BAB 2
LANDASAN TEORI

2.1 Penyakit *Powdery Mildew* pada Daun Labu

Powdery Mildew adalah penyakit jamur yang menyerang daun tanaman hortikultura, termasuk labu (*Cucurbita* spp.). Infeksi ditandai oleh lapisan miselium putih menyerupai tepung yang menutupi permukaan daun, menghambat penetrasi cahaya dan fotosintesis, sehingga dapat mengurangi kadar klorofil dan menurunkan hasil panen secara signifikan [31, 6]. Patogen utama seperti *Erysiphe cichoracearum* dan *Podosphaera xanthii* berkembang optimal pada suhu 20–27 °C dan kelembaban relatif 60–90% [7]. Gejala ini secara visual mudah dikenali dan menjadi dasar bagi sistem deteksi penyakit berbasis citra. Berikut adalah ilustrasi gejala penyakit *Powdery Mildew* oleh [17]:

Gambar 2.1. Ilustrasi bercak putih khas *Powdery Mildew* yang muncul pada permukaan daun labu. Sumber: [17].

Powdery Mildew melewati siklus hidup yang melibatkan produksi spora

konidia pada permukaan daun, yang kemudian tersebar melalui angin, air, atau kontak langsung antar daun. Ketika kondisi lingkungan—seperti suhu hangat dan kelembaban tinggi—memenuhi syarat, spora ini berkecambah di permukaan daun, menembus lapisan sel inang, dan membentuk miselium yang memproduksi konidia baru secara aseksual. Dengan siklus yang hanya memerlukan beberapa hari, infeksi dapat menyebar sangat cepat dalam satu lahan budidaya.

Dari perspektif fisiologi tanaman, keberadaan lapisan miselium menyebabkan gangguan pada stomata dan menghambat pertukaran gas, sehingga tanaman mengalami stres air dan penurunan efisiensi fotosintesis. Secara ekonomi, wabah *Powdery Mildew* dapat menurunkan hasil panen hingga 50% dan meningkatkan biaya produksi akibat kebutuhan aplikasi fungisida berulang. Oleh karena itu, deteksi dini yang akurat dan cepat sangat penting untuk mengoptimalkan strategi pengendalian, mengurangi penggunaan kimia, dan menjaga keberlanjutan produksi hortikultura.

2.2 Pra-pemrosesan Citra

Pra-pemrosesan citra adalah tahap awal penting dalam sistem pengenalan berbasis visi komputer. Tujuannya adalah untuk menyiapkan data dalam bentuk yang konsisten dan optimal untuk proses pelatihan model. Dua langkah penting yang sering digunakan dalam pra-pemrosesan adalah dan normalisasi nilai piksel.

2.2.1 Resize dan Normalisasi

Citra digital yang digunakan dalam sistem pengenalan berbasis *Convolutional Neural Network* (CNN) umumnya memiliki dimensi yang tidak seragam. Untuk memastikan kompatibilitas dengan arsitektur CNN, setiap citra perlu diseragamkan ukurannya melalui proses *resize*. Langkah ini penting agar dimensi input konsisten antar citra, memungkinkan efisiensi dalam proses pelatihan serta pemrosesan dalam bentuk *batch*. Beberapa arsitektur CNN mensyaratkan dimensi input tetap, seperti 224×224 atau 256×256 piksel, tergantung pada struktur dan kebutuhan arsitektur yang digunakan.

Setelah dilakukan *resize*, tahap berikutnya adalah normalisasi nilai piksel. Secara umum, nilai intensitas piksel dalam citra RGB berada dalam rentang $[0, 255]$. Rentang ini dianggap terlalu besar untuk diproses langsung oleh jaringan saraf, karena dapat menyebabkan ketidakseimbangan numerik pada proses propagasi

sinyal. Oleh karena itu, nilai piksel biasanya dinormalisasi ke dalam rentang $[0, 1]$ menggunakan transformasi sebagai berikut:

$$x' = \frac{x}{255.0} \tag{2.1}$$

dengan x adalah nilai piksel asli dan x' adalah hasil normalisasi. Proses ini tidak hanya menyederhanakan skala input, tetapi juga memiliki dampak penting dalam menjaga kestabilan pelatihan jaringan saraf.

