# BAB 2
# LANDASAN TEORI

## 2.1 Tanaman Melon dan Penyakit Powdery Mildew

Tanaman melon merupakan salah satu komoditas hortikultura yang banyak dibudidayakan karena memiliki nilai ekonomi yang tinggi. Meskipun demikian, produktivitas tanaman melon dapat dipengaruhi oleh berbagai faktor, salah satunya adalah serangan penyakit. Oleh karena itu, pada bagian ini dibahas karakteristik tanaman melon serta penyakit *powdery mildew* sebagai landasan dalam pengembangan sistem segmentasi dan estimasi tingkat keparahan penyakit.

### 2.1.1 Tanaman Melon

Tanaman melon (*Cucumis melo* L.) merupakan tanaman hortikultura yang termasuk dalam famili *Cucurbitaceae* dan banyak dibudidayakan di wilayah tropis maupun subtropis, termasuk Indonesia. Buah melon memiliki nilai ekonomi yang tinggi karena digemari masyarakat berkat rasanya yang manis, kandungan air yang tinggi, serta kandungan gizi seperti vitamin C, vitamin E, serat, dan berbagai senyawa antioksidan yang bermanfaat bagi kesehatan [1]. Selain dikonsumsi secara langsung, melon juga dimanfaatkan sebagai bahan baku berbagai produk pangan dan minuman sehingga permintaan terhadap komoditas ini terus meningkat.

Budidaya melon dapat dilakukan di lahan terbuka maupun di dalam *greenhouse*. Penggunaan *greenhouse* memungkinkan pengendalian faktor lingkungan, seperti suhu, kelembapan, intensitas cahaya, dan sirkulasi udara, sehingga pertumbuhan tanaman dan kualitas buah dapat lebih terjaga. Namun, kondisi lingkungan yang lembap di dalam *greenhouse* juga dapat meningkatkan risiko perkembangan penyakit yang disebabkan oleh jamur, salah satunya adalah *powdery mildew*.

### 2.1.2 Penyakit Powdery Mildew

*Powdery mildew* merupakan penyakit *fungal* yang umum menyerang tanaman melon, disebabkan oleh jamur *Podosphaera xanthii*. Infeksi penyakit ini bermanifestasi pada bagian daun tanaman, yang ditandai dengan terbentuknya lapisan berwarna putih menyerupai serbuk tepung pada permukaan daun [3]. Infeksi *powdery mildew* terjadi melalui penyebaran konidia yang menempel pada permukaan daun. Jamur kemudian membentuk appressoria dan haustoria untuk menembus jaringan epidermis daun dan menyerap nutrisi dari tanaman. Setelah proses penetrasi berlangsung, jamur membentuk

hifa epifitik yang berkembang pada permukaan daun sehingga menyebabkan penyebaran infeksi secara cepat.

Pada tahap awal infeksi, gejala *powdery mildew* diawali dengan kemunculan bercak putih berukuran kecil yang dapat muncul pada kedua sisi permukaan daun. Apabila tidak ditangani, infeksi akan terus berkembang sehingga bercak meluas dan berpotensi menutupi hampir seluruh permukaan daun. Kondisi ini menyebabkan terganggunya proses fotosintesis karena berkurangnya luas daun efektif yang dapat menyerap cahaya matahari. Selain itu, infeksi *powdery mildew* juga dapat menurunkan efisiensi penggunaan air, mempercepat proses nekrosis daun, serta berdampak negatif terhadap kualitas dan kuantitas hasil panen melon [8].

Pada lingkungan budidaya *greenhouse*, perkembangan *powdery mildew* dapat berlangsung lebih cepat akibat tingginya kelembapan dan sirkulasi udara yang kurang optimal. Penyebaran penyakit yang terjadi secara masif dapat menyebabkan penurunan produktivitas tanaman dalam waktu singkat apabila tidak segera ditangani. Oleh karena itu, deteksi dini dan estimasi tingkat keparahan penyakit menjadi sangat penting untuk mendukung pengambilan keputusan pengendalian penyakit secara tepat dan efisien [6].

### 2.1.3 Tingkat Keparahan Penyakit

Tingkat keparahan penyakit (*disease severity*) merupakan ukuran proporsi area jaringan tanaman yang menunjukkan gejala penyakit dibandingkan dengan total area organ tanaman yang diamati [15]. Pada penyakit tanaman berbasis daun, *severity* umumnya dinyatakan dalam bentuk persentase luas area daun yang terinfeksi. Informasi ini sangat penting dalam bidang fitopatologi karena digunakan untuk memantau perkembangan penyakit, mengevaluasi efektivitas pengendalian, serta mendukung pengambilan keputusan dalam penggunaan fungisida dan manajemen budidaya tanaman.

Secara konvensional, estimasi *severity* dilakukan melalui pengamatan visual oleh tenaga ahli menggunakan skala tertentu, seperti skala persentase atau *Horsfall–Barratt scale* [16]. Metode visual relatif mudah diterapkan, namun memiliki keterbatasan berupa subjektivitas penilaian, inkonsistensi antar pengamat (*inter-rater variability*), serta potensi kesalahan estimasi pada tingkat infeksi rendah maupun bercak berukuran kecil [15].

Pada penelitian ini, estimasi *severity* dilakukan secara otomatis menggunakan pendekatan segmentasi semantik berbasis *deep learning*. Persentase infeksi dihitung menggunakan rumus:

$$\text{Severity } (\%) = \frac{A_{\text{bercak}}}{A_{\text{daun}}} \times 100\% \tag{2.1}$$

di mana $A_{\text{bercak}}$ adalah jumlah piksel area terinfeksi hasil segmentasi semantik dan $A_{\text{daun}}$ adalah jumlah piksel area daun hasil *background removal*. Berdasarkan nilai persentase yang diperoleh, tingkat keparahan diklasifikasikan menjadi beberapa kategori sebagai

berikut[17]:

Tabel 2.1. Klasifikasi tingkat keparahan (*severity*) *Powdery Mildew*

| Skala | Persentase Infeksi | Keterangan |
|---|---|---|
| 0 | 0% | Tidak terdapat gejala infeksi. |
| 1 | 0.1% – 10% | Infeksi sangat ringan, bercak terbatas pada sebagian kecil area daun. |
| 2 | 10.1% – 25% | Infeksi ringan dengan area bercak mulai meluas pada permukaan daun. |
| 3 | 25.1% – 50% | Infeksi sedang dengan area bercak menutupi sebagian besar permukaan daun. |
| 4 | 50.1% – 75% | Infeksi berat dengan bercak padat dan menyebar luas pada daun. |
| 5 | $>$ 75% | Infeksi sangat berat dengan sebagian besar area daun tertutup bercak. |

## 2.2 Pengolahan Citra Digital

Pengolahan citra digital menjadi dasar dalam pengembangan sistem segmentasi pada penelitian ini. Berbagai konsep yang berkaitan dengan representasi citra dan proses pengolahan data diperlukan agar informasi visual pada citra dapat dimanfaatkan secara optimal oleh model deep learning. Oleh karena itu, bagian ini menjelaskan konsep dasar citra digital sebagai landasan penelitian.

