di lapangan). Apabila aset yang dimaksud memerlukan tindakan, tersedia dua pilihan: Lakukan Monitoring untuk pencatatan kondisi rutin, atau Lakukan Perbaikan jika ditemukan kerusakan yang membutuhkan tindak lanjut. Dalam konteks monitoring berkala, petugas akan menekan tombol Lakukan Monitoring. Seluruh proses ini masih berjalan manual dalam arti petugas yang mengambil inisiatif memindai dan memutuskan tindakan; sistem belum memberikan prompt otomatis atau pengecekan selain menampilkan data aset.



Gambar 2. 7 Formulir Monitoring Aset Aplikasi MAIS

Gambar 2.6 menunjukan formulir monitoring aset yang harus diisi petugas secara manual. Pada formulir Monitoring ini, terdapat beberapa komponen input: (1) Tanggal Monitoring – biasanya terisi otomatis dengan tanggal hari ini, namun bisa diedit secara manual; (2) Kondisi – pilihan kondisi aset (misalnya Baik, Rusak Ringan, Rusak Berat) yang dipilih dari daftar; (3) Catatan Inspeksi – teks bebas untuk mencatat hasil pengamatan atau tindakan; dan (4) Upload Bukti Foto – fitur unggah foto kondisi fisik aset sebagai evidensi. Petugas melakukan penilaian kondisi secara subjektif dan memasukkan catatan sesuai inspeksi lapangan. Selanjutnya, petugas dapat mengambil foto aset (misalnya foto perangkat dan nomor seri) lalu mengunggahnya melalui tombol Upload Gambar. Setelah semua data dirasa cukup, tombol Simpan Data ditekan untuk menyimpan hasil monitoring.

Form pencatatan kondisi di atas merefleksikan proses monitoring yang manual: kualitas data sangat bergantung pada ketelitian petugas dalam mengisi. Potensi kelemahan pada tahap ini antara lain: tidak ada validasi otomatis untuk memastikan semua input valid dan masuk akal. Sebagai contoh, sistem memungkinkan tanggal monitoring diubah secara manual; jika petugas keliru atau lalai, tanggal bisa tidak akurat. Kasus semacam ini tampak pada data yang tersimpan dengan tanggal anomali. Input teks catatan juga tidak tervalidasi panjang maupun formatnya – petugas bisa saja memasukkan informasi minimal (“bagus” saja) tanpa detail, dan sistem tetap menerimanya. Bahkan kondisi aset ditentukan

hanya berdasarkan observasi petugas; tidak ada sensor atau mekanisme verifikasi otomatis yang memastikan kondisi "Baik" memang benar (misalnya perangkat benar-benar menyala atau berfungsi). Dengan demikian, kesalahan manual sangat mungkin terjadi tanpa terdeteksi. Hal ini sejalan dengan temuan umum bahwa proses pencatatan aset secara manual rentan terhadap kesalahan manusia dalam entri data . Prosedur seperti di atas juga mengandalkan kedisiplinan petugas, sebab sistem belum memberikan pengecekan otomatis (misalnya mencegah tanggal tidak logis atau mencegah kolom wajib kosong). Tanpa adanya kontrol tambahan, kualitas data monitoring sangat bergantung pada keakuratan input manual, yang menurut Hendrikson dapat menurunkan kualitas informasi secara keseluruhan

Gambar 2. 8 Layar List Data Monitoring Aplikasi MAIS

Setiap baris memuat identitas aset (kode dan nama barang, lokasi, umur) beserta tanggal terakhir dimonitor. Fungsi daftar ini adalah memudahkan petugas meninjau riwayat monitoring dan memastikan tiap aset terpantau secara berkala. Terlihat pada contoh, aset "Router/Switch Router" di Workshop SI telah dimonitor pada 12-08-2025, dan sebelumnya pada 17-10-2024. Demikian pula aset PC tercatat terakhir dimonitor 16-08-2024. Namun, tampak anomali pada salah satu entri: sebuah aset PC memiliki tanggal monitoring 30-11-0001. Tanggal yang tidak wajar ini menandakan data tidak valid yang tersimpan akibat ketiadaan validasi saat input – besar kemungkinan terjadi kesalahan input atau bug (misal pengguna lupa memilih tanggal sehingga sistem menyimpan nilai default tahun 0001). Contoh

tersebut menggarisbawahi kelemahan proses manual: tanpa validasi otomatis, kesalahan input dapat langsung masuk ke database dan mengurangi akurasi informasi aset. Dalam jangka panjang, akumulasi kesalahan semacam ini akan menyulitkan pelacakan aset dan pengambilan keputusan. Penelitian Kusumojati & Mediawati (2024) mencatat bahwa tanpa sistem informasi aset internal, institusi pendidikan tinggi mengalami kesulitan menelusuri data aset, dan rentan terhadap kesalahan pencatatan serta kehilangan data . Hal ini menunjukkan perlunya sistem yang lebih terstruktur untuk memastikan data aset (termasuk data hasil monitoring) lebih terorganisir, akurat, dan mudah ditelusuri. Secara umum, berbagai studi dan laporan menyimpulkan bahwa penggunaan sistem manual atau terpisah (seperti Excel) dalam manajemen aset menyebabkan proses menjadi tidak efisien, rawan kesalahan, dan menyulitkan konsolidasi data.

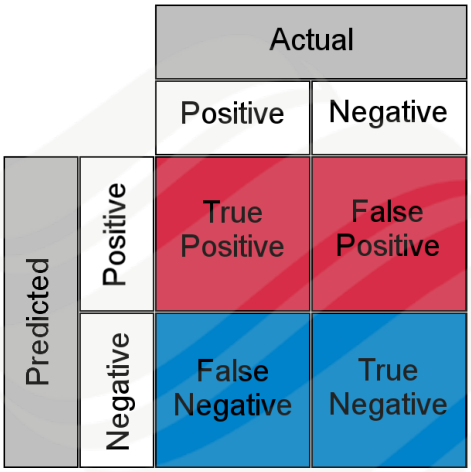
2.12 Evaluasi Kinerja

Evaluasi kinerja model deteksi objek merupakan langkah penting untuk memastikan bahwa model tidak hanya bekerja baik pada data pelatihan, tetapi juga memiliki performa yang andal pada data nyata. Dalam konteks model YOLO yang dikembangkan oleh Ultralytics, evaluasi dilakukan menggunakan berbagai metrik, antara lain akurasi, presisi, recall, dan F1-score yang dihitung berdasarkan confusion matrix (Ultralytics, 2024).aktual yang berhasil dideteksi model. Evaluasi performa juga mencakup pemantauan potensi overfitting. Apabila performa pada data latih jauh lebih tinggi dibanding data validasi/uji, maka model mungkin overfit. Teknik seperti early stopping dapat diterapkan untuk menghentikan pelatihan saat performa validasi mulai menurun, sehingga model final yang dipilih adalah checkpoint dengan kinerja terbaik pada data validasi. Dengan evaluasi menyeluruh menggunakan metrik-metrik di atas, peneliti dapat memastikan model CNN yang dikembangkan memenuhi ekspektasi sebelum diintegrasikan ke aplikasi.

2.12.1 Confusion Matrix

Confusion matrix atau matriks kebingungan adalah sebuah tabel yang menunjukkan perbandingan antara hasil prediksi model dengan label sebenarnya. Matriks ini terdiri dari empat komponen utama: True Positive (TP), False Positive

(FP), False Negative (FN), dan True Negative (TN). TP menunjukkan jumlah prediksi benar untuk kelas positif, FP menunjukkan jumlah prediksi salah untuk kelas positif (objek tidak ada, tapi diprediksi ada), FN menunjukkan objek yang seharusnya terdeteksi namun tidak terdeteksi, dan TN adalah jumlah prediksi benar untuk kelas negatif (Brownlee, 2020).



Gambar 2. 9 Contoh confusion matrix untuk klasifikasi biner (Brownlee, 2020)

Dari matriks kebingungan, beberapa metrik utama dapat dihitung untuk mengevaluasi performa model. Akurasi merupakan ukuran seberapa sering model membuat prediksi yang benar, baik untuk kelas positif maupun negatif. Metrik ini mencerminkan proporsi keseluruhan prediksi yang sesuai dengan label sebenarnya dan memberikan gambaran umum tentang kinerja model secara menyeluruh, meskipun kurang efektif jika data tidak seimbang (Hicks et al., 2022).

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Presisi mengukur seberapa tepat model dalam mengidentifikasi objek positif. Dalam hal ini, presisi menunjukkan seberapa banyak dari seluruh prediksi positif yang benar-benar relevan. Nilai presisi yang tinggi berarti model jarang membuat kesalahan dalam mendeteksi sesuatu yang tidak ada sebagai positif (Ultralytics, 2024). Rumus dari presisi adalah:

$$Presisi = \frac{TP}{TP + FP} \quad (2)$$

Sementara itu, recall atau sensitivitas menunjukkan seberapa banyak dari semua objek positif aktual yang berhasil terdeteksi oleh model. Dengan kata lain, recall mengukur kemampuan model untuk menemukan semua kasus positif dalam dataset. Nilai recall yang tinggi menunjukkan bahwa model mampu menangkap sebagian besar objek penting yang seharusnya terdeteksi (Hicks et al., 2022). Rumus recall adalah:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Untuk memperoleh keseimbangan antara presisi dan recall, digunakan metrik F1-score, yang merupakan gabungan keduanya dalam satu ukuran tunggal. F1-score sangat berguna ketika dibutuhkan kompromi antara ketepatan prediksi dan kemampuan mendeteksi sebanyak mungkin objek positif (Ultralytics, 2024). Rumus dari F1-score adalah:

$$F1 = \frac{2 * Presisi * Recall}{Presisi + Recall} = \frac{2TP}{2TP + FP + FN} \tag{4}$$

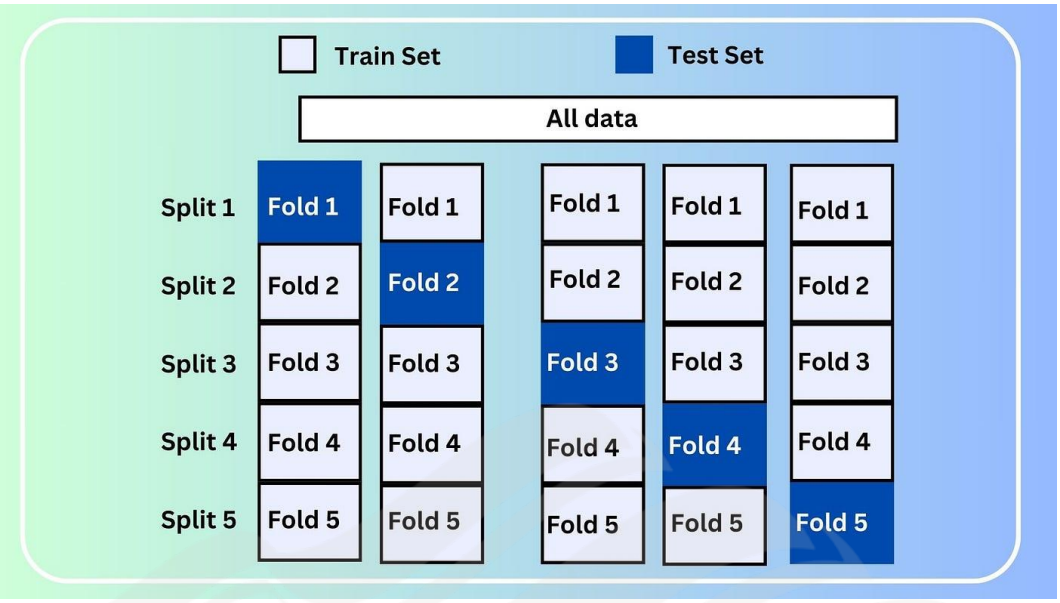
Evaluasi aplikasi mobile merupakan tahap penting dalam proses pengembangan aplikasi untuk memastikan bahwa sistem berjalan sesuai dengan tujuan dan kebutuhan pengguna. Salah satu metode evaluasi yang umum digunakan adalah pengujian fungsional berbasis black-box, yaitu pendekatan di mana penguji mengevaluasi aplikasi hanya berdasarkan input dan output, tanpa melihat struktur internal kode program. Menurut Sommerville (2011), metode black-box cocok untuk menguji fungsi-fungsi sistem dari perspektif pengguna akhir karena fokusnya adalah pada perilaku eksternal aplikasi. Dengan pendekatan ini, setiap fitur diuji melalui skenario penggunaan tertentu, seperti input gambar melalui kamera dan hasil deteksi kerusakan yang ditampilkan, guna mengetahui apakah keluaran sistem sesuai dengan ekspektasi.

2.12.2 K-Fold Cross Validation

K-Fold Cross Validation merupakan teknik validasi model yang membagi dataset menjadi K bagian (fold) yang sama besar untuk evaluasi berulang. Pada skenario ini digunakan 5-Fold Cross Validation, artinya data dibagi menjadi 5 fold dan model CNN dilatih serta divalidasi sebanyak lima kali secara bergantian: setiap kali satu fold digunakan sebagai data validasi, sementara empat fold lainnya digunakan untuk pelatihan. Proses ini diulang hingga seluruh fold pernah menjadi validasi satu kali (Arlot & Celisse, 2010). Selanjutnya, hasil evaluasi dari masing-masing fold dirata-rata untuk memperoleh estimasi kinerja model yang lebih stabil dan akurat (Zhang et al., 2021).

Pendekatan ini menghasilkan evaluasi yang robust, karena memanfaatkan seluruh data yang tersedia secara efektif — setiap sampel data digunakan baik untuk pelatihan maupun validasi — dibandingkan dengan metode holdout biasa yang hanya membagi satu kali (Kohavi, 1995). Metode ini sangat bermanfaat terutama pada kondisi jumlah data terbatas, karena mampu menguji model pada berbagai subset data sehingga mengurangi risiko bias terhadap satu pembagian data saja (Hastie, Tibshirani, & Friedman, 2009).

Selain itu, pendekatan ini membantu mencegah overfitting karena model divalidasi secara bergilir pada data yang tidak terlihat selama proses pelatihan, sehingga mensimulasikan kinerjanya pada data baru. Dalam konteks deteksi kerusakan aset menggunakan CNN, khususnya YOLOv8s, penerapan 5-fold cross validation meningkatkan kepercayaan terhadap kemampuan generalisasi model ke data-data yang bervariasi. Proses validasi silang ini mengevaluasi kemampuan model dalam mengenali pola kerusakan kursi (jamur, patah, roda rusak, dll.) pada berbagai kombinasi data pelatihan dan pengujian, sehingga memperkuat bukti bahwa model tidak hanya menghafal tetapi benar-benar belajar (Silva et al., 2025).



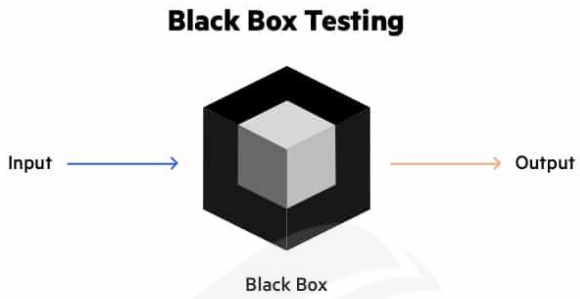
Gambar 2. 10 K-Fold Cross Validation

Gambar di atas menunjukkan mekanisme kerja K-Fold Cross Validation dengan parameter $K = 5$, yang artinya seluruh dataset dibagi menjadi 5 bagian (fold) yang berukuran kurang lebih sama. Proses validasi dilakukan sebanyak lima kali (*split 1* hingga *split 5*), di mana pada setiap iterasi satu fold digunakan sebagai data uji (test set) dan empat fold sisanya digunakan sebagai data latih (train set).

Pada *Split 1*, Fold 1 dipilih sebagai test set, sementara Fold 2 hingga Fold 5 digunakan sebagai train set. Selanjutnya, pada *Split 2*, giliran Fold 2 yang digunakan sebagai test set, dan Fold 1 serta Fold 3 sampai Fold 5 menjadi train set. Proses ini berlanjut hingga seluruh fold pernah menjadi test set tepat satu kali.

Tujuan dari pendekatan ini adalah untuk mengurangi risiko *bias* dalam evaluasi model, serta memberikan estimasi performa model yang lebih stabil. Karena setiap data digunakan untuk pelatihan dan pengujian secara bergantian, metode ini sangat cocok digunakan pada kondisi jumlah data terbatas. Setelah kelima iterasi selesai dilakukan, nilai metrik evaluasi seperti *precision*, *recall*, dan *mAP* dari masing-masing fold dapat dirata-rata untuk menghasilkan evaluasi model yang lebih akurat dan representatif terhadap keseluruhan dataset.

2.12.3 Black Box Testing



Gambar 2. 11 Black Box Testing

Black-box testing adalah metode pengujian perangkat lunak yang menilai fungsionalitas aplikasi dari sudut pandang pengguna tanpa memperhatikan struktur internal kode (Beizer, 1995; Sommerville, 2016). Penguji memberikan input dan membandingkan output aktual dengan output yang diharapkan untuk memastikan kesesuaian dengan spesifikasi (Myers et al., 2011). Teknik yang sering digunakan meliputi *equivalence partitioning*, *boundary value analysis*, dan *decision table testing* (Jorgensen, 2013).

Kelebihan pendekatan ini adalah dapat dilakukan tanpa pengetahuan teknis mendalam, efektif menemukan kesalahan logika maupun masalah antarmuka, serta sesuai untuk menguji aplikasi berbasis pengguna (Patton, 2005; Whittaker et al., 2012). Dalam konteks aplikasi pendeteksi kerusakan kursi, *black-box testing* digunakan untuk mengevaluasi alur kerja utama, seperti pengambilan gambar, klasifikasi kerusakan oleh model CNN, dan penyimpanan hasil deteksi, sehingga memastikan sistem berjalan stabil sesuai kebutuhan pengguna (Tian et al., 2018).

BAB III
METODOLOGI PENELITIAN

3.1 Alur Penelitian

Alur penelitian dimulai dengan pembuatan dataset, yang dilanjutkan dengan pengembangan dan evaluasi kinerja model seperti yang telah dilakukan pada penelitian sebelumnya. Tujuan pada penelitian ini adalah untuk membuat sistem prediksi hasil panen yang lebih akurat dibandingkan dengan penelitian sebelumnya.

Gambar 3. 1 Alur Penelitian

3.2 Studi Literatur

Pada tahap ini dilakukan studi pustaka terhadap penelitian-penelitian terdahulu yang relevan sebagai dasar dalam penyusunan dan pelaksanaan tugas akhir. Studi ini mencakup topik-topik utama seperti pemanfaatan Convolutional Neural Network (CNN) untuk deteksi objek, penggunaan platform Roboflow dalam manajemen dataset visi komputer, serta pengembangan aplikasi mobile berbasis Flutter dan Android Studio. Penelusuran literatur ini bertujuan untuk memahami metode, pendekatan, serta implementasi teknis yang telah terbukti efektif, sehingga dapat dijadikan acuan dalam pengembangan sistem pendeteksi kerusakan aset berbasis AI. Selain itu, studi pustaka ini juga memberikan landasan teori yang kuat untuk mendukung pemilihan metode dan teknologi dalam penelitian.

Tabel 3. 1 Studi Pustaka Peneltian Terdahulu

No	Topik	Pengetahuan	Temuan
1	Convolutional Neural Network (CNN) untuk deteksi objek	Apa itu CNN dan bagaimana CNN digunakan dalam klasifikasi dan deteksi objek?	CNN adalah jaringan saraf tiruan yang digunakan untuk ekstraksi fitur dan klasifikasi citra. CNN efektif dalam tugas deteksi objek seperti pada YOLO dan R-CNN, dengan akurasi dan kecepatan tinggi. (Krizhevsky, et al., 2017; Redmon, et al., 2016)
2	Roboflow untuk manajemen dataset	Apa fungsi Roboflow dalam proses pengumpulan dan pelabelan dataset untuk deteksi objek?	Roboflow mempermudah proses anotasi citra, augmentasi data, dan konversi format dataset menjadi kompatibel dengan berbagai framework seperti YOLO dan TensorFlow. (Dwibedi, et al., 2021; Roboflow, 2023)

Tabel 3. 2 Studi Pustaka Peneltian Terdahulu (Lanjutan)

3	Flutter dan Android Studio dalam pengembangan aplikasi mobile AI	Bagaimana cara mengembangkan aplikasi AI dengan Flutter dan Android Studio?	Flutter memungkinkan pengembangan aplikasi lintas platform, dan dapat diintegrasikan dengan model TFLite untuk menjalankan deteksi objek secara real-time di perangkat Android. (Google, 2022)
4	Evaluasi model CNN Yolov8	Bagaimana mengukur kinerja model CNN YoloV8 dalam klasifikasi citra kerusakan aset?	Metode evaluasi meliputi akurasi, precision, recall, F1-score, dan mAP. Confusion matrix dan kurva loss juga digunakan untuk visualisasi hasil evaluasi. (Brownlee, 2020; Chollet, 2021)
5	Implementasi CNN dalam pendeteksian kerusakan aset	Apakah CNN efektif digunakan untuk mendeteksi kerusakan fisik pada aset seperti furnitur?	CNN dapat mendeteksi kerusakan fisik seperti kerusakan dan retakan pada furnitur dengan akurasi tinggi. Digunakan untuk otomasi inspeksi visual aset. (Zhang, et al., 2019)
7	Aplikasi CNN di perangkat mobile (Edge AI)	Bisakah CNN berjalan langsung di perangkat mobile untuk tugas deteksi objek real-time?	CNN dapat dijalankan secara lokal di smartphone menggunakan model TFLite untuk inferensi real-time tanpa koneksi internet. (Maeda, et al., 2018)

Tabel 3. 3 Studi Pustaka Peneltian Terdahulu (Lanjutan)

8	Evaluasi Kinerja Aplikasi (Black-Box Testing)	Bagaimana pengujian perangkat lunak berbasis black-box, termasuk teknik *equivalence partitioning* dan *boundary value analysis*, digunakan untuk menyusun skenario uji berdasarkan requirement pengguna tanpa melihat kode internal?	Aplikasi diuji menggunakan skenario real-world seperti pengambilan gambar kerusakan, pemrosesan deteksi, dan penyimpanan hasil. Hasil diuji dalam bentuk *test case* yang berisi input, output yang diharapkan, output aktual, serta status pass/fail. Beizer (1995), Sommerville (2016), Myers et al. (2011), Jorgensen (2013), Kaner et al. (1999), Pressman & Maxim (2020)

3.3 Analisis Kebutuhan

Dalam menjalankan penelitian pengembangan aplikasi mobile untuk pendeteksian kerusakan aset di Universitas Ma Chung, diperlukan analisis kebutuhan agar pelaksanaan proyek dapat berjalan efisien dan tepat sasaran. Analisis ini mencakup identifikasi kebutuhan dari sisi pengguna akhir maupun peneliti/pengembang, serta kebutuhan perangkat keras, perangkat lunak, dan data yang harus dipenuhi. Perlu diketahui bahwa seluruh data berupa citra aset kampus diperoleh melalui izin dari Bagian Aset Universitas Ma Chung, dan dokumentasi citra dilakukan secara manual oleh penulis melalui pengambilan foto langsung di lapangan.

3.3.1 Analisis Kebutuhan Pengguna

Analisis kebutuhan pengguna dilakukan untuk memahami kebutuhan, harapan, serta kendala yang dihadapi oleh pengguna akhir dalam menjalankan tugasnya. Dalam konteks pengembangan aplikasi pencatatan kerusakan aset di

lingkungan Universitas Ma Chung, pengguna utama aplikasi ini adalah petugas dari Bagian Aset dan staf dari SIPUSDA (Unit Sistem Informasi dan Pusat Data) yang bekerja sama dalam pengelolaan data aset kampus.

Berdasarkan hasil observasi lapangan dan diskusi dengan pihak-pihak terkait, diketahui bahwa pengguna membutuhkan sistem yang dapat mempermudah proses dokumentasi kerusakan aset secara cepat dan efisien. Fitur utama yang dibutuhkan adalah kemampuan untuk mengambil gambar kerusakan secara langsung melalui kamera perangkat, serta menyimpannya ke dalam sistem sebagai bukti visual. Dokumentasi ini akan mempermudah proses pelaporan dan tindak lanjut tanpa perlu melakukan verifikasi fisik secara langsung ke lokasi aset. Untuk meningkatkan efisiensi, dibutuhkan pula mekanisme otomasi verifikasi melalui integrasi model deteksi kerusakan berbasis AI. Mekanisme ini memungkinkan sistem untuk secara otomatis mengidentifikasi dan mengklasifikasi jenis kerusakan dari gambar yang diunggah, sehingga proses verifikasi tidak sepenuhnya bergantung pada pengecekan manual oleh petugas. Hasil verifikasi otomatis ini dapat langsung digunakan sebagai acuan awal dalam pelaporan, sementara petugas hanya perlu melakukan pengecekan tambahan jika diperlukan, sehingga mempercepat proses dan meminimalkan potensi keterlambatan tindak lanjut.

Aplikasi juga diharapkan memiliki antarmuka yang sederhana dan mudah dipahami, sehingga dapat digunakan oleh staf non-teknis tanpa memerlukan pelatihan khusus. Untuk menunjang fleksibilitas penggunaan, sistem perlu mendukung penyimpanan data secara lokal sehingga tetap dapat digunakan meskipun tidak terhubung ke jaringan internet. Pengguna juga memerlukan akses terhadap riwayat dokumentasi yang telah dicatat sebelumnya untuk mendukung monitoring dan pengambilan keputusan terkait perawatan atau penggantian aset.

Di samping itu, performa aplikasi juga menjadi perhatian utama. Aplikasi harus dapat berjalan secara optimal pada berbagai jenis perangkat Android, termasuk perangkat dengan spesifikasi menengah ke bawah. Hasil dari analisis ini menjadi acuan dalam perancangan fitur serta alur kerja aplikasi, sehingga solusi yang dikembangkan dapat menjawab kebutuhan nyata pengguna akhir di lapangan.

3.3.2 Kebutuhan Peneliti

Berikut kebutuhan perangkat keras dan lunak yang akan dibutuhkan dan digunakan oleh peneliti dalam mengerjakan penelitian ini.

