PROYEK AKHIR - VI231733

RANCANG BANGUN SISTEM ALAT *HANDHELD* UNTUK KLASIFIKASI BIJI JAGUNG MENGGUNAKAN *IMAGE PROCESSING* DENGAN METODE YOLOV8-PSO

MUHAMMAD RAFI ZIDAN HILMI

NRP 2042221125

Dosen Pembimbing 1

Ir. Sefi Novendra Patrialova, S.Si., M.T.

NPP 1991201712053

Dosen Pembimbing 2

Akhmad Ibnu Hija, S.T., M.T.

NPP 1996202311049

Program Studi Sarjana Terapan Teknologi Rekayasa Instrumentasi

Departemen Teknik Instrumentasi

Fakultas Vokasi

Institut Teknologi Sepuluh Nopember

Surabaya

2026

PROYEK AKHIR – VI231733

RANCANG BANGUN SISTEM ALAT *HANDHELD* UNTUK KLASIFIKASI BIJI JAGUNG MENGGUNAKAN *IMAGE PROCESSING* DENGAN METODE YOLOV8-PSO

MUHAMMAD RAFI ZIDAN HILMI

NRP 2042221125

Dosen Pembimbing 1

Ir. Sefi Novendra Patrialova, S.Si., M.T.

NPP 1991201712053

Dosen Pembimbing 2

Akhmad Ibnu Hija, S.T., M.T.

NPP 1996202311049

Program Studi Sarjana Terapan Teknologi Rekayasa Instrumentasi

Departemen Teknik Instrumentasi

Fakultas Vokasi

Institut Teknologi Sepuluh Nopember

Surabaya

2026

FINAL PROJECT – VI231733

DESIGN AND DEVELOPMENT OF A HANDHELD DEVICE SYSTEM FOR CORN KERNEL CLASSIFICATION USING IMAGE PROCESSING WITH THE YOLOV8-PSO METHOD

MUHAMMAD RAFI ZIDAN HILMI

NRP 2042221125

Supervisor 1

Ir. Sefi Novendra Patrialova, S.Si., M.T.

NPP 1991201712053

Supervisor 2

Akhmad Ibnu Hija, S.T., M.T.

NPP 1996202311049

Study Program Applied Bachelor Program of Instrumentation Engineering

Department of Instrumentation Engineering

Faculty of Vocational Studies

Institut Teknologi Sepuluh Nopember

Surabaya

2026

LEMBAR PENGESAHAN PROYEK AKHIR

RANCANG BANGUN SISTEM ALAT *HANDHELD* UNTUK KLASIFIKASI BIJI JAGUNG MENGGUNAKAN *IMAGE PROCESSING* DENGAN METODE YOLOV8-PSO

Oleh:

Muhammad Rafi Zidan Hilmi

NRP.2042221125

Surabaya, 23 Juli 2026

Menyetujui,

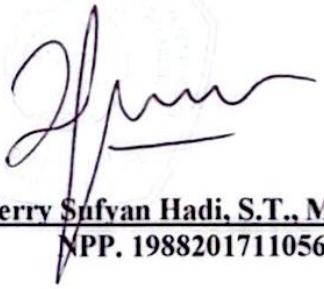
Menyetujui,
Dosen Pembimbing 1

Ir. Sefi Novendra Patrialova, S.Si., M.T.
NPP. 199120172053

Menyetujui,
Dosen Pembimbing 2

Akhmad Ibnu Hija, S.T., M.T.
NPP. 1996202311049

Mengetahui,
Kepala Departemen
Teknik Instrumentasi FV-ITS



Ir. Herry Sufyan Hadi, S.T., M.T., Ph.D
NPP. 1988201711056

Halaman ini sengaja dikosongkan.

APPROVAL SHEET

DESIGN AND DEVELOPMENT OF A HANDHELD DEVICE SYSTEM FOR CORN KERNEL CLASSIFICATION USING IMAGE PROCESSING WITH THE YOLOV8+PSO METHOD

By:

Muhammad Rafi Zidan Hilmi

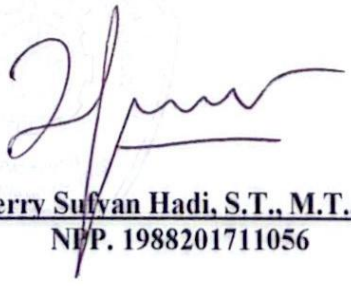
NRP.2042221125

Surabaya, 23 July 2026

Acknowledge,

Advisor 1	Advisor 2
	
Ir. Sefi Novendra Patrialova, S.Si., M.T.	**Akhmad Ibnu Hija, S.T., M.T.**
NPP. 199120172053	**NPP. 1996202311049**

Acknowledged,
Head of Department
Instrumentation Engineering FV-ITS



Ir. Herry Sufyan Hadi, S.T., M.T., Ph.D
NPP. 198820171056

Halaman ini sengaja dikosongkan.

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM ALAT *HANDHELD* UNTUK KLASIFIKASI BIJI JAGUNG MENGGUNAKAN *IMAGE PROCESSING* DENGAN METODE YOLOV8-PSO

PROYEK AKHIR

Diajukan untuk memenuhi salah satu syarat

memperoleh gelar Sarjana Terapan Teknik pada

Program Studi Sarjana Terapan Teknologi Rekayasa Instrumentasi

Departemen Teknik Instrumentasi

Fakultas Vokasi

Institut Teknologi Sepuluh Nopember

Oleh : **MUHAMMAD RAFI ZIDAN HILMI**

NRP. **2042221125**

Disetujui oleh Tim Penguji Proyek Akhir :

1. Ir. Sefi Novendra Patrialova, S.Si., M.T. — Pembimbing
2. Akhmad Ibnu Hija, S.T., M.T. — Ko-pembimbing
3. Dr. Ir. Arief Abdurrakhman, S.T., M.T. — Penguji
4. Ahmad Radhy, S.Si., M.Si. — Penguji

SURABAYA

Juli, 2026

Halaman ini sengaja dikosongkan.

APPROVAL SHEET

DESIGN AND DEVELOPMENT OF A HANDHELD DEVICE FOR CORN KERNEL CLASSIFICATION USING IMAGE PROCESSING WITH THE YOLOV8-PSO METHOD

FINAL PROJECT

Submitted to fulfil one of the requirements

For obtaining a Bachelor of Applied Engineering degree at

Applied Bachelor of Instrumentation Engineering Technology

Departement of Instrumentation Engineering

Faculty of Vocation

Institut Teknologi Sepuluh Nopember

By: **MUHAMMAD RAFI ZIDAN HILMI**

NRP. **2042221125**

Approved by Final Project Examiner Team :

1. Ir. Sefi Novendra Patrialova, S.Si., M.T. — *Advisor*
2. Akhmad Ibnu Hija, S.T., M.T. — *Co-Advisor*
3. Dr. Ir. Arief Abdurrakhman, S.T., MT. — *Examiner*
4. Ahmad Radhy, S.Si., M.Si. — *Examiner*

SURABAYA

July, 2026

Halaman ini sengaja dikosongkan.

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP	:	Muhammad Rafi Zidan Hilmi / 2042221125
Program studi	:	Teknologi Rekayasa Instrumentasi
Dosen Pembimbing / NIP	:	Ir. Sefi Novendra Patrialova, S.Si., M.T. / 199120172053
		Akhmad Ibnu Hija, S.T., M.T. / 1996202311049

dengan ini menyatakan bahwa Proyek Akhir dengan judul "**RANCANG BANGUN SISTEM ALAT *HANDHELD* UNTUK KLASIFIKASI BIJI JAGUNG MENGGUNAKAN *IMAGE PROCESSING* DENGAN METODE YOLOv8-PSO**" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 23 Juli 2026

Mahasiswa,

Muhammad Rafi Zidan Hilmi
NRP. 2042221125

Mengetahui,

Dosen Pembimbing 1	Dosen Pembimbing 2
Ir. Sefi Novendra Patrialova, S.Si., M.T.	Akhmad Ibnu Hija, S.T., M.T.
NPP. 199120172053	NPP. 1996202311049

Halaman ini sengaja dikosongkan.

STATEMENT OF ORIGINALITY

The undersigned below:

Student Name / NRP	:	Muhammad Rafi Zidan Hilmi / 2042221125
Study Programme	:	Instrumentation e
Advisor / NPP	:	Ir. Sefi Novendra Patrialova, S.Si., M.T. / 199120172053
		Akhmad Ibnu Hija, S.T., M.T. / 1996202311049

I hereby declare that the Final Project entitled **'DESIGN AND DEVELOPMENT OF A HANDHELD DEVICE FOR CORN KERNEL CLASSIFICATION USING IMAGE PROCESSING WITH THE YOLOv8-PSO METHOD'** is my own work, is original, and has been written in accordance with scientific writing standards.

Should any inconsistencies with this statement be found in the future, I am willing to accept sanctions in accordance with the applicable regulations at the Sepuluh Nopember Institute of Technology.

Surabaya, 23 July 2026

Student,

Muhammad Rafi Zidan Hilmi
NRP. 2042221125

Acknowledge,

Advisor 1	Advisor 2
Ir. Sefi Novendra Patrialova, S.Si., M.T.	Akhmad Ibnu Hija, S.T., M.T.
NPP. 199120172053	NPP. 1996202311049

Halaman ini sengaja dikosongkan.

ABSTRAK

RANCANG BANGUN SISTEM ALAT *HANDHELD* UNTUK KLASIFIKASI BIJI JAGUNG MENGGUNAKAN *IMAGE PROCESSING* DENGAN METODE YOLOV8-PSO

Nama Mahasiswa / NRP	**:**	**Muhammad Rafi Zidan Hilmi / 2042221125**
Departemen	**:**	**Teknik Instrumentasi**
Dosen Pembimbing 1	**:**	**Ir. Sefi Novendra Patrialova, S.Si., M.T.**
Dosen Pembimbing 2	**:**	**Akhmad Ibnu Hija, S.T., M.T.**

Abstrak

Klasifikasi biji jagung secara manual cenderung memakan waktu dan tidak selalu konsisten. Penelitian ini merancang Alat *Handheld* berbasis *image processing* untuk mengklasifikasikan biji jagung menjadi dua kelas, yaitu *Good* dan *Bad*, menggunakan YOLOv8m yang dioptimasi *Particle Swarm Optimization* (PSO) pada *hyperparameter*. Sistem dilengkapi kamera, pencahayaan *LED*, *tray* grid dengan *ROI*, dan HMI untuk menampilkan hasil secara *real-time*. Dataset yang digunakan berjumlah 1.702 citra. Pengujian dilakukan pada beberapa skenario *tray* dengan variasi warna *tray* dan komposisi sampel, menggunakan metrik *precision, recall, F1-Score, mAP,* serta *Confution Matrix.* Hasil menunjukkan YOLOv8m+PSO lebih baik dibandingkan *baseline* YOLOv8m, dengan *F1-Score* 0.96, akurasi 0.93 dan waktu proses 7 detik, sedangkan *baseline* memperoleh *F1-Score* 0.92, akurasi 0.86, dan waktu proses 10 detik. Peningkatan juga terlihat dari naiknya TP dan turunnya FN.

Kata kunci: biji jagung, *handheld*, YOLOv8m, *Particle Swarm Optimization*, klasifikasi, *real-time*.

Halaman ini sengaja dikosongkan.

ABSTRACT

DESIGN OF A HANDHELD SYSTEM FOR CORN GRAIN CLASSIFICATION USING IMAGE PROCESSING WITH THE YOLOV8-PSO METHOD

Student Name / NRP	**:**	**Muhammad Rafi Zidan Hilmi / 2042221125**
Department	**:**	**Instrumentation Engineering**
Advisor 1	**:**	**Ir. Sefi Novendra Patrialova, S.Si., M.T.**
Advisor 2	**:**	**Akhmad Ibnu Hija, S.T., M.T.**

Abstract

Manual classification of corn kernels tends to be time-consuming and inconsistent. This study designed a handheld image processing device to classify corn kernels into two classes, namely Good and Bad, using YOLOv8m optimised by Particle Swarm Optimization (PSO) on hyperparameters. The system is equipped with a camera, LED lighting, a grid tray with ROI, and an HMI to display results in real-time. The dataset used consists of 1,702 images. Testing was conducted on several tray scenarios with variations in tray colour and sample composition, using precision, recall, F1-Score, mAP, and Confusion Matrix metrics. The results show that YOLOv8m+PSO is better than the baseline YOLOv8m, with an F1-Score of 0.96, accuracy of 0.93, and processing time of 7 seconds, while the baseline obtained an F1-Score of 0.92, accuracy of 0.86, and processing time of 10 seconds. Improvements were also seen in the increase in TP and decrease in FN.

Keywords: corn kernels, handheld, YOLOv8m, Particle Swarm Optimisation, classification, real-time.

Halaman ini sengaja dikosongkan.

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penyusunan laporan Tugas Akhir dengan lancar. Tugas akhir ini disusun sebagai salah satu syarat akademik untuk menyelesaikan masa studi di Program Studi Teknik Instrumentasi. Dalam proses pengerjaan dan penyusunan laporan Tugas Akhir ini, penulis menyadari bahwa kelancaran yang didapat tidak lepas dari doa, dukungan, bimbingan, serta bantuan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang tulus dan sebesar – besarnya kepada :

1. Allah SWT, Tuhan semesta alam, atas segala rahmat, hidayah, kekuatan, serta kemudahan yang tiada henti diberikan kepada penulis dari awal hingga selesainya Tugas Akhir ini.
2. Kedua orang tua tercinta, Bapak Muchammad Muslichan dan Ibu Nunuk Savitri, beserta kedua adik saya, yang senantiasa memberikan doa yang tak pernah putus, kasih sayang tiada tara, serta dukungan moral maupun material hingga penulis dapat menyelesaikan studi ini.
3. Ibu Ir. Sefi Novendra Patrialova, S.Si., M.T., selaku Dosen Pembimbing 1 dan Bapak Akhmad Ibnu Hija, S.T., M.T., selaku Dosen Pembimbing 2 yang telah sabar meluangkan waktu, tenaga, dan pikiran untuk memberikan bimbingan serta arahan yang sangat berharga.
4. Bapak Dr. Ir. Arief Abdurrakhman, S.T., M.T., selaku Dosen Penguji 1, dan Bapak Ahmad Radhy, S.Si., M.Si., selaku Dosen Penguji 2 yang telah memberikan evaluasi, kritik, serta masukan yang membangun demi kesempurnaan Tugas Akhir ini.
5. I28.045 Ramadhani Anandhitya Bagus Arianto, selaku *partner* Tugas Akhir yang telah berjuang bersama, membagi beban kerja, bertukar pikiran, bertukar keluh kesah, bertukar motivasi, dukungan, dan tidak kenal lelah bekerja sama dari awal hingga selesainya proyek ini.
6. Teman – teman ”MaduTj” yang tidak bisa disebutkan satu persatu namanya, yang telah menjadi keluarga dan mewarnai perjalanan suka duka selama menempuh pendidikan di jurusan ini.
7. Teman – teman ”Kuma”, terkhusus untuk Achmad Reyhan R, Galuh Ayu L, dan Khafidz Rizal M, atas segala bantuan, motivasi, kebersamaan, serta dukungan luar biasa yang selalu diberikan.
8. Rekan – rekan seperjuangan angkatan saya, I28.Tranquillis Incedo, yang telah menjadi keluarga besar, saling mendukung, bertukar pikiran, dan berjuang bersama selama masa perkuliahan.
9. Terakhir, kepada seseorang yang pernah menjadi bagian dari perjalanan hidup penulis, meskipun namanya kini tak lagi dapat disebutkan. Walau pada akhirnya ia tidak berhasil menunggu proses kelulusan penulis hingga akhir, terimakasih atas dukungan, kebersamaan, dan pelajaran berharga yang pernah diberikan.

Penulis menyadari Tugas Akhir ini belumlah sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan. Akhir kata, semoga laporan ini bermanfaat bagi pembaca dan perkembangan ilmu pengetahuan, khususnya di bidang instrumentasi.

Surabaya, 20 Juli 2026

Muhammad Rafi Zidan Hilmi

Halaman ini sengaja dikosongkan.

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan.

DAFTAR TABEL

Halaman ini sengaja dikosongkan.

BAB 1 PENDAHULUAN

1.1 Latar Belakang

PT East West Seed Indonesia (EWINDO), pemegang merek *Cap Panah Merah.* Merupakan Perusahaan benih sayuran terpadu yang telah beroperasi sejak tahun 1990. EWINDO terkenal sebagai "Sahabat Petani yang Paling Baik", EWINDO fokus pada benih berkualitas tinggi dengan kemurnian genetic tinggi dan daya kecambah optimal. Semua dijalankan oleh tim operasional dibidang pemuliaan dan pembenihan, serta dibatasi oleh 400 varietas unggul yang telah tersebar di lebih dari 7 juta petani hortikultura Indonesia (EWINDO, 2021).

Proses klasifikasi biji jagung di PT East West Seed Indonesia (EWINDO) masih dilakukan secara manual menggunakan inspeksi visual oleh tenaga manusia, yang rentan terhadap kesalahan subjektif, konsistensi mutu hasil kelas biji, serta kurang efisien dari segi waktu dan biaya. Pendekatan manual ini juga melelahkan operator dan tidak efisien dalam jumlah produksi besar. Studi Hu et al. (2025) dalam *Classification of maize seed hyperspectral images based on variable depth convolutional kernels* secara ekspilit menjelaskan bahwa metode manual memerlukan tenaga kerja intensif dan memiliki tingkat kesalahan tinggi, terutama ketika menangani variasi kompleks biji jagung dalam kondisi lapangan (Hu *et al.*, 2025).

PT East West Seed Indonesia (EWINDO) berharap dapat mengimplementasikan sistem otomatis berbasis teknologi *Image Processing* yang ringan dan mudah digunakan di lapangan untuk klasifikasi biji jagung. Sistem ini idealnya terintegrasi ke dalam perangkat *handheld*, mampu mengenali kualitas biji jagung secara *real-time* dengan akurasi tinggi, dan tahan terhadap variasi lingkungan seperti pencahayaan dan latar belakang. Selain itu, dari sisi performa sistem, teknologi yang digunakan diharapkan mampu menyeimbangkan antara akurasi deteksi, kecepatan dalam memproses, serta kestabilan hasil pada proses sehingga dapat dioperasikan langsung oleh petani tanpa memerlukan keahlian teknis yang tinggi (Hu *et al.*, 2025).

Sebagai solusi, penelitian ini bertujuan mengembangkan sebuah alat *handheld* untuk klasifikasi biji jagung secara otomatis menggunakan metode YOLOv8 yang diooptimasi dengan *Particle Swarm Optimization* (PSO). YOLOv8 dipilih karena keunggulannya dalam mendeteksi objek secara *real-time* dengan akurasi dan presisi yang tinggi terutama pada objek yang kecil, sementara PSO digunakan untuk mengoptimalkan parameter *model* sehingga menghasilkan performa deteksi yang baik. Dalam implementasi pada *Deep Learning,* Pytorch dipiluh dibandingkan TensorFlow karena arsitektur YOLOv8 dibangun secara natif dalam ekosistem PyTorch, yang menjamin kompabilitas penuh dan efisiensi komputasi, serta menawarkan fleksibilitas grafik yang dinamis untuk mempermudah itegrasi algoritma optimasi PSO. Sistem ini dirancang agar ringan, praktis, dan memiliki kesimbangan pada akurasi, kecepatan pemrosesan, dan efisiensi komputasi, sehingga siap untuk digunakan pada lapangan. Dengan demikian sistem ini diharapkan dapat meningkatkan efisiensi klasifikasi, mengurangi pada ketergantungan pada tenaga kerja manual, serta mendukung produktifitas distribusi biji jagung (Ayyad *et al.*, 2025).

Pemilihan metode YOLOv8-PSO diambil untuk menutup celah dari penelitian terdahulu: misalnya (Chen *et al.*, 2024) menggunakan YOLOv8 tapi belum mengevaluasi kualitas biji secara menyeluruh, (Niu *et al.*, 2024) fokus pada identifikasi varietas dalam laboratorium dengan komputasi tinggi, (Pativada, 2024) belum menguji dalam kondisi lapangan, (Ayyad *et al.*, 2025) menunjukkan masalah dalam mendeteksi objek kecil atau dalam pencahayaan buruk, (Che *et al.*, 2025) terbatas efisiensi dalam lingkungan laboratorium, (Elgamily *et al.*, 2025)

menggunakan optimasi hibrid PSO-GWO yang terlalu kompleks untuk *handheld,* (Fan and Zhou, 2025) memakai sistem berbasis *X-ray* yang tidak praktis untuk *edge device*, (Magdin and Balogh, 2025) memerlukan komputasi tinggi dan belum diuji di perangkat portabel, (Kılcı and Koklu, 2025) hanya diuji pada produk biskuit, kurang relevan untuk biji jagung. Dengan menggunakan pendekatan YOLOv8 dikombinasikan PSO, penelitian ini menawarkan solusi yang lebih ringan, adaptif, dan efektif untuk perangkat *handheld* di lapangan, menyasar celah-celah dari penelitian sebelumnya serta meningkatkan akurasi dan kecepatan deteksi biji dalam kondisi nyata.

Pengujian performa dalam penelitian ini dilakukan dengan cara membandingkan hasil deteksi *model* YOLOv8 tanpa optimasi PSO dengan *model* YOLOv8 yang telah di optimasi oleh PSO. Perbandingan ini difokuskan pada beberapa metrik utama yaitu *precision*, *recall*, *F1-Score*, serta *mAP* sebagai indikator akurasi deteksi. Selain itu, aspek efisiensi sistem seperti kecepatan inferensi dan jumlah frame per *second* juga diukur untuk menilai kelayakan *model* pada perangkat *handheld*. Dengan pendekatan ini, dapat dianalisis sejauh mana optimasi PSO mampu meningkatkan akurasi dan kecepatan YOLOv8, serta memastikan sistem klasifikasi biji jagung bekerja secara optimal.

1.2 Rumusan Masalah

Berdasarkan latar belakang, identifikasi masalah dan ruang lingkup penelitian, maka rumusan masalah yang akan dikaji dalam Tugas Akhir ini adalah sebagai berikut.

1. Bagaimana merancang sistem alat *handheld* berbasis *image processing* yang mampu melakukan klasifikasi biji jagung secara otomatis di lingkungan lapangan?
2. Bagaimana penerapan metode YOLOv8 dalam mendeteksi dan mengklasifikasi biji jagung berdasarkan karakteristik warna dan tekstur?
3. Bagaimana Particle Swarm Optimization (PSO) dapat diimplementasikan untuk mengoptimalkan *hyperparameter* pada *model* YOLOv8 guna meningkatkan akurasi klasifikasi biji jagung?

1.3 Batasan Masalah

Untuk membatasi ruang lingkup agar penelitian lebih berfokus dan sesuai tujuan, maka batasan masalah yang ditetapkan adalah sebagai berikut.

1. Penelitian ini hanya memfokuskan pada klasifikasi dua kelas biji jagung, yaitu jagung baik atau jagung rusak.
2. Metode yang digunakan adalah YOLOv8 yang dioptimasi menggunakan *Particle Swarm Optimization* (PSO).
3. Penelitian ini tidak fokus pada modifikasi struktur internal YOLOv8 maupun algoritma PSO, melainkan penerapan dan pengujian optimasi *hyperparameter*.
4. Penelitian ini tidak membahas tentang jenis varietas jagung.

1.4 Tujuan

Tujuan dari penelitian ini adalah untuk merancang dan mengembangkan sebuah alat *handheld* berbasis *Image Processing* menggunakan metode YOLOv8 yang dioptimasi Particle Swarm Optimization (PSO) guna melakukan klasifikasi biji jagung menjadi dua kategori, yaitu jagung baik atau jagung buruk. Secara khusus penelitian ini bertujuan untuk:

1. Merancang sistem alat *handheld* berbasis *image processing* yang dapat melakukan klasifikasi biji jagung secara otomatis dilingkungan lapangan.

2. Menerapkan metode YOLOv8 untuk mendeteksi dan mengklasifikasi biji jagung guna menghasilkan sistem yang mampu bekerja secara *real-time* dengan tingkat akurasi yang baik.

Mengimplementasikan metode *Particle Swarm Optimization* (PSO) untuk mengoptimalkan *hyperparameter* pada *model* YOLOv8 dengan tujuan meningkatkan akurasi klasifikasi biji jagung dan memastikan kinerja sistem tetap efisien pada perangkat *handheld.*

1.5 Manfaat

Penelitian ini diharapkan dapat memberi manfaat sebagai berikut:

1. Memberikan solusi teknologi yang efisien dan praktis untuk membantu proses klasifikasi biji jagung secara otomatis, sehingga meningkatkan konsistensi mutu dan mengurangi ketergantungan pada tenaga kerja manual.
2. Menjadi referensi dalam penerapan metode YOLOv8-PSO untuk klasifikasi objek pertanian dalam perangkat *handheld* yang ringan dan *portable*.
3. Menambah literatur dan pengayaan dalam bidang pengolahan citra digital serta penerapan metode optimasi dalam sistem klasifikasi berbasis *deep learning.*

Halaman ini sengaja dikosongkan.

BAB 2 TINJAUAN PUSTAKA

2.1 Hasil Penelitian Terdahulu

Jagung merupakan salah satu komoditas pangan utama di Indonesia, yang banyak dimanfaatkan dalam industri pangan dan pakan ternak. Untuk menjaga kualitasnya, proses klasifikasi biji jagung menjadi hal penting, namun hingga kini masih banyak dilakukan secara manual. Kemajuan teknologi *Image Processing* dan *deep learning* seperti YOLOv8 membuka peluang baru dalam klasifikasi objek secara otomatis. Dengan mengombinasikan metode YOLOv8 dan *Particle Swarm Optimization* (PSO), sistem klasifikasi dapat dioptimalkan agar lebih akurat dan efisien

Patiavada (2024) **mengimplementasikan YOLOv8** untuk **klasifikasi empat jenis benih** (padi, kacang polong, kedelai, dan gandum) **secara *real-time***. Dengan **dataset** hasil augmentasi **sebanyak 2.237 gambar**, *mode*l ini mencapai ***mAP* 97.3%, *Precision* 91.5%, dan *recall* 95.3%.** meskipun melampaui performa YOLOv5, penelitian ini **masih terbatas pada kondisi labolatorium** dan belum diuji pada perangkat *edge* maupun kondisi lapangan langsung (Pavan Kumar Pativada, 2024).

Chen et Al (2024) menerapkan **YOLOv8 untuk mendeteksi keretakan biji jagung** pada kondisi lapangan **dengan latar belakang tidak seragam dan objek yang terhalang oleh objek lain (*occlusion)* meskipun mencapai akurasi 94.3%,** penelitian ini masih berfokus pada deteksi visual permukaan, dan **tidak memperhatikan kualitas benih maupun implementasi pada perangkat *handheld*** (Chen *et al.*, 2024).

Niu et Al. (2024) melakukan **identifikasi varietas jagung** melalui metode ***transfer learning*** pada arsitektur YOLOv8. Dengan melatih *model* **terhadap tiga jenis varietas berbeda,** sistem ini **mampu melakukan klasifikasi otomatis dengan akurasi melampaui 95%.** Fokus penelitian **masih sebatas pada pengenalan jenis** tanpa memberikan aspek **kualitas atau kerusakan biji**. Selain itu, ketergantungan pada **perangkat komputasi tinggi** dilingkungan labolatorium **membuat *model* yang digunakan belum dapat diterapkan secara portabel dilapangan** (Niu *et al.*, 2024).

Midigudla et al. (2025) **membandingkan *model* YOLOv8, EfficientDet, dan Detectron 2** menggunakan **6.000 citra** hasil augmentasi. Hasil pengujian menunjukkan varian **YOLOv8x mencapai *mAP50* sebesar 42.6%,** mengungguli varian **YOLOv8l (41.4%), YOLOv8m (40.5%)** dan **YOLOv8s (39.5%).** Mesikpun **Detectron 2 mencatat *mAP50* 78%** dan **EfficientDet mencapai 70.3%, YOLOv8 dinilai paling unggul dalam aspek kecepatan, akurasi, dan skalabilitas untuk aplikasi *real-time.*** Namun efektivitas *model* ini masih **memiliki keterbatasan pada kondisi pencahayaan yang bervariasi** (Sai *et al.*, 2025).

Ayyad et al (2025). **Mengoptimasi YOLOv8 menggunakan *Particle Swarm Optimization* (PSO)** untuk **mendeteksi tingkat kesehatan dan kematangan buah tomat**. Melalui proses **pelatihan sebanyak 500 *epoch,*** integrasi ini **menghasilkan *mAP* sebesar 0.89** yang mengungguli performa versi YOLO sebelumnya. Meskipun menunjukkan peningkatan performa***, model* ini masih memiliki kelemahan dalam mengenali objek berukuran kecil objek yang tertutup dedaunan, serta kondisi cahaya yang tidak seragam** (Ayyad *et al.*, 2025).

Che et al. (2025). **Mengembangkan sistem deteksi biji jagung pada konveyor** menggunakan **arsitektur YOLOv5** dan **kamera CMOS**. Sistem ini mencapai **akurasi 97,83%** dengan waktu inferensi sangat cepat, **yakni 19.8 ms per citra**. Meskipun efektif untuk menggunaan industri diatas konveyor, *model* ini **belum diuji pada kondisi lapangan yang**

dinamis. Peneliti juga mencatat **bahwa YOLOv5 masih memiliki keterbatasan** dalam **mendeteksi objek berukuran kecil** jika dibandingkan **dengan performa YOLOv8** (Che *et al.*, 2025).

Kilci & Koklu, (2025). Mengevaluasi **lima varian YOLOv8 untuk klasifikasi kualitas biskuit** berdasarkan dataset sebanyak **5.000 citra**. Hasil pengujian menunjukkan bahwa *model* **YOLOv8-m memberikan akurasi tertinggi sebesar 96.99%** dalam membedakan biskuit normal dari berbagai kategori cacat produksi. Meskipun memilki performa tinggi, penelitian ini **masih terbatas pada satu jenis produk** dan dilakukan dalam **skala labolatorium**, sehingga pada im**plementasinya pada ekosistem manufaktur skala besar masih memerlukan validasi lebih lanjut** (Kılcı and Koklu, 2025).

Elgamy et.al (2025). Mengeksplorasi **optimasi YOLOv8** menggunakan **algoritma hibrida PSO-GWO** untuk **meningkatkan akurasi deteksi**. Meski berhasil **mencapai *mAP* 75.6% dan presisi 81.2%**, integrasi algoritma ini berdampak pada meningkatnya kompleksitas arsitektur *model.* Akibatnya sistem tersebut menjadi kurang efisien untuk penggunaan pada perangkat *handheld* **karena tingginya kebutuhan sumber daya komputasi** (Elgamily *et al.*, 2025).

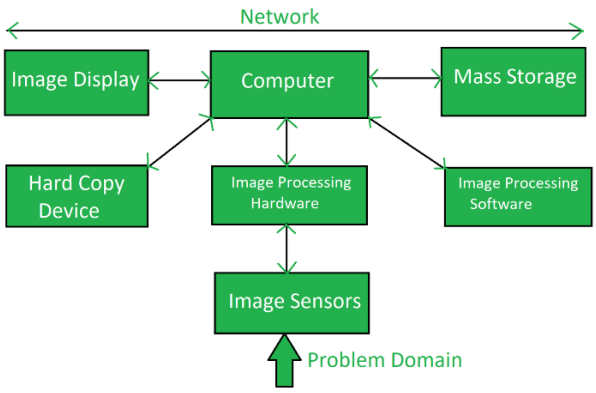
Magdin & Balogh (2025). **Mengevaluasi performa YOLOv8 dalam analisis video** dengan membandingkanya terhadap ***mode*l NasNetLarge dan InceptionV3**. Hasil pengujian menunjukkan **keunggulan YOLOv8 dengan akurasi 94.79% dan *F1-Score* tertinggi**. Meskipun unggul secara performa, pelatihan ini masih terbatas pada **penggunaan komputer berspesifikasi tinggi di labolatorium** dan **belum menguji efektivitas *mode*l** pada perangkat edge (Magdin and Balogh, 2025).

Fan & Zhou (2025). Beralih dari pendekatan visual **konvensional ke pencitraan X-Ray menggunakan *model lightweight* WKnet** untuk mendeteksi kualitas walnut secara non-dekstruktif. Sistem ini mencapai ***mAP* sebesar 98.69%** dengan waktu inferensi 11.9ms per citra. Meskipun **menunjukkan presisi tinggi di lapangan**, penggunaan metode ini pada perangkat portabel masih **sulit dilakukan karena ketergantungan pada unit GPU besar** dan kompleksitas **pada sistem deteksi X-Ray** (Fan and Zhou, 2025).

2.2 Dasar Teori

Berikut ini merupakan teori yang digunakan dalam menunjang penelitian Tugas Akhir sebagai berikut:

2.2.1 Pengolahan Citra Digital (*Image Processing*)



Gambar 2.1 *Component of Image Processing System*
(Sumber: (Greeksforgeeks, 2025))

Pengolahan citra digital merupakan proses transformasi atau analisis terhadap suatu gambar digital dengan bantuan komputer, dengan tujuan utama untuk meningkatkan kualitas visual atau mengekstraksi informasi penting dari gambar tersebut. Gambar digital sendiri merupakan representasi dua dimensi dari objek nyata, yang ditangkap oleh sensor kamera dalam bentuk nilai piksel.