### 2.2.1 Citra Digital

Citra digital merupakan bentuk representasi visual suatu objek yang dinyatakan dalam struktur matriks dua dimensi yang tersusun atas elemen-elemen diskret yang disebut piksel (picture element). Setiap piksel menyimpan nilai intensitas cahaya atau informasi warna pada posisi koordinat tertentu, sehingga citra dapat diinterpretasikan dan dianalisis secara komputasional [18]. Pada citra berwarna, representasi warna setiap piksel menggunakan model warna RGB (Red, Green, Blue) yang terdiri atas tiga kanal warna secara terpisah. Secara matematis, citra digital dapat dimodelkan sebagai fungsi dua dimensi f(x,y)f(x,y) f(x,y) yang merepresentasikan nilai intensitas warna pada koordinat piksel (x,y)(x, y) (x,y) Citra digital merupakan bentuk representasi visual suatu objek yang dinyatakan dalam struktur matriks dua dimensi yang tersusun atas elemen-elemen diskret yang disebut piksel (*picture element*). Setiap piksel menyimpan nilai intensitas cahaya atau informasi warna pada posisi koordinat tertentu, sehingga citra dapat diinterpretasikan dan dianalisis secara komputasional [18]. Pada citra berwarna, representasi warna setiap piksel

menggunakan model warna RGB (*Red, Green, Blue*) yang terdiri atas tiga kanal warna secara terpisah. Secara matematis, citra digital dapat dinyatakan sebagai fungsi dua dimensi $f(x,y)$ yang merepresentasikan intensitas warna pada koordinat piksel $(x,y)$:

$$I(x,y) = R(x,y),;G(x,y),;B(x,y) \tag{2.2}$$

dengan $R$, $G$, dan $B$ masing-masing memiliki rentang nilai integer $[0,255]$. Pada penelitian ini, citra daun melon direpresentasikan sebagai tensor tiga dimensi berukuran $512 \times 512 \times 3$ setelah melalui proses *resize*. Dimensi ini dipilih sebagai keseimbangan antara tingkat detail fitur visual yang cukup untuk mendeteksi bercak kecil dan efisiensi komputasi yang dapat dijalankan pada GPU dengan VRAM 16 GB.

### 2.2.2 Citra Tanaman pada Lingkungan Greenhouse

Citra tanaman yang diperoleh pada lingkungan nyata memiliki kompleksitas visual yang lebih tinggi dibandingkan citra laboratorium terkontrol. Variasi pencahayaan, bayangan, sudut pengambilan gambar, tekstur daun, serta keberadaan objek lain pada latar belakang dapat memengaruhi kualitas citra dan performa model segmentasi [19]. Pada lingkungan *greenhouse*, perubahan intensitas cahaya dan kelembapan dapat menyebabkan distribusi warna pada daun menjadi tidak konsisten. Keberadaan tali tanaman, struktur *greenhouse*, daun yang saling bertumpuk, dan pantulan cahaya juga dapat menjadi sumber gangguan dalam proses segmentasi area penyakit.

Menurut Barbedo et al. salah satu tantangan utama dalam penerapan *deep learning* untuk deteksi penyakit tanaman adalah rendahnya kemampuan generalisasi model ketika dihadapkan pada citra lapangan nyata yang memiliki variasi lingkungan tinggi. Oleh karena itu, penggunaan citra nyata dari lingkungan *greenhouse* pada penelitian ini diharapkan mampu meningkatkan kemampuan model dalam melakukan segmentasi penyakit pada kondisi budidaya sebenarnya.

### 2.2.3 Pre-processing Citra

*Pre-processing* citra merupakan peningkatan kualitas citra dan penyeragaman data input dicapai melalui tahapan *pre-processing*, yang merupakan langkah prasyarat sebelum data dieksekusi oleh arsitektur *deep learning* [20].. Pada penelitian ini, tahapan *pre-processing* meliputi dua proses utama.

1. *Resize*: Seluruh citra diseragamkan menjadi resolusi $512 \times 512$ piksel. Interpolasi bilinear digunakan untuk citra RGB, sedangkan interpolasi *nearest-neighbor*

digunakan untuk mask anotasi guna mempertahankan nilai label biner tanpa menghasilkan nilai interpolasi yang ambigu.

2. Normalisasi: Nilai piksel dinormalisasi sesuai dengan kebutuhan masing-masing arsitektur model. Untuk U-Net *scratch*, normalisasi dilakukan dengan membagi nilai piksel dengan 255 sehingga diperoleh rentang $[0,1]$:

$$I_{\text{norm}}(x,y) = \frac{I(x,y)}{255} \tag{2.3}$$

Untuk encoder MobileNetV2 *pretrained*, normalisasi menggunakan fungsi `preprocess_input` dari Keras yang menyesuaikan distribusi nilai piksel terhadap statistik dataset ImageNet, menghasilkan rentang $[-1, 1]$ sesuai dengan distribusi yang digunakan saat pelatihan backbone.

### 2.2.4 Augmentasi Data

Augmentasi data adalah metode untuk meningkatkan keragaman data pelatihan secara buatan melalui berbagai transformasi, baik geometris maupun fotometrik, tanpa mengubah makna atau label semantik pada citra tersebut [21]. Teknik ini menjadi penting pada penelitian dengan ketersediaan data yang terbatas, karena model yang dilatih menggunakan data yang lebih bervariasi umumnya memiliki kemampuan generalisasi yang lebih baik terhadap data baru serta lebih robust terhadap masalah *overfitting*.

Pada penelitian ini, augmentasi diimplementasikan menggunakan *library* Albumentations dengan pipeline `A.Compose` yang secara otomatis menjamin sinkronisasi transformasi geometrik antara citra dan *mask* anotasi secara bersamaan. Transformasi yang diterapkan meliputi dua kategori berikut.

- Transformasi Geometrik (diterapkan secara sinkron pada citra dan *mask*): rotasi acak hingga 180°, *horizontal* dan *vertical flip*, *shift-scale-rotate*, *elastic transform*, dan *grid distortion*. Transformasi ini mensimulasikan variasi orientasi dan posisi daun yang mungkin ditemukan pada kondisi pengambilan gambar nyata.
- Transformasi Fotometrik (diterapkan hanya pada citra, *mask* tidak berubah): *random brightness contrast*, *color jitter*, CLAHE (*Contrast Limited Adaptive Histogram Equalization*), *Gaussian noise*, *Gaussian blur*, dan *CoarseDropout* sebagai simulasi oklusi parsial. Transformasi ini mensimulasikan variasi kondisi pencahayaan dan kualitas citra yang dapat terjadi pada lingkungan *greenhouse*.

