

BAB II

KAJIAN LITERATUR

2.1 Kecerdasan Buatan (*Artificial Intelligence*) dan Pengolahan Citra

Kecerdasan buatan (*Artificial Intelligence/AI*) merupakan cabang ilmu komputer yang difokuskan pada penciptaan sistem atau mesin yang mampu meniru kemampuan kognitif manusia, seperti proses pembelajaran, penalaran, dan pengambilan keputusan. Seiring dengan peningkatan kapabilitas komputasi, AI telah banyak diimplementasikan di berbagai sektor, termasuk sektor pertanian untuk otomatisasi dan identifikasi masalah pada tanaman [12].

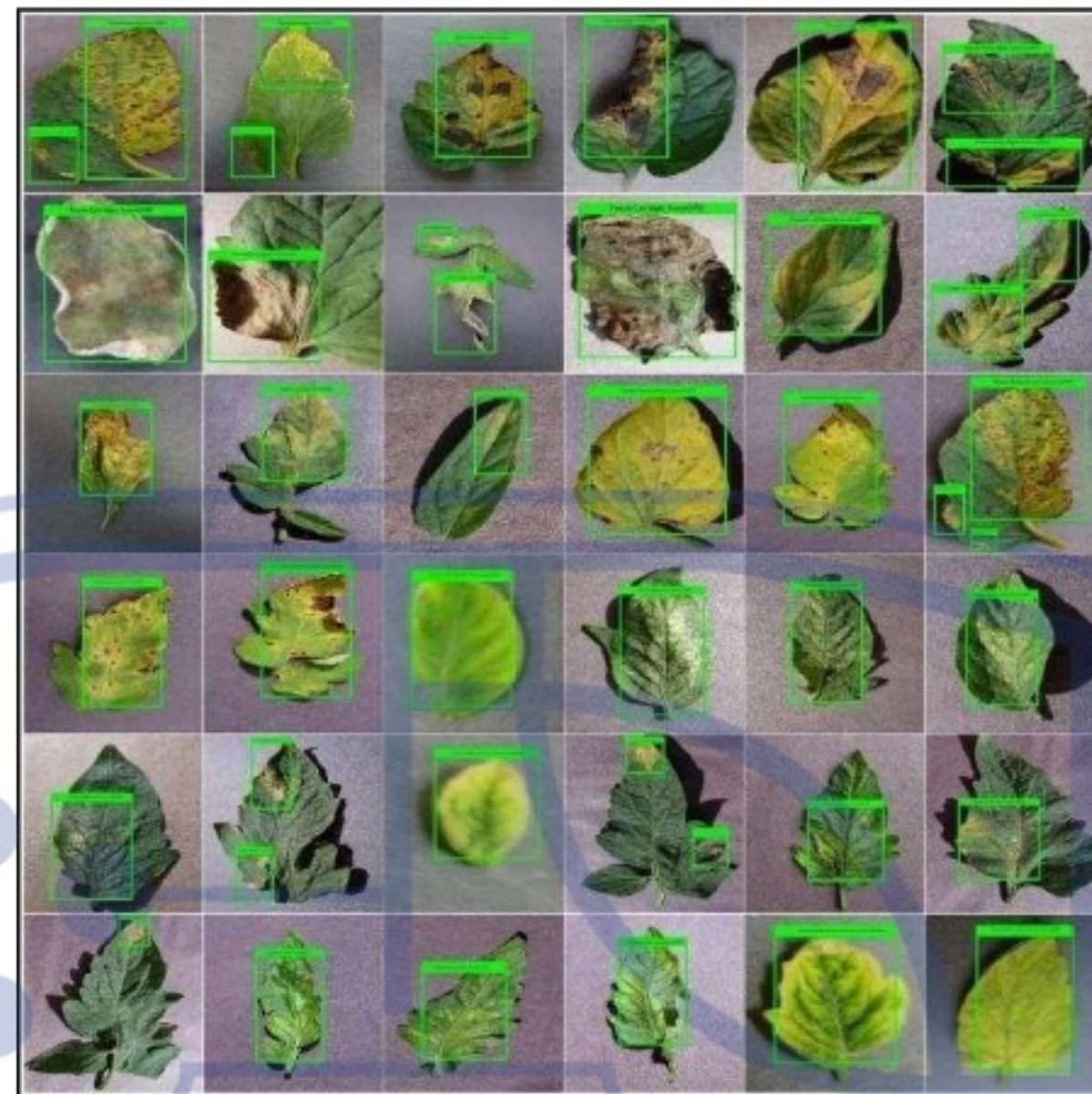
Dalam pemrosesan informasi visual, salah satu sub-bidang AI yang berkembang pesat adalah *Computer Vision*. Teknologi ini memungkinkan mesin untuk mengekstraksi, menganalisis, dan memahami informasi bermakna dari citra digital atau video. Untuk mendukung kinerja *computer vision*, diterapkan teknik pengolahan citra digital (*digital image processing*) [13]. Teknik ini mencakup manipulasi matematis terhadap matriks piksel citra, seperti konversi ruang warna, penyesuaian kontras, atau reduksi gangguan (*noise*) yang bertujuan menonjolkan fitur-fitur penting sebelum dianalisis lebih lanjut oleh algoritma *Deep Learning*.

2.2 Deteksi Objek (*Object Detection*)

Deteksi objek (*object detection*) merupakan salah satu konsep fundamental dan bidang penting dalam *computer vision*. Konsep ini tidak hanya berfokus pada pengenalan jenis objek di dalam sebuah gambar, tetapi juga menentukan di mana posisi objek tersebut berada. Pendekatan ini sangat berbeda dengan klasifikasi gambar konvensional yang hanya menghasilkan label untuk keseluruhan citra tanpa mengetahui lokasi spesifik dari objek tersebut [14].

Secara umum, deteksi objek bertujuan untuk menghasilkan tiga informasi utama dalam satu proses inferensi. Pertama, model harus mampu menentukan apakah suatu objek memang ada dalam area tertentu pada gambar. Kedua, model harus mengidentifikasi kelas atau kategori dari objek tersebut, misalnya dalam konteks penelitian ini, apakah daun yang dideteksi termasuk daun sehat, atau mengalami penyakit tertentu seperti bercak daun atau busuk ujung. Ketiga, model perlu menentukan koordinat dari *bounding box*, yaitu area

persegi yang mengelilingi objek tersebut agar sistem dapat menandai secara visual bagian gambar yang relevan [15].



Gambar 2.1 Bounding Box [16]

Salah satu elemen paling mendasar dalam sistem deteksi objek adalah bounding box atau kotak pembatas seperti pada gambar 2.1 di atas. Secara sederhana, bounding box adalah sebuah persegi panjang yang digunakan untuk menandai area di mana model memperkirakan keberadaan suatu objek [17]. Pada kasus penelitian ini, kotak menggambarkan bagian daun yang dideteksi mengandung penyakit, misalnya bercak, busuk, atau warna abnormal. Kotak pembatas ini memungkinkan sistem untuk secara visual menunjukkan hasil deteksi kepada pengguna aplikasi, sehingga pengguna dapat melihat secara langsung bagian daun yang bermasalah.

