

BAB II DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian mengenai penerapan *computer vision* pada bidang pertanian terus berkembang sebagai solusi untuk meningkatkan efisiensi proses inspeksi dan klasifikasi hasil panen, khususnya pada sistem klasifikasi tingkat kematangan buah berbasis *deep learning*. Rifatatiar mengembangkan sistem klasifikasi tingkat kematangan apel berbasis YOLOv8 yang diintegrasikan dengan mekanisme *conveyor* dan aktuator servo sebagai sistem penyortiran otomatis, serta diimplementasikan pada Raspberry Pi 5 dengan akselerator Hailo-8 sehingga kecepatan inferensi meningkat dari 11 FPS menjadi 30 FPS. Optimasi hiperparameter pada penelitian tersebut dilakukan menggunakan metode *One Factor At a Time* (OFAT), yaitu dengan mengubah satu hiperparameter secara bertahap pada rentang *learning rate*, *batch size*, *optimizer*, *weight decay*, dan jumlah *epoch*, sementara hiperparameter lainnya dipertahankan tetap. Proses pencarian hiperparameter dengan metode OFAT ini memakan total waktu pelatihan (*tuning time*) sekitar 8 jam pada GPU Tesla T4. Melalui pendekatan tersebut, diperoleh konfigurasi *optimizer* SGD, *learning rate* 0,01, *batch size* 16, *weight decay* 0,0005, dan *epoch* 200, yang menghasilkan *precision* sebesar 0,982, *recall* 0,998, *F1-score* 0,990, serta *mAP@0,50* sebesar 0,991. Meskipun demikian, proses pemasukan apel ke dalam *conveyor* pada penelitian tersebut masih dilakukan secara manual oleh operator, serta pencarian hiperparameter yang memakan waktu cukup lama dan dilakukan secara bertahap tersebut belum mengeksplorasi interaksi antarparameter secara langsung.

Keterbatasan pendekatan pencarian hiperparameter secara bertahap juga teridentifikasi pada penelitian di luar domain pertanian buah. Ramos et al. melakukan optimasi hiperparameter pada model YOLOv8l untuk deteksi asap dan kebakaran hutan, dengan menerapkan metode OFAT pada tahap awal untuk mengidentifikasi hiperparameter yang paling berpengaruh, yaitu *learning rate*, *optimizer*, *batch size*, *weight decay*, dan *epoch*. Hasil OFAT tersebut kemudian

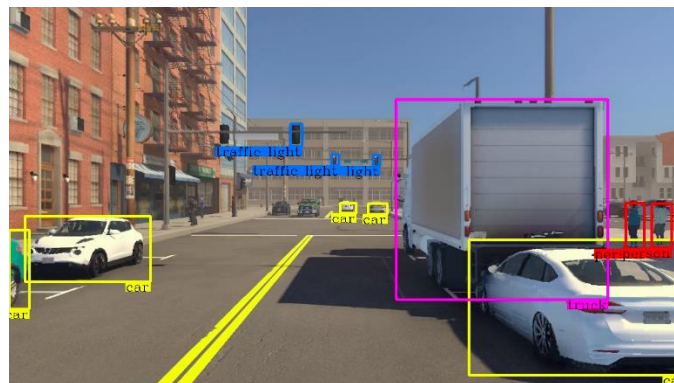
dijadikan dasar untuk menentukan ruang pencarian pada tahap Random Search (RS), karena OFAT dinilai belum mampu mengeksplorasi kombinasi antarhiperparameter secara menyeluruh. Model akhir hasil RS menunjukkan peningkatan precision sebesar 1,39%, recall sebesar 1,48%, F1-score sebesar 1,44%, $mAP@0,50$ sebesar 0,70%, dan $mAP@0,50:0,95$ sebesar 5,09% dibandingkan model hasil fine-tuning awal. Temuan ini menunjukkan bahwa meskipun OFAT dapat mengidentifikasi pengaruh masing-masing hiperparameter secara individual, Temuan tersebut menunjukkan bahwa keterbatasan OFAT dalam mengeksplorasi kombinasi hiperparameter tidak hanya ditemukan pada penelitian klasifikasi kematangan apel, tetapi juga pada penerapan YOLOv8 di domain lain. [8].

Perkembangan metode optimasi hiperparameter turut mendorong penggunaan pendekatan berbasis Bayesian Optimization yang dapat melakukan pencarian secara otomatis dan memanfaatkan hasil percobaan sebelumnya, salah satunya Tree-Structured Parzen Estimator (TPE) yang diimplementasikan melalui framework Optuna. Ratih dan Wijaya menerapkan Optuna pada model YOLOv8 untuk klasifikasi mahasiswa dan dosen, dengan mengoptimasi hiperparameter learning rate, momentum, dan weight decay secara otomatis berbasis TPE. Hasil penelitian menunjukkan bahwa optimasi tersebut meningkatkan $mAP@0,50:0,95$ dari 67,6% menjadi 68,8%, meskipun $mAP@0,50$ mengalami sedikit penurunan dari 92,8% menjadi 91,5%. Hasil ini mengindikasikan bahwa pencarian hiperparameter secara otomatis melalui TPE mampu mengarahkan model pada konfigurasi yang menghasilkan lokalisasi objek yang lebih presisi pada berbagai ambang batas IoU, meskipun terdapat trade-off pada metrik tertentu, sehingga pendekatan ini terbukti mampu memperoleh kombinasi hiperparameter yang lebih optimal dibandingkan proses tuning manual maupun bertahap [55].

Berdasarkan berbagai penelitian tersebut, dapat disimpulkan bahwa algoritma YOLOv8 telah terbukti memiliki performa yang baik untuk berbagai aplikasi computer vision, termasuk pada klasifikasi tingkat kematangan apel, dan bahwa optimasi hiperparameter memberikan pengaruh yang signifikan terhadap

peningkatan performa model. Namun, metode OFAT yang digunakan pada penelitian sebelumnya oleh Rifatatiar [4] memiliki keterbatasan yang sejalan dengan temuan Ramos et al. [8], yaitu kurang mampu mengeksplorasi interaksi antarhiperparameter secara langsung, sementara pendekatan berbasis TPE melalui Optuna sebagaimana diterapkan oleh Ratih dan Wijaya [55] menunjukkan potensi untuk mengatasi keterbatasan tersebut melalui pencarian yang lebih sistematis dan otomatis. Oleh karena itu, penelitian ini merupakan pengembangan lanjutan dari sistem yang dibangun oleh Rifatatiar [4], dengan mengganti metode optimasi hiperparameter dari OFAT menjadi Tree-Structured Parzen Estimator (TPE) melalui framework Optuna, serta menambahkan modul loading otomatis, untuk menghasilkan sistem klasifikasi tingkat kematangan apel yang lebih optimal dan lebih mendekati kebutuhan implementasi pada lingkungan industri.

2.2 Object Detection



Gambar 2. 1 *Object Detection* [80]

Object detection adalah salah satu bidang utama dalam *computer vision* yang bertujuan untuk mengidentifikasi serta menentukan lokasi objek dalam suatu gambar atau video. Proses ini mencakup dua tugas utama, yaitu *object localization* yang menentukan posisi objek dan *object classification* yang mengidentifikasi kategori dari objek tersebut. Teknologi ini memiliki berbagai aplikasi, termasuk pengenalan wajah, analisis perilaku manusia, kendaraan otonom, dan sistem keamanan. Dalam otomasi pergudangan, deteksi objek dapat digunakan untuk mengidentifikasi berbagai elemen seperti rak, kargo, kardus, pekerja, dan benda

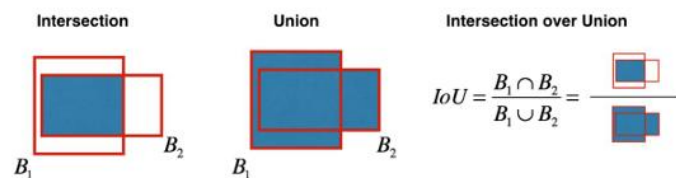
asing. Hal ini membantu sistem pendeteksian robot dalam mendeteksi hambatan di jalur pemindahan barang atau rak, sehingga meningkatkan efektivitas dan efisiensi kerja robot [18] [19].

Algoritma *object detection* membangun model berdasarkan contoh pelatihan untuk mengenali jenis objek tertentu. Jika objeknya tetap dan memiliki bentuk kaku, satu contoh mungkin cukup. Namun, dalam banyak kasus, dibutuhkan ratusan hingga ribuan contoh agar model dapat memahami variasi dalam suatu kelas objek. Semakin banyak informasi tentang perbedaan dalam suatu kelas yang dimasukkan langsung ke dalam model, semakin sedikit data pelatihan yang diperlukan [20]. Secara umum, terdapat dua pendekatan utama dalam *object detection*, yaitu metode tradisional dan metode berbasis *deep learning*. Metode tradisional seperti *Histogram of Oriented Gradients* (HOG) dan *Support Vector Machines* (SVM) masih digunakan hingga kini karena kesederhanaannya, meskipun memiliki keterbatasan dalam menangani variasi objek dan kondisi pencahayaan. Selain itu, ada juga metode seperti *Deformable Part-based Models* (DPM) dan *AdaBoost*, yang menggabungkan berbagai fitur untuk meningkatkan akurasi deteksi [21]. Namun, metode-metode ini cenderung memiliki performa yang lebih rendah dibandingkan pendekatan berbasis *deep learning*, terutama dalam lingkungan yang kompleks dan dinamis.