## 2.3 Segmentasi Citra

Segmentasi citra merupakan salah satu tahapan penting dalam pengolahan citra digital yang bertujuan untuk memisahkan objek yang menjadi perhatian dari bagian citra lainnya. Proses ini menjadi dasar bagi berbagai tugas analisis citra karena memungkinkan sistem mengenali lokasi, bentuk, dan batas objek secara lebih akurat. Pada penelitian ini, segmentasi citra digunakan untuk mengidentifikasi area daun melon dan area infeksi *powdery mildew*. Oleh karena itu, pada bagian ini dibahas konsep dasar segmentasi citra serta segmentasi semantik yang menjadi landasan dalam penelitian.

### 2.3.1 Konsep Dasar Segmentasi Citra

Segmentasi citra merupakan proses pemisahan citra digital menjadi beberapa wilayah (*region*) berdasarkan karakteristik tertentu seperti warna, intensitas, tekstur, atau pola spasial, sehingga objek yang penting dapat dipisahkan dan dianalisis secara individual [22]. Proses ini termasuk salah satu komponen utama dalam bidang *computer vision* karena membantu sistem dalam memahami struktur spasial dari objek dalam citra. Teknik segmentasi ini banyak diterapkan pada berbagai bidang pengolahan citra, termasuk analisis medis, pengenalan objek, sistem kendaraan otonom, hingga deteksi penyakit pada tanaman.

### 2.3.2 Segmentasi Semantik

Segmentasi semantik (*semantic segmentation*) adalah salah satu tugas dalam *computer vision* yang berfokus pada pemberian label kelas pada setiap piksel dalam citra secara menyeluruh (*dense prediction*) [22]. Berbeda dengan deteksi objek yang hanya menghasilkan kotak pembatas (*bounding box*), metode ini menghasilkan peta segmentasi beresolusi penuh sehingga bentuk dan batas objek dapat diidentifikasi dengan lebih presisi hingga tingkat piksel.

Dalam penelitian ini, segmentasi semantik digunakan untuk mengidentifikasi area piksel yang termasuk kategori *powdery mildew* dan area daun sehat. Pendekatan ini dipilih karena bentuk bercak penyakit bersifat tidak beraturan dan memiliki variasi ukuran yang tinggi, sehingga sulit direpresentasikan secara akurat menggunakan *bounding box* berbentuk persegi panjang. Selain itu, proses estimasi tingkat keparahan penyakit memerlukan pengukuran luas area infeksi secara presisi, sehingga penentuan batas bercak pada level piksel menjadi sangat penting.

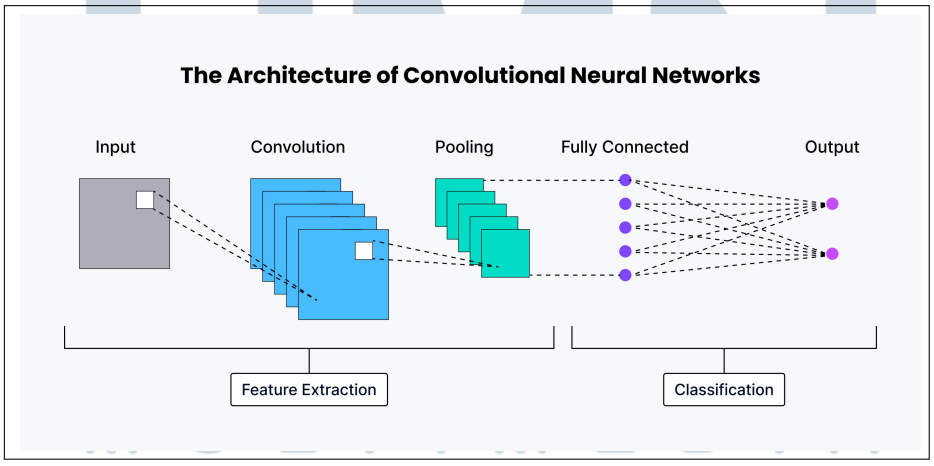
Sebagian besar arsitektur segmentasi semantik modern mengadopsi struktur *encoder-decoder*. Encoder bertugas mengekstraksi representasi fitur hierarkis dari citra masukan dengan resolusi spasial yang semakin berkurang (*downsampling*), sementara decoder bertugas merekonstruksi peta segmentasi dengan resolusi penuh melalui operasi

*upsampling*. Mekanisme *skip connection* yang menghubungkan langsung *feature map* dari encoder ke decoder pada resolusi yang bersesuaian mengatasi masalah hilangnya informasi spasial selama *downsampling*, memungkinkan model menggabungkan informasi kontekstual skala besar dengan detail spasial skala kecil.

## 2.4 Convolutional Neural Network (CNN)

*Convolutional Neural Network* (CNN) adalah salah satu arsitektur dalam *deep learning* yang umum digunakan pada pengolahan citra karena kemampuannya dalam mengekstraksi fitur visual secara otomatis dari data berbentuk grid seperti gambar digital [23]. Pada citra, informasi tersusun dari piksel yang memiliki keterkaitan spasial, sehingga CNN dirancang untuk mempelajari pola lokal secara bertahap, mulai dari fitur sederhana seperti tepi dan warna hingga pola yang lebih kompleks seperti tekstur, bentuk, dan struktur objek. Kemampuan ini sangat sesuai untuk penelitian ini, karena area daun dan gejala *powdery mildew* memiliki karakteristik visual yang berbeda dalam hal warna, tekstur, serta pola penyebarannya pada permukaan daun.

Komponen utama CNN terdiri atas lapisan konvolusi (*convolution layer*), fungsi aktivasi, dan lapisan *pooling*. Lapisan konvolusi berfungsi mengekstraksi pola lokal dari citra menggunakan kernel atau filter yang digeser pada area citra, sedangkan lapisan *pooling* digunakan untuk mereduksi ukuran spasial *feature map* dengan tetap mempertahankan informasi penting [23, 24]. Kombinasi lapisan tersebut memungkinkan CNN mempelajari representasi fitur secara hierarkis dari tingkat sederhana hingga kompleks.




Gambar 2.1. Ilustrasi dasar alur kerja *Convolutional Neural Network* (CNN) pada pengolahan citra.