Salah satu keunggulan pengolahan citra digital dibanding metode analog adalah kemampuannya dalam memanipulasi gambar secara fleksibel dan juga presisi yang tinggi, serta integrasi dengan kecerdasan buatan (AI) untuk optimasi dan klasifikasi objek dalam gambar. Penelitian Archana dan teman-teman 2024, pemrosesan citra digital ini semakin berkembang dengan pendekatan berbasis pembelajaran mendalam (*deep learning)*, yang secara signifikan meningkatkan kemampuan sistem dalam memahami dan menginterpretasi data visual secara otomatis (Archana and Jeevaraj, 2024).

2.2.1.1 Prinsip Pengolahan Citra Digital

Pengolahan citra digital pada dasarnya mengubah cahaya dari objek menjadi *data digital* yang bisa dibaca dan dianalisis oleh komputer. Cahaya yang dipantulkan dari permukaan objek ditangkap oleh sensor kamera, seperti sensor CMOS. Sensor ini mengubah cahaya menjadi sinyal Listrik, lalu sinyal tersebut dikonversi menjadi angka digital menggunakan komponen yang disebut ADC (*Analog-to-digital Converter*).

Sensor CMOS unggul dalam efisiensi daya dan kemampuan integrasi sistem dibandingkan sensor CCD. Sinyal Listrik yang diperolah oleh sensor kemudian diubah menjadi representasi numerik (metrik piksel) berdasarkan intensitas cahaya atau warna untuk setiap titik gambar (New Chips Poised to Revolutionize Photography, 2008). Setelah citra digital terbentuk, komputer menjalankan serangkaian proses seperti:

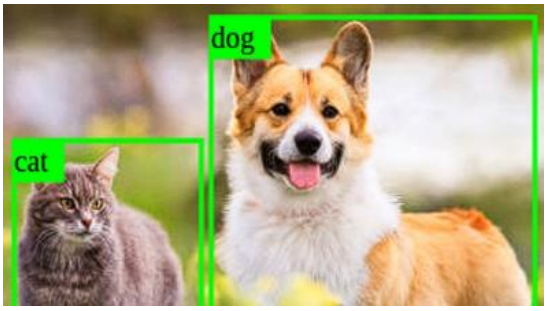
- *Enhancement* bertujuan meningkatkan kualitas visual citra (misalnya penajaman tepi, peningkatan pada kontras, atau pengurangan *noise*.
- *Segmentation* bertujuan memisahkan objek utama dari latar belakang dengan mengenali batas-batas visual.
- *Feature Extraction* bertujuan mengambil karakteristik penting seperti bentuk, ukuran, atau tekstur dari objek.
- *Classification* bertujuan untuk mengelompokan objek berdasarkan fitur tertentu menggunakan algoritma kecerdasan buatan (AI).

Metode pengolahan citra digital lebih unggul dibandingkan teknik analog karena fleksibilitas, kecepatan, dan presisi dalam analisis visual berbagai objek.

Dengan kemajuan teknologi *deep learning*, sistem deteksi objek seperti YOLOv8 berkembang menjadi salah satu *model* yang andal untuk mendeteksi objek secara *real-time*. *Model* ini dirancang dengan arsitektur yang lebih efisien sehingga mampu menghasilkan proses deteksi yang cepat tanpa mengurangi tingkat akurasi. Hasil tinjauan dari berbagai literatur menunjukkan bahwa YOLOv8 memberikan peningkatan akurasi dan efisiensi komputasi dibandingkan generasi sebelumnya, termasuk pada kondisi yang menantang seperti objek berukuran kecil, pencahayaan rendah, maupun latar belakang yang kompleks. Selain itu, YOLOv8 memiliki kemampuan melakukan ekstraksi fitur secara otomatis dari citra, sehingga proses identifikasi objek dapat dilakukan tanpa memerlukan perancangan fitur secara manual. Kemampuan tersebut menjadikan YOLOv8 banyak dimanfaatkan dalam berbagai aplikasi berbasis visi komputer. (Megantara and Utami, 2025).

2.2.2 Deteksi Objek dan Klasifikasi Citra

2.2.2.1 Klasifikasi Citra

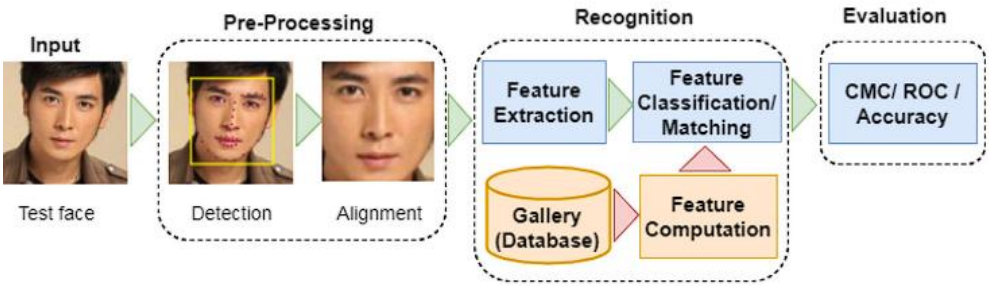


Gambar 2.2 Klasifikasi citra
(Sumber: (Andrey Germanov, 2024))

Klasifikasi citra adalah proses mengenali isi dari sebuah gambar digital dan mengelompokkan gambar tersebut ke dalam kelas tertentu berdasarkan ciri visual yang dimilikinya. Tujuan dari klasifikasi ini adalah agar sistem komputer dapat memahami gambar seperti halnya pada manusia dengan melihat bentuk, warna, tekstur, dan pola lalu memutuskan apakah gambar itu termasuk apel, jeruk, daun sehat, atau daun yang terkena penyakit.

Pada perkembangan teknologi, klasifikasi citra tidak hanya dilakukan dengan metode konvensional seperti SVM atau k-NN, tetapi kini banyak menggunakan pendekatan *deep learning,* seperti *Convensional Neural Network* (CNN) atau arsitektur deteksi objek seperti YOLO (*You Look Only Once)*. *Model* ini akan dilatih menggunakan ratusan hingga ribuan gambar berlabel, agar dapat belajar mengenali karakteristik visual dari masing – masing kelas secara otomatis. Bahkan untuk meningkatkan akurasi, beberapa penelitian memadukan *model* deteksi seperti YOLO dengan algoritma optimasi seperti *Particle Swarm Optimization* agar parameter *model* dapat disesuaikan dengan kondisi data secara lebih efektif (Ayyad *et al.*, 2025).

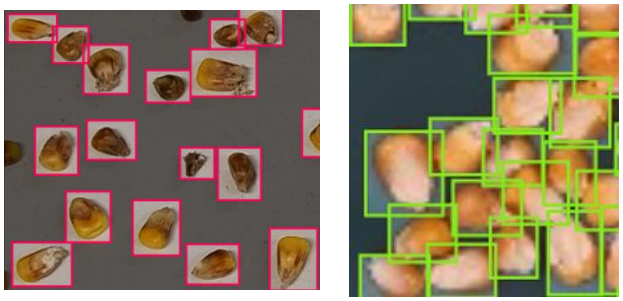
2.2.2.2 Deteksi Objek



Gambar 2.3 Deteksi objek
(Sumber: (Luo *et al.*, 2019))

Deteksi objek merupakan teknik dalam bidang komputer Vision yang bertujuan untuk mengenali dan menentukan lokasi objek dalam sebuah gambar atau video. Teknologi ini tidak hanya memberi label pada objek seperti pada klasifikasi citra, tetapi juga memberikan posisi objek dengan menggunakan bounding box. Deteksi objek telah berkembang pesat berkat algoritma *deep learning*, khususnya *Convolutional Neural Network* (CNN), yang mampu mempelajari pola visual dari data gambar. Menurut (Luo *et al.*, 2019), deteksi objek memainkan peran penting dalam berbagai aplikasi mulai dari kendaraan otonom hingga pertanian presisi, karena kemampuannya dalam mengenali objek secara akurat dalam berbagai kondisi visual (Luo *et al.*, 2019).

2.2.2.3 *Labeling* Tekstur Warna pada gambar



Gambar 2.4 *Labeling* tekstur warna pada gambar
(Sumber: (Grad, 2024))

Pelabelan pada citra (*image labelling*) adalah tahap awal yang penting dalam sistem klasifikasi berbasis deep learning, termasuk pada *model* YOLOv8-PSO. Proses ini dilakukan dengan memberikan *bounding box* dan label kelas pada objek yang akan diklasifikasi. Label tersebut digunakan sebagai data train (melatih data) agar *model* mampu mengenali, mengklasifikasi, dan membedakan objek secara otomatis.

Aspek penting yang diperhatikan dalam labeling gambar adalah warna dan tekstur seperti warna dan tekstur. Warna digunakan untuk mengenali objek berdasarkan visual seperti intensitas merah, hijau, dan biru (RGB) atau melalui konversi ke *model* warna lain seperti HSV, contohnya seperti biji jagung yang memiliki kualitas bagus memiliki warna kuning merata , sedangkan yang buruk kualitasnya akan memiliki corak gelap atau dominan ke arah yang gelap. Sedangkan tekstur merepresentasikan pola permukaan objek, seperti halus, kasar, atau berbintik. Tekstur dapat membantu membedakan objek dengan bentuk serupa, tetapi permukaan yang berbeda, seperti biji jagung baik (mengkilap) dan biji jagung buruk (tekstur berbintik atau belang).

Dalam penelitian oleh Ayyad et al., 2025 penggunaan YOLOv8 yang dioptimasi PSO berhasil meningkatkan akurasi deteksi kematangan dan penyakit pada tomat, dan bergantung pada ciri visual seperti warna kulit, dan tekstur permukaan pada buah tomat. Hal ini menunjukkan pentingnya *labeling* berdasarkan warna dan tekstur dalam mendukung sistem klasifikasi berbasis komputer (Ayyad *et al.*, 2025).

2.2.2.4 Peran Jumlah Epoch dalam pelatihan YOLOv8-PSO

Dalam konteks pelatihan *model* deteksi objek berbasis *deep learning* seperti YOLOv8, jumlah *epoch* merujuk pada beberapa kali seluruh dataset pelatihan diproses secara penuh oleh *model*. Setiap *epoch* memberikan kesempatan bagi *model* untuk mempelajari pola-pola dalam data dan memperbarui bobot jaringan agar prediksi menjadi lebih akurat.

Berdasarkan penelitian oleh Ayyad et al. 2025 dalam jurnal *Particle Swarm Optimization with YOLOv8 for Improved Detection Performance of Tomato Plants* jumlah *epoch* menjadi faktor krusial dalam mencapai konvergensi *model*, yaitu kondisi dimana nilai *loss* menurun stabil dan performa *model* terhadap validasi meningkat. Dalam penelitian tersebut, *model* YOLOv8 yang di optimasi menggunakan *Particle Swarm Optimization* (PSO) dilatih selama 500 *epoch* dan ditunjukkan bahwa peningkatan jumlah *epoch* berkontribusi signifikan terhadap perbaikan metrik seperti:

- *Box loss* bertujuan mengukur seberapa tepat posisi kotak prediksi terhadap objek.
- *Objectnes loss* bertujuan menilai confidence (keyakinan) *model* apakah ada objek dalam suatu area gambar.

- *Class loss* bertujuan mengukur kesalahan dalam mengklasifikasikan jenis objek yang terdeteksi.
- *Precision* menunjukkan presentasi deteksi yang benar dari semua objek yang berhasil di deteksi.
- *Recall* bertujuan untuk presentasi objek yang berhasil terdeteksi dari semua objek yang seharusnya terdeteksi.
- *Mean Average Precision* (*mAP@0.5*) bertujuan untuk menilai rata – rata akurasi *model* dalam mendeteksi objek dan mengklasifikasikan objek, dengan ambang kecocokan bounding box minimal 50%.

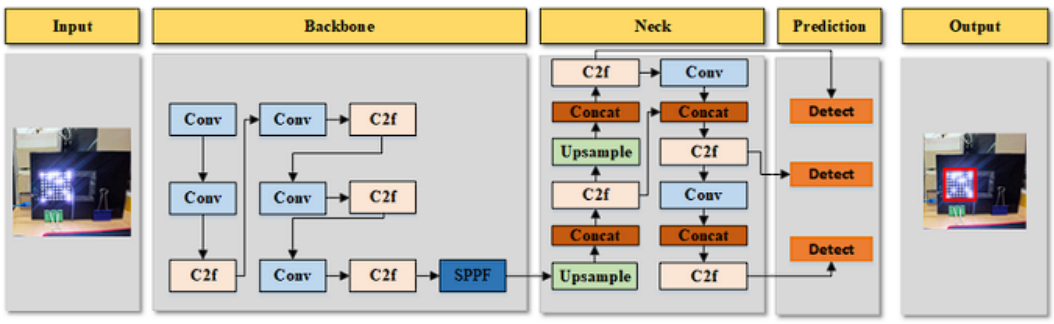
Namun demikian, pemilihan jumlah *epoch* tidak bisa dilakukan sembarangan. Terlalu sedikit *epoch* dapat menyebabkan *underfitting*, dimana *model* belum cukup belajar dari data. Sebaliknya, terlalu banyak *epoch* dapat menyebabkan *overfitting*, yakni *model* terlalu menyesuaikan diri pada data latih dan kehilangan kemampuan untuk melakukan generalisasi terhadap data baru. Dalam konteks YOLOv8-PSO, PSO dapat digunakan untuk mengoptimasi parameter pelatihan seperti learning rate, batch size, dan jumlah *epoch* agar *model* memperoleh konfigurasi *hyperparameter* terbaik. Oleh karena itu, jumlah *epoch* yang tepat merupakan salah satu kunci keberhasilan dalam pelatihan *model* klasifikasi dan deteksi objek (Ayyad *et al.*, 2025).

2.2.3 YOLOv8 (*You Look Only Once*)

Gambar 2.5 YOLOv8
(Sumber: (Yolo.org, 2023))

YOLOv8 (*You Look Only Once*) merupakan pengembangan terbaru dari algoritma deteksi objek yang dirancang untuk mendeteksi dan mengenali objek dalam gambar atau video secara cepat dan akurat. Tidak seperti versi-versi sebelumnya, YOLOv8 memperkenalkan pendekatan baru yang lebih efisien, termasuk penggunaan prediksi tanpa anchor, yang menyederhanakan proses pelatihan dan mengurangi ketergantungan pada parameter awal. Dalam arsitektur internalnya, YOLOv8 tetap memanfaatkan prinsip jaringan konvolutional untuk mengekstraksi fitur visual dari gambar, dimulai dari lapisan backbone yang menangkap detail spasial, kemudian dilanjutkan ke bagian neck yang menggabungkan fitur dari berbagai skala untuk membantu mendeteksi objek dengan ukuran berbeda (Yaseen, 2024).

2.2.3.1 Arsitektur / Layer dalam YOLO V8



Gambar 2.6 Arsitektur YOLOv8
(Sumber: (Yaseen, 2024))

YOLOv8 menggunakan arsitektur anchor-free, yang berarti tidak memerlukan anchor boxes seperti pada versi sebelumnya. Pendekatan ini menyederhanakan proses pelatihan dan mengurangi ketergantungan terhadap konfigurasi awal, yang sering kali memengaruhi hasil deteksi objek kecil. Arsitektur YOLOv8 terdiri dari tiga komponen utama yaitu backbone (biasanya CSPDarknet) untuk mengekstraksi fitur dari citra, Neck (seperti PANet atau FPN) untuk menggabungkan fitur Multi-skala, dan Head untuk melakukan prediksi bounding box dan klasifikasi objek secara langsung.

Dalam arsitektur jaringan convolutional seperti YOLOv8, *layer* (lapisan) adalah komponen utama yang membentuk proses – pemrosesan citra dari input yang mentah menjadi prediksi akhir berupa bounding box dan klasifikasi objek. YOLOv8 mengadopsi struktur modular yang terdiri dari tiga bagian utama, yaitu *backbone*, *neck*, dan *head*, dimana masing – masing tersusun dari berbagai jenis *layer* sebagaimana berikut ini

1. *Backbone layer*
   Fungsi dari *backbone layer* yaitu untuk mengekstraksi fitur visual dari gambar input. YOLOv8 menggunakan varian backbone modern seperti C2f (*Cross-Stage Partial with Focus*) yang memproses informasi spasial dan struktual dengan efisiensi tinggi.
   - Conv *Layer* (*Convolutional*) bertujuan untuk menangkap pola lokal seperti tepi dan tekstur.
   - BatchNorm *Layer* bertujuan untuk menstabilkan distribusi aktivasi dan mempercepat konvergensi.
   - SiLU/Leaky ReLU berfungsi sebagai aktivasi *non-linear* yang membuat jaringan dapat mempelajari representasi kompleks.
   - C2f *Modul* bertujuan untuk penyempurnaan dari C3 di YOLOv8 yang lebih ringan dan efisien.
2. *Neck layer*
   Fungsi dari *neck layer* yaitu menggabungkan fitur dari berbagai skala resolusi agar deteksi objek kecil dan besar bisa akurat.
   - FPN (*Feature Pyramid Network*) berfungsi sebagai penyalur informasi dari level deep ke level *swallow* dan sebaliknya.
   - *Concat Layer* berfungsi menggabungkan fitur dari layer berbeda.
   - *Upsample layer* berfungsi menaikkan resolusi fitur *map* untuk mempertahankan detail spasial.
3. *Head layer*
   Fungsi dari *head layer* adalah melakukan prediksi akhir, yaitu koordinat *bounding box*, confidence score, dan klasifikasi objek.
   - YOLOv8 memakai *anchor free detection head*, artinya tidak lagi bergantung pada anchor box seperti versi sebelumnya.
   - Output dari head layer berupa tiga komponen yaitu *regression* (lokasi dan ukuran kotak), *objectness* (*confidence* berisi objek) dan *class scores* (jenis objek)

Model ini dirancang modular dan ringan, memungkinkan penyesuaian untuk ukuran *model* kontroller. YOLOv8 juga mengadopsi fungsi *Loss* yang lebih canggih seperti CloU *Loss* (*Complete Intersection over Union)* untuk evaluasi akurasi prediksi bounding box secara lebih menyeluruh. Dalam implementasinya, YOLOv8 menunjukkan performa deteksi yang stabil bahkan dalam kondisi pencahayaan yang buruh dan *background* kompleks. Dalam penelitian (Yaseen, 2024), YOLOv8 tidak hanya meningkatkan kecepatan dan efisiensi, tetapi juga memberikan fleksibilitas *model* yang lebih besar, menjadikannya salah satu arsitektur deteksi yang paling kompatibel dalam aplikasi *edge devices* (Yaseen, 2024).

2.2.3.2 Rumus YOLOv8 dalam Matematika

Dalam algoritma YOLOv8, proses pelatihan *model* tidak hanya bergantung pada arsitektur jaringan yang efisien, tetapi juga pada formulasi matematis yang digunakan untuk mengukur kualitas prediksi. Formulasi ini dikenal sebagai fungsi *loss*, yang menjadi inti dari pembelajaran *model* deteksi objek.

Fungsi *loss* digunakan untuk mengevaluasi seberapa dekat prediksi *model* terhadap data sebenarnya (*ground truth*). Semakin kecil nilai *loss*, semakin baik performa model dalam mendeteksi dan mengklasifikasikan objek. Pada YOLOv8, komponen utama dalam fungsi *loss* mencakup tiga aspek yaitu akurasi posisi, *bounding box*, prediksi keberadaan objek, dan klasifikasi jenis objek. Salah satu pendekatan yang banyak digunakan dalam YOLOv8 adalah fungsi *Complete Intersection over Union* (CloU) *Loss*, yang merupakan pengembangan dari metrik IoU tradisional. CloU tidak hanya mempertimbangkan luas tumpang tindih antara kotak prediksi dan ground truth, tetapi juga memperhitungkan jarak pusat objek dan rasio aspek kotak untuk meningkatkan akurasi dan konvergensi *model* secara keseluruhan.

Berikut ini merupakan penjelasan dan formulasi matematis dari beberapa jenis fungsi *loss* yang digunakan dalam YOLOv8, sebagaimana dirujuk dalam jurnal (Zheng *et al.*, 2020).

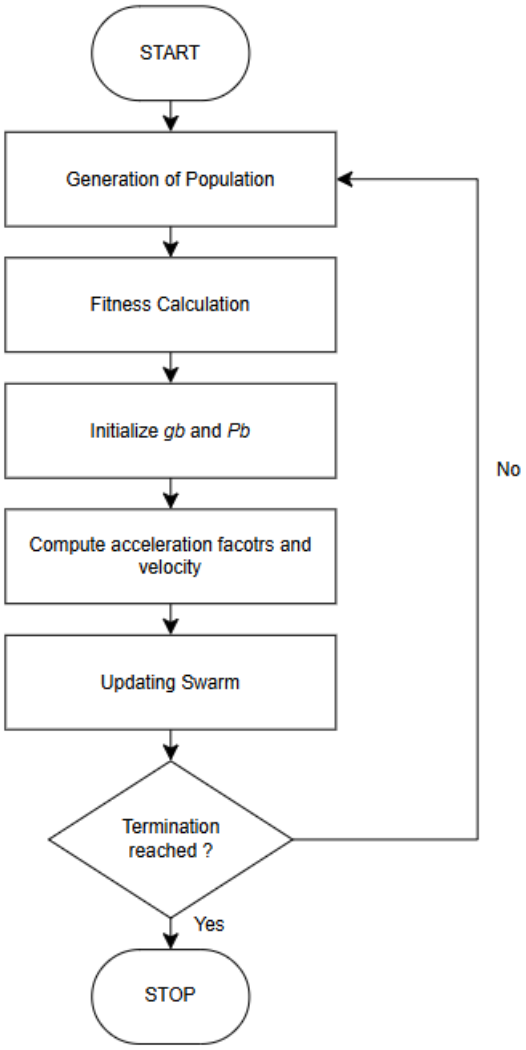
1. *Intersection over Union* (IoU)
   IoU adalah metrik dasar dalam YOLOv8 untuk menilai kesesuaian antara prediksi dan ground truth (prediksi *model* terhadap data sebenarnya). Nilai IoU berada diantara 0-1, semakin tinggi nilai IoU semakin akurat prediksi.
   Rumus:
   $$IoU = \frac{B \cap Bgt}{B \cup Bgt} \tag{2.1}$$
   - B: *bounding box* prediksi
   - B_{gt}: *bounding box ground truth*
2. IoU *Loss*
   Untuk mengoptimasi regresi bounding box, digunakan loss sebagai berikut:
   $$L_{IoU} = 1 - IoU \tag{2.2}$$
   Namun, IoU *loss* tidak dapat memberikan gradient jika tidak ada tumpang tindih (*non-overlapping boxes*) yang dimana bounding box prediksi dan bounding box ground truth memiliki area yang saling menutupi.
3. *Generalized* IoU (GloU) *Loss*
   Untuk mengatasi keterbatasan IoU *loss*, ditambahkan penalti terhadap area diluar gabungan box:
   $$L_{IoU} = 1 - IoU + \frac{|C-(B \cup Bgt)|}{|C|} \tag{2.3}$$
   - C: kotak terkecil yang mencakup B dan B_{gt}
4. *Distance* IoU (DIoU) *Loss*
   Mengintegrasikan jarak antara pusat kotak prediksi dan *ground truth*.
   $$L_{DIoU} = 1 - IoU + \frac{p2(b,bgt)}{C2} \tag{2.4}$$
   - ρ: jarak Euclidean antara pusat prediksi dan *ground truth*
   - c: panjang diagonal dari kotak yang mencakup keduanya

   Keunggulannya : konvergensi lebih cepat dan presisi lebih tinggi.
5. *Complete* IoU (CIoU) *Loss*
   CloU mempertimbangkan 3 faktor geometris penting, yaitu luas tumpang tindih, jarak pusat, dan konsistensi rasio aspek.
   $$L_{CIoU} = 1 - IoU + \frac{p2(b,bgt)}{C2} + \alpha v \tag{2.5}$$

- v: konsistensi rasio aspek
- α: Faktor pengatur bobot penalti

Penggunaan rumus-rumus ini bertujuan untuk meningkatkan konvergensi *model* serta akurasi deteksi objek dengan mempertimbangkan lebih dari sekedar tumpang tindih (Zheng *et al.*, 2020).

2.2.4 *Particle Swarm Optimization* (PSO)



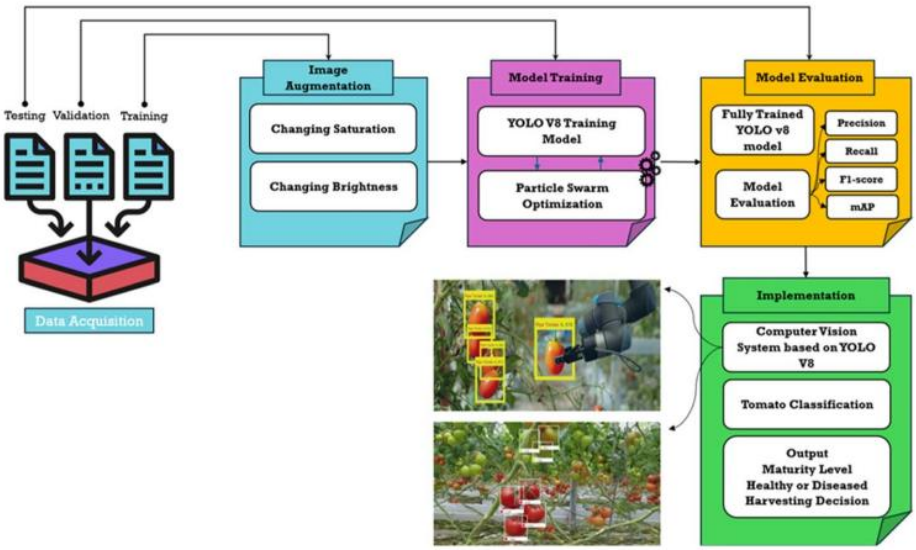
Gambar 2.7 *Particle Swarm Optimization* (PSO)
(Sumber: (Kalimuthu and Krishnamurthi, 2015))

Image Processing saat ini banyak mengandalkan *deep learning* untuk mendeteksi objek, salah satunya melalui algoritma *You Look Only Once,* merupakan *model* deteksi objek berbasis *convolutional neural network* (CNN) yang unggul dalam kecepatan akurasi. Namun, performa YOLOv8 sangat bergantung pada parameter-parameter seperti anchor boxes, threshold confidence, dan konfigurasi layer, yang sering ditentukan secara manual atau melalui eksperimen acak (Zhong *et al.*, 2025).

Particle Swarm Optimization (PSO) adalah algoritma metaheuristic yang terinspirasi dari perilaku sosial kawanan burung atau ikan. PSO bekerja dengan menyimulasikan sekelompok partikel yang menjelajahi ruang solusi dan saling berbagi informasi terbaik untuk mencapai

solusi global optimum. Dalam konteks YOLOv8, PSO dapat digunakan untuk mengoptimalkan hypermeter *model* secara otomatis.

2.2.4.1 *Hypermeter* PSO (*Particle Swarm Optimization*)



Gambar 2.8 Proses *Train* YOLOv8 menggunakan PSO
(Sumber:(Ayyad *et al.*, 2025))

Dalam penerapan PSO untuk optimasi *model deep learning* seperti YOLOv8, *hyperparameter* yang di optimasi bisa mencakup *learning rate*, IoU *threshold*, *confidence score*, jumlah *epoch*, dan ukuran batch. *Model* YOLOv8 dilatih dengan *hyperparameter* optimal yang dihasilkan oleh teknik PSO. Tujuan utama dari PSO adalah untuk menyelesaikan berbagai kombinasi dan secara acak mencari *global solution* (*gBest*) yang memiliki kualitas operasi yang cepat, paralelisme, fleksibilitas, kemudahan implementasi, dan konvergensi cepat (Ayyad *et al.*, 2025).

2.2.4.2 Skema Kerja PSO (*Particle Swarm Optimization*)

Skema Kerja dari Particle Swarm Optimization terinspirasi dari perilaku kawanan burung dan ikan. Setiap kandidat dari solusi yang dihasilkan disebut sebagai partikel (*pBest*) yang memiliki representasi solusi dan kecepatan. Proses optimasi berlangsung secara iterative (iterasi dalam proses PSO) dengan memanfaatkan dua jenis hasil optimasi, pengalaman individu (partikel) yang disebut *pBest* (particle best) dan pengalaman kolektif kawanan (global) yang disebut *gBest*). Menurut Marini & Walczak (2015), tahapan dari optimasi PSO bisa dijelaskan sebagai berikut:

1. Inisialisasi
   - PSO akan menentukan jumlah partikel dalam swarm, lalu setiap partikel akan diberikan secara otomatis posisi awal secara acak dalam ruang pencarian (swarm).
   - Kecepatan awal pada partikel juga diinisialisasi (bisa berupa nominal acak atau bernilai kecil) lalu fitness akan dihitung, selanjutnya posisi awal dicatat sebagai *pbest* (partikel best) dan nilai terbaik dari seluruh partikel disebut atau ditetapkan sebagai *gbest* (global best).
2. Evaluasi dari *Fitness*
   - Pada setiap iterasi, posisi partikel akan dievaluasi menggunakan fungsi objektif dan fitness akan menentukan kualitas dari solusi yang diwakili oleh partikel (*pbest*)

3. Pembaruan pada *pBest* dan *gBest*
   - Jika solusi saat ini lebih baik dari partikel best (*pbest*) maka akan diperbarui, jika solusi yang terbaik dari *pbest* lebih baik dari *gbest* maka *gbest* akan diperbarui.
4. Pembaruan pada Kecepatan dan Posisi
   - Kecepatan pada partikel diupdate berdasarkan tiga komponen, yaitu :
   - *Inersia*: mempertahankan dari arah gerakan sebelumnya
   - *Cognitive*: ketertarikan partikel kembali ke posisi terbaik dirinya (*pbest*)
   - *Social*: kecenderungan mengikuti posisi terbaik dari *swarm* (ruang pencarian) yang disebut *pbest*.
5. Evaluasi iterasi ke iterasi
   - Iterasi akan berlanjut hingga tercapai jumlah iterasi maksimum atau nilai *fitness* yang memenuhi kriteria optimal dan pada akhir proses, solusi terbaik *global Best* (*gbest*) dianggap sebagai hasil optimasi.

2.2.4.3 Struktur Dasar PSO

Particle Swarm Optimization (PSO) merupakan salah satu metode optimasi berbasis populasi yang terinspirasi dari perilaku kawanan burung, ikan, atau serangga dalam mencari sumber makanan. Setiap kandidat dari solusi hypermeter direpresentasikan sebagai partikel yang bergerak dalam ruang pencarian (*swarm*). Pergerakan partikel memengaruhi oleh dua hal yaitu pengalaman individu (*pbest*) dan pengalaman kelompok (*gbest*) (Perez and Behdinan, 2007). Adapun elemen struktur dari PSO sebagai berikut:

1. Partikel (*Particle*) menunjukkan solusi dalam proses penentuan *hyperparameter*.
2. Posisi (X_i) menunjukkan letak partikel dalam ruang pencarian (*swarm*).
3. Kecepatan (V_i) menentukan arah dan langkah pergerakan dari partikel.
4. *Pbest* (*personal best*) menunjukkan posisi *hypermeter* terbaik yang pernah didapat setelah proses pada masing – masing partikel.
5. *Gbest* (*global best*) menunjukan posisi hypermeter terbaik yang pernah didapat setelah proses oleh seluruh partikel.

2.2.4.4 Rumus Matematis dalam PSO (*Particle Swarm Optimization*)

Rumus dasar dalam *Particle Swarm Optimization* (PSO) sebenarnya terdiri dari dua komponen yaitu update kecepatan dan update posisi. Berikut penjelasan rumus dasar pada PSO:

1. Rumus Kecepatan pada PSO (*Velocity Update*)

$$v_i^{t+1} = w \,.\, v_i^t + c1 \,.\, ri \,.\, (pbest_i + x_i^t) + c_2 \,.\, r_2 \,.\, (gbest - x_i^t) \qquad (2.6)$$

Penjelasan:
- v_i^{t+1} : kecepatan partikel ke – i pada iterasi t + 1
- v_i^t : kecepatan partikel ke – i pada iterasi t
- x_i^t : posisi partikel ke – i pada iterasi t
- w : Inertia weight, pengendali kecepatan sebelumnya pada swarm
- C_1 : Koefisien Kognitif, bobot terbaik pada setiap partikel
- C_2 : Koefisien Sosial, bobot terbaik pada seluruh swarm
- r_1 , r_2: Bilangan acak random antara 0-1 menambah faktor eksplorasi
- $pbest_i$: Posisi terbaik dari partikel iterasi
- $gbest_i$: Posisi terbaik global seluruh partikel iterasi.

2. Rumus Posisi (*Position Update*)

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2.7)$$

Penjelasan:

- x_i^{t+1} : Posisi baru partikel ke – iterasi t + 1
- x_i^t : Posisi partikel ke – iterasi t.
- v_i^{t+1} : Kecepatan terbaru partikel ke – i

2.2.5 Pyhton

Gambar 2.9 Python
(Sumber: (Guido van rossum, 2025)

Python adalah bahasa pemrograman yang bersifat interpretatif, berorientasi objek, dan memiliki sintak yang sederhana serta mudah dipelajari. Bahasa ini pertama kali dikembangkan oleh Guido van Rossum pada akhir tahun 1980-an dan sejak itu menjadi salah satu bahasa pemrograman yang paling populer didunia. Python dirancang agar mudah dibaca dan ditulis, sehingga sangat cocok digunakan dalam bidang pendidikan, pengembangan aplikasi web, analisis data, kecerdasan buatan, dan pengolahan citra digital (Ryabtsev, 2024).