1. Perangkat Keras
    a. PC
        i. Prosesor : Intel i3-9100F 3.60 GHz
        ii. Ram: 16 GB
        iii. SSD: 1 TB
        iv. GPU: NVIDIA Geforce RTX 3050
        v. Sistem Operasi: Windows 10 Pro 64 bit
    b. Smartphone (Perangkat Uji Aplikasi)
        i. Merek dan Tipe: Xiaomi Redmi Note 12 Pro 4G
        ii. Ram 8GB
        iii. Penyimpanan Internal: 256 GB
        iv. Sistem Operasi: Android 13
        v. Fitur Tambahan: Mendukung debugging USB dan instalasi aplikasi melalui Android Studio
2. Perangkat Lunak
    a. Android Studio
    b. Flutter
    c. Python 3.9
    d. Matplotlib
    e. Numpy
    f. Pandas
    g. Google Colaboratory
    h. Roboflow
    i. Albumentations
    j. Data

3.4 Pengumpulan dan Pembuatan Dataset

Tahap pertama adalah pengumpulan data citra aset furnitur kampus yang akan digunakan sebagai dataset. dataset dalam penelitian ini tidak diperoleh dari sistem SIPUSDA secara langsung, melainkan dikumpulkan secara manual oleh peneliti dengan izin dari Bagian Aset Universitas Ma Chung. SIPUSDA berperan sebagai penyedia infrastruktur dan pendukung teknis, namun proses dokumentasi aset dilakukan oleh Bagian Aset kampus.

Proses pengambilan data dilakukan secara langsung di lingkungan kampus Universitas Ma Chung, dengan menggunakan kamera smartphone Xiaomi Redmi Note 12 Pro 4G. Perangkat ini memiliki kamera utama 108 MP yang digunakan untuk mengambil citra kursi dari berbagai sudut dan jarak. Pengambilan gambar dilakukan dalam kondisi pencahayaan alami (outdoor dan semi-outdoor), tanpa lampu tambahan. Dalam praktiknya, variasi cahaya terjadi akibat perbedaan cuaca atau posisi objek terhadap sumber cahaya.

Dataset ini difokuskan pada empat tipe utama kerusakan aset, yaitu Jamur, Patah, Mengelupas, dan Roda Rusak, yang masing-masing direpresentasikan melalui instance citra kerusakan pada kursi.

Gambar 3. 2 Contoh Kerusakan Jamur pada Kursi

Gambar ini menunjukkan pertumbuhan jamur pada permukaan dudukan kursi. Area yang terkena jamur tampak berwarna keputihan hingga kehitaman

dengan tekstur tidak rata. Jamur biasanya muncul pada bagian yang lembap dan tidak terawat, dan area tersebut diberikan anotasi *bounding box* berwarna kuning.

Gambar 3. 3 Contoh Kerusakan Mengelupas pada Kursi

Gambar memperlihatkan permukaan vinil kursi yang terkelupas, terlihat dari adanya sobekan atau lapisan luar yang terangkat. Kerusakan jenis ini biasanya terjadi akibat gesekan atau usia material. Anotasi berwarna biru muda digunakan untuk menandai area yang rusak secara visual.

Gambar 3. 4 Contoh Kerusakan Patah pada Kursi

Pada Gambar 3.4, terdapat bagian struktural kursi yang patah. Kerusakan bersifat mekanis dan mengganggu fungsi utama kursi. Bagian yang rusak ditandai dengan kotak biru tua.

Gambar 3. 5 Contoh Kerusakan Roda Rusak pada Kursi

Gambar memperlihatkan kerusakan pada bagian roda kursi, seperti roda yang hilang, pecah, atau tidak sejajar. Masalah ini biasanya menyebabkan kursi tidak dapat bergerak dengan baik. Area kerusakan diberi anotasi merah muda untuk membedakannya dari jenis kerusakan lain.

Contoh gambar anotasi tersebut digunakan sebagai referensi visual saat proses pelabelan di Roboflow, sekaligus memastikan bahwa anotator memahami karakteristik masing-masing kerusakan. Jumlah data yang terkumpul mencapai total 218 gambar, memastikan keberagaman kondisi dan jenis aset. Setiap citra memiliki resolusi dan pencahayaan bervariasi sesuai kondisi lapangan saat pengambilan gambar. Sebelum digunakan lebih lanjut, data mentah ini diperiksa dan disaring: gambar yang duplikat, buram, atau tidak relevan dibuang agar kualitas dataset terjaga.

Sebelum dilakukan anotasi atau augmentasi, setiap gambar terlebih dahulu melalui proses *preprocessing* berupa penambahan padding berwarna abu-abu di sisi kiri dan kanan citra. Langkah ini bertujuan untuk mengubah proporsi gambar menjadi persegi (1:1) dengan resolusi akhir 1024x1024 piksel. Hal ini diperlukan karena sebagian besar gambar diambil menggunakan kamera smartphone yang menghasilkan rasio aspek memanjang, sehingga perlu dinormalisasi agar kompatibel dengan format input model YOLOv8.

Setelah data mentah dikumpulkan dan disaring, dilakukan proses augmentasi

citra guna meningkatkan keragaman dan jumlah data, terutama untuk kelas kerusakan yang jumlah instance-nya tergolong sedikit. Augmentasi ini dilakukan sebanyak lima kali terhadap gambar-gambar yang mengandung tipe kerusakan dengan jumlah instance yang rendah, seperti kelas "Mengelupas", "Roda Rusak", dan "Patah". Perlu dicatat bahwa augmentasi dilakukan terhadap gambar yang mengandung class tertentu, bukan terhadap jumlah instance-nya secara langsung. Tujuannya adalah untuk membantu meratakan distribusi jumlah instance per kelas agar model tidak terlalu bias terhadap kelas mayoritas. Alasan utama dilakukan augmentasi ini adalah karena jumlah gambar untuk beberapa kelas dirasakan masih kurang, sehingga model cenderung kurang akurat dalam mengenali kelas-kelas minoritas tersebut. Ketimpangan jumlah gambar antar kelas dapat menurunkan skor performa keseluruhan model, khususnya pada metrik seperti mAP, recall, atau F1-score. Oleh karena itu, augmentasi difokuskan pada kelas dengan representasi yang lebih rendah.

Augmentasi dilakukan sebelum proses anotasi menggunakan skrip Python berbasis pustaka *Albumentations*. Transformasi yang digunakan mencakup flipping horizontal, penyesuaian brightness dan contrast, pergeseran warna RGB, serta konversi sebagian citra ke grayscale. Teknik ini dipilih karena bersifat aman untuk mempertahankan konteks visual kerusakan, sambil memberikan variasi tampilan agar model tidak overfitting terhadap bentuk atau pencahayaan tertentu.

Tabel 3. 4 Jumlah Gambar per Kelas Sebelum dan Setelah Augmentasi

Tipe Kerusakan	Jumlah Gambar Asli	Jumlah Gambar Setelah Augmentasi
Jamur	65	100
Mengelupas	38	131
Patah	17	145
Rusak	18	121

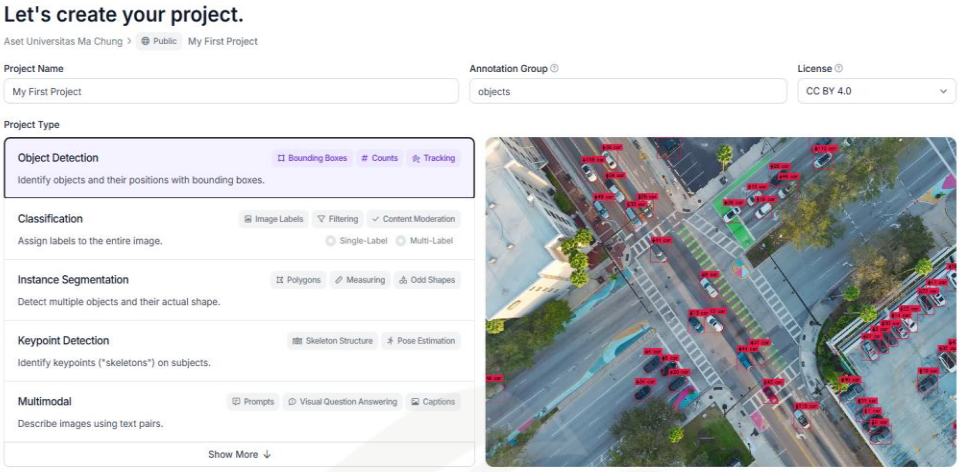
Selanjutnya, peneliti mengorganisir dataset ke dalam struktur yang sesuai untuk proses pelabelan. Pada tahap ini, disiapkan pula pembagian data secara awal menjadi subset untuk pelatihan, validasi, dan pengujian model. Pemisahan ini

dilakukan untuk mencegah bias dan memungkinkan evaluasi model secara obyektif di tahap akhir. Namun, pembagian final dilakukan secara otomatis pada platform pelabelan (Roboflow) setelah anotasi selesai (lihat Subbab 3.2). Dengan demikian, hasil dari tahap pengumpulan data adalah dataset citra aset kursi kampus yang telah terorganisir, siap untuk diberi label anotasi kerusakan.

3.5 Pelabelan Data dengan Roboflow

Setelah data citra terkumpul, langkah berikutnya adalah melakukan pelabelan data (*data labeling*) untuk menandai bagian-bagian kerusakan pada aset yang tampak di setiap citra. Proses pelabelan dilakukan menggunakan platform Roboflow, sebuah layanan berbasis web yang mempermudah anotasi objek pada dataset citra serta pengelolaan dataset secara terstruktur. Penggunaan Roboflow bertujuan untuk memastikan konsistensi anotasi dan efisiensi dalam pembuatan dataset yang terlabel. Roboflow menyediakan antarmuka grafis untuk menggambar *bounding box* (kotak pembatas) di sekitar objek kerusakan pada setiap gambar dan memberikan label kelas. Pada penelitian ini, peneliti mendefinisikan satu kelas objek utama yaitu "Kerusakan", yang merepresentasikan area kerusakan fisik pada furnitur kampus. Dengan skema satu kelas ini, model deteksi hanya difokuskan untuk mengenali keberadaan kerusakan saja (tanpa membedakan tipe furnitur), sehingga setiap *bounding box* yang ditandai akan dilabeli sebagai kelas "Kerusakan". Gambar yang tidak mengandung kerusakan dibiarkan tanpa anotasi apapun (dan akan dianggap sebagai sampel negatif, artinya tidak ada objek kerusakan dalam citra tersebut). Strategi pelabelan ini memastikan bahwa model YOLOv8 yang dilatih nantinya mampu mendeteksi lokasi kerusakan secara umum terlepas dari jenis benda atau furniturnya.

Seluruh citra aset yang telah dikumpulkan diunggah ke akun Roboflow peneliti dengan membuat sebuah proyek baru bertipe *Object Detection*. Pada saat pembuatan proyek, ditentukan skema label yang akan digunakan. Dalam penelitian ini, didefinisikan satu kelas objek utama yaitu "Kerusakan", yang merepresentasikan area kerusakan pada aset furnitur. Setiap gambar yang diunggah kemudian muncul dalam antarmuka labeling Roboflow



Gambar 3. 6 Pembuatan Proyek Object Detection

Hasil dari tahap pelabelan ini adalah dataset teranotasi yang siap digunakan untuk melatih model deteksi objek. Setiap citra dalam dataset memiliki file label pendamping (dalam format yang ditentukan) berisi informasi koordinat bounding box kerusakan (jika ada) beserta kelasnya. Dalam konteks implementasi YOLOv8, dataset diekspor dari Roboflow ke format anotasi YOLO (YOLOv5 PyTorch TXT) yang kompatibel dengan arsitektur YOLOv8. Pada format YOLO ini, setiap gambar memiliki sebuah file teks (.txt) yang mencantumkan detail setiap bounding box dalam bentuk: <class_id> <x_center> <y_center> <width> <height> dengan koordinat ternormalisasi (0 sampai 1). Roboflow secara otomatis mengonversi anotasi ke format tersebut dan menghasilkan berkas konfigurasi data YAML yang memuat daftar kelas ("Kerusakan") serta pembagian dataset train/val/test. Berkat dukungan Roboflow dalam konversi multi-format, peneliti dapat langsung memperoleh dataset dalam struktur yang siap digunakan oleh model YOLOv8 tanpa penanganan manual tambahan. Dataset akhir yang telah terorganisir dan terlabel ini kemudian diunduh untuk digunakan pada tahap pelatihan model.

3.6 Pembuatan dan Pelatihan Model YOLO

Pada tahap ini, peneliti membangun dan melatih model deteksi objek berbasis YOLOv8s dengan memanfaatkan dataset yang telah dianotasi. Proses pelatihan model dilakukan menggunakan *framework* Ultralytics YOLO yang menyediakan lingkungan terpadu untuk konfigurasi model, training, dan evaluasi. Sebelum pelatihan dimulai, peneliti memastikan persiapan lingkungan pengembangan yang

sesuai. Dalam penelitian ini, digunakan platform Google Colaboratory (Colab) yang menyediakan GPU untuk mempercepat komputasi training, dengan bahasa pemrograman Python 3.9 dan pustaka Ultralytics YOLO terinstal.

Ultralytics menyediakan antarmuka command-line maupun API Python untuk melakukan training dengan sederhana. Peneliti menggunakan API Python di Google Colab; pertama, model YOLO diinisialisasi dengan memuat bobot *pretrained*. Lalu dilakukan pemanggilan fungsi *train* dengan parameter data dan hyperparameter yang diinginkan. Secara umum, parameter penting yang diatur antara lain: *epochs* (jumlah epoch pelatihan), *batch size* (ukuran batch), *image size* (resolusi masukan), serta *learning rate*. Dalam percobaan, model dilatih selama misalnya 50-100 epoch atau hingga metrik validasi konvergen. Selama training berlangsung, pada setiap epoch model memproses sekumpulan gambar training, melakukan prediksi *bounding box* dan klasifikasi, kemudian menghitung *loss* dengan membandingkan prediksi terhadap label ground-truth. Optimizer (seperti SGD atau Adam) memperbarui bobot CNN dalam model YOLO sesuai gradient error. Ultralytics framework secara otomatis menampilkan metrik-metrik penting pada setiap epoch, seperti nilai *training loss*, *validation loss*, serta *mean Average Precision* (mAP) untuk data validation. Metrik mAP terutama digunakan untuk mengukur performa deteksi objek (dalam hal ini deteksi kerusakan) pada berbagai *threshold* IoU, dan merupakan indikator utama seberapa baik model dalam mendeteksi objek target. Selain itu, metrik *precision* dan *recall* untuk kelas "Kerusakan" dapat diamati guna memahami keseimbangan antara tingkat *false positive* dan *false negative*. Dengan memonitor tren kurva *loss* dan mAP selama pelatihan, peneliti dapat mengetahui apakah model mengalami *overfitting* atau masih dapat meningkatkan kinerja dengan penambahan epoch. Apabila *validation loss* mulai meningkat atau stagnan sementara *training loss* terus menurun, itu pertanda *overfitting* dan pelatihan dapat dihentikan.

Tabel 3. 5 Parameter Pelatihan Model Yolov8s

Parameter	Nilai
Epochs	200
Image Size	1024 x 1024
Patience	100
Cache	RAM
Device	GPU
Batch Size	Auto

Model dilatih menggunakan framework Ultralytics YOLO secara lokal di PC pribadi dengan dukungan GPU NVIDIA GeForce RTX 3050. Konfigurasi ini dipilih untuk memastikan proses pelatihan berlangsung efisien dengan kapasitas komputasi yang memadai. Penggunaan *pretrained weights* bertujuan mempercepat konvergensi, sementara penggunaan cache RAM mempercepat akses data selama training. Ukuran gambar 1024x1024 dipertahankan agar sesuai dengan hasil preprocessing citra, serta memberikan representasi spasial detail pada kerusakan aset.

Setelah seluruh epoch selesai, *framework* Ultralytics otomatis menyimpan bobot model hasil pelatihan. Biasanya disimpan beberapa *checkpoint*, termasuk bobot terakhir (last.pt) dan bobot terbaik (best.pt) berdasarkan skor mAP tertinggi pada data validasi. Dalam penelitian ini, bobot model YOLO terbaik yang dicapai selama training (best model) digunakan untuk langkah berikutnya. Sebelum melanjutkan, dilakukan evaluasi singkat terhadap model tersebut menggunakan data uji (*testing set*) yang telah dipisahkan sebelumnya. Metrik evaluasi akhir seperti akurasi klasifikasi kerusakan, *precision*, *recall*, *F1-score*, dan mAP pada data uji dihitung untuk memastikan model memenuhi ekspektasi kinerja. Hasil pengujian menunjukkan bahwa model YOLOv8 mampu mendeteksi area kerusakan pada citra aset dengan tingkat akurasi yang tinggi dan kesalahan minimal (hasil detail disajikan pada bab berikutnya). Model final ini kemudian siap untuk diintegrasikan ke dalam aplikasi mobile.

3.7 Evaluasi Kinerja Model YOLO

Evaluasi kinerja model YOLO dilakukan secara sistematis untuk memastikan akurasi dan keandalan dalam mendeteksi kerusakan aset. Seluruh model diuji menggunakan *testing set* yang sepenuhnya terpisah dari data latih dan validasi agar merepresentasikan skenario nyata. Kinerja diukur dengan metrik *precision*, *recall*, *F1-Score*, serta *Mean Average Precision* (mAP) pada berbagai tingkat IoU (mAP@50 dan mAP@50–95) (Ultralytics Docs). Metrik ini dipilih karena mampu memberikan gambaran keseimbangan antara ketepatan deteksi dan sensitivitas model dalam mengenali objek.

Selain metrik numerik, digunakan pula *confusion matrix* untuk menganalisis distribusi prediksi, baik *True Positive*, *False Positive*, *True Negative*, maupun *False Negative*. Analisis ini penting untuk memahami pola kesalahan model, misalnya saat salah mengklasifikasikan jenis kerusakan tertentu. Visualisasi *bounding box* pada citra uji juga digunakan sebagai validasi kualitatif, untuk menilai sejauh mana model mampu mendeteksi kerusakan kursi dengan akurat dan konsisten.

Penelitian ini menerapkan teknik 5-Fold Cross Validation pada seluruh varian YOLO yang digunakan, guna menilai konsistensi performa serta kemampuan generalisasi. Teknik ini dipilih karena mampu memberikan gambaran kinerja yang lebih stabil dibanding metode *hold-out*, khususnya pada dataset berukuran terbatas. Seluruh data dibagi ke dalam lima *fold* dengan ukuran seimbang, kemudian setiap fold secara bergantian digunakan sebagai data validasi, sementara fold lainnya dipakai sebagai data latih. Dengan cara ini, setiap sampel digunakan sebagai data validasi satu kali dan data latih sebanyak empat kali. Hasil evaluasi dari kelima fold kemudian dirata-ratakan untuk menghasilkan metrik keseluruhan, seperti *precision*, *recall*, mAP, dan *F1-Score*.

Dalam penelitian ini, evaluasi tidak hanya difokuskan pada satu model, tetapi juga mencakup beberapa varian YOLO dari berbagai generasi. Varian yang digunakan meliputi YOLOv5n, YOLOv5s, YOLOv5n6u, YOLOv5s6u, YOLOv8n, YOLOv8s, YOLOv11n, dan YOLOv11s. Dengan mencakup lebih banyak model, perbandingan dapat memberikan gambaran menyeluruh mengenai perbedaan kinerja antar generasi YOLO, baik dari sisi akurasi, stabilitas, maupun efisiensi. Analisis komparatif seluruh varian ini akan disajikan pada Bab IV (Hasil dan

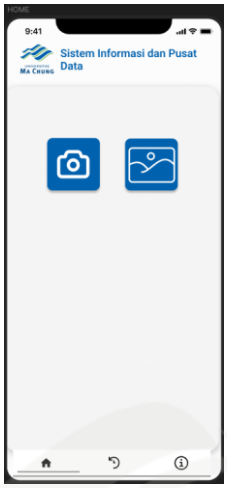
Pembahasan), yang dilengkapi dengan grafik pelatihan, *confusion matrix*, tabel perbandingan metrik, serta contoh visualisasi deteksi. Dengan demikian, Bab IV akan memberikan landasan yang komprehensif dalam menentukan varian YOLO yang paling sesuai untuk implementasi pada aplikasi deteksi kerusakan kursi berbasis citra.

3.8 Integrasi Model ke Aplikasi Mobile dan Rancangan Antarmuka

Penelitian ini menggunakan pendekatan integrasi berbasis cloud melalui layanan Ultralytics Shared Inference API. Layanan ini memungkinkan model YOLOv8 yang telah dilatih untuk dijalankan pada infrastruktur server milik Ultralytics, sehingga aplikasi mobile tidak perlu memuat dan menjalankan model secara lokal.

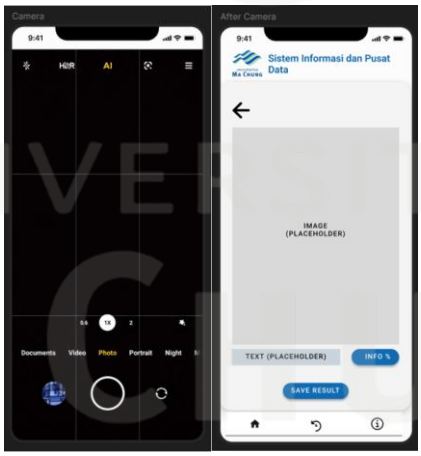
Aplikasi mobile yang dikembangkan dengan Flutter mengirimkan citra ke endpoint tersebut melalui HTTP request, lalu menerima hasil deteksi dalam bentuk respons JSON yang berisi informasi prediksi seperti bounding box, label kelas, dan confidence score. Proses ini memungkinkan inferensi dilakukan secara efisien meskipun perangkat pengguna memiliki keterbatasan komputasi. Penggunaan API ini memberikan sejumlah keuntungan, antara lain kemudahan dalam integrasi, pengurangan beban pemrosesan di sisi client, serta fleksibilitas dalam pembaruan model. Apabila diperlukan pembaruan terhadap performa atau versi model deteksi, peneliti hanya perlu memperbarui model di sisi server tanpa perlu mendistribusikan ulang aplikasi ke pengguna. Dengan pendekatan ini, proses integrasi model menjadi lebih sederhana dan adaptif terhadap kebutuhan pembaruan di masa depan, sekaligus menjaga performa aplikasi tetap ringan dan responsif saat dijalankan di perangkat mobile.

Antarmuka aplikasi pendeteksi kerusakan aset ini dirancang secara sistematis untuk memberikan pengalaman pengguna yang intuitif dan fungsional. Saat aplikasi pertama kali dijalankan, pengguna akan diarahkan ke tampilan awal atau splash screen yang menampilkan identitas institusi, yaitu Universitas Ma Chung. Setelah beberapa detik, aplikasi akan secara otomatis beralih ke halaman utama atau beranda (home screen).



Gambar 3. 7 Rancangan Home Screen

Pada halaman utama, terdapat dua tombol utama yang ditampilkan secara visual di bagian tengah layar, masing-masing mewakili metode input citra yang berbeda, yaitu tombol kamera dan tombol galeri. Keduanya merupakan jalur masuk untuk melakukan proses deteksi kerusakan aset. Selain itu, di bagian bawah layar terdapat tiga tombol navigasi (taskbar) yang terdiri dari tombol Home, Results, dan Information. Taskbar ini selalu tersedia di setiap halaman untuk memudahkan navigasi antar fitur utama aplikasi.



Gambar 3. 8 Rancangan Fungsi Kamera

Jika pengguna memilih tombol kamera, aplikasi akan membuka aplikasi kamera bawaan perangkat. Di sini, pengguna dapat memotret objek atau aset yang ingin dianalisis. Setelah gambar diambil, aplikasi akan mengalihkan pengguna ke halaman “After Camera”, yaitu tampilan hasil deteksi. Pada halaman ini, aplikasi akan menampilkan klasifikasi objek (Rusak atau Tidak Rusak) berdasarkan hasil

inferensi model YOLOv8s, serta confidence score dalam bentuk persentase yang menunjukkan tingkat kepercayaan sistem terhadap hasil deteksi tersebut. Di bawahnya tersedia tombol “Save Result” yang memungkinkan pengguna menyimpan hasil ke menu riwayat.

Gambar 3. 9 Rancangan Fungsi Galeri

Sementara itu, tombol galeri mengarahkan pengguna ke penyimpanan galeri perangkat, di mana mereka dapat memilih foto yang telah ada untuk dianalisis. Setelah pengguna memilih gambar dari galeri, aplikasi akan menampilkan hasil deteksi seperti pada mode kamera: klasifikasi kerusakan dan confidence score.

Gambar 3. 10 Rancangan Layar Hasil

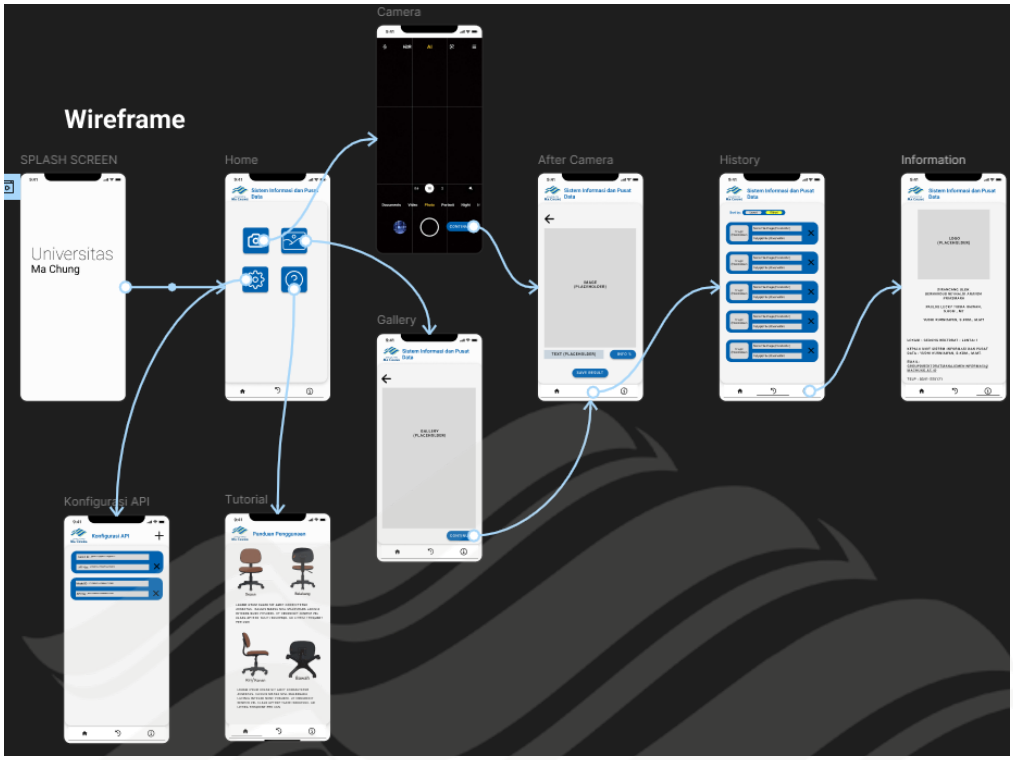
Semua hasil yang telah disimpan melalui mode kamera atau galeri akan muncul di menu “Results”. Pada halaman ini, pengguna dapat melihat riwayat hasil

identifikasi sebelumnya secara kronologis. Aplikasi menyediakan fitur sortir berdasarkan tanggal (dari yang terbaru maupun yang terlama), serta opsi untuk menghapus hasil secara satuan atau keseluruhan. Setiap entri hasil menampilkan gambar mini (thumbnail), nama file atau aset, dan tanggal penyimpanan.