Di sisi lain, metode *deep learning* telah mengalami perkembangan pesat dan semakin mendominasi dalam deteksi objek berkat akurasi tinggi serta peningkatan kecepatan pemrosesan. Beberapa teknik yang sering digunakan dalam kategori ini meliputi R-CNN, SSD, Faster R-CNN, serta YOLO. Salah satu keunggulan utama dari pendekatan *deep learning* adalah kemampuannya dalam mengurangi kebutuhan jumlah data pelatihan dengan mengintegrasikan fitur secara eksplisit ke dalam model [20]. Namun, meskipun kemajuan ini menjanjikan, masih terdapat tantangan yang harus diatasi. Beberapa tantangan utama dalam *object detection* meliputi ketergantungan pada kualitas data pelatihan, perubahan pencahayaan dan perspektif yang dapat mempengaruhi akurasi, serta kebutuhan daya komputasi yang tinggi untuk menjalankan model *deep learning* dalam aplikasi real-time.

2.2.1 Intersection Over Union (IoU)

Intersection Over Union (IoU) adalah metrik evaluasi yang digunakan untuk mengukur akurasi prediksi bounding box dalam deteksi objek dengan membandingkan area tumpang tindih antara bounding box hasil prediksi model dan bounding box ground truth (kebenaran dasar). Secara matematis, IoU dihitung dengan membagi luas area perpotongan (intersection) dari kedua bounding box dengan luas area gabungan (union) dari keduanya:



Gambar 2. 2 *IoU Box [81]*

Nilai IoU berkisar antara 0 hingga 1, di mana nilai 1 menunjukkan tumpang tindih sempurna antara prediksi dan ground truth, sedangkan nilai 0 berarti tidak ada tumpang tindih sama sekali[33]. Nilai ambang batas IoU sering digunakan untuk menentukan apakah sebuah prediksi dianggap sebagai deteksi yang benar (true positive) atau tidak. Misalnya, jika IoU antara prediksi dan ground truth lebih besar dari 0,5 atau 0,7, prediksi tersebut dianggap berhasil mendeteksi objek yang benar.

2.3 Deep Learning

Deep Learning telah menjadi pendekatan utama dalam pengembangan sistem deteksi objek modern karena kemampuannya menangani data visual yang kompleks dan menghasilkan akurasi tinggi. Metode ini menggunakan arsitektur Convolutional Neural Networks (CNN), yang mampu mengekstrak fitur hierarkis dari gambar melalui lapisan konvolusi bertingkat. CNN memungkinkan deteksi pola lokal seperti tepi atau tekstur pada lapisan awal, pengenalan bentuk kompleks pada lapisan lebih dalam, serta otomatisasi ekstraksi fitur tanpa memerlukan

intervensi manual. Beberapa algoritma unggulan dalam deteksi objek berbasis Deep Learning mencakup YOLO (You Only Look Once), SSD (Single Shot Detector), dan Faster R-CNN. YOLO, khususnya versi terbaru seperti YOLOv8, mampu mencapai mAP 76-81% dalam deteksi manusia dengan kecepatan hingga 45 fps pada GPU high-end, serta peningkatan akurasi hingga 5% berkat modifikasi arsitektur *head* [22]. SSD dikenal karena keseimbangan antara kecepatan dan akurasi, sehingga cocok untuk aplikasi real-time, sementara Faster R-CNN sangat akurat dalam mendeteksi objek kecil di lingkungan yang kompleks, seperti bawah air, dengan efisiensi tinggi melalui *Region Proposal Network* [23]. Kelebihan utama Deep Learning adalah kemampuannya untuk belajar dari data besar tanpa pemrograman eksplisit, menjadikannya solusi unggul dalam berbagai aplikasi, mulai dari pengenalan wajah hingga analisis gambar medis. Namun, tantangan seperti kebutuhan dataset berkualitas tinggi dan sumber daya komputasi yang besar tetap menjadi perhatian utama.

2.3.1 Hyperparamater

Hyperparameter merupakan komponen fundamental yang menentukan dinamika pembelajaran pada model deep learning. Pada deteksi apel menggunakan YOLOv8, optimalisasi hyperparameter menjadi langkah penting untuk mencapai performa terbaik, terutama dalam peningkatan akurasi dan mAP. Karakteristik visual apel—seperti variasi warna, tekstur kulit, tingkat kilau, dan adanya cacat halus—membuat konfigurasi hyperparameter tidak dapat digeneralisasi dan harus disesuaikan secara spesifik. Optimalisasi dilakukan pada hyperparameter yang terbukti memiliki pengaruh langsung terhadap stabilitas optimasi, kualitas konvergensi, dan kemampuan model mengekstraksi fitur penting dari gambar apel, yaitu learning rate, weight decay, batch size, serta jenis optimizer. Pemilihan set hyperparameter ini didasarkan pada hasil-hasil empiris dari banyak studi yang menunjukkan bahwa variabel-variabel tersebut paling menentukan performa pada sistem deteksi berbasis YOLO[55].

Learning rate (lr) dioptimasi karena merupakan pengendali utama besar langkah pembaruan bobot pada fase awal pelatihan. Nilai yang terlalu tinggi membuat model tidak stabil dan cenderung divergen, sedangkan nilai terlalu rendah membuat model bergerak lambat menuju optimum. Weight decay berfungsi sebagai regularisasi yang mengurangi risiko overfitting, terutama karena tekstur apel bisa memiliki noise halus yang tidak relevan, sehingga regularisasi yang baik membantu model fokus pada pola struktural yang lebih stabil[56].

Batch size juga dioptimasi dalam rentang terbatas, yaitu 16, 32, dan 64. Batch size dipertahankan dalam pilihan diskrit karena ukuran batch memengaruhi stabilitas estimasi gradien, tingkat regularisasi implicit, serta konsumsi memori GPU. Pemilihan rentang ini mempertimbangkan batas memori yang umum digunakan dalam penelitian YOLO serta konsistensi pelatihan. Batch yang terlalu kecil dapat menghasilkan gradien yang sangat bising, sementara batch terlalu besar memperlambat generalisasi. Oleh karena itu, pemilihan 16, 32, dan 64 merupakan kompromi ideal untuk eksplorasi performa tanpa memperluas ruang pencarian secara berlebihan[58].

Jenis optimizer, juga dimasukkan dalam ruang optimasi. Sebagai contoh, SGD cenderung memberikan generalisasi yang lebih stabil dan sering menghasilkan performa akhir yang baik meskipun lambat di awal. AdamW menawarkan adaptivitas gradien dan konvergensi lebih cepat, tetapi kadang menghasilkan generalisasi yang lebih buruk pada dataset tertentu. Karena karakteristik kedua optimizer ini berbeda secara mendasar dan masing-masing telah digunakan secara ekstensif dalam studi-studi YOLO, maka pemilihan optimizer menjadi variabel penting untuk dioptimalkan dalam konteks deteksi apel[59].

Beberapa *hyperparameter* sengaja tidak disertakan dalam proses optimasi karena tidak memberikan kontribusi informatif yang signifikan bagi algoritma Tree-structured Parzen Estimator (TPE). Jumlah *epoch*,

misalnya, tidak dioptimasi karena peningkatan *epoch* secara deterministik umumnya meningkatkan kualitas model hingga mencapai konvergensi. Dalam konteks hyperparameter optimization, jumlah *epoch* lebih tepat diperlakukan sebagai anggaran pelatihan (training budget) daripada variabel pencarian. Meskipun terdapat studi seperti "One Epoch Is All You Need" yang menunjukkan bahwa satu *epoch* pelatihan dapat mencukupi pada skenario dengan data yang sangat besar. Pada supervised learning, sinyal pembelajaran (seperti mAP dan loss) belum stabil pada *epoch* yang sangat awal karena model memerlukan beberapa iterasi untuk menyesuaikan bobot terhadap label *ground truth*, dan korelasi antara performa awal dengan performa akhir cenderung rendah[75].

Literatur optimasi hyperparameter pada pembelajaran terawasi, khususnya deteksi objek, umumnya menggunakan jumlah *epoch* awal dalam rentang 1–50 untuk evaluasi percobaan, dengan variasi tergantung pada tujuan: *epoch* rendah (1–5) untuk penyaringan kasar dan efisiensi komputasi tinggi, meskipun berisiko noise tinggi. *Epoch* menengah (10–20) sebagai kompromi antara stabilitas sinyal dan biaya, serta *epoch* lebih tinggi (30–50) untuk evaluasi yang lebih akurat pada kandidat terbaik [76][77]. Penggunaan *epoch* yang terlalu sedikit (misalnya 1 *epoch*) dapat menghasilkan metrik validasi yang sangat bising dan tidak representatif terhadap performa akhir model, sementara penggunaan *epoch* yang terlalu besar meningkatkan biaya komputasi secara signifikan dan mengurangi efisiensi proses pencarian[78].

Pelatihan final model dengan konfigurasi hyperparameter terbaik dilakukan menggunakan 50 *epoch* untuk memastikan konvergensi penuh dan evaluasi performa yang maksimal. Image size tidak dioptimasi karena pengaruh utamanya lebih dominan terhadap kecepatan pelatihan dan inferensi dibandingkan akurasi, sementara arsitektur YOLOv8 relatif *robust* terhadap variasi ukuran input.