Salah satu keunggulan Python adalah ekosistem library yang sangat luas, seperti NumPy untuk komputasi numerik, OpenCV untuk pengolahan citra, hingga Tensorflow dan Pytorch untuk *deeplearning*. Kemudahan integrasi antar pustaka dan sintaks yang bersih menjadikan Python pilihan utama dalam berbagai proyek riset dan industri teknologi. Dalam penelitian (Ryabtsev, 2024), Python kini telah menjadi alat yang sangat berpengaruh dalam pengembangan sistem berbasis kecerdasan buatan dan telah diterapkan dalam berbagai bidang seperti medis, keuangan, dan otomasi industri (Guido van rossum, 2025).

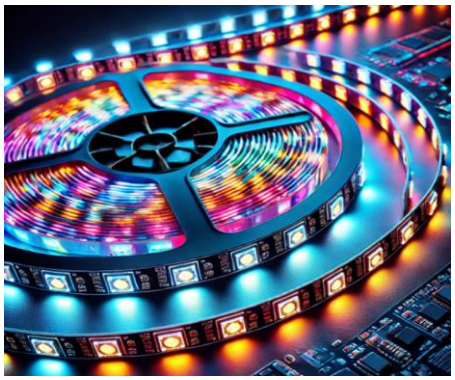
2.2.6 *Visual Studio Code* (Vscode)

Gambar 2.10 *Visual Studio Code* (Vscode)
(Sumber: (Erich Gamma, 2025)

Visual Studio Code (VS Code) adalah code editor gratis dan lintas platform yang dikembangkan oleh Microsoft, dirancang untuk memberikan pengalaman pengembangan ringkas namun kaya akan fitur. VS Code mendukung berbagai bahasa pemrograman seperti Python, JavaScript, C++, Java dan lainnya melalui sistem ekstensi yang fleksibel. Editor ini juga dilengkapi fitur-fitur canggih seperti *syntax highlightning* , *code completion, debugging , terminal terintegrasi,* serta dengan Git dan sistem kontrol versi lainnya (Tan, Chen and Jiao, 2024).

Visual Studio Code sering dipilih sebagai editor utama dalam pendidikan pengembangan perangkat lunak modern karena tampilannya yang ringan, dukungan lintas platform , dan mudah untuk digunakan. Dalam studi oleh Tan, Chen, & Jiao (2023), penggunaan VS Code pada kursus pemrograman dasar memperlihatkan bahwa mahasiswa merasa terbantu secara signifikan dalam mempelajari Python, berkat panduan modular dan integrasi plugin yang mengarahkan langsung ke praktik industri. Notebook yang interaktif dan struktur tutorial yang disusun dalam VS Code terbukti meningkatkan motivasi belajar mahasiswa dibanding editor konvensional (Tan, Chen and Jiao, 2024).

2.2.7 Pencahayaan Objek *LED*



Gambar 2.11 Lampu *strip LED white*
(Sumber: (Maida, 2024)

Lampu *LED* (*Light Emitting Diode*) menjadi salah satu sumber pencahayaan yang paling sering digunakan dalam sistem deteksi objek berbasis kamera, terutama karena efisiensi, umur Panjang, dan kestabilan intensitas cahaya yang dihasilkannya. Dalam pengolahan citra digital, penggunaan pencahayaan buatan seperti *LED* sangat penting untuk memastikan bahwa objek yang diamati terlihat jelas dan konsisten, terutama di lingkungan yang pencahayaannya berubah-ubah atau kurang mendukung, seperti ruangan gelap atau kondisi luar ruangan saat malam hari.

Salah satu penelitian relevan oleh Akış & Dişlitaş (2024) mengusulkan sistem pencahayaan cerdas berbasis *LED* yang diatur oleh algoritma pemrosesan citra warna. Mereka menunjukkan bahwa sistem *LED* pintar yang dikendalikan oleh parameter warna referensi (RGBY) mampu membantu deteksi dan posisi objek secara presisi, sambil mengurangi konsumsi energi secara signifikan bahkan pada iluminasi hanya sekitar 50 lux sistem tetap mampu mengenali objek secara akurat (Akış and Dişlitaş, 2024).

2.2.8 Biji Jagung (*Corn Kernels*)

Gambar 2.12 Biji jagung
(Sumber: (Admin, 2024)

Karakteristik biji jagung seperti ukuran, bentuk, dan densitas, sangat penting dalam penentuan kualitas benih serta proses klasifikasi otomatis. Parameter seperti diameter rata-rata, sphericity, dan bulk density sering diukur untuk menilai kualitas biji dan memprediksi daya tanam. Studi oleh Singh et al. (2017) menemukan bahwa nilai diameter rata-rata biji berkisar antara 7,0–7,7 mm, sphericity sekitar 0,75–0,79, serta bulk density mencapai 733–750 kg/m³, yang memberikan dasar penting dalam desain sistem pendeteksi fisik menggunakan citra digital (Brar *et al.*, 2017).

Akurasi klasifikasi citra untuk membedakan biji jagung baik versus rusak sangat bergantung pada fitur visual yang menonjol: seperti tekstur permukaan, warna—terutama variasi kuning dan belang—serta kondisi fisik seperti retak atau permukaan tidak rata. Penelitian oleh menggunakan analisis komponen utama (PCA) menunjukkan bahwa fitur-fitur seperti moisture content, angle of repose, dan ukuran partikel secara signifikan mempengaruhi penilaian kualitas benih, serta dapat direpresentasikan sebagai parameter deteksi visual dalam sistem otomatisasi berbasis citra (Tang *et al.*, 2021).

2.2.8.1 Biji Jagung Berdasarkan Warna

Gambar 2.13 Perbandingan biji jagung buruk/baik
(Sumber: (Dayat, 2025))

Kualitas biji jagung dapat dinilai secara visual melalui warna yang direpresentasikan dalam ruang warna RGB (*red-green-blue*). Biji jagung yang sehat atau berkualitas baik umumnya memiliki warna yang lebih cerah, seperti dominasi kuning cerah, sedangkan biji yang rusak atau terserang penyakit menunjukkan perubahan warna berupa kehitaman, bercak gelap, atau tampak lebih kusam. Perbedaan karakteristik warna tersebut dapat diterjemahkan menjadi perbedaan nilai intensitas pada masing-masing kanal merah (*red*), hijau (*green*), dan biru (*blue*), sehingga informasi warna dapat dimanfaatkan sebagai salah satu parameter dalam proses identifikasi kualitas biji jagung secara otomatis.

Dengan memanfaatkan nilai RGB, sistem pengolahan citra mampu mengubah karakteristik visual menjadi data numerik yang dapat dianalisis lebih lanjut untuk membedakan kondisi biji jagung.Studi oleh (Pawening, Zaskiya and Hasanah, 2024) menggunakan fitur warna RGB (nilai intensitas ketiga warna) dan tekstur GLCM berhasil mengklasifikasikan kualitas biji jagung (baik atau buruk) dengan akurasi hingga 75% menggunakan dataset 150 gambar, membuktikan bahwa distribusi nilai R,G,dan B menjadin indikator visual yang signifikan dalam membedakan kualitas biji jagung (Pawening, Zaskiya and Hasanah, 2024).

2.2.8.2 Biji Jagung Berdasarkan Tekstur

Gambar 2.14 Tekstur biji jagung
(Sumber: (Dayat, 2025)

Tekstur merupakan salah satu fitur penting dalam analisis citra yang digunakan untuk membedakan permukaan atau pola objek termasuk klasifikasi jagung. Tekstur dapat diartikan sebagai distribusi spasial dari intensitas piksel dalam suatu wilayah gambar merepresentasikan karakter visual permukaan objek, seperti kekasaran, kehalusan, atau kebentukan pola. Dalam konteks biji jagung, tekstur mencerminkan kondisi fisik permukaan biji, apakah licin, kasar, retak, berlubang, atau memiliki bercak akibat kerusakan biologis atau mekanis.

Secara ilmiah, tekstur dapat dijelaskan melalui variasi pola permukaan. Tekstur yang seragam memperlihatkan ketidakteraturan, kontras yang tajam, serta tingkat kekasaran yang lebih tinggi. Variasi ini dapat diukur dengan parameter seperti keragaman, keteraturan, dan homogenitas permukaan. Biji jagung dengan tekstur halus akan menunjukkan nilai homogenitas yang tinggi, sedangkan biji dengan permukaan pecah atau kusam memperlihatkan nilai keragaman yang lebih besar. Oleh karena itu, analisis tekstur menjadi pendekatan yang penting untuk memahami karakteristik biji jagung secara lebih objektif dan mendalam (Chandra and Sembiring, 2024).

2.2.9 Definisi TP, TN, FP, dan FN pada Evaluasi *Model* Deteksi Biji Jagung

Dalam penelitian ini, kinerja *model* YOLOv8-PSO diukur menggunakan empat indikator utama yang berasal dari *confution matrix,* yaitu *True Positive* (TP), *True Negative* (TN), *False Postive* (FP), dan *False Negative* (FN). *Confution Matrix* digunakan untuk membandingkan hasil deteksi *model* dengan kondisi sebenarnya (*ground truth)* pada dataset biji jagung yang telah dilabeli sebelumnya. Metode ini banyak digunakan dalam evaluasi sistem deteksi dan klasifikasi objek karena dapat memberikan gambaran yang jelas mengenai tingkat akurasi dan kesalahan *model* (Hossain et al., 2025).

1. *True Positive* (TP)
   True Positve adalah kondisi dimana *model* berhasil mendeteksi biji jagung dengan benar, baik dari sisi lokasi maupun kelasnya sesuai dengan label. Pada penelitian ini, deteksi dianggap benar jika *Intersection over Union* (IoU) antara kotak prediksi (Ayyad *et al.*, 2025).
2. *True Negative* (TN)
   True Negative terjadi ketika *model* tidak memberikan prediksi pada area yang memang tidak terdapat biji jagung. Walaupun metrik ini jarang digunakan sebagai indikator utama pada deteksi objek, TN tetap berperan dalam menghitung akurasi keseluruhan *model* (Egyed, 2023).

3. *False Positive* (FP)
   False Postive adalah kondisi dimana *model* mendeteksi biji jagung padahal sebenarnya tidak ada, atau memprediksi kelas yang salah. Kesalahan ini dapat mengurangi tingkat presisi sistem (Marsico, 2023).
4. *False Negative* (FN)
   False Negative terjadi ketika *model* gagal mendeteksi biji jagung yang sebenarnya ada. Kesalahan ini biasanya menyebabkan penurunan nilai *recall*, karena semakin banyak objek yang terlewat, semakin rendah kemampuan *model* dalam menemukan semua objek yang relevan.

2.2.9.1 Menentukan TP, TN, FP, dan FN pada Deteksi Biji Jagung

Dalam sistem deteksi objek seperti YOLOv8-PSO, penentuan nilai *True Postive* (TP), *True Negative* (TN), *False Postive* (FN) , dan *False Negative* (FN) dilakukan melalui perbandingan antara hasil prediksi *model* dengan ground truth yang telah dilabeli sebelumnya. Proses ini penting karena menjadi dasar perhitungan metrik evaluasi seperti *precision, recall* , F1-Score, dan *accuracy* yang digunakan untuk mengukur kinerja *model*.

1. Menyiapkan Data Evaluasi
   Ground Truth berisi koordinat bounding box dan kelas objek, serta hasil prediksi *model* yang memuat Lokasi, kelas, dan *confidence score*
2. Menentukan ambang batas
   Prediksi dianggap benar jika nilai IoU memenuhi syarat, dan *confidence score*.
3. Pencocokan
   Prediksi dengan kelas benar dan IoU > *threshold* dihitung sebagai TP, prediksi yang salah kelas atau IoU *threshold* sebagai FP, objek *ground truth* yang tidak terdeteksi menjadi FN. area kosong yang diabaikan dihitung sebagai TN.
4. Perhitungan Otomatis
   YOLOv8 secara otomatis menghitung TP, FP, FN saat evaluasi dan menurunkannya menjadi metrik performa seperti *precision* dan *mAP*.

2.2.9.2 Rumus Evaluasi Berbasis TP, TN, FP, dan FN pada Klasifikasi Biji Jagung

Evaluasi performa deteksi dan klasifikasi biji jagung dilakukan dengan menghitung metrik *precision*, *recall*, *F1-Score*, dan *accuracy*. Semua metrik ini dihitung berdasarkan nilai TP, TN, FP, dan FN yang diperoleh dari *confution matrix*. Pendekatan ini umum digunakan dalam penelitian deteksi objek modern, termasuk di bidang pertanian presisi (Hossain et al., 2025)/

1. *Precision*
   Precision mengukur tingkat ketepatan prediksi positif *model*. Nilai *precision* yang tinggi menunjukkan bahwa sebagian besar deteksi yang dilakukan *model*, memang benar.
   $$Precision = \frac{TP}{TP+FP} \quad (2.8)$$
2. *Recall*
   Recall mengukur kemampuan *model* dalam menemukan semua objek positif yang sebenarnya ada. Nilai *recall* tinggi berarti *model* jarang melewatkan objek yang relevan.
   $$Recall = \frac{TP}{TP+FN} \quad (2.9)$$
3. F1-*Score*
   F1-*Score* merupakan rata-rata dari *precision* dan *recall*. Metrik ini berguna saat dibutuhkan keseimbangan antara menghindari (FP dan FN) (Marsico, 2023).
   $$F1 = 2\ x\ \frac{Precision\ x\ Recall}{Precision+Recall} \quad (2.10)$$

Contoh:
Jika *precision* = 0.90 dan *recall* = 0.85, maka F1 = 2 x (0.90 x 0.85) / (0.90 + 0.85) = 0.875

4. *Accuracy*
   Accuracy mengukur proporsi prediksi benar (baik positif maupun negatif) terhadap seluruh prediksi yang dilakukan.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.11)$$

2.2.9.3 Mencari True Postive, True Negative, False Positive, False Negative

Pada deteksi objek, nilai-nilai ini diperoleh dengan membandingkan *bounding box* prediksi dengan *bounding box ground truth* menggunakan metrik *Intersection Over Union* (IoU) dan kelas objek. Nilai IoU digunakan untuk mengukur seberapa besar tumpang tindih antara area prediksi dan area sebenarnya, sedangkan label kelas memastikan kesesuaian kategori objek (misalnya jagung baik atau jagung buruk).

1. *True Postive* (TP)
   Prediksi benar jika *bounding box* memiliki
   - IoU > *threshold*
   - Kelas prediksi sama dengan kelas *ground truth*
   - Artinya *model* berhasil mendeteksi objek dengan lokasi dan kelas yang tepat.

   Rumus:

$$TP = \sum_{i=1}^{n}[IoU(Bgt, B\,pred\,) \geq Tiou \wedge Cgt = Cpred) \quad (2.12)$$

   Penjelasan:
   - n: Jumlah prediksi
   - B_{gt}: *Bounding Box ground truth* (label sebenarnya)
   - B_{pred}: *Bounding Box* prediksi *model*
   - IoU (B_{gt}, B_{pred}): nilai IoU antara prediksi dan ground truth
   - Tiou: Ambang batas IoU (misalnya 0.5) yang menentukan prediksi
   - C_{gt}: Kelas objek sebenarnya (*ground truth class*)
   - ∧: Logika AND
2. *True Negative* (TN)
   Area yang tidak mengandung objek dan *model* juga tidak membuat prediksi pada area tersebut. Pada deteksi objek, TN jarang digunakan karena sulit mendefinisikan seluruh area negative.
   Rumus:

$$TN = Total\ Area\ Negatif - FP \quad (2.13)$$

3. *False Postive* (FP)
   Prediksi salah jika:
   - IoU < *threshold* atau
   - Kelas prediksi berbeda dari *ground truth*
   - Kesalahan ini menunjukkan *model* mendeteksi sesuatu yang sebenarnya tidak ada atau salah kelas.

   Rumus:

$$FP = \sum_{i=1}^{n}[IoU\ (Bgt, Bpred) < Tiou \vee\ Cgt \neq Cpred] \quad (2.14)$$

   Penjelasan:
   - n: Jumlah prediksi
   - B_{gt}: *Bounding box ground truth* (label sebenarnya)
   - B_{pred}: *Bounding box* hasil prediksi
   - IoU (B_{gt}, B_{pred}): nilai IoU antara prediksi dan *ground truth*
   - Tiou: ambang batas IoU (misalnya 0.5)

- C_{gt}: kelas objek sebenarnya
- C_{pred}: kelas objek hasil prediksi
- V: logika OR

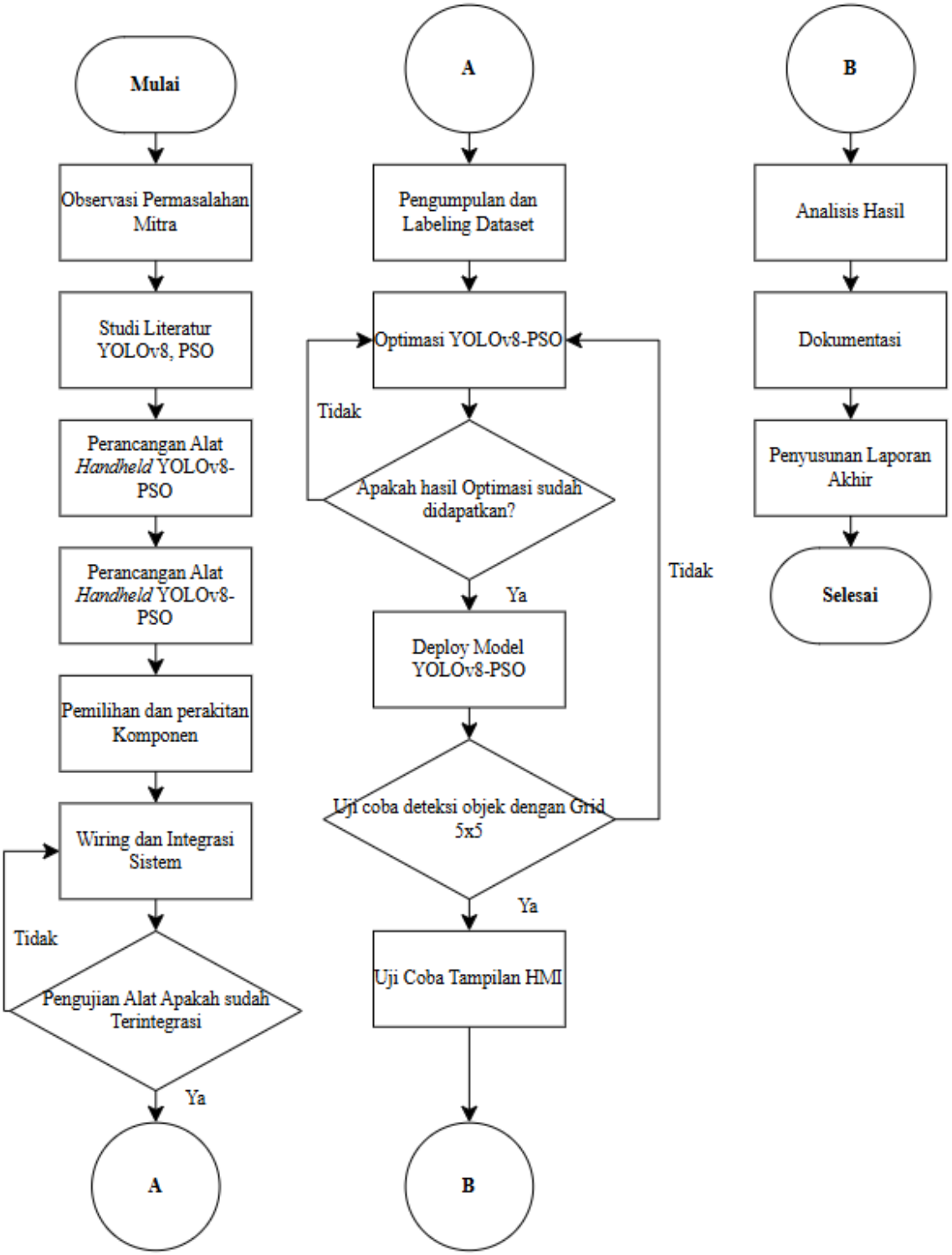
4. *False Negative* (FN)
   Objek yang ada pada *ground truth* tetapi tidak berhasil terdeteksi oleh *model* dengan benar (tidak ada *bounding box* prediksi yang valid).
   Rumus :

$$FN = \sum_{j=1}^{m} [\nexists\, Bpred\ dengan\ IoU\ \geq Tiou\ \wedge\ Cgt = Cpred] \qquad (2.15)$$

   Keterangan *Variable* :
   - m: jumlah *ground truth object* (objek asli dalam dataset)
   - B_{pred}: *bounding box* hasil prediksi
   - IoU: IoU antara prediksi dan *ground truth*
   - Tiou: *treshold* IoU (misalnya 0.5)
   - C_{gt}: kelas objek sebenarnya
   - C_{pred}: kelas hasil prediksi
   - $\nexists$: tidak ditemukan prediksi yang valid

BAB 3 METODOLOGI

Metodologi penelitian bertujuan untuk mengembangkan sistem alat *handheld* klasifikasi biji jagung menggunakan *Image Processing* dengan metode YOLOv8-PSO. Proses pengembangan sistem dilakukan secara bertahap sebagaimana gambar diagram alir metodologi sebagai berikut:



Gambar 3.1 Diagram alir pengerjaan
(Sumber: Dokumen Pribadi)

3.1 Observasi Permasalahan pada Mitra

Observasi dilakukan terhadap proses klasifikasi benih jagung di lingkungan mitra, yaitu PT East West Seed Indonesia (EWINDO), guna memahami alur kerja, keterbatasan, serta kebutuhan riil yang ada di lapangan. Berdasarkan hasil observasi dan diskusi bersama operator lapangan, ditemukan bahwa proses klasifikasi biji jagung masih dilakukan manual dengan cara memilah – milah perbiji.

Metode manual ini tidak menjamin konsistensi hasil klasifikasi terutama volume sortir yang meningkat menjelang masa tanam. EWINDO menyampaikan keinginan untuk mengurangi ketergantungan terhadap tenaga kerja manual dan tidak efisien dari segi waktu, serta mencari solusi berbasis teknologi yang dapat membantu meningkatkan akurasi, efisiensi waktu, dan konsistensi hasil klasifikasi.

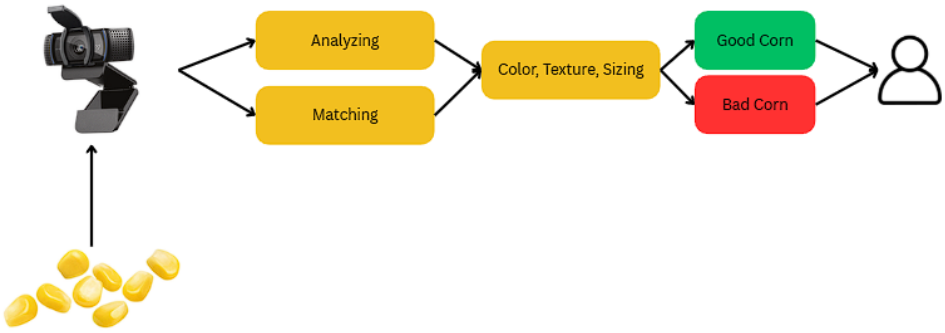
Permasalahan tersebut menjadi alasan peneliti untuk merancang sistem klasifikasi biji jagung secara otomatis yang berbasis *image processing* yang dapat diintegrasikan dalam alat *handheld* dan diharapkan mampu bekerja secara *real-time*, portabel, dan efisien sesuai dengan harapan kondisi kerja dilapangan.

3.2 Studi Literatur

Peneliti melakukan studi literatur untuk memahami teori dan pendekatan yang relevan dengan pengembangan sistem alat *handheld* yang akan digunakan dalam klasifikasi biji jagung. Studi literatur ini menggunakan metode YOLOv8-PSO. Dalam prosesnya penelitian ini menelurusi berbagai jurnal, artikel, dan laporan penelitian terdahulu yang membahas tentang deteksi objek, klasifikasi dalam bidang pertanian maupun industri. Tujuan dari penelitian ini adalah menganalisis metode serta sistem yang dikembangkan dari literatur sebelumnya, sekaligus mengevaluasi kelebihan dan kekurangannya. Hasil dari kajian literatur ini diharapkan sebagai dasar dalam merancang sistem yang lebih optimal dan sesuai dengan kebutuhan penelitian.

3.3 Desain Perancangan *Software* dan *Model*

3.3.1 *Model* Sistem Deteksi



Gambar 3.2 Alur sistem YOLOv8m-PSO
(Sumber: Dokumen Pribadi)

Gambar 3.2 menunjukkan bagaimana nanti sistem klasifikasi biji jagung bekerja. Proses dimulai dari biji jagung yang tertangkap kamera, kamera akan mendeteksi biji jagung lalu

meneruskan ke proses *model*. *Model* akan menganalisa dan menentukan berdasarkan warna, tekstur, dan ukuran biji jagung. Lalu yang terakhir *model* akan menampilkan *bounding box* berwarna hijau yang berarti biji terdeteksi biji baik, dan merah terdeteksi biji buruk.

Hal tersebut akan dianalisis oleh *model* klasifikasi hal ini menggunakan YOLOv8m tanpa PSO dan *model* YOLOv8 yang dibantu optimasi oleh PSO. Kedua *model* ini bertujuan untuk pembanding apakah PSO meningkatkan akurasi deteksi, dan sebagainya. Tabel dibawah ini memuat informasi permasalahan, penyebab, serta solusi yang diambil untuk mengatasi agar sistem dapat bekerja secara optimal.

Tabel 3.1 Kemungkinan permasalahan, penyebab, dan solusi

No	Permasalahan	Penyebab	Solusi
1.	Hasil deteksi kurang jelas	Pencahayaan, terdapat bayangan yang mempengaruhi kualitas deteksi.	Menambahkan lampu *LED* yang merata.
2.	Biji Kecil sulit terdeteksi	*Model* kesulitan membedakan biji kecil dari *background.*	Pembesaran ukuran gambar pada proses pelatihan.
3.	Latihan *model* terlalu lama	Terlalu banuak kombinasi pengaturan yang dicoba.	Menggunakan PSO untuk mencari pengaturan *hypermeter* terbaik.
4.	*Model* sulit membedakan biji baik dan biji buruk	Warna, tekstur, hampir sama.	Memperbanyak data ctra , dan menkonfigurasi ukuran confidence.
5.	Posisi biji *tray* tidak tepat	Biji tidak pas dikotak *tray*, dan sudut kamera kurang tepat.	Menggunakan *ROI cell* untuk konfigurasi ukuran *tray* asli dan *tray* output di HMI.

3.3.2 Perancangan Sistem Deteksi

Sub-bab perancangan sistem deteksi bertujuan untuk memberikan informasi bagaimana *model* YOLOv8m tanpa PSO, dan *model* YOLOv8m-PSO dibuat, dan dievaluasi sehingga tercipta *model* yang akan di integrasikan pada HMI/GUI.

3.3.2.1 Struktur *Layer* YOLOv8m

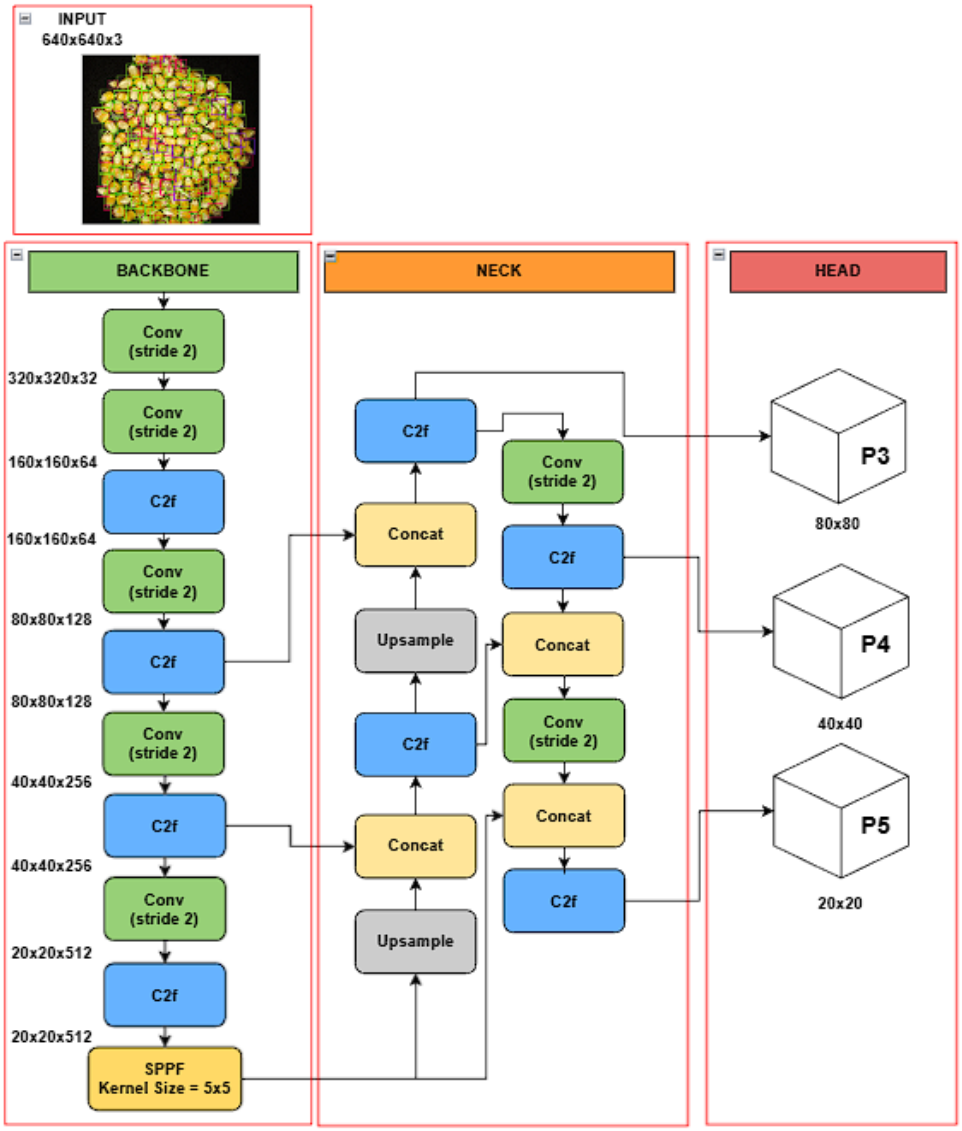
Struktur layer pada YOLOv8m dalam mendeteksi memiliki tiga komponen utama, *backbone*, *neck*, dan *head*. Setiap bagian memiliki rangkaian *layer* yang menjalankan fungsi yang berbeda seperti tabel dibawah ini:

Tabel 3.2 Bagian Struktur YOLOv8m

Bagian Utama	Rangkaian	Fungsi
Backbone	*Conv*	Mengambil ciri – ciri dasar dari gambar, seperti bentuk, tepi, dan pola yang terlihat jelas.
	C2f	Mengolah ciri – ciri tersebut agar lebih detail dan kaya, sehingga *model* bisa mengenali objek dengan baik.
	SPPF	Membantu *model* melihat area yang lebih luas dari gambar tanpa memperlambat proses, sehingga gambar dataset lebih bisa dapat di proses

Bagian Utama	Rangkaian	Fungsi
Neck	*Upsample*	Membesarkan ukuran supaya objek kecil tidak terlewat saat dideteksi.
	Concat	Menggabungkan fitur dari bagian gambar yang berbeda agar informasi yang didapat lebih lengkap
	C2f	Memperbaiki hasil gabungan fitur dan membuatnya lebih jelas sebelum masuk ke proses deteksi
	Conv (sride =2)	Mengcilkan fitur dengan rapi untuk menyesuaikan ukuran saat digabung dengan fitur yang lain.
Head	*Detect Head* P3, P4, P5	Menentukan letak objek, dan sebagainya pada tiga ukuran, yaitu Kecil, Sedang, dan Besar

Dengan memahami fungsi pada setiap rangkaian modul, proses deteksi menjadi lebih jelas dan terlihat bagaimana YOLOv8m mengolah gambar secara efisien sebelum menghasilkan deteksi akhir. Gambar dibawah ini menunjukkan setiap layer, dan komponen didalam nya.



Gambar 3.3 *Layer* model YOLOv8m
(Sumber: Dokumen Pribadi)

Pada Gambar 3.3, secara keseluruhan YOLOv8m murni terdiri dari tiga bagian utama yaitu *backbone*, *neck*, dan *head* yang bekerja berurutan dalam memproses gambar. *Backbone* berfungsi mengekstraksi ciri – ciri penting gambar melalui Conv dan C2f sambil menurunkan resolusi fitur agar informasi visual menjadi lebih padat dan mudah diolah. Selanjutnya, bagian neck menggabungkan fitur dari berbagai skala menggunakan *Upsample*, *Concat*, *Conv Stride 2,* dan C2f sehingga *model* memperoleh representasi yang seimbang antara detail objek kecil maupun pola objek besar. Hasil pengolahan ini kemudian diteruskan kebagian head yang terdiri tiga tingkat deteksi (P3, P4, P5), masing – masing bertugas menghasilkan prediksi lokasi dan ukuran objek pada skala kecil, sedang, dan besar. Dengan alur ini, *model* YOLOv8m mampu melakukan deteksi objek secara cepat, akurat, dan konsisten pada berbagai ukuran.