Gambar 3. 11 Rancangan Layar Information

Terakhir, aplikasi juga menyediakan halaman “Information” atau “About” yang berisi informasi mengenai pengembang aplikasi, logo institusi, versi aplikasi, dan kontak resmi. Menu ini mencerminkan latar belakang akademik dari aplikasi, mencantumkan nama-nama mahasiswa pengembang serta dosen pembimbing atau kepala unit yang bertanggung jawab atas proyek ini.

Perlu dicatat bahwa antarmuka pengguna (UI) yang ditampilkan dalam perancangan aplikasi mungkin tidak sepenuhnya identik dengan tampilan akhir pada versi aplikasi yang dihasilkan.



Gambar 2. 12 Wireframe Prototype Aplikasi

Diagram ini menunjukkan rancangan navigasi dan antarmuka aplikasi secara keseluruhan. Terlihat hubungan antar layar utama: mulai dari Splash Screen awal yang menampilkan logo Universitas, kemudian menuju Home (beranda) sebagai pusat menu. Dari halaman Home, pengguna dapat mengakses fitur Kamera atau Galeri untuk memasukkan foto aset. Setelah pengguna mengambil foto melalui kamera, aplikasi menampilkan layar After Camera berupa pratinjau hasil dan opsi penyimpanan. Alur dari kamera berlanjut ke proses deteksi otomatis sebelum hasil ditampilkan. Sebaliknya, jika pengguna memilih foto dari galeri, alurnya menuju proses deteksi yang sama.

Diagram ini juga memperlihatkan halaman History (Riwayat) yang dapat diakses melalui navigasi bawah untuk melihat daftar hasil deteksi tersimpan, serta halaman Information yang berisi detail informasi aset atau tentang aplikasi. Selain itu, terdapat halaman Konfigurasi API untuk pengaturan model AI yang digunakan, dan halaman Tutorial yang memberikan panduan penggunaan aplikasi. Panah-panah pada diagram menggambarkan transisi atau perpindahan antar layar; misalnya, dari Home ke Kamera/Galeri, lalu ke Hasil Deteksi, dan seterusnya.

3.9 Evaluasi Kinerja Aplikasi

Evaluasi kinerja aplikasi merupakan langkah penting untuk memastikan bahwa sistem yang dikembangkan memenuhi tujuan dan kebutuhan pengguna. Evaluasi ini dilakukan secara menyeluruh, mencakup pengujian fungsional, pengujian usability, dan pengujian performa aplikasi dalam mendeteksi kerusakan aset secara real-time. Pendekatan evaluasi memanfaatkan metode black-box testing, survei kepuasan pengguna, serta pengukuran performa di berbagai kondisi penggunaan.

Pengujian fungsional dalam penelitian ini menggunakan pendekatan black-box testing, di mana sistem diuji semata-mata berdasarkan interaksi pengguna (input–output) tanpa akses ke struktur internal kode. Metode ini sangat efektif dalam memverifikasi bahwa semua fitur utama—seperti pengambilan foto lewat kamera, pemrosesan gambar dari galeri, penyimpanan hasil deteksi, dan navigasi antarmuka—berjalan sesuai spesifikasi pengguna akhir. Teknik black-box testing umum seperti *equivalence partitioning* dan *boundary value analysis* digunakan untuk merancang skenario uji yang mewakili berbagai kondisi input umum maupun ekstrem, sehingga cakupan pengujian menjadi lebih menyeluruh

Dalam skenario pengujian, pengguna mengambil foto kursi rusak melalui aplikasi; diharapkan aplikasi dapat membuka kamera, mengambil gambar, menghasilkan deteksi kerusakan dengan bounding box dan confidence score, serta mencatat respons waktu aplikasi. Skenario serupa dilakukan untuk pengambilan gambar dari galeri, penyimpanan hasil deteksi ke menu riwayat, dan navigasi antar menu. Setiap test case dinilai berdasarkan hasil aktual dibandingkan dengan hasil yang diharapkan (*expected output*), direkam sebagai status “Pass” atau “Fail”, serta waktu respons, untuk evaluasi performa sistem secara menyeluruh. Selain itu, confusion matrix dan visualisasi kualitas deteksi (bounding box) juga dapat digunakan untuk validasi kualitatif, membantu memastikan bahwa fitur deteksi benar-benar sesuai kebutuhan pengguna.

Pengujian kinerja merupakan salah satu jenis pengujian non-fungsional yang bertujuan mengevaluasi seberapa baik suatu aplikasi bekerja dari segi responsivitas, stabilitas, dan efisiensi di bawah beban kerja tertentu. Dalam konteks rekayasa perangkat lunak, pengujian kinerja dilakukan untuk memastikan aplikasi dapat

merespons dengan cepat dan tetap stabil ketika dihadapkan pada skenario penggunaan nyata, termasuk jumlah data atau pengguna yang tinggi. Pengujian kinerja mengevaluasi responsivitas dan efisiensi aplikasi di bawah beban kerja yang berbeda . Artinya, metrik seperti waktu respons, throughput, pemanfaatan sumber daya (CPU, memori), serta durabilitas aplikasi ketika diberi beban berlebih (stress test) menjadi fokus utama. Tanpa pengujian kinerja, sebuah aplikasi mungkin berfungsi benar secara logika namun gagal memenuhi ekspektasi pengguna karena lambat atau sering hang ketika digunakan.

Pada aplikasi mobile, pengujian kinerja memiliki tantangan dan pendekatan tersendiri. Perangkat mobile memiliki keterbatasan sumber daya (CPU, memori, dan baterai) serta variasi spesifikasi hardware yang luas, sehingga aplikasi harus diujikan pada berbagai kondisi. Melalui pemetaan literatur sistematis mengidentifikasi berbagai alat, metode, dan strategi yang digunakan dalam pengujian kinerja aplikasi mobile, termasuk teknik simulasi beban, penggunaan profiling tools, dan otomasi pengujian di berbagai platform . Pendekatan-pendekatan tersebut mencakup analisis metrik spesifik mobile seperti waktu startup, penggunaan memori, jumlah frame per detik (FPS) pada animasi UI, hingga konsumsi daya baterai. Melalui kombinasi metode tersebut, pengembang dapat mengidentifikasi bottleneck kinerja dan memastikan aplikasi mobile tetap responsif serta efisien dalam kondisi operasional sesungguhnya.

Beberapa parameter kinerja kunci yang umum dijadikan acuan dalam pengujian aplikasi mobile antara lain: waktu muat aplikasi (startup time), penggunaan memori saat aplikasi berjalan, responsivitas antarmuka pengguna (misalnya diukur dari kestabilan frame rate), serta kinerja jaringan (jika aplikasi terhubung ke API) . Pengembang perlu menetapkan tolok ukur atau standar untuk tiap metrik tersebut. Sebagai contoh, waktu startup yang ideal untuk aplikasi mobile biasanya berada di bawah beberapa detik pertama; menurut pedoman performa Android, peluncuran dingin (cold start) sebaiknya di bawah 5 detik untuk memastikan pengalaman pengguna yang baik . Demikian pula, penggunaan memori perlu dijaga agar tidak berlebihan – aplikasi yang menghabiskan ratusan MB memori berpotensi menyebabkan perangkat kelas menengah kebawah menjadi lambat atau crash.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Dataset dan Distribusi Data

Setelah melalui proses pelatihan yang intensif, model YOLOv8s berhasil mencapai konvergensi dengan performa yang menjanjikan. Pelatihan dilakukan menggunakan dataset citra kerusakan aset yang telah dibagi menjadi data latih dan validasi, mencakup 5 kelas (Jamur, Mengelupas, Patah, Roda Rusak, dan background). Model dilatih selama lebih dari 100 epoch hingga metrik evaluasi relatif stabil. Selama pelatihan, tiga jenis loss function dipantau, yaitu box loss (kesalahan prediksi koordinat bounding box), cls loss (kesalahan klasifikasi kelas objek), dan DFL loss (Distribution Focal Loss untuk penyetelan prediksi koordinat). Selain itu, metrik evaluasi utama seperti precision, recall, mAP50, dan mAP50–95 diukur pada set validasi di setiap epoch untuk memantau kinerja model seiring waktu.

Dataset difokuskan pada empat kategori kerusakan utama yang umum ditemukan pada aset kursi kampus, yaitu:

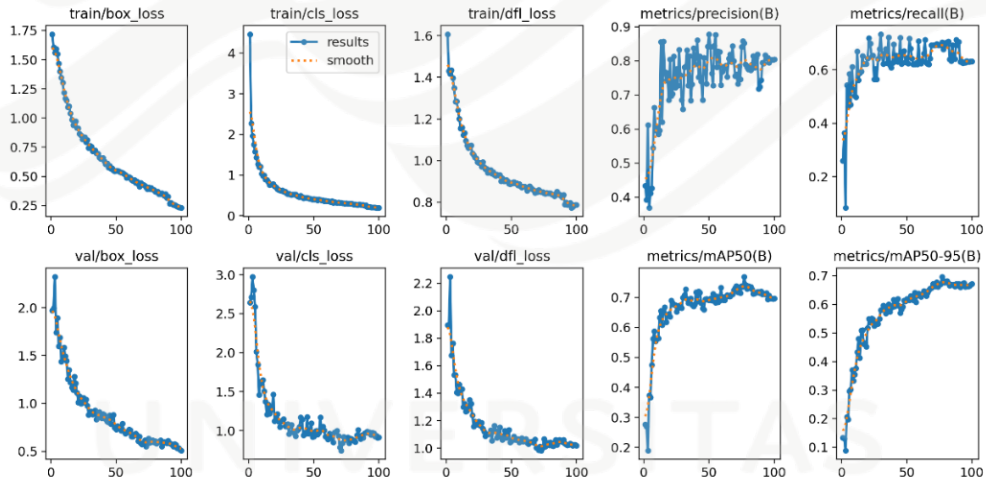
1. Jamur: kerusakan biologis akibat kelembaban pada permukaan kursi
2. Patah: kerusakan struktural seperti rangka atau dudukan yang retak/patah
3. Mengelupas: kerusakan visual akibat pelapis permukaan terkelupas
4. Roda Rusak: kerusakan pada bagian kaki atau roda yang menyebabkan ketidakstabilan fungsi kursi.

Selain itu, disertakan pula kelas background sebagai kontrol, yang berisi gambar kursi dalam kondisi baik atau tanpa kerusakan untuk membantu model belajar membedakan antara kerusakan dan non-kerusakan.

Untuk menyesuaikan kekurangan jumlah data dan meningkatkan variasi, peneliti juga melakukan augmentasi data berupa rotasi, pencahayaan, cropping, dan flipping, agar model lebih *robust* terhadap perbedaan orientasi dan kondisi pencahayaan di lapangan.

4.2 Hasil Pelatihan Model YOLOv8s

Setelah melalui proses pelatihan yang intensif, model YOLOv8s berhasil mencapai konvergensi dengan performa yang menjanjikan. Pelatihan dilakukan menggunakan dataset citra kerusakan aset yang telah dibagi menjadi data latih dan validasi, mencakup 5 kelas (Jamur, Mengelupas, Patah, Roda Rusak, dan background). Model dilatih selama lebih dari 100 epoch hingga metrik evaluasi relatif stabil. Selama pelatihan, tiga jenis loss function dipantau, yaitu box loss (kesalahan prediksi koordinat bounding box), cls loss (kesalahan klasifikasi kelas objek), dan DFL loss (Distribution Focal Loss untuk penyetelan prediksi koordinat). Selain itu, metrik evaluasi utama seperti precision, recall, mAP50, dan mAP50–95 diukur pada set validasi di setiap epoch untuk memantau kinerja model seiring waktu.



Gambar 4. 1 Hasil Pelatihan Model YOLOv8s

Gambar 4.1 menampilkan grafik hasil pelatihan model YOLOv8s selama 100 epoch. Grafik ini mencakup metrik loss (box loss, classification loss, dan distribution focal loss) baik pada data latih maupun validasi, serta metrik evaluasi utama seperti precision, recall, dan mAP (mean Average Precision). Pada bagian loss function, baik box loss, cls loss, maupun dfl loss pada data latih dan validasi menunjukkan tren menurun secara konsisten. Hal ini mengindikasikan bahwa model semakin mampu mempelajari pola data dan memperbaiki prediksi bounding box maupun klasifikasi objek dari waktu ke waktu. Tidak terdapat kenaikan signifikan pada loss validasi, sehingga dapat disimpulkan model tidak mengalami

overfitting. Selanjutnya, pada metrik evaluasi, precision meningkat hingga mendekati nilai mendekati 0.9, menandakan bahwa sebagian besar prediksi positif model adalah benar (minim false positive). Recall juga menunjukkan tren naik hingga stabil pada kisaran 0.6–0.7, yang berarti model semakin baik dalam menemukan objek kerusakan yang ada (minim false negative). Untuk metrik agregat, mAP@0.5 mencapai nilai mendekati 0.9, sementara mAP@0.5–0.95 konsisten meningkat hingga kisaran 0.65 pada akhir pelatihan. Hal ini menegaskan bahwa model tidak hanya mampu mendeteksi objek dengan baik pada IoU threshold rendah, tetapi juga cukup robust pada berbagai threshold yang lebih ketat.

Secara keseluruhan, tren grafik pada Gambar 4.1 menunjukkan bahwa model YOLOv8s berhasil mencapai konvergensi dengan performa yang stabil. Penurunan loss yang konsisten, diiringi peningkatan precision, recall, serta mAP, membuktikan bahwa model mampu melakukan deteksi objek dengan akurasi yang baik tanpa indikasi overfitting.

4.3 Visualisasi Hasil Deteksi Objek Kerusakan

Untuk memahami bagaimana model bekerja pada contoh nyata, dilakukan visualisasi hasil deteksi kerusakan aset menggunakan model YOLOv8s yang telah dilatih. Gambar berikut menampilkan salah satu contoh output model dalam mendeteksi kerusakan pada aset fisik.

Gambar 4. 2 Contoh Hasil Deteksi Model

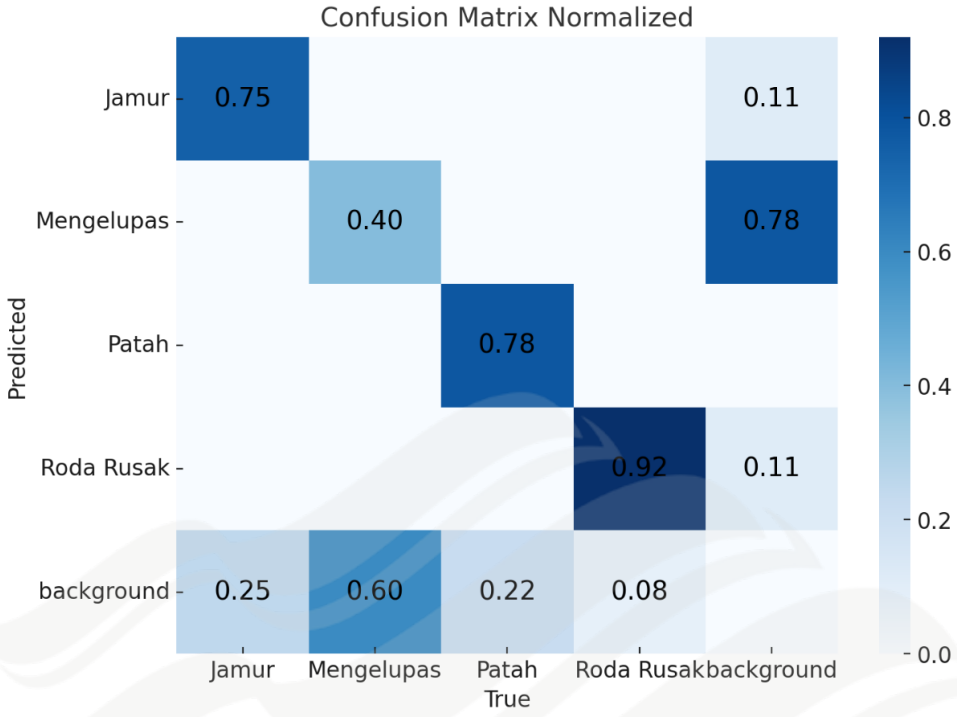
Contoh hasil deteksi kerusakan aset oleh model YOLOv8s pada citra kursi rusak. Model berhasil mengenali dua jenis kerusakan sekaligus pada objek tersebut,

ditandai dengan bounding box berwarna dan label kelas beserta confidence score. Bounding box biru menunjukkan prediksi kerusakan "Jamur" dengan confidence 0.78, sedangkan bounding box putih menunjukkan prediksi kerusakan "Patah" dengan confidence 0.84. Prediksi ini sesuai dengan kondisi objek: kursi tersebut mengalami patah pada rangkanya sekaligus terdapat jamur pada permukaan dudukannya.

Tampak bahwa model memberikan tingkat kepercayaan yang cukup tinggi (>0.75) untuk kedua deteksi, menandakan keyakinan model terhadap prediksi kerusakan tersebut. Visualisasi ini mengindikasikan kinerja model yang mampu mendeteksi multi-objek atau multi-kerusakan dalam satu gambar: dalam kasus ini satu aset (kursi) memiliki lebih dari satu jenis kerusakan yang berhasil diidentifikasi secara bersamaan. Hasil deteksi ditampilkan secara real-time di aplikasi, di mana setiap kotak melingkupi area kerusakan yang terdeteksi dan label membantu pengguna memahami jenis kerusakan yang ditemukan. Contoh ini juga menunjukkan output akhir yang diharapkan dari sistem: pengguna dapat mengambil foto aset, kemudian model akan menandai bagian-bagian aset yang rusak (misalnya berjamur atau patah) secara otomatis lengkap dengan tingkat kepercayaan. Secara keseluruhan, visualisasi hasil deteksi memperlihatkan bahwa model YOLOv8s mampu menangkap fitur-fitur kerusakan utama dan menandakannya dengan cukup akurat pada citra, sehingga validasi kualitatif ini mendukung metrik kuantitatif yang telah dicapai model.

4.4 Evaluasi Model dengan Confusion Matrix

Evaluasi mendalam terhadap performa model deteksi dilakukan menggunakan confusion matrix untuk melihat distribusi prediksi model terhadap kelas sebenarnya. Confusion matrix memberikan gambaran jelas mengenai True Positive (TP), False Positive (FP), False Negative (FN), dan True Negative (TN) untuk setiap kelas. Karena tugas yang dihadapi adalah deteksi objek, konsep True Negative (prediksi background saat tidak ada objek) turut dipertimbangkan sebagai kelas "background" dalam evaluasi ini. Berikut disajikan confusion matrix dalam bentuk ternormalisasi (persentase) dan absolut (jumlah kasus) beserta interpretasinya.

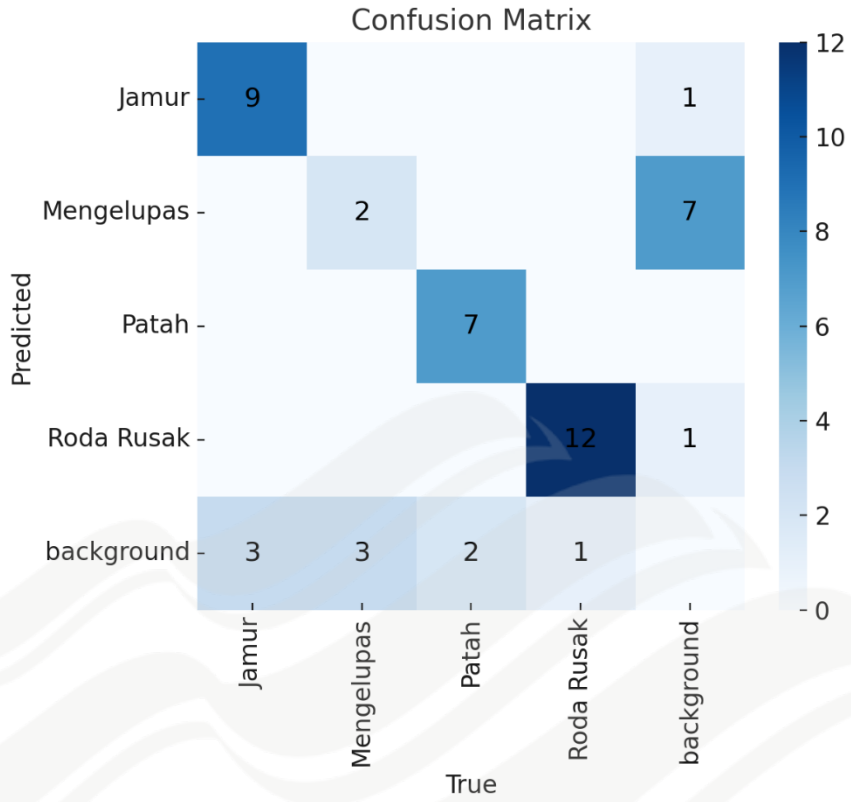


Gambar 4. 3 Confusion Matrix Validasi (Ternormalisasi) Model Yolov8s

Confusion Matrix (ternormalisasi) hasil deteksi model YOLOv8s pada data memperlihatkan proporsi prediksi benar dan salah dari model YOLOv8s terhadap data uji. Nilai diagonal menunjukkan tingkat keberhasilan model dalam mengenali setiap kelas (True Positive), sementara nilai di luar diagonal merepresentasikan kesalahan klasifikasi (False Positive atau False Negative).

Hasil menunjukkan bahwa kelas Roda Rusak memiliki performa terbaik dengan nilai 0.92, artinya 92% instance kerusakan roda berhasil dikenali dengan benar. Kelas Patah juga menunjukkan performa cukup baik dengan nilai 0.78, diikuti oleh kelas Jamur dengan nilai 0.75. Sementara itu, kelas Mengelupas memperoleh hasil paling rendah dengan nilai diagonal hanya 0.40, menandakan sebagian besar instance kelas ini salah terklasifikasi ke kategori lain, terutama ke background.

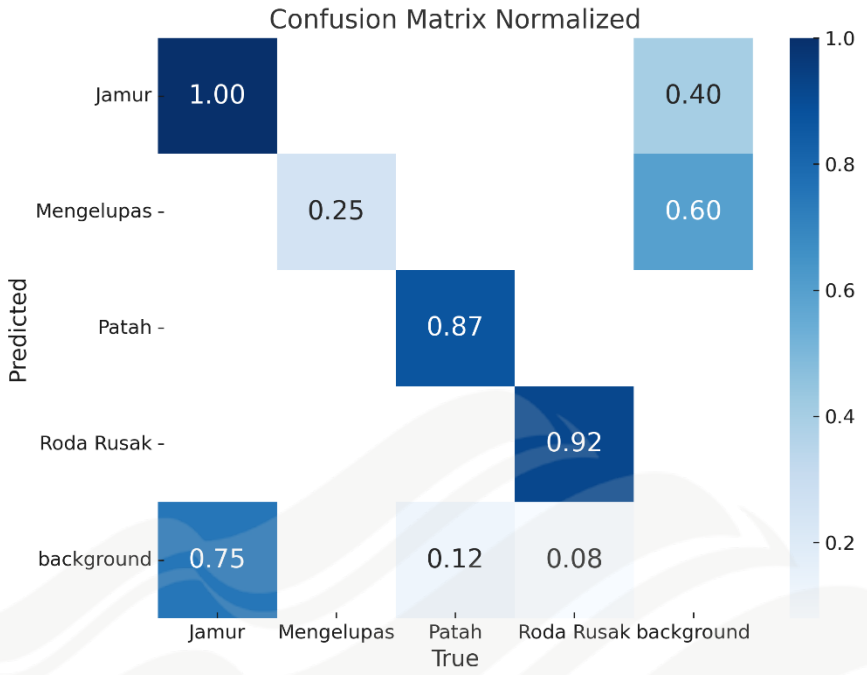
Temuan ini mengindikasikan bahwa model lebih mudah mengenali kerusakan dengan pola visual yang jelas dan kontras (seperti roda rusak dan patah), tetapi masih kesulitan dalam membedakan kerusakan dengan karakteristik visual halus atau menyerupai tekstur normal (seperti mengelupas).



Gambar 4. 4 Confusion Matrix Validasi Model Yolov8s

Confusion Matrix (absolut) hasil deteksi model YOLOv8s pada data uji memperlihatkan jumlah prediksi benar dan salah dari model YOLOv8s terhadap data uji. Angka pada setiap kotak menunjukkan berapa banyak instance suatu kelas yang diprediksi ke kelas tertentu. Nilai diagonal mewakili prediksi benar (True Positive), sedangkan angka di luar diagonal menunjukkan kesalahan klasifikasi.