2.3.2 *Preconditioning*

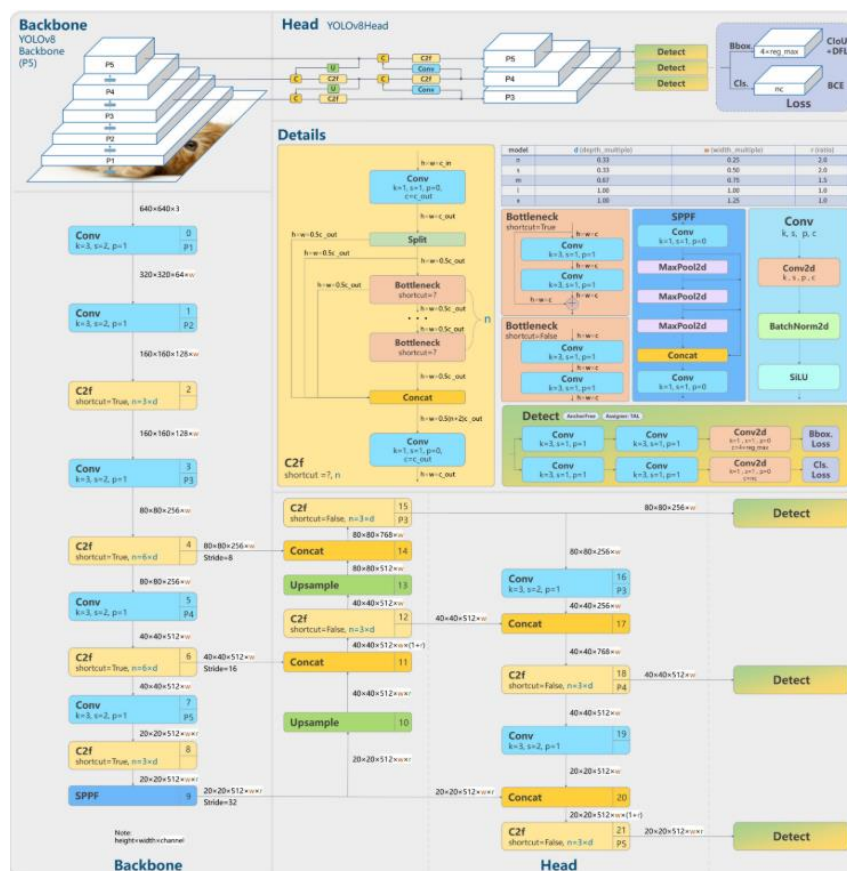
Preconditioning memainkan peran penting dalam *object detection* ini, *Preconditioning* mencakup serangkaian teknik komprehensif yang diterapkan sebelum dan selama pelatihan model untuk meningkatkan kinerja dan akurasi. Data preprocessing membentuk fondasi *Preconditioning* yang efektif, dimulai dengan normalisasi gambar yang menskalakan nilai piksel ke rentang standar (biasanya 0-1), sehingga menghasilkan perhitungan gradien yang lebih stabil dan konvergensi yang lebih cepat. Hal ini dilengkapi dengan strategi augmentasi data yang komprehensif termasuk rotasi, membalik, penskalaan, jitter warna, penyesuaian kecerahan, dan pemangkasan acak-semuanya secara artifisial memperluas kumpulan data pelatihan Anda dan membantu model YOLOv8 menjadi tangguh terhadap variasi tampilan, orientasi, kondisi pencahayaan, dan latar belakang apel. Image quality enhancement techniques semakin memperkuat prapengolahan dengan menerapkan filter pengurangan noise, peningkatan kontras, dan penyeimbangan warna untuk meningkatkan ekstraksi fitur dari gambar apel. Penanganan ketidakseimbangan kelas melalui pengambilan sampel berbobot atau fungsi kehilangan khusus memastikan semua varietas apel dipelajari dengan sama baiknya, terlepas dari representasi mereka dalam kumpulan data. Teknik penyempurnaan bounding box mengoptimalkan aspek pelokalan model Anda, meningkatkan presisi deteksi melalui optimasi anchor box dan penyesuaian berbasis IoU.

Preconditioning arsitektur YOLOv8 sendiri melibatkan inisialisasi transfer learning, di mana bobot yang telah dilatih sebelumnya pada dataset besar memberikan titik awal yang lebih unggul dibandingkan dengan inisialisasi acak. Teknik regularisasi seperti dropout, normalisasi batch, dan peluruhan bobot diintegrasikan secara strategis untuk mencegah overfitting. Terakhir, pengoptimalan *hyperparameter* merupakan elemen prakondisi yang paling penting, di mana parameter seperti laju pembelajaran, ukuran batch, dan *hyperparameter* khusus model disetel secara sistematis untuk

mencapai kinerja yang optimal dalam mendeteksi dan mengelompokkan varietas apel di berbagai kondisi.

2.3.3 YOLOv8

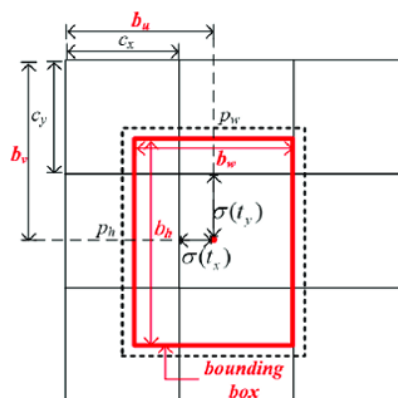
Alih-alih membingkai masalah deteksi objek sebagai tantangan klasifikasi, Redmon memperkenalkan algoritma YOLO (*You Only Look Once*), yang terkenal karena kecepatan deteksinya. YOLO mampu mencapai kecepatan hingga 45 frame per detik (fps) dalam versi standar dan hingga 155 fps dalam versi cepat, dengan efisiensi tinggi dibandingkan metode deteksi objek lainnya[24]. Salah satu versi terbaru dari algoritma ini adalah YOLOv8, yang dikembangkan oleh Ultralytics. YOLOv8 dikenal karena kecepatan dan akurasi yang baik, memungkinkan penerapan yang lebih luas dalam berbagai aplikasi. Arsitektur Yolo dapat divisualisasikan pada gambar 2.3.



Gambar 2. 3 Arsitektur YOLO [82]

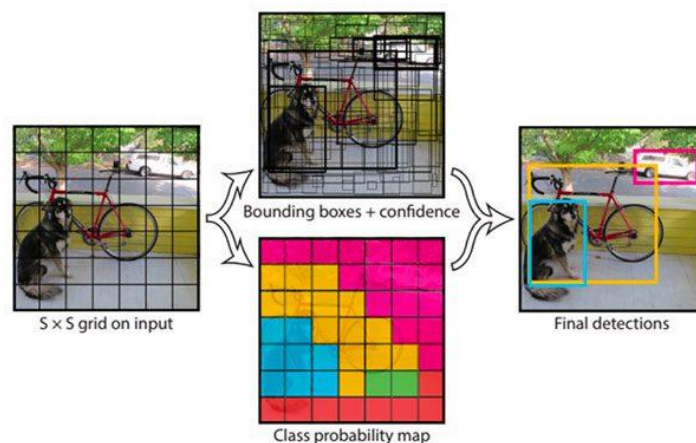
Pada gambar 2.3. YOLOv8 merupakan arsitektur deteksi objek mutakhir yang dirancang untuk menggabungkan efisiensi komputasi dan akurasi tinggi melalui struktur tiga komponen utamanya: Backbone, Neck, dan Head. Backbone YOLOv8 menggunakan CSPDarknet dengan modul C2f yang mengintegrasikan koneksi cross-stage partial untuk mengurangi redundansi komputasi, dilengkapi lapisan SPPF (Spatial Pyramid Pooling Fast) untuk menangani variasi skala objek, serta fungsi aktivasi Mish yang meningkatkan kemampuan ekstraksi fitur hierarkis [25][26].

Proses ini menghasilkan lima level fitur (P1 hingga P5) yang menangkap informasi mulai dari tekstur dasar hingga bentuk objek kompleks. Bagian Neck mengombinasikan teknik PANet dan BiFPN untuk agregasi fitur multi-skala, memadukan informasi dari lapisan resolusi tinggi (detail lokal) dan rendah (konteks global) melalui upsampling dinamis dan concatenation adaptif, sehingga memungkinkan deteksi objek berukuran kecil, sedang, dan besar secara simultan [27]. YOLOv8 juga mendukung berbagai mode operasional, seperti pelatihan pada dataset khusus, validasi model, prediksi gambar atau video baru, ekspor model ke format lain, pelacakan objek secara *real-time*, serta evaluasi kinerja melalui mode *benchmark*. Kombinasi fitur-fitur ini menjadikan YOLOv8 sebagai solusi yang efisien untuk deteksi objek di berbagai kondisi lingkungan [28]



Gambar 2. 4 Bounding Box Deteksi Objek [83]

YOLO menggunakan model terintegrasi untuk mendeteksi objek dengan membuat jaringan saraf tunggal yang memprediksi beberapa *bounding boxes* dan probabilitas kelas di dalamnya secara bersamaan. Sistem YOLO bekerja dengan cara membagi gambar input ke dalam kisi-kisi $S \times S$, di mana salah satu sel *grid* bertugas mendeteksi objek jika pusatnya terletak di dalam sel *grid* tersebut[29]. Setiap sel *grid* memprediksi *bounding box* dan *confidence score*, di mana *confidence score* menunjukkan tingkat kepercayaan dan akurasi model dalam mengidentifikasi keberadaan objek di dalam kotak[29]. Dalam YOLO, seluruh proses deteksi dilakukan dalam satu langkah dengan memproses gambar secara keseluruhan dalam satu forward pass melalui jaringan saraf. YOLO membagi gambar input menjadi grid dan untuk setiap grid cell, model memprediksi bounding box dan probabilitas kelas objek yang terdapat dalam sel tersebut. Setiap bounding box yang diprediksi terdiri dari koordinat pusat, dimensi, dan skor kepercayaan (*confidence score*) yang menunjukkan seberapa yakin model tentang keberadaan objek dalam bounding box tersebut. Model YOLO juga menggabungkan fitur dari berbagai lapisan untuk menangkap detail objek pada berbagai skala yang memungkinkannya untuk mendeteksi objek kecil dan besar dengan baik. Proses deteksi dari model YOLO dapat dilihat pada Gambar 2.4 berikut ini.