Gambar 2.1 menunjukkan alur dasar CNN dalam melakukan ekstraksi fitur citra. Bagian awal CNN berfungsi sebagai tahap *feature extraction*, yaitu mengekstraksi fitur

visual dari citra masukan melalui proses konvolusi dan *pooling*. Fitur yang dihasilkan kemudian digunakan oleh model untuk melakukan tugas lanjutan seperti klasifikasi, deteksi objek, maupun segmentasi semantik. Jika citra masukan dinyatakan sebagai (I(x,y)) dan kernel konvolusi dinyatakan sebagai (K(i,j)), maka operasi konvolusi dapat dituliskan sebagai berikut:

$$F(x,y) = \sum_m \sum_n I(x+m, y+n), K(m,n) \tag{2.4}$$

Pada persamaan tersebut, (S(x,y)) merupakan nilai *feature map* pada koordinat ((x,y)), (I(x+i,y+j)) merupakan nilai piksel citra masukan, dan (K(i,j)) merupakan bobot kernel. Hasil operasi konvolusi berupa *feature map* yang menunjukkan respons model terhadap pola visual tertentu pada citra.

Setelah proses konvolusi, fungsi aktivasi digunakan untuk menambahkan sifat nonlinier ke dalam model. Salah satu fungsi aktivasi yang paling sering digunakan adalah *Rectified Linear Unit* (ReLU), yang dapat dinyatakan sebagai berikut:

$$f(z) = \max(0, z) \tag{2.5}$$

Fungsi ReLU mempertahankan nilai positif dan mengubah nilai negatif menjadi nol sehingga model dapat mempelajari hubungan nonlinier yang lebih kompleks secara efisien [24].

Setelah fungsi aktivasi, lapisan *pooling* digunakan untuk mereduksi dimensi spasial *feature map*. Salah satu metode yang paling umum digunakan adalah *max pooling*, yaitu memilih nilai maksimum pada suatu wilayah lokal:

$$P(x,y) = \max_{(i,j)\in R} S(x+i, y+j) \tag{2.6}$$

dengan (R) menyatakan wilayah lokal yang digunakan dalam proses *pooling*. Proses ini membantu mengurangi kebutuhan komputasi serta membuat representasi fitur lebih stabil terhadap perubahan posisi objek pada citra.

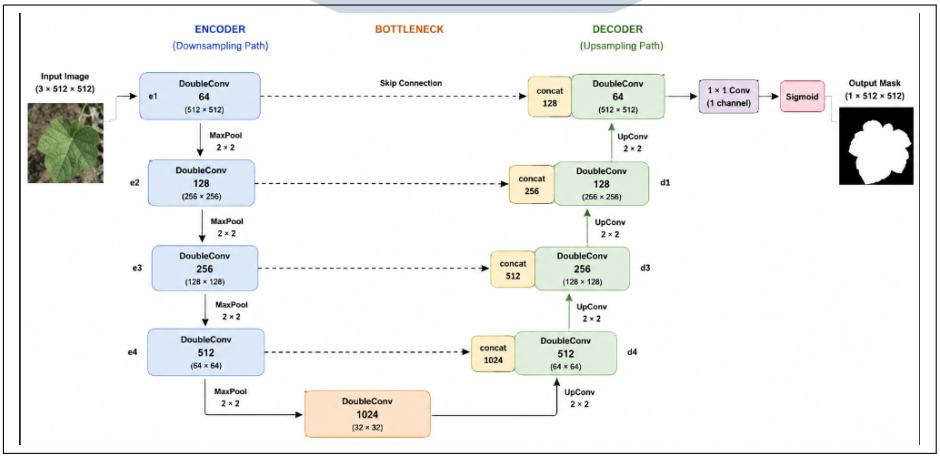
CNN mempelajari fitur visual secara hierarkis. Lapisan awal umumnya menangkap fitur sederhana seperti garis, sudut, dan tepi, sedangkan lapisan yang lebih dalam mampu menangkap fitur yang lebih kompleks seperti tekstur daun, pola bercak penyakit, dan struktur objek secara keseluruhan [23]. Kemampuan ini menjadi dasar bagi berbagai model segmentasi modern yang digunakan pada penelitian ini. Dalam penelitian ini, CNN menjadi fondasi utama bagi dua model yang digunakan, yaitu U-Net untuk segmentasi daun (*background removal*) dan U-Net dengan encoder MobileNetV2 untuk segmentasi area

*powdery mildew*. Kedua model tersebut memanfaatkan kemampuan CNN dalam melakukan ekstraksi fitur visual secara otomatis sehingga area daun dan area infeksi dapat diidentifikasi secara lebih akurat pada tingkat piksel.

## 2.5 U-Net

U-Net merupakan salah satu arsitektur *deep learning* yang banyak digunakan dalam tugas segmentasi semantik karena mampu menghasilkan segmentasi yang akurat meskipun hanya tersedia data pelatihan dalam jumlah terbatas. Arsitektur ini dirancang untuk menggabungkan informasi spasial dan fitur tingkat tinggi melalui mekanisme *skip connection*, sehingga sangat sesuai untuk melakukan segmentasi objek pada citra. Pada penelitian ini, U-Net digunakan pada tahap pertama untuk melakukan segmentasi daun dan *background removal* sebelum proses segmentasi area infeksi *powdery mildew* dilakukan [25].

Arsitektur U-Net yang digunakan dalam penelitian ini ditunjukkan pada Gambar 2.2. Arsitektur tersebut terdiri atas tiga bagian utama, yaitu *encoder* (*downsampling path*), *bottleneck*, dan *decoder* (*upsampling path*) yang saling terhubung melalui mekanisme *skip connection*.




Gambar 2.2. Ilustrasi arsitektur U-Net yang digunakan pada tahap segmentasi daun (*background removal*).

Berdasarkan Gambar 2.2, citra masukan berukuran $512 \times 512 \times 3$ terlebih dahulu diproses pada bagian *encoder*. Setiap blok *encoder* terdiri atas modul *DoubleConv*, yaitu dua lapisan konvolusi berukuran $3 \times 3$ yang masing-masing diikuti oleh *Batch Normalization* dan fungsi aktivasi ReLU. Setelah proses *DoubleConv*, diterapkan operasi *Max Pooling* berukuran $2 \times 2$ untuk mengurangi resolusi *feature map* menjadi setengahnya. Seiring bertambahnya kedalaman jaringan, jumlah filter meningkat secara bertahap dari 64, 128, 256, hingga 512 sehingga model mampu mengekstraksi fitur dengan tingkat

representasi yang semakin kompleks. Akibat proses tersebut, ukuran *feature map* berubah dari $512 \times 512$ menjadi $256 \times 256$, $128 \times 128$, $64 \times 64$, hingga $32 \times 32$.

Pada bagian terdalam jaringan terdapat *bottleneck*, yang berfungsi sebagai pusat ekstraksi fitur sebelum proses rekonstruksi dilakukan. Bagian ini menggunakan modul *DoubleConv* dengan 1024 filter tanpa melalui proses *Max Pooling*. Melalui representasi fitur pada tingkat abstraksi yang lebih tinggi, *bottleneck* mempertahankan informasi penting hasil proses *downsampling* sebelum diteruskan ke tahap berikutnya.