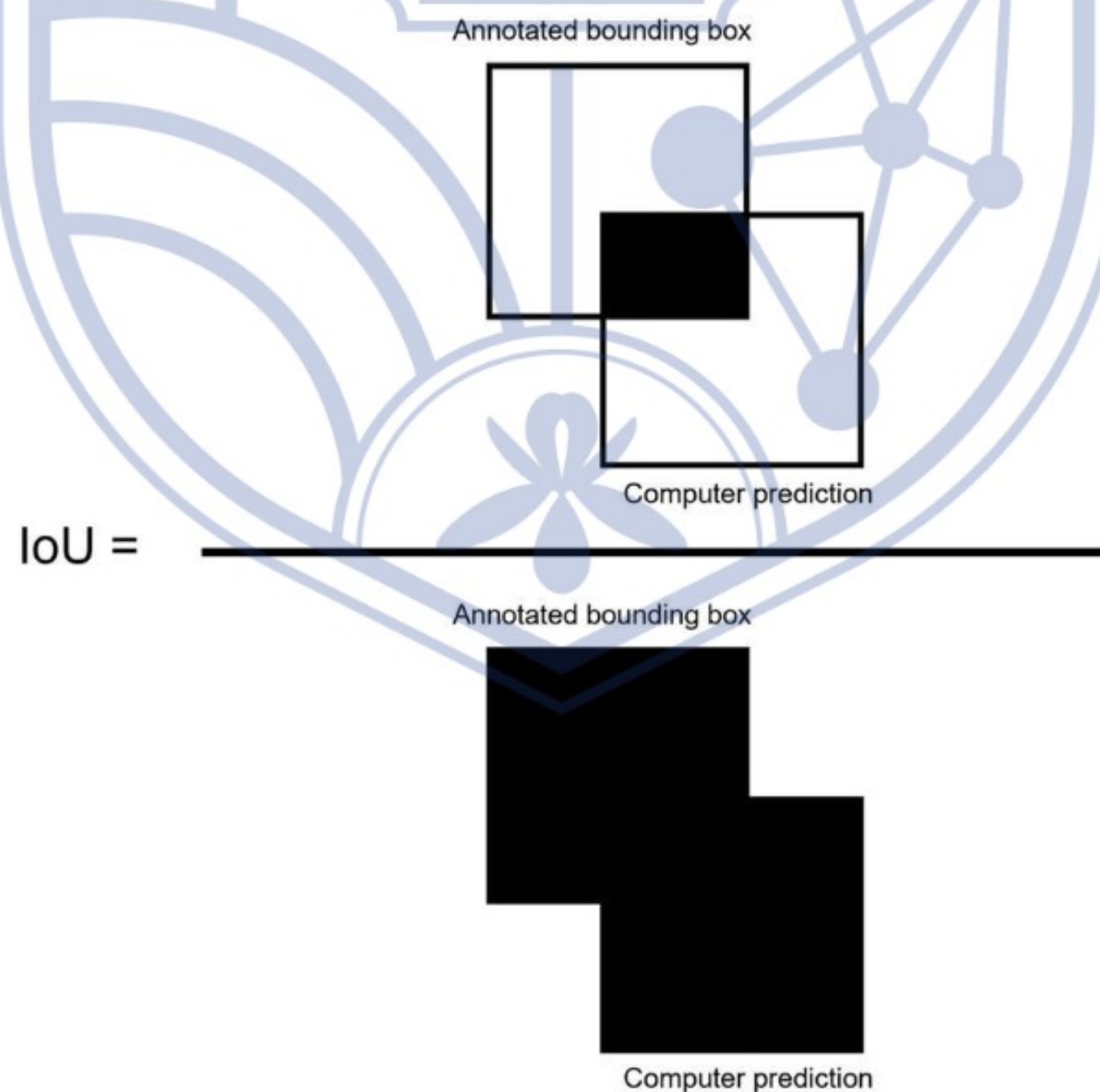
Bounding box biasanya direpresentasikan dalam empat parameter utama, yaitu titik koordinat titik kiri atas (x, y) dan ukuran lebar serta tinggi (*width, height*). Nilai – nilai koordinat ini biasanya dinormalisasikan terhadap ukuran gambar input (0 – 1), agar tetap proporsional meskipun ukuran citra berubah.

Tabel 2.1 Komponen Bounding Box [17]

Parameter	Keterangan
x	Koordinat sumbu horizontal titik kiri atas kotak
y	Koordinat sumbu vertikal titik kiri atas kotak

width (w)	Lebar kotak pembatas
height (h)	Tinggi kotak pembatas
confidence	Nilai kepercayaan model terhadap deteksi
class_id	Label atau kategori objek yang terdeteksi

Selama proses inferensi, model deteksi tidak hanya menghasilkan satu *bounding box*, melainkan bisa memunculkan banyak prediksi yang saling tumpang tindih. Oleh karena itu, diperlukan proses penyaringan hasil menggunakan teknik yang disebut *Non-Maximum Suppression (NMS)*. Algoritma *NMS* akan memilih kotak dengan skor kepercayaan tertinggi (*confidence score*) dan menghapus kotak lain yang memiliki tumpang tindih (*Intersection over Union / IoU*) agar hasil akhir tetap bersih dan tidak redundan. Untuk mengukur tingkat tumpang tindih antara dua kotak deteksi (prediksi dan *ground truth*) digunakan metrik *Intersection over Union (IoU)*.



Gambar 2.2 Intersection over Union [17]

Nilai *IoU* menentukan seberapa akurat *bounding box* prediksi terhadap posisi sebenarnya dari objek. Semakin tinggi nilai *IoU*, semakin baik kualitas deteksi model.

Rumus dari *Intersection over Union* (*IoU*) :

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}} \quad (1)$$

Keterangan :

B_p = *bounding box* hasil prediksi model

B_{gt} = *bounding box* sebenarnya (*ground truth*)

$\cap$ = luas daerah yang tumpang tindih

$\cup$ = luas daerah antara kedua kotak

Dalam praktiknya, prediksi dianggap benar jika nilai $IoU > 0.5$. Standar tersebut sering digunakan dalam evaluasi model deteksi untuk mengukur kualitasnya.

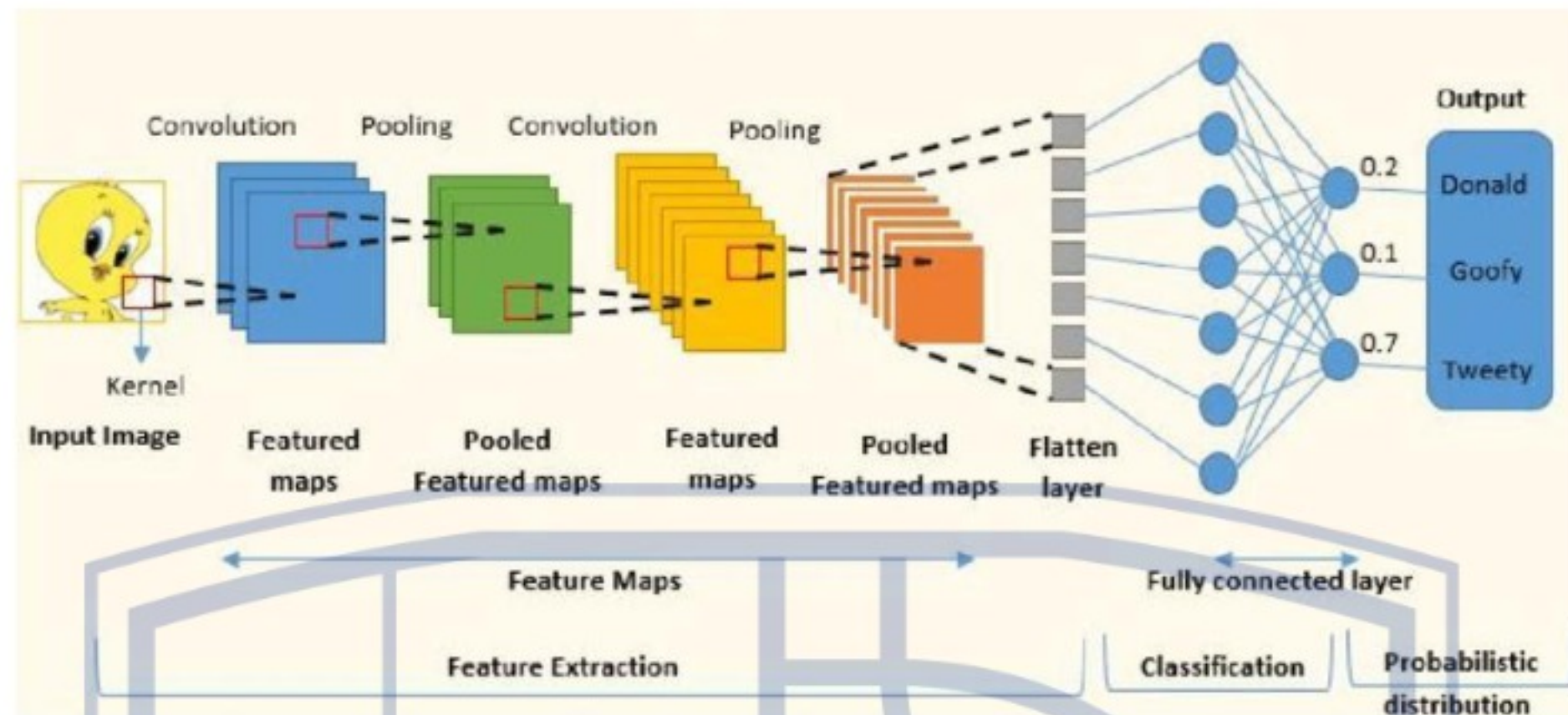
Dalam konteks implementasi di aplikasi mobile berbasis *Flutter* dan *TensorFlow Lite*, hasil *bounding box* dari model biasanya masih dalam skala koordinat normalisasi. Oleh karena itu, proses konversi (*scaling*) dilakukan agar koordinat kotak tersebut sesuai dengan ukuran tampilan pada layar *smartphone*. Dengan cara ini, kotak pembatas yang dihasilkan model dapat ditampilkan secara proporsional di atas gambar yang diambil pengguna, menjadikan hasil deteksi lebih intuitif dan mudah dipahami.

