BAB II TINJAUAN PUSTAKA

2.1. Data, Informasi, dan Pengetahuan

Dalam era digital saat ini, pemahaman mengenai data, informasi, dan pengetahuan menjadi sangat penting dalam pengembangan sistem berbasis teknologi. Ketiganya membentuk hirarki yang sangat berkaitan dan memiliki peran krusial dalam pengambilan keputusan yang berbasis bukti (*evidence-based decision making*)[2]. Data merupakan kumpulan fakta yang belum memiliki makna secara kontekstual, biasanya berupa angka, simbol, atau hasil pengukuran yang diperoleh melalui instrumen tertentu. Data kemudian diproses menjadi informasi, yaitu data yang telah diorganisasi atau dianalisis sehingga memiliki makna dan dapat diinterpretasikan secara efektif. Selanjutnya, informasi yang dipadukan dengan pengalaman, wawasan dan pemahaman manusia akan membentuk pengetahuan, yang memungkinkan pengambilan keputusan strategis serta penciptaan solusi atas berbagai permasalahan. Hirarki ini digambarkan dalam bentuk piramida, dimana data menjadi dasar, informasi berbeda di tengah sebagai hasil interpretasi, dan pengetahuan berada di puncak sebagai bentuk pemahaman yang aplikatif dan solutif[4].

Gambar 2. 1 Piramida Data, Informasi, Pengetahuan

2.2. Citra Digital

Citra digital merupakan representasi visual dari suatu objek atau pemandangan dalam bentuk format digital, yaitu dalam bentuk matriks dua dimensi yang terdiri dari elemen-elemen terkecil yang disebut piksel (*pixel*)[5]. Setiap piksel memiliki nilai intensitas tertentu yang menunjukkan tingkat kecerahan atau warna pada titik tersebut. Citra digital dapat diperoleh melalui berbagai perangkat seperti kamera digital, pemindah (*scanner*), atau sensor satelit, dan biasanya direpresentasikan dalam dua bentuk utama yaitu citra keabuan (*greyscale*) dan citra berwarna (RGB atau *Red Green Blue*)[5].

Gambar 2. 2 RGB

Pengolahan citra digital adalah serangkaian Teknik dan algoritma yang digunakan untuk memperbaiki kualitas citra, mengekstrak informasi, atau memodifikasi tampilan visual dari citra tersebut. Proses ini melibatkan berbagai tahapan seperti peningkatan citra (*image enhancement*), segmentasi, deteksi tepi hingga analisis pola[5]. Pengolahan citra digunakan secara luas dalam berbagai bidang, seperti kedokteran (misalnya dalam analisis medis), penginderaan jauh, industry manufaktur, keamanan, dan kecerdasan buatan. Tujuan utama dari pengolahan citra adalah untuk

menghasilkan citra yang lebih informatif atau mempermudah komputer maupun manusia dalam memahami dan menafsirkan isi citra tersebut.

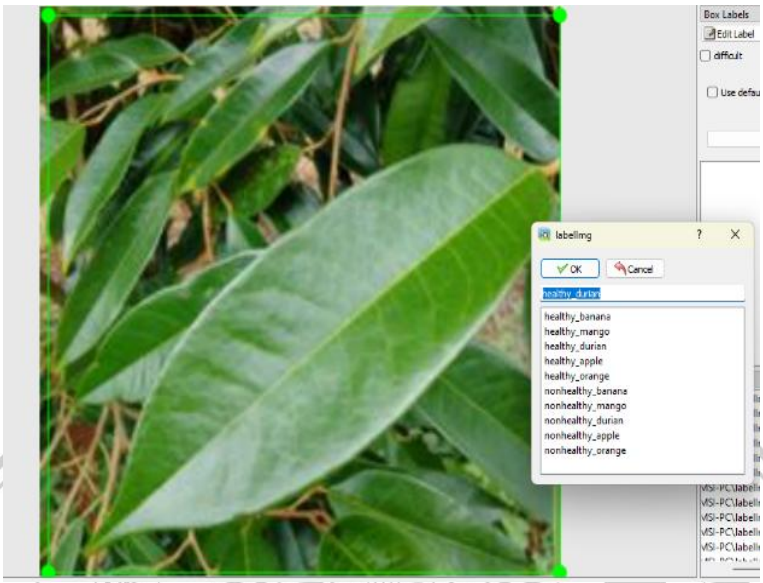
2.3. *Image Processing*

Image preprocessing merupakan tahap awal dalam pengolahan citra yang bertujuan untuk meningkatkan kualitas gambar dan menyiapkannya agar sesuai untuk proses pelatihan model, seperti normalisasi ukuran gambar, konversi warna, peningkatan kontras, dan pengurangan *noise*[5]. Setelah *preprocessing*, dataset kemudian dibagi menjadi dua bagian, yaitu data *train* untuk melatih model dan data *test* untuk menguji performa model, dengan masing-masing terdiri dari folder *images* (berisi gambar) dan *labels* (berisi file anotasi). Gambar diklasifikasikan berdasarkan kondisi daun ke dalam dua kategori utama, yaitu *healthy* dan *unhealthy*, serta dikelompokkan sesuai dengan jenis tanaman seperti *apple, banana, durian, mango* dan *orange*. Proses anotasi dilakukan menggunakan perangkat lunak *LabelImg*, yang menghasilkan file berformat .txt berisi informasi koordinat *bounding box* dan label kelas untuk setiap objek dalam gambar. File anotasi ini disimpan secara terpisah di folder labels yang sesuai dengan struktur *train* dan *test*, memastikan keteraturan data yang diperlukan untuk pelatihan model deteksi objek.

2.4. *LabelImg*

LabelImg adalah sebuah alat anotasi gambar berbasis antarmuka grafis (GUI) yang digunakan untuk memberi label pada objek dalam gambar[6], terutama untuk keperluan pelatihan model deteksi objek seperti YOLO. Gambar yang ditampilkan menunjukkan proses pelabelan objek daun menggunakan *LabelImg*, di mana pengguna membuat kotak pembatas (*bounding box*) berwarna hijau di sekitar daun durian yang menjadi objek utama. Setelah itu, jendela input label muncul dan memungkinkan pengguna memilih label dari daftar kelas, seperti *healthy_durian, unhealthy_mango*, hingga *healthy_orange*. Proses penggunaan *LabelImg* dimulai dengan membuka gambar yang ingin dianotasi, kemudian membuat *bounding box*

dan memberikan nama label sesuai kelas objek tersebut. Setelah semua objek diberi label, anotasi disimpan dalam format XML (Pascal VOC) atau TXT (YOLO), yang kemudian digunakan sebagai data pelatihan untuk model deteksi objek[6].