3.3.2.2 *Dataset*

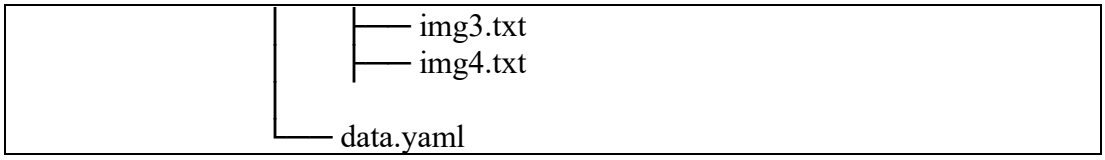
Pada saat melatih *model*, hal pertama yang harus didapatkan adalah dataset berupa contoh gambar jagung baik, dan jagung buruk. Gambar – gambar tersebut dikumpulkan dari berbagai sumber, atau hasil pemotretan langsung dengan variasi pencahayaan yang berbada dan sudut pandang yang berbeda agar *model* dapat mengenali objek dalam kondisi nyata. Setelah itu dilakukan proses anotasi menggunakan website *roboflow* dimana setiap biji jagung diberi kotak pembatas yang disebut (*Bounding Box*) dan diberi label (*good = baik)* dan (*bad = buruk*). Adapun langkah – langkah proses pembuatan dataset sebagai berikut:

1. Mengumpulkan gambar biji jagung dengan 2 kategori yaitu (*good / bad*) dengan variasi sudut dan pencahayaan yang berbeda.
2. Melakukan labeling dan anotasi menggunakan website *Roboflow*
3. Melabeling dengan label *Good* untuk jagung kualitas baik, dan *Bad* untuk jagung berkualitas buruk.
4. Menyimpan dengan format YOLO
   - Format YOLO biasanya berupa Image.txt dengan isi sebagai berikut.

```
0  0.512  0.430  0.300  0.420
1  0.600  0.220  0.100  0.150
```

   Baris 1 (objek jagung kelas 0) adalah jagung baik
   Baris 2 (objek jagung kelas 1) adalah jagung buruk
   - Pastikan struktur folder dataset *model* YOLO seperti ini

```
dataset/
├── images/
│   ├── train/
│   │   ├── img1.jpg
│   │   └── img2.jpg
│   └── val/
│       ├── img3.jpg
│       └── img4.jpg
└── labels/
    ├── train/
    │   ├── img1.txt
    │   └── img2.txt
    └── val/
```

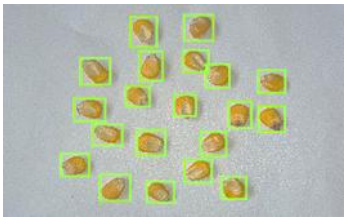
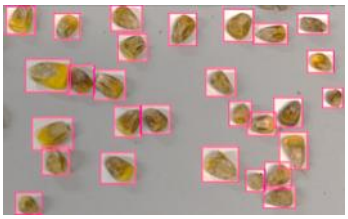


Folder ini berisi gambar dengan format (jpg/png/dsb) dan dibagi menjadi Train/ (untuk melatih data) dan Val/ (untuk memvalidasi data). Umumnya Val/ digunakan untuk evaluasi performa *model* selama pelatihan (*train*)

- Selanjutnya Data.yaml, berisi nc yang berfungsi sebagai jumlah kelas , dan names berfungsi sebagai nama – nama kelas yang sesuai dengan angka di file.txt

5. Hasil anotasi tersebut disimpan sebagai file.txt/gambar dan disertakan dalam dataset.

Tabel 3.3 *Labeling*

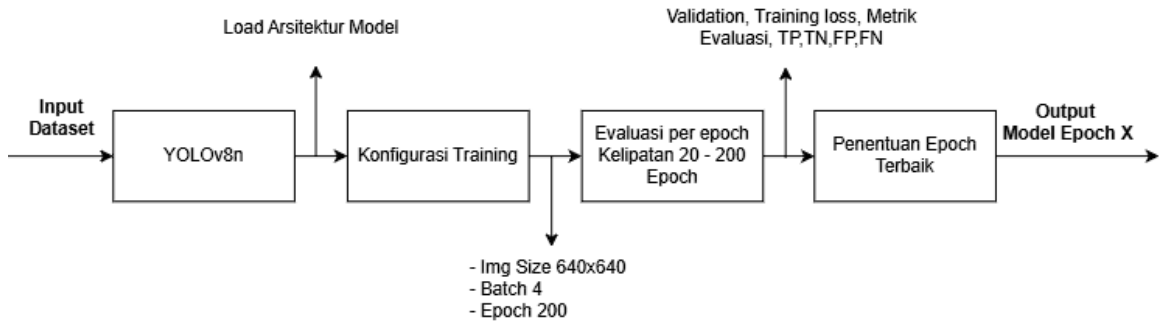
Contoh Label Biji Jagung	***Class***	**Jumlah Gambar Pelatihan (*Train*)**	**Jumlah Gambar Validasi (*Validation*)**	**Jumlah Test (*Testing*)**	**Total**
	*Good*	480	142	103	725
	*Bad*	679	150	148	977
TOTAL		1.159	292	251	1.702

Tabel 3.4 *Stratified Split*

Set	***Good***	***Bad***	**Total**	**Proporsi**
Train Image	480	679	1.159	68.11%
Validation Image	142	150	292	17.15%
Test Image	103	148	251	14.74%
TOTAL	725	977	1.702	100%

Tabel ini menunjukkan pembagian dataset sekunder citra biji jagung menggunakan metode *stratified split,* yang dimana jumlah citra kelas “Baik” dan “Buruk” dibagi secara proporsional kedalam subset train, validation, dan test. Total dataset berjumlah 1.702 citra yang terdiri dari 725 citra kelas baik dan 977 citra kelas buruk. Pembagian proporsi sebesar 68.11% untuk train, 17.15% untuk validasi, dan 17.74% untuk test yang memastikan bahwa distribusi kelas tetap seimbang pada seluruh subtest, dataset ini diambil dari roboflow, maka dari itu jumlah gambar class *good* dan *bad* tidak seimbang.

3.3.2.3 *Train Model* YOLOv8m



Gambar 3.4 *Flowchart Train Model* YOLOv8m tanpa PSO
(Sumber: Dokumen Pribadi)

Diagram diatas menjelaskan alur pelatihan *model* YOLOv8m yang digunakan pada penelitian ini. Proses dimulai dari memasukkan dataset, kemudian arstitekur YOLOv8m dimuat sebagai *model* dasar tanpa PSO. Setelah itu dilakukan konfigurasi training seperti ukuran gambar 640 x 640, *batch size* 4, dan jumlah *epoch* 200. *Model* kemudian dilatih dan setiap 20 *epoch* dilakukan proses evaluasi untuk melihat perubahan nilai *loss*m akurasi, serta metrik seperti TP,TN, FP, FN. Dari hasil evaluasi berkala tersebut, dipilih satu *epoch* dengan performa deteksi terbaik untuk dijadikan *model* pembanding dari *model* yang dioptimasi PSO. Pendekatan ini memastikan bahwa *model* yang digunakan benar – benar memberikan hasil deteksi paling stabil dibandingkan *epoch* lainnya.

Tabel 3.5 *Output Train* YOLOv8m

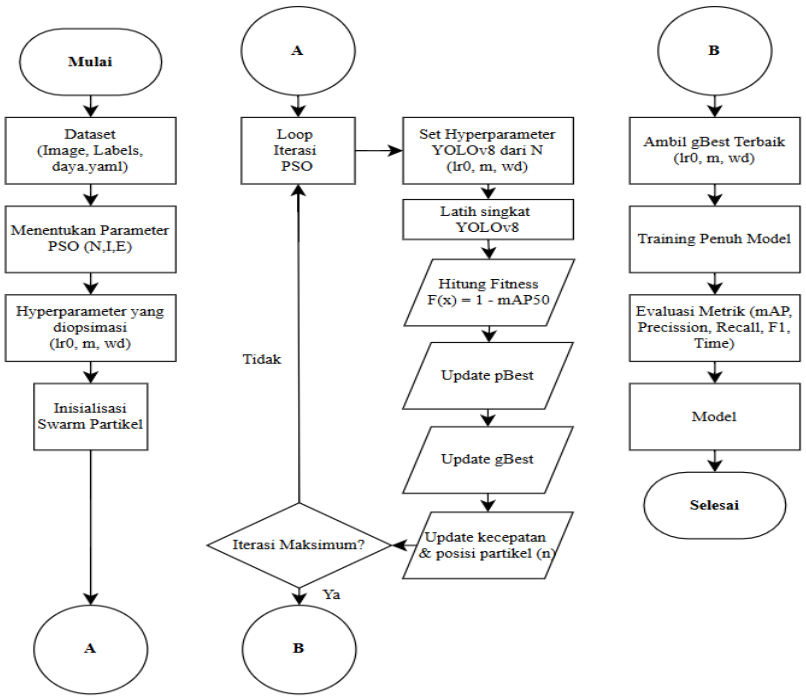
Nama *File* Model per/10 *Epoch*	Tipe *File*
Best.pt	Pytorch
Epoch0.pt	Pytorch
Epoch20.pt	Pytorch
Epoch40.pt	Pytorch
Epoch60.pt	Pytorch
Epoch80.pt	Pytorch
Epoch100.pt	Pytorch
Epoch120.pt	Pytorch
Epoch140.pt	Pytorch
Epoch160.pt	Pytorch
Epoch180.pt	Pytorch
Last.pt (Epoch 200)	Pytorch

Model hasil pelatihan disimpan dalam format *file* .pt (Pytorch) pada setiap *epoch* tertentu, sebagaimana ditunjukkan pada Tabel 3.5. *File Best.pt* merupakan model dengan performa terbaik berdasarkan hasil evaluasi selaam proses pelatihan, sedangkan *file Last.pt* merupakan *model* yang dihasilkan pada *epoch* terakhir (*epoch* ke-200). *Model – model* yang disimpan tersebut selanjutnya digunakan pada tahap evaluasi dan pengujian performa deteksi objek.

3.3.2.4 *Train Model* YOLOv8m + PSO

Particle Swarm Optimization dipakai untuk mencari kombinasi terbaik dari tiga *hyperparameter* penting YOLOv8m seperti *Learning Rate (*lr0), *Momentum* (m), dan *Weight Decay* (w). Setiap iterasi (I) yang berlangsung, PSO mencoba mencari kombinasi yang berbeda,

melatih mode dengan singkat, lalu menilai performa dari pelatihan tersebut menggunakan *mAP50.* Kombinasi paling bagus disimpan sebagai *gBest* (*global best*). Setelah loop selesai, *hyperparameter* terbaik tersebut dipakai untuk melatih YOLOv8m secara penuh hingga *epoch* selesai. *Model* yang dihasilkan itulah yang digunakan untuk deteksi objek.



Gambar 3.5 *Flowchart* proses optimasi YOLOv8m menggunakan PSO
(Sumber: Dokumen Pribadi)

Tabel 3. 6 Konfigurasi *Train* YOLO menggunakan gBest

```
from ultralytics import YOLO
import torch

# KONFIGURASI

DATA = r"D:\TA NEW DESEMBER\Dataset Jagung Rafdan\data.yaml"
MODEL = "yolov8m.pt"
PROJECT_DIR = r"D:\TA NEW DESEMBER\YOLOv8m + PSO\Train Final Menggunakan gBest"
RUN_NAME = "YOLOv8_PSO_Final"

# KONFIGURASI TRAINING

EPOCHS  = 200
IMG_SIZE = 640
BATCH   = 4
WORKERS = 0

# HASIL TERBAIK DARI PSO

LR0         = 0.01
```

MOMENTUM = 0.958917697
WEIGHT_DECAY = 0.000692865

Tabel 3.6 Menunjukkan konfigurasi pelatihan YOLOv8m yang dilakukan menggunakan konfigurasi *hyperparameter* terbaik yang diperoleh dari proses optimasi PSO.

Tabel 3.7 *Output Train* YOLOv8m+PSO

Nama File *Model* per/10 *Epoch*	Tipe File
Best.pt	Pytorch
Epoch0.pt	Pytorch
Epoch20.pt	Pytorch
Epoch40.pt	Pytorch
Epoch60.pt	Pytorch
Epoch80.pt	Pytorch
Epoch100.pt	Pytorch
Epoch120.pt	Pytorch
Epoch140.pt	Pytorch
Epoch160.pt	Pytorch
Epoch180.pt	Pytorch
Last.pt (Epoch 200)	Pytorch

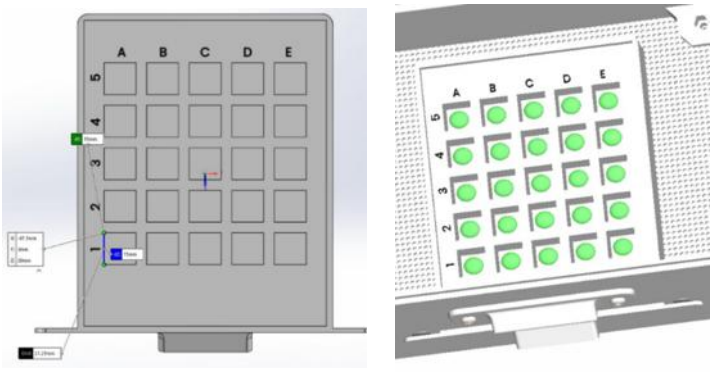
3.3.3 Desain Perancangan *Hardware*

3.3.3.1 Pengukuran Biji Jagung

Pengukuran biji jagung bertujuan sebagai acuan fisik untuk memperkirakan panjang dan lebar biji jagung. Dalam penelitian ini dilakukan dengan memanfaatkan *tray* khusus yang terdiri 25 slot kotak kecil berbentuk grid 5 x 5. Setiap kotak berukuran 1cm x 1cm sehingga menampung satu biji jagung. Kamera dipasang diatas *tray* dan mengambil gambar keseluruhan biji secara bersamaan, kemudian citra hasil tangkapan digunakan sebagai analisis performa *model* dalam mendeteksi dan mengklasifikasi biji jagung.

Dalam literatur, biji jagung umumnya memiliki dimentasi rata – rata sekitar 7-12mm, dengan lebar 4-8mm. Dengan adanya *tray* slot kecil berukuran 1,5cm x 1,5cm per slot ukuran rata – rata tersebut jagung dapat diposisikan dengan rapi untuk memudahkan *model* mendeteksi dan memudahkan peneliti untuk menganalisis.

Penggunaaan struktur *grid* pada *tray* ini sangat krusial untuk meminimalisir terjadinya tumpang tindih antar objek saat proses akuisisi citra berlangsung. Kerapian posisi fisik biji jagung memastikan bahwa setiap butir memliki pembatasan ruang visual (*bounding box*) yang jelas dan terisolasi di dalam gambar. Kondisi pemotretan yang terstruktur ini secara langsung akan mengoptimalkan proses ekstraksi fitur spasial oleh arsitektur *deep learning* yang diterapkan. Pemisahan objek yang tegas akan mengurangi kebingungan *model* terhadap latar belakang objek maupun objek di sekitarnya.



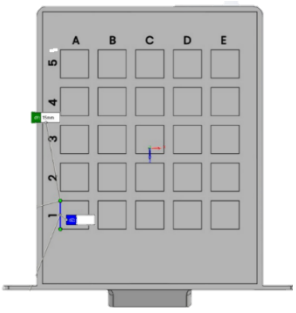
Gambar 3.6 *Tray* tempat biji jagung
(Sumber: Dokumen Pribadi)

3.3.3.2 Pemilihan Komponen

Pemilihan komponen dilakukan untuk pembuatan sistem alat *handheld* klasifikasi biji jagung menggunakan *image processing* berbasis metode YOLOv8-PSO. Setiap komponen memiliki peran penting dan spesifik dalam mendukung fungsi sistem secara keseluruhan.

Tabel 3.8 Kebutuhan komponen penelitian

No.	Nama Komponen	Fungsi	Spesifikasi	Gambar
1.	Laptop Lenovo Ideapad Gaming 3	Berfungsi sebagai pusat kendali penuh seluruh sistem, mulai dari membaca data kamera, menjalankan *model* YOLOv8-PSO, hingga mengatur tampilan hasil klasifikasi pada layar HMI	• Layar 15.6” inch FHD (1980x1080) • Refresh Rate 165hz • AMD Ryzen 7 5000 series • NVIDIA Gforce RTX 3050 • RAM 16gb DDR4 • SSD 512gb • Sistem Operasi Windows 11 • Port USB 3.2 Gen 1, USB 3.2 Gen 1&2.	
2.	Kamera Logitech C920	Berfungsi sebagai sensor yang bertujuan untuk menangkap dan mengirim citra gambar dari *tray*	• Resolusi 1080p/30FPS • Lensa kaca Full HD • FoV 78’ • Koreksi Cahaya Otomatis • Konektivitas Port USB-A • Berat 162g	

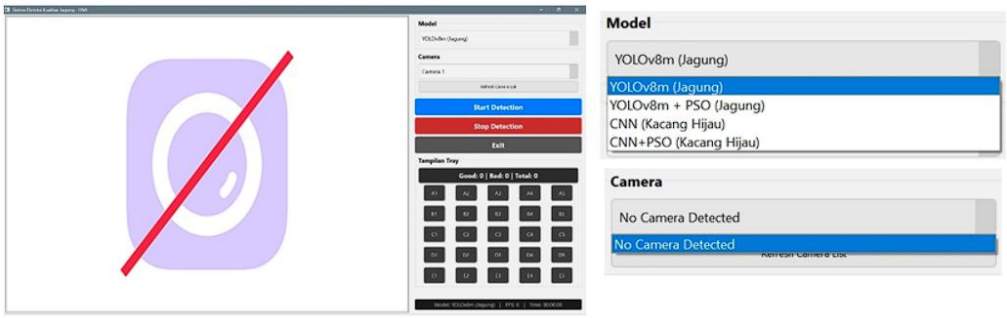
No.	Nama Komponen	Fungsi	Spesifikasi	Gambar
3.	*LED* Strip Flexibel	Berfungsi sebagai suplai cahaya yang seragam dan terkendali yang memastikan akuisisi gambar tidak terganggu oleh bayangan cahaya luar	• Tegangan 6v-12v • Port USB-A • Warna Putih	
4.	Filament	Material yang digunakan untuk cetak print 3D	• Material ABS	
5.	*Tray* Print 3D	Berfungsi untuk menampung jagung dan memberikan penempatan yang rapih agar *model* mendeteksi dengan baik dan HMI terintegrasi dengan baik	• Material Filamen • Diameter 1,75mm / 2.85mm	

3.4 Perancangan Media Pengambilan Data

Perancangan media pengambilan data bertujuan untuk memastikan proses akuisisi data berlangsung secara akurat, efisien, dan mudah digunakan oleh pengguna. Media yang dirancang terdiri dari alat *handheld* sebagai perangkat fisik dan antarmuka grafis (GUI) sebagai sarana interaksi pengguna dan sistem.

3.4.1 Tampilan GUI

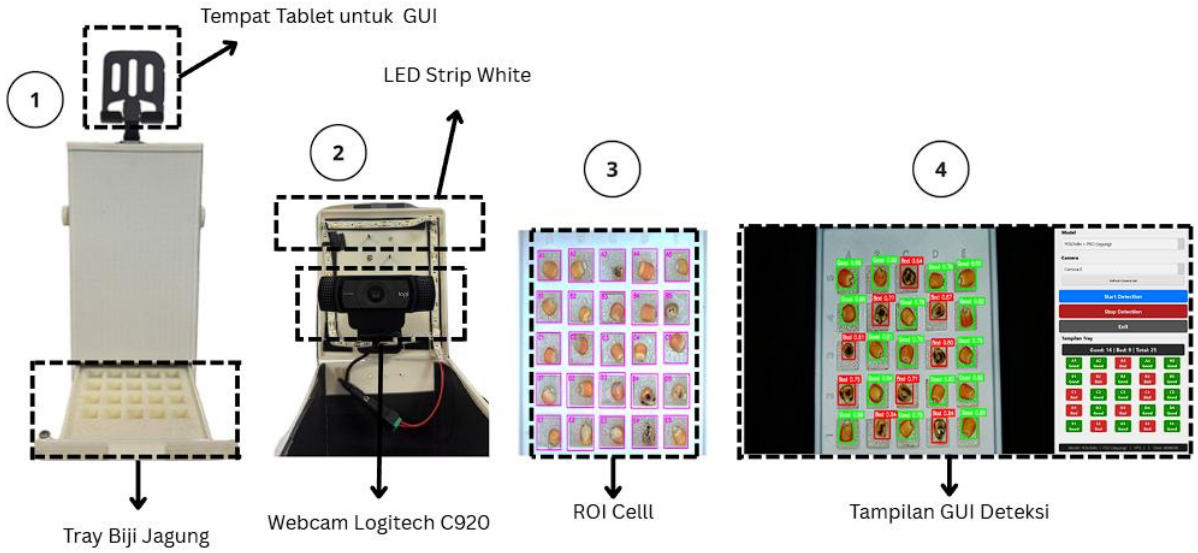
Graphical User Interface (GUI) pada sistem deteksi kualitas jagung dirancang sebagai *Human Machine Interface* (HMI) untuk memudahkan pengguna dalam melakukan pengambilan dan analisis data secara visual. GUI menampilkan area utama berupa citra hasil tangkapan kamera secara *real-time* sebagai media pemantauan objek yang sedang dianalisis.



Gambar 3.7 Tampilan GUI
(Sumber: Dokumen Pribadi)

Pada bagian panel kontrol, pengguna dapat memilih *model* deteksi yang digunakan serta kamera yang terhubung pada sistem, dengan indikator *No Camera Detected* apabila perangkat kamera belum terhubung. Selain itu, GUI dilengkapi dengan tombol pada kontrol utama yaitu *Start Detection, Stop Detection,* dan *Exit* untuk mengatur jalannya sistem. Hasil deteksi ditampilkan dalam bentuk jumlah objek *Good, Bad,* dan *Total* biji jagung yang terdetejsi, serta visualisasi posisi objek pada *tray* yang ditandai oleh *ROI*. Informasi tambahan seperti *model* yang aktif, nilai FPS, dan waktu operasi juga ditampilkan sebagai indikator kinerja sistem selama proses pengambilan data berlangsung.

3.4.2 Persiapan Akuisisi Citra Pada Sistem Deteksi



Gambar 3.8 Klasifikasi citra biji jagung
(Sumber: Dokumen Pribadi)

Persiapan akuisisi citra biji jagung diawali dengan perangkat pada alat *handheld*. Tablet ditempatkan pada bagian atas alat sebagai media tampilan GUI, sedangkan biji jagung diletakkan pada *tray* khusus berbentuk grid yang berada di bagian bawah. *Tray* ini dirancang agar setiap biji berada pada posisi tetap dan tidak saling tumpang tindih, sehingga memudahkan proses pada pengamatan. Kamera webcam Logitech C920 dipasang tepat di atas *tray* dengan posisi tegak lurus untuk memastikan seluruh area *tray* dapat tertangkap dalam satu bidang pandang. Untuk mendukung kualitas pada pendeteksian, digunakan *LED* strip berwarna putih sebagai sumber pencahayaan utama agar kondisi pencahayaan merata dan bayangan dapat diminimalkan.

Citra yang dihasilkan oleh kamera kemudian diproses dengan pembagian *Region Of Interest* (*ROI*) berbentuk sel yang menyesuaikan dengan posisi kotak pada *tray*. Setiap sel *ROI* merepresentasikan satu posisi biji jagung sehingga memudahkan sistem dalam menentukan lokasi dan hasil deteksi. Hasil proses deteksi dan klasifikasi kualitas biji jagung selanjutnya ditampilkan pada antarmuka GUI, lengkap dengan penandaan posisi serta kategori kualitas biji. Dengan konfigurasi perangkat dan alur akuisisi citra yang terstruktur, sistem dapat bekerja secara konsisten dan membantu proses deteksi biji jagung secara lebih akurat.

3.5 Pengambilan Data

3.5.1 Definisi *True Posive*, *True Negative*, *False Positive*, dan *False Negative*

Definisi dari *True Posive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) ditunjukkan pada Tabel 3.9.

Tabel 3.9 Definisi TP, TN, FP, dan FN

Metrik	Keterangan
True Positive (TP)	Kondisi ketika *model* berhasil mendeteksi biji jagung dengan benar, baik dari segi lokasi objek, maupun kelas pada objek.
True Negative (TN)	Kondisi ketika *model* tidak memberikan prediksi pada area yang memang tidak mengandung objek biji jagung. Meskipun metrik ini jarang digunakan sebagai indikator utama dalam deteksi objek, TN tetap berperan sebagai perhitungan akurasi.
False Positive (FP)	Kondisi ketika *model* mendeteksi biji jagung pada area yang sebenarnya tidak mengandung objek atau memprediksi kelas yang salah (*good = bad, bad = good).* Kesalahan ini dapat berpengaruh menurunkan tingkat *precision* sistem deteksi.
False Negative (FN)	Kondisi ketika *model* gagal mendeteksi biji jagung yang sebenarnya ada pada citra. Nilai FN yang tinggi menunjukkan rendahnya kemampuan model dalam menemukan seluruh objek yang relevan dan berdampak pada penurunan nilai *recall.*

3.5.2 Evaluasi *Model* YOLOv8m

Pada saat train *model* YOLOv8m tanpa optimasi PSO, dilakukan evaluasi per-*epoch* untuk menentukan *epoch* ke-x yang mampu untuk mendeteksi biji jagung secara akurat melalui tabel atau grafik dibawah ini.

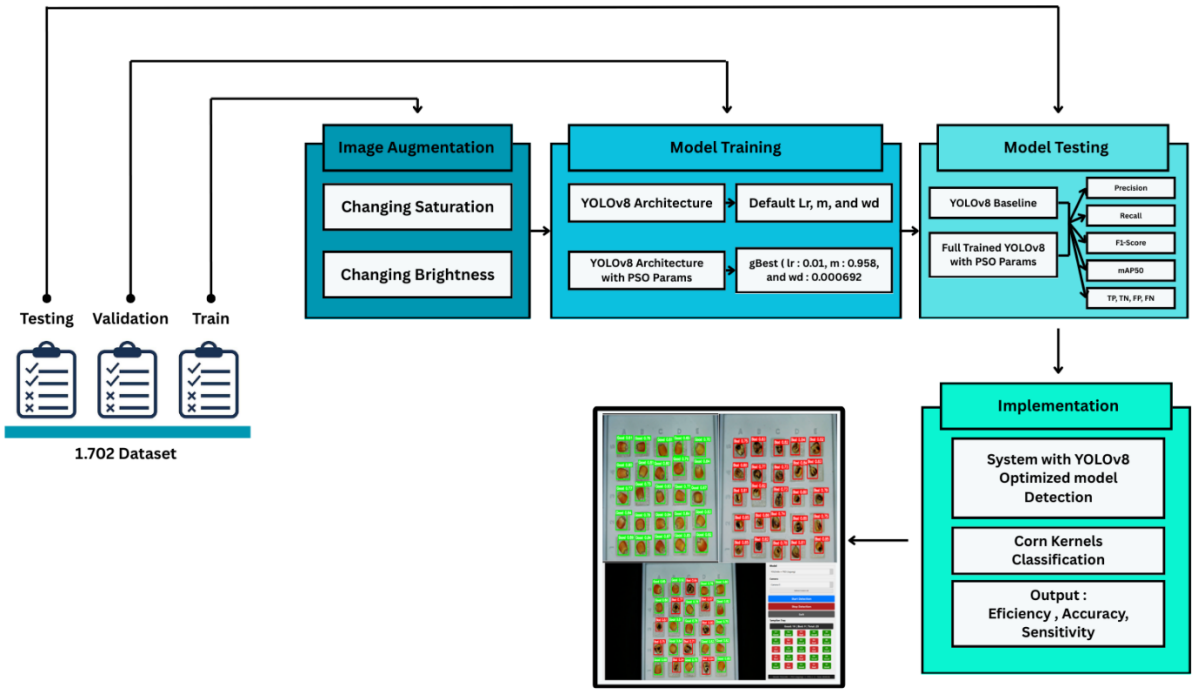
Tabel 3.10 Konfigurasi awal

Jumlah *Epoch*	200
Batch Size	4
Ukuran Gambar	640 x 640 Piksel
Learning Rate	0.01
Optimizer	SGD • *Learning Rate*: 0.01 (*args.yaml*) • *Momentum:* 0.937 (*args.yaml*) • *Weight Decay*: 0.0005 (*args.yaml*)
Environment	Visual Studio Code, Python 3.12, GPU NVIDIA RTX 3050

Tabel 3.10 Konfigurasi awal bertujuan untuk menghasilkan *model* YOLOv8m sebagai *model* pembanding *model* YOLOv8m yang dioptimasi PSO. Selama pelatihan, akan secara otomatis menampilkan metrik – metrik yang dihasilkan dari pelatihan dan digunakan untuk evaluasi selanjutnya. Grafik Val.Loss, dan Train.Loss, bertujuan untuk mengevaluasi performa *model* dalam masa pelatihan dari *epoch* 0 – X dengan menyimpan hasil per-20 *epoch*, kelipatan 20 *epoch* dipilih untuk efisiensi waktu. Fungsi dari *Validation Loss* dan *Training Loss* adalah untuk mengetahui *model* dalam mempelajari pola dari dataset selama pelatihan. *Training Loss* sebagai petunjuk bagaimana tingkat kesalahan *model* pada data training. *Validation Loss* sebagai petunjuk tingkat seberapa baik *model* bisa menebak data baru yang belum pernah terlihat sebelumnya. Tabel ini digunakan untuk melihat konsistensi performa pada *model* secara kuantitatif pada seluruh data evaluasi, sehingga tidak bergantung pada skenario pengujian *model.*

Gambar grafik bertujuan untuk evaluasi performa *model* dengan memonitor perubahan metrik seteksi seperti TP, FP, dan FN pada setiap *epoch* pelatihan. Pola ketiga metrik tersebut digunakan untuk mengidentifikasi fase pembelajaran *model*, yaitu fase *underfitting*, kondisi stabil, dan fase *overfitting*. Tabel ini bertujuan untuk mengidentifikasi perubahan metrik dari 3 *epoch* terpilih , titik performa paling seimbang dapat diidentifikasi. *Epoch* dengan peningkatan TP yang konsisten, penurunan FP dan FN, serta stabilitas metrik deteksi dipilih sebagai *epoch final* yang akan digunakan sebagai *model* pembanding pada bab 4.

3.5.3 Optimasi *Hyperparameter Model* YOLOv8m + PSO



Gambar 3.9 Proses pembuatan *model* YOLOv8m , dan YOLOv8m+PSO
(Sumber: Dokumen Pribadi)

Optimasi *hyperparameter* merupakan tahap penting untuk memperoleh konfigurasi *model* yang menghasilkan kinerja deteksi dan klasifikasi terbaik. Pada penelitian ini, metode YOLOv8m dioptimasi menggunakan algoritma *Particle Swarm Optimization* (PSO) untuk menemukan kombinasi *hyperparameter* pelatihan yang optimal. PSO dipilih karena mampu

melakukan pencarian solusi yang eifisien, serta telah terbukti efektif pada optimasi *model* berbasis *Deep Learning*.

Tujuan dari tahap ini nememukan kombinasi *hyperparameter* terbaik (*image size*, *batch*, *learning rate*, *momentum*, dan *weight decay*) yang memberikan performa evaluasi awal terbaik berdasarkan metrik (*Precision*, *Recall*, F1-Score dan *mAP*50). *Hyperparameter* ini berada pada bagian training pipeline, khususnya pada komponen optimizer, seingga memengaruhi dinamika bobot selama proses pelatihan *model*. Adapun langkah – langkah optimasi untuk memulai proses pencarian *hyperparameter* pada *Particle Swarm Optimization* sebagai berikut :

1. Persiapan Dataset
   - Menggunakan *dataset* citra jagung yang telah dilabeli sesuai format *folder model* YOLO.
2. Menentukan Ruang Pencarian PSO
   - *Learning Rate* = 0.001 – 0.01 berfungsi sebagai besar langkah pembaruan bobot.
   - *Momentum* = 0.80 – 0.98 berfungsi menstabilkan arah gradien agar tidak berfluktuasi.
   - *Weight Decay* = 0.0001 – 0.001 batasan agar *model* tiddak mengalami *overfitting*.
   - *Batch Size* = 2 – 8 jumlah sample per iterasi (disesuaikan tergantung spesifikasi GPU).
   - *Image Size* = 640 x 640 berfungsi sebagai resolusi input dan memengaruhi detail deteksi dan waktu inferensi.

   Ketiga *hyperparameter* inti (*learning rate, weight decay*, dan *momentum*) bekerja pada *optimizer* (SGD) yang digunakan YOLOv8m untuk mempengaruhi bobot *model*.