Salah satu alasan utama dilakukannya normalisasi adalah untuk mencegah dua permasalahan umum dalam pelatihan jaringan saraf dalam (*deep neural networks*), yaitu *exploding gradients* dan *vanishing gradients*. Kedua fenomena ini terjadi pada saat proses propagasi balik (*backpropagation*) dan dapat menghambat efektivitas pembelajaran jaringan.

- *Exploding gradients* merujuk pada kondisi di mana nilai gradien menjadi sangat besar, sehingga menyebabkan pembaruan bobot yang ekstrem dan membuat model tidak stabil, bahkan dapat menghasilkan nilai tak hingga (*NaN*). Hal ini umumnya disebabkan oleh bobot awal yang terlalu besar atau akumulasi dari banyak lapisan tanpa normalisasi antar lapisan.

- Sebaliknya, *vanishing gradients* terjadi ketika nilai gradien menjadi sangat kecil (mendekati nol), sehingga pembaruan bobot menjadi tidak signifikan. Hal ini membuat jaringan kesulitan mempelajari representasi yang mendalam, terutama pada lapisan awal.

Dengan menormalkan nilai piksel input, distribusi data yang masuk ke jaringan menjadi lebih terkendali. Hal ini membantu mencegah penyimpangan nilai aktivasi maupun gradien secara ekstrem—baik terlalu besar maupun terlalu kecil. Secara keseluruhan, normalisasi berkontribusi pada kestabilan numerik selama pelatihan, mempercepat proses konvergensi, dan meningkatkan peluang jaringan mencapai minimum lokal dari fungsi *loss* dengan lebih efisien.

2.2.2 Augmentasi Data

Untuk meningkatkan kemampuan generalisasi model dan mencegah *overfitting*, teknik *data augmentation* diterapkan dengan cara melakukan transformasi visual ringan terhadap citra latih. Transformasi ini menghasilkan variasi sintetis dari data asli tanpa mengubah labelnya, sehingga memungkinkan

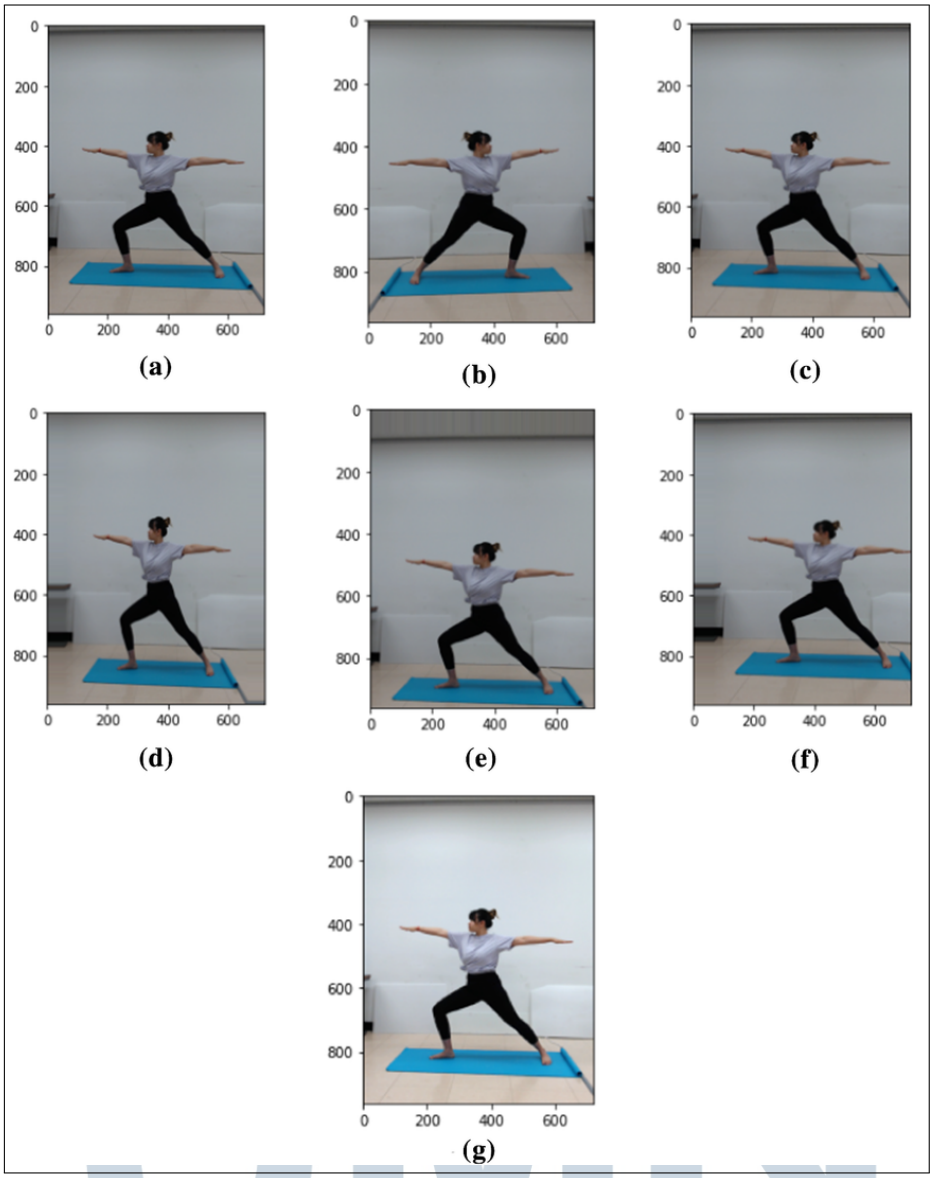
model untuk belajar menjadi lebih *invariant* terhadap perubahan posisi, orientasi, dan skala objek dalam citra.