2.3 *Deep Learning dan Convolutional Neural Network (CNN)*

Dalam perkembangan teknologi *computer vision*, pendekatan yang paling banyak digunakan saat ini adalah *deep learning*. *Deep learning* merupakan bagian dari *machine learning* yang memanfaatkan jaringan saraf tiruan (*neural network*) dengan banyak lapisan (*deep layers*) untuk mempelajari representasi data secara otomatis. Berbeda dengan metode konvensional yang memerlukan ekstraksi fitur secara manual, *deep learning* mampu mengekstraksi fitur penting secara langsung dari data mentah seperti citra digital.

Salah satu arsitektur *deep learning* yang paling umum digunakan dalam pengolahan citra adalah *Convolutional Neural Network (CNN)*. *CNN* dirancang khusus untuk

memproses data berbentuk grid seperti gambar, dengan memanfaatkan operasi konvolusi untuk mengekstraksi fitur visual seperti tepi, tekstur, dan pola objek.



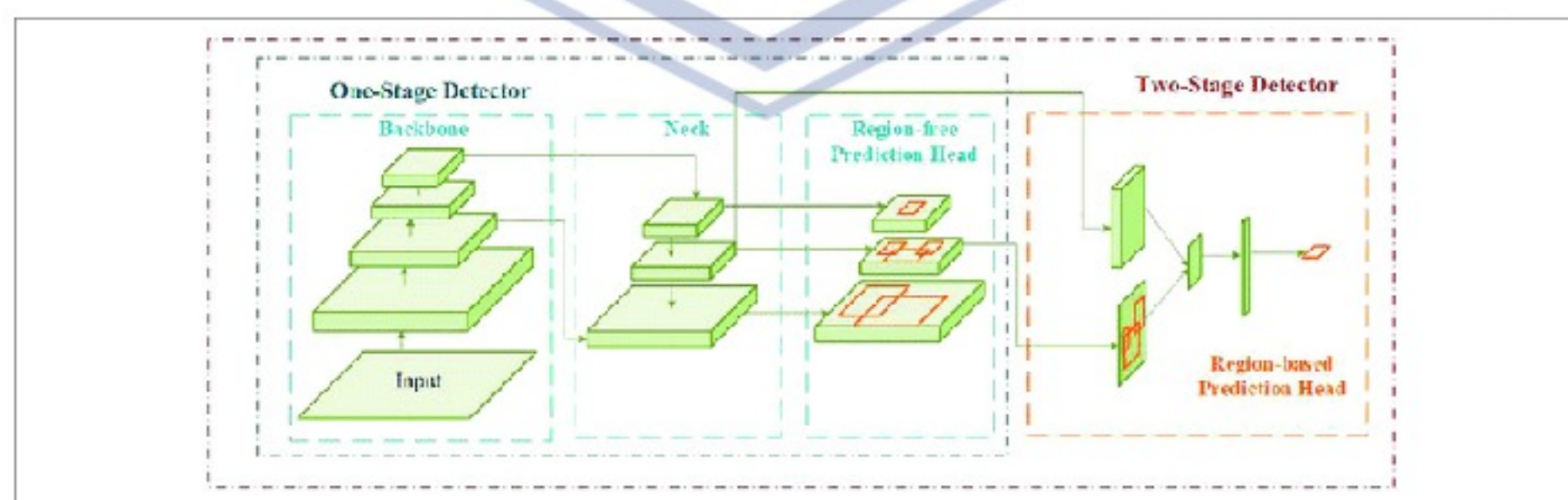
Gambar 2.3 Convolutional Neural Network (CNN) [18]

CNN terdiri dari beberapa komponen utama yaitu *convolution layer*, *pooling layer*, dan *fully connected layer*. Pada lapisan awal, CNN biasanya mendeteksi fitur sederhana seperti garis dan sudut, kemudian pada lapisan yang lebih dalam akan mengenali pola yang lebih kompleks seperti bentuk objek dan karakteristik spesifik.

Convolution layer berfungsi untuk mengekstraksi fitur dari citra, *pooling layer* digunakan untuk mengurangi dimensi fitur sehingga lebih efisien, sedangkan *fully connected layer* digunakan untuk proses klasifikasi [18].

2.4 Metode Deteksi Objek

Deteksi objek berbasis *deep learning* secara umum dibagi menjadi dua pendekatan utama, yaitu *two-stage detector* dan *one-stage detector*. Kedua pendekatan ini memiliki perbedaan mendasar dalam cara mendeteksi objek serta perbandingan akurasi dan kecepatan.



Gambar 2.4 Two-Stage Detector dan One-Stage Detector [19]

Pendekatan two-stage detector bekerja dalam dua tahap. Tahap pertama adalah menghasilkan kandidat lokasi objek menggunakan metode region proposal, kemudian tahap kedua adalah melakukan klasifikasi dan penyempurnaan bounding box. Sebaliknya, pendekatan one-stage detector melakukan proses deteksi objek dalam satu tahap (single forward pass). Model langsung memprediksi lokasi bounding box dan kelas objek secara bersamaan tanpa melalui tahap regional proposal. Contoh algoritma yang menggunakan pendekatan ini adalah R-CNN dan Faster R-CNN [17]. Metode ini dikenal memiliki akurasi tinggi, namun memiliki waktu inferensi yang relatif lebih lambat sehingga kurang cocok untuk aplikasi real-time.

2.5 YOLO (*You Only Look Once*)

Salah satu algoritma yang termasuk dalam kategori one-stage detector adalah YOLO (*You Only Look Once*). YOLO pertama kali diperkenalkan oleh Redmon et al. pada tahun 2016 sebagai pendekatan deteksi objek yang mampu bekerja secara real-time dengan akurasi yang kompetitif.

Konsep utama *YOLO* adalah melakukan deteksi objek hanya dalam satu kali proses (*single forward pass*) pada jaringan saraf. Berbeda dengan metode sebelumnya yang memisahkan proses pencarian lokasi objek dan klasifikasi, *YOLO* menggabungkan kedua proses tersebut dalam satu jaringan terpadu. Model membagi gambar menjadi grid dan untuk setiap grid memprediksi *bounding box*, *confidence score*, serta kelas objek [20].

Keunggulan utama YOLO terletak pada kecepatannya yang tinggi, sehingga sangat cocok untuk aplikasi real-time seperti sistem pengawasan, kendaraan otonom yang memerlukan pemrosesan cepat. Selain itu, YOLO juga memiliki arsitektur yang relatif sederhana dan efisien dibandingkan metode two-stage detector.

Seiring perkembangan teknologi, *YOLO* mengalami berbagai peningkatan dari versi ke versi untuk meningkatkan akurasi dan efisiensi. Perkembangan ini meliputi perbaikan pada arsitektur jaringan, teknik pelatihan, serta optimasi performa untuk berbagai kebutuhan.