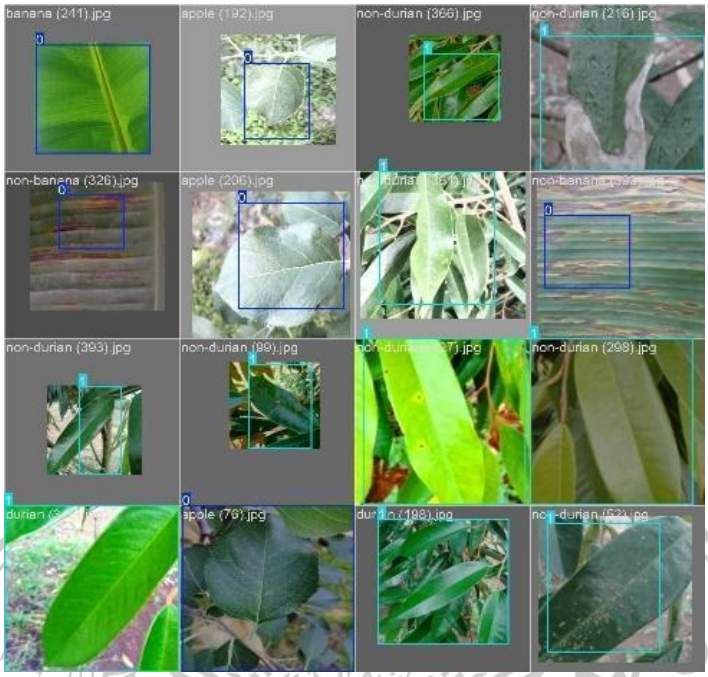


Gambar 2. 3 LabelImg

2.5. *Computer Vision*

Computer vision adalah cabang dari kecerdasan buatan (*Artificial Intelligence*/AI) yang memungkinkan komputer dan sistem untuk mengekstraksi, menganalisis, dan memahami informasi dari citra atau video digital secara otomatis, dengan cara yang menyerupai kemampuan penglihatan manusia[6]. Tujuan utama dari *computer vision* adalah untuk mengembangkan model dan algoritma yang dapat mengenali pola visual, menginterpretasikan objek, dan mengambil keputusan berdasarkan data visual yang diperoleh dari dunia nyata. Teknologi ini mencakup berbagai tugas seperti deteksi objek, klasifikasi gambar, segmentasi citra, pelacakan gerakan, pengenalan wajah, dan analisis citra medis. Dengan kemajuan dalam pembelajaran mesin, khususnya *deep learning* dan *penggunaan convolutional neural networks* (CNN), *computer vision* telah menjadi salah satu teknologi kunci dalam berbagai bidang seperti otomasi industri,

kendaraan otonom, keamanan, pertanian presisi, dan sistem diagnosis kesehatan berbasis gambar.

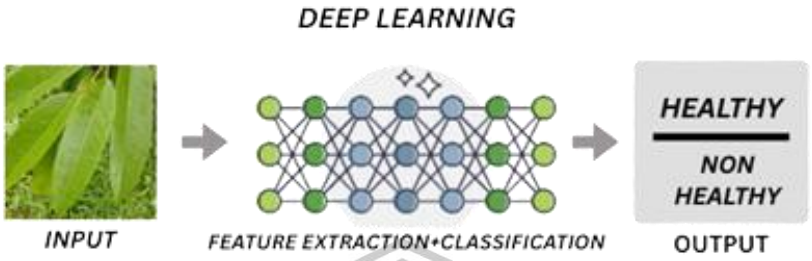


Gambar 2. 4 Hasil Train LabelImg dengan Computer Vision

2.6. *Deep Learning*

Deep learning adalah cabang dari pembelajaran mesin (*machine learning*) yang berfokus pada penggunaan jaringan syaraf tiruan berlapis-lapis *(deep neural networks*) untuk mempelajari representasi data secara otomatis dan hierarkis[7]. Berbeda dengan metode tradisional yang mengandalkan fitur buatan manusia (*handcrafted features), deep learning* mampu mengekstraksi fitur langsung dari data mentah, seperti gambar, teks, atau suara, tanpa intervensi manual yang signifikan. Arsitektur umum yang digunakan dalam *deep learning* meliputi *Convolutional Neural Networks* (CNN) untuk pemrosesan citra, *Recurrent Neural Networks* (RNN) dan *Long Short-Term Memory* (LSTM) untuk data berurutan seperti teks dan suara, serta transformer untuk tugas pemahaman bahasa alami[7]. *Deep learning* telah mendorong kemajuan besar dalam berbagai aplikasi seperti pengenalan wajah, deteksi penyakit melalui citra medis, pemrosesan bahasa alami, mobil otonom, dan sistem rekomendasi. Keunggulan utama dari *deep*

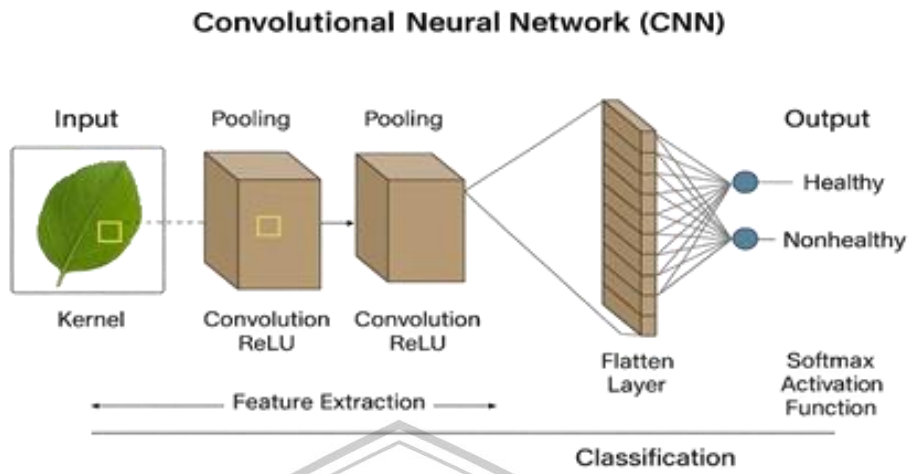
learning terletak pada kemampuannya dalam menangani data dalam jumlah besar dan kompleks, serta performa yang sangat baik dalam tugas-tugas klasifikasi dan prediksi.



Gambar 2. 5 Sistem Kerja Deep Learning

2.7. *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan salah satu arsitektur *deep learning* yang sangat efektif dalam mengolah data berbentuk *grid* seperti citra digital. CNN dirancang untuk secara otomatis dan adaptif belajar fitur dari data melalui proses konvolusi, *pooling*, dan lapisan *fully connected*. Proses ini dimulai dari *input layer* yang menerima citra sebagai masukan, diikuti oleh beberapa *convolution layer* yang mengekstraksi fitur-fitur penting, dan *ReLU activation* yang memperkenalkan non-linearitas. Kemudian, *pooling layer* digunakan untuk mereduksi dimensi spasial data sambil mempertahankan informasi utama. Setelah beberapa kali iterasi konvolusi dan *pooling*, data diflatkan melalui *flatten layer* dan dilanjutkan ke *fully connected layer* untuk pengambilan keputusan[8]. Akhirnya, *output layer* menggunakan *softmax function* untuk menghasilkan probabilitas klasifikasi ke dalam beberapa kelas. Keunggulan utama CNN terletak pada kemampuannya dalam mengekstraksi fitur spasial secara hierarkis, mulai dari tepi sederhana hingga pola yang lebih kompleks, tanpa memerlukan proses ekstraksi fitur secara manual. Arsitektur ini telah terbukti unggul dalam berbagai aplikasi pengenalan pola, seperti klasifikasi gambar, deteksi objek, dan segmentasi citra, serta menjadi fondasi dalam pengembangan sistem kecerdasan buatan berbasis pengolahan visual.