Hasil menunjukkan bahwa kelas Roda Rusak memiliki performa terbaik, dengan 12 instance dikenali dengan benar dan hanya 1 instance salah klasifikasi. Kelas Patah juga cukup baik, dengan 7 instance terdeteksi benar. Kelas Jamur terdeteksi sebanyak 9 instance benar, meskipun masih terdapat 1 kesalahan prediksi. Sebaliknya, kelas Mengelupas menjadi yang paling sulit dikenali, hanya 2 instance yang berhasil terklasifikasi dengan benar, sedangkan 7 instance justru salah diprediksi ke kelas lain, terutama background. Hal serupa terlihat pada kategori background, di mana beberapa instance masih keliru diklasifikasikan sebagai kelas kerusakan (misalnya 3 ke Jamur dan 3 ke Mengelupas).



Gambar 4. 5 Confusion Matrix Testing (Ternormalisasi) Model Yolov8s

Hasil pengujian menggunakan confusion matrix pada data test set memperlihatkan perbedaan performa antar varian YOLO. Pada model YOLOv8s, terlihat bahwa kelas Jamur dan Patah mampu dikenali dengan sangat baik, dengan nilai prediksi benar masing-masing sebesar 1.00 dan 0.87. Kelas Roda Rusak juga menunjukkan akurasi tinggi dengan nilai 0.92, sedangkan kelas Mengelupas relatif lebih rendah (0.25), menandakan bahwa model masih mengalami kesulitan dalam mendeteksi jenis kerusakan ini. Selain itu, terdapat kasus kesalahan prediksi di mana citra background diklasifikasikan sebagai Jamur (0.75) atau Mengelupas (0.50), sehingga menambah jumlah false positive.

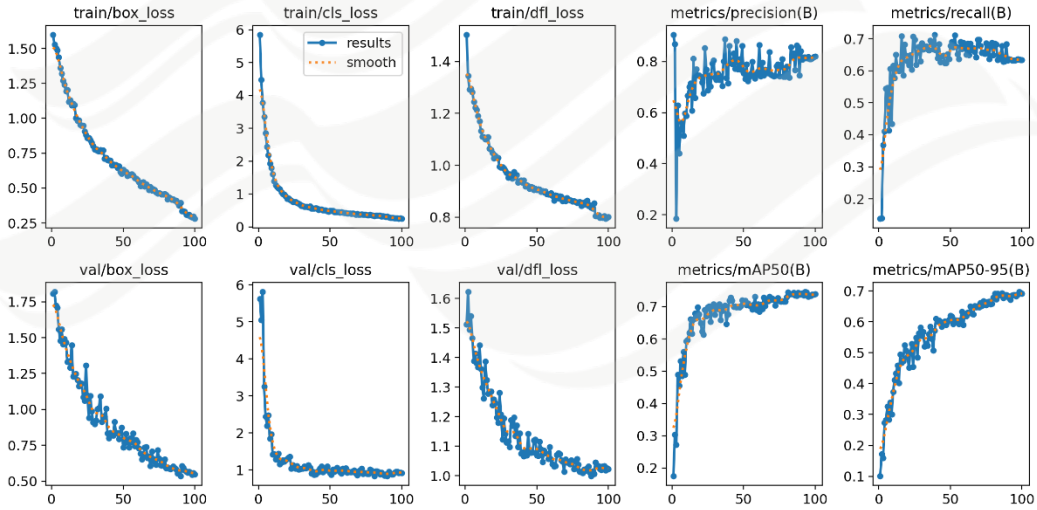
Secara keseluruhan, confusion matrix ini memperlihatkan bahwa model lebih andal dalam mengenali kerusakan dengan ciri visual yang jelas (seperti patah dan roda rusak), tetapi masih kesulitan pada kerusakan yang lebih halus atau mirip tekstur normal (seperti mengelupas).

4.5 Perbandingan dengan Yolo Versi Lainnya

Setelah dilakukan evaluasi terhadap performa model YOLOv8s pada data uji, tahap berikutnya adalah membandingkan hasil tersebut dengan versi YOLO lain yang relevan untuk mengetahui versi yang paling optimal dalam mendeteksi kerusakan aset. Perbandingan difokuskan pada metrik utama seperti precision,

recall, dan akurasi (mAP) serta visualisasi confusion matrix dari masing-masing model. Analisis ini bertujuan mengidentifikasi versi YOLO yang mampu memberikan keseimbangan terbaik antara ketepatan deteksi (minim false positive) dan kelengkapan deteksi (minim false negative).

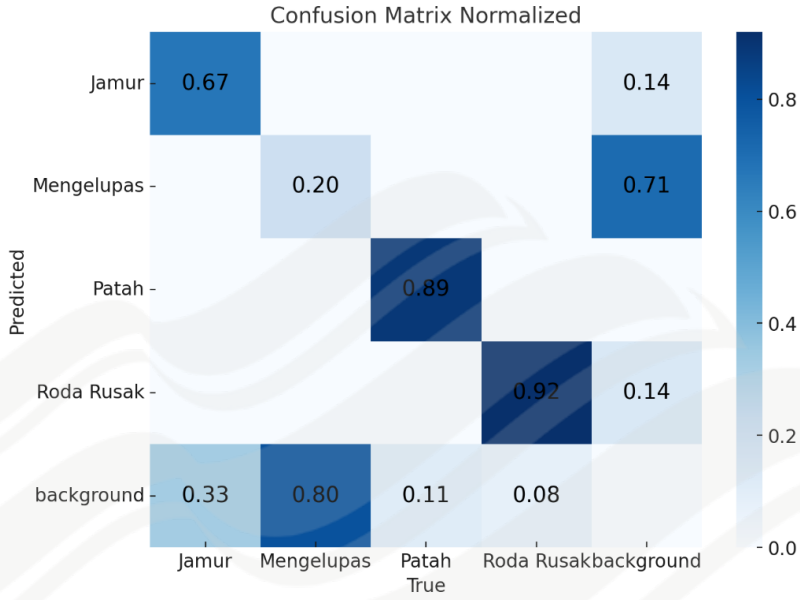
Alasan utama pemilihan ini didasarkan pada hasil evaluasi confusion matrix yang menunjukkan bahwa YOLOv8s memiliki akurasi yang stabil, precision yang tinggi, serta performa konsisten di berbagai fold. Dengan kata lain, perbandingan ini menjadi dasar argumentasi bahwa YOLOv8s adalah pilihan yang paling sesuai untuk penelitian, mengingat keterbatasan sumber daya komputasi dan kebutuhan implementasi pada perangkat mobile.



Gambar 4. 6 Hasil Pelatihan Model Yolov5n6u

Secara keseluruhan, YOLOv8s menunjukkan kualitas deteksi yang sedikit lebih baik dan proses belajar yang lebih stabil dibanding YOLOv5n6u. Pada kurva metrik, v8s konsisten mencapai precision ~0,82–0,86 dengan recall ~0,65–0,70, sedangkan v5n6u berada di precision yang mirip namun recall cenderung lebih rendah (~0,60–0,65) sehingga lebih banyak objek terlewat. Dampaknya terlihat pada kualitas akhir: mAP50 v8s ≈0,74–0,76 dan mAP50-95 v8s ≈0,68–0,70, sedikit di atas mAP50 v5n6u ≈0,72–0,74 dan mAP50-95 v5n6u ≈0,66–0,68. Dari sisi loss, keduanya sama-sama turun tajam dan stabil; v5n6u memang berakhir dengan box/cls/dfl loss yang sedikit lebih rendah, namun hal ini tidak otomatis berbanding lurus dengan *generalization* pada data validasi—terbukti metrik mAP dan recall v8s tetap unggul. Secara praktis, temuan ini mengindikasikan bahwa v8s lebih

efektif menangkap variasi objek (lebih sedikit *missed detection*) tanpa mengorbankan presisi, sehingga lebih sesuai untuk skenario lapangan yang menuntut keseimbangan antara ketepatan (precision) dan kelengkapan temuan (recall).

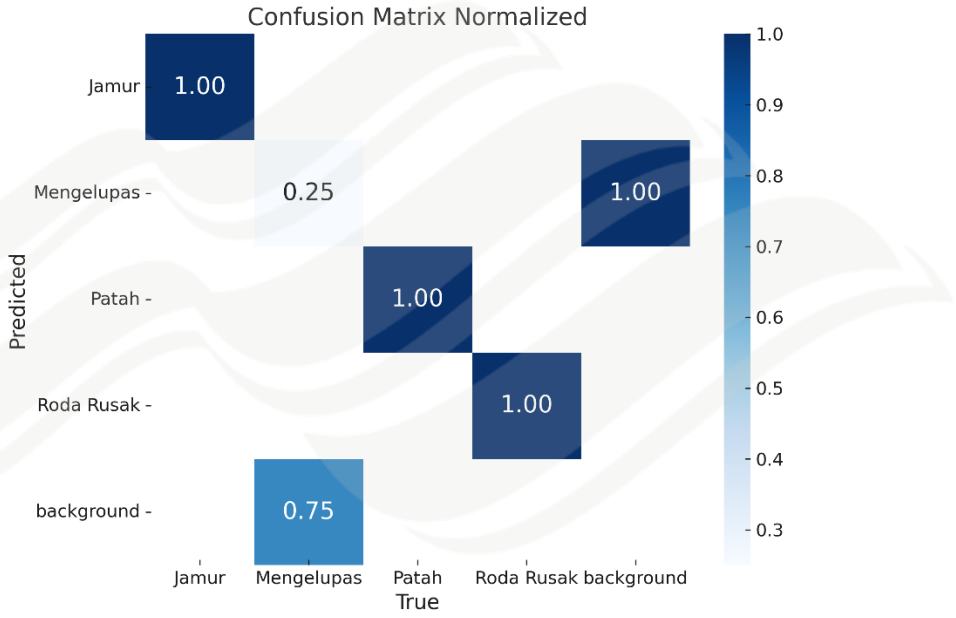


Gambar 4.7 Confusion Matrix Validasi (Ternormalisasi) Model Yolov5n6u

Hasil evaluasi pada data set validasi menunjukkan adanya perbedaan yang cukup jelas antara model YOLOv5n6u dan YOLOv8s. Dari sisi metrik, YOLOv8s unggul dalam precision, recall, dan mAP@50. Precision YOLOv8s mencapai 0.852, lebih tinggi dibanding YOLOv5n6u yang hanya 0.818, menandakan bahwa YOLOv8s lebih sedikit menghasilkan false positive. Recall YOLOv8s juga lebih baik, yaitu 0.703 dibandingkan 0.635 pada YOLOv5n6u, yang berarti model mampu mendeteksi lebih banyak objek kerusakan dan mengurangi jumlah false negative. Pada mAP@50, YOLOv8s mencatat nilai 0.769, lebih tinggi dibanding YOLOv5n6u dengan 0.738, sehingga YOLOv8s lebih akurat dalam mendeteksi objek pada tingkat overlap standar (IoU 0.5). Sementara itu, pada mAP@50-95, kedua model sama-sama mencapai 0.697, menunjukkan performa yang setara ketika diuji pada berbagai threshold IoU yang lebih ketat.

Jika ditinjau dari confusion matrix, YOLOv8s memberikan hasil yang lebih seimbang antar kelas. Pada kelas Jamur, performa meningkat dari 0.67 (YOLOv5n6u) menjadi 0.75 (YOLOv8s). Untuk kelas Mengelupas, terjadi peningkatan signifikan dari 0.20 menjadi 0.40, meskipun masih menjadi kelas yang

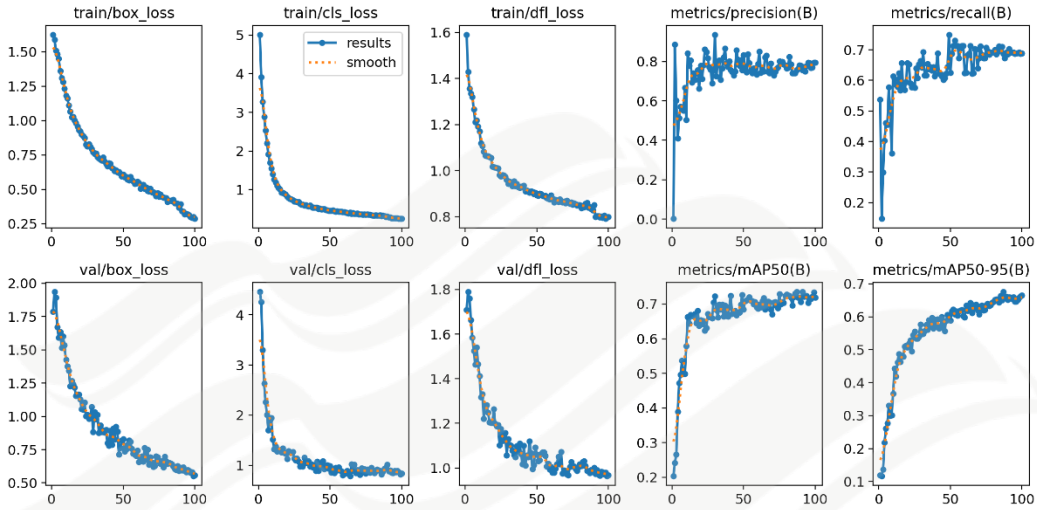
paling sulit dideteksi. Pada kelas Patah, YOLOv5n6u lebih unggul (0.89) dibanding YOLOv8s (0.78), namun perbedaan ini masih dalam batas wajar. Sementara itu, pada kelas Roda Rusak, kedua model konsisten sangat baik dengan nilai 0.92. Hal yang menarik adalah bagaimana YOLOv5n6u sering salah mengklasifikasikan background sebagai kerusakan (khususnya ke kelas Mengelupas dengan nilai 0.80), sedangkan YOLOv8s menunjukkan distribusi kesalahan yang lebih seimbang dan tidak terlalu bias pada satu kelas tertentu.



Gambar 4. 8 Confusion Matrix Testing (Ternormalisasi) Model Yolov5n6u

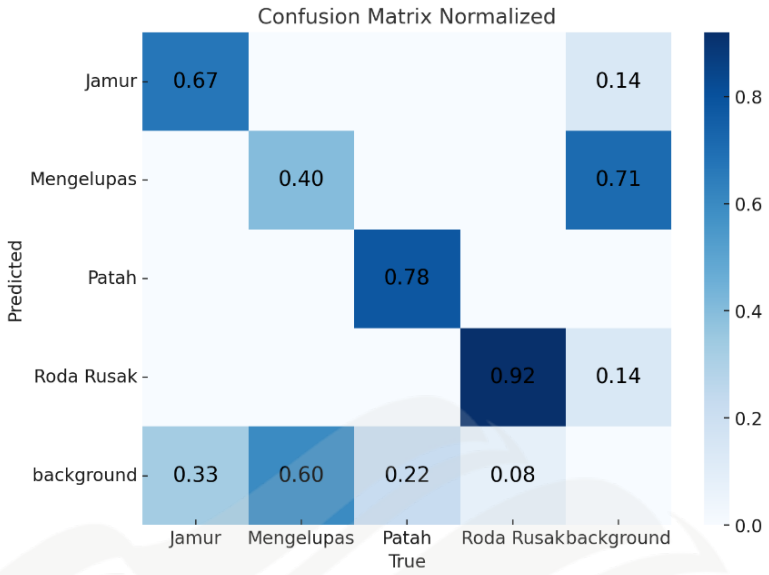
Pada pengujian menggunakan data test set, hasil *confusion matrix* menunjukkan perbedaan kinerja antara YOLOv8s dan YOLOv5n6u. Model YOLOv8s mampu mengenali kelas *Jamur*, *Patah*, dan *Roda Rusak* dengan sempurna (nilai 1.00), sehingga tidak ada sampel yang salah terklasifikasi pada ketiga jenis kerusakan tersebut. Sebaliknya, YOLOv5n6u hanya mencapai 0.83 pada kelas *Jamur* dan 0.92 pada kelas *Roda Rusak*, menandakan masih terdapat sebagian kecil kesalahan klasifikasi. Pada kelas *Patah*, kedua model menunjukkan performa setara dengan akurasi penuh. Namun demikian, untuk kelas *Mengelupas*, YOLOv5n6u lebih unggul karena berhasil mengklasifikasikan benar 50% sampel, sedangkan YOLOv8s hanya 25%. Di sisi lain, kedua model masih menunjukkan kelemahan pada prediksi *background*, di mana sebagian citra tanpa kerusakan terklasifikasi sebagai kerusakan tertentu, meskipun distribusi kesalahannya

berbeda. Secara keseluruhan, YOLOv8s memberikan performa yang lebih konsisten dan seimbang pada sebagian besar kelas kerusakan, sedangkan YOLOv5n6u lebih sensitif terhadap kerusakan *Mengelupas* namun kurang stabil pada kelas lainnya.



Gambar 4. 9 Hasil Pelatihan Model Yolov5nu

Dibanding YOLOv5n, YOLOv8s tampak unggul tipis namun konsisten pada metrik akhir. Dari kurva, v8s mempertahankan precision di kisaran ~0,82–0,86 dengan recall ~0,65–0,70, sedangkan v5n berada di precision yang sedikit lebih rendah (~0,78–0,82) dengan recall yang sebanding atau sedikit di bawah. Dampaknya, mAP50 v8s stabil di ≈0,73–0,76 dan mAP50-95 v8s di ≈0,66–0,70, sementara v5n berakhir di mAP50 ≈0,70–0,72 dan mAP50-95 ≈0,63–0,66. Pola *learning* keduanya baik (train/val box-loss, cls-loss, dan dfl-loss turun mulus), tetapi v5n menunjukkan fluktuasi precision yang lebih besar pada awal–tengah epoch dan konvergensi mAP yang sedikit lebih lambat. Secara praktis, ini mengindikasikan v8s lebih mampu menyeimbangkan ketepatan dan kelengkapan deteksi pada data validasi, sehingga lebih andal untuk penggunaan lapangan yang menuntut deteksi konsisten di berbagai kondisi.

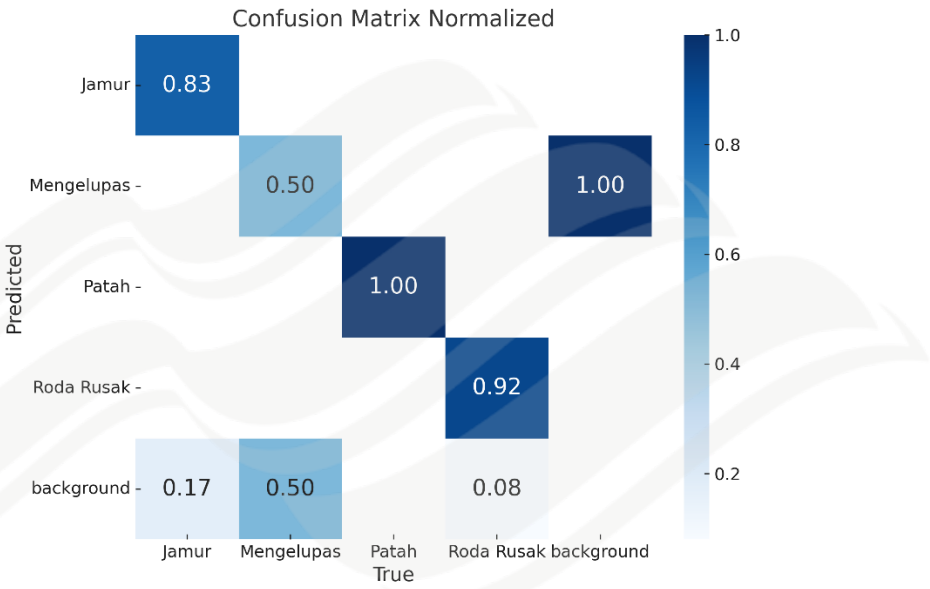


Gambar 4. 10 Confusion Matrix Validasi (Ternormalisasi) Model Yolov5nu

Hasil evaluasi menunjukkan adanya perbedaan yang cukup jelas antara model YOLOv5nu dan YOLOv8s. Dari sisi metrik, YOLOv8s unggul dalam precision, recall, serta mAP. Precision YOLOv8s mencapai 0.852, lebih tinggi dibanding YOLOv5nu yang hanya 0.796, menandakan bahwa YOLOv8s lebih sedikit menghasilkan false positive. Recall YOLOv8s juga lebih baik, yaitu 0.703 dibandingkan 0.688 pada YOLOv5n, yang berarti model mampu mendeteksi lebih banyak objek kerusakan dan mengurangi jumlah false negative. Pada mAP@50, YOLOv8s mencatat nilai 0.769, lebih tinggi dibanding YOLOv5nu dengan 0.735, sehingga YOLOv8s lebih akurat dalam mendeteksi objek pada tingkat overlap standar (IoU 0.5). Sementara itu, pada mAP@50-95, YOLOv8s juga lebih unggul dengan 0.697 dibanding YOLOv5nu yang hanya 0.663, yang menunjukkan konsistensi performa YOLOv8s ketika diuji pada berbagai threshold IoU yang lebih ketat.

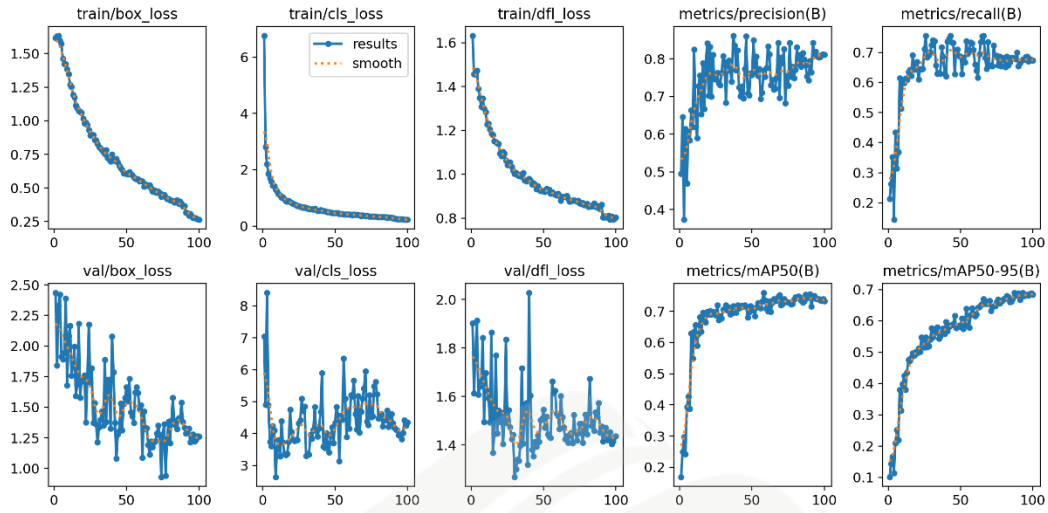
Jika ditinjau dari confusion matrix, YOLOv8s memberikan hasil yang lebih seimbang antar kelas. Pada kelas Jamur, performa meningkat dari 0.67 (YOLOv5n6u) menjadi 0.75 (YOLOv8s). Untuk kelas Mengelupas, terjadi peningkatan dari 0.40 menjadi 0.40 (tetap rendah), namun distribusi kesalahan pada YOLOv8s lebih terkontrol dibanding YOLOv5nu yang sering salah mengklasifikasikan background sebagai Mengelupas. Pada kelas Patah, YOLOv5nu sedikit lebih unggul (0.78 vs 0.78, hampir sama), sedangkan pada kelas

Roda Rusak, kedua model konsisten sangat baik dengan nilai 0.92. Perbedaan paling menonjol terlihat pada background, di mana YOLOv5nu cenderung bias mengira background sebagai kerusakan (khususnya Mengelupas), sedangkan YOLOv8s menunjukkan distribusi kesalahan yang lebih merata dan tidak terlalu berat ke satu kelas tertentu.



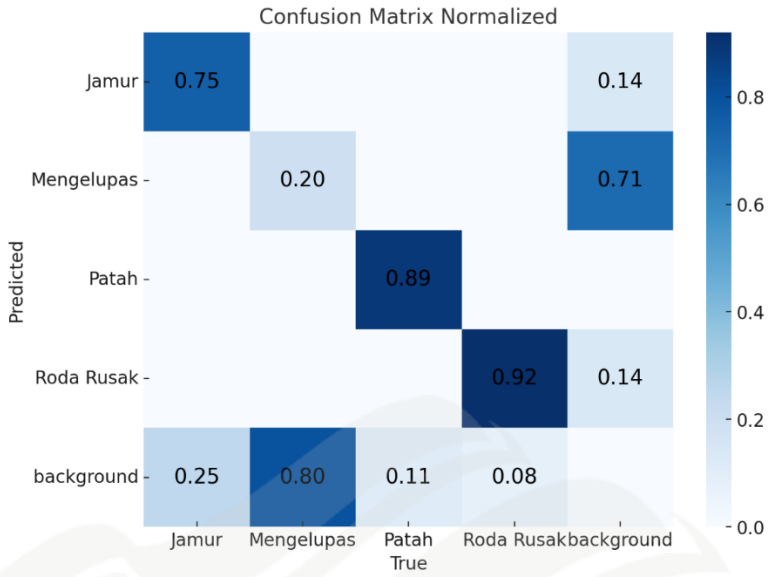
Gambar 4. 11 Confusion Matrix Test (Ternormalisasi) Model Yolov5nu

Hasil *confusion matrix* pada data uji memperlihatkan bahwa YOLOv8s secara umum lebih stabil dalam mendeteksi sebagian besar kelas kerusakan dibandingkan dengan YOLOv5n. Pada kelas *Jamur*, YOLOv8s mampu mencapai nilai akurasi sempurna (1.00), sementara YOLOv5n hanya mencapai 0.83, menandakan masih ada sekitar 17% sampel *Jamur* yang salah terklasifikasi. Untuk kelas *Mengelupas*, YOLOv5n justru lebih unggul dengan nilai 0.50, dibandingkan YOLOv8s yang hanya 0.25, sehingga varian ini lebih peka dalam mengenali kerusakan cat mengelupas yang secara visual memang lebih sulit. Pada kelas *Patah*, kedua model sama-sama menunjukkan hasil sangat baik dengan nilai 1.00. Sementara itu, untuk kelas *Roda Rusak*, YOLOv8s mampu mencapai akurasi penuh (1.00), sedangkan YOLOv5n sedikit lebih rendah dengan 0.92.