2.6 YOLOv8 (*You Only Look Once Version 8*)

Berdasarkan perkembangan algoritma deteksi objek berbasis *YOLO* yang terus mengalami peningkatan dari waktu ke waktu, pada penelitian ini digunakan *YOLOv8* sebagai model utama untuk proses deteksi karena versi ini merupakan penyempurnaan terbaru dari keluarga *You Only Look Once (YOLO)*. *YOLOv8* dirancang dengan

menekankan keseimbangan antara akurasi dan efisiensi komputasi, sekaligus memberikan fleksibilitas tinggi dalam berbagai jenis tugas seperti *object detection*, *image classification*, hingga *instance segmentation*. Keunggulan utama *YOLOv8* terletak pada arsitekturnya yang modular dan kemudahan penerapan. Pendekatan ini membuat *YOLOv8* menjadi lebih mudah diintegrasikan dengan berbagai sistem, termasuk sistem, berbasis robotika atau pengendalian otomatis.

2.6.1 Sejarah Singkat YOLO

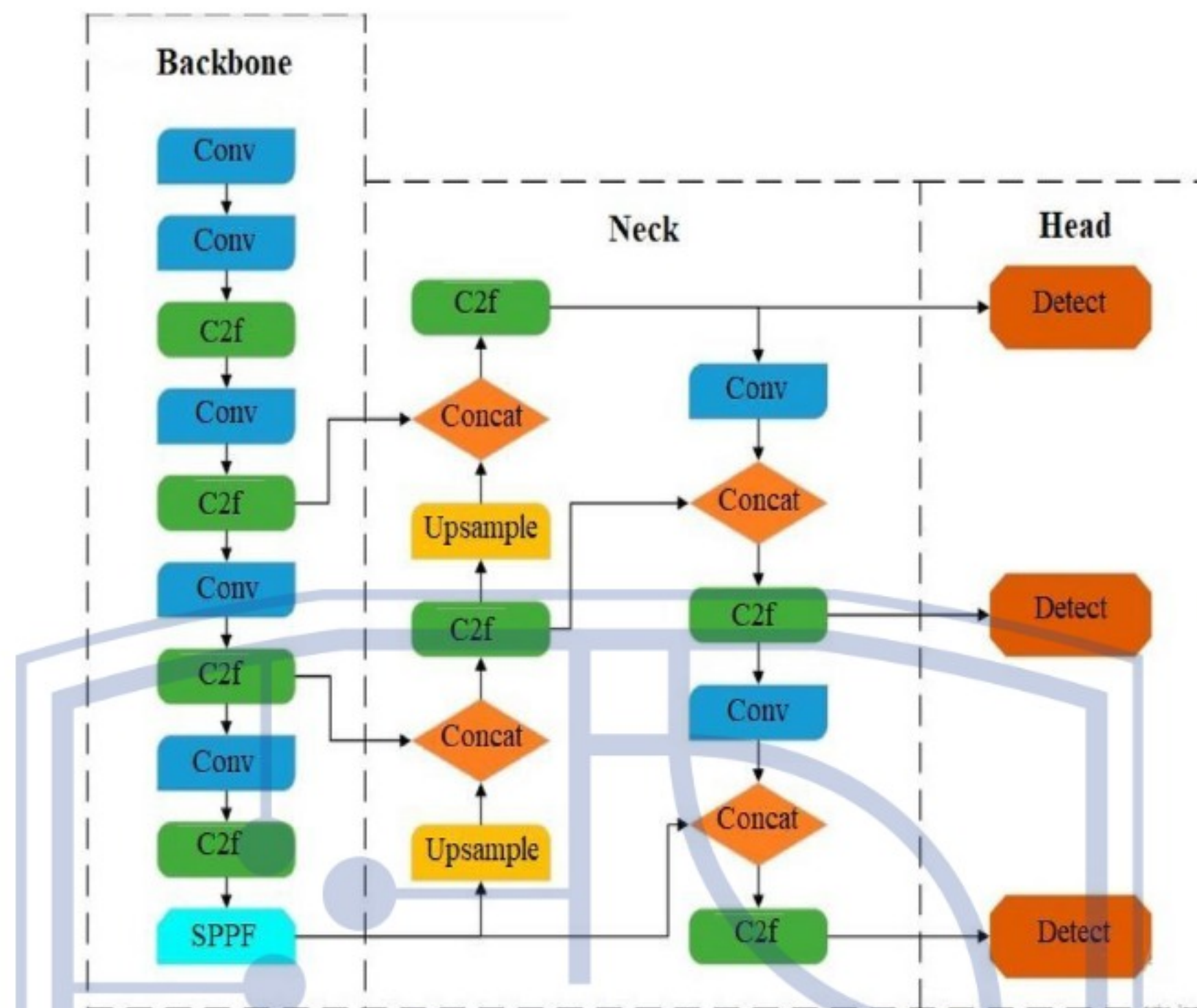
Perjalanan YOLO dimulai pada tahun 2015 ketika Redmon et al. memperkenalkan konsep deteksi objek real-time yang efisien melalui pendekatan “You Only Look Once”. Ide dasarnya adalah melakukan deteksi dalam satu kali forward pass pada jaringan, sehingga seluruh proses prediksi dapat dilakukan secara tepat tanpa harus memisahkan tahap region proposal seperti pada metode sebelumnya misalnya R-CNN [21]. Seiring waktu, berbagai versi YOLO dikembangkan untuk meningkatkan performa baik dari sisi kecepatan maupun akurasi.

Versi *YOLOv3* memperkenalkan *multi-scale feature fusion* melakukan arsitektur *Feature Pyramid Network (FPN)*, diikuti *YOLOv4* yang menambahkan teknik augmentasi dan optimasi *bag of freebies*, serta *YOLOv5* yang populer karena modularitas dan kemudahan implementasinya dalam lingkungan *PyTorch*. Selanjutnya, *YOLOv7* memperkenalkan penyempurnaan pada bagian *training strategy* dan *detection head*.

YOLOv8 hadir sebagai hasil evolusi dari semua versi tersebut dengan membawa pendekatan yang lebih modern. Salah satu perubahan mendasar adalah transisi ke model *anchor-free*, yang menghilangkan ketergantungan pada *predefined anchor boxes* [21], sehingga proses deteksi menjadi lebih adaptif terhadap berbagai bentuk dan ukuran objek. Selain itu, *YOLOv8* juga mengadopsi *decoupled head*, di mana jalur prediksi kelas dan kotak deteksi dipisahkan agar proses optimasi masing – masing dapat berlangsung lebih efektif dan independen [22].

2.6.2 Arsitektur YOLOv8

Secara garis besar, arsitektur *YOLOv8* terdiri atas tiga komponen utama yaitu *Backbone*, *Neck*, *Head* yang bekerja secara sinergis untuk menghasilkan deteksi yang cepat dan akurat. Struktur ini mengadopsi prinsip *end-to-end learning* sehingga seluruh proses, mulai dari ekstraksi fitur hingga prediksi akhir, dapat dilakukan dalam satu kesatuan jaringan konvolusional seperti pada gambar 2.5 di halaman selanjutnya [23].



Gambar 2.5 Arsitektur YOLOv8 [20]

a. Backbone

Bagian *backbone* berfungsi sebagai ekstraktor fitur utama dari citra masukan. Pada tahap ini, jaringan konvolusional mengekstraksi berbagai representasi visual mulai dari fitur dasar seperti tepi dan tekstur hingga pola kompleks di lapisan yang lebih dalam. *YOLOv8* menggunakan modul *C2f* (*Cross Stage Partial fused*) yang menggantikan modul *C3* pada versi sebelumnya. *C2f* mengoptimalkan efisiensi jumlah parameter dan mempercepat proses pelatihan dengan tetap menjaga kualitas representasi fitur yang dihasilkan [24].

b. Neck (Feature Fusion)

Setelah fitur diekstraksi oleh *backbone*, bagian *neck* bertugas menggabungkan informasi dari berbagai skala untuk memastikan deteksi objek berukuran kecil hingga besar dapat dilakukan dengan baik. *YOLOv8* mengombinasikan pendekatan *FPN* (*Feature Pyramid Network*) dan *PAN* (*Path Aggregation Network*) yang bekerja secara *top-down* dan *bottom-up* untuk memperkuat konsistensi fitur antar level [25].

c. Head (Detection Head)

Pada bagian akhir arsitektur, *YOLOv8* menggunakan *decoupled head* di mana jalur prediksi untuk kelas dan lokasi objek dipisahkan. Pendekatan ini memungkinkan model untuk mengoptimalkan masing – masing tugas secara lebih spesifik, sehingga meningkatkan stabilitas dan akurasi hasil deteksi. Sebagai model *anchor-free*, *YOLOv8* tidak lagi

bergantung pada *anchor box* yang telah ditentukan sebelumnya. Hal ini membuat proses deteksi menjadi lebih fleksibel, sederhana, dan cepat diterapkan pada berbagai kondisi data [26].