Selanjutnya, proses rekonstruksi dilakukan pada bagian *decoder*. Setiap blok *decoder* diawali dengan proses *upsampling* menggunakan *Transposed Convolution* berukuran $2 \times 2$ yang meningkatkan resolusi *feature map* menjadi dua kali lipat. Hasil *upsampling* kemudian digabungkan (*concatenate*) dengan *feature map* dari *encoder* yang memiliki resolusi sama melalui mekanisme *skip connection*. Setelah proses penggabungan, *feature map* kembali diproses menggunakan modul *DoubleConv* untuk menyempurnakan representasi fitur sekaligus mengurangi jumlah kanal secara bertahap dari 1024 menjadi 512, kemudian 256, 128, dan 64.

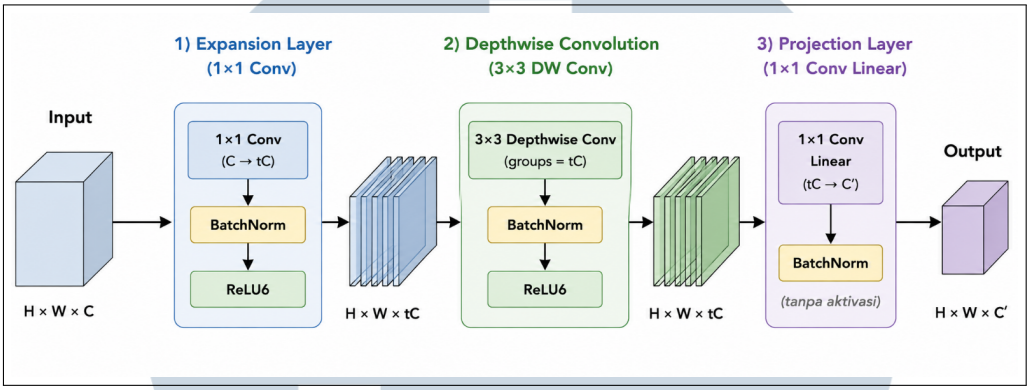
Mekanisme *skip connection* berperan penting dalam mempertahankan informasi spasial yang diperoleh selama proses *encoding*. Dengan menggabungkan *feature map* dari *encoder* ke *decoder*, informasi mengenai bentuk dan batas objek yang berpotensi hilang akibat proses *downsampling* dapat dipulihkan kembali. Oleh karena itu, hasil segmentasi mampu mempertahankan detail objek dengan lebih baik dibandingkan apabila hanya mengandalkan proses *upsampling*.Pada lapisan keluaran digunakan konvolusi berukuran $1 \times 1$ untuk memetakan setiap piksel menjadi satu kanal keluaran. Selanjutnya, fungsi aktivasi sigmoid menghasilkan nilai probabilitas setiap piksel termasuk ke dalam kelas daun atau latar belakang. Nilai probabilitas tersebut kemudian dikonversi menjadi *mask* segmentasi biner yang menjadi hasil akhir proses *background removal*.

## 2.6 MobileNetV2

MobileNetV2 adalah salah satu arsitektur CNN yang dirancang untuk menghasilkan ekstraksi fitur yang efisien dengan kebutuhan komputasi yang relatif rendah. Arsitektur ini menggabungkan *depthwise separable convolution*, *inverted residual block*, dan *linear bottleneck* untuk menekan jumlah parameter sekaligus mempertahankan kemampuan representasi fitur yang baik [10]. Dalam penelitian ini, MobileNetV2 digunakan sebagai *encoder* pada arsitektur U-Net untuk melakukan segmentasi area infeksi *powdery mildew*. Penggunaan bobot *pretrained* dari ImageNet diharapkan dapat meningkatkan kemampuan model dalam menangkap fitur visual pada daun dan area penyakit secara lebih efektif dibandingkan dengan pelatihan dari awal (*training from scratch*).

### 2.6.1 Arsitektur MobileNetV2

Arsitektur dasar MobileNetV2 yang digunakan pada penelitian ini ditunjukkan pada Gambar 2.3.




Gambar 2.3. Struktur *inverted residual block* pada MobileNetV2 yang menjadi komponen dasar setiap lapisan encoder pada arsitektur segmentasi yang diusulkan.

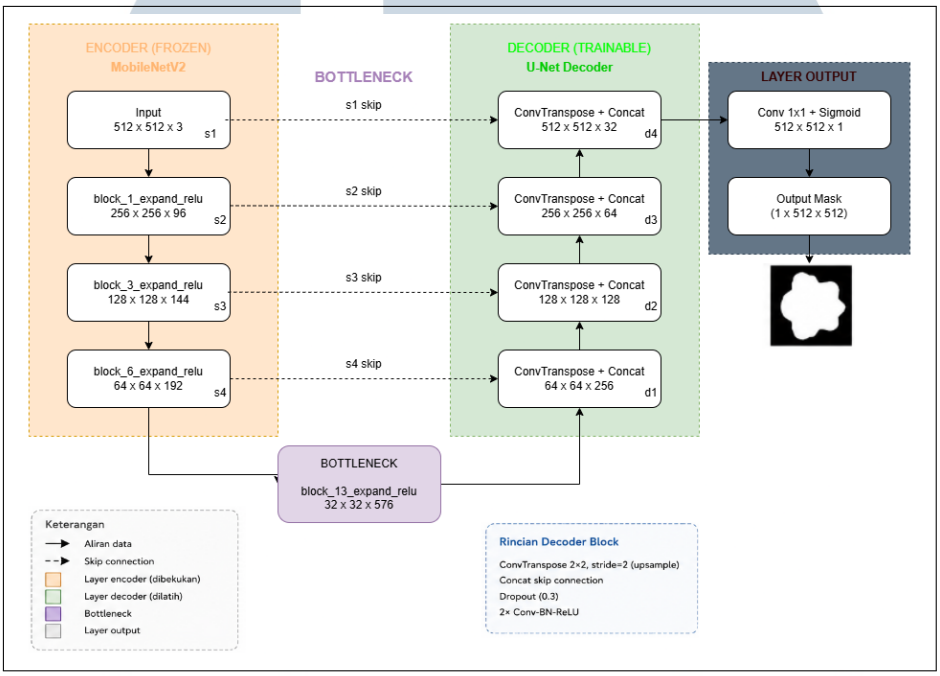
Berdasarkan Gambar 2.3, MobileNetV2 dibangun menggunakan blok utama yang disebut *inverted residual block*. Blok ini terdiri atas *expansion layer*, *depthwise convolution*, dan *projection layer*. Berbeda dengan konvolusi standar, *depthwise convolution* melakukan proses konvolusi secara terpisah pada setiap kanal fitur sehingga jumlah parameter dan kebutuhan komputasi dapat dikurangi secara signifikan. Selain itu, MobileNetV2 menerapkan konsep *linear bottleneck* yang memungkinkan informasi fitur tetap dipertahankan meskipun jumlah kanal pada lapisan keluaran diperkecil. Kombinasi *depthwise separable convolution*, *inverted residual block*, dan *linear bottleneck* menjadikan MobileNetV2 memiliki efisiensi komputasi yang tinggi dengan kemampuan ekstraksi fitur yang tetap baik.