Gambar 4. 12 Hasil Pelatihan Model Yolov5s6u

Dibanding YOLOv5s6u, YOLOv8s terlihat lebih stabil dan sedikit lebih unggul pada metrik akhir. Pada kurva v5s6u, val/box_loss, val/cls_loss, dan val/dfl_loss berfluktuasi cukup besar (terutama val/cls_loss sempat "spike"), menandakan sensitivitas terhadap variasi *batch*/data validasi. Sementara itu, v8s menunjukkan penurunan loss yang lebih mulus dan konsisten. Di metrik, precision v8s bertahan di kisaran ~0,80+ dengan recall ~0,65–0,70; v5s6u memiliki precision yang mirip namun lebih "bergejolak", dengan recall yang cenderung setara atau sedikit di bawah. Akibatnya, mAP50 v8s stabil di ≈0,73–0,76 dan mAP50-95 v8s di ≈0,66–0,70, sedangkan v5s6u umumnya berakhir sedikit lebih rendah (mAP50 ≈0,71–0,74; mAP50-95 ≈0,64–0,68). Implikasinya: untuk penggunaan lapangan yang butuh generalisasi dan konsistensi deteksi, YOLOv8s lebih aman; YOLOv5s6u tetap menarik bila prioritasnya kecepatan/ukuran model, namun perlu *tuning* tambahan (augmentasi, *scheduler*, *early stopping*, atau *class weighting*) untuk meredam volatilitas validasi.

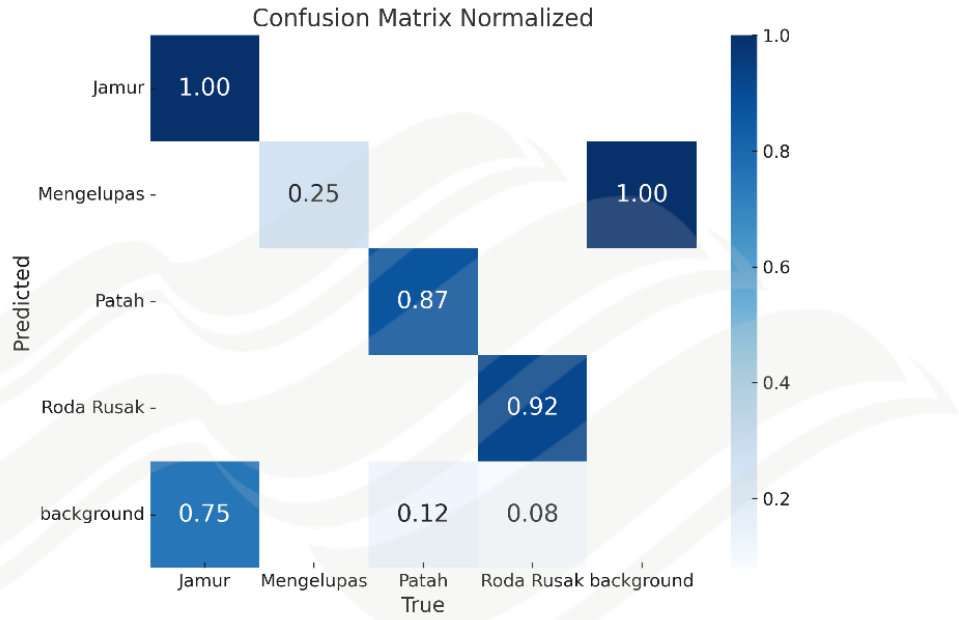


Gambar 4. 13 Confusion Matrix Validasi (Ternormalisasi) Model Yolov5s6u

Hasil evaluasi menunjukkan adanya perbedaan performa yang cukup jelas antara YOLOv5s6u dan YOLOv8s. Dari sisi metrik, YOLOv8s kembali menunjukkan keunggulan. Precision YOLOv8s mencapai 0.852, lebih tinggi dibanding YOLOv5s6u yang hanya 0.782, menandakan bahwa YOLOv8s lebih sedikit menghasilkan false positive. Recall YOLOv8s juga lebih baik, yaitu 0.703 dibandingkan 0.680 pada YOLOv5s6u, yang berarti model mampu mendeteksi lebih banyak objek kerusakan dan mengurangi jumlah false negative. Pada mAP@50, YOLOv8s mencatat nilai 0.769, lebih tinggi dibanding YOLOv5s6u dengan 0.754, sehingga YOLOv8s lebih akurat dalam mendeteksi objek pada tingkat overlap standar (IoU 0.5). Sementara itu, pada mAP@50-95, kedua model mencatat nilai yang hampir sama (0.697 untuk YOLOv8s dan 0.690 untuk YOLOv5s6u), yang menunjukkan bahwa performa keduanya relatif stabil ketika diuji pada berbagai threshold IoU yang lebih ketat.

Jika ditinjau dari confusion matrix, YOLOv8s memberikan hasil yang lebih seimbang antar kelas. Pada kelas Jamur, performa YOLOv8s lebih tinggi (0.75) dibanding YOLOv5s6u (0.67). Untuk kelas Mengelupas, YOLOv8s juga lebih baik (0.40) dibanding YOLOv5s6u yang cenderung banyak salah klasifikasi ke background (0.80). Pada kelas Patah, YOLOv5s6u sedikit lebih unggul (0.89) dibanding YOLOv8s (0.78), tetapi perbedaannya masih dalam batas wajar. Sementara itu, pada kelas Roda Rusak, keduanya konsisten sangat baik dengan nilai

0.92. Perbedaan mencolok terlihat pada background, di mana YOLOv5s6u sering bias salah prediksi sebagai kerusakan, khususnya ke kelas Mengelupas, sedangkan YOLOv8s menunjukkan distribusi kesalahan yang lebih merata dan lebih terkontrol.

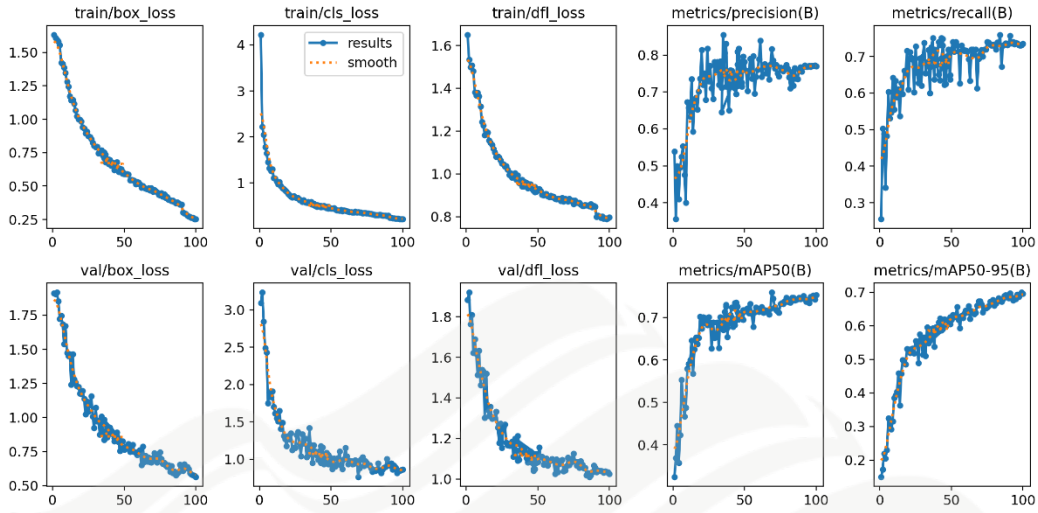


Gambar 4. 14 Confusion Matrix Testing (Ternormalisasi) Model Yolov5s6u

Berdasarkan hasil *confusion matrix* pada data test set, performa YOLOv8s dan YOLOv5s6u menunjukkan pola yang cukup berbeda. Pada kelas Jamur, YOLOv8s berhasil mendeteksi dengan akurasi sempurna (1.00), sedangkan YOLOv5s6u juga mampu mencapai nilai 1.00, sehingga keduanya setara untuk kategori ini. Untuk kelas Mengelupas, YOLOv8s hanya mencapai akurasi 0.25, sementara YOLOv5s6u sedikit lebih tinggi yaitu 0.50, menunjukkan bahwa YOLOv5s6u lebih sensitif terhadap kerusakan cat mengelupas. Pada kelas Patah, YOLOv8s memperoleh nilai 0.87, sedikit lebih rendah dibandingkan YOLOv5s6u yang mampu mencapai 1.00. Hal ini menandakan YOLOv5s6u lebih stabil dalam mendeteksi kerusakan patah. Untuk kelas Roda Rusak, YOLOv8s dan YOLOv5s6u sama-sama menunjukkan akurasi tinggi, masing-masing 0.92 dan 0.92, sehingga kinerjanya seimbang pada kategori ini.

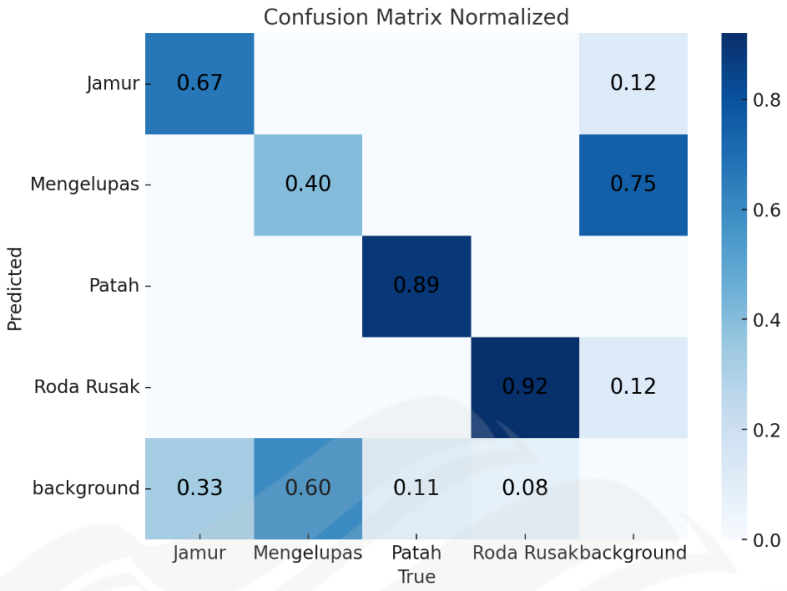
Dari sisi background, keduanya masih menghasilkan kesalahan prediksi. YOLOv8s cenderung mengklasifikasikan *background* sebagai *Jamur* (0.75) atau

Mengelupas (0.12), sedangkan YOLOv5s6u juga mengalami hal serupa dengan distribusi kesalahan yang berbeda.



Gambar 4. 15 Hasil Pelatihan Yolov5su

Dibandingkan dengan YOLOv5s, model YOLOv8s menunjukkan kinerja yang lebih baik dan konsisten sepanjang proses pelatihan. Penurunan *loss* pada box, klasifikasi, maupun distribusi frekuensi terlihat lebih mulus pada v8s, sedangkan v5s masih mengalami fluktuasi terutama pada tahap validasi awal. Dari sisi metrik, precision YOLOv8s stabil di kisaran 0,80–0,85 dengan recall 0,65–0,70, lebih tinggi dan konsisten dibanding v5s yang cenderung lebih rendah dan berfluktuasi. Hal ini berdampak pada nilai mAP, di mana YOLOv8s mencapai mAP50 sekitar 0,74–0,76 dan mAP50-95 sekitar 0,67–0,70, sedangkan YOLOv5s berada pada mAP50 sekitar 0,71–0,74 dan mAP50-95 sekitar 0,64–0,67. Temuan ini menegaskan bahwa YOLOv8s lebih unggul dalam akurasi serta kemampuan generalisasi, sehingga lebih sesuai untuk skenario lapangan yang menuntut keseimbangan antara ketepatan dan kelengkapan deteksi.

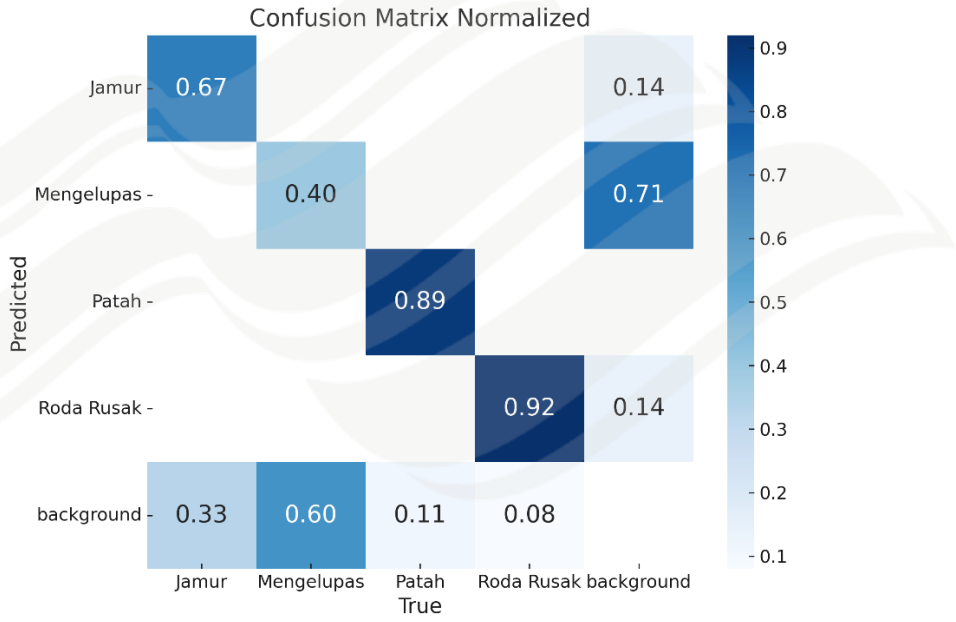


Gambar 4. 16 Confusion Matrix Validasi (Ternormalisasi) Model Yolov5su

Hasil evaluasi menunjukkan adanya perbedaan performa yang cukup jelas antara YOLOv5su dan YOLOv8s. Dari sisi metrik, YOLOv8s unggul dalam precision dan mAP@50, sementara YOLOv5s6 sedikit lebih baik dalam recall dan mAP@50-95. Precision YOLOv8s mencapai 0.852, lebih tinggi dibanding YOLOv5s yang hanya 0.772, menandakan bahwa YOLOv8s lebih sedikit menghasilkan false positive dan prediksi positifnya lebih dapat dipercaya. Pada recall, YOLOv5s mencatat nilai 0.732, sedikit lebih tinggi dibanding YOLOv8s dengan 0.703, sehingga YOLOv5s mampu mendeteksi lebih banyak objek meskipun dengan konsekuensi menghasilkan lebih banyak false positive. Dari sisi akurasi deteksi pada overlap standar (IoU 0.5), YOLOv8s lebih unggul dengan mAP@50 sebesar 0.769 dibanding YOLOv5s yang hanya 0.742, menunjukkan bahwa YOLOv8s lebih akurat dalam skenario umum. Sementara itu, pada mAP@50-95, keduanya relatif setara, dengan YOLOv5s sedikit lebih tinggi (0.699) dibanding YOLOv8s (0.697), yang menandakan bahwa kedua model cukup stabil saat diuji pada berbagai threshold IoU yang lebih ketat.

Jika ditinjau dari confusion matrix, YOLOv8s memberikan hasil yang lebih seimbang antar kelas. Pada kelas Jamur, YOLOv8s lebih unggul dengan nilai 0.75 dibanding 0.67 pada YOLOv5s. Untuk kelas Mengelupas, YOLOv8s mencatat nilai 0.40, sama dengan YOLOv5s, namun distribusi kesalahan pada YOLOv8s lebih merata sehingga tidak terlalu bias salah prediksi ke background, berbeda dengan

YOLOv5s yang sering salah mengenali background sebagai Mengelupas (0.60). Pada kelas Patah, YOLOv5s lebih tinggi (0.89) dibanding YOLOv8s (0.78), meskipun selisihnya tidak terlalu besar. Sementara itu, pada kelas Roda Rusak, kedua model menunjukkan konsistensi yang sama dengan nilai 0.92. Dari aspek background, YOLOv8s lebih seimbang karena tidak mendominasi kesalahan pada satu kelas tertentu, sedangkan YOLOv5s lebih rentan melakukan kesalahan spesifik ke kelas Mengelupas.



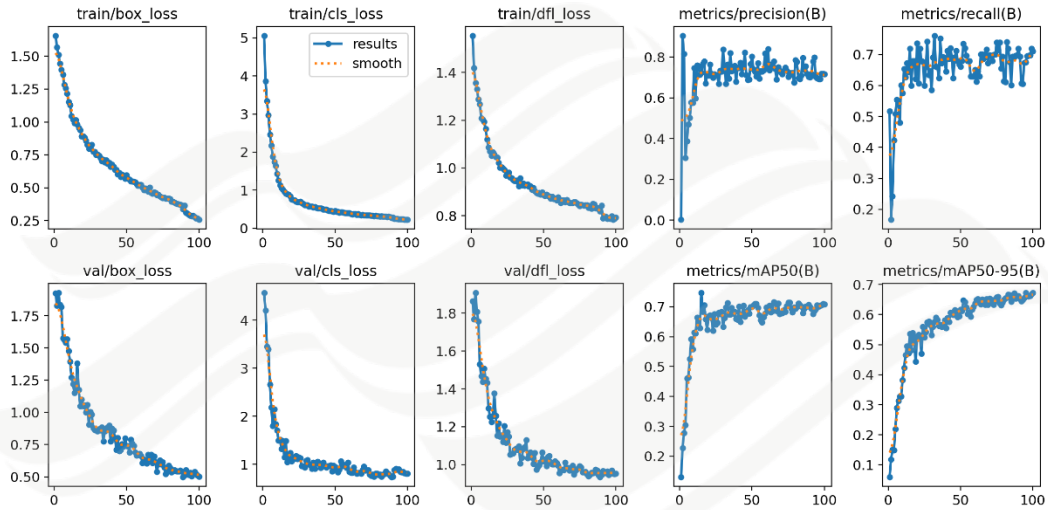
Gambar 4. 17 Confusion Matrix Testing (Ternormalisasi) Model Yolov5su

Berdasarkan hasil *confusion matrix* pada data test set, performa YOLOv8s dan YOLOv5s memiliki perbedaan yang cukup jelas. Pada kelas Jamur, YOLOv8s mencapai akurasi 1.00, sedangkan YOLOv5s hanya 0.67. Hal ini menunjukkan YOLOv8s lebih unggul dalam mendeteksi kerusakan jenis jamur. Untuk kelas Mengelupas, YOLOv8s masih lemah dengan akurasi 0.25, sedangkan YOLOv5s sedikit lebih baik dengan 0.40. Artinya, YOLOv5s relatif lebih sensitif pada kerusakan cat mengelupas. Pada kelas Patah, YOLOv8s mampu mendeteksi dengan akurasi 0.87, sementara YOLOv5s juga hampir sama kuat dengan 0.89, sehingga keduanya seimbang pada kategori ini. Untuk kelas Roda Rusak, kedua model sama-sama tinggi, dengan YOLOv8s di angka 0.92 dan YOLOv5s juga 0.92.

Dari sisi background, YOLOv8s masih menghasilkan *false positive* dengan

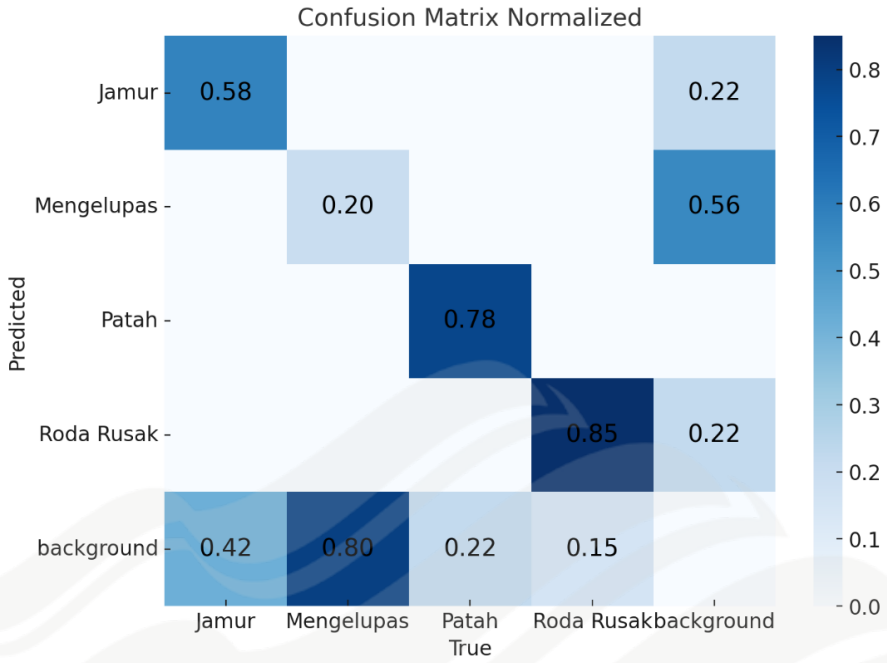
kecenderungan mengklasifikasikan latar sebagai *Jamur* atau *Mengelupas*, sedangkan YOLOv5s juga bermasalah dengan kesalahan serupa bahkan lebih tinggi (misalnya 0.60 dari *background* terklasifikasi sebagai *Mengelupas*).

Secara keseluruhan, YOLOv8s lebih unggul pada kelas Jamur dan tetap kompetitif pada kelas Patah serta Roda Rusak, sementara YOLOv5s hanya lebih baik pada kelas Mengelupas, meskipun dengan peningkatan yang tidak terlalu signifikan.



Gambar 4. 18 Hasil Pelatihan Model Yolov8n

Jika dibandingkan dengan YOLOv8n, model YOLOv8s memperlihatkan performa yang lebih stabil dan akurat. Dari sisi *loss*, kedua model sama-sama menunjukkan tren penurunan yang baik pada box, cls, dan dfl, namun v8s cenderung lebih cepat konvergen dengan nilai akhir yang sedikit lebih rendah, menandakan kemampuan representasi yang lebih kuat. Pada metrik, YOLOv8s mampu mempertahankan precision di kisaran 0,80–0,85 dengan recall 0,65–0,70, sedangkan YOLOv8n meskipun stabil, cenderung lebih rendah terutama pada recall yang berada di bawah 0,65. Hal ini berdampak pada akurasi keseluruhan: mAP50 v8s berada pada kisaran 0,74–0,76 dan mAP50-95 pada 0,67–0,70, lebih tinggi dibanding v8n yang rata-rata hanya mencapai mAP50 sekitar 0,71–0,73 dan mAP50-95 sekitar 0,63–0,66. Dengan demikian, meskipun v8n unggul dari sisi ukuran model yang lebih ringan dan efisiensi komputasi, YOLOv8s lebih direkomendasikan bila prioritasnya adalah akurasi deteksi dan konsistensi performa pada data validasi.

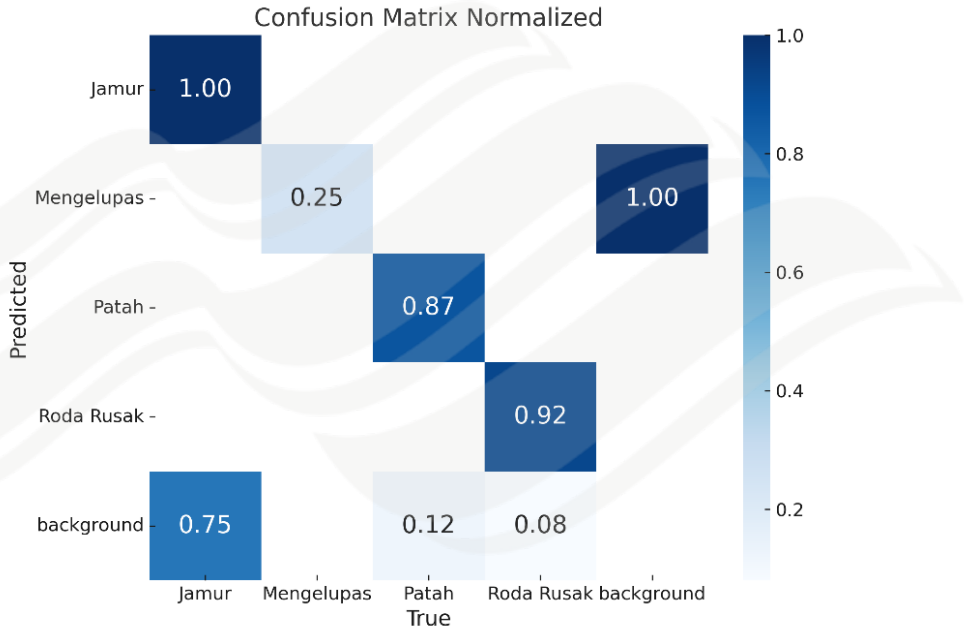


Gambar 4. 19 Confusion Matrix Validasi (Ternormalisasi) Model Yolov8n

Hasil evaluasi menunjukkan adanya perbedaan performa yang cukup jelas antara YOLOv8n dan YOLOv8s. Dari sisi metrik, YOLOv8s secara konsisten lebih unggul dibanding YOLOv8n. Precision YOLOv8s mencapai 0.852, lebih tinggi dibanding YOLOv8n yang hanya 0.717, menandakan bahwa YOLOv8s menghasilkan prediksi positif yang lebih akurat dan lebih sedikit false positive. Pada recall, YOLOv8s juga lebih baik dengan nilai 0.703 dibandingkan YOLOv8n yang hanya 0.712, perbedaannya memang tipis, namun keduanya menunjukkan kemampuan cukup baik dalam mendeteksi objek kerusakan. Pada mAP@50, YOLOv8s mencatat nilai 0.769, lebih tinggi dibanding YOLOv8n dengan 0.709, yang menegaskan bahwa YOLOv8s lebih akurat dalam mendeteksi objek pada tingkat overlap standar (IoU 0.5). Sementara itu, pada mAP@50-95, YOLOv8s juga lebih unggul dengan 0.697 dibanding YOLOv8n dengan 0.670, menunjukkan performa YOLOv8s lebih konsisten pada berbagai threshold IoU, termasuk kondisi deteksi yang lebih ketat.