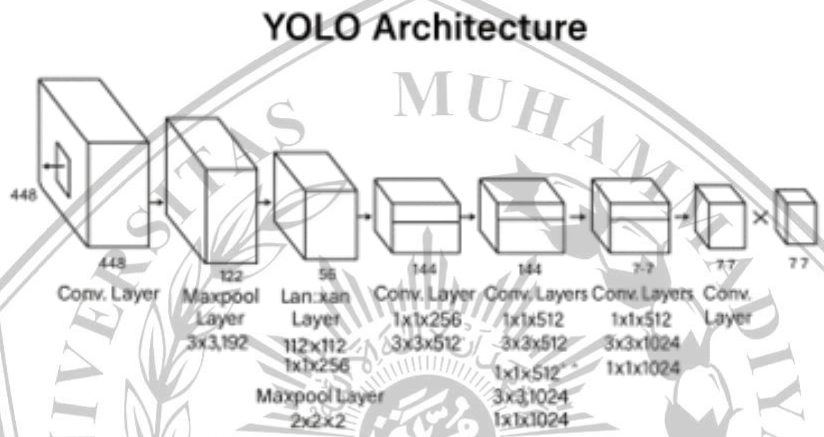


Gambar 2. 6 Sistem Kerja CNN

2.8. YOLOv8

You Only Look Once (YOLO) merupakan algoritma deteksi objek *real-time* berbasis *Convolutional Neural Network* (CNN) yang menggunakan pendekatan jaringan saraf tunggal untuk memproses seluruh gambar dalam satu evaluasi guna memprediksi *bounding box* dan probabilitas kelas secara langsung[7]. YOLO telah berkembang dari versi awal hingga YOLOv8, dengan peningkatan akurasi dan efisiensi pada setiap versi. YOLOv8, yang digunakan dalam penelitian ini, mampu mendeteksi objek dengan cepat dan akurat, ditunjukkan dengan nilai *Average Precision* (AP) mencapai 65%. Arsitektur YOLOv8 umumnya terdiri dari 24 lapisan konvolusi dan 2 lapisan *fully connected*, serta memanfaatkan lapisan reduksi 1x1 untuk mengurangi kedalaman *feature map* sebelum dilanjutkan ke lapisan konvolusi 3x3[9]. Berdasarkan arsitektur jaringan YOLO, proses dimulai dari input gambar berukuran 448x448 piksel yang melewati serangkaian lapisan konvolusi dan *max pooling* untuk mengekstraksi fitur[10]. Kombinasi lapisan 1x1 dan 3x3 digunakan untuk menangkap representasi spasial dan mengurangi kompleksitas jaringan. Hasil akhirnya dipetakan ke dalam grid 7x7 pada lapisan output, di mana setiap sel grid memprediksi *bounding box* dan kelas objek secara langsung, memungkinkan deteksi yang cepat dan efisien dalam satu kali proses evaluasi[11]. Pendekatan satu tahap (*single-stage*) yang digunakan oleh

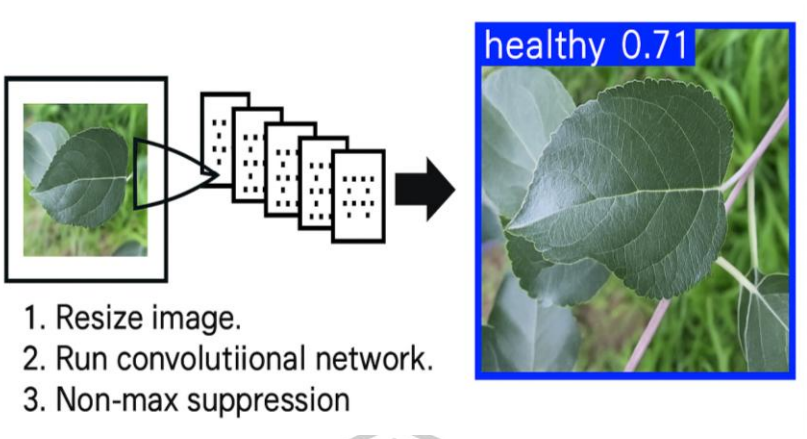
YOLO menjadikannya lebih cepat dibandingkan metode dua tahap seperti R-CNN, karena tidak memerlukan proses region proposal secara terpisah. Kecepatan dan efisiensi ini membuat YOLO sangat cocok untuk aplikasi *real-time* seperti sistem pemantauan video, kendaraan otonom, dan robotika. Selain itu, YOLOv8 juga mendukung *transfer learning* dan *fine-tuning* yang memudahkan adaptasi model terhadap dataset khusus dengan jumlah data terbatas.



Gambar 2. 7 Arsitektur YOLOv8

Proses deteksi objek dengan menggunakan YOLOv8 terdiri dari tiga tahapan utama, sebagaimana digambarkan pada Gambar[10]

1) Mengubah ukuran gambar ke resolusi tetap (660x660 piksel) yang sesuai dengan input model YOLOv8.
2) Memproses gambar melalui jaringan *convolutional* untuk mengenali fitur penting seperti bentuk, tekstur, dan pola.
3) Menghapus prediksi kotak yang tumpang tindih, hanya menyisakan kotak terbaik dengan nilai *confidence* tertinggi.



Gambar 2. 8 Sistem Deteksi YOLOv8

YOLO membagi gambar input ke dalam grid berukuran S × S. Setiap sel dalam grid bertanggung jawab mendeteksi objek jika pusat objek jatuh di dalam sel tersebut. Masing-masing sel memprediksi *bounding box* sebanyak B, beserta skor keyakinan (*confidence score*) dan probabilitas kelas bersyarat sebanyak C. Skor keyakinan mencerminkan keakuratan prediksi kotak terhadap objek yang sebenarnya. Jika tidak ada objek, nilainya nol; jika ada, nilainya setara dengan *Intersection over Union* (IoU) antara kotak prediksi dan *ground truth*[12]. Setiap kotak terdiri dari lima elemen: koordinat pusat (x, y), lebar (w), tinggi (h), dan skor keyakinan. Koordinat (x, y) dinormalisasi relatif terhadap sel grid, sedangkan (w, h) dinormalisasi terhadap ukuran gambar. Output YOLO berbentuk vektor berdimensi S × S × (B × (5 + C))[11]. Penentuan prediksi akhir didasarkan pada skor *class confidence*, yaitu gabungan dari probabilitas kelas dan skor keyakinan kotak. Skor ini menunjukkan seberapa tepat model dalam mengklasifikasi dan melokalisasi objek. Berikut cara menghitung *class confidence score* untuk setiap kontak prediksi, yaitu:

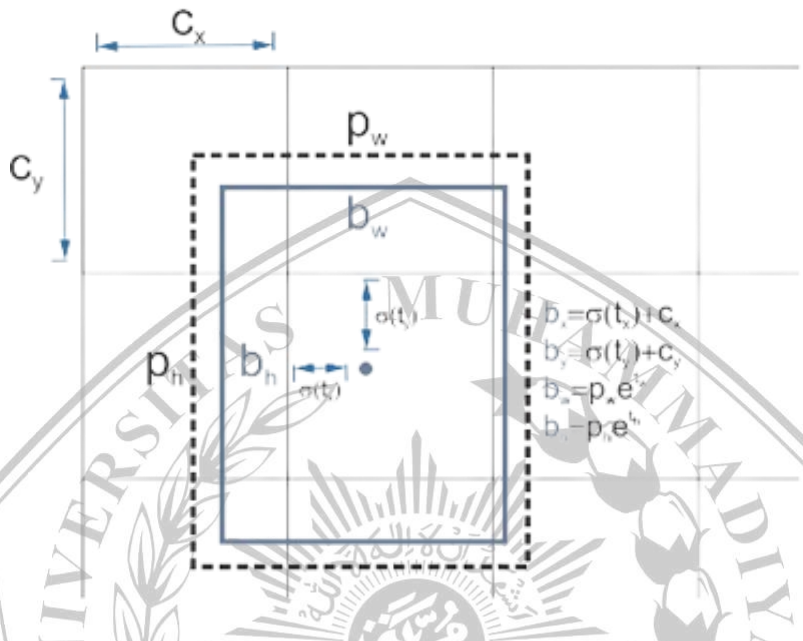
$$Pr\ Pr\ (Class_i \mid Object)\ .Pr\ Pr\ (Object)\ .IoU_{pred,truth} = Pr\ Pr\ (Class)\ .IoU_{pred,truth} \quad (1)$$

Keterangan :

a. **Pr(Class$_i$ | Object):** Probabilitas bersyarat bahwa objek termasuk dalam kelas ke-i, jika memang terdapat objek.

b. **Pr(Object):** Probabilitas keberadaan objek dalam kotak prediksi.

c. **$IoU_{pred,truth}$:** Tingkat kesesuaian antara kotak prediksi dan kotak kebenaran (*ground truth*).



Gambar 2. 9 Bounding Box YOLOv8

Berdasarkan persamaan tersebut, diperoleh nilai *confidence* untuk setiap kelas tertentu. Nilai ini mencerminkan seberapa besar kemungkinan suatu kelas muncul di dalam kotak prediksi, serta seberapa tepat posisi kotak tersebut. Seperti yang ditunjukkan pada gambar di atas, YOLOv8 memperlakukan deteksi objek sebagai masalah regresi. Metode ini membagi gambar ke dalam grid, di mana setiap sel grid memprediksi *bounding box*, skor keyakinan untuk setiap kotak, dan probabilitas dari masing-masing kelas[10].

2.9. Deteksi Penyakit Daun

Deteksi penyakit pada daun tanaman merupakan salah satu langkah penting dalam menjaga kesehatan tanaman dan meningkatkan hasil pertanian. Teknologi berbasis pengolahan citra dan kecerdasan buatan, seperti *Convolutional Neural Network* (CNN) dan algoritma deteksi objek

seperti YOLOv8, digunakan untuk mengidentifikasi gejala awal penyakit melalui gambar daun. Dengan mendeteksi perubahan warna, bentuk, atau pola pada daun secara otomatis, sistem ini memungkinkan petani untuk mengambil tindakan cepat dan tepat, sehingga mengurangi risiko penyebaran penyakit dan meminimalkan kerugian produksi[8].

2.9.1. *Apple*

Tanaman apel memiliki daun berbentuk oval dengan warna hijau cerah yang menandakan kondisi sehat. Daun sehat akan tampak rata, segar, dan bebas dari bercak atau kerusakan. Namun, daun yang tidak sehat sering menunjukkan gejala penyakit seperti bercak hitam akibat apple *scab* atau warna kuning kecoklatan dari infeksi karat. Selain perubahan warna, daun juga bisa menggulung, layu, atau rontok jika penyakit sudah berkembang lebih parah.

Gambar 2. 10 *Apple Healthy*

Gambar 2. 11 *Apple Unhealthy*

2.9.2. *Banana*

Daun tanaman pisang berbentuk panjang lebar dan berwarna hijau merata ketika dalam kondisi sehat. Permukaannya licin dan lentur, serta tidak terdapat bintik atau robekan alami. Ketika terserang penyakit seperti sigatoka atau layu bakteri, daun pisang akan menunjukkan bercak hitam memanjang, warna menguning di tepi, dan akhirnya mengalami pengeringan atau robek. Gejala tersebut secara signifikan mengganggu proses fotosintesis dan pertumbuhan tanaman.

Gambar 2. 12 *Banana Healthy*

Gambar 2. 13 *Banana Unhealthy*

2.9.3. *Durian*

Tanaman durian memiliki daun lonjong berwarna hijau gelap di permukaan atas dan coklat keemasan di bagian bawah. Daun sehat biasanya keras, tidak mudah rontok, dan tidak memiliki bintik atau cacat. Jika terserang penyakit seperti busuk akar atau infeksi jamur, daun durian dapat menguning, memperlihatkan bintik-bintik kecil, serta tampak melengkung atau rontok lebih awal dari waktunya. Gejala ini bisa menjadi indikasi gangguan serius pada sistem akar dan nutrisi tanaman.

Gambar 2. 14 *Durian Healthy*

Gambar 2. 15 *Durian Unhealthy*

2.9.4. *Mango*

Daun mangga sehat berwarna hijau tua dengan permukaan mengkilap dan bentuk yang simetris. Daunnya kuat dan tidak mudah robek atau berubah warna. Ketika terkena penyakit seperti *anthracnose* atau embun tepung, daun akan mengalami bercak hitam, kusam, hingga muncul

lapisan putih tipis di permukaannya. Infeksi ini dapat menyebabkan kerusakan jaringan daun dan menghambat produktivitas pohon mangga secara keseluruhan.

Gambar 2. 16 *Mango Healthy*

Gambar 2. 17 *Mango Unhealthy*

2.9.5. *Orange*

Tanaman jeruk memiliki daun berbentuk oval dengan warna hijau terang hingga hijau tua, mengilap, dan tanpa bercak saat dalam kondisi sehat. Namun, saat terserang penyakit seperti *citrus canker* atau *virus tristeza*, daun dapat menunjukkan gejala bercak kuning atau coklat, luka berkerak, keriting, dan kerontokan dini. Gangguan pada daun jeruk ini tidak hanya mempengaruhi penampilan tanaman, tetapi juga dapat berdampak langsung pada kualitas dan kuantitas buah yang dihasilkan.