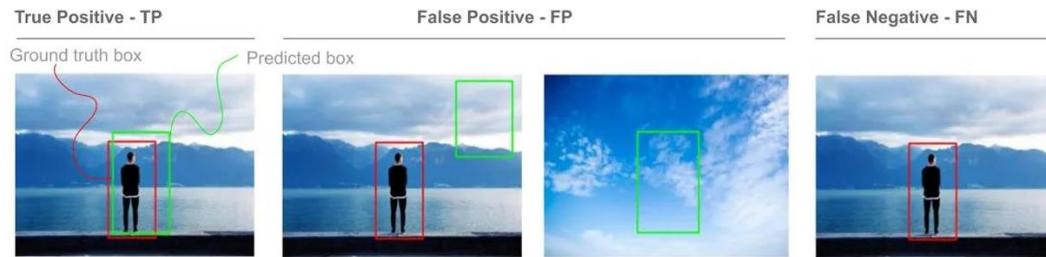


Gambar 2. 5 Output YOLO [84]

2.3.4 *Confusion Matrix*

Matriks konfusi merupakan instrumen evaluasi yang digunakan untuk mengukur efektivitas model klasifikasi dengan cara membandingkan hasil prediksi dengan nilai sebenarnya dari data uji. Matrix ini menunjukkan perbandingan antara klasifikasi sebenarnya dengan klasifikasi yang diprediksi oleh model. Pada deteksi objek, confusion matrix membantu mengevaluasi efektivitas model dalam mendeteksi dan mengklasifikasikan objek dengan benar. Elemen utama dalam confusion matrix untuk object detection meliputi:

- a) True Positive (TP): Deteksi yang benar dari bounding box ground-truth. Ini berarti model berhasil mengidentifikasi dan mengklasifikasikan objek yang benar-benar ada dalam gambar dengan tepat.
- b) False Positive (FP) : Deteksi yang salah dari objek yang tidak ada atau deteksi yang salah tempat dari objek yang sebenarnya ada. Ini menunjukkan kasus di mana model menghasilkan deteksi palsu atau keliru menempatkan bounding box pada objek yang ada.
- c) False Negative (FN) : Bounding box ground-truth yang tidak terdeteksi oleh model. Ini menunjukkan objek yang ada dalam gambar tetapi tidak berhasil dikenali atau dideteksi oleh model.
- d) True Negative (TN) : Dalam deteksi objek, True Negative secara teoritis merepresentasikan area gambar yang benar-benar tidak mengandung objek dan tidak diprediksi sebagai objek. Namun, konsep ini tidak memiliki definisi operasional yang jelas karena tidak ada unit evaluasi yang konsisten (misalnya per piksel, grid, atau bounding box). Selain itu, ruang background yang sangat luas menyebabkan jumlah TN menjadi sangat besar dan tidak informatif. Oleh karena itu, TN tidak digunakan dalam evaluasi deteksi objek, dan metrik seperti precision, recall, serta mean Average Precision (mAP) hanya berfokus pada TP, FP, dan FN.



Gambar 2. 6 *Ilustrasi True Positive (TP), False Positive (FP), dan False Negative (FN) pada deteksi objek [85]*

Dalam deteksi objek, hasil True Negative (TN) tidak berlaku karena terdapat jumlah bounding box yang tidak terdeteksi dalam gambar yang tidak terbatas. Dengan kata lain, tidak mungkin untuk mengidentifikasi semua area dalam gambar yang tidak mengandung objek sebagai True Negative karena jumlah potensi area yang harus dipertimbangkan adalah tak terhingga[30]. Dengan menganalisis TP, FP, dan FN, kita dapat mengukur seberapa baik model mendeteksi objek yang ada, mengidentifikasi kesalahan dalam deteksi, dan meningkatkan performa model untuk mengurangi deteksi yang salah serta meningkatkan kemampuan deteksi objek yang sebenarnya ada.

a) Presisi

Presisi adalah metrik yang mengukur kemampuan model untuk hanya mengidentifikasi objek yang relevan. Dalam deteksi objek, presisi menunjukkan persentase prediksi positif yang benar dari total prediksi positif yang diklaim oleh model [31]. Secara matematis, presisi dihitung dengan rumus:

$$\text{Presisi} = \frac{TP}{TP + FP} \quad (2.1)$$

Presisi merepresentasikan tingkat ketepatan model dalam melakukan identifikasi, dengan berfokus pada kemampuannya menghindari deteksi yang tidak seharusnya (false positive). Semakin tinggi nilai presisi, semakin sedikit jumlah deteksi palsu yang dihasilkan model,

menunjukkan bahwa model lebih efektif dalam meminimalkan kesalahan deteksi. Namun, untuk mendapatkan gambaran menyeluruh tentang kinerja model, presisi sering dikombinasikan dengan metrik lain seperti recall dan F1 Score.

b) Recall

Recall adalah metrik evaluasi yang mengukur kemampuan model untuk mendeteksi semua objek yang relevan dalam sebuah gambar[32]. Pada deteksi objek, recall menggambarkan seberapa baik model dalam menemukan dan mengidentifikasi semua instance dari objek yang sebenarnya ada, tanpa melewatkan satu pun. Secara matematis, recall dihitung dengan rumus:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.2)$$

Recall memberikan gambaran tentang kemampuan model dalam mengurangi kesalahan deteksi dengan memastikan bahwa semua objek yang relevan berhasil ditemukan. Metrik ini mengukur kemampuan sistem untuk meminimalisir kegagalan deteksi dengan berhasil menemukan proporsi maksimal dari objek yang relevan. Tingginya nilai recall mencerminkan kapasitas model untuk mengenali mayoritas objek yang ada, sehingga meminimalkan insiden *false negative* situasi dimana objek yang seharusnya terdeteksi justru terlewatkan.

c) F1-Score

F1-Score adalah metrik evaluasi dalam machine learning yang menggabungkan dua metrik penting, yaitu presisi (precision) dan recall, ke dalam satu nilai yang merepresentasikan keseimbangan antara keduanya. F1-Score dihitung sebagai rata-rata harmonis dari presisi dan recall, dengan rumus:

$$\text{F1 - Score} = \frac{2 \times \text{Presisi} \times \text{Recall}}{\text{Presisi} + \text{Recall}} \quad (2.3)$$

Nilai F1-Score berkisar antara 0 hingga 1, di mana nilai 1 menunjukkan performa sempurna (presisi dan recall sama-sama tinggi). Di sisi berlawanan, nilai F1-Score yang mendekati atau mencapai 0 mengindikasikan kinerja yang sangat lemah, mencerminkan ketidakmampuan model dalam mencapai keseimbangan efektif antara presisi dan recall. Skor rendah ini menandakan bahwa model memiliki defisiensi signifikan baik dalam hal banyaknya false positive, false negative, atau kombinasi keduanya.

d) Average Precision (AP)

Average Precision (AP) adalah metrik yang digunakan untuk mengevaluasi kinerja model deteksi objek dengan mengukur seberapa baik model dalam mengidentifikasi dan mengklasifikasikan objek pada berbagai ambang batas IoU. AP dihitung sebagai area di bawah kurva Precision-Recall (PR)[34]. Kurva ini dihasilkan dengan memvariasikan nilai ambang batas IoU yang digunakan untuk menentukan apakah suatu prediksi dianggap sebagai True Positive atau tidak. Semakin besar area di bawah kurva ini, semakin baik performa model dalam mendeteksi objek. Secara matematis, AP dapat dihitung menggunakan persamaan berikut:

$$AP = \int_0^1 P(r)dr \quad (2.4)$$

Dimana $P(r)$ adalah presisi sebagai fungsi dari recall r . Dalam praktiknya, AP dihitung dengan merata-ratakan presisi pada beberapa titik recall diskrit yang dihasilkan saat ambang batas bervariasi. Nilai AP untuk setiap kelas objek dalam dataset kemudian dirata-ratakan untuk menghasilkan Mean Average Precision (mAP) yang memberikan gambaran umum tentang kinerja model di seluruh kelas yang ada. Nilai AP berkisar antara 0 hingga 1, di mana nilai yang lebih tinggi menunjukkan kinerja model yang lebih baik.

e) Mean Average Precision (mAP)

Mean Average Precision (mAP) adalah metrik yang digunakan untuk mengevaluasi performa keseluruhan sistem deteksi objek di seluruh kelas dalam dataset. mAP dihitung dengan merata-ratakan nilai Average Precision (AP) untuk setiap kelas yang mencerminkan efektivitas model dalam mendeteksi berbagai jenis objek[35]

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (2.5)$$

dimana N adalah jumlah kelas dalam dataset dan AP_i adalah Average Precision untuk kelas ke- i . mAP memberikan gambaran umum tentang kinerja model dalam mendeteksi berbagai jenis objek. Nilai mAP yang tinggi menunjukkan bahwa model memiliki performa deteksi yang baik dengan tingkat kesalahan yang rendah, sehingga metrik ini sangat berguna untuk membandingkan dan meningkatkan model deteksi objek.

f) Loss

Loss merupakan metrik fundamental dalam sistem deteksi objek yang mengkuantifikasi disparitas antara prediksi yang dihasilkan model dengan ground truth atau nilai aktual. Fungsi loss berperan sebagai kompas yang mengarahkan proses pembelajaran model, menyediakan sinyal numerik tentang seberapa akurat prediksi saat ini dan bagaimana model harus menyesuaikan parameter-parameternya. Berikut beberapa jenis loss yang digunakan:

- Box Loss

Box loss adalah jenis bounding box regression loss yang berfungsi untuk membantu model memahami posisi dan ukuran yang tepat dari bounding box yang mengelilingi objek yang terdeteksi. Loss ini mengukur kesalahan dalam prediksi koordinat dan dimensi bounding box dibandingkan dengan ground truth

dengan tujuan mengurangi perbedaan antara kotak yang diprediksi dan yang sebenarnya.