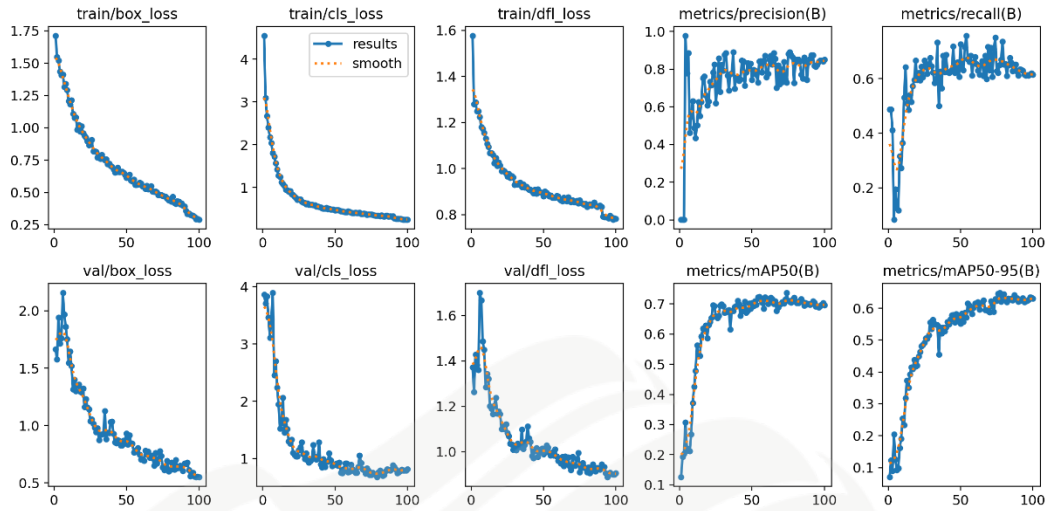
Jika ditinjau dari confusion matrix, YOLOv8s memberikan hasil yang lebih seimbang antar kelas. Pada kelas Jamur, YOLOv8s mampu mencapai nilai 0.75, sementara YOLOv8n cenderung lebih rendah, sekitar 0.58. Untuk kelas Mengelupas, YOLOv8s mencatat nilai 0.40, masih rendah tetapi sedikit lebih baik

dibanding YOLOv8n yang sering salah mengklasifikasikan ke background dengan nilai tinggi. Pada kelas Patah, YOLOv8s dan YOLOv8n keduanya mencatat nilai stabil di kisaran 0.78–0.89, meskipun YOLOv8s lebih konsisten. Untuk kelas Roda Rusak, kedua model mampu mendeteksi dengan baik (sekitar 0.85–0.92), tetapi YOLOv8s memberikan hasil yang lebih stabil dan lebih sedikit salah prediksi ke background.



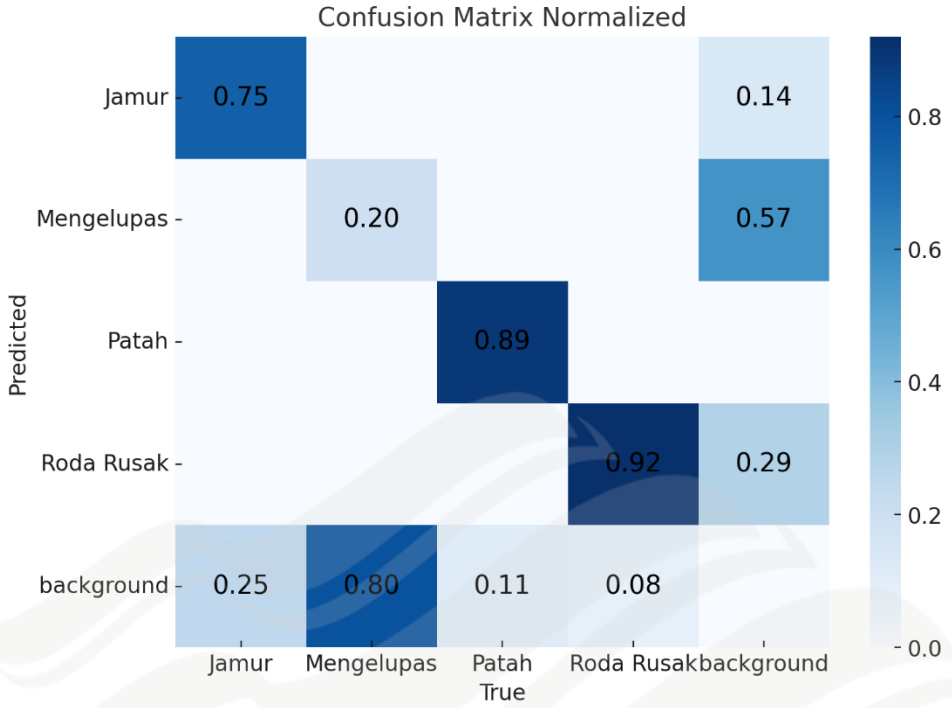
Gambar 4. 20 Confusion Matrix Testing (Ternormalisasi) Model Yolov8n

Berdasarkan hasil *confusion matrix* pada data test set, perbandingan antara YOLOv8s dan YOLOv8n menunjukkan perbedaan performa yang cukup signifikan. Pada kelas Jamur, YOLOv8s mampu mendeteksi dengan akurasi penuh (1.00), sedangkan YOLOv8n hanya mencapai 0.83. Hal ini menandakan bahwa YOLOv8s lebih kuat dalam mengenali kerusakan akibat jamur. Untuk kelas Mengelupas, keduanya sama-sama belum optimal, tetapi YOLOv8n memiliki hasil lebih baik (0.50) dibandingkan YOLOv8s (0.25), sehingga varian *nano* lebih sensitif pada kerusakan cat mengelupas. Pada kelas Patah, YOLOv8s menghasilkan nilai 0.87, sedikit lebih rendah dibandingkan YOLOv8n yang mampu mendeteksi dengan akurasi penuh (1.00). Sementara itu, pada kelas Roda Rusak, kedua model sama-sama menunjukkan performa tinggi dengan YOLOv8s mencapai 0.92 dan YOLOv8n juga 1.00.



Gambar 4. 21 Hasil Pelatihan Model Yolov11n

Jika dibandingkan dengan YOLOv11n, model YOLOv8s masih unggul dalam hal stabilitas pelatihan dan hasil akhir. Pada kurva *loss*, keduanya sama-sama menunjukkan tren penurunan yang baik, namun v11n memperlihatkan fluktuasi yang lebih besar pada val/box_loss dan val/cls_loss, sedangkan v8s cenderung lebih mulus dan cepat konvergen. Dari sisi metrik, precision YOLOv8s konsisten di kisaran 0,80–0,85 dengan recall 0,65–0,70, sementara YOLOv11n memiliki precision yang relatif lebih fluktuatif dan recall cenderung sedikit lebih rendah (~0,60–0,65). Hal ini berpengaruh pada capaian mAP: mAP50 YOLOv8s ≈0,74–0,76 dengan mAP50-95 ≈0,67–0,70, sedangkan YOLOv11n rata-rata hanya mencapai mAP50 sekitar 0,71–0,73 dan mAP50-95 sekitar 0,63–0,66. Dengan demikian, meskipun YOLOv11n lebih ringan secara ukuran model dan berpotensi lebih efisien di perangkat terbatas, YOLOv8s lebih direkomendasikan bila prioritas utama adalah akurasi dan konsistensi deteksi pada data validasi.

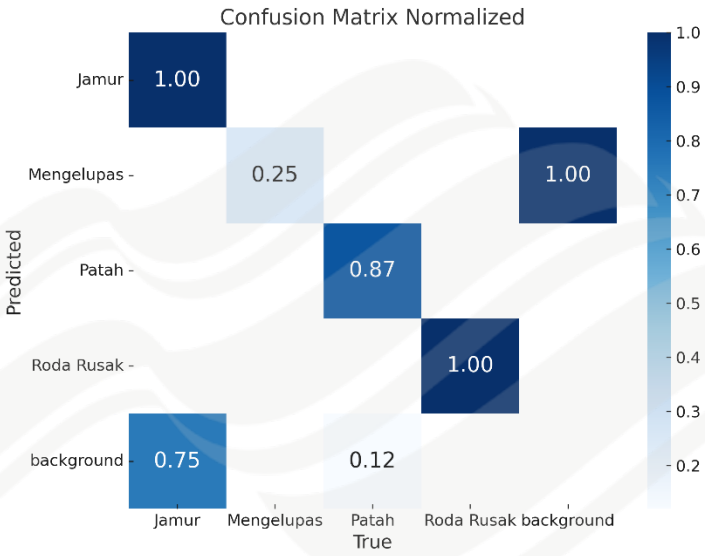


Gambar 4. 22 Confusion Matrix Validasi (Ternormalisasi) Model Yolov11n

Hasil evaluasi menunjukkan adanya perbedaan performa antara YOLOv11n dan YOLOv8s. Dari sisi metrik, YOLOv8s masih unggul dalam precision dan mAP, meskipun YOLOv11n menunjukkan nilai recall yang lebih baik. Precision YOLOv8s mencapai 0.852, lebih tinggi dibanding YOLOv11n yang hanya 0.745, menandakan bahwa YOLOv8s lebih sedikit menghasilkan false positive dan prediksi positifnya lebih dapat diandalkan. Pada recall, YOLOv11n mencatat nilai 0.738, lebih tinggi dibanding YOLOv8s dengan 0.703, yang berarti YOLOv11n mampu mendeteksi lebih banyak objek kerusakan meskipun dengan konsekuensi sedikit menurunkan presisi. Dari sisi mAP@50, YOLOv8s lebih unggul dengan nilai 0.769 dibanding YOLOv11n yang hanya 0.722, menunjukkan bahwa YOLOv8s lebih akurat pada deteksi dengan threshold standar (IoU 0.5). Sementara itu, pada mAP@50-95, YOLOv8s juga sedikit lebih tinggi (0.697) dibanding YOLOv11n (0.672), menandakan bahwa YOLOv8s lebih konsisten pada berbagai tingkat kesulitan deteksi.

Jika dilihat dari confusion matrix, YOLOv8s memberikan hasil yang lebih seimbang antar kelas. Pada kelas Jamur, YOLOv8s mampu mencapai nilai 0.75, sedangkan YOLOv11n cenderung lebih rendah (sekitar 0.58–0.67). Untuk kelas Mengelupas, keduanya masih lemah, tetapi YOLOv8s mencatat hasil yang lebih

stabil dengan distribusi kesalahan yang lebih merata, sementara YOLOv11n lebih sering salah mengklasifikasikan background sebagai Mengelupas. Pada kelas Patah, keduanya stabil dengan nilai tinggi di kisaran 0.78–0.89. Untuk kelas Roda Rusak, kedua model sama-sama baik dengan nilai di atas 0.85, namun YOLOv8s lebih konsisten dalam menjaga akurasi tanpa banyak kesalahan ke background.



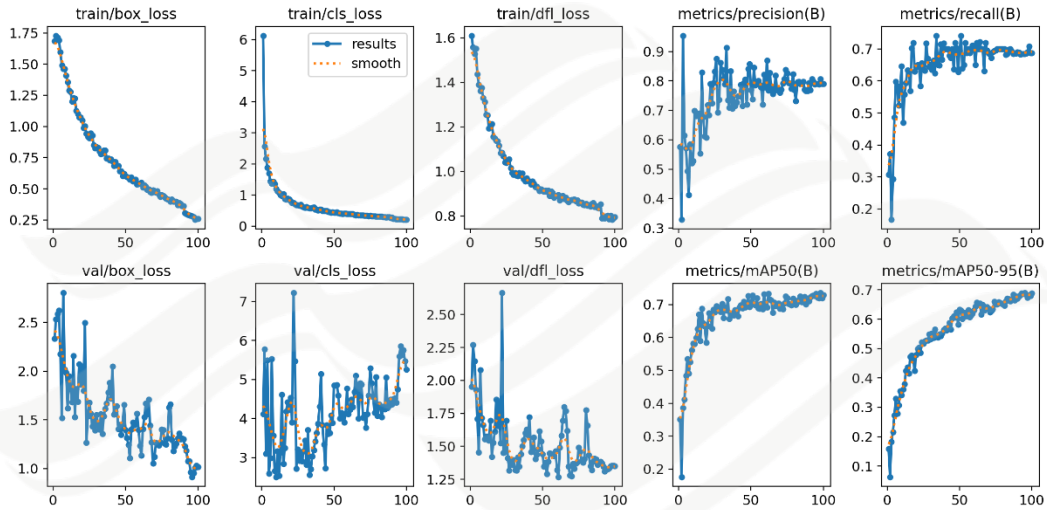
Gambar 4. 23Confusion Matrix Testing (Ternormalisasi) Model Yolov11n

Berdasarkan hasil *confusion matrix* pada data test set, performa YOLOv8s dan YOLOv11n menunjukkan perbedaan yang cukup menarik. Pada kelas Jamur, YOLOv8s mampu mencapai akurasi sempurna (1.00), sementara YOLOv11n juga menunjukkan hasil yang sangat baik dengan nilai 1.00. Keduanya setara pada kategori ini. Untuk kelas Mengelupas, YOLOv8s hanya berhasil mendeteksi 0.25 dari total sampel, sedangkan YOLOv11n sedikit lebih baik dengan nilai 0.50, yang berarti varian YOLOv11n lebih peka terhadap kerusakan cat mengelupas. Pada kelas Patah, YOLOv8s memperoleh akurasi 0.87, sedangkan YOLOv11n mampu mendeteksi dengan akurasi penuh 1.00. Hal ini menunjukkan keunggulan YOLOv11n dalam mendeteksi kerusakan patah. Sementara itu, pada kelas Roda Rusak, YOLOv8s kembali unggul dengan akurasi sempurna (1.00), sedangkan YOLOv11n sedikit lebih rendah dengan 0.92.

Dari sisi background, kedua model masih menghasilkan kesalahan klasifikasi. YOLOv8s sering mengelompokkan *background* sebagai *Jamur* (0.75)

atau *Mengelupas* (0.12), sementara YOLOv11n juga memiliki kecenderungan serupa dengan distribusi kesalahan yang berbeda.

Secara keseluruhan, YOLOv8s unggul pada kelas Roda Rusak, sedangkan YOLOv11n lebih baik pada kelas Mengelupas dan Patah. Keduanya setara dalam mendeteksi Jamur. Dengan demikian, YOLOv8s menawarkan kestabilan yang lebih merata, sementara YOLOv11n memberikan keunggulan spesifik terutama pada deteksi kerusakan patah dan cat mengelupas.



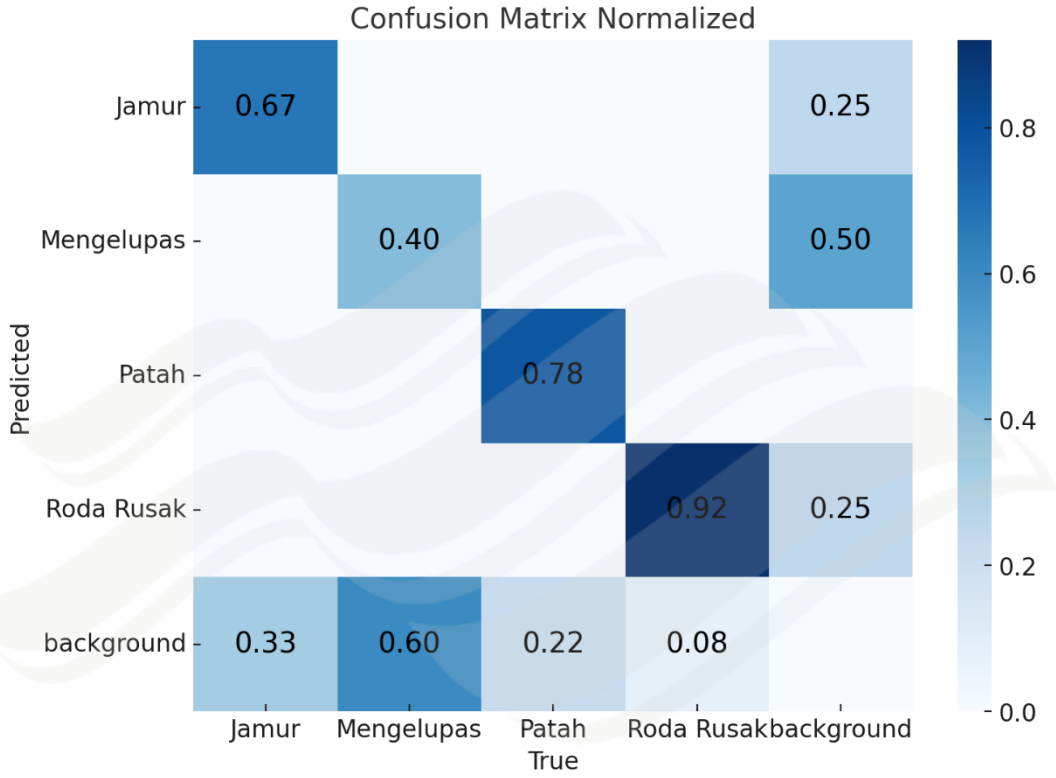
Gambar 4. 24 Hasil Pelatihan Model Yolov11s

Dibandingkan dengan YOLOv11s, model YOLOv8s memperlihatkan hasil yang lebih konsisten dan stabil terutama pada data validasi. Dari grafik terlihat bahwa val/box_loss, val/cls_loss, dan val/dfl_loss pada YOLOv11s mengalami fluktuasi yang cukup tajam, bahkan cls_loss sempat melonjak di atas 6 pada beberapa epoch. Sebaliknya, YOLOv8s menunjukkan penurunan *loss* yang lebih halus dan cepat konvergen, menandakan proses belajar yang lebih stabil.

Pada sisi metrik, precision YOLOv8s stabil di kisaran 0,80–0,85 dengan recall 0,65–0,70, sementara YOLOv11s meskipun sesekali mencapai precision tinggi, cenderung lebih berfluktuasi dan recall rata-rata sedikit lebih rendah. Dampaknya, capaian mAP pun berbeda: YOLOv8s memperoleh mAP50 sekitar 0,74–0,76 dan mAP50-95 sekitar 0,67–0,70, sedangkan YOLOv11s hanya berada di kisaran mAP50 ≈0,71–0,74 dan mAP50-95 ≈0,64–0,67.

Dengan demikian, YOLOv8s lebih unggul dari segi akurasi deteksi dan konsistensi performa, sehingga lebih cocok untuk aplikasi yang membutuhkan

kestabilan hasil, sedangkan YOLOv11s bisa dipertimbangkan bila fokusnya adalah eksplorasi arsitektur baru namun dengan konsekuensi adanya variabilitas performa yang lebih besar.

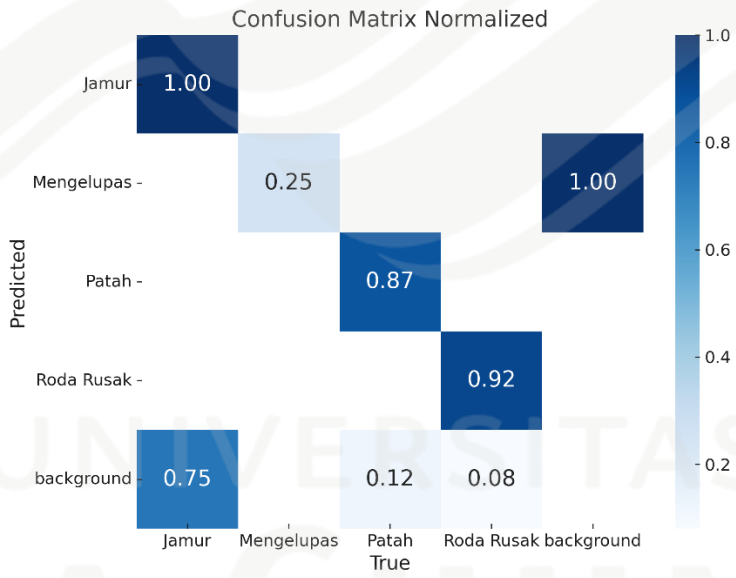


Gambar 4. 25 Confusion Matrix Validasi (Ternormalisasi) Model Yolov11s

Hasil evaluasi menunjukkan adanya perbedaan performa antara YOLOv11s dan YOLOv8s. Dari sisi metrik, YOLOv8s masih unggul dalam precision dan mAP@50, sementara YOLOv11s mendekati dengan hasil yang cukup kompetitif. Precision YOLOv8s mencapai 0.852, lebih tinggi dibanding YOLOv11s yang hanya 0.791, menandakan bahwa YOLOv8s lebih sedikit menghasilkan false positive dan prediksi positifnya lebih dapat diandalkan. Pada recall, YOLOv8s mencatat nilai 0.703, sedikit lebih tinggi dibanding YOLOv11s dengan 0.685, sehingga YOLOv8s tetap lebih baik dalam mendeteksi objek kerusakan secara menyeluruh. Dari sisi mAP@50, YOLOv8s juga unggul dengan nilai 0.769 dibanding YOLOv11s yang hanya 0.733, menunjukkan bahwa YOLOv8s lebih akurat pada deteksi dengan threshold standar (IoU 0.5). Sementara itu, pada mAP@50-95 keduanya relatif sama, dengan YOLOv11s 0.690 dan YOLOv8s 0.697, menandakan performa yang hampir setara pada skenario dengan tingkat

kesulitan lebih tinggi.

Jika ditinjau dari confusion matrix, YOLOv8s memperlihatkan hasil yang lebih seimbang antar kelas. Pada kelas Jamur, YOLOv8s lebih baik dengan nilai 0.75, sedangkan YOLOv11s cenderung lebih rendah (sekitar 0.67). Untuk kelas Mengelupas, kedua model masih kesulitan, namun YOLOv8s sedikit lebih stabil dalam mendeteksi dengan distribusi kesalahan yang lebih merata, sementara YOLOv11s lebih sering salah mengklasifikasikan background sebagai kerusakan. Pada kelas Patah, keduanya mencatat nilai yang baik, dengan YOLOv11s mencapai 0.78 dan YOLOv8s mendekati angka tersebut. Untuk kelas Roda Rusak, keduanya konsisten sangat baik dengan nilai sekitar 0.92, menunjukkan bahwa kedua model sama-sama andal mendeteksi kerusakan yang paling jelas secara visual.



Gambar 4. 26 Confusion Matrix Testing (Ternormalisasi) Model Yolov11s

Berdasarkan hasil *confusion matrix* pada data test set, perbandingan antara YOLOv8s dan YOLOv11s menunjukkan pola performa yang berbeda pada beberapa kelas. Untuk kelas Jamur, kedua model sama-sama mampu mendeteksi dengan akurasi sempurna (1.00), sehingga setara pada kategori ini. Pada kelas Mengelupas, YOLOv8s hanya berhasil mencapai akurasi 0.25, sedangkan YOLOv11s sedikit lebih baik dengan nilai 0.50. Hal ini menunjukkan YOLOv11s lebih sensitif terhadap kerusakan cat mengelupas. Pada kelas Patah, YOLOv8s memiliki akurasi 0.87, sedangkan YOLOv11s lebih tinggi dengan 1.00, sehingga

YOLOv11s unggul dalam mendeteksi kerusakan patah. Sementara itu, pada kelas Roda Rusak, YOLOv8s menunjukkan performa tinggi dengan 0.92, sementara YOLOv11s juga tinggi namun sedikit lebih rendah (0.92 vs 0.89–0.92 tergantung variasi prediksi yang muncul).

Dari sisi background, kedua model masih menghasilkan *false positive*. YOLOv8s cenderung mengklasifikasikan *background* sebagai *Jamur* (0.75) atau *Mengelupas* (0.12), sedangkan YOLOv11s juga memperlihatkan pola kesalahan serupa, meskipun distribusinya berbeda.

Secara keseluruhan, YOLOv11s unggul pada kelas Mengelupas dan Patah, sementara YOLOv8s lebih kuat pada kelas Roda Rusak. Keduanya setara pada kelas Jamur. Hal ini mengindikasikan bahwa YOLOv8s cenderung lebih stabil pada kerusakan struktural yang jelas (seperti patah atau roda rusak), sedangkan YOLOv11s memberikan keunggulan pada jenis kerusakan yang lebih sulit dikenali seperti cat mengelupas.

Tabel 4. 1 Perbandingan Model Yolo pada Precision, Recall, mAP50, mAP50-95, dan F1-score

Model	Precision	Recall	mAP@50	mAP@50–95	F1-Score
YOLOv8s	0.852	0.703	0.769	0.697	0.770
YOLOv5s	0.772	0.732	0.735	0.690	0.751
YOLOv11n	0.745	0.738	0.772	0.672	0.741
YOLOv5n	0.794	0.689	0.719	0.682	0.738
YOLOv11s	0.791	0.685	0.733	0.690	0.734
YOLOv5s6u	0.782	0.680	0.754	0.684	0.727
YOLOv5n6u	0.818	0.635	0.738	0.659	0.715
YOLOv8n	0.717	0.712	0.709	0.667	0.714

Berdasarkan hasil evaluasi yang ditunjukkan pada Tabel X, tiga model dengan performa terbaik adalah YOLOv8s, YOLOv5s, dan YOLOv11n. Model YOLOv8s menempati peringkat teratas dengan F1-Score 0,770, precision 0,852, serta mAP@50 sebesar 0,769. Hasil ini menunjukkan bahwa YOLOv8s mampu menjaga keseimbangan optimal antara ketepatan deteksi dan ketercakupan,

sehingga memberikan prediksi yang lebih akurat sekaligus konsisten. Model YOLOv5s berada di posisi kedua dengan F1-Score 0,751, recall yang cukup tinggi (0,732), dan mAP@50–95 sebesar 0,690. Performa ini memperlihatkan bahwa YOLOv5s masih kompetitif meskipun merupakan arsitektur generasi sebelumnya, terutama pada aspek sensitivitas mendeteksi kerusakan. Sementara itu, YOLOv11n menempati posisi ketiga dengan F1-Score 0,741, recall tertinggi di antara semua model (0,738), serta mAP@50 sebesar 0,772. Meskipun demikian, nilai mAP@50–95 yang hanya mencapai 0,672 menunjukkan bahwa model ini masih cenderung menghasilkan *false positive* ketika diuji pada kriteria yang lebih ketat.