2.6.3 Penggunaan YOLOv8

YOLOv8 hadir dengan berbagai varian ukuran model, mulai dari YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), hingga YOLOv8x (extra large). Setiap varian menawarkan kompromi berbeda antara kecepatan dan akurasi. Dalam konteks aplikasi berbasis edge device versi ringan seperti YOLOv8n dan YOLOv8s menjadi pilihan ideal karena memiliki latensi rendah dan efisiensi tinggi [27].

Keunggulan desain modular YOLOv8 memungkinkan pengembang untuk memilih model yang paling sesuai dengan kebutuhan aplikasinya. Misalnya pada sistem pengawasan lalu lintas atau deteksi wajah di kamera keamanan, varian small sering digunakan karena mampu bekerja cepat dengan tingkat akurasi yang cukup tinggi. Di sisi lain, untuk aplikasi analisis medis berbasis citra seperti deteksi sel kanker atau klasifikasi jaringan, varian large atau extra large lebih disukai karena memerlukan pemahaman fitur yang lebih kompleks dan presisi yang lebih tinggi. Dengan demikian, YOLOv8 tidak hanya unggul dari sisi kecepatan, tetapi juga dari kemampuan beradaptasi terhadap berbagai jenis data dan kondisi pemrosesan yang berbeda.

Tabel 2.2 Perbandingan Varian Model YOLOv8 [28]

Varian	Ukuran Model	Kecepatan (ms / frame)	Akurasi (mAP@0.5)	Kebutuhan Komputasi	Keterangan
<i>YOLOv8n</i>	Sangat kecil	~2-5ms	0.78	Rendah	Cocok untuk mobile / IoT
<i>YOLOv8s</i>	Kecil	~5-10ms	0.81	Rendah – Sedang	Ideal untuk aplikasi real-time
<i>YOLOv8m</i>	Sedang	~10-20ms	0.85	Sedang	Kompromi antara kecepatan dan akurasi

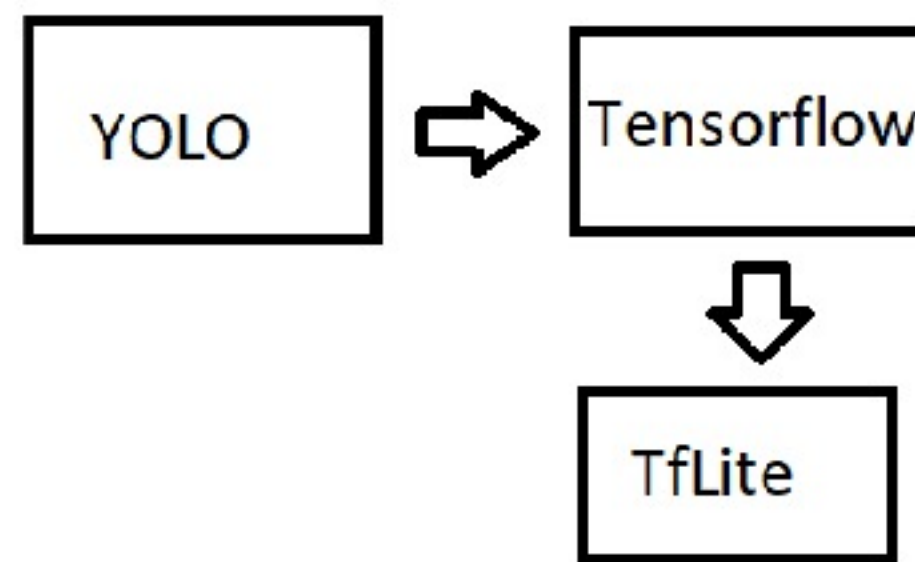
YOLOv8l	Besar	~20-30ms	0.88	Tinggi	Digunakan di sistem server
YOLOv8x	Sangat Besar	>30ms	0.90	Sangat Tinggi	Presisi tertinggi, cocok untuk riset

Model *YOLOv8* ini dapat diimplementasikan baik pada lingkungan komputasi tinggi maupun perangkat kecil atau *smartphone* untuk kebutuhan deteksi di lapangan. Dengan dukungan pustaka populer seperti *PyTorch* dan kompatibilitas dengan berbagai *framework* konversi seperti *ONNX* dan *TensorFlow Lite*, *YOLOv8* berhasil menjadi salah satu model deteksi objek yang paling adaptif dan serbaguna di era *deep learning* saat ini.

2.7 TensorFlow Lite (TF-Lite)

Dalam pengembangan sistem deteksi objek yang diimplementasikan pada perangkat *mobile*, efisiensi dan kinerja menjadi faktor utama yang harus diperhatikan. Model deteksi seperti *YOLOv8*, meskipun memiliki performa tinggi dalam hal akurasi, umumnya dikembangkan dan dilatih menggunakan *TensorFlow* dalam lingkungan komputasi yang kuat seperti GPU atau server. Namun, untuk dapat dijalankan secara *on-device* pada *smartphone* atau perangkat *edge*, diperlukan proses konversi dan optimasi agar model tersebut lebih ringan, cepat, dan hemat sumber daya. Pada tahap inilah *TensorFlow Lite* (TF-Lite) memainkan peran yang sangat penting sebagai solusi dari kebutuhan tersebut [29].

Salah satu keunggulan utama *TF-Lite* adalah kemampuannya melakukan optimasi model untuk menyesuaikan keterbatasan perangkat. Teknik seperti *quantization* memungkinkan bobot model dikurangi presisinya sehingga ukuran model menjadi lebih kecil dan proses inferensi lebih cepat. Selain itu, *pruning* dapat digunakan untuk menghilangkan bobot dengan kontribusi rendah terhadap prediksi, mengurangi kompleksitas komputasi tanpa menurunkan akurasi secara signifikan.



Gambar 2.6 TensorFlow Lite (TF-Lite) [30]

Kelebihan lainnya adalah fleksibilitas integrasi dengan berbagai *platform* pengembangan aplikasi seperti yang dapat dilihat pada Gambar 2.6. Sebagai contoh, pengembang Android dapat menggunakan Java API, C++ API, dan NNAPI untuk menghubungkan model berformat .tflite ke dalam aplikasi, sedangkan pengembang iOS dapat memanfaatkan C++ API. Struktur ini memungkinkan model dijalankan secara lokal pada perangkat tanpa memerlukan koneksi ke *server*, sehingga respons deteksi menjadi lebih cepat. Kemampuan TF-Lite sangat krusial karena memungkinkan penerapan model deteksi seperti YOLOv8 untuk memproses data langsung di lapangan, yang pada akhirnya memberikan solusi praktis dan efisien bagi pengguna perangkat *mobile*.