- Class Loss

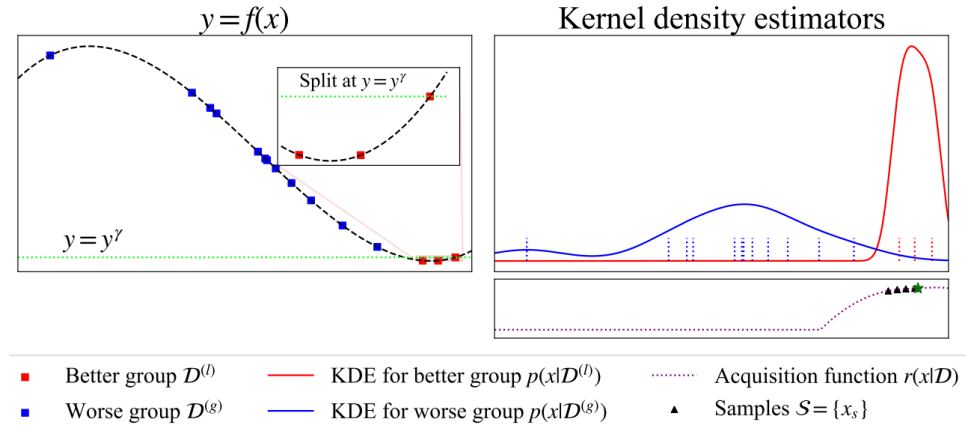
Class loss berkaitan dengan akurasi klasifikasi objek yang terdeteksi. Tujuan utama class loss adalah memastikan bahwa setiap objek yang terdeteksi diklasifikasikan dengan benar ke dalam salah satu kategori yang telah ditentukan. Loss ini diukur berdasarkan perbedaan antara probabilitas kelas yang diprediksi dan label kelas ground truth.

- Distribution Focal Loss (DFL)

Distribution focal loss berfungsi untuk meningkatkan akurasi model, terutama dalam skenario deteksi objek yang kompleks dengan menekankan pada objek yang sulit dideteksi. Loss ini menyesuaikan bobot kesalahan secara berbeda berdasarkan tingkat kesulitan, sehingga model dapat lebih baik dalam mengestimasi kategori objek. Loss ini juga mengatasi masalah ketidakseimbangan kelas dengan memberikan fokus lebih pada contoh-contoh yang sulit diklasifikasikan.

2.4 TPE (Tree-Structured Parzen Estimator)

Tree-Structured Parzen Estimator (TPE) merupakan algoritma optimasi hiperparameter berbasis *Sequential Model-Based Optimization* (SMBO) yang menggunakan pendekatan Bayesian. Dalam penelitian ini, TPE digunakan untuk menavigasi ruang pencarian hiperparameter model YOLOv8 secara efisien guna memaksimalkan metrik evaluasi pada proses *grading* apel. Berbeda dengan metode *Grid Search* atau *Random Search* yang bersifat "buta" terhadap hasil eksperimen sebelumnya, TPE membangun model probabilistik yang cerdas untuk memprediksi konfigurasi mana yang paling mungkin memberikan performa terbaik pada iterasi selanjutnya [77]. Berikut visualisasi bagaimana TPE bekerja:



Gambar 2. 7 Visualisasi cara TPE bekerja [77]

Gambar 2.6 menunjukkan proses kerja Tree-Structured Parzen Estimator (TPE) dalam menentukan konfigurasi hyperparameter berikutnya. Hasil percobaan sebelumnya terlebih dahulu diurutkan berdasarkan nilai performa, kemudian dibagi menjadi dua kelompok, yaitu kelompok better yang berisi konfigurasi dengan performa terbaik dan kelompok worse yang berisi konfigurasi dengan performa lebih rendah. TPE kemudian membentuk distribusi probabilitas menggunakan Kernel Density Estimation (KDE) untuk masing-masing kelompok. Kandidat hyperparameter selanjutnya diambil dari distribusi kelompok better dan dibandingkan berdasarkan rasio kepadatannya terhadap kelompok worse. Kandidat dengan rasio terbaik dipilih sebagai percobaan berikutnya, sehingga setiap percobaan baru memanfaatkan hasil percobaan sebelumnya dan pencarian secara bertahap diarahkan menuju kombinasi hyperparameter yang lebih optimal[77].

2.4.1. Mekanisme

Pada pendekatan Bayesian optimization konvensional, seperti yang berbasis *Gaussian Process* (GP), model probabilistik dibangun dengan memodelkan distribusi probabilitas nilai objektif y (misalnya *validation loss* atau *mAP*) terhadap konfigurasi hiperparameter x , yaitu:

$$p(y|x) \quad (2.6)$$

TPE Namun, pendekatan ini memiliki keterbatasan ketika diterapkan pada ruang pencarian yang berdimensi tinggi, bersifat tidak kontinu, mengandung variabel kategorikal, atau memiliki struktur bersyarat. Untuk mengatasi keterbatasan tersebut, TPE menggunakan pendekatan alternatif dengan membalik hubungan probabilistik, yaitu memodelkan distribusi hiperparameter berdasarkan nilai performa:

$$p(x|y) \quad (2.7)$$

Pendekatan ini memungkinkan TPE untuk menangani ruang pencarian yang kompleks secara lebih fleksibel dan efisien.

2.4.2 Representasi Data Percobaan

Misalkan tersedia kumpulan hasil percobaan sebelumnya

$$D = (x_i, y_i): i = 1, \dots, N \quad (2.8)$$

x_i = adalah vektor hiperparameter pada percobaan ke- i

y_i = adalah nilai objektif (misalnya mAP)

N = adalah jumlah total percobaan

Setiap pasangan (x_i, y_i) merepresentasikan satu proses pelatihan model secara penuh.

2.4.3 Penentuan Ambang Batas Kuantil

Untuk TPE menentukan nilai ambang batas γ berdasarkan kuantil γ dari seluruh nilai performa.

$$P(y \leq y_\gamma) = \gamma \quad (2.9)$$

Nilai γ biasanya berada pada rentang 0.1–0.25.

2.4.4 Pembagian Data Menjadi Kelompok Baik dan Buruk

Berdasarkan nilai γ , data percobaan dibagi menjadi dua kelompok:

Kelompok Baik (Good)

$$D_l = (x_i|y_i) \leq y_\gamma \quad (2.10)$$

Kelompok Buruk (Bad)

$$D_g = (x_i|y_i) > y_\gamma \quad (2.11)$$

Kelompok D_l berisi konfigurasi hiperparameter yang menghasilkan performa relatif baik, sedangkan D_g berisi konfigurasi dengan performa relatif buruk

2.4.5 Definisi Distribusi Probabilistik Bersyarat

Distribusi probabilitas hiperparameter didefinisikan secara potongan sebagai berikut:

$$p(x|y, D) = \begin{cases} p(x|D_l), & y \leq y_\gamma \\ p(x|D_g), & y > y_\gamma \end{cases} \quad (2.12)$$

Distribusi ini kemudian disederhanakan dengan notasi:

$$l(x) = p(x|D_l) \quad (2.13)$$

$$g(x) = p(x|D_g) \quad (2.14)$$

Distribusi $l(x)$ merepresentasikan karakteristik hiperparameter yang cenderung menghasilkan performa baik, sedangkan $g(x)$ merepresentasikan hiperparameter dengan performa buruk.

2.4.6 Estimasi Densitas Menggunakan Parzen Estimator (KDE)

Distribusi $l(x)$ dan $g(x)$ diestimasi menggunakan *Kernel Density Estimation* (KDE), di mana kernel Gaussian ditempatkan pada setiap titik data historis.

$$l(x) = \sum w_i K(x, x_i) \quad (2.15)$$

$$g(x) = \sum w_j K(x, x_j) \quad (2.16)$$

$K(x, x_i)$ adalah fungsi kernel Gaussian

w_i dan w_j adalah bobot masing-masing sampel

2.4.7 Fungsi Akuisisi: Expected Improvement (EI)

Untuk menentukan konfigurasi hiperparameter baru yang akan diuji, TPE menggunakan fungsi akuisisi *Expected Improvement* (EI). Dalam TPE, EI dapat diturunkan sehingga proporsional terhadap rasio antara densitas kelompok baik dan buruk.

$$EI(x) \propto l(x)/g(x) \quad (2.17)$$

Nilai EI yang besar menunjukkan bahwa konfigurasi hiperparameter x memiliki probabilitas tinggi untuk berada pada area performa baik dan probabilitas rendah berada pada area performa buruk.

2.4.8 Optimasi Konfigurasi Hiperparameter

Konfigurasi hiperparameter terbaik pada iterasi berikutnya ditentukan dengan memaksimalkan rasio tersebut.

$$x^* = \operatorname{argmax}_x l(x)/g(x) \quad (2.18)$$

Vektor x^* merupakan satu set lengkap hiperparameter (misalnya learning rate, weight decay, dan parameter lainnya) yang kemudian digunakan untuk melatih model YOLOv8 dan menghasilkan nilai performa baru.