Gambar 2. 18 *Orange Healthy*

Gambar 2. 19 *Orange Unhealthy*

2.10. Data Preprocessing

Data *preprocessing* adalah tahap awal yang krusial dalam proses pelatihan model *machine learning* atau *deep learning* [13], termasuk dalam

tugas deteksi objek. Tahapan ini bertujuan untuk memastikan bahwa data yang digunakan berada dalam kondisi bersih, seragam, dan siap diproses oleh algoritma pembelajaran. Proses *preprocessing* dapat mencakup berbagai langkah seperti normalisasi ukuran gambar, konversi format file (misalnya dari XML ke TXT untuk format YOLO), perubahan skala piksel gambar (*rescaling*) agar bernilai antara 0 hingga 1, serta augmentasi data untuk meningkatkan jumlah dan keragaman data latih[12]. Augmentasi dapat meliputi rotasi*, flipping, zooming*, atau perubahan pencahayaan agar model lebih *robust* terhadap variasi di dunia nyata.

Selain itu, dalam konteks deteksi objek, *preprocessing* juga mencakup penyesuaian koordinat *bounding box* setelah augmentasi dilakukan agar tetap akurat sesuai posisi objek pada gambar baru. Dataset juga perlu diklasifikasikan ke dalam direktori yang sesuai seperti *training, validation*, dan *test set,* agar evaluasi performa model lebih terstruktur. *Preprocessing* tidak hanya membantu dalam mempercepat konvergensi model, tetapi juga berperan penting dalam meningkatkan akurasi dan generalisasi model terhadap data baru yang belum pernah dilihat sebelumnya.

2.11. *Labels* dan *Label Correlogram*

Pada tahap *labeling*, proses anotasi dilakukan dengan mengklasifikasikan objek daun ke dalam dua kategori utama, yaitu *healthy* (sehat) dan *unhealthy* (tidak sehat). Grafik batang pada gambar memperlihatkan distribusi jumlah objek untuk masing-masing kelas, di mana jumlah daun sehat mendominasi dataset dengan lebih dari 1.000 instansi, sementara daun tidak sehat tercatat sekitar 800 instansi[14]. Visualisasi lain yang ditampilkan termasuk persebaran *bounding box* dalam gambar, yang membantu mengidentifikasi konsistensi anotasi dan distribusi spasial objek dalam dataset.

Label *correlogram* memberikan wawasan yang lebih mendalam mengenai karakteristik *bounding box* dari masing-masing anotasi. *Scatter*

plot dan *histogram* menunjukkan persebaran posisi koordinat pusat (x, y) dan ukuran *bounding box* (*width, height*)[14]. Terlihat bahwa mayoritas *bounding box* terkonsentrasi di tengah gambar (sekitar koordinat 0.5), sementara ukuran *bounding box* bervariasi namun cenderung besar (*width* dan *height* mendekati 1.0). Informasi ini sangat penting untuk evaluasi kualitas data anotasi sebelum digunakan dalam pelatihan model deteksi objek, karena ketidakseimbangan atau anotasi yang tidak konsisten dapat mempengaruhi performa model.

2.12. YAML

YAML (*Yet Another Markup Language*) merupakan format file berbasis teks yang sederhana dan mudah dibaca manusia, banyak digunakan untuk menyimpan konfigurasi dalam berbagai proyek *machine learning*, termasuk pelatihan model deteksi objek seperti YOLO (*You Only Look Once*)[15]. Dalam konteks deteksi objek, file YAML digunakan untuk mendefinisikan konfigurasi dataset yang mencakup beberapa elemen penting, seperti jalur direktori untuk data pelatihan (*train*), validasi (val), dan opsional data pengujian (*test*), jumlah total kelas (*nc atau number of classes*), serta daftar nama kelas atau label (*names*) yang akan dideteksi oleh model. Struktur YAML sangat sensitif terhadap spasi dan indentasi, sehingga penulisannya harus tepat agar tidak terjadi kesalahan saat parsing oleh sistem[15]. Contoh struktur dasar file YAML untuk YOLO adalah sebagai berikut:

```yaml
path: /content/drive/MyDrive/TA Kens Urganis/leaf disease.v2i.yolov8
train: train/images
val: test/images
nc: 10
names: ['apple', 'banana', 'durian', 'mango',
        'non-apple', 'non-banana', 'non-durian',
        'non-mango', 'non-orange', 'orange']
```

Gambar 2. 20 File YAML

File YAML ini kemudian dibaca oleh *script* pelatihan YOLO agar model mengetahui di mana letak data gambar dan anotasi, berapa banyak kelas yang harus diprediksi, serta nama label yang digunakan sebagai

referensi selama proses pelatihan, validasi, dan inferensi. YAML menjadi jembatan penting antara struktur data anotasi dan pemahaman model terhadap kelas-kelas objek yang ada dalam dataset.

2.13. *OS* dan *Glob*

OS dan *Glob* adalah dua modul penting dalam bahasa pemrograman *Python* yang sering digunakan untuk menangani file dan direktori, terutama dalam proses pra-pemrosesan data pada proyek *machine learning* dan *computer vision*[15]. Modul os (*Operating System*) memungkinkan interaksi langsung dengan sistem file komputer, seperti membuat folder, membaca daftar file dalam direktori, menggabungkan path file menggunakan *os.path.join*, serta melakukan operasi lain seperti memindahkan, mengganti nama, atau menghapus file. Modul ini memberikan fleksibilitas dalam mengelola struktur file yang kompleks, sehingga sangat membantu dalam menyiapkan dataset secara otomatis.

Sementara itu, *glob* digunakan untuk melakukan pencarian file berdasarkan pola tertentu (*pattern matching*). Modul ini berguna ketika kita ingin mengambil semua file dengan ekstensi tertentu (misalnya, .jpg, .png, atau .txt) dari suatu folder atau subfolder. Dengan menggunakan *glob.glob(*'path/*.jpg'), kita bisa mengakses seluruh file gambar dalam direktori tersebut tanpa harus menyebutkan nama satu per satu[16]. Dalam *workflow* deteksi objek, glob sering digunakan untuk mengumpulkan file gambar dan file anotasi agar bisa diproses secara *batch*, sementara os digunakan untuk membuat struktur folder dataset, memeriksa keberadaan file, atau menyimpan hasil olahan. Kombinasi keduanya memungkinkan proses automasi dataset menjadi lebih efisien dan dinamis.

2.14. *Python*

Python adalah bahasa pemrograman tingkat tinggi yang bersifat *interpreted*, *open-source*, dan multiparadigma, yang pertama kali dikembangkan oleh Guido van Rossum dan dirilis pada tahun 1991. *Python* dikenal karena sintaksisnya yang sederhana dan mudah dipahami, sehingga

sangat cocok untuk pemula maupun pengembang profesional. Bahasa ini mendukung berbagai paradigma pemrograman seperti prosedural, berorientasi objek, dan fungsional. *Python* memiliki komunitas yang besar dan aktif serta ekosistem pustaka (*library*) yang sangat luas, seperti *NumPy* dan *Pandas* untuk analisis data, *Matplotlib* dan *Seaborn* untuk visualisasi, serta *TensorFlow*, *PyTorch*, dan *OpenCV* untuk pengembangan sistem kecerdasan buatan dan computer vision [17].