Tabel 3.11 Konfigurasi *Particle Swarm Optimization*

Parameter	Simbol	Rentang Nilai	Keterangan
Jumlah Partikel	N	5	Jumlah dari solusi yang diuji pada setiap iterasi.
Jumlah Iterasi	I	5	Jumlah sikluis pembaruan posisi untuk partikel.
Epoch per Partikel	E	10	Durasi pelatihan (train) YOLOv8 untuk setiap kombinasi parameter.
Learning Rate	lr_0	0.001 – 0.01	Mengatur laju pembelajaran pada *model*.
Momentum	m	0.90 – 0.98	Mengontrol keteraturan pembelajaran *model* untuk mencegah *overfitting*.
Weight Decay	wd	5e-5 – 1e-3	Mengatur pengaruh kecepatan sebelumnya terhadap posisi baru.
Koefisien Inersia	w	0.4 – 0.9	Mengatur pengaruh kecepatan sebelumnya terhadap posisi baru.
Koefisien Kognitif	C_1	2.0	Bobot pengaruh posisi terbaik individu (*pbest)*.
Koefisien Sosial	C_2	2.0	Bobot pengaruh posisi terbaik global (*gbest*).
Fitness	f(x)	1 – *mAP*50	Digunakan untuk menilai kualitas kombinasi parameter.

Rentang *learning rate*, *momentum*, dan *weight decay* dipilih mengikuti batas aman yang biasa digunakan saat melatih *model* YOLO. *Learning rate* dengan rentang 0.001 – 0.1 dianggap stabil. *Momentum* dibuat 0.05 – 1.0 agar PSO mecoba variasi pergerakan

gradien dari pelan ke stabil. *Weight decay* berada pada 0.0025 – 0.001 untuk mencegah *overfitting*. Sementara parameter PSO seperti inertia weight (0.4 – 0.9) dan koefisien kognitif (C_1 dan C_2 = 2) mengikuti standar dari penelitian PSO yang terbukti stabil sejak awal pengembangannya.

3. Proses PSO
   - PSO menghasilkan kombinasi *hyperparameter* acak untuk setiap partikel pada iterasi pertama.
   - Setiap partikel mewakili satu kombinasi dari *Learning Rate*, *momentum*, *weight decay*, *batch size*, dan *image size*.
   - *Model* dilatih singkat dengan mini *epoch* untuk mendapatkan nilai metrik awal.
   - Hasil metrik tersebut digunakan untuk menghitung nilai *firness*.
   - Pada tiap iterasi, PSO memperbarui posisi partikel menggunakan *pBest* (posisi terbaik individu) dan *gBest* (posisi terbaik seluruh *swarm*).
4. Mekanisme PSO

   Setiap partikel merepresentasikan satu kombinasi *hyperparameter* YOLOv8m. Posisi partikel menunjukkan nilai *hyperparameter*, sedangkan kecepatan partikel menentukan arah perubahan posisi pada iterasi berikutnya. Pembaruan pada kecepatan dan posisi partikel dilakukan dengan menggunakan persamaan berikut :

$$v_i^{t+i} + c1 \cdot r1(pBest_i - x_i^t) + c2 \cdot r2(gBest - x_i^t) \quad (3.1)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (3.2)$$

   Dimana v_i merupakan kecepatan partikel ke-i, x_i merupakan posisi partikel (kombinasi *hyperparameter*), w adalah koefisien inersia, c_2 merupakan koefisien kognitif dan sosial, serta r_1 dan r_2 adalah bilangan acak antara 0 dan 1. Nilai *pBest* menunjukkan posisi terbaik yang pernah dicapai partikel, sedangkan *gBest* menunjukkan posisi terbaik dari seluruh partikel. Setelah posisi diperbarui, nilai *hyperparameter* dibatasi agar tetap berada pada rentang pencarian yang telah ditentukan.
5. Pencatatan Nilai Evaluasi
   Setelah iterasi selesai dilakukan pencatatan nilai metrik.
6. Setelah *hyperparameter* terbaik dari PSO ditemukan, selanjutnya melakukan *train final* dengan *momentum* x, *weight decay* x, dan *learning rate* x.

Adapun keterangan dan fungsi setiap *hyperparameter* sebagai berikut:

1. *Learning Rate* = mengatur besarnya langkah dari pembaruan bobot dan mempengaruhi *model* agar cepat untuk konvergensi
2. *Momentum* = menstabilkan gradien agar pembaruan bobot tidak acak
3. *Weight Decay* = mencegah bobot menjadi terlalu besar (regulasi)
4. *Batch Size* = menentukan jumlah data per-iterasi dan mempengaruhi stabilitas dan kecepatan pada saat training
5. *Image Size* = menentukan resolusi input dan detail fitur yang dapat diekstraksi
6. *Precision* = berfungsi mengetahui seberapa tepat *model* mendeteksi objek benar (minim FP)
7. *Recall* = berfungsi mengetahui seberapa lengkap *model* dalam mendeteksi objek (minim FN)
8. F1-*Score* = berfungsi mengetahui keseimbangan antara *precision* dan *recall*
9. *mAP* = rata – rata akurasi deteksi seluruh kelas
10. *Fitness* = nilai objektif PSO untuk menentukan *hyperparameter* terbaik

3.5.4 Rumus Metrik Evaluasi

Rumus berikut merupakan rumus yang digunakan untuk penelitian ini selama evaluasi metrik, stabilitas *model* mendeteksi, dan berhubungan dengan nilai – nilai yang akan dihasilkan dalam penelitian:

- Rumus *Precision*

$$Precision = \frac{TP}{TP+FP} \tag{3.3}$$

Precision berfungsi sebagai prediksi benar paling banyak (sedikit salah deteksi).

- Rumus *Recall*

$$Recall = \frac{TP}{TP+FN} \tag{3.4}$$

Recall berfungsi sebagai objek yang berhasil terdeteksi.
FN = *False Negative* merupakan *model* tidak berhasil mendeteksi objek.

- Rumus F1-*Score*

$$\text{F1} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.5}$$

F1-*Score* berfungsi untuk mengukur tingkat keseimbangan dari *Precision* dan *Recall*.

- Rumus *mAP*50

$$mAP_{50} = \frac{1}{N} \sum_{i=1}^{N} API \ (Iou = 0.50) \tag{3.6}$$

- Rumus *mAP*50-90

$$mAP0.50 - 0.90 = \frac{1}{10} \sum_{IoU \in [0.50,0.55,\ldots,dst]}^{C} mAPIoU \tag{3.7}$$

- Rata – rata conf

$$Conf = \frac{Total\ Conf}{Jumlah\ objek\ yang\ terdeteksi} \tag{3.8}$$

- Akurasi

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.9}$$

3.6 Perbandingan Evaluasi *Model* YOLOv8m dan YOLOv8m+PSO

Pada subbab ini dilakukan perbandingan metrik evaluasi dari YOLOv8m tanpa PSO dan YOLOv8m dengan PSO. Perbandingan ini melibatkan metrik *precision*, *recall*, *F1-Score*, dan *mAP* tertinggi dari setiap *model*. Lalu membandingkan juga dengan metrik evaluasi deteksi dengan nilai TP, TN, FP, FN, *Precision*, *Recall*, *F1-Score*, dan *mAP50*.

3.7 Pengambilan Data Final

Setelah melakukan perbandingan evaluasi *model baseline* dengan *model* optimasi PSO, dilakukan pengambilan *data final* dengan cara menempatkan biji jagung dengan variasi baik seluruh *tray*, buruk seluruh *tray*, dan yang terakhir biji jagung baik dan buruk dicampur dalam *tray* 5 x 5 (25 biji jagung).

3.8 Pembuatan Laporan

Tahap akhir yang dilakukan adalah dengan membuat laporan tentang penelitian ini, sehingga hasil yang ditunjukan dapat menjelaskan keseluruhan proses yang akan dirancang dan dilaksanakan sesuai dengan hasil dan data yang diperoleh. Semua tindakan yang dilakukan akan didokumentasi berupa foto dan video.

Halaman ini sengaja dikosongkan.

BAB 4 HASIL DAN PEMBAHASAN

4.1 Hasil Pelatihan *Model* YOLOv8

Pelatihan *model* YOLOv8 bertujuan untuk memperoleh *model* yang nantinya akan digunakan sebagai pembanding antara YOLOv8 tanpa PSO, dan YOLOv8 yang dioptimasi PSO. Pelatihan *model* YOLOv8 tanpa PSO dilakukan menggunakan arsitektur YOLOv8m dengan konfigurasi dasar sebagaimana ditunjukkan pada Tabel 4.1.

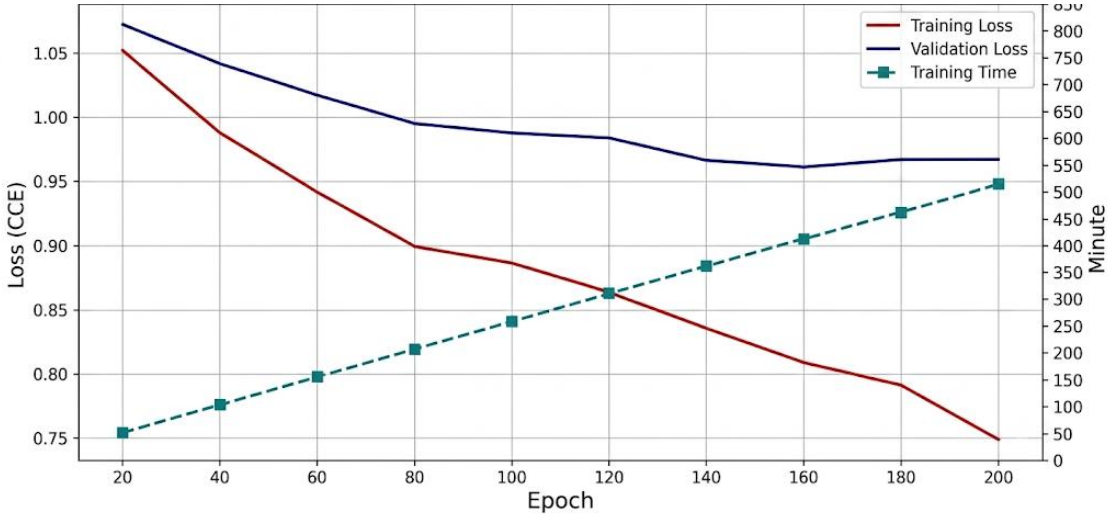
Tabel 4.1 Konfigurasi awal

Jumlah *Epoch*	200
Batch Size	4
Ukuran Gambar	640 x 640 Piksel
Learning Rate	0.01
Optimizer	SGD • *Learning Rate*: 0.01 (*args.yaml*) • *Momentum*: 0.937 (*args.yaml*) • *Weight Decay*: 0.0005 (*args.yaml*)
Environment	Visual Studio Code, Python 3.12, GPU NVIDIA RTX 3050

Selama pelatihan, *model* secara otomatis mencatat metrik evaluasi seperti *box loss, classification loss, precision, recall, F1-Score,* dan *mAP*. Semua hasil ini tersimpan pada file results.csv dan *model* juga akan diuji menggunakan gambar biji jagung baik, dan biji jagung buruk untuk mengetahui performa deteksinya. Selain itu, nilai optimizer merupakan konfigurasi default yang digunakan dalam *baseline*, dan menjadi acuan pembanding sebelum dilakukan proses optimasi menggunakan PSO.

4.1.1 Grafik Training Loss dan Validation Loss

Fungsi dari *Training Loss* dan *Validation Loss* bertujuan untuk mengetahui *model* mempelajari pola dari dataset selama proses pelatihan. *Training Loss* menunjukkan tingkat kesalahan model pada data training, lalu *Validation Loss* menunjukkan tingkat seberapa baik *model* bisa *menebak* data baru yang belum pernah terlihat sebelumnya.



Gambar 4.1 Grafik *Training Loss* dan *Validation Loss model* YOLOv8

Pada Gambar 4.1 menampilkan perubahan nilai *loss* terhadap jumlah *epoch* serta waktu yang dibutuhkan untuk pelatihan *model* YOLOv8m. Pada awal proses pelatihan (*epoch* 20-60), nilai *loss* mengalami penurunan yang drastis, menandakan model masih proses pembelajaran awal. Memasuki *epoch* 80-120, nilai loss mulai melambat dari data pelatihan (*train data)* ataupun validasi *(validation test)* yang menunjukkan bahwa *model* mulai mencapai titik konvergensi dan peningkatan akurasi sudah tidak signifikan. Setelah *epoch* 120 hingga akhir pelatihan, *training loss* masih menurun dan *validation loss* relatif stabil, menandakan bahwa tambahan *epoch* tidak memberikan peningkatan terhadap generalisasi *model*. Sementara waktu pelatihan meningkat hingga mencapai 501 menit (8 jam 36 menit) pada *epoch* ke-200, menunjukkan bahwa durasi tiap *epoch* terhadap proses pelatihan berjalan stabil tanpa gangguan.

4.1.2 Metrik Evaluasi *Model*

Metrik ini digunakan untuk melihat konsistensi performa *model* secara kuantitatif pada seluruh data evaluasi, sehingga tidak hanya bergantung pada skenario pengujian pada *model*.

Tabel 4. 2 Nilai *Precision*, *Recall*, F1-*Score*, dan *mAP Training* YOLOv8

epoch	***precision***	***recall***	***F1-Score***	***mAP*50**	***mAP*50-95**
20	0.86098	0.89236	0.876389	0.92958	0.66197
40	0.89158	0.92	0.905567	0.93825	0.68778
60	0.91657	0.92384	0.920191	0.95545	0.70308
80	0.91305	0.93233	0.922589	0.95342	0.70811
100	0.90782	0.93151	0.919512	0.95515	0.71298
120	**0.92244**	**0.94271**	**0.932465**	**0.95909**	**0.72044**
140	0.92432	0.93435	0.929308	0.95871	0.72454
160	0.92156	0.93827	0.92984	0.95723	0.72467
180	0.92195	0.93994	0.930858	0.95776	0.72277

Berdasarkan Tabel 4.2, diketahui bahwa nilai F1-*Score* tertinggi diperoleh pada *epoch* 120 dengan nilai sebesar 0.9324 yang menunjukkan bahwa pada tahap ini *model* mencapai keseimbangan terbaik antara jumlah deteksi benar dan jumlah kesalahan pada data validasi. Nilai F1-*Score* tertinggi pada *epoch* 120, namun peneliti perlu mempertimbangkan hasil metrik dengan hasil deteksi pada objek nyata. Oleh karena itu, meskipun *epoch* 180 unggul secara metrik, perlu diperhatikan untuk pengujian stabilitas deteksi pada *model.*

4.1.3 Uji Deteksi Gambar *Model* Pada kelipatan 20 *Epoch*

Dilakukan pengujian deteksi langsung terhadap *model* pada setiap kelipatan 20 *epoch* (20, 40, 60, 80, 100, 120, 140, 160, 180, 200). Pengujian ini bertujuan untuk membandingkan performa deteksi nyata dari setiap *model* yang tersimpan, bukan hanya berdasarkan nilai metrik pelatihan.

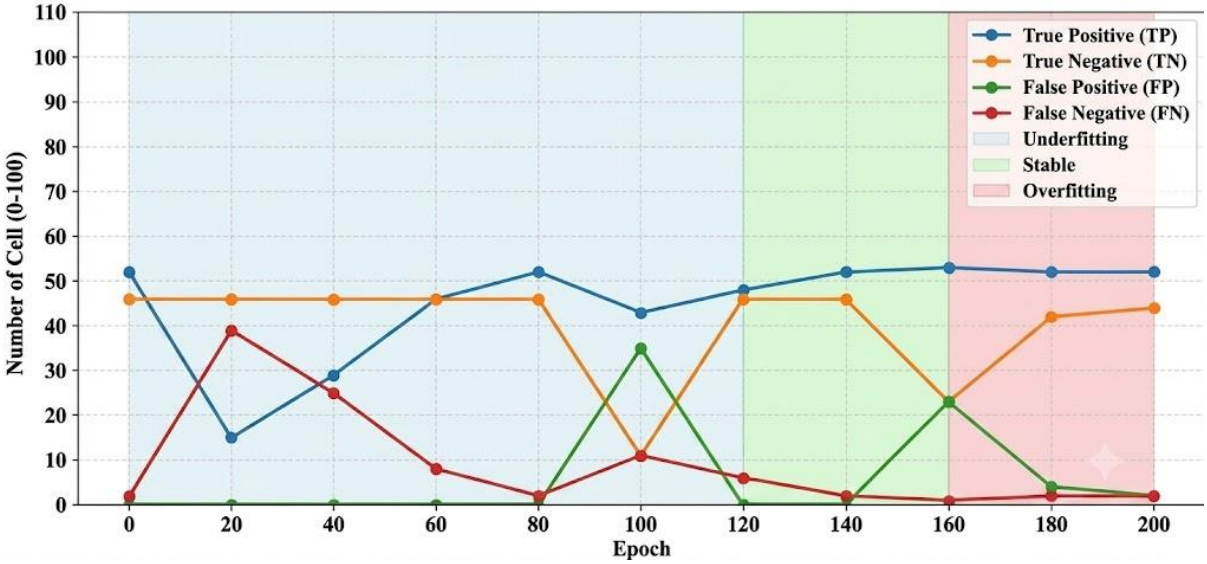
Tabel 4.3 Nilai TP, TN, FP, dan FN pada deteksi gambar

Epoch	**TP**	**TN**	**FP**	**FN**	***Precision***	***Recall***	***F1-Score***	***mAP*50**
0	52	46	0	2	1	0.963	0.981	0.772
20	15	46	0	39	1	0.278	0.435	0.929
40	29	46	0	25	1	0.537	0.699	0.94
60	46	46	0	8	1	0.852	0.92	0.96
80	52	46	0	2	1	0.963	0.981	0.958
100	43	11	35	11	0.551	0.796	0.652	0.958

Epoch	TP	TN	FP	FN	*Precision*	*Recall*	*F1-Score*	*mAP*50
120	48	46	0	6	1	0.889	0.941	0.958
140	**52**	**46**	**0**	**2**	**1**	**0.963**	**0.981**	**0.959**
160	53	23	23	1	0.697	0.981	0.815	0.956
180	52	42	4	2	0.929	0.963	0.945	0.958
200	52	44	2	2	0.963	0.963	0.963	0.954

Berdasarkan Tabel 4.3, terlihat adanya fluktuasi performa yang cukup terlihat pada *epoch* ke-100, dimana nilai *Precision mengalami* penurunan drastis menjadi 0.551. Penurunan tersebut disebabkan oleh jumlah *False Positive* (FP) yang mencapai 35, Kondisi ini mengidentifikasikan bahwa pada fase tersebut, *model* mengalami ketidakstabilan dalam proses belajarnya. Tingginya angka FP menunjukkan bahwa *model* menjadi terlalu agresif, yaitu salah mengklasifikasikan area latar belakang (*background)* atau *noise* sebagai biji jagung. Hal ini wajar terjadi pada pelatihan *model* dengan *optimizer* standar (SGD) tanpa bantuan algoritma optimasi seperti PSO, dimana bobot *model* masih berfluktuasi mencari titik konvergensi yang tepat. Sehingga sempat *miss* sebelum akhirnya stabil.

Namun, ketidakstabilan ini berhasil diperbaiki seiring bertambahnya iterasi pada pelatihan, terbukti pada *epoch* ke-140 *model* kembali menunjukkan performa yang sangat stabil dengan *False Positive* kembali ke angka 0 dan *Precision* mencapai nilai sempurna 1.000. Didukung dengan nilai *Recall* sebesar 0.963 dan *F1-Score* tertinggi di angka 0.981, serta *mAP*50 yang mencapai 0.959, maka *epoch* ke 140 ditetapkan sebagai titik optimal. Oleh karena itu, *model* pada *epoch* 140 yang selanjutnya dipilih sebagai *model baseline* (tanpa PSO) untuk dibandingkan dengan *model* hasil optimasi PSO.



Gambar 4.2 Grafik TP, TN, FP, dan FN pada deteksi objek

Berdasarkan grafik pada gambar 4.2, dapat disimpulkan bahwa *model* meningkat seiring bertambahnya *epoch*, dimana fase *underfitting* terlihat pada *epoch* 0-120 dengan nilai TP yang masih naik turun dan FN yang tinggi. *Model* mulai stabil pada rentang *epoch* 120 – 160 ditandai oleh kenaikan TP hingga mendekati nilai maksimum dan penurunan FN menjadi sangat rendah. Pada puncaknya yaitu *epoch* 140 dengan TP tertinggi dan FN mendekati 0 yang mengidentifikasikan *model* berada pada titik akurasi dan stabil. Setelah *epoch* 160, performa mulai menurun akibat kenaikan FP dan FN yang menunjukkan gejala *overfitting* dan penuruan

kemampuan generalisasi. Secara keseluruhan grafik, *epoch* 140 merupakan *checkpoint* yang stabil untuk digunakan.

4.1.4 Penentuan *Epoch* Terbaik

Untuk memperoleh *epoch* terbaik yang akan digunakan sebagai *model baseline* sebelum dibandingan dengan yang dioptimasi PSO, dilakukan perbandingan antara performa deteksi pada citra uji dan metrik evaluasi hasil pelatihan YOLOv8m. Perbandingan ini dilakukan karena kedua jenis metrik tersebut mempresentasikan dua domain pengujian yang berbeda. Metrik pelatihan mencerminkan rata – rata kinerja *model* terhadap seluruh data validasi, sedangkan metrik inferensi nyata mempresentasikan performa *model* pada kondisi operasional yang menyerupai penerapan sesungguhnya. Oleh karena itu, pemilihan *epoch* terbaik harus mempertimbangkan kedua sudut pandang tersebut secara bersamaan. Ringkasan hasil evaluasi untuk *epoch* kandidat (120, 140, dan 160) ditampilkan pada tabel 4.4.

Tabel 4.4 Perbandingan metrik uji deteksi gambar dan metrik pelatihan YOLOv8

Penilaian	Metrik	*Epoch* 120	*Epoch* 140	*Epoch* 160
Uji Deteksi pada citra gambar	TP	48	52	53
	TN	46	46	23
	FP	0	0	23
	FN	6	2	1
	Precision	1	1	0.697
	Recall	0.889	0.963	0.981
Metrik *Training / Validation* YOLOv8	F1 – *Score*	0.941	0.981	0.815
	Precision	0.92244	0.92432	0.92156
	Recall	0.94271	0.93435	0.93827
	F1 – *Score*	0.932465	0.92930	0.92984
	*mAP*50	0.72044	0.95871	0.95723

Berdasarkan Tabel 4.4 vertikal per-*epoch*, terlihat bahwa performa *model* meningkat dari *epoch* 120 hingga mencapai kondisi optimal pada epoch 140. Pada epoch 120, *model* sudah mampu menghasilkan *precision, recall*, dan F1-*Score* sebesar 1.00 namun belum didukung oleh tingkat *conf* yang tinggi. *Epoch* 100 dan 120 menunjukkan peningkatan dari sisi metrik pelatihan, tetapi hasil uji deteksi gambar masih menghasilkan FN dan FP pada beberapa cell. *Epoch* 140 menjadi yang terbaik karena mencapai TP yang tinggi, FN FP yang rendah serta mempunyai *precision*, *recall*, F1-*Score*, dan *Conf* yang tinggi. Selain itu, metrik *training/validation* pada *epoch* 140 adalah yang tertinggi diantara semua *epoch* yang diuji. Hal ini menegaskan bahwa *epoch* 140 merupakan *model* yang paling akurat, stabil, dan layak untuk dipilih sebagai *model* pembanding dengan yang dioptimasi oleh PSO.

4.2 Penggunaan *Particle Swarm Optimization* (PSO)

4.2.1 Parameter Konfigurasi PSO

Proses optimasi *model* YOLOv8 dilakukan menggunakan algoritma *Particle Swarm Optimization* (PSO). Penerapan PSO bertujuan untuk mencari kombinasi terbaik dari pelatihan *(hyperparameter)* yang secara langsung memengaruhi hasil deteksi, seperti *learning rate* (laju pembelajaran), *momentum,* dan *weight decay.* Ketiga parameter tersebut berperan penting

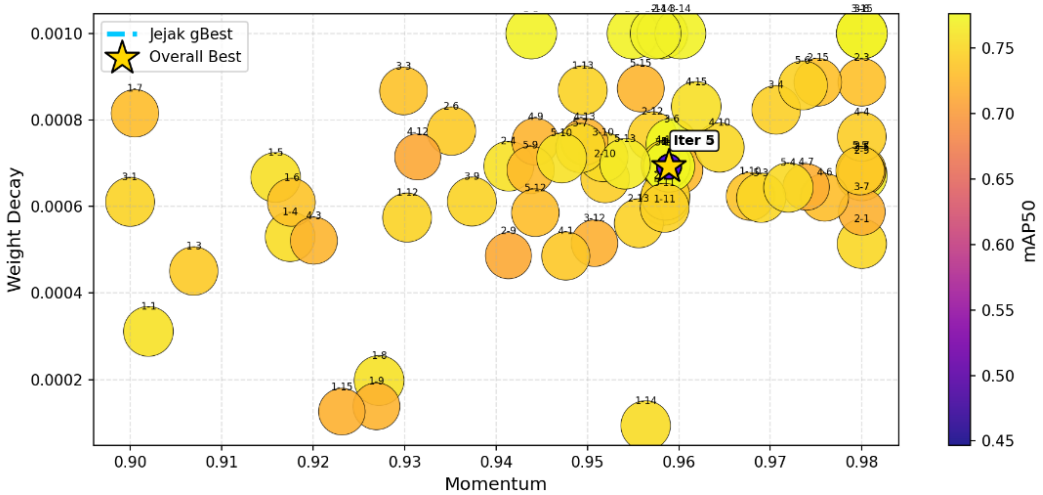
dalam menentukan kecepatan konvergensi *model*, kestabilan proses pelatihan, serta kemampuan *model* dalam melakukan generalisasi terhadap data baru.

Algoritma PSO bekerja dengan prinsip pergerakan sekumpulan partikel yang disebut *swarm* dalam ruang pencarian untuk menemukan solusi yang disebut nilai *fitness*. Setiap partikel mewakili satu kombinasi *hyperparameter,* dan pergerakannya dikontrol oleh dua komponen utama, yaitu pengalaman terbaik dari individu (*global best / gBest)* dan pengalaman terbaik dari kelompok *(partikel best / pBest)*. Dalam setiap iterasi, posisi partikel diperbarui berdasarkan kecepatan sebelumnya, jarak terhadap posisi terbaik secara individu, serta jarak terhadap posisi terbaik secara global, hingga ditemukan konfigurasi yang menghasilkan *fitness* minimum.

Tabel 4.5 Konfigurasi parameter *Particle Swarm Optimization* (PSO)

Parameter	Simbol	Rentang Nilai	Keterangan
Jumlah Partikel	N	15	Jumlah dari solusi yang diuji pada setiap iterasi.
Jumlah Iterasi	I	5	Jumlah sikluis pembaruan posisi untuk partikel
Epoch per Partikel	E	10	Durasi pelatihan (train) YOLOv8 untuk setiap kombinasi parameter.
Learning Rate	**lr_0**	**0.001 – 0.01**	Mengatur laju pembelajaran pada *model*
Momentum	**m**	**0.90 – 0.98**	Mengontrol keteraturan pembelajaran *model* untuk mencegah overfitting
Weight Decay	**wd**	**5e-5 – 1e-3**	Mengatur pengaruh kecepatan sebelumnya terhadap posisi baru
Koefisien Inersia	w	0.4 – 0.9	Mengatur pengaruh kecepatan sebelumnya terhadap posisi baru
Koefisien Kognitif	C_1	2.0	Bobot pengaruh posisi terbaik individu (*pbest)*
Koefisien Sosial	C_2	2.0	Bobot pengaruh posisi terbaik global (*gbest*)
Fitness	f(x)	1 – *mAP*50	Digunakan untuk menilai kualitas kombinasi parameter.

4.2.2 Hasil Pencarian Particle Swarm Optimization (PSO)



Gambar 4.3 Visualisasi ruang pencarian PSO

Proses optimasi dengan metode *Particle Swarm Optimization* (PSO) dilakukan untuk mencari kombinasi parameter terbaik pada *model* YOLOv8m. Parameter yang dioptimasi meliputi *learning rate (lr0), momentum,* dan *weight decay* yang masing – masing memiliki fungsi yang signifikan terhadap kestabilan pelatihan *model*. Hasil eksplorasi ruang pencarian PSO divisualisasikan pada Gambar 4.2.2. setiap titik mewakili posisi partikel pada kombinasi *hyperparameter* tertentu, dengan warna dan ukuran titik menunjukkan besarnya nilai *mAP*50 yang diperoleh pada konfigurasi tersebut. Garis biru putus – putus menunjukkan lintasan posisi. *Global Best (gBest)* selama proses iterasi, sedangkan pada penelitian ini bintang kuning menandai posisi overall best solution.

Berdasarkan hasil pada Gambar 4.2.2, dapat dilihat bahwa proses pencarian kombinasi *hyperparameter* terbaik menggunakan algoritma *Particle Swarm Optimization* (PSO) dengan parameter yang dioptimasi berupa learning rate (lr0), momentum, dan weight decay. Setiap partikel mempresentasikan satu kombinasi *hyperparameter* yang diuji dengan melatih *model* YOLOv8m selama 10 *epoch*, kemudian ouput dari prosesnya dinilai berdasarkan nilai *mAP*50 sebagai fungsi objektif. Tabel 4.2.2 berikut merangkum hasil global best (*gbest*) yang diperoleh pada setiap iterasi PSO, yaitu kombinasi *hyperparameter* dengan nilai *mAP*50 tertinggi di iterasi tersebut.

Tabel 4.6 *Output gBest* hasil *Particle Swarm Optimization*

Iterasi	Partikel	Lr0	*Momentum*	*Weight Decay*	*Precision*	*Recall*	*mAP*50
1	2	0.01	0.958917697	0.000692865	0.71649	0.79658	0.7762
2	2	0.01	0.958917697	0.000692865	0.71649	0.79658	0.7762
3	2	0.01	0.958917697	0.000692865	0.71649	0.79658	0.7762
4	2	0.01	0.958917697	0.000692865	0.71649	0.79658	0.7762
5	2	0.01	0.958917697	0.000692865	0.71649	0.79658	0.7762

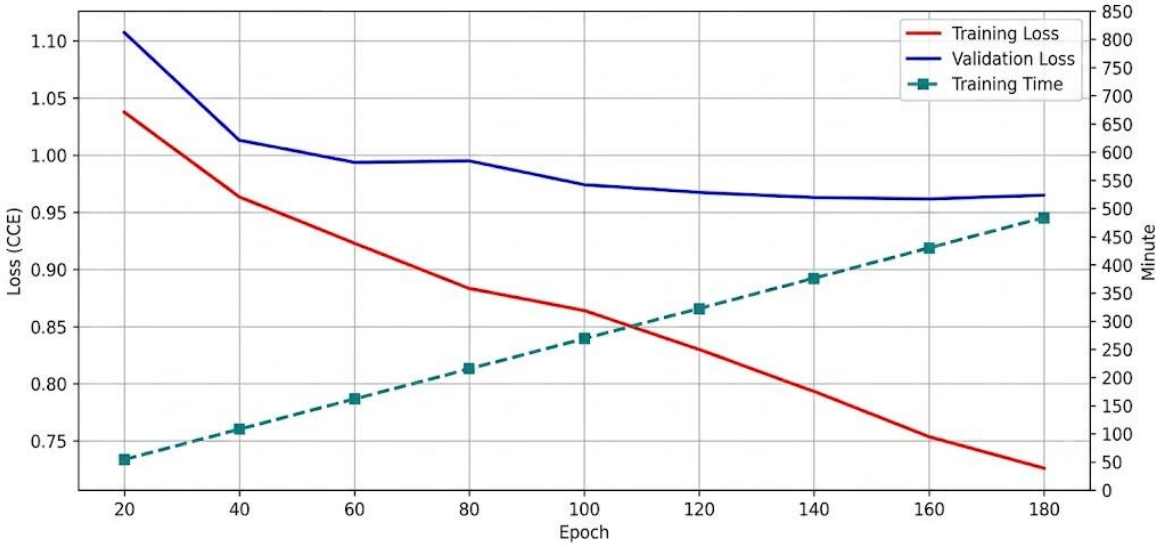
Berdasarkan Tabel 4.2.2, proses *Particle Swarm Optimization* menghasilkan nilai *gBest* yang sama pada setiap iterasi, yang menunjukkan bahwa algoritma PSO mencapai kondisi konvergen pada tahap awal optimasi. Kombinasi *hyperparameter* berupa *learning rate* sebesar 0.01, *momentum* sebesar 0.958917697, dan *weight decay* sebesar 0.000692865 secara konsisten menghasilkan nilai *mAP*50 tertinggi sebesar 0.7762. kondisi ini mengidentifikasikan bahwa solusi optimal telah ditemukan sejak iterasi awal, sehingga iterasi selanjutnya tidak memberikan peningkatan performa yang signifikan. Oleh karena itu, parameter hasil PSO tetap dipilih untuk pelatihan *model* YOLOv8m-PSO karena memberikan performa terbaik yang konsisten, sementara peningkatan kinerja *model* selanjutnya lebih ditentukan oleh proses pelatihan dan evaluasi *model*.