2.4.9 Fungsi Objektif YOLOv8

Dalam penelitian ini terdapat dua tingkat optimasi yang berbeda, yaitu optimasi pada tingkat model YOLOv8 dan optimasi pada tingkat hiperparameter menggunakan algoritma TPE. Kedua optimasi ini memiliki fungsi objektif yang berbeda namun saling berkaitan secara langsung dalam suatu proses optimasi bersarang (*nested optimization*). Agar tidak terjadi kekeliruan konseptual, penting untuk membedakan secara jelas fungsi objektif pada masing-masing tingkat optimasi tersebut serta menjelaskan bagaimana keduanya berinteraksi.

YOLOv8 merupakan model berbasis *deep learning* yang dilatih melalui proses optimasi parameter jaringan saraf, yaitu bobot dan bias, yang secara kolektif dinotasikan sebagai parameter model θ .

Tujuan utama pelatihan YOLOv8 adalah meminimalkan fungsi loss yang merepresentasikan kesalahan prediksi model terhadap data pelatihan. Fungsi loss YOLOv8 umumnya terdiri dari beberapa komponen, seperti *bounding box loss*, *objectness loss*, dan *classification loss*. Secara matematis, proses pelatihan YOLOv8 dapat dirumuskan sebagai:

$$\theta^*(x) = \operatorname{argmin}_{\theta} L(\theta; x, D_{\text{train}}) \quad (2.19)$$

dengan:

- θ adalah parameter model YOLOv8 (bobot dan bias),
- x adalah vektor hiperparameter (learning rate, weight decay, dan lain-lain),
- D_{train} adalah dataset pelatihan
- $L(\theta; x, D_{\text{train}})$ adalah fungsi loss YOLOv8 pada data pelatihan,
- $\theta^*(x)$ adalah parameter model optimal yang diperoleh untuk konfigurasi hiperparameter x .

Pada tahap ini, hiperparameter x tidak dioptimasi oleh YOLOv8, melainkan dianggap sebagai nilai tetap yang mengatur proses optimasi parameter θ . Dengan demikian, YOLOv8 hanya menyelesaikan optimasi tingkat dalam (*inner optimization*).

Setelah proses pelatihan selesai dan parameter optimal $\theta^*(x)$ diperoleh, model YOLOv8 dievaluasi menggunakan dataset validasi untuk mengukur performa model. Proses evaluasi ini dapat dituliskan sebagai:

$$y = \operatorname{Eval}(\theta^*(x), D_{\text{val}}) \quad (2.20)$$

dengan:

- D_{val} adalah dataset validasi,
- $\operatorname{Eval}(\cdot)$ adalah fungsi evaluasi model (misalnya perhitungan mAP),

- y adalah nilai metrik evaluasi yang dihasilkan.

Nilai y bersifat skalar, dan merepresentasikan kualitas model YOLOv8 yang dilatih menggunakan konfigurasi hiperparameter x

2.4.10 Fungsi Objektif TPE

Berbeda dengan YOLOv8, algoritma TPE tidak mengakses parameter jaringan saraf θ secara langsung. TPE hanya mengamati hubungan antara konfigurasi hiperparameter x dan nilai performa y yang dihasilkan oleh YOLOv8. Oleh karena itu, TPE mendefinisikan fungsi objektifnya sendiri sebagai:

$$f(x) = y \quad (2.21)$$

atau secara eksplisit:

$$f(x) = Eval(\theta^*(x), D_{val}) \quad (2.21)$$

Fungsi $f(x)$ ini merupakan fungsi objektif TPE, yang memetakan vektor hiperparameter ke nilai performa model YOLOv8. Tujuan optimasi hiperparameter menggunakan TPE adalah mencari konfigurasi hiperparameter optimal x^* yang memaksimalkan nilai fungsi objektif tersebut:

$$x^* = \operatorname{argmax}_x f(x) \quad (2.22)$$

Dengan menggabungkan optimasi tingkat dalam dan tingkat luar, keseluruhan proses optimasi hiperparameter dapat dirumuskan sebagai berikut:

$$x^* = \operatorname{argmax}_x Eval(\operatorname{argmin}_{\theta} L(\theta; x, D_{train}), D_{val}) \quad (2.23)$$

Persamaan ini menunjukkan bahwa:

1. Untuk setiap konfigurasi hiperparameter x , YOLOv8 terlebih dahulu mencari parameter jaringan saraf optimal $\theta^*(x)$ dengan meminimalkan fungsi loss.

2. Parameter $\theta^*(x)$ kemudian dievaluasi untuk menghasilkan nilai performa y .
3. Algoritma TPE menggunakan nilai y tersebut untuk menentukan konfigurasi hiperparameter yang lebih baik.

Dengan menggabungkan optimasi tingkat dalam dan tingkat luar, keseluruhan proses optimasi hiperparameter dapat dirumuskan sebagai berikut:

$$D = (x_i, y_i): i = 1, \dots, N \quad (2.24)$$

dengan:

- x_i adalah konfigurasi hiperparameter pada percobaan ke- i ,
- $y_i = f(x_i)$ adalah nilai performa yang dihasilkan oleh YOLOv8,
- N adalah jumlah total percobaan.

Berdasarkan dataset tersebut, TPE membangun model probabilistik $p(x|y)$ dan memilih konfigurasi hiperparameter baru dengan memaksimalkan fungsi akuisisi:

$$EI(x) \propto l(x)/g(x) \quad (2.25)$$

yang secara probabilistik mengarahkan pencarian ke area hiperparameter dengan peluang performa tinggi.

2.4.11 Interaksi antara Fungsi Objektif YOLOv8 dan TPE

Interaksi antara kedua fungsi objektif dapat dirangkum dalam langkah-langkah berikut:

1. TPE mengusulkan konfigurasi hiperparameter x_k
2. YOLOv8 dilatih menggunakan konfigurasi tersebut:

$$\theta_k = \operatorname{argmin}_{\theta} L(\theta; x_k, D_{\text{train}}) \quad (2.26)$$

3. Model YOLOv8 dievaluasi:

$$y_k = Eval(\theta_k, D_{val}) \quad (2.27)$$

4. Pasangan (x_k, y_k) ditambahkan ke dataset:

$$D = D \cup (x_k, y_k) \quad (2.28)$$

5. TPE memperbarui model probabilistik dan mengulangi proses hingga kriteria penghentian terpenuhi

Dengan demikian, fungsi objektif YOLOv8 berperan sebagai sumber evaluasi performa, sedangkan fungsi objektif TPE berperan sebagai pengendali proses pencarian hiperparameter.

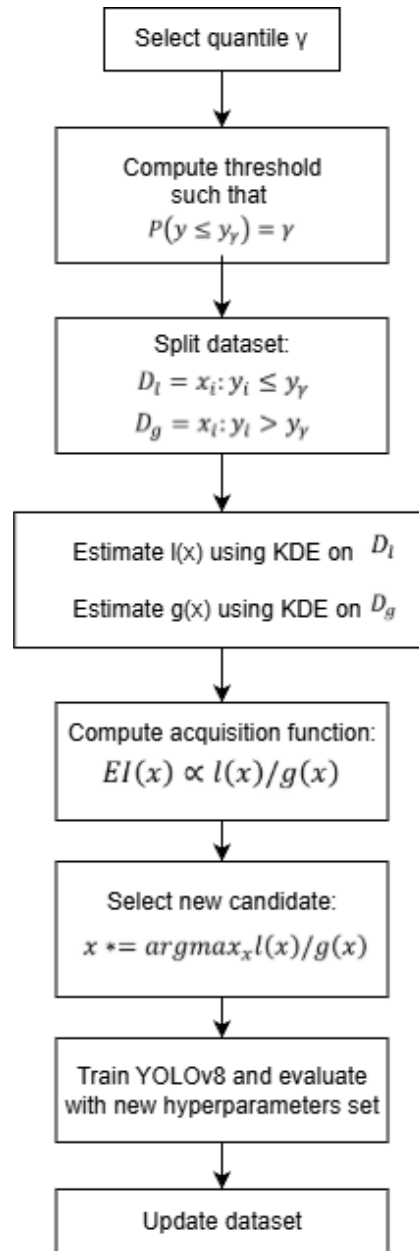
2.4.12 *Set Up Trial TPE*

Dalam konteks model YOLOv8, tidak semua hiperparameter memiliki pengaruh yang sama terhadap performa model. Oleh karena itu, penelitian ini hanya mengoptimasi hiperparameter yang menurut literatur memiliki dampak paling signifikan terhadap konvergensi, stabilitas, dan generalisasi model. Hiperparameter yang paling sensitif adalah learning rate. Studi oleh Jocher (2023) dan Ramos (2024) menunjukkan bahwa learning rate memiliki pengaruh terbesar terhadap kualitas deteksi karena mengontrol besar langkah gradien pada setiap iterasi. Nilai yang terlalu besar menyebabkan osilasi atau divergensi, sedangkan nilai terlalu kecil menghasilkan pelatihan yang sangat lambat. Karena sensitivitasnya yang tinggi, learning rate dioptimasi menggunakan distribusi log-uniform agar TPE dapat menjelajahi area kritis secara lebih halus [37].

Weight decay juga termasuk hiperparameter penting yang memengaruhi kemampuan generalisasi model. Nilai weight decay yang terlalu kecil dapat menyebabkan model menghafal pola tertentu secara berlebihan, sedangkan nilai yang terlalu besar dapat menghambat kemampuan jaringan dalam mempelajari fitur visual apel yang relevan. Penelitian terdahulu menekankan bahwa tuning weight decay dapat

meningkatkan mAP secara signifikan, sehingga parameter ini juga dioptimasi secara kontinu oleh TPE [38].