Kombinasi *depthwise separable convolution*, *inverted residual block*, dan *linear bottleneck* menjadikan MobileNetV2 memiliki efisiensi komputasi yang tinggi dengan kemampuan ekstraksi fitur yang tetap baik. Karakteristik tersebut menjadikan MobileNetV2 sesuai untuk digunakan sebagai encoder pada arsitektur U-Net dalam penelitian ini, sebagaimana dijelaskan pada subbagian berikut.

### 2.6.2 U-Net dengan Encoder MobileNetV2 Pretrained

*Transfer learning* adalah pendekatan yang menggunakan kembali pengetahuan dari model yang telah dilatih pada dataset berukuran besar untuk diterapkan pada tugas lain yang masih memiliki keterkaitan [26]. Pada penelitian ini, MobileNetV2 yang telah dilatih pada dataset ImageNet digunakan sebagai encoder pada arsitektur U-Net untuk segmentasi

area *powdery mildew*. Setiap lapisan encoder terdiri atas rangkaian *inverted residual block* sebagaimana ditunjukkan pada Gambar 2.3, sehingga proses ekstraksi fitur berlangsung secara efisien. Encoder yang telah melalui proses *pretraining* ini sebelumnya sudah belajar berbagai fitur visual dasar, seperti tepi, tekstur, dan pola warna. Hal ini membuat proses pelatihan menjadi lebih stabil, sekaligus membantu mengurangi risiko *overfitting*, terutama ketika data yang tersedia dalam jumlah terbatas.




Gambar 2.4. Arsitektur U-Net dengan encoder MobileNetV2 untuk segmentasi area *powdery mildew*.

Arsitektur yang digunakan ditunjukkan pada Gambar 2.4. Encoder MobileNetV2 dioperasikan dalam kondisi *frozen* sehingga parameter encoder tidak diperbarui selama pelatihan. *Feature map* yang digunakan sebagai *skip connection* diambil dari empat titik pada arsitektur. $s_1$ diambil langsung dari citra masukan ($512\times512\times3$) sebelum memasuki *backbone*, sedangkan $s_2$, $s_3$, dan $s_4$ diambil dari lapisan *intermediate* MobileNetV2, yaitu `block_1_expand_relu` ($256\times256\times96$), `block_3_expand_relu` ($128\times128\times144$), dan `block_6_expand_relu` ($64\times64\times192$). Lapisan `block_13_expand_relu` ($32\times32\times576$) digunakan sebagai *bottleneck* yang menjadi titik masukan bagi bagian decoder.

Pada bagian decoder, proses *upsampling* dilakukan menggunakan *transposed convolution* yang secara matematis dapat dinyatakan sebagai:

$$\mathbf{F}up = \text{ConvTranspose}\left(\mathbf{F}enc\right) \tag{2.7}$$

di mana $\mathbf{F}_{in}$ adalah *feature map* masukan, $\mathbf{W}_{up}$ adalah bobot *transposed convolution*, dan $\mathbf{F}_{up}$ adalah *feature map* keluaran dengan resolusi yang lebih tinggi. Hasil *upsampling* kemudian digabungkan dengan *feature map encoder* melalui operasi *concatenation*:

$$\mathbf{F}dec = \mathrm{Conv}\big([\mathbf{F}up, |, \mathbf{F}_{skip}]\big) \tag{2.8}$$

di mana $\mathbf{F}_{skip}$ adalah *feature map* dari *skip connection* encoder, $[\cdot\|\cdot]$ menyatakan operasi *concatenation* pada dimensi kanal, dan $\mathrm{Conv}(\cdot)$ adalah blok konvolusi yang terdiri atas *ConvTranspose*, *dropout*, konvolusi 3×3, *batch normalization*, dan aktivasi ReLU. Proses dekoding dilakukan secara bertahap dari resolusi terendah menuju resolusi penuh. Dimulai dari *bottleneck* (32×32×576), decoder $d_1$ menghasilkan *feature map* berukuran 64×64×256 dengan menggabungkan hasil *upsampling* dan $s_4$. Proses ini dilanjutkan pada $d_2$ (128×128×128), $d_3$ (256×256×64), hingga $d_4$ (512×512×32) yang menggabungkan fitur dengan $s_1$ untuk memulihkan resolusi penuh citra masukan. Struktur ini memungkinkan model memanfaatkan fitur semantik encoder sekaligus mempertahankan detail spasial yang diperlukan untuk menentukan batas area infeksi secara akurat.

Lapisan keluaran menggunakan konvolusi 1×1 dengan fungsi aktivasi sigmoid:

$$\hat{y}(i,j) = \sigma\big(f(i,j)\big) = \frac{1}{1+e^{-f(i,j)}} \tag{2.9}$$

di mana $f(i,j)$ adalah nilai keluaran konvolusi pada piksel $(i,j)$ dan $\hat{y}(i,j) \in [0,1]$ merupakan probabilitas piksel tersebut termasuk area *powdery mildew*. Nilai $\hat{y}(i,j)$ kemudian dibinarisasi menggunakan nilai *threshold* $\tau$ yang ditentukan melalui proses *threshold tuning* pada *validation set*:

$$\hat{m}(i,j) = \begin{cases} 1, & \text{jika } \hat{y}(i,j) \geq \tau \; 0, \\ \text{jika } \hat{y}(i,j) < \tau & \end{cases} \tag{2.10}$$

di mana $\hat{m}(i,j) = 1$ menunjukkan piksel termasuk area bercak *powdery mildew* dan $\hat{m}(i,j) = 0$ menunjukkan area daun sehat atau latar belakang. Mask biner $\hat{m}$ yang dihasilkan selanjutnya digunakan untuk menghitung persentase luas area infeksi sebagai dasar estimasi tingkat keparahan penyakit.

## 2.7 Fungsi Loss untuk Segmentasi

Pada tugas segmentasi penyakit tanaman, distribusi piksel antara area daun sehat dan area infeksi umumnya tidak seimbang (*class imbalance*). Area bercak *powdery mildew*

sering kali hanya menempati sebagian kecil dari keseluruhan citra daun, sehingga model berpotensi lebih mudah mempelajari kelas mayoritas dibandingkan kelas minoritas. Oleh karena itu, pemilihan fungsi *loss* menjadi faktor penting untuk memastikan model mampu mendeteksi area infeksi secara akurat [27, 28].