4.3 Hasil Pelatihan *Model* YOLOv8m dengan optimasi PSO

4.3.1 Grafik *Training Loss* dan *Validation Loss*

Gambar 4.4 memperlihatkan grafik *training loss* dan *validation loss model* YOLOv8m dengan parameter PSO. Selama training sampai 200 *epoch,* terlihat bahwa training loss mengalami penurunan secara konsisten seiring bertambahnya *epoch,* sedangkan validation loss menurun signifikan hingga sekitar *epoch* 100 dan kemudian cenderung stabil. Kondisi ini menunjukkan bahwa *model* mulai mencapai stabil pada *epoch* tersebut, dimana peningkatan performa pada data validasi tidak signifikan meskipun pelatihan dilanjutkan. Particle Swarm

Optimization dalam penentuan parameter, konvergensi yang lebih cepat pada *epoch* 40 – *epoch* 100 menunjukkan kemampuan PSO dalam mencari parameter optimal secara efisien.



Gambar 4.4 *Training loss*, *validation loss*, dan waktu pelatihan

4.3.2 Metrik Evaluasi *Model*

Metrik ini digunakan untuk melihat konsistensi performa *model* secara kuantitatif pada seluruh data evaluasi, sehingga tidak hanya bergantung pada skenario pengujian pada *model*.

Tabel 4.7 Metrik evaluasi titik stabil pada YOLOv8m + PSO

Epoch	***Precision***	***Recall***	**F1-*Score***	***mAP*50**	***mAP*50-95**
10	0.85282	0.88381	0.86804	0.90872	0.63486
20	0.89246	0.9172	0.90466	0.93701	0.67669
30	0.89429	0.9268	0.91025	0.95224	0.68932
40	0.90502	0.9339	0.91923	0.95345	0.70628
50	0.91612	0.92625	0.92116	0.95339	0.70581
60	0.92046	0.93958	0.92992	0.95961	0.71645
70	0.91585	0.92899	0.92237	0.95674	0.71796
80	0.92649	0.93791	0.93217	0.96018	0.714
90	0.92444	0.93922	0.93177	0.96197	0.72351
100	0.9246	0.94132	0.93289	0.96172	0.72346

Berdasarkan Tabel 4.7, terlihat bahwa pada *epoch* ke-10 nilai *precision*, *recall*, f1-score, serta *mAP* masih relatif lebih rendah dibandingkan *epoch* selanjutnya. Hal ini menunjukkan bahwa pada tahap awal pelatihan, meskipun parameter telah dioptimasi menggunakan PSO, *model* masih berada dalam proses penyesuaian terhadap data pelatihan. Seiring bertambahnya *epoch* hingga *epoch* ke-40 dan ke-50, seluruh metrik evaluasi mengalami peningkatan yang cukup signifikan, yang menandakan bahwa *model* mulai mampu melakukan deteksi objek dengan lebih baik dan konsisten.

4.3.3 Uji Deteksi gambar *Model*

Dilakukan pengujian deteksi langsung terhadap *model* pada setiap *epoch*. Pengujian ini bertujuan untuk membandingkan performa deteksi nyata dari setiap *model* yang tersimpan, bukan hanya berdasarkan nilai metrik pelatihan.

Tabel 4.8 Hasil nilai TP, TN, FP, dan FN deteksi pada 10 *epoch* terbai**k**

No.	*Epoch*	TP	TN	FP	FN	*Precision*	*Recall*	F1-*Score*	*mAP*50
1.	3	52	46	0	2	1	0.963	0.9811	0.8352
2.	24	52	46	0	2	1	0.963	0.9811	0.9341
3.	30	52	46	0	2	1	0.963	0.9811	0.9505
4.	34	52	46	0	2	1	0.963	0.9811	0.9558
5.	40	52	46	0	2	1	0.963	0.9811	0.9564
6.	**41**	**52**	**46**	**0**	**2**	**1**	**0.963**	**0.9811**	**0.9608**
7.	45	52	46	0	2	1	0.963	0.9811	0.9583
8.	14	51	46	0	3	1	0.9444	0.9714	0.9352
9.	12	50	46	0	4	1	0.9259	0.9615	0.9326
10.	28	50	46	0	4	1	0.9259	0.9615	0.9521

Berdasarkan hasil evaluasi nilai *True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN)* pada tabel 4.3.3, dapat dilihat bahwa performa klasifikasi paling optimal diperoleh pada *epoch* 41. Pada *epoch* ini *model* menghasilkan nilai *F1-Score* tertinggi sebesar 0.981 dengan nilai *Precision* sebesar 1000 lalu *recall* sebesar 0.963 dan *mAP*50 tertinggi 0.9608. Hal ini menunjukkan bahwa *model* mampu melakukan klasifikasi dengan tingkat kesalahan yang rendah ditandai dengan tidak adanya kesalahan (FP = 0), serta jumlah (FN) = 2). Dengan demikian *epoch* 41 dipilih sebagai *epoch* terbaik berdasarkan keseimbangan antara *precision* dan *recall* dalam evaluasi deteksi berbasis *confution matrix.*

4.4 Analisis Perbandingan Hasil YOLOv8 dan YOLOv8-PSO

4.4.1 Hasil Metrik Evaluasi Pelatihan *Precision*, *Recall*, F1-*Score*, TP, TN, FP, dan FN

Tabel 4.9 *Precision*, *Recall*, F1-*Score*, dan *mAP*

Model	*Epoch*	*Precision*	*Recall*	*F1-Score*	*mAP*50
YOLOv8m	140	0.92432	0.93435	0.92930	0.95871
YOLOv8m + PSO	41	0.90866	0.92904	0.91874	0.95611

Tabel 4.10 Hasil Metrik Evaluasi Pelatihan Deteksi Gambar (TP, TN, FP, FN)

Model	*Epoch*	TP	TN	FP	FN	*Precision*	*Recall*	*F1-Score*	*mAP*50
YOLOv8m	140	52	46	0	2	1	0.963	0.981	0.959
YOLOv8m+PSO	41	52	46	0	2	1	0.963	0.981	0.960

Berdasarkan hasil evaluasi pada metrik pelatihan, *model* YOLOv8m menunjukkan performa yang sedikit lebih baik dibandingkan YOLOv8m+PSO. YOLOv8m memperoleh nilai *precision, recall, F1-Score,* dan *mAP50* yang lebih tinggi, namun membutuhkan jumlah *epoch* yang cukup besar untuk mencapai hasil terbaiknya, yaitu 140 *epoch*. Sebaliknya YOLOv8m+PSO mampu mencapai performa optimal hanya dalam 41 *epoch,* meskipun nilai metrik pelatihannya sedikit lebih rerndah. Hal ini menunjukkan bahwa penerapan *Particle Swarm Optimization* (PSO) membantu mempercepat proses pelatihan pada *model*, sehingga *model* dapat mencapai kondisi konvergen dalam waktu yang lebih singkat.

Pada tahap evaluasi deteksi gambar, kedua *model* menunjukkan hasil yang hampir sama. Nilai TP, TN, FP, dan FN yang dihasilkan oleh YOLOv8m dan YOLOv8m+PSO identik, yang mengindikasikan bahwa kemampuan keduanya dalam mendeteksi objek pada data uji relatif seimbang. Selain itu, nilai *precision, recall,* dan *F1-Score* pada kedua *model* juga sama, dengan

perbedaan yang sangat kecil pada nilai *mAP50,* dimana YOLOv8m+PSO sedikit lebih unggul. Dengan demikian dapat disimpulkan bahwa YOLOv8m+PSO mampu mempertahankan performa deteksi yang setara dengan YOLOv8m, sekaligus memberikan efisiensi yang lebih baik dalam proses pelatihan.

4.5 Hasil Uji Deteksi *Tray*

Penerapan *Region of Interest* (*ROI*) untuk setiap posisi jagung pada *tray*, *Tray* pengujian dibagi menjadi 25 sel (5 x 5) yang masing – masing mempresentasikan satu posisi jagung, diberi penamaan A1 hinga E5.

Gambar 4.5 *Region of Interest* (*ROI*)

Penerapan *ROI* bertujuan untuk membatasi area deteksi *model* sehingga setiap objek jagung hanya dievaluasi pada area yang ditujukan. Dengan pendekatan ini, sistem mampu menghasilkan hasil prediksi *model* ke posisi *tray* tertentu secara konsisten, sekaligus mengurangi kemungkinan (*double detection)* atau deteksi pada luar area pengamatan.

Tabel 4.11 Cuplikan kode menentukan *ROI tray* jagung

ROIS = { 'A1': (363, 114, 452, 200), 'A2': (481, 109, 578, 206), 'A3': (601, 110, 700, 203), 'A4': (726, 109, 822, 209), 'A5': (851, 111, 945, 204), 'B1': (358, 233, 454, 325), 'B2': (480, 233, 576, 330), 'B3': (602, 235, 698, 329), 'B4': (724, 232, 823, 332), 'B5': (849, 232, 944, 331), 'C1': (358, 355, 450, 448), 'C2': (482, 353, 577, 449), 'C3': (606, 356, 697, 452), 'C4': (725, 355, 817, 448), 'C5': (848, 354, 939, 445), 'D1': (357, 474, 452, 569),	. = sumbu x1 . = sumbu y1 . = sumbu x2 . = sumbu y2

'D2': (475, 475, 577, 575), 'D3': (599, 472, 699, 576), 'D4': (721, 476, 824, 577), 'D5': (847, 476, 946, 577), 'E1': (358, 597, 451, 699), 'E2': (476, 597, 578, 701), 'E3': (600, 600, 696, 695), 'E4': (725, 599, 822, 695), 'E5': (848, 598, 940, 694), }	

Setiap *ROI* didefinisikan menggunakan sistem koordinat dua dimensi berbasis piksel dengan format (x_1, y_1, x_2, y_2) dimana (x_1, y_1) merepresentasikan titik kiri atas dan (x_2, y_2) merepresentasikan titik kanan bawah area *tray*. Koordinat *ROI* ditentukan secara manual untuk setiap posisi *tray* dari A1 – E5.

4.6 Hasil Pengambilan Data Deteksi Klasifikasi Biji Jagung

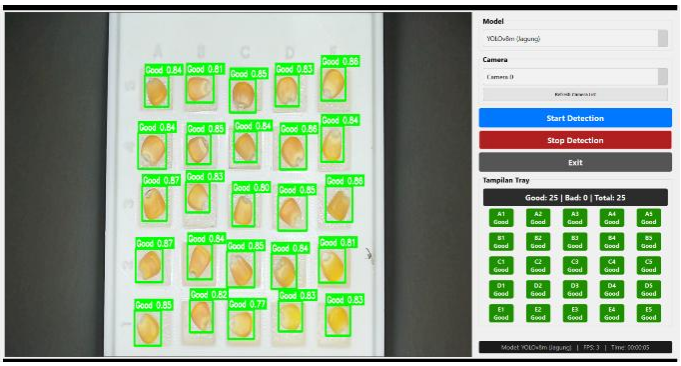
Proses pengambilan data pada penelitian ini menggunakan kamera yang akan menangkap objek biji jagung dan diklasifikasikan. Adapun variasi seperti variasi 1 (*tray* berwarna cerah dengan jagung baik mengisi *tray*), variasi 2 (*tray* berwarna gelap dengan jagung baik mengisi *tray*), variasi 3 (*tray* berwarna cerah dengan jagung buruk mengisi *tray*), variasi 4 (*tray* berwarna gelap dengan jagung buruk mengisi *tray*), variasi 5 (*tray* berwarna cerah dengan jagung baik dan buruk mengisi *tray*), dan yang terkahir variasi 6 (*tray* berwarna gelap dengan jagung baik dan buruk mengisi *tray*).

Variasi yang berbeda – beda ini diharapkan bisa dijadikan evaluasi seberapa baik performa pada *model* mendeteksi dengan warna latar belakang yang berbeda, keakuratan *model* dalam mengklasifikasikan biji jagung, dan metrik evaluasi lainnya. Selanjutnya akan bisa diketahui *model* mana yang memiliki performa paling baik dalam mendeteksi dan mengklasifikasi biji jagung secara akurat

.

4.6.1 YOLOv8m

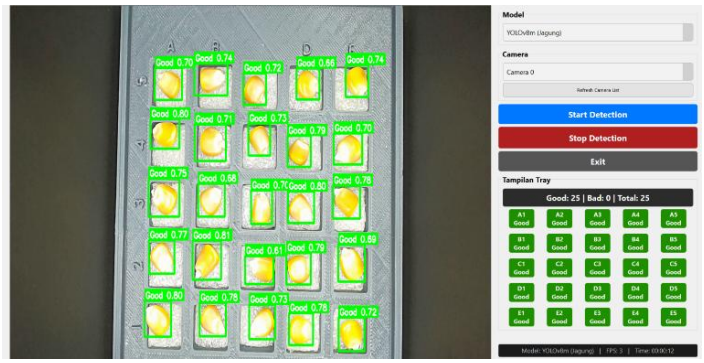
Tabel 4.12 Variasi 1 *tray* cerah biji jagung kualitas baik

Gambar	TP	TN	FP	FN	Rata Conf
	25	0	0	0	0.83
Jumlah Total Biji yang terdeteksi:	25				
Waktu Proses:	6 Detik				

- *Precision* : $\frac{25TP}{25TP+0FP} = \mathbf{1}$
- *Recall* : $\frac{25TP}{25TP+0FN} = \mathbf{1}$
- *F1-Score* : $2 \times \frac{1 \times 1}{1+1} = \frac{2}{2} = \mathbf{1}$
- Akurasi : $\frac{25TP+0TN}{25TP+0\,TN+0FP+0FN} = \mathbf{1}$

Berdasarkan Tabel 4.12, menunjukkan kinerja sempurna dengan berhasil mendeteksi seluruh biji (25 biji) secara tepat tanpa kesalahan (TP = 25, FP = 0, FN = 0). Hal ini menghasilkan nilai *Precision, Recall, F1-Score,* dan akurasi sebesar 1. Dengan rata – rata tingkat keyakinan (*confidence)* sebesar 0.83 dan waktu proses 6 detik, hasil ini membuktikan bahwa *model* bekerja sangat efektif dan akurat pada variasi *tray* cerah dengan biji jagung baik.

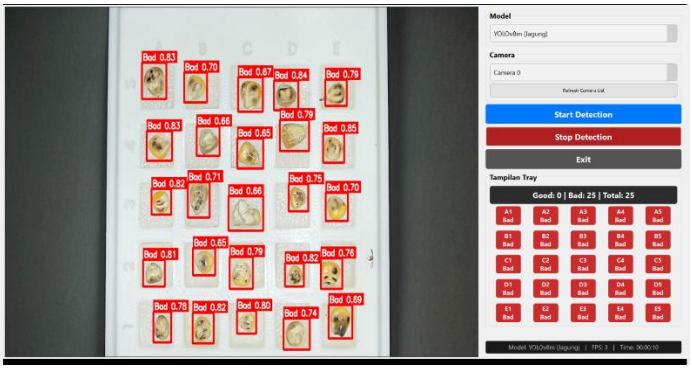
Tabel 4.13 Variasi 2 *tray* gelap biji jagung kualitas baik

Gambar	TP	TN	FP	FN	Rata Conf
	25	0	0	0	0.73
Jumlah Total Biji yang terdeteksi:	25				
Waktu Proses:	12 Detik				

- *Precision* : $\frac{25TP}{25TP+0FP} = \mathbf{1}$
- *Recall* : $\frac{25TP}{25TP+0FN} = \mathbf{1}$
- *F1-Score* : $2 \times \frac{1 \times 1}{1+1} = \frac{2}{2} = \mathbf{1}$
- Akurasi : $\frac{25TP+0TN}{25TP+0\,TN+0FP+0FN} = \mathbf{1}$

Pada Tabel 4.13, *model* tetap mempertahankan akurasi sempurna dengan mendeteksi 25 biji secara tepat tanpa kesalahan (*Precisio, Recall, F1-Score* = 1). Namun, terjadi penurunan efisiensi dibandingkan variasi *tray* cerah. Tingkat keyakinan (*confidence)* turun menjadi 0.73 dan waktu proses melambat 2 kali lipat menjadi 12 detik. Hal ini menunjukkan bahwa meskipun *model* mampu mengenali objek dengan benar, latar belakang gelap membuat *model* membutuhkan waktu komputasi lebih lama untuk memproses gambar. Penurunan tingkat keyakinan dan lonjakan waktu komputasi ini umumnya disebabkan oleh minimnya kontras antar tepi objek biji jagung dengan layar belakang gelap *tray*. Akibatnya, lapisan konvlusi pada arsitektur *model* harus melakukan komputasi yang lebih berat saat mengekstraksi fitur batas (*edge detection*) agar dapat memisahkan objek dari latar belakangnya secara akurat.

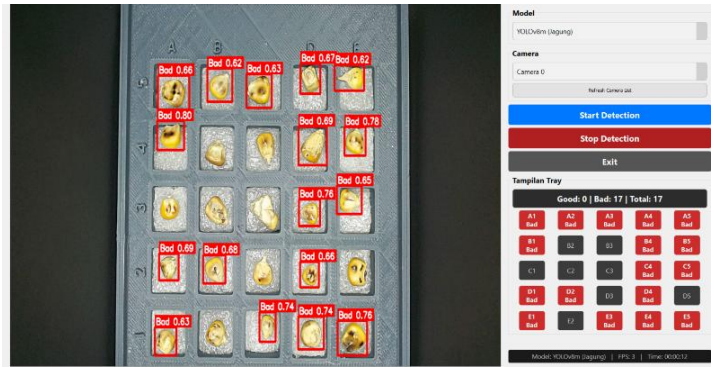
Tabel 4.14 Variasi 3 *tray* cerah biji jagung kualias buruk

Gambar	TP	TN	FP	FN	Rata Conf
	25	0	0	0	0.75
Jumlah Total Biji yang terdeteksi:	25				
Waktu Proses:	10 Detik				

- *Precision* : $\frac{25TP}{25TP+0FP} = \mathbf{1}$
- *Recall* : $\frac{25TP}{25TP+0FN} = \mathbf{1}$
- *F1-Score* : $2 \times \frac{1 \times 1}{1+1} = \frac{2}{2} = \mathbf{1}$
- Akurasi : $\frac{25TP+0TN}{25TP+0\,TN+0FP+0FN} = \mathbf{1}$

Pada Tabel 4.14, *model* berhasil mempertahankan akurasi sempurna (Akurasi = 1) dengan mendeteksi seluruh 25 biji jagung secara tepat. Meskipun demikian, terdapat indikasi penurunan efisiensi dibandingkan pengujian pada biji baik (variasi 1). Hal ini ditandai dengan penurunan rata – rata *confidence score* menjadi 0.75 dan peningkatan waktu proses menjadi 10 detik. Hal tersebut mengidentifikasi bahwa fitur visual pada biji buruk memiliki kompleksitas atau variabilitas yang lebih tinggi, sehingga menuntut waktu komputasi (inferensi) yang lebih lama meskipun kondisi pencahayaan atau latar belakang lebih optimal.

Tabel 4.15 Variasi 4 *tray* gelap biji jagung kualitas buruk

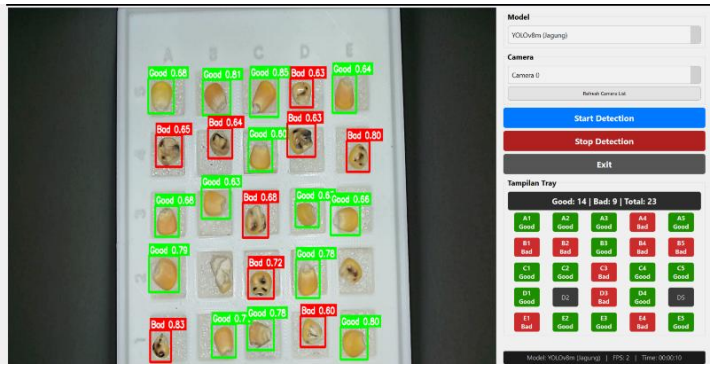
Gambar	TP	TN	FP	FN	Rata Conf
	17	0	0	8	0.69
Jumlah Total Biji yang terdeteksi:	17				
Waktu Proses:	12 Detik				

- *Precision* : $\frac{17TP}{17TP+0FP} = \mathbf{1}$

- *Recall* : $\frac{17TP}{17TP+8FN} = \mathbf{0.68}$
- *F1-Score* : $2 \times \frac{1 \times 0.68}{1+0.68} = \frac{1.36}{1.68} = \mathbf{0.80}$
- Akurasi : $\frac{17TP+0TN}{17TP+0\,TN+0FP+8FN} = \mathbf{0.68}$

Pada Tabel 4.15, performa *model* mengalami penurunan yang signifikan dibandingkan variasi – variasi sebelumnya. Berbeda dengan pengujian lain yang mencapai hasil sempurna, pada skenario ini nilai akurasi dan *recall* turun hingga 0.68 akibat tingginya angka *False Negative*, dimana 8 biji jagung gagal terdeteksi oleh *model*. Meskipun demikian, *model* tetap mempertahankan nilai *precision* tinggi (*precision = 1)* yang mendakan tidak adanya kesalahan identifikasi (*False Positive)*. Rendahnya rata – rata *confidence score* (0.69) serta lamanya waktu proses (12 detik) mengidentifikasikan bahwa kombinasi latar belakang gelap dan variabilitas bentuk pada biji buruk sangat menghambat proses ekstraksi fitur, sehingga membatasi kemampuan *model* untuk membedakan objek dari latar belakang (*background)* secara efektif.

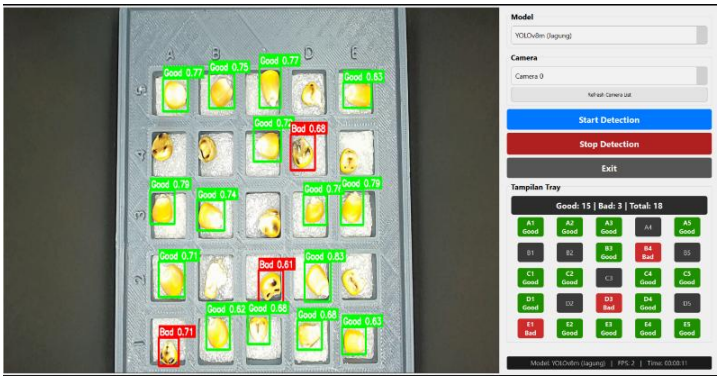
Tabel 4.16 Variasi 5 *tray* cerah biji jagung kualitas baik dan buruk

Gambar	TP	TN	FP	FN	Rata Conf
	22	0	1	2	0.70
Jumlah Total Biji yang terdeteksi:	23				
Waktu Proses:	11 Detik				

- *Precision* : $\frac{22TP}{22TP+1FP} = \mathbf{0.95}$
- *Recall* : $\frac{22TP}{22TP+2FN} = \mathbf{0.91}$
- *F1-Score* : $2 \times \frac{0.95 \times 0.91}{0.95+0.91} = \frac{1.72}{1.86} = \mathbf{0.93}$
- Akurasi : $\frac{17TP+0TN}{17TP+0\,TN+0FP+8FN} = \mathbf{0.68}$

Pada Tabel 4.17, kompleksitas deteksi meningkat akibat adanya objek beragam dalam satu *tray* pengujian. Hal ini berdampak pada munculnya kedua jenis kesalahan sekaligus, yakni 1 *False Positive* dan 2 *False Negative*, yang mengevaluasi nilai akurasi menjadi 0.88. meskipun demikian, performa *model* secara keseluruhan termasuk sangat baik, dibuktikan dengan *F1-Score* sebesar 0.93, yang menunjukkan keseimbangan antara *Precision* (0.95) dan *Recall* (0.91). Rata – rata *confidence score* sebesar 0.70 dan waktu proses 11 detik mengidentifikasikan bahwa variasi visual antar objek dalam satu *tray* menuntut beban komputasi yang lebih berat bagi *model* untuk membedakan karakteristik setiap biji secara akurat.

Tabel 4.17 Variasi 6 *tray* gelap biji jagung kualitas baik dan buruk

Gambar	TP	TN	FP	FN	Rata Conf
	15	0	2	7	0.71
Jumlah Total Biji yang terdeteksi:	18				
Waktu Proses:	11 Detik				

- *Precision* : $\frac{15TP}{15TP+2FP} = \mathbf{0.88}$
- *Recall* : $\frac{15TP}{15TP+7FN} = \mathbf{0.68}$
- *F1-Score* : $2 \times \frac{0.88 \times 0.68}{0.88+0.68} = \frac{1.19}{1.56} = \mathbf{0.76}$
- Akurasi : $\frac{15TP+0TN}{15TP+0\,TN+2FP+7FN} = \mathbf{0.62}$

Pada pengujian variasi ini, Tabel 4.17 hasil menunjukkan penurunan performa yang paling drastis dibandingkan seluruh skenario sebelumnya. Kombinasi kondisi latar belakang gelap dan objek yang campur menciptakan tantangan deteksi yang tinggi, yang ditandai dengan nilai akurasi menjadi 0.62. Kesulitan *model* dalam mengenali seluruh objek terlihat jelas dari tingginya angka *False Negative* sebanyak 7, yang mempengaruhi nilai *recall* turun menjadi 0.68. Meskipun demikian, nilai *precision* sebesar 0.88 menandakan bahwa *model* masih cukup stabil dalam meminimalisir kesalahan deteksi (*False Positive)* saat sebuah objek terdeteksi. Rata – rata *confidence score* 0.71 dan waktu proses 11 detik menandakan bahwa *model* menghadapi ketidakstabilan visual yang tinggi dalam membedakan fitur antar biji pada kondisi latar belakang yang kurang ideal.

Berikut merupakan hasil rata – rata yang meliputi waktu proses, true positive, true negative, false positive, f1-score, dan akurasi pada pengujian deteksi langsung *model* YOLOv8m tanpa optimasi dari *Particle Swarm Optimization*:

Tabel 4.18 Rata – rata hasil pengujian YOLOv8m

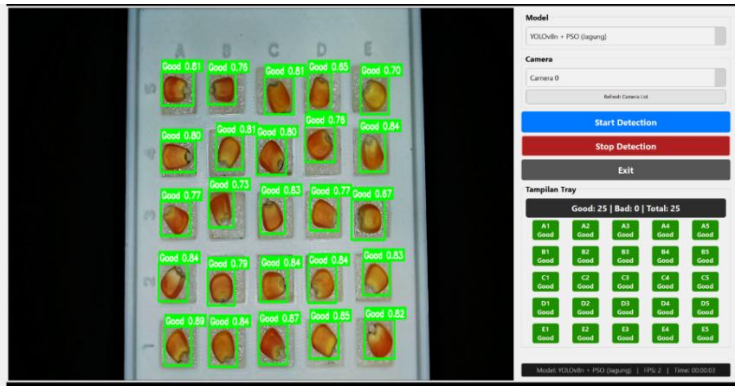
Parameter	Nilai rata – rata
Waktu Proses (detik)	**10.33**
True Positive	**21.50**
True Negative	**0**
False Positive	**0.50**
False Negative	**2.83**

Parameter	Nilai rata – rata
F1-Score	**0.92**
Akurasi	**0.86**

Berdasarkan rekapitulasi hasil pengujian pada *model baseline* YOLOv8m tanpa optimasi, *model* menunjukkan kinerja yang cukup stabil dengan rata – rata waktu proses sebesar 10.33 detik. Secara umum, *model* mengenali mayoritas objek dengan baik, ditunjukkan oleh rata – rata *True Positive* sebesar 21.50. Namun, analisis terhadap kesalahan deteksi memperlihatkan bahwa *model* memiliki kecenderungan melakukan kesalahan *(FN)* yang mencapai 2.83 jauh lebih tinggi dibandingan (*FP)* yang hanya bernilai 0.50. kondisi ini mengasilkan nilai akurasi rata – rata sebesar 0.86. Meskipun demikian, kestabilan antara *precision* dan *recall* masih tergolong tinggi, yang direpresentasikan oleh *F1-Score* sebesar 0.92. Hasil ini menjadi parameter awal untuk mengevaluasi efektivitas penerapan algoritma pada *model* selanjutnya.

4.6.2 YOLOv8m + PSO

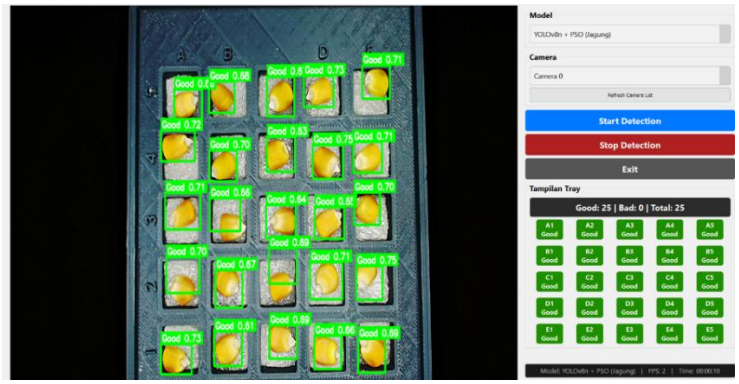
Tabel 4. 19 Variasi 1 *tray* cerah biji jagung kualitas baik

Gambar	TP	TN	FP	FN	Rata Conf
	25	0	0	0	0.79
Jumlah Total Biji yang terdeteksi:	25				
Waktu Proses:	3 Detik				

- *Precision* : $\frac{25TP}{25TP+0FP} = \mathbf{1}$
- *Recall* : $\frac{25TP}{25TP+0FN} = \mathbf{1}$
- *F1-Score* : $2 \times \frac{1 \times 1}{1+1} = \frac{2}{2} = \mathbf{1}$
- Akurasi : $\frac{25TP+0TN}{25TP+0\,TN+0FP+0FN} = \mathbf{1}$

Pada Tabel 4.19, implemetasi algoritma *Particle Swarm Optimization* (PSO) pada skenario ini memberikan dampak efisiensi yang sangat signifikan. Meskipun metrik kinerja utama seperti akurasi, *precision,* dan *recall* tetap bertahan pada angka yang tinggi 1 setara dengan *model baseline*. Terdapat peningkatan drastis dari sisi kecepatan komputasi. Waktu proses berhasil dipangkas menjadi 3 detik, jauh lebih singkat dibandingkan *model* tanpa optimasi yang membutuhkan waktu 6 detik. Penurunan tipis pada rata – rata *confidence score* menjadi 0.79 tidak mengurangi keandalan sistem, melainkan mengindikasikan bahwa *model* yang telah dioptimasi mampu mencapai hasil yang tepat dengan penggunaan sumber daya komputasi yang jauh lebih efisien.

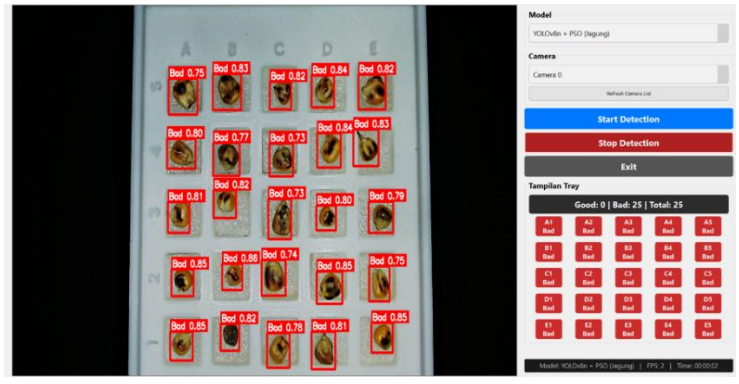
Tabel 4.20 Variasi 2 *Tray* Gelap Biji Jagung Kualitas Baik

Gambar	TP	TN	FP	FN	Rata Conf
	25	0	0	0	0.68
Jumlah Total Biji yang terdeteksi:	25				
Waktu Proses:	11 Detik				

- *Precision* : $\frac{25TP}{25TP+0FP} = \mathbf{1}$
- *Recall* : $\frac{25TP}{25TP+0FN} = \mathbf{1}$
- *F1-Score* : $2 \times \frac{1 \times 1}{1+1} = \frac{2}{2} = \mathbf{1}$
- Akurasi : $\frac{25TP+ 0TN}{25TP+0\,TN+0FP+0FN} = \mathbf{1}$

Pada Tabel 4.20, dengan implementasi PSO, *model* berhasil mempertahankan konsistensi akurasi sempurna (1) dengan mendeteksi seluruh 25 biji tanpa kesalahan deteksi. Efek optimasi ini terlihat pada efisiensi waktu, dimana tercatat waktu proses selama 11 detik, sedikit lebih cepat dibandingkan *model baseline* yang membutuhkan 12 detik. Namun, penurutan rata – rata *confidence score* ke angka 0.68 menandakan bahwa kondisi latar belakang minim tetap menjadi hambatan yang signifikan. Hal ini menunjukkan bahwa meskipun algoritma mampu memangkas beban komputasi, kompleksitas visual akibat *tray* gelap membatasi ruang optimalisasi, sehingga peningkatan kecepatan tidak sedratis yang terjadi pada kondisi *tray* cerah.