2.8 Dart

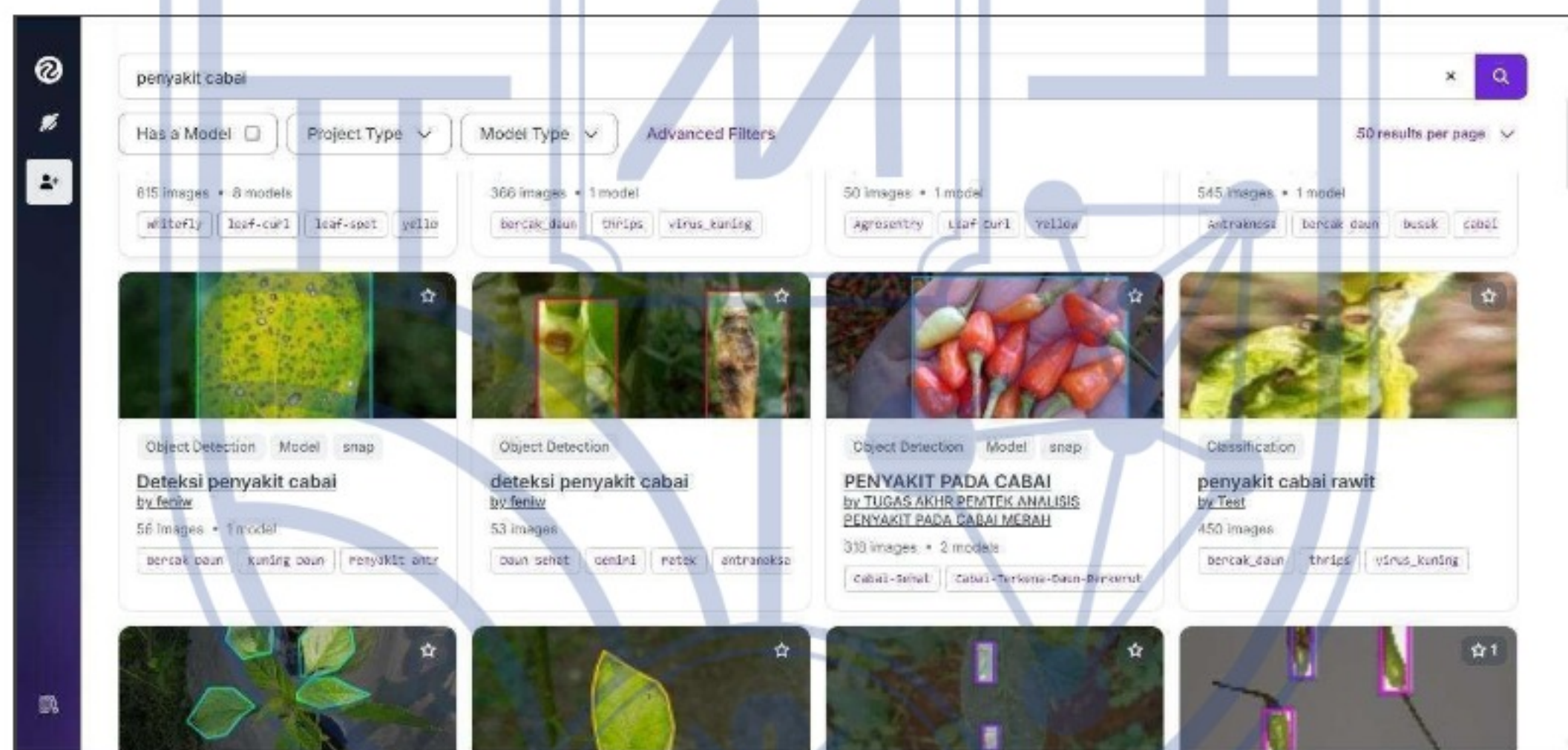
Selain konsep pendukung dalam *machine learning*, pengembangan aplikasi antarmuka yang mampu menampilkan hasil deteksi objek secara *real-time* juga merupakan aspek yang penting. Untuk kebutuhan tersebut, Dart menjadi salah satu bahasa pemrograman yang sering digunakan karena berbagai keunggulan yang ditawarkannya. Dart dikenal dengan sintaksnya yang modern, ringkas, dan mudah dipahami, sehingga memudahkan dalam penulisan kode yang bersih, terstruktur, dan mudah dipelihara. Integrasi Dart yang mulus dengan *framework* Flutter menjadi salah satu keunggulan utamanya, karena Flutter memungkinkan pembuatan aplikasi *mobile* lintas *platform* dengan performa tinggi [31].

Dalam pengembangan aplikasi seluler, Dart digunakan untuk membangun antarmuka yang responsif, mengelola state secara efisien, dan memproses input pengguna dengan lancar. Lebih dari sekadar tampilan visual, penggunaan Dart juga memberikan fleksibilitas untuk menambahkan fitur seperti integrasi basis data lokal. Hal ini memungkinkan sebuah aplikasi untuk tidak hanya menjadi sarana visualisasi, tetapi juga

berfungsi sebagai platform interaktif yang mendukung analisis data dan dokumentasi. Dengan demikian, bahasa pemrograman ini memainkan peran krusial dalam menjembatani fungsionalitas model deteksi objek dengan pengalaman pengguna yang intuitif dan fungsional.

2.9 Roboflow

Dalam penelitian ini, Roboflow digunakan untuk mengelola anotasi, pembagian dataset, pre-processing, augmentasi, dan ekspor data ke format YOLOv8. Setelah seluruh citra dan label diperiksa, data sumber terlebih dahulu ditempatkan ke dalam subset training, validation, dan testing. Pre-processing berupa auto-orient, penyeragaman resolusi, serta pemeriksaan label diterapkan secara konsisten, sedangkan augmentasi hanya diterapkan pada citra yang berada pada subset training. Dengan urutan tersebut, citra hasil augmentasi tidak berpindah ke subset validation atau testing sehingga risiko kebocoran data dapat dihindari [45].



Gambar 2.7 Roboflow [32]

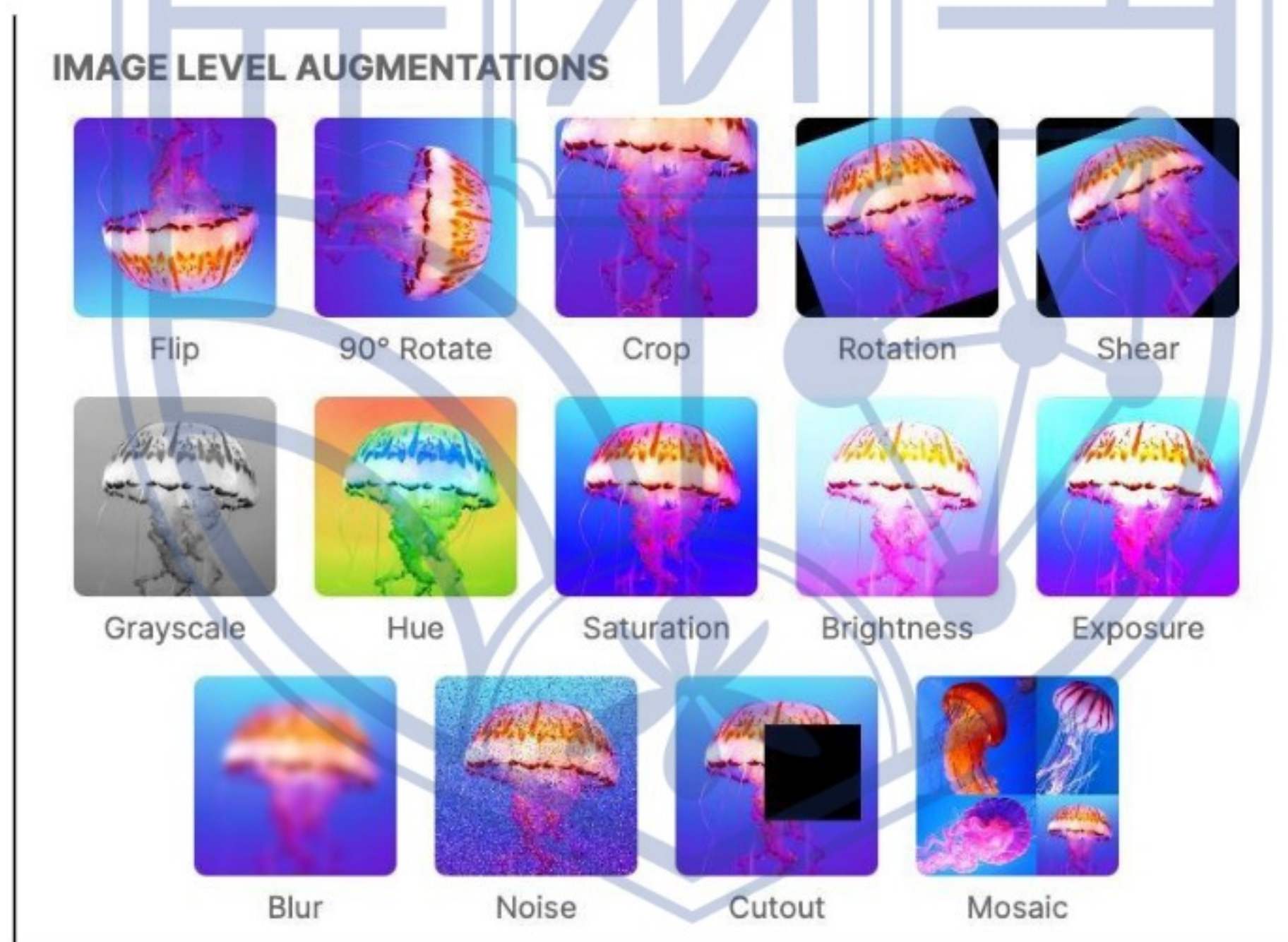
Roboflow menyediakan fasilitas ekspor dataset ke format yang kompatibel dengan YOLOv8, TensorFlow, PyTorch, dan framework lain. Fasilitas tersebut mengurangi kebutuhan konversi anotasi secara manual dan membantu menjaga konsistensi struktur folder, berkas label, serta konfigurasi dataset sebelum proses pelatihan dilakukan.