Kelebihan *Python* juga terletak pada kemampuannya untuk digunakan dalam berbagai platform dan domain aplikasi, termasuk pengembangan web, otomasi sistem, analisis data, pengolahan citra, pembelajaran mesin, serta *Internet of Things* (IoT). Dengan alat bantu seperti *Jupyter Notebook*, pengembang dapat menulis, menjalankan, dan mendokumentasikan kode secara interaktif, menjadikan *Python* sangat populer di kalangan ilmuwan data dan peneliti. Ketersediaan paket pihak ketiga dan kemampuan integrasi dengan bahasa lain seperti C/C++, Java, dan R juga memperkuat posisinya sebagai salah satu bahasa pemrograman paling serbaguna dan banyak digunakan di dunia teknologi saat ini.

2.15. *Ultralytics*

Ultralytics adalah organisasi pengembang perangkat lunak yang dikenal luas sebagai pencipta dan pengelola utama dari model deteksi objek YOLO versi terbaru, terutama YOLOv5 hingga YOLOv8[15]. Mereka menyediakan pustaka *Python* bernama *ultralytics*, yang menyederhanakan proses implementasi model deteksi objek berbasis YOLO melalui antarmuka yang mudah digunakan, fleksibel, dan didukung dokumentasi lengkap. Pustaka ini memungkinkan pengguna untuk melakukan pelatihan (*training*), validasi (*validation*), inferensi (*inference*), hingga evaluasi performa model hanya dengan beberapa baris kode. Selain itu, *Ultralytics* aktif mengembangkan fitur-fitur tambahan seperti pelacakan objek (*object tracking*), segmentasi, serta integrasi dengan berbagai format dataset populer seperti COCO dan YOLO[15]. Kemudahan instalasi, efisiensi

model, serta pembaruan yang konsisten menjadikan *Ultralytics* sebagai salah satu platform utama bagi peneliti, pengembang, dan praktisi computer vision di seluruh dunia.

2.16. *OpenCV*

OpenCV (*Open Source Computer Vision Library*) adalah pustaka *open-source* yang dirancang untuk pengolahan citra dan visi komputer secara *real-time*. Dikembangkan pertama kali oleh Intel dan sekarang dikelola oleh komunitas terbuka, OpenCV menyediakan berbagai fungsi dan algoritma canggih untuk tugas-tugas seperti deteksi wajah, pelacakan objek, segmentasi gambar, transformasi geometris, serta pengolahan video[15]. Pustaka ini mendukung berbagai bahasa pemrograman seperti Python, C++, dan Java, dan dapat dijalankan pada berbagai platform termasuk Windows, Linux, macOS, dan Android. OpenCV sering digunakan dalam aplikasi berbasis kecerdasan buatan, robotika, serta sistem pengawasan visual karena kemampuannya dalam mengolah data visual secara efisien dan fleksibel.

2.17. *Google Colaboratory*

Google Colaboratory atau *Google Colab* adalah layanan berbasis *cloud* yang disediakan secara gratis oleh Google untuk menjalankan kode *Python* secara interaktif menggunakan *Jupyter Notebook*[17]. *Colab* memungkinkan pengguna untuk menulis, menjalankan, dan membagikan kode dengan mudah tanpa perlu menginstal perangkat lunak di komputer lokal. Salah satu keunggulan utamanya adalah dukungan terhadap GPU dan TPU, yang sangat berguna untuk pelatihan model *machine learning* dan *deep learning* dengan kecepatan lebih tinggi. Selain itu, *Colab* terintegrasi langsung dengan *Google Drive*, sehingga memudahkan penyimpanan dan pengelolaan file proyek. Fitur-fitur seperti kolaborasi waktu nyata, visualisasi data, serta kemampuan menjalankan pustaka populer seperti *TensorFlow*, *PyTorch*, dan *OpenCV* menjadikan *Google Colab* sebagai alat

yang sangat populer di kalangan peneliti, mahasiswa, dan praktisi kecerdasan buatan[17].

2.18. *GitHub*

GitHub adalah platform berbasis web yang digunakan untuk mengelola dan berbagi kode sumber menggunakan sistem kontrol versi *Git*. GitHub memungkinkan pengembang untuk berkolaborasi dalam proyek perangkat lunak, melacak perubahan kode, dan mengelola versi secara efisien. Melalui fitur seperti *repository*, *commit*, *pull request*, dan *issue tracking*, GitHub mempermudah proses pengembangan perangkat lunak secara tim maupun individu[16]. Selain itu, *GitHub* juga mendukung dokumentasi proyek melalui file README dan menyediakan layanan *GitHub Pages* untuk hosting halaman web statis. Dengan komunitas global yang besar, *GitHub* telah menjadi pusat distribusi terbuka bagi pustaka, skrip, dan proyek *open-source* di berbagai bidang seperti *data science, machine learning,* pengembangan web, hingga otomatisasi sistem.

2.19. *Predictions*

Predictions dalam konteks deteksi objek menggunakan model seperti YOLO mengacu pada hasil inferensi model terhadap data gambar baru, di mana sistem secara otomatis mengenali dan mengklasifikasikan objek berdasarkan pelatihan sebelumnya. Proses prediksi ini menghasilkan *bounding box* yang menunjukkan lokasi objek dalam gambar serta label kelas yang mengidentifikasi kategori objek, seperti daun sehat (*healthy*) atau tidak sehat (*unhealthy*), beserta skor kepercayaan (*confidence score*) yang menunjukkan tingkat keyakinan model terhadap prediksi tersebut[18].

2.19.1. *Healthy*

Prediksi daun sehat ditandai dengan kondisi daun yang utuh, berwarna hijau merata, dan tidak menunjukkan gejala kerusakan seperti bercak, lubang, atau perubahan warna. Model yang terlatih dengan baik akan mampu mendeteksi dan mengklasifikasikan daun dalam kondisi

optimal ini sebagai kelas *healthy*, yang biasanya diasosiasikan dengan tanaman yang tumbuh secara normal dan tidak terserang penyakit[18].

Gambar 2. 21 Prediksi *Apple Healthy*

Gambar 2. 22 Prediksi *Banana Healthy*

Gambar 2. 23 Prediksi *Durian Healthy*

Gambar 2. 24 Prediksi *Mango Healthy*

Gambar 2. 25 Prediksi *Orange Healthy*

2.19.2. *Unhealthy*

Sebaliknya, prediksi *unhealthy* mengacu pada deteksi daun yang mengalami kerusakan atau menunjukkan tanda-tanda penyakit, seperti bercak coklat, tepian mengering, perubahan warna tidak merata, atau adanya lubang akibat hama. Model akan mengklasifikasikan objek ini ke dalam kelas *unhealthy* berdasarkan fitur visual yang telah dipelajari selama proses pelatihan, sehingga dapat membantu dalam pemantauan kesehatan tanaman secara otomatis.