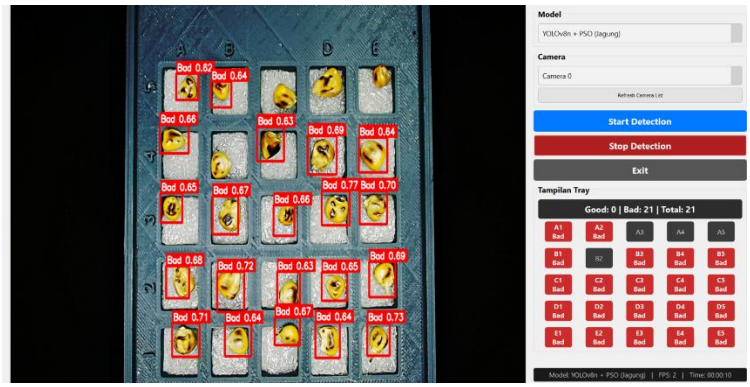
Tabel 4.21 Variasi 3 *tray* cerah biji jagung kualias buruk

Gambar	TP	TN	FP	FN	Rata Conf
	25	0	0	0	0.80
Jumlah Total Biji yang terdeteksi:	25				
Waktu Proses:	2 Detik				

- *Precision* : $\frac{25TP}{25TP+0FP} = \mathbf{1}$
- *Recall* : $\frac{25TP}{25TP+0FN} = \mathbf{1}$
- *F1-Score* : $2 \times \frac{1 \times 1}{1+1} = \frac{2}{2} = \mathbf{1}$
- Akurasi : $\frac{25TP+0TN}{25TP+0\,TN+0FP+0FN} = \mathbf{1}$

Pada Tabel 4.21 menunjukkan lonjakan performa dari penerapan algoritma PSO. Pada kategori deteksi biji buruk dengan *tray* cerah, sistem tidak hanya mempertahankan akurasi 1, tetapi juga mencatat waktu proses tercepat yakni hanya 2 detik. Capaian ini merupakan peningkatan efisiensi sebesar 5 kali lipat dibandingkan *model* tanpa optimasi (*baseline)* yang membutuhkan waktu 10 detik. Selain itu, peningkatan pada rata – rata *confience score* menjadi 0.80 (dibandingkan 0.75 pada *baseline*). Hal ini membuktikan bahwa pada kondisi *tray* cerah, optimasi parameter PSO sangat efektif dalam meningkatkan *sensitive* kemampuan *model* menganli fitur kompleks pada biji burukm sekaligus memangkas beban komputasi secara ekstrem.

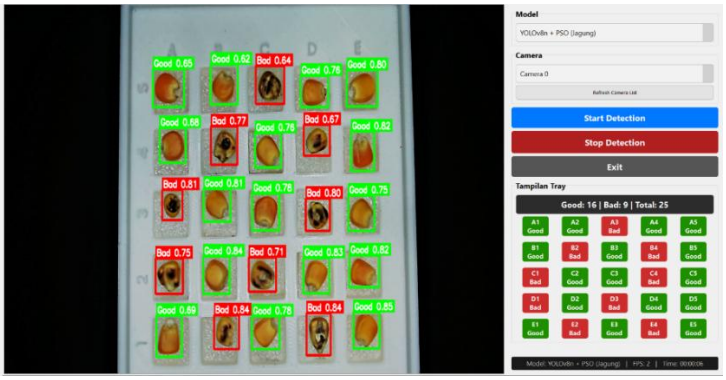
Tabel 4.22 Variasi 4 *tray* gelap biji jagung kualitas buruk

Gambar	TP	TN	FP	FN	Rata Conf
	21	0	0	4	0.67
Jumlah Total Biji yang terdeteksi:	21				
Waktu Proses:	10 Detik				

- *Precision* : $\frac{21TP}{21TP+0FP} = \mathbf{1}$
- *Recall* : $\frac{21TP}{21TP+4FN} = \mathbf{0.84}$
- *F1-Score* : $2 \times \frac{1 \times 0.84}{1+0.84} = \frac{1.68}{1.84} = \mathbf{0.91}$
- Akurasi : $\frac{21TP+0TN}{21TP+0\,TN+0FP+4FN} = \mathbf{0.84}$

Pada Tabel 4.22, optimasi PSO pada skenario ini terbukti sangat efektif, meningkatkan akurasi dan *recall* secara signifikan dari 68% menjadi 84%. Peningkatan ini terjadi karena penurunan angka *False Negative* hingga 50% (dari 8 menjadi 4) serta percepatan pada waktu proses menjadi 10 detik. Meskipun *tray* gelap masih menekan rata – rata *confidence score* (0.67), hasil ini mengonfirmasi kemampuan PSO dalam memperbaiki sensitivitas deteksi *model* secara drastis pada kondisi visual yang kompleks.

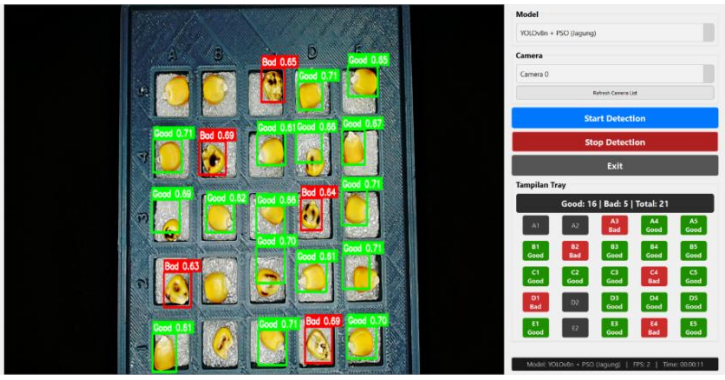
Tabel 4.23 Variasi 5 *tray* cerah biji jagung kualitas baik dan buruk

Gambar	TP	TN	FP	FN	Rata Conf
	25	0	0	0	0.76
Jumlah Total Biji yang terdeteksi:	25				
Waktu Proses:	6 Detik				

- *Precision* : $\frac{25TP}{25TP+0FP} = \mathbf{1}$
- *Recall* : $\frac{25TP}{25TP+0FN} = \mathbf{1}$
- *F1-Score* : $2 \times \frac{1 \times 1}{1+1} = \frac{2}{2} = \mathbf{1}$
- Akurasi : $\frac{25TP + 0TN}{25TP+0\,TN+0FP+0FN} = \mathbf{1}$

Pada Tabel 4.23, penerapan PSO pada variasi campuran berhasil menghindari kesalahan deteksi *(False Positive)* yang sebelumnya terdapat pada *model baseline*, menghasilkan akurasi sempurna 100%. Tidak hanya lebih akurat, efisiensi waktu juga meningkat drastis menjadi 6 detik, dan kenaikan *confidence score* 0.76 membuktikan bahwa *model* optimasi PSO mampu menangani keragaman objek dengan jauh lebih sensitif dan akurat.

Tabel 4.24 Variasi 6 *tray* gelap biji jagung kualitas baik dan buruk

Gambar	TP	TN	FP	FN	Rata Conf
	18	0	3	4	0.66
Jumlah Total Biji yang terdeteksi:	21				
Waktu Proses:	11 Detik				

- *Precision* : $\frac{18TP}{18TP+3FP} = \mathbf{0.85}$
- *Recall* : $\frac{18TP}{18TP+4FN} = \mathbf{0.81}$

- *F1-Score* : $2 \times \frac{0.85 \times 0.81}{0.85+0.81} = \frac{1.37}{1.66} = \mathbf{0.82}$
- Akurasi : $\frac{18TP+0TN}{18TP+0\,TN+3FP+4FN} = \mathbf{0.72}$

Pada Tabel 4.24, integrasi PSO pada skenario terberat ini terbukti efektif menaikkan akurasi menjadi 0.72, meningkat signifikan dibandingkan *model baseline* yang hanya mencapai 0.62. Peningkatan performa paling krusial terutama pada nilai *recall* menjadi 0.81 dimana *model* berhasil menekan jumlah objek yang terlewat (*False Negative)* hingga tersisa 4 FN. Meskipun sensitivitas deteksi membaik, faktor *tray* gelap tetap menjadi tantangan utama, ditunjukkan oleh rata – rata *confidence score confidence score* yang diangka 0.66 dan waktu proses yang sama di 11 detik.

Berikut merupakan hasil rata – rata yang meliputi waktu proses, true positive, true negative, false positive, f1-score, dan akurasi pada pengujian deteksi langsung *model* YOLOv8m dengan optimasi *learning rate, momentum, dan weight decay* dari *Particle Swarm Optimization*:

Tabel 4.25 Rata – rata Hasil Pengujian YOLOv8m+PSO

Parameter	Nilai Rata – rata
Waktu Proses (detik)	**7.17**
True Positive	**23.17**
True Negative	**0**
False Positive	**0.50**
False Negative	**1.33**
F1-Score	**0.96**
Akurasi	**0.93**

Pada Tabel 4.25, secara keseluruhan, itegrasi optimasi PSO terbukti behasil meningkatkan performa *model* secara signifikan dibandingkan *baseline.* Hal ini dibuktikan dengan kenaikan rata – rata akurasi menjadi 0.93, yang di dorong oleh kemampuan *model* memangkas tingkat kesalahan deteksi (*False Negative)* hingga lebih dari separuhnya, menjadi rata – rata 1.33. Selain lebih akurat, efisiensi dalam hal komputasi juga membaik dengan rata – rata waktu proses yang lebih singkat, yakni 7,17 detik (lebih cepat dari *model baseline* 10.33 detik). Stabilitas angka pada *False Positive* yang tetap rendah di (0.50), menegaskan bahwa peningkatan sensitivitas deteksi tersebut tercapai tanpa mengurangi seletivitas *model* terhadap objek.

4.6.3 Analisis Perbandingan Performa YOLOv8m dan YOLOv8m+PSO

Setelah pada subbab 4.6.1 dan 4.6.2 dipaparkan hasil deteksi dan analisis masing – masing *model* pada setiap variasi pengujian, tahap berikutnya adalah menyatukan temuan tersebut ke dalam satu pembahasan perbandingan yang lebih jelas. Pada penelitian ini terdapat 6 variasi skenario, yaitu variasi 1 *tray* berwarna cerah dengan diisi jagung baik, variasi 2 *tray* berwarna gelap dengan diisi jagung baik, variasi 3 *tray* berwarna cerah diisi dengan jagung buruk, variasi 4 *tray* berwarna gelap diisi dengan jagung buruk, variasi 5 *tray* berwarna cerah diisi dengan jagung baik dan buruk, variasi 6 *tray* berwarna gelap diisi dengan jagung baik dan buruk. 6 skenario ini dipakai untuk mengevaluasi kestabilan performa *model* terhadap perbedaan latar belakang, dan pencahayaan, sekaligus melihat keakuratan *model* dalam mengklasifikasikan biji jagung.

Tabel 4.26 Perbandingan performa TP, TN, FP, dan FN pada setiap skenario

Skenario	YOLOv8				YOLOv8+PSO			
	TP	**TN**	**FP**	**FN**	**TP**	**TN**	**FP**	**FN**
Tray Cerah dengan Jagung Baik (*Good)*	25	0	0	0	25	0	0	0
Tray Gelap dengan Jagung Baik (*Good)*	25	0	0	0	25	0	0	0
Tray Cerah dengan Jagung Buruk (*Bad)*	25	0	0	0	25	0	0	0
Tray Gelap dengan Jagung Buruk (*Bad)*	17	0	0	8	21	0	0	4
Tray Cerah dengan Jagung Campur (*Good / dan Bad)*	22	0	1	2	25	0	0	0
Tray Gelap dengan Jagung Campur (*Good dan Bad)*	15	0	2	7	18	0	3	4

Berdasarkan tabel perbandingan TP, TN, FP, dan FN untuk enam skenario tersebut, terlihat bahwa pada kondisi yang relatif mudah yaitu *tray* cerah dengan jagung baik, *tray* gelap dengan jagung baik, dan *tray* cerah dengan jagung buruk, kedua *model* menunjukkan performa yang sama baiknya. Pada ketiga skenario ini, YOLOv8m dan YOLOv8m+PSO sama – sama mampu menghasilkan deteksi yang "bersih" (TP tinggi, tanpa FP, dan FN), sehingga optimasi PSO tidak tampak memberikan perbedaan yang menonjol ketika objek dan kondisi pencahayaan masih mendukung. Kondisi ini juga selaras dengan temuan pada pengujian deteksi gambar, dimana kedua *model* menghasilkan performa yang hampir sama dari sisi metrik evaluasi.

Perbedaan mulai bisa terlihat ketika masuk pada skenario yang lebih susah, terutama pada *tray* gelap dengan jagung buruk. Pada kondisi ini, YOLOv8m menunjukkan adanya objek yang tidak terdeteksi (FN), sedangkan YOLOv8m+PSO mampu menurunkan jumlah FN dan menaikkan TP. Secara praktis, ini menunjukkan bahwa optimasi PSO membantu *model* menjadi lebih peka terhadap karakteristik jagung buruk ketika pencahayaan menurun, sehingga biji yang sebelumnya tidak terdeteksi dapat tertangkap lebih banyak. Pola yang mirip juga terlihat pada skenario campur, pada *tray* cerah campur, YOLOv8m+PSO cenderung lebih bersih karena TP meningkat dan kesahalan deteksi berkurang, sedangkan pada *tray* gelap campur, YOLOv8m+PSO tetap menunjukkan peningkatan TP dan penurunan FN, namun diikuti kenaikan FP yang kecil. Artinya, pada kondisi paling sulit (*tray* gelap+campur), PSO membuat *model* lebih sensitif sehingga objek yang terlewat berkurang, tetapi ada konsekuensi berupa tambahan prediksi yang tidak sesuai.

Jika ditarik pada ringkasan rata – rata deteksi *tray*, kecenderungan tersebut menjadi lebih jelas, YOLOv8m+PSO menunjukkan peningkatan jumlah *True Positive* dan penurunan *False Negative* dibandingkan YOLOv8m, yang kemudian berdampak langsung pada kenaikan nilai

F1-Score dan akurasi. Hal ini sejalan dengan ringkasan metrik rata – rata performa deteksi *tray* pada sub-bab 4.6.3, dimana *model* YOLOv8m+PSO memiliki rata – rata TP lebih tinggi dan FN lebih rendah, serta nilai F1-Score dan akurasi yang lebih baik dibandingkan *model baseline*.

Disisi lain, dari sudut pandang efisiensi pelatihan *model*, pembahasan sebelumnya juga menunjukkan bahwa meskipun metrik pelatihan *model* tanpa optimasi bisa sedikit lebih tinggi, perbedaannya relatif kecil. Namun YOLOv8m+PSO dapat mencapai performa yang mendekati *model* tanpa optimasi PSO dengan jumlah *epoch* yang jauh lebih sedikit, sehingga PSO lebih terlihat perannya dalam meningkatkan efisiensi optimasi selama pelatihan dan membantu *model* lebih cepat mencapai kondisi konvergen / stabil.

Tabel 4.27 Perbandingan rata – rata metrik performa Deteksi *Tray*

Model	**Nama File**	**Waktu Proses**	**TP**	**TN**	**FP**	**FN**	**F1-Score**	**Akurasi**
YOLOv8m	Epoch140.pt	10 Detik	21.50	0	0.50	2.83	0.92	0.86
YOLOv8m+PSO	Epoch41.pt	7 Detik	23.17	0	0.50	1.33	0.96	0.93

Tabel 4.27 menunjukkan hasil pengujian kemudian dirangkum dalam bentuk rata – rata metrik performa performa deteksi *tray*. Pada tabel rata – rata, YOLOv8m menghasilkanwaktu proses 10 detik dengan nilai TP rata – rata 21.50, FP 0.50, dan FN 2.83. Sementara itu, YOLOv8m+PSO menunjukkan waktu proses yang lebih cepat yaitu 7 detik, dengan nilai TP rata – rata meningkat menjadi 23.17 dan FN menurun menjadi 1.33, sedangkan FP tetap pada nilai 0.50. Hasil ini menegaskan bahwa penerapan PSO mampu meningkatkan jumlah deteksi benar dan mengurangi jumlah objek yang tidak terdeteksi, tanpa menambah kesalahan deteksi secara rata – rata.

Peningkatan tersebut juga terlihat pada metrik *F1-Score* dan akurasi. YOLOv8m memiliki *F1-Score* sebesar 0.92 dengan akurasi 0.86, sedangkan YOLOv8m+PSO memiliki *F1-Score* sebesar 0.96 dengan akurasi 0.93. Dengan demikian, selain memberikan peningkatan pada TN dan FN, penggunaan PSO juga berdampak pada kualitas performa *model* secara keseluruhan.

Halaman ini sengaja dikosongkan.

BAB 5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini berhasil merancang sistem *handheld* otomatis untuk klasifikasi biji jagung yang stabil menggunakan peraturan *ROI*. Penerapan PSO pada YOLOv8m terbukti meningkatkan kinerja deteksi secara signifikan, yang terlihat dari kenaikan nilai TP, penurunan FN, dan efisiensi waktu proses yang lebih baik dibandingkan *model* tanpa optimasi. Mengacu pada rumusan masalah dan tujuan penelitian, maka dapat disimpulkan:

1. Perancangan sistem *handheld* berbasis *image processing* berhasil direalisasikan dan dapat digunakan untuk klasifikasi biji jagung secara otomatis pada media *tray* dengan bantuan *ROI*, sehingga proses deteksi lebih ter-arah dan siap diterapkan pada kondisi lapangan yang terkontrol.
2. Penerapan YOLOv8m mampu mendeteksi dan mengklasifikasi biji jagung berdasarkan karakteristik visual dengan performa yang baik, dengan *checkpoint baseline* terbaik berada pada *epoch* 140 menghasilkan *Precision* 0.92432, *Recall* 0.93435, *F1-Score* 0.92930, dan *mAP*50 0.95871.
3. Implementasi PSO untuk optimasi *hyperparameter* terbukti meningkatkan performa YOLOv8m sekaligus mempercepat proses. Pada pengujian *tray*, *model* YOLOv8m+PSO memiliki *F1-Score* 0.96, dan akurasi 0.93 dengan waktu proses 7 detik, lebih baik dibandingkan YOLOv8m *baseline* dengan *F1-Score* 0.92, akurasi 0.86, dan waktu proses 10 detik. Peningkatan juga terlihat dari TP yang naik dan FN yang turun.

5.2 Saran

Saran pada penelitian ini disusun berdasarkan hasil pengujian dan pembahasan, khususnya pada aspek penerapan sistem di lapangan, kestabilan performa *model*, serta peningkatan kinerja setelah optimasi *hyperparameter*. Oleh karena itu, saran yang diberikan bersifat aplikatif dan realistis, serta diharapkan dapat menjadi acuan untuk pengembangan dan penelitian selanjutnya sebagai berikut.

1. Disarankan dilakukan penerapan dan uji coba lebih lanjut di lingkungan kerja yang sesungguhnya dengan menyusun panduan penggunaan (SOP) singkat serta melibatkan beberapa operator. Hal ini bertujuan untuk memastikan sistem tetap stabil, mudah digunakan, dan konsisten dalam mendukung proses sortir.
2. Untuk meningkatkan kesiapan penggunaan di lapangan, disarankan dilakukan pengujian pada kondisi yang lebih bervariasi, seperti perbedaan karakteristik kerusakan biji, kemiripan visual antar kelas, dan variasi kondisi pencahayaan.
3. Mengingat PSO terbukti meningkatkan performa dan mempercepat proses, penelitian selanjutnya disarankan mengfokuskan pada pemantapan konfigurasi *hyperparameter* terbaik serta pengujian efisiensi penggunaan berulang. Dengan demikian, manfaat peningkatan akurasi dan kecepatan dapat lebih terukur.

Halaman ini sengaja dikosongkan.

DAFTAR PUSTAKA

Admin (2024) *Rahasia Jagung Super: Bongkar Kelebihan Jagung Hibrida yang Bikin Petani Untung Besar*, *jafranindonesia.co.id*. Available at: https://jafranindonesia.co.id/blog/rahasia-jagung-super-bongkar-kelebihan-jagung-hibrida-yang-bikin-petani-untung-besar.

Akış, U. and Dişlitaş, S. (2024) 'Intelligent Lighting System Using Color-Based Image Processing for Object Detection in Robotic Handling Applications', *Applied Sciences (Switzerland)*, 14(7). Available at: https://doi.org/10.3390/app14073002.

Andrey Germanov (2024) *How to create YOLOv8-based object detection web service using Python, Julia, Node.js, JavaScript, Go and Rust*, *dev.to*. Available at: https://dev.to/andreygermanov/how-to-create-yolov8-based-object-detection-web-service-using-python-julia-nodejs-javascript-go-and-rust-4o8e.

Archana, R. and Jeevaraj, P.S.E. (2024) *Deep learning models for digital image processing: a review*, *Artificial Intelligence Review*. Springer Netherlands. Available at: https://doi.org/10.1007/s10462-023-10631-z.

Ayyad, S.M. *et al.* (2025) 'Particle swarm optimization with YOLOv8 for improved detection performance of tomato plants', *Journal of Big Data*, 12(1). Available at: https://doi.org/10.1186/s40537-025-01206-6.

Brar, I.S. *et al.* (2017) 'Studies on Physical Properties of Maize (Zea mays L.) Seeds', *International Journal of Current Microbiology and Applied Sciences*, 6(10), pp. 963–970. Available at: https://doi.org/10.20546/ijcmas.2017.610.116.

Chandra, D. and Sembiring, S. (2024) 'Meningkatkan Efisiensi Pemrosesan Citra Untuk Klasifikasi Kualitas Biji Jagung Berbasis Tekstur', *Jurnal Ilmiah Multidisiplin Ilmu Komputer*, 1(2), pp. 60–73. Available at: https://geloraciptanusantara.org/jurnal/index.php/jimik/article/view/167.

Che, Y. *et al.* (2025) 'Real-Time Detection of Varieties and Defects in Moving Corn Seeds Based on YOLO-SBWL', *Agriculture (Switzerland)*, 15(7). Available at: https://doi.org/10.3390/agriculture15070685.

Chen, S. *et al.* (2024) 'Soft X-ray image recognition and classification of maize seed cracks based on image enhancement and optimized YOLOv8 model', *Computers and Electronics in Agriculture*, 216(August 2023), p. 108475. Available at: https://doi.org/10.1016/j.compag.2023.108475.

Dayat (2025) *Grading biji jagung berdasarkan tingkat kerusakan*, *academy.bertani*. Available at: https://academy.bertani.co/wiki/panen-pascapanen-jagung.

Egyed, G.K.M.W.K.G.A.P.G.A. (2023) *Analysis and Propagation of Feature Revisions in Preprocessor-based Software Product Lines*, *IEEE Access*. Available at: https://ieeexplore.ieee.org/document/10123456/authors.

Elgamily, K.M. *et al.* (2025) *Enhanced object detection in remote sensing images by applying metaheuristic and hybrid metaheuristic optimizers to YOLOv7 and YOLOv8*, *Scientific Reports*. Available at: https://doi.org/10.1038/s41598-025-89124-8.

Erich Gamma (2025) *Visual Studio Code*, *Wikipedia.org*. Available at: https://en.wikipedia.org/wiki/Visual_Studio_Code.

EWINDO (2021) *PT EAST WEST SEED INDONESIA*, *panahmerah.id*. Available at:

https://panahmerah.id/id/about?utm_source=chatgpt.com.

Fan, X. and Zhou, J. (2025) 'Nondestructive Detection and Quality Grading System of Walnut Using X-Ray Imaging and Lightweight WKNet', *Foods*, 14(13), pp. 1–18. Available at: https://doi.org/10.3390/foods14132346.

Grad (2024) *Corn Computer Vision Mode*, *universe.roboflow*. Available at: https://universe.roboflow.com/grad-jmjxr/corn-lycsy.

Greeksforgeeks (2025) *Components of Image Processing System*, *computer vision*. Available at: https://www.geeksforgeeks.org/computer-vision/components-of-image-processing-system/)/.

Guido van rossum (2025) *Pyhton (Programming Language)*, *Wikipedia.org*. Available at: https://en.wikipedia.org/wiki/Python_(programming_language).

Hu, Y. *et al.* (2025) 'Classification of maize seed hyperspectral images based on variable-depth convolutional kernels', *Frontiers in Plant Science*, 16(June), pp. 1–16. Available at: https://doi.org/10.3389/fpls.2025.1599231.

Kalimuthu, M. and Krishnamurthi, I. (2015) 'Semantic-based facial image-retrieval system with aid of adaptive particle swarm optimization and squared Euclidian distance', *Journal of Applied Mathematics*, 2015. Available at: https://doi.org/10.1155/2015/284378.

Kılcı, O. and Koklu, M. (2025) 'Automated Classification of Biscuit Quality Using YOLOv8 Models in Food Industry', *Food Analytical Methods*, 18(5), pp. 815–829. Available at: https://doi.org/10.1007/s12161-025-02755-5.

Luo, H. *et al.* (2019) 'A Survey on Deep Learning Based Person Re-identification', *Zidonghua Xuebao/Acta Automatica Sinica*, 45(11), pp. 2032–2049. Available at: https://doi.org/10.16383/j.aas.c180154.

Magdin, M. and Balogh, Z. (2025) 'Comparison classification algorithms and the YOLO method for video analysis and object detection', *Scientific Reports*, 15(1), pp. 1–13. Available at: https://doi.org/10.1038/s41598-025-09814-1.

Maida (2024) *LED Strip with Raspberry Pi 5*, *bog post catalogue*. Available at: https://www.raspberrypibox.com/drive-led-strip-with-raspberry-pi-5/?utm_source=chatgpt.com.

Marsico, M. De (2023) *Pattern Recognition Letters*, *Science Direct*. Available at: https://www.sciencedirect.com/journal/pattern-recognition-letters.

Megantara, N.A. and Utami, E. (2025) 'Object Detection Using YOLOv8 : A Systematic Review', *SISTEMASI : Jurnal Sistem Informasi*, 14(3), pp. 1186–1193. Available at: http://sistemasi.ftik.unisi.ac.id.

New Chips Poised to Revolutionize Photography, F. (2008) *Pemrosesan citra digital*, *wired.com*. Available at: https://www.wired.com/2008/10/new-chips-poise/?utm_source=chatgpt.com.

Niu, S. *et al.* (2024) 'Research on a Lightweight Method for Maize Seed Quality Detection Based on Improved YOLOv8', *IEEE Access*, 12(January), pp. 32927–32937. Available at: https://doi.org/10.1109/ACCESS.2024.3365559.

Pativada, P.K. (2024) 'Real-time detection and classification of plant seeds using YOLOv8 object detection model', (Table 10), pp. 4–6.

Pavan Kumar Pativada (2024) 'Real-time detection and classification of plant seeds using

YOLOv8 object detection model', *Αγαη*, 15(1), pp. 37–48.

Pawening, R.E., Zaskiya, K.D. and Hasanah, S. (2024) 'Classification of Corn Seed and Cob Quality Based on Texture and RGB Color Features Using Backpropagation Method', 7(4), pp. 346–356.

Perez, R.E. and Behdinan, K. (2007) 'Particle swarm approach for structural design optimization', *Computers and Structures*, 85(19–20), pp. 1579–1588. Available at: https://doi.org/10.1016/j.compstruc.2006.10.013.

Ryabtsev, A. (2024) *Reasons Why Python is Good For AI and Machine Learning*, *Backend Competech and Tech Lead*. Available at: https://djangostars.com/blog/why-python-is-good-for-artificial-intelligence-and-machine-learning/.

Sai, R. *et al.* (2025) 'A comparative analysis of deep learning models for waste segregation : YOLOv8 , EfficientDet , and Detectron 2', *Multimedia Tools and Applications*, 84(29), pp. 35941–35964. Available at: https://doi.org/10.1007/s11042-025-20647-y.

Tan, J., Chen, Y. and Jiao, S. (2024) *Visual Studio Code in Introductory Computer Science Course: An Experience Report*, *ASEE Annual Conference and Exposition, Conference Proceedings*. Association for Computing Machinery. Available at: https://doi.org/10.18260/1-2--48259.

Tang, H. *et al.* (2021) 'Evaluation of physical characteristics of typical maize seeds in a cold area of north china based on principal component analysis', *Processes*, 9(7). Available at: https://doi.org/10.3390/pr9071167.

Yaseen, M. (2024) 'What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector', 8, pp. 1–10. Available at: http://arxiv.org/abs/2409.07813.

Yolo.org (2023) *Experience YOLOv8*, *yolov8.org*. Available at: https://yolov8.org/.

Zheng, Z. *et al.* (2020) 'Distance-IoU loss: Faster and better learning for bounding box regression', *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, (2), pp. 12993–13000. Available at: https://doi.org/10.1609/aaai.v34i07.6999.

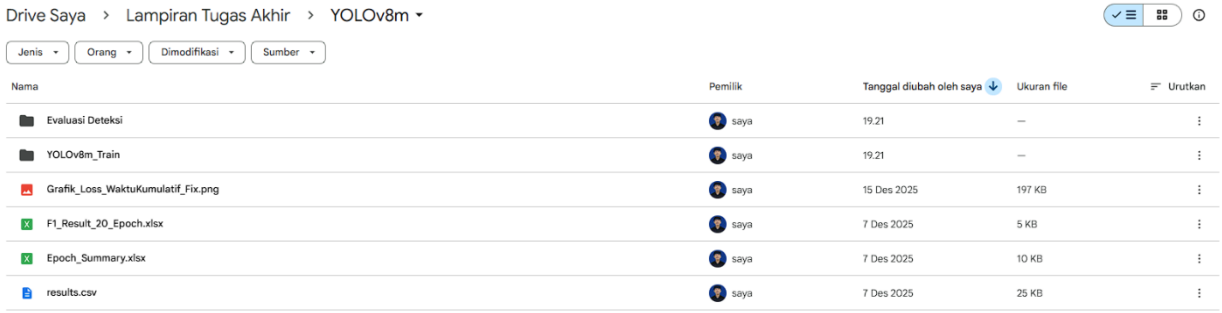
Zhong, H. *et al.* (2025) 'PS-YOLO: A Lighter and Faster Network for UAV Object Detection', *Remote Sensing*, 17(9). Available at: https://doi.org/10.3390/rs17091641.

Halaman ini sengaja dikosongkan.