Di samping parameter kontinu, penelitian ini juga mengoptimasi parameter diskrit seperti batch size dan jenis optimizer. Batch size memengaruhi stabilitas gradien serta konsumsi memori, dan penggunaan nilai yang terlalu besar atau terlalu kecil dapat berdampak pada kualitas pelatihan. Rentang nilai yang digunakan mengacu pada penelitian YOLOv8 dan eksperimen terdahulu yang menunjukkan bahwa batch size sedang memberikan kompromi terbaik antara kecepatan dan stabilitas. Jenis optimizer, seperti SGD dan AdamW, juga memengaruhi dinamika pembaruan bobot serta laju konvergensi. Kedua optimizer tersebut menunjukkan karakteristik berbeda dalam penelitian terdahulu, sehingga pemilihan melalui TPE memberikan kesempatan untuk secara adaptif menentukan konfigurasi yang paling sesuai dengan dataset apel yang digunakan [14]. Berikut Flowchart Algoritma TPE dan Proses Trial TPE:



Gambar 2. 8 *Flowchart* Algoritma TPE

Sebaliknya, hiperparameter seperti jumlah *epoch* tidak dimasukkan dalam proses optimasi TPE. Hal ini dilakukan karena *epoch* lebih berfungsi sebagai durasi pelatihan, bukan faktor yang mengubah dinamika internal model secara signifikan. Menjadikan *epoch* sebagai variabel optimasi justru akan meningkatkan waktu komputasi secara drastis tanpa memberikan informasi tambahan yang bermanfaat bagi TPE. Praktik terbaik dalam

pelatihan model YOLO menunjukkan bahwa *epoch* dapat ditetapkan pada nilai tetap selama proses pencarian hiperparameter, misalnya 50 *epoch* untuk mempercepat setiap percobaan TPE. Setelah konfigurasi hiperparameter terbaik ditemukan, pelatihan final kemudian dilakukan menggunakan jumlah *epoch* yang lebih besar, seperti 150 atau 200 *epoch*, untuk memaksimalkan kemampuan model [38].

Dengan demikian, penggunaan TPE dalam penelitian ini difokuskan pada hiperparameter yang benar-benar memberikan dampak signifikan terhadap performa YOLOv8, yaitu learning rate, weight decay, batch size, dan optimizer. Pendekatan ini memastikan bahwa proses optimasi berjalan efisien sekaligus mampu menghasilkan konfigurasi yang memberikan peningkatan nyata terhadap akurasi dan stabilitas deteksi kualitas apel. TPE memungkinkan pemilihan hiperparameter yang lebih adaptif dan berbasis probabilistik, sehingga model yang dihasilkan lebih optimal dan siap digunakan pada proses grading apel secara otomatis [41].

2.5 Roboflow



Gambar 2. 9 Roboflow [86]

Roboflow adalah platform end-to-end yang dirancang khusus untuk memudahkan pengembangan model computer vision, dengan fokus pada pengelolaan dataset, preprocessing gambar, dan deployment model. Platform ini menyediakan serangkaian alat yang komprehensif untuk mengotomatisasi dan mengoptimalkan alur kerja dalam proyek computer vision, termasuk sistem deteksi

objek seperti YOLOv8 yang Anda implementasikan untuk deteksi dan klasifikasi apel.

Fitur utama Roboflow mencakup manajemen dataset yang intuitif, memungkinkan pengguna untuk mengimpor, mengorganisir, dan memberi anotasi pada gambar dengan efisien. Platform ini menawarkan kemampuan anotasi yang kuat untuk berbagai tugas computer vision seperti deteksi objek, segmentasi, dan klasifikasi. Roboflow juga menyediakan berbagai teknik augmentasi data yang dapat dikonfigurasi untuk meningkatkan keragaman dataset, termasuk rotasi, penskalaan, pemotongan, dan penyesuaian kecerahan yang sangat berguna untuk melatih model deteksi apel yang robust terhadap variasi kondisi pengambilan gambar.

2.6 Ultralytics



Gambar 2. 10 Ultralytics [87]

Ultralytics adalah perusahaan yang dikenal karena pengembangan model deteksi objek berbasis algoritma YOLO (You Only Look Once). Didirikan oleh Glenn Jocher, Ultralytics telah merilis beberapa versi dari YOLO, termasuk YOLOv11. Model-model ini dirancang untuk memberikan kinerja tinggi dalam deteksi objek secara real-time, dengan fokus pada efisiensi komputasi dan akurasi yang lebih baik dibandingkan versi sebelumnya[36]. YOLO menggunakan pendekatan unik dengan membagi gambar menjadi grid dan memprediksi bounding box serta kelas objek secara simultan. Ini berbeda dari metode tradisional yang memerlukan pemisahan antara deteksi dan klasifikasi. Dalam pengembangan YOLOv11, Ultralytics memperkenalkan arsitektur baru yang lebih efisien, yang

memungkinkan model untuk beroperasi dengan lebih cepat dan akurat. YOLOv8, yang dirilis pada Januari 2023, membawa inovasi tambahan dengan menghilangkan penggunaan anchor boxes dalam prediksi, sehingga mengurangi beban komputasi. Model ini juga menggunakan modul C2f untuk meningkatkan akurasi melalui pemrosesan fitur kontekstual. Dengan evaluasi menggunakan dataset MS COCO 2017, YOLOv8 menunjukkan peningkatan akurasi signifikan, mencapai 53,9% pada threshold IoU 0,5.

2.7 Raspberry Pi 5

Raspberry Pi 5 merupakan **single-board computer (SBC)** generasi terbaru yang dikembangkan oleh Raspberry Pi Foundation untuk mendukung berbagai aplikasi komputasi, otomasi, Internet of Things (IoT), robotika, serta *edge computing*. Dibandingkan generasi sebelumnya, Raspberry Pi 5 menawarkan peningkatan performa yang signifikan melalui penggunaan prosesor yang lebih cepat, antarmuka I/O berkecepatan tinggi, serta dukungan PCI Express (PCIe) untuk perangkat eksternal. Perangkat ini mampu menjalankan sistem operasi berbasis Linux, seperti Raspberry Pi OS maupun Ubuntu, sehingga banyak digunakan sebagai platform pengembangan sistem *embedded* dan aplikasi *computer vision*.

Pada penelitian ini, Raspberry Pi 5 berfungsi sebagai perangkat utama yang menjalankan model YOLOv8 untuk melakukan klasifikasi tingkat kematangan apel secara *real-time*. Proses inferensi dipercepat menggunakan **Hailo AI HAT+** yang terhubung melalui antarmuka PCIe, sehingga beban komputasi AI dialihkan ke akselerator khusus. Selain itu, Raspberry Pi 5 juga bertugas mengendalikan perangkat keras lainnya, seperti Pi Cam untuk akuisisi citra dan Arduino Mega sebagai pengendali aktuator melalui komunikasi serial. Dengan kombinasi performa CPU yang tinggi, dukungan sistem operasi Linux, serta kemampuan ekspansi melalui PCIe, Raspberry Pi 5 menjadi platform yang sesuai untuk implementasi sistem klasifikasi berbasis *edge AI*.



Gambar 2. 11 *Board Raspberry Pi [88]*

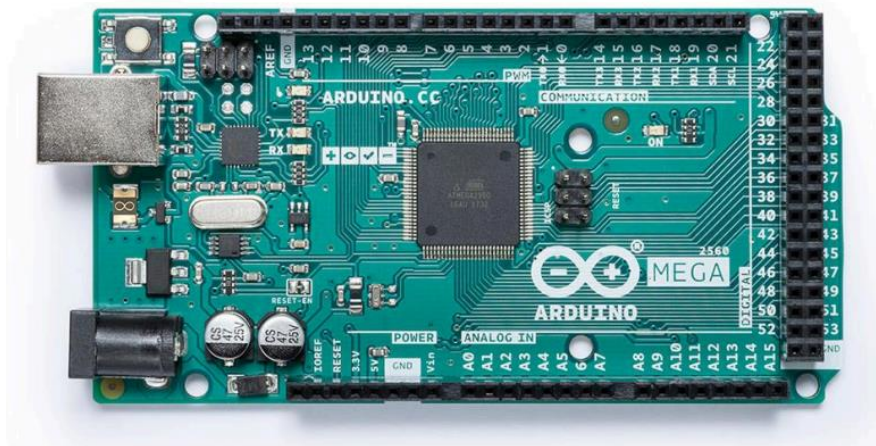
OpenCV merupakan pustaka *open-source* yang banyak digunakan untuk pengolahan citra digital dan *computer vision*. Library ini kompatibel dengan Raspberry Pi 5 dan digunakan bersama Python untuk melakukan akuisisi citra, pra-pemrosesan, serta menampilkan hasil deteksi dari model YOLOv8 secara *real-time*. Spesifikasi lengkap Raspberry Pi 5 yang digunakan pada penelitian ini dapat dilihat pada Tabel 2.1.