Gambar 2. 26 Prediksi *Apple Unhealthy*

Gambar 2. 27 Prediksi *Banana Unhealthy*

Gambar 2. 28 Prediksi *Durian Unhealthy*

Gambar 2. 29 Prediksi *Mango Unhealthy*

Gambar 2. 30 Prediksi *Orange Unhealthy*

2.20. Results

Dalam konteks pelatihan model *deep learning*, khususnya untuk tugas deteksi objek, *results* merujuk pada hasil evaluasi model yang mencakup metrik kinerja dan nilai *loss* (kerugian) selama proses pelatihan dan validasi. *Results* dihasilkan dari pencatatan otomatis oleh *framework* pelatihan, seperti YOLO atau *PyTorch*, yang mencatat performa model pada setiap *epoch*. Cara kerjanya dimulai dari proses pelatihan model menggunakan dataset, di mana pada setiap iterasi model melakukan prediksi dan hasilnya dibandingkan dengan label *ground truth*[19]. Selisih antara prediksi dan *ground truth* dihitung sebagai *loss*, yang digunakan untuk memperbarui bobot model melalui proses *backpropagation*. Selain itu, pada setiap *epoch*, model juga dievaluasi dengan menghitung metrik seperti *precision*, *recall*, dan *mean Average Precision* (mAP) menggunakan data validasi untuk mengetahui sejauh mana kemampuan model dalam mendeteksi dan mengklasifikasi objek dengan benar. Hasil-hasil ini direkam dalam bentuk grafik dan log untuk dianalisis guna memahami apakah model mengalami peningkatan kinerja atau justru *overfitting* atau *underfitting*[19].

Hasil pelatihan model menunjukkan tren konvergensi yang cukup stabil pada sebagian besar metrik dan fungsi kehilangan (*loss function*). Nilai *train loss* termasuk *train/box_loss*, *train/cls_loss*, dan *train/dfl_loss* mengalami penurunan konsisten selama proses pelatihan, yang mengindikasikan bahwa model semakin baik dalam mengenali dan memetakan pola dalam data pelatihan[19]. Di sisi lain, nilai *validation loss* (*val/box_loss, val/cls_loss*, dan *val/dfl_loss*) juga menunjukkan pola penurunan, meskipun terdapat fluktuasi, yang wajar terjadi pada data validasi. Metrik evaluasi seperti *precision*, *recall*, dan *mean Average Precision* (mAP) baik pada IoU 50 maupun 50-95 menunjukkan adanya peningkatan performa model, meskipun tampak ketidakstabilan pada beberapa bagian *epoch*, terutama pada metrik *precision* dan *mAP50-95*, yang cenderung fluktuatif[20].

Grafik baris atas menunjukkan kerugian (*loss*) pada data pelatihan, serta metrik *precision* dan *recall*, sedangkan baris bawah menampilkan kerugian pada data validasi dan metrik mAP. Secara umum, ketiga jenis *train loss* menunjukkan penurunan progresif, namun terdapat lonjakan tajam di akhir pelatihan yang kemungkinan besar disebabkan oleh *learning rate scheduling* atau *overfitting*[20]. Grafik *precision(B)* sangat fluktuatif, yang bisa jadi menunjukkan ketidakkonsistenan model dalam mengidentifikasi objek tertentu. Nilai *recall(B)* stabil di kisaran 0.4–0.5, namun tidak mengalami peningkatan signifikan. Pada metrik *mAP*, baik mAP50 maupun mAP50-95 tampak meningkat di awal pelatihan dan kemudian mulai stagnan atau bahkan menurun sedikit, menunjukkan bahwa performa deteksi model cenderung plateau setelah mencapai titik tertentu.

2.20.1 *Classification Loss (cls_loss)*

Digunakan untuk mengukur seberapa akurat model dalam mengklasifikasi objek ke dalam kelas yang benar.

$$L_{cls} = -\sum_{i=1}^{c} [y_i \, log \, log \, (y_i) + (1 - y_i) \, log \, log \, (1 - y_i) \qquad (2)$$

Keterangan :

a. C = jumlah kelas
b. y_i = label sebenarnya untuk kelas ke-i
c. $y\hat{}_I$ = probabilitas prediksi untuk kelas ke-i

Pada YOLOv8, biasanya digunakan *Binary Cross Entropy* (BCE) per kelas karena model mendukung *multi-label classification* (satu objek bisa memiliki lebih dari satu label dalam kasus tertentu)[20].

2.20.2 *Box Loss (box_loss)*

Digunakan untuk mengukur seberapa akurat prediksi *bounding box* dibandingkan dengan *ground truth*.

$$L_{box} = 1 - CIoU(b, b\ \ ) \quad (3)$$

Keterangan :

a. *b = ground truth bounding box*
b. *b^ = bounding box* hasil prediksi
c. *CIoU* menghitung kesamaan berdasarkan :
    1. IoU (*Intersection over Union)*
    2. Jarak pusat antar *box*
    3. Rasio aspek

CIoU adalah perbaikan dari IoU, GloU, dan DIoU karena mempertimbangkan lebih banyak factor dalam kesesuaian bentuk dan posisi *bounding box*[21].

2.20.3 *Distribution Focal Loss (dfl_loss)*

Digunakan untuk meningkatkan presisi regresi posisi *bounding box* dengan pendekatan distribusi diskrit. Dengan konsep utama yakni alih-alih memprediksi koordinat sebagai nilai tunggal, DFL memprediksi probabilitas pada beberapa *bins* dan menghitung ekspektasi. Dengan DFL, regresi *bounding box* menjadi lebih stabil dan akurat karena prediksi tidak lagi bergantung pada satu titik estimasi[21]. Rumus umum :

$$L_{dfl} = -\sum_{i=l}^{N} \quad CrossEntropy(p_i, t_i) \tag{4}$$

Keterangan :

a. N = jumlah sisi *bounding box* (biasanya 4: kiri, atas, kanan, bawah)
b. P_i = distribusi prediksi pada sisi ke-i
c. T_i = distribusi target pada sisi ke-i

2.21. *Confusion Matrix*

Confusion matrix adalah sebuah tabel yang digunakan untuk mengevaluasi kinerja model klasifikasi dengan membandingkan antara label aktual dan hasil prediksi. Matriks ini terdiri dari empat komponen utama yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). *True Positive* menunjukkan jumlah prediksi benar pada kelas positif, sementara *True Negative* menunjukkan jumlah prediksi benar pada kelas negatif[22]. Sebaliknya, *False Positive* menunjukkan jumlah prediksi salah ketika model memprediksi positif padahal seharusnya negatif, dan *False Negative* terjadi ketika model memprediksi negatif padahal seharusnya positif. Dari kombinasi nilai-nilai ini, berbagai metrik performa dapat dihitung untuk mengevaluasi akurasi dan keandalan model dalam berbagai skenario klasifikasi.