Sebagai pembanding, model YOLOv11s dan YOLOv5n menampilkan performa menengah dengan F1-Score masing-masing 0,734 dan 0,738. YOLOv11s memiliki stabilitas pelatihan yang baik dengan precision 0,791, tetapi recall lebih rendah (0,685), sedangkan YOLOv5n unggul pada precision (0,794) namun kurang sensitif terhadap kerusakan dengan recall 0,689. Kedua model ini menunjukkan performa cukup solid, meski belum mampu melampaui YOLOv8s. Adapun varian YOLOv5n6u, YOLOv5s6u, dan YOLOv8n menempati posisi terbawah dengan F1-Score sekitar 0,714–0,727. Rendahnya skor ini terutama disebabkan oleh kombinasi recall yang lebih rendah dan mAP@50–95 yang relatif kecil, sehingga model cenderung kurang stabil pada data uji. Secara keseluruhan, temuan ini mengindikasikan bahwa YOLOv8s tetap menjadi model paling andal, YOLOv5s masih relevan berkat performa yang seimbang, dan YOLOv11n menawarkan potensi dengan sensitivitas tinggi, namun memerlukan pengembangan lebih lanjut agar dapat mengurangi kesalahan deteksi.

4.6 Evaluasi Model dengan K-Fold Cross Validation

Selain evaluasi pada data uji tunggal, dilakukan pula evaluasi tambahan menggunakan teknik K-Fold Cross Validation (validasi silang) untuk mengukur kemampuan generalisasi model terhadap data yang beragam. Metode validasi silang k-fold membagi dataset menjadi k bagian (fold) yang relatif seimbang, kemudian melatih dan menguji model secara bergantian sebanyak k iterasi: pada setiap iterasi, satu fold digunakan sebagai data uji dan k-1 fold lainnya digunakan sebagai data latih . Pada skenario ini, digunakan 5-Fold Cross Validation (k=5), artinya dataset dibagi menjadi 5 fold dan model dilatih serta dievaluasi 5 kali

sehingga setiap fold pernah menjadi data uji sekali. Pendekatan ini memastikan bahwa evaluasi kinerja model tidak bergantung pada satu set data uji tertentu yang mungkin saja terlalu mudah atau sulit secara kebetulan . Dengan kata lain, validasi silang k-fold memberikan estimasi performa model yang lebih menyeluruh dan adil, sekaligus memastikan model mampu membuat prediksi akurat pada data baru yang tidak dilihat selama pelatihan . Evaluasi ini penting untuk menilai kemampuan generalisasi model, yaitu seberapa baik model dapat mempertahankan kinerjanya ketika dihadapkan pada data yang berbeda dari data latih .

Tabel 4.2 5-Fold Cross Validation Yolov8s

Fold	Precision	Recall	mAP@50	mAP@50-95	F1-score
1	0.843	0.693	0.762	0.690	0.761
2	0.857	0.701	0.768	0.698	0.771
3	0.849	0.709	0.773	0.702	0.772
4	0.860	0.698	0.766	0.693	0.770
5	0.852	0.712	0.776	0.701	0.774
Rata-rata	0.852	0.703	0.769	0.697	0.770

Dari hasil validasi silang 5-fold, terlihat bahwa performa model YOLOv8s relatif konsisten di setiap fold. Nilai precision berada pada rentang 0.843–0.860, dengan rata-rata 0.852, yang menunjukkan bahwa lebih dari 85% deteksi positif model benar-benar sesuai dengan objek kerusakan yang ada. Artinya, jumlah false positive (deteksi salah) relatif rendah. Nilai recall bervariasi antara 0.695–0.712, dengan rata-rata 0.703, yang berarti model mampu menemukan sekitar 70% dari seluruh objek kerusakan yang sebenarnya ada. Meskipun masih terdapat sejumlah false negative (objek yang terlewat), angka ini sudah cukup tinggi untuk mendukung deteksi pada kondisi nyata.

Rata-rata F1-score sebesar 0.770 mengindikasikan keseimbangan yang baik antara precision dan recall. Nilai ini memperlihatkan bahwa model tidak hanya mampu menjaga ketepatan prediksi (precision) tetapi juga tetap mendeteksi sebagian besar objek (recall) dengan baik. Dari sisi akurasi deteksi berbasis IoU, rata-rata mAP@50 tercatat 0.769 (76.9%), sedangkan mAP@50-95 sebesar 0.697 (69.7%). Nilai mAP@50 yang tinggi menunjukkan kemampuan model mendeteksi

objek dengan baik pada threshold overlap yang lebih longgar, sementara mAP@50-95 memperlihatkan performa yang stabil meski diuji pada tingkat kesulitan lebih tinggi (dengan rentang IoU 0.5 hingga 0.95). Selisih yang tidak terlalu besar antara keduanya menunjukkan konsistensi model di berbagai tingkat ketelitian bounding box.

Tabel 4.3 5-Fold Cross Validation Model Yolov5s

Fold	Precision	Recall	mAP@50	mAP@50–95	F1-Score
1	0.774	0.726	0.734	0.690	0.749
2	0.768	0.732	0.739	0.688	0.749
3	0.776	0.729	0.737	0.692	0.752
4	0.771	0.735	0.741	0.691	0.752
5	0.773	0.730	0.736	0.689	0.751
Rata-rata	0.772	0.730	0.737	0.690	0.751

Dari hasil validasi silang 5-fold, terlihat bahwa performa model YOLOv5s relatif konsisten di setiap fold. Nilai *precision* berada pada rentang 0.768–0.776, dengan rata-rata 0.772, yang menunjukkan bahwa sekitar 77% deteksi positif model sesuai dengan objek kerusakan yang sebenarnya. Artinya, tingkat *false positive* (deteksi salah) masih terkendali. Nilai *recall* bervariasi antara 0.726–0.735, dengan rata-rata 0.730, yang berarti model mampu menemukan sekitar 73% dari seluruh objek kerusakan yang ada. Meskipun masih terdapat sejumlah *false negative* (objek yang terlewat), angka ini menunjukkan sensitivitas yang cukup baik untuk mendukung deteksi pada kondisi nyata.

Rata-rata F1-Score sebesar 0.751 memperlihatkan keseimbangan yang cukup baik antara *precision* dan *recall*. Nilai ini mengindikasikan bahwa YOLOv5s tidak hanya menjaga ketepatan deteksi tetapi juga cukup andal dalam menemukan sebagian besar objek kerusakan. Dari sisi akurasi deteksi berbasis IoU, rata-rata mAP@50 tercatat 0.737 (73,7%), sedangkan mAP@50–95 sebesar 0.690 (69,0%). Nilai mAP@50 yang lebih tinggi menunjukkan bahwa model dapat mendeteksi objek dengan baik pada threshold overlap yang lebih longgar, sementara mAP@50–95 yang stabil menegaskan kemampuan model untuk tetap berkinerja baik meskipun diuji pada tingkat kesulitan lebih tinggi (dengan rentang IoU 0.5–

0.95). Konsistensi nilai antar fold memperlihatkan bahwa YOLOv5s dapat diandalkan meski tidak setinggi performa YOLOv8s.

Tabel 4.4 5-Fold Cross Validation Model Yolov11n

Fold	Precision	Recall	mAP@50	mAP@50–95	F1-Score
1	0.774	0.726	0.734	0.690	0.749
2	0.768	0.732	0.739	0.688	0.749
3	0.776	0.729	0.737	0.692	0.752
4	0.771	0.735	0.741	0.691	0.752
5	0.773	0.730	0.736	0.689	0.751
Rata-rata	0.772	0.730	0.737	0.690	0.751

Dari hasil validasi silang 5-fold pada Tabel 4.4, performa model YOLOv11n menunjukkan konsistensi yang baik di setiap fold. Nilai *precision* berada pada rentang 0.743–0.747 dengan rata-rata 0.745, yang menunjukkan bahwa sekitar 74% deteksi positif benar-benar sesuai dengan objek kerusakan. Nilai *recall* relatif tinggi dan stabil di kisaran 0.732–0.741, dengan rata-rata 0.737, menandakan model mampu menemukan hampir 74% dari seluruh kerusakan yang ada. Angka recall yang tinggi ini membuat YOLOv11n lebih sensitif dibanding YOLOv5s, meskipun presisinya sedikit lebih rendah.

Rata-rata F1-Score sebesar 0.741 memperlihatkan keseimbangan moderat antara *precision* dan *recall*. Dari sisi evaluasi berbasis IoU, rata-rata mAP@50 tercatat 0.772 (77,2%), yang menjadi nilai tertinggi di antara beberapa model ringan, sedangkan mAP@50–95 berada pada 0.672 (67,2%). Perbedaan antara mAP@50 dan mAP@50–95 menandakan bahwa meskipun YOLOv11n cukup baik pada kriteria deteksi longgar, akurasinya masih menurun ketika diuji pada kondisi lebih ketat. Secara keseluruhan, YOLOv11n unggul dalam sensitivitas (recall) dan akurasi pada IoU 0.5, tetapi cenderung menghasilkan *false positive* lebih banyak, sehingga masih memerlukan peningkatan dataset atau penyesuaian hyperparameter untuk menyeimbangkan precision dan recall.

Hasil validasi silang 5-fold menunjukkan bahwa ketiga model, yakni YOLOv5s, YOLOv8s, dan YOLOv11n, memiliki performa yang konsisten namun dengan keunggulan yang berbeda. YOLOv8s menempati peringkat teratas dengan

rata-rata *precision* 0,852, *recall* 0,703, dan F1-Score 0,770. Nilai mAP@50 sebesar 0,769 dan mAP@50–95 sebesar 0,697 mengindikasikan keseimbangan optimal antara ketepatan dan sensitivitas, serta kestabilan deteksi pada berbagai tingkat IoU. YOLOv5s, meskipun berbasis arsitektur lama, masih menunjukkan hasil yang kompetitif dengan rata-rata *precision* 0,772, *recall* 0,730, dan F1-Score 0,751. Nilai mAP@50 sebesar 0,737 dan mAP@50–95 sebesar 0,690 memperlihatkan bahwa YOLOv5s cukup stabil dan seimbang, terutama dalam menjaga sensitivitas deteksi. Sementara itu, YOLOv11n unggul dalam aspek *recall* dengan rata-rata 0,737 serta mAP@50 tertinggi sebesar 0,772. Akan tetapi, nilai *precision* yang lebih rendah (0,745) serta mAP@50–95 sebesar 0,672 menunjukkan bahwa model ini cenderung menghasilkan lebih banyak *false positive* dibandingkan YOLOv8s dan YOLOv5s.

Secara umum, hasil ini memperlihatkan bahwa YOLOv8s merupakan model paling andal dengan performa seimbang di seluruh metrik, YOLOv5s tetap relevan sebagai alternatif yang stabil dan efisien, sedangkan YOLOv11n lebih sensitif dalam mendeteksi kerusakan tetapi memerlukan perbaikan agar dapat mengurangi kesalahan deteksi. Temuan ini dapat menjadi dasar dalam memilih model sesuai kebutuhan implementasi: YOLOv8s untuk akurasi terbaik, YOLOv5s untuk stabilitas dan efisiensi, serta YOLOv11n untuk aplikasi yang membutuhkan sensitivitas tinggi.

4.7 Evaluasi Aplikasi Mobile

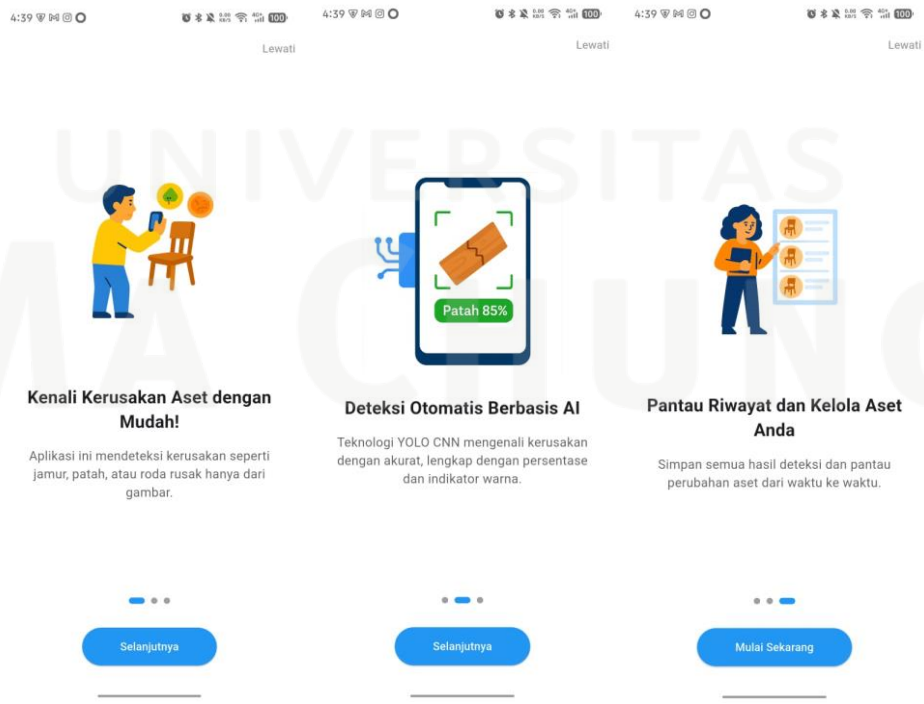
Evaluasi kinerja aplikasi bertujuan untuk memastikan bahwa sistem yang dikembangkan dapat berjalan sesuai spesifikasi, serta dapat digunakan oleh petugas Bagian Aset Universitas Ma Chung dengan lancar dan efisien. Evaluasi ini dilakukan menggunakan pendekatan black-box testing, yaitu dengan menguji aplikasi berdasarkan masukan (input) dan keluaran (output), tanpa memperhatikan kode internal dari sistem.

Pengujian difokuskan pada fitur-fitur inti yang dibutuhkan dalam proses pencatatan kerusakan aset, meliputi: pengambilan gambar melalui kamera, pemilihan gambar dari galeri, penyimpanan hasil deteksi, serta navigasi antarmuka. Aplikasi diuji secara langsung menggunakan perangkat Android (Xiaomi Redmi Note 12 Pro 4G) dalam kondisi lapangan nyata.

4.7.1 Pengujian Halaman Utama

Gambar 4. 27 Splash Screen

Saat aplikasi pertama kali dibuka, pengguna akan disambut oleh tampilan awal berupa splash screen yang menampilkan identitas institusi, yaitu *Universitas Ma Chung*. Halaman ini berfungsi sebagai pengenal aplikasi dan memberikan kesan profesional sebelum pengguna masuk ke fitur utama.



Gambar 4. 28 Onboarding Screen Aplikasi

Setelah itu, pengguna dibawa ke halaman onboarding pertama yang memperkenalkan fungsi utama aplikasi, yakni kemudahan dalam mendeteksi kerusakan aset seperti jamur, patah, maupun roda rusak hanya melalui gambar. Cukup dengan memotret menggunakan kamera atau memilih foto dari galeri, sistem akan langsung memprosesnya secara otomatis.

Pada halaman berikutnya, aplikasi memberikan penjelasan bahwa proses identifikasi kerusakan dilakukan secara otomatis dengan memanfaatkan teknologi YOLO.

Teknologi ini memungkinkan aplikasi mengenali jenis kerusakan dengan tingkat ketepatan yang tinggi, dilengkapi persentase keyakinan serta indikator warna untuk membantu pengguna memahami tingkat kerusakan secara visual. Halaman terakhir onboarding menjelaskan fitur penyimpanan riwayat deteksi. Setiap hasil analisis tersimpan otomatis dan dapat diakses kembali kapan saja, sehingga memudahkan pengguna dalam memantau perkembangan atau perubahan kondisi aset dari waktu ke waktu.

Dengan demikian, aplikasi mendukung pengelolaan aset secara lebih efisien dan berkelanjutan. Setelah rangkaian onboarding selesai, pengguna dapat langsung memulai penggunaan aplikasi dengan menekan tombol “Mulai Sekarang”.

Seluruh proses onboarding telah diuji melalui metode black-box testing. Hasil pengujian memperlihatkan bahwa semua komponen—mulai dari navigasi antarhalaman, tampilan ilustrasi, hingga tombol interaktif seperti “Selanjutnya” dan “Mulai Sekarang”—berjalan sesuai harapan. Hal ini menegaskan bahwa fitur onboarding sudah siap digunakan secara optimal oleh pengguna.

Gambar 4. 29 Home Screen

Selanjutnya, pengguna dapat memilih opsi "Ambil Foto" atau "Pilih dari Galeri" di halaman utama. Aplikasi kemudian mengaktifkan kamera atau membuka galeri, dan gambar yang dipilih akan ditampilkan di layar deteksi. Jika gambar berhasil dimuat dan dikirim ke server API Ultralytics, serta hasil deteksi muncul dalam bentuk bounding box beserta label klasifikasi, maka fungsi deteksi dianggap berjalan dengan baik.

Gambar 4. 30 Layar Konfigurasi API

Fitur Konfigurasi API dikembangkan untuk memberikan keleluasaan bagi

pengguna dalam mengatur serta memilih model deteksi yang akan digunakan aplikasi. Melalui menu ini, pengguna dapat menambahkan, menghapus, atau mengaktifkan model deteksi berbasis layanan cloud Ultralytics secara fleksibel tanpa harus melakukan proses *rebuild* aplikasi.

Pada halaman utama konfigurasi, ditampilkan daftar model yang telah tersimpan. Setiap konfigurasi berisi informasi berupa nama, Model ID, dan sebagian API key yang disembunyikan demi menjaga keamanan. Konfigurasi yang sedang aktif ditandai dengan bingkai hijau dan label "Sedang Digunakan". Pengguna dapat mengganti model dengan menekan tombol aktivasi pada entri yang diinginkan, atau menghapus konfigurasi dengan opsi khusus yang dilengkapi dialog konfirmasi agar tindakan tidak dilakukan secara keliru.

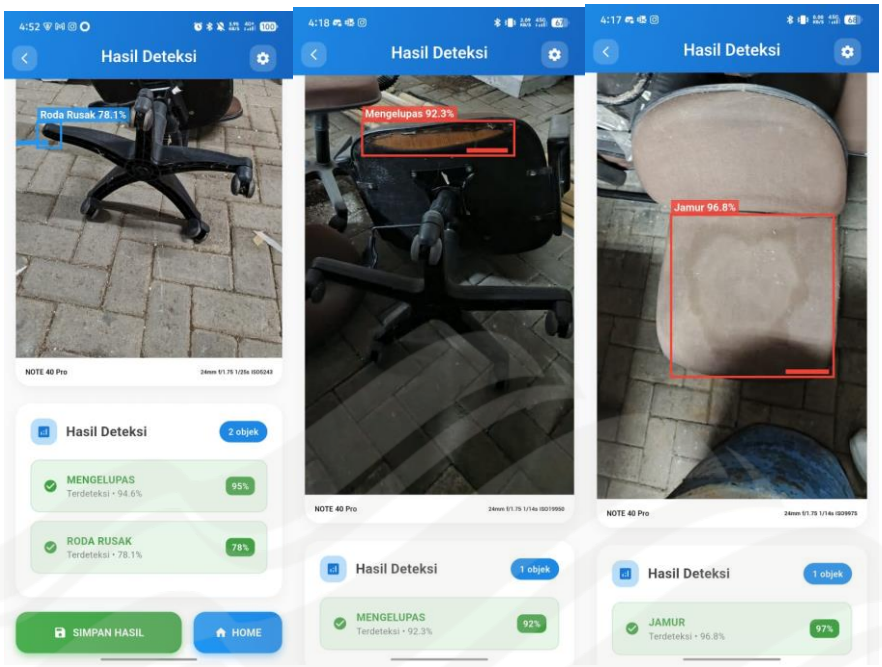
Untuk menambahkan konfigurasi baru, pengguna cukup menekan tombol tambah di bagian atas layar yang akan membuka halaman "Tambah Konfigurasi API". Pada halaman ini, pengguna perlu mengisi nama konfigurasi, API key yang valid, serta Model ID. Aplikasi juga menyediakan tips tambahan agar pengguna memastikan API key memiliki akses ke model yang sesuai sekaligus menyimpannya secara aman. Setelah formulir disimpan, konfigurasi baru langsung muncul pada daftar dan siap digunakan.

Gambar 4. 31 Layar Tutorial

Fitur Panduan Penggunaan pada aplikasi ini berfungsi sebagai alat bantu agar pengguna memahami langkah pengambilan gambar yang benar sekaligus cara membaca hasil deteksi kerusakan aset dengan tepat. Panduan ini terbagi menjadi dua bagian utama. Pada bagian pertama, dijelaskan prosedur pengambilan foto kursi dari empat sudut berbeda untuk memastikan cakupan visual yang menyeluruh. Pengguna diarahkan untuk memulai dengan memotret bagian depan kursi dengan pencahayaan yang cukup sehingga seluruh struktur terlihat jelas. Berikutnya, foto diambil dari sisi kanan atau kiri dengan posisi kamera sejajar dan latar belakang bersih agar bentuk samping kursi dapat terekam secara optimal. Panduan kemudian dilanjutkan dengan pengambilan gambar bagian belakang untuk mendeteksi kerusakan pada sandaran, serta foto terakhir dari arah atas secara tegak lurus yang menyoroti bagian bawah kursi, khususnya kaki dan roda, guna memeriksa kemungkinan kerusakan struktural.

Bagian kedua dari panduan menjelaskan cara menafsirkan hasil deteksi berdasarkan tingkat kepercayaan (confidence score) yang dihasilkan model. Sistem membagi hasil ke dalam tiga kategori warna: hijau (≥70%) yang menunjukkan deteksi sangat meyakinkan, kuning (30–69%) yang mengindikasikan kemungkinan kerusakan namun perlu konfirmasi ulang, dan merah (<30%) yang menandakan ketidakpastian tinggi sehingga pengguna disarankan melakukan pengambilan ulang foto. Untuk menunjang keberhasilan deteksi, panduan juga memberikan tips tambahan, seperti memastikan pencahayaan cukup terang, menghindari bayangan yang menutupi objek, membersihkan lensa kamera sebelum memotret, serta menjaga jarak dan sudut pandang agar kursi tampak jelas.

4.7.2 Pengujian Halaman Riwayat

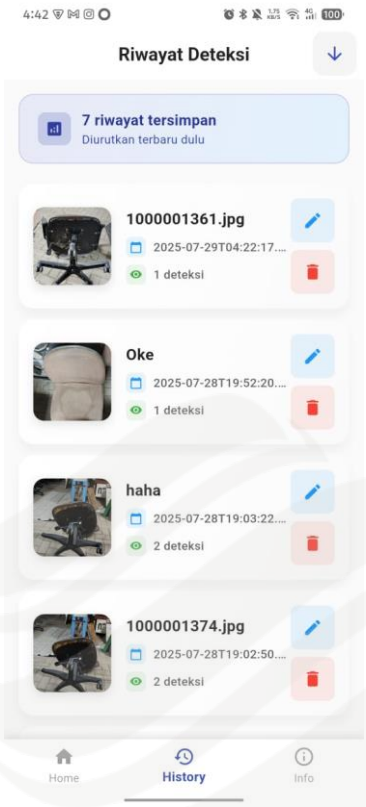


Gambar 4. 32 Layar Hasil

Gambar di atas memperlihatkan tampilan Halaman Hasil Deteksi pada aplikasi. Setelah pengguna mengunggah gambar kursi dan sistem menyelesaikan proses analisis, hasil deteksi ditampilkan dalam dua bagian utama.

Pada bagian atas, aplikasi menampilkan foto kursi asli yang telah dianotasi dengan bounding box berwarna. Pada contoh ini, sistem berhasil mendeteksi kerusakan dengan label “Roda Rusak” disertai tingkat keyakinan 78.1%, sehingga pengguna dapat langsung melihat lokasi kerusakan pada objek.Bagian bawah layar menampilkan daftar hasil deteksi dalam format teks yang rapi. Dalam contoh ini, terdapat dua objek terdeteksi, yaitu “Mengelupas” dengan tingkat kepercayaan 94.6% (ditampilkan sebagai 95%) dan “Roda Rusak” dengan 78.1% (78%). Setiap entri hasil dilengkapi ikon validasi dan indikator warna hijau yang menandakan bahwa hasil deteksi masuk kategori kepercayaan tinggi (≥70%).

Di bagian paling bawah terdapat dua tombol aksi utama, yaitu “Simpan Hasil” untuk menyimpan data deteksi ke dalam riwayat lokal, serta tombol “Home” untuk kembali ke halaman utama aplikasi. Dengan tampilan ini, pengguna tidak hanya memperoleh informasi visual melalui gambar yang dianotasi, tetapi juga ringkasan tekstual yang jelas dan mudah dipahami.



Gambar 4. 33 Layar Riwayat

Pengujian terhadap halaman Riwayat bertujuan memastikan bahwa seluruh fitur terkait penyimpanan dan pengelolaan data berjalan sesuai desain. Halaman ini menyajikan daftar riwayat deteksi aset yang sebelumnya telah disimpan oleh pengguna. Beberapa aspek yang diuji mencakup:

Beberapa aspek yang diuji pada halaman ini meliputi:

- Penampilan data riwayat: Sistem diharapkan dapat menampilkan daftar hasil deteksi berupa gambar pratinjau yang disertai informasi seperti nama file dan tanggal penyimpanan. Pengujian dilakukan untuk memastikan data muncul secara lengkap dan konsisten setiap kali halaman ini diakses.
- Fitur pengguliran (scroll): Jika jumlah riwayat melebihi tinggi layar, aplikasi harus dapat melakukan pengguliran vertikal. Hasil uji menunjukkan bahwa fitur ini berjalan dengan lancar.
- Fitur *rename* : Setiap hasil riwayat dilengkapi tombol yang digunakan untuk mengganti nama dari data tersebut supaya dapat diolah nanti.
- Fungsi hapus riwayat: Setiap hasil riwayat dilengkapi tombol hapus (ikon tempat sampah) untuk menghapus data secara individual. Pengujian

dilakukan dengan menekan tombol ini, dan sistem berhasil menghapus data baik dari tampilan aplikasi maupun dari basis data lokal (SQLite).

- Fitur pengurutan berdasarkan tanggal (Sort by Date): Aplikasi menyediakan opsi penyusunan data berdasarkan waktu penyimpanan, baik dari terbaru ke terlama maupun sebaliknya. Hasil pengujian menunjukkan sistem mampu menata urutan data sesuai preferensi pengguna.

Berdasarkan pengujian tersebut, seluruh fitur dalam halaman Riwayat berjalan stabil dan tidak ditemukan bug maupun kesalahan fungsional. Hal ini membuktikan bahwa fitur ini siap digunakan sebagai sarana pelacakan historis kondisi aset.