Tabel 2. 1 Spesifikasi Raspberry Pi5

Parameter	Keterangan
GPU	VideoCore VII GPU, mendukung OpenGL ES 3.1 dan Vulkan 1.2
CPU	Broadcom BCM2712, Quad-core Arm Cortex-A76 64-bit @ 2,4 GHz
Memory	8 GB LPDDR4X SDRAM
Storage	Slot microSD untuk sistem operasi, mendukung boot melalui NVMe SSD (PCIe)
Display	2 × micro-HDMI (hingga 4Kp60), 2-lane MIPI DSI
Power	USB Type-C 5 V DC
USB	2 × USB 3.0, 2 × USB 2.0

Dimensi	85 mm × 56 mm
Other	Gigabit Ethernet, Wi-Fi 802.11ac dual-band, Bluetooth 5.0/BLE

2.8 Arduino Mega



Gambar 2. 12 Arduino Mega [89]

Arduino Mega adalah board mikrokontroler berbasis ATmega2560. Board ini memiliki sumber daya I/O yang jauh lebih besar dibanding Arduino Uno, yaitu 54 pin digital (15 di antaranya mendukung PWM), 16 pin input analog, empat port UART hardware, kristal osilator 16 MHz, koneksi USB, jack power, header ICSP, serta tombol reset. Arduino Mega tetap mengusung sifat open-source dan menggunakan bahasa pemrograman berbasis C/C++ melalui Arduino IDE. Loader USB yang telah terintegrasi memudahkan proses pemrograman tanpa membutuhkan programmer eksternal. Spesifikasi Arduino Mega dapat dilihat pada Tabel 2.3.

Tabel 2. 2 Spesifikasi Arduino

Parameter	Keterangan
Tegangan Input	7-12V
Pin I/O Digital	54 (15 PWM)
Pin Input Analog	16
Arus DC per Pin I/O	20 mA

Dimensi	101.5 mm × 53.4 mm
---------	--------------------

Arduino Pada penelitian ini, Arduino Mega digunakan untuk mengendalikan servo motor dan motor power window. Servo motor bekerja menggunakan sinyal PWM 50 Hz (periode 20 ms) yang dihasilkan oleh pin PWM Arduino melalui library **Servo.h**, memungkinkan pengaturan sudut presisi (0° – 180°). Motor power window, yang beroperasi pada 12 V dan membutuhkan arus lebih besar, dikendalikan menggunakan driver motor BTS7960 untuk mengatur arah dan kecepatan putaran. Driver ini menerima sinyal PWM dari Arduino Mega sehingga pengaturan motor dapat dilakukan secara aman dan stabil.

2.9 Camera Pi V3



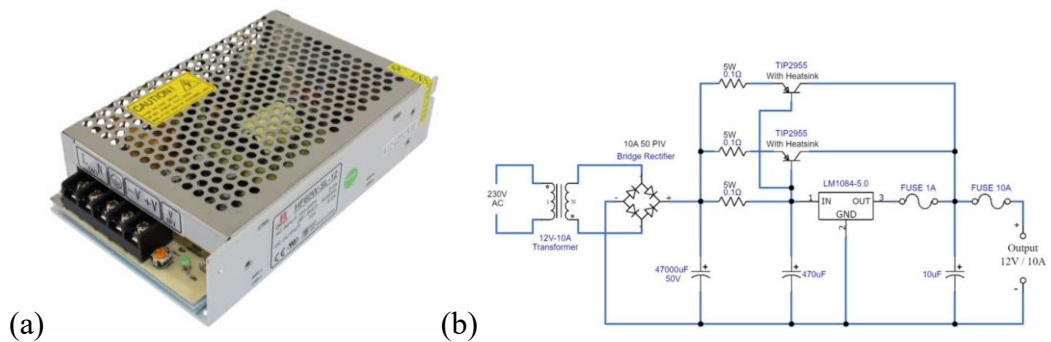
Gambar 2. 13 Camera Pi V3 [90]

Camera Pi V3 Wide adalah modul kamera resmi generasi ketiga dari Raspberry Pi Foundation yang menggunakan sensor Sony IMX708 dengan sudut pandang lebih lebar. Varian ini dirancang untuk aplikasi yang membutuhkan cakupan area yang luas, seperti computer vision, robotika, pemantauan, dan sistem otomatisasi. Dibandingkan dengan varian standar, V3 Wide mampu menangkap area lebih besar tanpa perlu menambah jarak 29 kamera, sehingga sangat cocok digunakan pada conveyor atau sistem sortir buah yang membutuhkan deteksi objek dalam area lebar. Selain itu, Camera Pi V3 Wide tetap membawa semua keunggulan generasi ketiga, yaitu autofocus, dukungan HDR, serta kinerja yang lebih baik pada kondisi cahaya rendah.

Tabel 2. 3 Spesifikasi Camera

Parameter	Keterangan
Resolusi Gambar	4608 × 2592 piksel (12 Megapiksel)
Sensor	Sony IMX708
Interface	CSI-2 (Camera Serial Interface) melalui flat ribbon (22-pin pada Raspberry Pi 5, adaptor tersedia untuk 15-pin lama)
Field of View	102° diagonal (wide-angle)
Fitur	Autofocus, HDR, performa cahaya rendah lebih baik

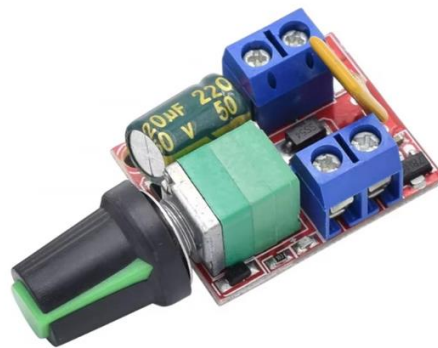
2.10 Power Supply

**Gambar 2. 14** (a) *Power Supply* (b) Rangkaian *Power Supply* [91]

Power Supply atau catu daya merupakan komponen elektronik yang berperan sebagai penyedia energi listrik untuk berbagai perangkat elektronik yang terhubung dengannya. Secara fundamental, catu daya berfungsi sebagai sistem penyearah dan filter karena mengkonversi arus bolak-balik (AC) menjadi arus searah (DC). Istilah *power supply* juga dikenal dengan sebutan converter tegangan listrik atau *electric power converter*. Mekanisme kerja *power supply*, khususnya tipe *switching power supply* 12V 15A yang digunakan dalam penelitian ini, melibatkan empat tahap utama yaitu penyearahan, penyaringan, regulasi, dan proteksi pada komponen

internal. Prosesnya dimulai ketika tegangan AC 220V dari jaringan listrik PLN dikonversi menggunakan rangkaian bridge rectifier (dioda penyearah) untuk menghasilkan sumber listrik DC. Selanjutnya, untuk meningkatkan stabilitas tegangan, digunakan kapasitor filter, dan pengaturan tegangan dilakukan melalui IC atau switching regulator yang berfungsi sebagai rangkaian pengatur. Hal ini memastikan output 12V DC yang dihasilkan memiliki tingkat stabilitas yang optimal.

2.11 Dimmer DC



Gambar 2. 15 (a) Visual BTS7960 [92]

Dimmer DC 5A 4V–35V merupakan modul pengatur kecepatan motor DC yang bekerja menggunakan metode Pulse Width Modulation (PWM). Modul ini dirancang untuk mengatur kecepatan putaran motor DC dengan cara mengubah duty cycle sinyal PWM tanpa mengubah besar tegangan sumber secara langsung. Dimmer ini mendukung tegangan masukan 4V hingga 35V DC dengan kemampuan arus maksimum 5A, sehingga sesuai digunakan untuk mengendalikan motor DC berdaya kecil hingga menengah. Pada penelitian ini, dimmer digunakan untuk mengatur kecepatan putaran motor DC pada mekanisme penyuplai apel dengan menyesuaikan duty cycle melalui potensiometer yang tersedia pada modul. Pengaturan kecepatan ini memungkinkan laju putaran motor disesuaikan sehingga diperoleh kecepatan suplai apel yang diinginkan. Berikut merupakan spesifikasi dari Dimmer DC 5A 4V–35V yang digunakan pada penelitian ini.

Tabel 2. 4 Spesifikasi Dimmer

Parameter	Deskripsi
Tegangan	5 – 35V DC
<i>Working Duty Cycle</i>	0 – 100%
<i>Peak Current</i>	5A
Berat	120 gram
Mode Kontrol	Potensio

2.12 Motor DC 12V

**Gambar 2. 16** Motor DC 12 V [93]

Motor DC 12V merupakan aktuator elektromekanis yang mengubah energi listrik arus searah menjadi gerakan rotasi dan digunakan sebagai mekanisme pengatur suplai apel menuju konveyor agar aliran buah lebih terkontrol. Untuk mendukung kestabilan pada kecepatan rendah, motor dilengkapi gearbox sehingga mampu menghasilkan torsi yang lebih besar dan konsisten. Pengaturan kecepatan dan timing dilakukan menggunakan metode Pulse Width Modulation (PWM) melalui driver agar suplai apel dapat diatur secara presisi dan mengurangi risiko penumpukan objek pada konveyor.

Tabel 2. 5 Spesifikasi Motor DC 12V

Parameter	Deskripsi
Tegangan Nominal	12 V DC
Torsi	10 – 30 kg·cm

Arus Beban	0.5 – 2 A
Metode Kontrol	PWM (melalui driver)

Pada sistem ini digunakan mekanisme roda silinder berlapis silikon lunak sebagai regulator suplai apel pada bidang miring 45° , sehingga motor hanya mengatur distribusi gerakan apel melalui gaya gesek dan rolling contact, bukan menggerakkan konveyor utama secara langsung. Kebutuhan torsi dihitung menggunakan persamaan $T = F \times r$, dengan T sebagai torsi motor, F sebagai gaya efektif beban, dan r sebagai jari-jari roda. Berdasarkan asumsi massa total apel sebesar 2,25 kg dan jari-jari roda 0,05 m, diperoleh kebutuhan torsi aktual sekitar 0,8–1,2 Nm setelah mempertimbangkan faktor keamanan terhadap deformasi silikon dan potensi jamming. Pendekatan ini sejalan dengan penelitian mengenai rotating feeding device yang menyatakan bahwa kebutuhan torsi pada mekanisme feeder berputar dipengaruhi oleh beban material dan kondisi dinamis sistem [90].