### 2.7.1 Binary Cross-Entropy Loss

*Binary Cross-Entropy* (BCE) adalah fungsi *loss* yang sering digunakan dalam masalah klasifikasi biner, termasuk pada segmentasi biner di tingkat piksel. Fungsi ini menghitung selisih antara nilai probabilitas hasil prediksi dan label sebenarnya pada setiap piksel secara terpisah.

$$\mathscr{L}\text{BCE} = -\frac{1}{N}\sum i = 1^{N} [y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \tag{2.11}$$

di mana $N$ adalah jumlah total piksel, $y_i \in \{0,1\}$ merupakan label *ground truth*, dan $\hat{y}_i \in [0,1]$ merupakan probabilitas prediksi model. BCE bekerja dengan baik pada distribusi kelas yang relatif seimbang. Namun, pada segmentasi penyakit tanaman sering terjadi ketidakseimbangan kelas karena jumlah piksel bercak jauh lebih sedikit dibandingkan piksel non-bercak. Akibatnya, model dapat memperoleh nilai *loss* yang rendah meskipun masih gagal mendeteksi sebagian area infeksi.

### 2.7.2 Dice Loss

*Dice Loss* diturunkan dari koefisien Sørensen-Dice yang mengukur tumpang tindih antara dua himpunan. Berbeda dengan BCE yang beroperasi per piksel secara independen, *Dice Loss* beroperasi secara global:

$$\mathscr{L}_{\text{Dice}} = 1 - \frac{2\sum_i y_i \hat{y}_i + \varepsilon}{\sum_i y_i + \sum_i \hat{y}_i + \varepsilon} \tag{2.12}$$

di mana $\varepsilon$ adalah konstanta kecil untuk menghindari pembagian dengan nol. *Dice Loss* lebih tahan terhadap ketidakseimbangan kelas dibandingkan BCE karena berfokus pada tingkat *overlap* antara area prediksi dan *ground truth*. Karena menekankan tingkat kesesuaian atau tumpang tindih (*overlap*) antara hasil prediksi dan *ground truth*, Dice Loss dianggap lebih cocok digunakan pada tugas segmentasi dibandingkan BCE. Pendekatan ini mendorong model untuk memaksimalkan area irisan antara mask prediksi dan mask referensi, sehingga performa segmentasi pada objek kecil dapat meningkat.

### 2.7.3 Focal Tversky Loss

*Tversky Loss* merupakan generalisasi dari *Dice Loss* yang memperkenalkan parameter $\alpha$ dan $\beta$ untuk mengontrol penalti terhadap *False Positive* (FP) dan *False Negative* (FN) secara terpisah [28]:

$$T(\alpha,\beta) = \frac{\sum_i y_i \hat{y}_i + \varepsilon}{\sum_i y_i \hat{y}_i + \alpha \sum_i (1-y_i)\hat{y}_i + \beta \sum_i y_i (1-\hat{y}_i) + \varepsilon} \tag{2.13}$$

*Focal Tversky Loss* merupakan pengembangan dari *Tversky Loss* yang diperkenalkan oleh Abraham et al.(2019) untuk mengatasi permasalahan ketidakseimbangan kelas (*class imbalance*) pada tugas segmentasi citra [27]. Dengan menambahkan faktor fokus $\gamma$, fungsi loss ini memberikan penalti yang lebih besar pada area objek yang sulit diprediksi, terutama objek berukuran kecil yang sering terabaikan oleh model:

$$\mathscr{L}_{\text{FTL}} = (1 - T(\alpha,\beta))^{\gamma} \tag{2.14}$$

Pada penelitian ini digunakan nilai $\alpha = 0{,}7$ sehingga kesalahan berupa *False Negative* memperoleh penalti yang lebih besar dibandingkan *False Positive*. Strategi ini dipilih karena kegagalan mendeteksi area infeksi (*missed lesion*) dianggap lebih merugikan dibandingkan deteksi positif palsu. Selain itu, nilai $\gamma = 0{,}75$ digunakan untuk meningkatkan fokus model terhadap piksel yang sulit diprediksi, khususnya area bercak *powdery mildew* yang berukuran relatif kecil dibandingkan keseluruhan area daun. Untuk memperoleh kemampuan klasifikasi piksel sekaligus mempertahankan kualitas segmentasi pada data yang tidak seimbang, fungsi *loss* yang digunakan pada penelitian ini merupakan kombinasi aditif antara *Binary Cross-Entropy* (BCE) dan *Focal Tversky Loss*, yang dirumuskan sebagai berikut:

$$\mathscr{L}\text{total} = \mathscr{L}\text{BCE} + \mathscr{L}_{\text{FTL}} \tag{2.15}$$

Kombinasi BCE dan *Focal Tversky Loss* digunakan karena kedua fungsi *loss* memiliki karakteristik yang saling melengkapi. BCE membantu model mempelajari klasifikasi piksel secara lokal, sedangkan *Focal Tversky Loss* mengoptimalkan kualitas segmentasi secara global melalui peningkatan *overlap* antara prediksi dan *ground truth*. Dengan menggabungkan keduanya, model diharapkan mampu menghasilkan segmentasi yang lebih stabil sekaligus lebih sensitif terhadap area infeksi berukuran kecil.

## 2.8 Perhitungan Area Infeksi dan Estimasi Tingkat Keparahan

Setelah model segmentasi menghasilkan *mask* daun dan area infeksi *powdery mildew*, langkah selanjutnya adalah menghitung persentase area infeksi sebagai dasar estimasi tingkat keparahan penyakit. Perhitungan dilakukan dengan memanfaatkan hasil segmentasi pada kedua tahap, yaitu *mask* daun hasil *background removal* dan *mask* area infeksi hasil segmentasi *powdery mildew*. Persentase area infeksi diperoleh dengan membandingkan jumlah piksel pada area infeksi terhadap jumlah piksel pada keseluruhan area daun. Tahapan perhitungan tersebut dijelaskan pada subbab berikut.

### 2.8.1 Perhitungan Area Daun

Area daun ($A_{\text{daun}}$) diperoleh dari output mask biner Stage 1 (*background removal*). Jumlah piksel bernilai 1 (foreground) pada mask tersebut merepresentasikan area daun dalam satuan piksel:

$$A_{\text{daun}} = \sum_{x,y} M_{\text{daun}}(x,y) \tag{2.16}$$

di mana $M_{\text{daun}}(x,y) \in \{0,1\}$ adalah mask *background removal* dan koordinat $(x,y)$ merentang seluruh dimensi citra.