Versi dataset yang telah dibuat menjadi acuan tetap bagi proses pelatihan karena memuat citra, anotasi, pembagian subset, pre-processing, dan konfigurasi augmentasi yang digunakan. Dengan demikian, tahapan persiapan data dapat dijelaskan secara terstruktur dan direproduksi pada proses pelatihan berikutnya.

2.10 Teknik Augmentasi Data

Dalam sistem deteksi objek berbasis *deep learning* seperti *YOLOv8*, kualitas dan keragaman dataset memegang peranan yang sangat penting terhadap performa model. Dataset yang terbatas atau memiliki variasi kondisi yang rendah berpotensi menyebabkan model mengalami *overfitting*, yaitu kondisi dimana model bekerja sangat baik pada data pelatihan tetapi kurang mampu melakukan generalisasi pada data baru. Oleh karena itu, teknik augmentasi data menjadi salah satu strategi yang digunakan untuk meningkatkan keragaman data tanpa harus menambah jumlah citra secara manual [33].

Augmentasi data merupakan proses transformasi citra secara sistematis dengan tetap mempertahankan label dan anotasi bounding box yang sesuai. Pada penelitian ini, pembagian training, validation, dan testing dilakukan terlebih dahulu. Augmentasi kemudian diterapkan hanya pada subset training, sedangkan citra validation dan testing tidak diperbanyak melalui augmentasi. Roboflow juga mendokumentasikan bahwa augmentasi pada dataset version diterapkan pada citra yang ditandai sebagai Train [45].



Gambar 2.8 Teknik Augmentasi Dataset [33]

Beberapa teknik augmentasi utama yang digunakan dataset pada penelitian ini antara lain sebagai berikut :

a. Rotasi (*Rotation*)

Rotasi citra dilakukan untuk mensimulasikan variasi sudut pengambilan gambar daun tanaman di lapangan. Dalam kondisi nyata, pengguna aplikasi dapat mengambil gambar dengan orientasi yang berbeda – beda, sehingga model perlu dilatih agar tetap mampu mengenali objek meskipun citra mengalami rotasi tertentu.

b. Perubahan Kecerahan (*Brightness Adjustment*)

Perubahan tingkat kecerahan digunakan untuk mensimulasikan variasi kondisi pencahayaan, seperti cahaya matahari yang terlalu terang, bayangan, atau kondisi cahaya rendah. Dalam deteksi penyakit tanaman, perbedaan intensitas cahaya dapat memengaruhi tampilan warna dan bercak penyakit.

c. *Flip Horizontal* dan *Vertical*

Teknik *flip* atau pembalikan citra secara horizontal maupun vertikal digunakan untuk meningkatkan variasi tampilan objek tanpa mengubah karakteristik dasarnya. Pada dataset daun tanaman, bentuk dan pola penyakit tetap relevan meskipun citra dibalik.

d. *Scaling* dan *Zoom*

Augmentasi *scaling* atau *zoom* diterapkan untuk mensimulasikan variasi jarak pengambilan gambar antara kamera dan objek. Dalam praktik lapangan, pengguna dapat mengambil gambar daun dari jarak dekat maupun agak jauh.

Penerapan teknik augmentasi dilakukan secara terkontrol agar karakteristik objek dan posisi bounding box tetap relevan. Augmentasi yang berlebihan dapat menghasilkan distorsi citra yang tidak realistis. Oleh karena itu, rotasi, perubahan kecerahan, flip horizontal, dan zoom hanya digunakan pada data training untuk menambah variasi pembelajaran tanpa mencemari data validation dan testing.

Secara keseluruhan, augmentasi data berperan penting dalam meningkatkan kemampuan generalisasi model *YOLOv8*, terutama ketika diimplementasikan pada aplikasi mobile berbasis *TensorFlow Lite*. Dengan dataset yang lebih bervariasi, model diharapkan mampu memberikan hasil deteksi penyakit tanaman yang lebih akurat dan andal dalam berbagai kondisi lingkungan.

2.11 Metrik Evaluasi Model

Selain komponen utama dalam pengembangan model deteksi objek, proses evaluasi model juga merupakan tahapan yang sangat penting untuk memastikan bahwa model yang dihasilkan memiliki performa yang baik dan dapat diandalkan. Evaluasi model dilakukan untuk mengukur sejauh mana kemampuan model dalam mendeteksi objek secara akurat serta meminimalkan kesalahan prediksi.

Beberapa metrik evaluasi yang umum digunakan dalam sistem deteksi objek meliputi *Precision*, *Recall*, *F1-Score*, dan *mean Average Precision* (mAP). Metrik-metrik ini berfungsi untuk memberikan gambaran menyeluruh mengenai tingkat akurasi dan performa suatu model deteksi objek, seperti YOLOv8, dalam mengenali dan melokalisasi target pada sebuah citra.

Precision digunakan untuk mengukur tingkat ketepatan prediksi model terhadap objek yang terdeteksi. Nilai *precision* yang tinggi menunjukkan bahwa sebagian besar objek yang terdeteksi oleh model merupakan objek yang benar. *Recall* digunakan untuk mengukur kemampuan model dalam menemukan seluruh objek yang seharusnya terdeteksi dalam suatu citra. Sementara itu, *F1-Score* merupakan rata-rata harmonis antara *precision* dan *recall* yang digunakan untuk menyeimbangkan kedua metrik tersebut.

Selain itu, metrik utama dalam evaluasi deteksi objek adalah *mean Average Precision* (mAP). Metrik ini menghitung rata-rata nilai *precision* pada berbagai tingkat *recall*, sehingga mampu memberikan evaluasi yang lebih komprehensif terhadap performa model dalam berbagai kondisi.

Secara teori rumus *Precision* sebagai berikut :

$$Precision = \frac{TP}{TP+FP} \dots\dots\dots(2)$$

Secara teori rumus *Recall* sebagai berikut :

$$Recall = \frac{TP}{TP+FN} \dots\dots\dots(3)$$

Keterangan :

TP = *True Positive*

FP = *False Positive*

FN = *False Negative*

Secara teoritis rumus F1-Score sebagai berikut :

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \dots\dots\dots (4)$$

Sebagai hasil akhirnya mAP dihasilkan dari rumus *Average Precision* terlebih dahulu sebagai berikut :

$$Average Precision = \int_0^1 p(r) dr \dots\dots\dots (5)$$

Keterangan :

$p(r)$ = Nilai *Precision* sebagai fungsi dari *Recall*

dr = Perubahan sangat kecil pada nilai *Recall*

Lalu nilai mAP diperoleh dengan menghitung rata-rata nilai *Average Precision* sebelumnya, rumusnya sebagai berikut :

$$mAP = \frac{1}{k} \sum_{i=1}^k AP_i \dots\dots\dots (6)$$

Keterangan :

k = Jumlah kelas objek dalam dataset

AP_i = Nilai *Average Precision* untuk kelas ke- i

Dengan memahami dan menerapkan metrik evaluasi ini, secara sistematis dapat menilai keakuratan model *YOLOv8* dalam mendeteksi objek pada dataset yang digunakan. Metrik – metrik tersebut menjadi fondasi penting sebelum implementasi model ke perangkat mobile menggunakan *TensorFlow Lite*, memastikan sistem deteksi objek yang dikembangkan dapat bekerja secara andal di lapangan.