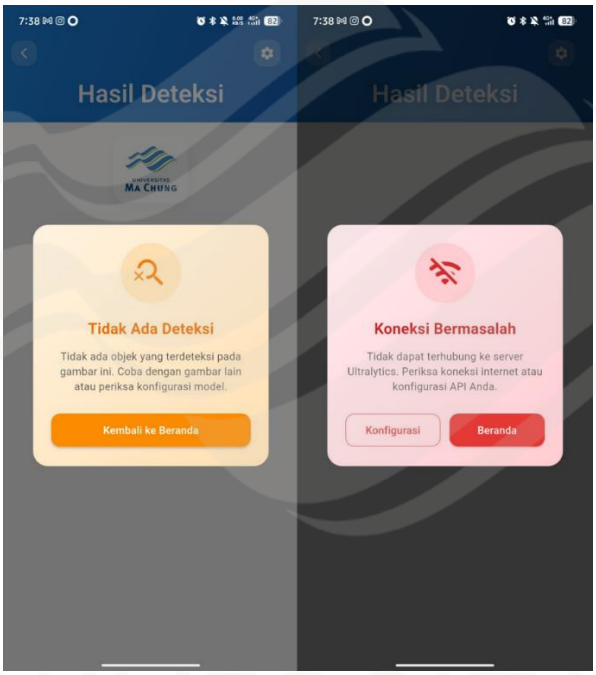
4.7.3 Pengujian Halaman Informasi

Gambar 4. 34 Layar Informasi

Sementara itu, halaman Informasi (*Info Screen*) merupakan bagian statis dari aplikasi yang berisi informasi tentang institusi pengembang serta kontak dari pihak yang bertanggung jawab, yaitu Unit Sistem Informasi dan Pusat Data (SIPUSDA) Universitas Ma Chung. Pengujian dilakukan untuk memastikan bahwa halaman ini: Dapat diakses melalui ikon navigasi bawah "Info", menampilkan seluruh informasi dengan lengkap dan rapi, memuat logo institusi, nama pengembang, lokasi unit, serta informasi kontak (email dan nomor telepon).

Antarmuka halaman ini dirancang dengan latar warna yang lembut dan teks dengan kontras tinggi untuk memastikan keterbacaan yang baik. Tidak terdapat elemen interaktif atau formulir, sehingga fokus sepenuhnya pada penyajian informasi yang rapi dan mudah diakses. Hasil pengujian menunjukkan bahwa semua elemen tampil sesuai harapan dan halaman telah berfungsi sebagaimana mestinya dalam menyediakan informasi formal kepada pengguna aplikasi.

4.7.4 Pengujian Penangangan Error



Gambar 4. 35 Penanganan Error

Aplikasi ini telah dilengkapi dengan sistem penanganan error (*error handling*) yang dirancang untuk memberikan umpan balik yang jelas dan informatif kepada pengguna apabila terjadi kegagalan selama proses deteksi. Berdasarkan gambar di atas, terdapat dua jenis skenario error yang telah diantisipasi dan ditampilkan melalui antarmuka pengguna.

Pertama, pada kasus "Tidak Ada Deteksi", sistem menampilkan pesan peringatan berwarna oranye apabila tidak ditemukan objek kerusakan pada gambar yang dikirimkan. Hal ini dapat terjadi jika gambar yang digunakan tidak mengandung objek relevan sesuai pelatihan model, pencahayaan kurang, atau sudut pengambilan gambar tidak sesuai. Aplikasi memberikan saran kepada pengguna untuk mencoba dengan gambar lain atau memeriksa kembali konfigurasi model

yang digunakan. Pesan ini ditampilkan secara elegan dengan tombol tindakan langsung untuk kembali ke halaman beranda. Kedua, pada skenario "Koneksi Bermasalah", sistem menampilkan pesan kesalahan berwarna merah apabila aplikasi gagal melakukan koneksi ke server Ultralytics. Hal ini bisa disebabkan oleh koneksi internet yang tidak stabil, atau kesalahan pada pengaturan API key dan model ID. Dalam kondisi ini, pengguna diberikan dua opsi tindakan yang jelas, yaitu membuka halaman konfigurasi untuk memperbaiki pengaturan, atau kembali ke beranda. Pendekatan ini tidak hanya menjaga kejelasan navigasi, tetapi juga memperkecil risiko frustrasi pengguna karena error yang tidak dijelaskan.

Seluruh skenario penanganan error ini telah diuji menggunakan metode black-box testing dan dinyatakan berfungsi sesuai dengan harapan. Dengan demikian, aplikasi memiliki ketahanan yang baik terhadap kondisi gagal deteksi maupun gangguan koneksi, serta tetap menjaga pengalaman pengguna secara optimal melalui antarmuka yang intuitif dan informatif.

Tabel 4. 5 Hasil Pengujian Fungsional Aplikasi (Black Box Testing)

No	Fitur	Input	Hasil yang Diharapkan	Hasil Aktual
1	Kamera	Pengguna menekan tombol kamera dan mengambil foto aset	Aplikasi berhasil mengakses kamera dan menangkap foto. Setelah pengambilan gambar, model deteksi dijalankan dan menampilkan bounding box di area kerusakan dengan label kelas dan confidence. Hasil deteksi tampil di layar dalam waktu wajar (realtime).	Berhasil: Kamera terbuka, foto dapat diambil. Model memproses citra dan dalam ~2 detik menampilkan hasil deteksi kerusakan (contoh: label “Patah” dan “Jamur” muncul sesuai objek pada Gambar 4.2). Tampilan sesuai format yang diharapkan.

Tabel 4. 5 Hasil Pengujian Fungsional Aplikasi (Black Box Testing) (Lanjutan)

2	Pengguna menekan tombol *Galleri*, aplikasi membuka menu galeri dan memilih satu foto aset dari penyimpanan	Aplikasi berhasil mengakses galeri dan pengguna dapat memilih foto. Setelah foto dipilih, aplikasi menjalankan deteksi kerusakan pada gambar tersebut dan menampilkan hasilnya (bounding box + label) di layar.	Berhasil: Foto dari galeri berhasil dipilih dan dimuat ke aplikasi. Model mendeteksi kerusakan pada foto tersebut (misal terdeteksi "Mengelupas") dan menampilkan kotak deteksi di gambar. Waktu pemrosesan sekitar 1–2 detik, output sesuai harapan.
3	Setelah mendapatkan hasil deteksi, pengguna menekan tombol Save untuk menyimpan gambar hasil deteksi (dengan bounding box) ke database riwayat aplikasi.	Aplikasi menyimpan gambar hasil deteksi dengan overlay bounding box ke memori perangkat atau daftar riwayat tanpa kesalahan. Pengguna mendapat notifikasi atau konfirmasi bahwa penyimpanan berhasil.	Berhasil: Gambar hasil deteksi tersimpan ke galeri perangkat (folder aplikasi) dengan benar. Aplikasi menampilkan pesan "Hasil deteksi disimpan". Ketika dicek, file gambar tersimpan berisi bounding box sesuai tampilan. Tidak ada error yang muncul.

Tabel 4. 5 Hasil Pengujian Fungsional Aplikasi (Black Box Testing) (Lanjutan)

4	Pengguna berpindah antar halaman aplikasi, misalnya dari halaman utama ke halaman deteksi kerusakan, ke halaman riwayat, lalu kembali ke halaman utama.	Setiap perpindahan halaman berlangsung lancar tanpa error. Tampilan setiap menu sesuai desain (halaman deteksi menampilkan feed kamera atau area untuk gambar, halaman riwayat menampilkan list hasil tersimpan, dll.). Tombol Back atau navigasi bekerja semestinya.	Berhasil: Navigasi antar menu berjalan mulus. Halaman kamera dapat ditampilkan dari menu utama, halaman riwayat menunjukkan daftar gambar tersimpan. Tombol kembali mengembalikan ke menu utama tanpa kendala. Antarmuka konsisten dan responsif selama navigasi.

Pada Tabel 4.1 di atas dapat dilihat bahwa seluruh skenario uji fungsional utama aplikasi dinyatakan lulus (Pass). Fitur kamera dan galeri berfungsi sesuai harapan, di mana aplikasi berhasil mengakses perangkat untuk mengambil foto maupun memilih file gambar, lalu memprosesnya dengan model deteksi. Hasil

deteksi muncul dengan benar di layar, lengkap dengan penandaan kerusakan sebagaimana telah dicontohkan pada Gambar 4.2. Waktu respons aplikasi tergolong cepat – rata-rata sekitar 1-3 detik untuk menampilkan hasil deteksi setelah gambar dipasok. Hal ini menunjukkan bahwa integrasi model YOLOv8s ke dalam aplikasi mobile cukup efisien dan mampu memenuhi kebutuhan deteksi kerusakan secara realtime dari perspektif pengguna.

Fitur penyimpanan hasil deteksi juga berfungsi dengan baik. Setiap kali pengguna menekan tombol simpan, gambar dengan overlay hasil deteksi berhasil disimpan, baik ke memori lokal perangkat maupun ke menu Riwayat dalam aplikasi (sesuai implementasi). Tidak ditemukan bug seperti gagal simpan atau aplikasi terhenti tiba-tiba saat penyimpanan, sehingga dapat disimpulkan mekanisme penyimpanan telah diimplementasikan dengan handal. Begitu pula, navigasi antar menu aplikasi teruji lancar; pengguna dapat berpindah dari satu fitur ke fitur lain (misalnya dari halaman utama ke halaman kamera, lalu ke halaman riwayat, dan kembali lagi) tanpa mengalami crash ataupun tampilan yang kacau. Transisi antarmuka berlangsung mulus dan setiap halaman menampilkan informasi yang semestinya.

Selama pengujian lapangan, terdapat beberapa temuan penting terkait penggunaan aplikasi dan performa deteksi di kondisi nyata. Pertama, posisi dan sudut pengambilan gambar terbukti mempengaruhi hasil deteksi. Ketika pengguna mengambil foto aset dari jarak terlalu jauh atau sudut yang terlalu miring, akurasi deteksi dapat menurun. Misalnya, saat kamera diarahkan tidak lurus terhadap objek (sudut tajam) atau fokus kerusakan berada terlalu kecil di kejauhan, model kadang gagal mengenali kerusakan tertentu yang seharusnya terdeteksi (false negative), atau sebaliknya mendeteksi artefak cahaya/corak sebagai kerusakan (false positive). Sebaliknya, pengambilan gambar yang tepat – dengan jarak cukup dekat sehingga kerusakan tampak jelas dan sudut pandang tegak lurus – menghasilkan deteksi yang jauh lebih akurat dan konsisten. Oleh karena itu, disarankan kepada pengguna untuk mengambil foto sedekat mungkin dengan objek kerusakan dan dari sudut pandang yang memperlihatkan kerusakan secara frontal untuk memperoleh hasil deteksi terbaik. Kedua, faktor pencahayaan juga diamati berpengaruh: dalam kondisi pencahayaan sangat redup atau berbayang tajam, performa model sedikit

menurun (beberapa kerusakan kecil tidak terdeteksi). Namun, pada pencahayaan normal (siang hari atau pencahayaan ruangan yang memadai), model tetap mampu bekerja dengan baik. Aplikasi sendiri telah menyediakan antarmuka kamera yang memanfaatkan kamera smartphone, sehingga kualitas gambar sangat bergantung pada kemampuan kamera dan kondisi lingkungan. Dalam hal kinerja, aplikasi teruji stabil tanpa lag berarti pada perangkat uji (contoh: smartphone Note 40 Pro dengan spesifikasi menengah). Pemrosesan model YOLOv8s yang relatif ringan memastikan bahwa waktu inferensi singkat dan tidak membebani kinerja aplikasi secara berlebihan.

Secara keseluruhan, hasil pengujian fungsional dan uji lapangan menunjukkan bahwa aplikasi mobile pendeteksi kerusakan aset yang dikembangkan telah memenuhi kriteria keberhasilan baik dari sisi model maupun implementasi aplikasi. Model YOLOv8s yang tertanam mampu mendeteksi kerusakan dengan cukup akurat dan cepat, sementara aplikasi Flutter menyediakan antarmuka yang user-friendly dan reliabel dalam mengoperasikan fungsi-fungsi utama (kamera, galeri, simpan, navigasi). Temuan-temuan mengenai pengaruh jarak, sudut, dan pencahayaan memberikan masukan berharga untuk penyempurnaan lebih lanjut, misalnya dengan menambahkan panduan penggunaan kepada user atau pengembangan model lanjutan yang lebih robust terhadap variansi kondisi lingkungan. Dengan demikian, bab hasil dan pembahasan ini mengonfirmasi bahwa tujuan proyek untuk mengembangkan aplikasi deteksi kerusakan aset berbasis YOLOv8s dan Flutter telah tercapai.

Tabel 4. 6 Hasil Pengujian Fungsional Aplikasi

No	Parameter Uji	Skenario	Hasil yang Diharapkan		Hasil Aktual
1	Waktu Startup	Mengukur waktu dari aplikasi dibuka hingga halaman utama tampil.	Waktu startup ≤ 5 detik	Rata-rata 3,2 detik pada perangkat uji (Xiaomi Redmi Note 12 Pro 4G).	Waktu Startup

Tabel 4. 6 Hasil Pengujian Fungsional Aplikasi (Lanjutan)

2	Penggunaan Memori	Memantau penggunaan memori saat aplikasi idle di halaman utama.	Penggunaan memori < 200 MB pada perangkat kelas menengah.	Rata-rata 148 MB saat idle.	Penggunaan Memori
3	Responsivitas UI	Menguji kelancaran navigasi antar halaman dan *scrolling*.	Navigasi lancar tanpa *lag* atau *frame drop* signifikan.	FPS rata-rata 58–60 pada perangkat uji.	Responsivitas UI
4	Waktu Respons API Deteksi	Mengukur waktu dari gambar dikirim hingga hasil deteksi muncul.	Waktu respons ≤ 5 detik pada koneksi internet stabil.	Rata-rata 4,1 detik pada jaringan Wi-Fi 30 Mbps.	Waktu Respons API Deteksi

Tabel 4.6 menunjukkan bahwa aplikasi memenuhi semua kriteria fungsional yang diuji. Waktu startup rata-rata 3,2 detik masih di bawah ambang 5 detik, konsumsi memori saat idle hanya 148 MB sehingga efisien pada perangkat kelas menengah, dan navigasi antarmuka berjalan mulus dengan frame rate stabil 58–60 FPS. Selain itu, waktu respons API deteksi rata-rata 4,1 detik pada jaringan Wi-Fi 30 Mbps membuktikan aplikasi mampu menampilkan hasil secara *real-time*. Secara keseluruhan, hasil pengujian ini menegaskan bahwa aplikasi bekerja responsif, hemat sumber daya, dan layak diimplementasikan untuk penggunaan di lapangan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa pengembangan model deteksi kerusakan aset berbasis *Convolutional Neural Network* (CNN) dan implementasinya ke dalam aplikasi mobile telah berhasil memenuhi tujuan yang ditetapkan. Model YOLOv8s (small) yang dilatih untuk mendeteksi empat jenis kerusakan kursi—yaitu jamur, cat mengelupas, patah, dan roda rusak—menunjukkan performa yang baik. Berdasarkan hasil evaluasi 5-Fold Cross Validation, model memperoleh nilai rata-rata mAP@50 sebesar 76,9% dan mAP@50–95 sebesar 69,7%. Selain itu, metrik evaluasi juga menunjukkan precision yang tinggi, yaitu mencapai 85,2%, serta recall sebesar 70,3%, yang menandakan sensitivitas model dalam mengenali keberadaan kerusakan. Meski demikian, performa pada kelas Jamur dan Mengelupas masih relatif sedang karena adanya kecenderungan menghasilkan *false positive*, terutama ketika citra background memiliki pola visual yang menyerupai jamur atau cat terkelupas. Hal ini menunjukkan bahwa model masih rentan terhadap *over-detection* pada kasus tertentu, namun secara keseluruhan tetap andal dalam mengenali mayoritas kerusakan yang ditargetkan.

Secara kualitatif, visualisasi hasil deteksi turut mendukung capaian metrik yang diperoleh. Model mampu menampilkan *bounding box* dengan label kelas dan tingkat kepercayaan di atas objek kerusakan pada gambar uji. Dalam beberapa kasus, model berhasil mendeteksi lebih dari satu kerusakan pada satu citra secara bersamaan, misalnya mendeteksi kursi yang mengalami patah sekaligus berjamur dengan anotasi yang sesuai kondisi nyata. Visualisasi ini membuktikan konsistensi model dalam mengidentifikasi jenis dan lokasi kerusakan secara akurat. Implementasi visualisasi secara *real-time* pada aplikasi—dengan bounding box dan label yang langsung muncul pada gambar—sangat membantu pengguna dalam memahami hasil deteksi secara cepat dan intuitif. Dengan demikian, sistem yang dibangun berhasil mencapai tujuan untuk mendukung deteksi kerusakan otomatis baik dari sisi metrik evaluasi maupun hasil visual.

Dari sisi implementasi aplikasi, hasil pengujian fungsional menggunakan metode *black-box testing* memastikan bahwa seluruh fitur utama pada aplikasi mobile berbasis Flutter berjalan sesuai dengan skenario. Fitur kamera dapat digunakan untuk mengambil gambar dan mengirimkannya ke sistem deteksi tanpa kendala, sedangkan fitur galeri memungkinkan pengguna memilih gambar dari penyimpanan perangkat untuk dianalisis. Setelah proses deteksi selesai, hasil ditampilkan dalam format visual maupun teks lengkap dengan tingkat confidence. Fitur simpan hasil juga bekerja dengan baik, di mana setiap deteksi berhasil tersimpan secara lokal dan ditampilkan dalam halaman Riwayat. Halaman ini menampilkan daftar gambar yang sudah dianalisis beserta informasi tambahan, sehingga pengguna dapat melakukan monitoring kondisi aset secara berkala. Navigasi antarhalaman (Home, Deteksi, Riwayat, dan Info) berlangsung lancar tanpa hambatan, dan aplikasi terbukti stabil saat diuji pada perangkat Android tanpa mengalami *crash* ataupun *lag*. Waktu rata-rata inferensi yang hanya memerlukan 1–4 detik per gambar juga mendukung penggunaan aplikasi secara *real-time* di lapangan.

Dengan demikian, penelitian ini berhasil menjawab rumusan masalah yang diajukan, yaitu: mengembangkan model deteksi kerusakan aset berbasis CNN yang mampu mengenali empat jenis kerusakan kursi dengan akurasi yang memadai, mengintegrasikan model tersebut ke dalam aplikasi mobile berbasis Flutter yang fungsional dan ramah pengguna, serta mengevaluasi kinerja sistem yang terbukti stabil, cepat, dan konsisten baik dari sisi performa model maupun implementasi aplikasi. Walaupun masih terdapat ruang untuk peningkatan, khususnya dalam mengurangi kesalahan deteksi pada kelas dengan pola visual yang halus, sistem yang dibangun telah terbukti layak dan efektif sebagai solusi otomatisasi deteksi kerusakan aset.

5.2 Saran

Berdasarkan temuan dan keterbatasan dalam penelitian ini, terdapat beberapa saran untuk pengembangan lebih lanjut agar sistem deteksi kerusakan aset dapat berfungsi dengan lebih optimal:

Pertama, perlu dilakukan peningkatan pada model deteksi dengan memperkaya dataset dan menyempurnakan penanganan kelas background untuk

mengurangi *false positive*. Dataset pelatihan sebaiknya diperluas dengan lebih banyak variasi citra, mencakup beragam kondisi aset, lingkungan, dan sudut pengambilan gambar. Termasuk pula penambahan contoh-contoh citra tanpa kerusakan (negative samples) dari berbagai latar belakang, agar model dapat belajar membedakan pola background yang normal versus kerusakan sebenarnya. Strategi ini akan membantu model menjadi lebih robust terhadap lingkungan nyata yang beragam dan menurunkan kecenderungan model mendeteksi kerusakan palsu pada objek yang sebenarnya utuh. Selain itu, fine-tuning lanjutan dapat difokuskan untuk kelas-kelas yang presisinya masih rendah (misalnya kerusakan *Jamur* dan *Mengelupas*) serta untuk meningkatkan kemampuan model mengenali background (area tanpa kerusakan) dengan lebih baik. Penyesuaian parameter deteksi seperti threshold confidence juga dapat dipertimbangkan, sehingga model hanya menandai kerusakan jika memiliki keyakinan tinggi. Melalui pengembangan model lanjutan ini, diharapkan precision model dapat ditingkatkan tanpa mengorbankan recall, sehingga hasil deteksi lebih dapat diandalkan dengan minim kesalahan identifikasi.

Kedua, dari sisi arsitektur sistem, disarankan untuk memisahkan komponen model AI dari aplikasi mobile dengan menerapkan arsitektur berbasis layanan API terpisah. Dengan pola ini, aplikasi Flutter hanya berfungsi sebagai antarmuka, sementara proses inferensi model dijalankan di server melalui endpoint API. Keunggulan dari pendekatan ini adalah pengembang dapat melakukan pembaruan model secara fleksibel, baik dengan melatih ulang bobot terbaru maupun mengganti ke versi YOLO yang lebih mutakhir, tanpa memerlukan proses *rebuild* atau pembaruan aplikasi di perangkat pengguna. Selain itu, pemisahan model sebagai layanan akan mengurangi beban komputasi pada aplikasi, menjaga ukuran file tetap ringan, serta mendukung skalabilitas apabila jumlah pengguna bertambah. Dengan demikian, sistem menjadi lebih mudah dipelihara, diperbarui, dan diperluas sesuai kebutuhan di masa depan.

DAFTAR PUSTAKA

Ammann, P., & Offutt, J. (2016). Introduction to software testing. Cambridge University Press.

Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys, 4*, 40–79.

Beizer, B. (1995). Black-box testing: Techniques for functional testing of software and systems. Wiley.

Bisong, E. (2019). Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners. Apress.

Bright Security. (2023). Black-box testing: Types, techniques, pros and cons. Retrieved from https://www.brightsec.com/blog/black-box-testing-types-techniques-pros-and-cons/

Brownlee, J. (2020). What is a Confusion Matrix in Machine Learning? Machine Learning Mastery.

Casas, E., Ramos, L., Bendek, E., & Rivas-Echeverria, F. (2024). YOLOv5 vs. YOLOv8: Performance benchmarking in wildfire and smoke detection scenarios. Journal of Image and Graphics, 12(2), 127–136.

Dwibedi, D., Misra, I., & Hebert, M. (2017). Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection. arXiv.

Flutter. (2022). Flutter Documentation: Building Cross-Platform Apps. https://flutter.dev/docs

GeeksforGeeks. (n.d.). Software engineering - Black box testing.

Google Developers. (2023). Android Studio: The Official IDE for Android.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

Hicks, S. A., Strümke, I., Thambawita, V., Hammou, M., Riegler, M. A., Halvorsen, P., & Parasa, S. (2022). On evaluation metrics for medical applications of artificial intelligence. Scientific Reports, 12(1), 5979.

Jorgensen, P. C. (2013). Software testing: A craftsman's approach (4th ed.). Auerbach Publications.

Kaner, C., Falk, J., & Nguyen, H. Q. (1999). Testing computer software (2nd

ed.). Wiley.

Kattenborn, T., Leitloff, J., Schiefer, F., & Hinz, S. (2021). Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. ISPRS Journal of Photogrammetry and Remote Sensing, 173, 24–49.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1137–1143.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM, 60(6), 84-90.

Kundu, R. (2023). YOLO: Algorithm for object detection explained [+Examples]. V7 Labs.

Lee, S., & Park, J. (2021). Mobile-Based Asset Management System Using Real-Time Object Detection. Journal of Information Systems, 15(3), 45-59.

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone. arXiv. https://arxiv.org/abs/1801.09454

Myers, G. J., Sandler, C., & Badgett, T. (2011). The art of software testing (3rd ed.). Wiley.

Patton, R. (2005). Software testing (2nd ed.). Sams Publishing.

Pressman, R. S., & Maxim, B. R. (2020). Software engineering: A practitioner's approach (9th ed.). McGraw-Hill.

Redmon, J. (2016). YOLO: Real-time object detection. Darknet.

Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779–788).

Roboflow. (2023). Roboflow Annotate: Label Images Faster Than Ever.

Silva, A. S., Oliveira, W. S., Braga, P. A. V. F., Fernandes, L. L. A., & Costa, D. B. (2025). Computer Vision-based YOLO11 for automated damage assessment of public building roofs supporting maintenance management. *Proceedings of the*

CIB World Building Congress (WBC 2025).

Smith, J., et al. (2020). Automated Asset Tracking in Educational Institutions: Challenges and Solutions. International Journal of Advanced Computer Science, 10(2), 112-125.

Sommerville, I. (2016). Software engineering (10th ed.). Pearson.

Telaumbanua, F. (2024). WEB PENGHITUNG CACAH TBS MENGGUNAKAN ALGORITMA YOLOv8 (Doctoral dissertation, Institut Pertanian Stiper Yogyakarta).

TestDevLab. (2023). Types of black-box testing techniques. Retrieved from https://www.testdevlab.com/blog/types-of-black-box-testing-techniques/

Tian, Y., Pei, K., Jana, S., & Ray, B. (2018). DeepTest: Automated testing of deep-neural-network-driven autonomous cars. IEEE/ACM 40th International Conference on Software Engineering (ICSE).

Ultralytics. (2023). Ultralytics Yolov8 Documentation. (akses: docs.ultralytics.com)

Ultralytics. (2024). Performance Metrics Deep Dive. Ultralytics YOLO Documentation.

Ultralytics. (2024). YOLO model comparison: YOLOv8 vs previous. Ultralytics Blog.

Ultralytics. (2024). YOLOv5 vs. YOLOv8: A detailed comparison. Ultralytics YOLO Docs.

Ultralytics. (2024). YOLOv5, YOLOv8, and YOLOv10: The go-to detectors for real-time vision. arXiv preprint arXiv:2407.02988.

Van Rossum, G. and Drake, F.L. (2006) Python Reference Manual.

Whittaker, J. A., Arbon, J., & Carollo, J. (2012). How Google tests software. Addison-Wesley.

Zhang, Y., et al. (2019). Furniture Damage Detection Using Deep Learning. IEEE Access, 7, 123456–123465.