### 2.8.2 Perhitungan Area Infeksi

Area infeksi ($A_{\text{bercak}}$) diperoleh dari output mask biner Stage 2 (segmentasi semantik *powdery mildew*):

$$A_{\text{bercak}} = \sum_{x,y} M_{\text{bercak}}(x,y) \tag{2.17}$$

di mana $M_{\text{bercak}}(x,y) \in \{0,1\}$ adalah mask segmentasi penyakit setelah *thresholding* dan *post-processing* (penghapusan komponen kecil (*small connected components removal*)).

### 2.8.3 Perhitungan Persentase Infeksi

Persentase infeksi dihitung menggunakan Persamaan 2.1. Pada implementasinya, luas area daun dan luas area bercak direpresentasikan sebagai jumlah piksel pada mask hasil segmentasi. Oleh karena itu, Persamaan 2.1 dapat dituliskan kembali dalam bentuk operasi piksel sebagai berikut:

$$Severity(\%) = \frac{\sum_{x,y} (M_{\text{daun}}(x,y) \cdot M_{\text{bercak}}(x,y))}{\sum_{x,y} M_{\text{daun}}(x,y)} \times 100 \tag{2.18}$$

di mana $M_{\text{daun}}(x,y)$ merupakan mask hasil *background removal* dan $M_{\text{bercak}}(x,y)$ merupakan mask hasil segmentasi *powdery mildew*. Operasi perkalian kedua mask digunakan untuk memastikan bahwa area infeksi yang dihitung hanya berada pada wilayah daun. Nilai *severity* kemudian dipetakan ke beberapa kategori keparahan sesuai Tabel 2.1.

## 2.9 Evaluasi Model Segmentasi

Evaluasi model segmentasi dilakukan untuk mengukur kemampuan model dalam menghasilkan hasil segmentasi yang sesuai dengan *ground truth*. Proses evaluasi dilakukan dengan membandingkan hasil prediksi dan anotasi pada tingkat piksel sehingga kualitas segmentasi dapat dinilai secara kuantitatif. Pada penelitian ini, performa model dievaluasi menggunakan beberapa metrik yang umum digunakan pada tugas segmentasi semantik, yaitu *Dice Coefficient*, *Intersection over Union* (IoU), *Precision*, dan *Recall*. Masing-masing metrik memberikan sudut pandang yang berbeda dalam menilai kualitas segmentasi, sehingga penggunaannya secara bersama-sama dapat memberikan gambaran yang lebih komprehensif mengenai performa model.

### 2.9.1 Dice Coefficient

*Dice Coefficient*, yang ekuivalen dengan *F1-Score* pada segmentasi biner, mengukur tingkat kesesuaian antara mask prediksi dan mask *ground truth* [29]. Metrik ini merupakan rata-rata harmonik antara *precision* dan *recall*, sehingga mampu mengevaluasi keseimbangan keduanya.

$$\text{Dice} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} = \frac{2|P \cap G|}{|P| + |G|} \tag{2.19}$$

di mana $P$ merupakan himpunan piksel hasil prediksi dan $G$ merupakan himpunan piksel *ground truth*. Nilai *Dice Coefficient* berada pada rentang 0 hingga 1, di mana nilai 0 menunjukkan tidak adanya tumpang tindih antara hasil prediksi dan *ground truth*, sedangkan nilai 1 menunjukkan kesesuaian sempurna. Pada penelitian ini, *Dice Coefficient* digunakan sebagai salah satu metrik utama karena mampu merepresentasikan kualitas segmentasi secara keseluruhan.

### 2.9.2 Intersection over Union (IoU)

*Intersection over Union* (IoU), atau sering disebut sebagai *Jaccard Index*, mengukur rasio antara area irisan dan area gabungan dari mask prediksi dan mask *ground truth* [29]. Nilai IoU dihitung menggunakan Persamaan 2.20.

$$\text{IoU} = \frac{|P \cap G|}{|P \cup G|} = \frac{TP}{TP + FP + FN} \tag{2.20}$$

IoU merupakan salah satu metrik yang paling umum digunakan dalam evaluasi segmentasi semantik karena mampu memberikan penalti yang lebih ketat terhadap kesalahan segmentasi dibandingkan *Dice Coefficient*. Semakin besar nilai IoU, semakin tinggi tingkat kesesuaian antara hasil segmentasi dan *ground truth*.

Hubungan matematis antara IoU dan *Dice Coefficient* dapat dinyatakan sebagai berikut:

$$\text{Dice} = \frac{2 \cdot \text{IoU}}{1 + \text{IoU}} \tag{2.21}$$

sehingga untuk tingkat kesesuaian yang sama, nilai IoU umumnya lebih rendah dibandingkan nilai *Dice Coefficient*.

Dengan demikian, evaluasi pada penelitian ini diarahkan untuk menilai kualitas segmentasi yang dihasilkan pada setiap tahapan sistem. Metrik *Precision*, *Recall*, *Dice Coefficient*, dan IoU digunakan untuk mengukur kesesuaian hasil segmentasi terhadap *ground truth* pada tingkat piksel. Nilai-nilai tersebut menjadi indikator kemampuan model dalam mempertahankan area daun pada tahap *background removal* maupun dalam mendeteksi area infeksi *powdery mildew* pada tahap segmentasi penyakit. Hasil evaluasi yang baik diharapkan dapat mendukung perhitungan persentase area infeksi secara lebih akurat sehingga estimasi tingkat keparahan penyakit (*severity*) yang dihasilkan menjadi lebih reliabel.

### 2.9.3 Precision

*Precision* mengukur tingkat ketepatan prediksi area positif (bercak) yang benar-benar merupakan area bercak aktual [30]. Nilai *precision* dihitung menggunakan Persamaan 2.22.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.22}$$

Nilai *precision* yang tinggi menunjukkan bahwa model jarang menghasilkan deteksi bercak palsu (*false positive*). Dalam konteks segmentasi penyakit tanaman, *precision* menggambarkan kemampuan model dalam menghindari kesalahan identifikasi area daun sehat sebagai area infeksi.

### 2.9.4 Recall

*Recall*, atau yang dikenal sebagai *sensitivity* maupun *True Positive Rate* (TPR), merupakan metrik yang digunakan untuk mengukur seberapa besar proporsi area bercak yang sebenarnya berhasil terdeteksi oleh model [30]. Nilai *recall* tersebut dapat dihitung menggunakan Persamaan 2.23.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2.23}$$

Nilai *recall* yang tinggi menunjukkan bahwa sebagian besar area infeksi berhasil terdeteksi oleh model. Pada penelitian ini, *recall* menjadi metrik yang penting karena kegagalan mendeteksi area infeksi (*false negative*) dapat menyebabkan estimasi tingkat keparahan penyakit menjadi lebih rendah dari kondisi sebenarnya.