LAMPIRAN

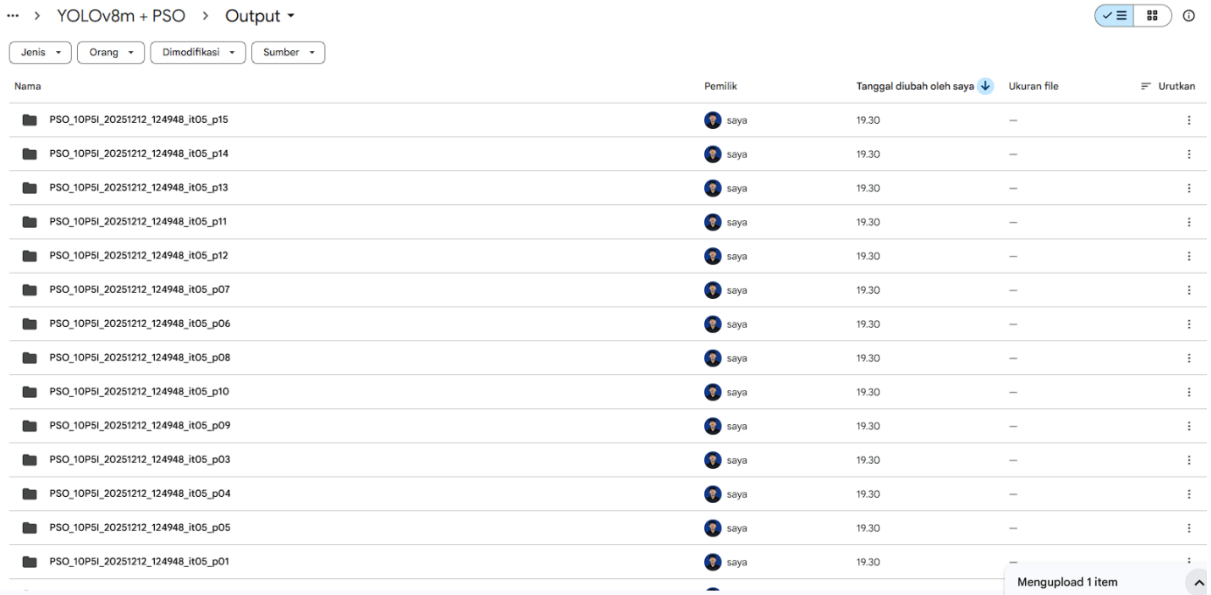
Lampiran 1. Log Train YOLOv8m (Tanpa PSO)

https://drive.google.com/drive/folders/1DG6wBYjAFykwRWPxtzvxEtoFaKriQ8KF?usp=sharing



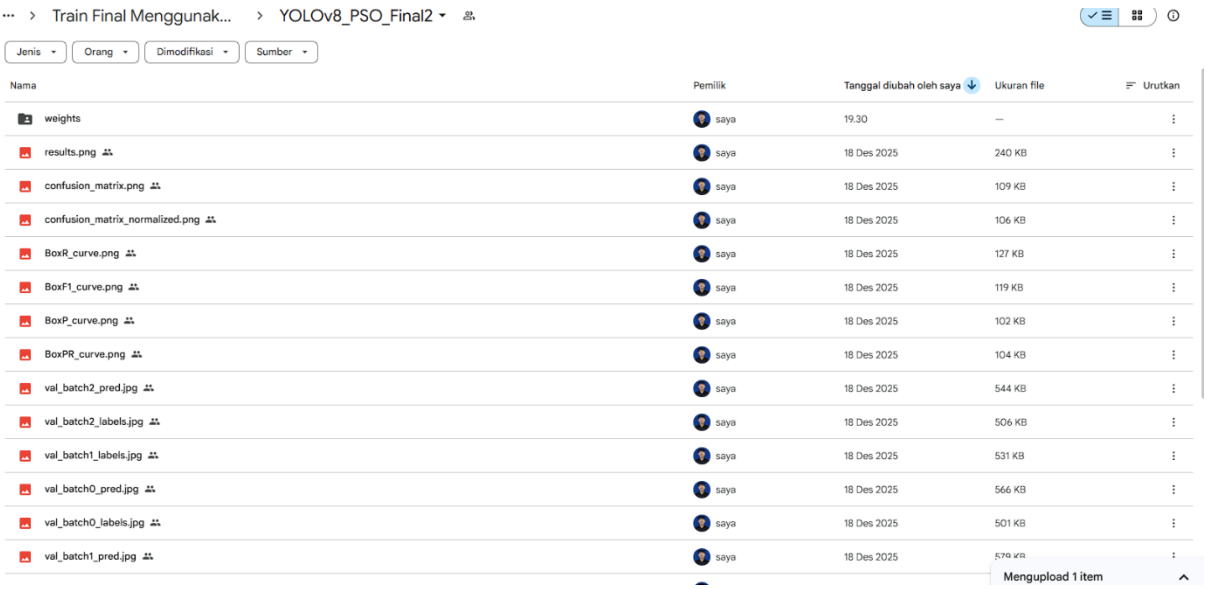
Lampiran 2. Log Pencarian Hyperparameter Pada YOLOv8m

https://drive.google.com/drive/folders/1uMNISlcT96IvTxY4MfaOjJSUh9VLKOBR?usp=sharing



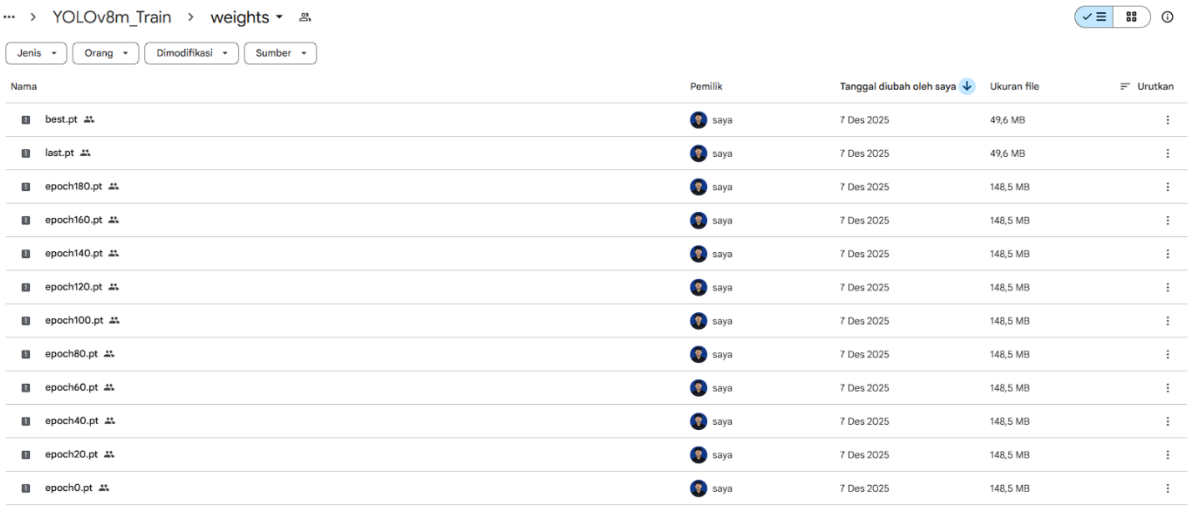
Lampiran 3. Log Train YOLOv8 dengan hyperparameter PSO

https://drive.google.com/drive/folders/1ezJgxZi808sf8DvqmJSAHbuiR7zDely1?usp=sharing



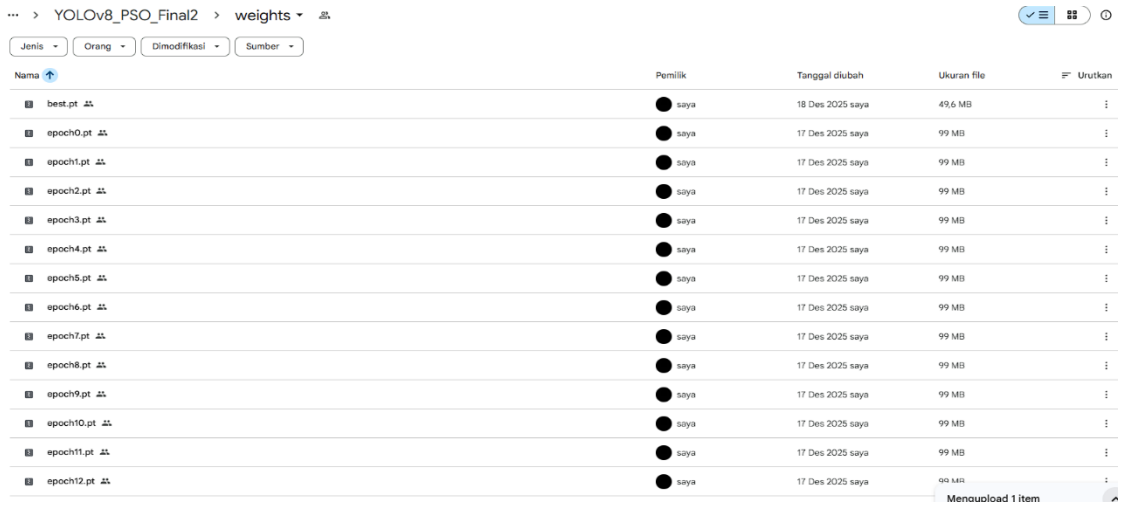
Lampiran 4. Model YOLOv8m Baseline

https://drive.google.com/drive/folders/18b6O5730g0ciTbefIyRKpc_qknpti1qO?usp=sharing



Lampiran 5. Model YOLOv8m+PSO

https://drive.google.com/drive/folders/18lYAXlavkS0k_N4oQGC9UNGUzRh9679A?usp=sharing

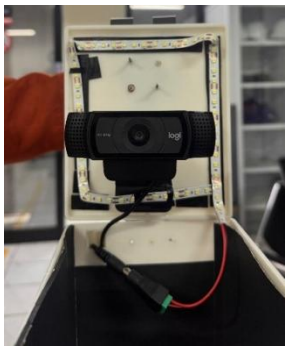


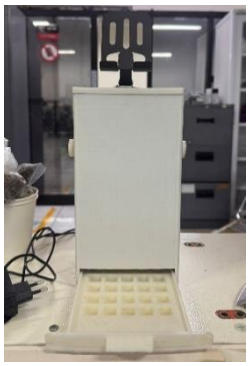
Lampiran 6. Software HMI Pengambilan Data

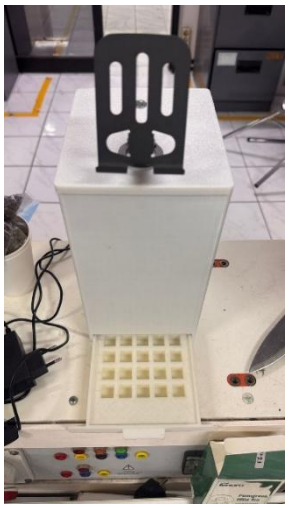
https://drive.google.com/drive/folders/1r9h7U0PlCztNdvJN6IWc38wGlmGOaflv?usp=sharing

guidesign.py	saya	23 Des 2025 saya	9 KB
main.py	saya	21 Des 2025 saya	19 KB
popwindow.py	saya	12 Des 2025 saya	4 KB
roi_cell.json	saya	12 Des 2025 saya	2 KB
roi_cell.py	saya	12 Des 2025 saya	836 byte
roicalibration.py	saya	12 Des 2025 saya	2 KB
snapshot_20251212_203216.jpg	saya	12 Des 2025 saya	178 KB
snapshot_20251221_205451.jpg	saya	21 Des 2025 saya	378 KB
summary.py	saya	12 Des 2025 saya	2 KB

Lampiran 7. Perancangan Media Skenario Deteksi ‘







Lampiran 8. Coding Train YOLOv8m

```
from ultralytics import YOLO
import torch, os, shutil
import pandas as pd
from datetime import datetime

# CONFIG
```

```
DATA_PATH = r"D:\TA NEW DESEMBER\Dataset Jagung Rafdan\data.yaml"
MODEL = "yolov8m.pt"
EPOCHS = 200
BATCH = 4
IMG_SIZE = 640

PROJECT_DIR = r"D:/TA NEW DESEMBER/YOLOv8m"
RUN_NAME = "YOLOv8m_Train"

RESULT_CSV_DEST = r"D:/TA NEW DESEMBER/YOLOv8m/results.csv"
SUMMARY_XLSX = r"D:/TA NEW DESEMBER/YOLOv8m/Epoch_Summary.xlsx"

# GPU CHECK
if torch.cuda.is_available():
    print(f" GPU: {torch.cuda.get_device_name(0)}")
else:
    print(" GPU tidak terdeteksi, training berjalan di CPU")

# TRAINING
model = YOLO(MODEL)
start = datetime.now()
print(f"\n Training dimulai: {start}\n")

results = model.train(
    data=DATA_PATH,
    epochs=EPOCHS,
    imgsz=IMG_SIZE,
    batch=BATCH,
    device=0,
    save=True,
    save_period=20,
    workers=0,
    project=PROJECT_DIR,
    name=RUN_NAME,
    exist_ok=True
)

end = datetime.now()
print(f"\n Training selesai dalam: {end - start}")

# COPY RESULTS.CSV
src_csv = os.path.join(PROJECT_DIR, RUN_NAME, "results.csv")

if os.path.exists(src_csv):
    shutil.copy(src_csv, RESULT_CSV_DEST)
    print(f" results.csv disalin ke: {RESULT_CSV_DEST}")
else:
    print(" results.csv tidak ditemukan! Training belum selesai total.")

# SUMMARY SETIAP 20 EPOCH
```

```
if os.path.exists(RESULT_CSV_DEST):
    df = pd.read_csv(RESULT_CSV_DEST)
    df20 = df[df["epoch"] % 20 == 0]
    df20.to_excel(SUMMARY_XLSX, index=False)
    print(f" Summary kelipatan 20 epoch disimpan di: {SUMMARY_XLSX}")
else:
    print(" CSV tidak ditemukan — summary tidak dibuat.")
```

Lampiran 9. Coding Pencarian Hyperparameter pada YOLOv8m

```
import os
import time
import json
import math
import random
import shutil
import datetime as dt
from pathlib import Path
import pandas as pd
import torch
from ultralytics import YOLO

# Path dataset dan output
DATA = r"D:\TA NEW DESEMBER\Dataset Jagung Rafdan\data.yaml"
MODEL = "yolov8m.pt"
SAVE_DIR = r"D:\TA NEW DESEMBER\YOLOv8m + PSO\Output"
IMG_SIZE = 640
BATCH = 2

# Parameter PSO
NUM_PARTICLES = 15
NUM_ITERS = 5
EPOCHS_PER_RUN = 10

# Range hyperparameter yang mau dicari
BOUNDS = {
    "lr0": (0.001, 0.01),
    "momentum": (0.90, 0.98),
    "weight_decay": (5e-5, 1e-3),
}

# Parameter PSO standar
W_MIN = 0.4
W_MAX = 0.9
C1 = 2.0
C2 = 2.0
RSEED = 42
```

```
def buat_folder(path):
    Path(path).mkdir(parents=True, exist_ok=True)

def batasi_nilai(val, minimum, maksimum):
    if val < minimum:
        return minimum
    if val > maksimum:
        return maksimum
    return val

def timestamp():
    return dt.datetime.now().strftime("%Y%m%d_%H%M%S")

def baca_hasil_csv(folder_run):
    # Baca metrics dari hasil training
    csv_path = folder_run / "results.csv"
    if not csv_path.exists():
        raise FileNotFoundError(f"File tidak ditemukan: {csv_path}")

    df = pd.read_csv(csv_path)
    df.columns = df.columns.str.strip()

    if len(df) == 0:
        raise ValueError("CSV kosong")

    baris_terakhir = df.iloc[-1].to_dict()

    def ambil_nilai(nama_kolom, default=0.0):
        if nama_kolom in baris_terakhir and pd.notna(baris_terakhir[nama_kolom]):
            return float(baris_terakhir[nama_kolom])
        return default

    return {
        "mAP50": ambil_nilai("metrics/mAP50(B)", 0.0),
        "mAP50_95": ambil_nilai("metrics/mAP50-95(B)", 0.0),
        "precision": ambil_nilai("metrics/precision(B)", 0.0),
        "recall": ambil_nilai("metrics/recall(B)", 0.0),
    }

def random_antara(minimum, maksimum):
    return random.random() * (maksimum - minimum) + minimum

def buat_partikel_baru(bounds):
    # Inisialisasi posisi random
    posisi = {}
    for param, (min_val, max_val) in bounds.items():
        posisi[param] = random_antara(min_val, max_val)

    # Velocity awal = 0
```

```
    velocity = {k: 0.0 for k in posisi.keys()}

    return {
        "pos": posisi,
        "vel": velocity,
        "pbest_pos": posisi.copy(),
        "pbest_fit": math.inf
    }

def update_partikel(partikel, gbest_pos, bounds, w):
    # Update velocity dan posisi sesuai rumus PSO
    for param in partikel["pos"].keys():
        r1 = random.random()
        r2 = random.random()

        # Rumus velocity PSO
        vel_baru = (w * partikel["vel"][param] +
                    C1 * r1 * (partikel["pbest_pos"][param] - partikel["pos"][param]) +
                    C2 * r2 * (gbest_pos[param] - partikel["pos"][param]))

        # Update posisi
        pos_baru = partikel["pos"][param] + vel_baru

        # Pastikan dalam batas
        min_bound, max_bound = bounds[param]
        pos_baru = batasi_nilai(pos_baru, min_bound, max_bound)

        partikel["vel"][param] = vel_baru
        partikel["pos"][param] = pos_baru

def hitung_fitness(metrics):
    # Fitness = 1 - mAP50, jadi semakin kecil semakin bagus
    return 1.0 - float(metrics.get("mAP50", 0.0))

def bersihkan_gpu():
    if torch.cuda.is_available():
        torch.cuda.empty_cache()
        torch.cuda.synchronize()

def main():
    random.seed(RSEED)
    buat_folder(SAVE_DIR)
    ROOT = Path(SAVE_DIR)

    nama_session = f"PSO_10P5I_{timestamp()}"
    buat_folder(ROOT / nama_session)

    # File output
    file_csv = ROOT / nama_session / "PSO_log.csv"
    file_excel = ROOT / nama_session / "PSO_log.xlsx"
```

```
file_json = ROOT / nama_session / "best_params.json"
file_txt = ROOT / nama_session / "best_params.txt"

# Buat swarm (kumpulan partikel)
swarm = []
for i in range(NUM_PARTICLES):
    swarm.append(buat_partikel_baru(BOUNDS))

# Global best
gbest_pos = swarm[0]["pos"].copy()
gbest_fit = math.inf
gbest_metrics = None
gbest_folder = None

log_semua = []

total_run = NUM_PARTICLES * NUM_ITERS
waktu_per_run = 1.0  # jam
total_jam = total_run * waktu_per_run

print("="*70)
print("  PSO untuk YOLOv8m")
print("="*70)
print(f"Dataset: 1,159 train | 292 val")
print(f"Partikel: {NUM_PARTICLES}")
print(f"Iterasi: {NUM_ITERS}")
print(f"Epochs per run: {EPOCHS_PER_RUN}")
print(f"Total run: {total_run}")
print(f"Estimasi waktu: {total_jam:.0f} jam ({total_jam/24:.1f} hari)")
print(f"\nMulai: {dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
print(f"Output: {ROOT / nama_session}")
print("="*70)

waktu_mulai = time.time()

# Loop iterasi PSO
for iterasi in range(1, NUM_ITERS + 1):
    # Hitung inertia weight (menurun linear)
    w = W_MAX - (W_MAX - W_MIN) * ((iterasi - 1) / NUM_ITERS)

    print(f"\n{'='*70}")
    print(f"  Iterasi {iterasi}/{NUM_ITERS} (w={w:.4f})")
    print(f"{'='*70}")

    # Loop partikel
    for idx_partikel, partikel in enumerate(swarm, start=1):
        lr0 = partikel["pos"]["lr0"]
        momentum = partikel["pos"]["momentum"]
        weight_decay = partikel["pos"]["weight_decay"]
```

```
        nama_run = f"{nama_session}_it{iterasi:02d}_p{idx_partikel:02d}"
        folder_run = ROOT / nama_run

        # Hapus folder kalau ada
        if folder_run.exists():
            shutil.rmtree(folder_run, ignore_errors=True)
            time.sleep(1)

        run_ke = (iterasi - 1) * NUM_PARTICLES + idx_partikel
        persen = (run_ke / total_run) * 100

        print(f"\nPartikel {idx_partikel}/{NUM_PARTICLES} [Run {run_ke}/{total_run} -
{persen:.1f}%]")
                                        print(f"lr0={lr0:.6f},       momentum={momentum:.6f},
weight_decay={weight_decay:.6f}")

        bersihkan_gpu()

        waktu_run = time.time()

        try:
            model = YOLO(MODEL)

            hasil = model.train(
                data=DATA,
                epochs=EPOCHS_PER_RUN,
                imgsz=IMG_SIZE,
                batch=BATCH,
                lr0=lr0,
                momentum=momentum,
                weight_decay=weight_decay,
                workers=0,
                project=SAVE_DIR,
                name=nama_run,
                exist_ok=False,
                verbose=False,
                cache=False,
                device=0,
                save=True,
                save_period=-1,
                plots=False,
                val=True,
                patience=50,
                pretrained=True,
                optimizer='auto',
            )

            metrics = baca_hasil_csv(folder_run)
            fitness = hitung_fitness(metrics)
            durasi = time.time() - waktu_run
```

```
            # Update personal best
            if fitness < partikel["pbest_fit"]:
                partikel["pbest_fit"] = fitness
                partikel["pbest_pos"] = partikel["pos"].copy()
                print(f"-> NEW PBEST partikel {idx_partikel}")

            # Update global best
            if fitness < gbest_fit:
                gbest_fit = fitness
                gbest_pos = partikel["pos"].copy()
                gbest_metrics = metrics.copy()
                gbest_folder = str(folder_run)
                                        print(f"->  NEW  GBEST!  fitness={gbest_fit:.6f},
mAP50={metrics['mAP50']:.4f}")

            # Hitung F1-score
            P = float(metrics.get("precision", 0.0))
            R = float(metrics.get("recall", 0.0))
            if (P + R) > 0:
                F1 = (2 * P * R) / (P + R)
            else:
                F1 = 0.0

            # Simpan log
            log_semua.append({
                "iter": iterasi,
                "particle": idx_partikel,
                "batch": BATCH,
                "lr0": lr0,
                "momentum": momentum,
                "weight_decay": weight_decay,
                "mAP50": metrics.get("mAP50", 0.0),
                "mAP50_95": metrics.get("mAP50_95", 0.0),
                "precision": P,
                "recall": R,
                "F1": F1,
                "fitness(1-mAP50)": fitness,
                "duration_sec": round(durasi, 2),
                "duration_min": round(durasi / 60, 2),
                "run_dir": str(folder_run),
            })

           print(f"mAP50={metrics.get('mAP50',0):.4f} | P={P:.4f} R={R:.4f} F1={F1:.4f}")
            print(f"Fitness={fitness:.6f} | Durasi={durasi/60:.1f} menit")

            # Bersihkan memory
            del model
            del hasil
            import gc
```

```
            gc.collect()
            bersihkan_gpu()

        except Exception as e:
            durasi = time.time() - waktu_run
            print(f"ERROR: {e}")

            log_semua.append({
                "iter": iterasi,
                "particle": idx_partikel,
                "batch": BATCH,
                "lr0": lr0,
                "momentum": momentum,
                "weight_decay": weight_decay,
                "mAP50": 0.0,
                "mAP50_95": 0.0,
                "precision": 0.0,
                "recall": 0.0,
                "F1": 0.0,
                "fitness(1-mAP50)": math.inf,
                "duration_sec": round(durasi, 2),
                "duration_min": round(durasi / 60, 2),
                "run_dir": str(folder_run),
                "error": str(e)
            })

    # Update posisi semua partikel
    print("\nUpdate posisi swarm...")
    for partikel in swarm:
        update_partikel(partikel, gbest_pos, BOUNDS, w)

    # Cek diversity (seberapa menyebar partikel)
    import numpy as np
    div_lr0 = np.std([p["pos"]["lr0"] for p in swarm])
    div_mom = np.std([p["pos"]["momentum"] for p in swarm])
    div_wd = np.std([p["pos"]["weight_decay"] for p in swarm])

                    print(f"Diversity:    lr0={div_lr0:.6f},    momentum={div_mom:.6f},
weight_decay={div_wd:.6f}")

    if div_lr0 < 0.001 and div_mom < 0.005 and div_wd < 0.00005:
        print("WARNING: Swarm terlalu konvergen!")

    # Simpan log tiap iterasi
    df = pd.DataFrame(log_semua)
    df.to_csv(file_csv, index=False)
    with pd.ExcelWriter(file_excel, engine="openpyxl") as writer:
        df.to_excel(writer, index=False, sheet_name="PSO_LOG")

    # Hitung estimasi waktu
```

```
        waktu_jalan = (time.time() - waktu_mulai) / 3600
        run_selesai = iterasi * NUM_PARTICLES
        run_tersisa = total_run - run_selesai
        rata_per_run = waktu_jalan / run_selesai if run_selesai > 0 else waktu_per_run
        sisa_jam = run_tersisa * rata_per_run
        selesai_di = dt.datetime.now() + dt.timedelta(hours=sisa_jam)

        print(f"\nGBest saat ini:")
                print(f"lr0={gbest_pos['lr0']:.6f},    momentum={gbest_pos['momentum']:.6f},
wd={gbest_pos['weight_decay']:.6f}")
        print(f"fitness={gbest_fit:.6f}, mAP50={(gbest_metrics or {}).get('mAP50',0):.4f}")
        print(f"Waktu berjalan: {waktu_jalan:.1f} jam")
        print(f"Sisa estimasi: {sisa_jam:.1f} jam")
        print(f"Perkiraan selesai: {selesai_di.strftime('%Y-%m-%d %H:%M:%S')}")

    # Simpan hasil terbaik
    total_durasi = time.time() - waktu_mulai
    hasil_terbaik = {
        "best_hyperparams": {
            "lr0": round(gbest_pos["lr0"], 8),
            "momentum": round(gbest_pos["momentum"], 8),
            "weight_decay": round(gbest_pos["weight_decay"], 8),
        },
        "best_metrics": gbest_metrics,
        "best_fitness(1-mAP50)": gbest_fit,
        "best_run_dir": gbest_folder,
        "total_duration_hours": round(total_durasi / 3600, 2),
        "dataset_info": {
            "total_images": 1702,
            "train": 1159,
            "val": 292,
            "test": 251,
            "classes": ["Baik (725)", "Buruk (977)"]
        },
        "config": {
            "num_particles": NUM_PARTICLES,
            "num_iters": NUM_ITERS,
            "epochs_per_run": EPOCHS_PER_RUN,
            "bounds": BOUNDS
        },
    }

    with open(file_json, "w", encoding="utf-8") as f:
        json.dump(hasil_terbaik, f, indent=2)

    with open(file_txt, "w", encoding="utf-8") as f:
        f.write(json.dumps(hasil_terbaik, indent=2))

    print("\n" + "="*70)
    print("PSO SELESAI")
```

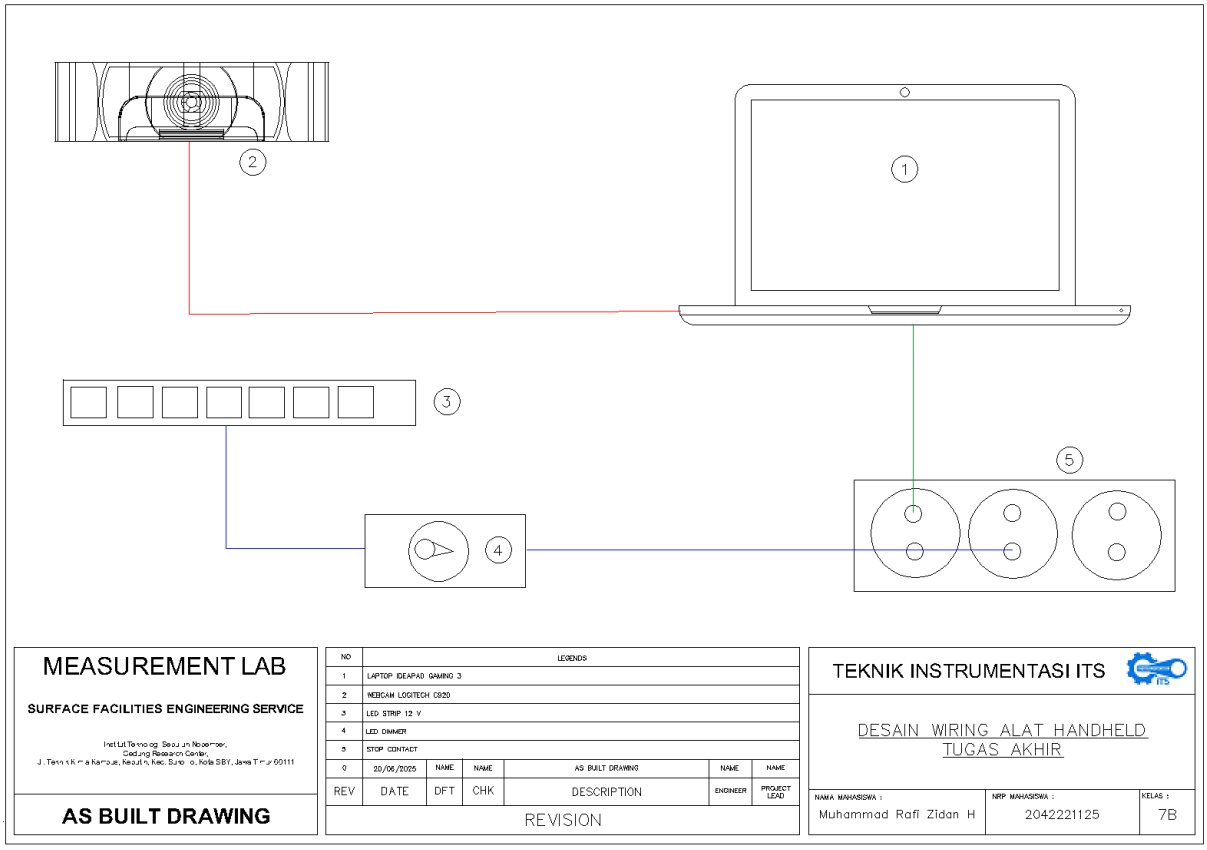
```
    print("="*70)
    print(f"Best hyperparams: {hasil_terbaik['best_hyperparams']}")
    print(f"Best metrics: {hasil_terbaik['best_metrics']}")
    print(f"Best fitness: {hasil_terbaik['best_fitness(1-mAP50)']:.6f}")
    print(f"Total durasi: {hasil_terbaik['total_duration_hours']:.2f} jam")
    print(f"\nFile log: {file_csv}")
    print(f"File excel: {file_excel}")
    print(f"Best params: {file_json}")

if __name__ == "__main__":
    main()
```

Lampiran 10. Coding Train Final menggunakan Hyperparameter terbaik YOLOv8m

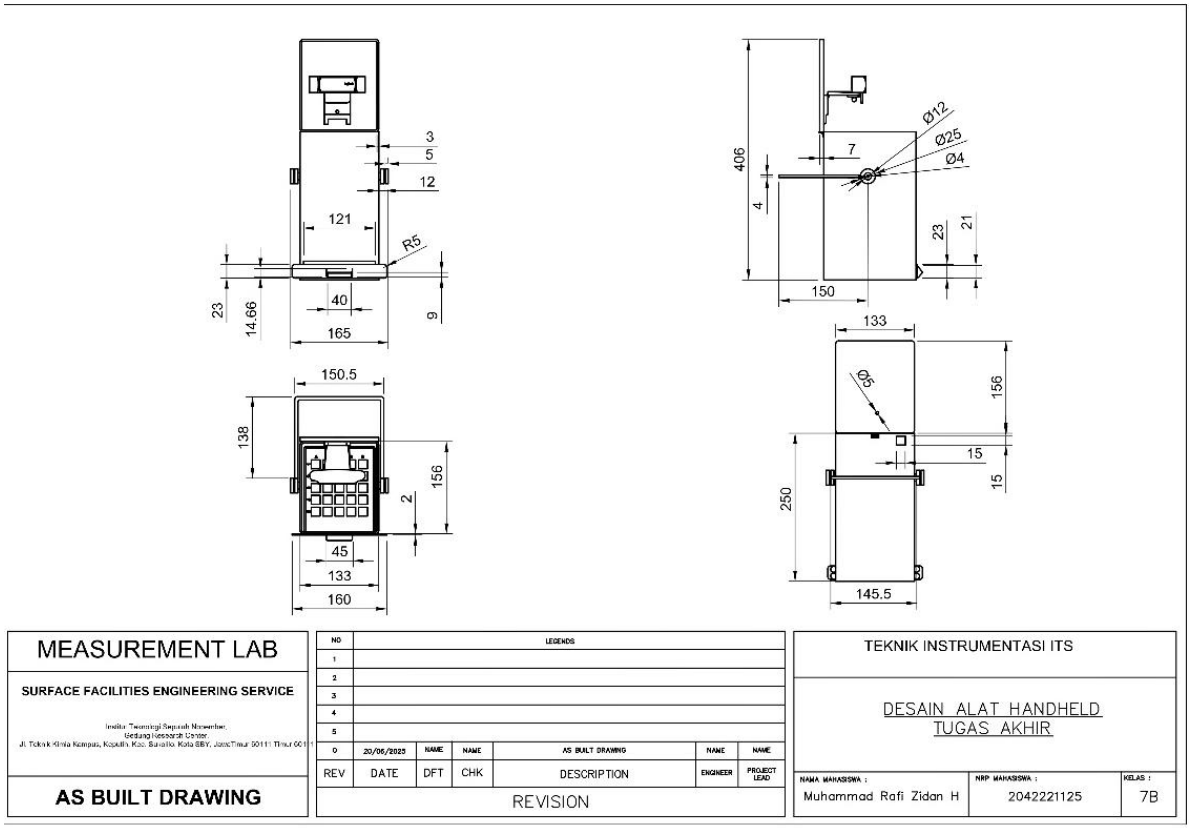
```
from ultralytics import YOLO
import torch

# KONFIGURASI

DATA = r"D:\TA NEW DESEMBER\Dataset Jagung Rafdan\data.yaml"
MODEL = "yolov8m.pt"
PROJECT_DIR = r"D:\TA NEW DESEMBER\YOLOv8m + PSO\Train Final Menggunakan gBest"
RUN_NAME = "YOLOv8_PSO_Final"

# KONFIGURASI TRAINING

EPOCHS   = 200
IMG_SIZE = 640
BATCH    = 4
WORKERS  = 0

# HASIL TERBAIK DARI PSO

LR0          = 0.01
MOMENTUM     = 0.958917697
WEIGHT_DECAY  = 0.000692865

# VALIDASI GPU

assert torch.cuda.is_available(), ""
print("GPU Aktif:", torch.cuda.get_device_name(0))

# LOAD MODEL

model = YOLO(MODEL)

# TRAINING

model.train(
```

```
    data=DATA,
    epochs=EPOCHS,
    imgsz=IMG_SIZE,
    batch=BATCH,

    #  PSO PARAMETERS
    optimizer="SGD",
    lr0=LR0,
    momentum=MOMENTUM,
    weight_decay=WEIGHT_DECAY,

    #  GPU
    device=0,

    #  LOG & CHECKPOINT
    project=PROJECT_DIR,
    name=RUN_NAME,
    save=True,
    save_period=1,
    verbose=False,
    plots=True,

    # LAINNYA
    workers=WORKERS,
    cache=False,
)
```

Lampiran 11. Wiring Alat



Lampiran 12. Desain Alat Pengambilan Data



Halaman ini sengaja dikosongkan.

BIODATA PENULIS

Muhammad Rafi Zidan Hilmi, lahir di kota Gresik pada 20 Juni 2004. Mahasiswa Departement Teknik Instrumentasi Fakultas Vokasi Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2022-2026. Aktif dalam mengembangkan ilmu hardskill dan softskill. Dalam mengembangkan ilmu softskill penulis aktif dalam beberapa organisasi, penulis pernah menjabat Ketua Komisi Kontrol. Teknik Instrumentasi kabinet Sinergi Aksara periode 2024 DPA HIMATEKINS. Dalam menunjang hardskill, penulis menjabat sebagai staff Labolatorium Instrumentasi Control. Sebagai penutup, penulis mengucapkan syukur kepada Tuhan Yang Maha Esa karena dengan berkatnya Laporan Proyek Akhir dapat diselesaikan.