2.21.1. Presisi (*Precision*)

Presisi adalah metrik yang menunjukkan seberapa akurat model dalam memprediksi kelas positif, yaitu berapa banyak dari semua prediksi positif yang benar-benar relevan. Presisi tinggi berarti sedikit prediksi positif yang salah, yang sangat penting dalam kasus di mana konsekuensi dari false positive tinggi, seperti dalam diagnosis penyakit serius atau sistem keamanan. Rumus presisi adalah:

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

2.21.2. *Recall*

Recall, juga dikenal sebagai *sensitivitas* atau *true positive rate*, mengukur seberapa baik model dalam menemukan semua kasus positif yang sebenarnya ada. *Recall* tinggi penting dalam konteks dimana melewatkan data positif (*false negative*) sangat merugikan, seperti dalam deteksi kanker atau penyaringan keamanan bandara. Rumus *recall* adalah:

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

2.21.3. *F1-Score*

F1-*Score* adalah rata-rata harmonis antara presisi dan *recall*. Metrik ini digunakan ketika kita ingin menyeimbangkan antara presisi dan *recall*, terutama dalam situasi di mana data tidak seimbang. Nilai F1-*Score* berkisar antara 0 hingga 1, dengan 1 menunjukkan performa terbaik. F1-*Score* sangat berguna untuk mendapatkan gambaran umum performa model ketika tidak cukup hanya melihat presisi atau *recall* saja. Rumus F1-*Score* adalah:

$$F1\ score = 2\ x\ \frac{Precision\ x\ Recall}{Precision + Recall} \tag{7}$$

2.21.4. mAP (*Mean Average Precision*)

Mean Average Precision (mAP) adalah metrik yang umum digunakan dalam evaluasi model deteksi objek dan klasifikasi multi-label. mAP mengukur rata-rata dari nilai *Average Precision* (AP) untuk setiap kelas. AP sendiri adalah area di bawah kurva *Precision-Recall* untuk sebuah kelas tertentu. Rumus umum untuk mAP adalah:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} \quad AP_i \tag{8}$$

Di mana N adalah jumlah kelas, dan APi adalah *Average Precision* untuk kelas ke-i. Semakin tinggi nilai mAP, semakin baik model dalam melakukan deteksi atau klasifikasi terhadap semua kelas secara konsisten. mAP sangat penting dalam bidang *computer vision*, seperti dalam kompetisi COCO dan *PASCAL* VOC, karena mencerminkan performa model pada tingkat detil dan keseluruhan.

2.22. Analisa Performa Algoritma

Analisis performa algoritma CNN dengan arsitektur YOLOv8 dalam mendeteksi penyakit daun pada tanaman hortikultura tropis bertujuan untuk mengevaluasi sejauh mana model ini mampu mengidentifikasi dan mengklasifikasikan jenis-jenis penyakit secara akurat dan efisien. YOLOv8, sebagai salah satu arsitektur terbaru dari keluarga *You Only Look Once*, menawarkan keunggulan dalam hal kecepatan dan akurasi deteksi berkat desain modular yang dioptimalkan serta integrasi fitur-fitur lanjutan seperti *anchorfree detection* dan mekanisme adaptif[22]. Dalam konteks tanaman hortikultura tropis yang memiliki beragam jenis daun dan gejala penyakit yang kompleks, penggunaan YOLOv8 memungkinkan pemrosesan citra secara *real-time* dengan tetap mempertahankan ketelitian dalam mengenali pola visual yang halus, seperti bercak, perubahan warna, atau deformasi daun akibat infeksi jamur, bakteri, atau virus.

Evaluasi performa dilakukan dengan menggunakan metrik seperti presisi, *recall*, F1-*Score*, dan *mean Average Precision* (mAP) berdasarkan *confusion matrix*. Hasil analisis menunjukkan bahwa YOLOv8 mampu mencapai nilai mAP yang tinggi pada sebagian besar kelas penyakit, menandakan bahwa model mampu mendeteksi dan mengklasifikasikan objek secara konsisten pada berbagai kondisi pencahayaan dan latar belakang. Selain itu, nilai F1-*Score* yang seimbang menunjukkan bahwa model tidak hanya akurat dalam memprediksi kasus positif, tetapi juga cukup andal dalam mengenali semua kasus positif yang ada. Dengan demikian, algoritma CNN berbasis YOLOv8 terbukti efektif sebagai solusi cerdas dalam sistem pemantauan kesehatan tanaman hortikultura tropis,

yang sangat dibutuhkan untuk mendukung produktivitas pertanian berkelanjutan di wilayah beriklim tropis.

2.23. Curva (*Curve* Evaluasi)

Kurva evaluasi adalah representasi grafis yang digunakan untuk menilai performa model klasifikasi, terutama dalam memahami *trade-off* antara berbagai metrik seperti presisi, *recall*, dan *threshold* klasifikasi. Beberapa jenis kurva yang umum digunakan adalah *Precision-Recall Curve* dan *Receiver Operating Characteristic* (ROC) *Curve* [13]. *Precision-Recall Curve* menggambarkan hubungan antara presisi dan *recall* pada berbagai nilai *threshold*, dan sangat berguna saat menangani dataset yang tidak seimbang, karena lebih fokus pada performa terhadap kelas positif. Sementara itu, ROC *Curve* menunjukkan hubungan antara *True Positive Rate* (*Recall*) dan *False Positive Rate*, memberikan gambaran umum tentang kemampuan model dalam membedakan antara kelas positif dan negatif. Area di bawah kurva (AUC) dari masing-masing grafik memberikan ukuran kuantitatif terhadap kinerja model; semakin mendekati 1, maka semakin baik model dalam melakukan klasifikasi[23]. Kurva-kurva ini penting dalam proses evaluasi karena dapat membantu menentukan *threshold* optimal serta mengidentifikasi apakah model mengalami *overfitting* atau *underfitting* [24].

2.24. Perbandingan dengan Studi Sebelumnya

Penelitian ini membandingkan penerapan arsitektur YOLOv8 untuk deteksi penyakit daun tanaman hortikultura tropis dengan beberapa studi sebelumnya. Chitraningrum, N., Banowati, L., Herdiana, D., Mulyati, B., Sakti, I., Fudholi, A., Saputra, H., Farishi, S., Muchtar, C., & Andria, A. (2023) membandingkan YOLOv5 dan YOLOv8 untuk deteksi penyakit pada daun jagung dan menemukan bahwa YOLOv8 memiliki akurasi lebih tinggi, tetapi penelitian ini hanya terbatas pada jagung dan tidak mencakup tanaman tropis lainnya[25]. Berbeda dengan penelitian ini yang berfokus pada berbagai jenis tanaman tropis dengan dataset yang lebih beragam.

Selain itu, penelitian oleh Abid, M. S. Z., Jahan, B., Al Mamun, A., Hossen, M. J., & Mazumder, S. H. (2024) mengembangkan sistem deteksi penyakit daun menggunakan CNN dan menunjukkan akurasi tinggi sebesar 96,5%. Namun, penelitian tersebut masih bergantung pada dataset yang tersedia dan belum mengoptimalkan deteksi objek secara langsung seperti YOLOv8[21]. Penelitian ini mengatasi keterbatasan tersebut dengan menggunakan anotasi dataset yang lebih presisi untuk mendeteksi penyakit secara langsung dalam satu tahap pemrosesan. Studi lainnya oleh He, Y., Peng, Y., Wei, C., Zheng, Y., Yang, C., & Zou, T. (2024) mengembangkan KTD-YOLOv8 untuk mendeteksi penyakit pada daun stroberi dengan peningkatan akurasi dan efisiensi. Namun, model tersebut masih menghadapi keterbatasan dalam kualitas dataset dan kondisi dunia nyata[20]. Penelitian ini mengatasi masalah tersebut dengan mengevaluasi model dalam berbagai kondisi pencahayaan dan latar belakang untuk meningkatkan generalisasi deteksi penyakit pada tanaman tropis.