2.12 Jenis Penyakit Tanaman Yang Dijadikan Objek Deteksi

Pada penelitian ini, sistem deteksi penyakit tanaman difokuskan pada beberapa jenis penyakit utama yang umum menyerang tanaman pangan strategis di Indonesia, khususnya jagung, padi, dan cabai. Pemilihan kelas penyakit dilakukan berdasarkan tingkat prevalensi di lapangan, dampaknya terhadap produktivitas tanaman, serta karakteristik visual yang dapat diamati secara jelas melalui citra daun. Penyakit-penyakit tersebut memiliki pola gejala yang berbeda, seperti perubahan warna, munculnya bercak, lesi, atau deformasi jaringan daun, sehingga sesuai untuk dianalisis menggunakan pendekatan deteksi objek berbasis citra.

2.12.1 Antraknosa



Gambar 2.9 Antraknosa [34]

Antraknosa merupakan penyakit tanaman yang disebabkan oleh jamur dari genus *Colletotrichum* dan banyak menyerang tanaman hortikultura seperti cabai dan jagung. Gejala antraknosa pada daun umumnya ditandai dengan bercak berwarna coklat kehitaman, berbentuk bulat atau tidak beraturan, yang dapat meluas seiring perkembangan infeksi. Pada kondisi lembap, bagian tengah bercak sering tampak cekung dan dapat dikelilingi oleh halo berwarna kekuningan. Penyakit ini dapat menyebabkan penurunan kualitas dan kuantitas hasil panen secara signifikan apabila tidak dikendalikan sejak dini [35].

2.12.2 Bercak Cercospora



Gambar 2.10 Bercak Cercospora [36]

Penyakit bercak daun *Cercospora* disebabkan oleh jamur *Cercospora* spp. dan banyak ditemukan pada tanaman jagung dan cabai. Gejala utama penyakit ini berupa bercak kecil berbentuk oval hingga memanjang dengan warna coklat keabu-abuan di bagian tengah dan tepi yang lebih gelap. Bercak-bercak tersebut dapat menyatu dan menyebabkan daun mengalami klorosis hingga nekrosis. Pola bercak yang kontras dengan jaringan daun sehat menjadikan penyakit ini cukup representatif untuk dideteksi menggunakan model deteksi berbasis visi komputer [36].

2.12.3 Hawar Daun



Gambar 2.11 Hawar Daun [37]

Hawar daun merupakan istilah umum untuk penyakit yang menyebabkan kerusakan luas pada jaringan daun, biasanya ditandai dengan bercak besar berwarna coklat hingga kehitaman yang menyebar cepat. Penyakit ini dapat disebabkan oleh patogen jamur maupun bakteri, tergantung jenis tanamannya. Pada tanaman padi dan jagung, hawar daun menyebabkan daun mengering lebih awal sehingga mengganggu proses fotosintesis dan menurunkan hasil panen. Karakteristik visual berupa area nekrosis yang luas dan tepi tidak beraturan menjadi ciri utama penyakit ini [37].

2.12.4 Penyakit Bulai



Gambar 2.12 Penyakit Bulai [38]

Penyakit bulai merupakan penyakit penting pada tanaman jagung yang disebabkan oleh *Peronosclerospora* spp. Gejala awal bulai ditandai dengan perubahan warna daun menjadi kekuningan atau keputihan, diikuti oleh pertumbuhan jamur berwarna putih keabuan pada permukaan bawah daun. Infeksi berat dapat menyebabkan pertumbuhan tanaman terhambat dan kegagalan pembentukan tongkol. Perubahan warna daun yang kontras dan pola infeksi yang khas menjadikan penyakit bulai relevan sebagai objek deteksi berbasis citra [38].

2.12.5 Penyakit Gosong



Gambar 2.13 Penyakit Gosong [39]

Penyakit gosong atau *smut* disebabkan oleh jamur *Ustilago* spp., yang umum menyerang tanaman jagung. Penyakit ini ditandai dengan munculnya benjolan atau pembengkakan pada bagian tanaman, termasuk daun dan tongkol, yang kemudian pecah dan mengeluarkan massa spora berwarna hitam. Secara visual, penyakit ini memiliki ciri yang cukup mencolok sehingga dapat diidentifikasi melalui citra dengan resolusi yang memadai [39].

2.12.6 Penggerek Batang



Gambar 2.14 Penggerek Batang [40]

Penggerek batang merupakan hama utama pada tanaman padi dan jagung yang disebabkan oleh larva serangga tertentu. Serangan penggerek batang sering kali ditandai dengan gejala sekunder pada daun, seperti daun mengering, berlubang, atau berubah warna akibat terganggunya aliran nutrisi. Meskipun hama ini bukan penyakit daun secara langsung, dampaknya pada visual daun tetap relevan untuk dianalisis dalam sistem deteksi berbasis citra sebagai indikasi gangguan tanaman [40].

2.12.7 Penyakit Karat



Gambar 2.15 Penyakit Karat [41]

Penyakit karat disebabkan oleh jamur dari ordo *Pucciniales* dan umum ditemukan pada tanaman jagung. Gejalanya berupa bintik-bintik kecil berwarna coklat kemerahan hingga oranye pada permukaan daun. Seiring waktu, bintik tersebut dapat menyebar dan menyebabkan daun mengering. Pola bercak karat yang khas dan berulang menjadikan penyakit ini cocok untuk dideteksi menggunakan pendekatan deep learning [41].

2.12.8 Hawar Daun Bakteri (Bacterial Leaf Blight)



Gambar 2.16 Hawar Daun Bakteri (Bacterial Leaf Blight) [42]

Hawar Daun Bakteri (Bacterial Leaf Blight) merupakan penyakit penting pada tanaman padi yang disebabkan oleh bakteri *Xanthomonas oryzae* pv. *oryzae*. Penyakit ini ditandai dengan gejala berupa garis memanjang berwarna kekuningan pada daun yang kemudian berubah menjadi coklat dan mengering. Infeksi berat dapat menyebabkan kehilangan hasil panen yang signifikan. Pola garis memanjang dan perubahan warna daun menjadi ciri visual utama yang relevan untuk sistem deteksi otomatis [42].

2.12.9 Bercak Coklat (Brown Spot)



Gambar 2.17 Bercak Coklat (Brown Spot) [43]

Bercak Coklat (Brown Spot) adalah penyakit daun pada tanaman padi yang disebabkan oleh jamur *Bipolaris oryzae*. Penyakit ini ditandai dengan bercak coklat berbentuk oval dengan pusat berwarna lebih terang. Bercak Coklat (Brown Spot) dapat menyerang tanaman pada berbagai fase pertumbuhan dan berkontribusi terhadap penurunan kualitas gabah. Kontras antara bercak dan jaringan daun sehat menjadikannya sesuai sebagai kelas deteksi berbasis citra [43].

2.12.10 Penyakit Gosong Daun (Leaf Smut)



Gambar 2.18 Penyakit Gosong Daun (Leaf Smut) [44]

Penyakit Gosong Daun (Leaf Smut) merupakan penyakit pada tanaman padi yang disebabkan oleh jamur *Entyloma oryzae*. Penyakit ini ditandai dengan munculnya garis atau bercak kecil berwarna hitam di sepanjang tulang daun. Meskipun sering dianggap kurang agresif dibanding penyakit lain, Penyakit Gosong Daun (Leaf Smut) tetap berpotensi menurunkan kualitas tanaman apabila terjadi secara luas. Pola bercak hitam yang konsisten menjadi ciri visual utama yang dapat dipelajari oleh model deteksi [44].