Beberapa transformasi umum yang digunakan dalam augmentasi citra antara lain:

- **Rotasi**: memutar citra hingga beberapa derajat (misalnya $\pm 15°$) untuk meniru orientasi daun yang bervariasi.
- **Perpindahan (shift)**: menggeser citra secara horizontal atau vertikal untuk merepresentasikan perbedaan posisi objek.
- **Shear**: menerapkan distorsi paralel ringan pada citra guna mensimulasikan perubahan sudut pandang.
- **Zoom**: memperbesar atau memperkecil citra dalam rentang tertentu untuk menangani variasi ukuran objek.
- **Flip horizontal**: mencerminkan citra secara horizontal untuk menangani simetri visual objek.
- **Fill mode**: metode interpolasi (misalnya *nearest*) yang digunakan untuk mengisi area kosong akibat transformasi geometris.

Teknik augmentasi dapat diterapkan secara *offline* (pra-pelatihan) maupun secara dinamis (*online*) selama proses pelatihan berlangsung. Tujuan utamanya adalah memperluas keragaman data latih agar model tidak hanya menghafal pola visual tetap, tetapi juga mampu menggeneralisasi terhadap berbagai variasi citra yang mungkin muncul di dunia nyata.

Gambar 2.2 berikut memberikan ilustrasi visual terhadap beberapa bentuk augmentasi citra yang umum digunakan dalam pelatihan model berbasis CNN. Berikut adalah ilustrasi oleh [32]:



Gambar 2.2. Contoh hasil augmentasi citra: gambar asli dan hasil transformasi seperti rotasi, flipping, *shear*, zoom, serta perubahan kecerahan. Sumber: [32].

2.3 Penanganan Ketidakseimbangan Kelas

Ketidakseimbangan kelas merupakan kondisi ketika jumlah sampel pada satu kelas jauh lebih besar dibanding kelas lainnya. Dalam konteks pelatihan model klasifikasi, kondisi ini dapat menyebabkan model lebih fokus dalam meminimalkan kesalahan pada kelas mayoritas, karena kontribusinya lebih dominan dalam fungsi *loss*. Akibatnya, kelas minoritas cenderung diabaikan dan performa model menjadi bias.

Secara umum, ketidakseimbangan dapat diatasi melalui dua pendekatan: menyeimbangkan distribusi data secara eksplisit (melalui teknik seperti *oversampling*, *undersampling*, atau augmentasi sintetis), atau dengan memberikan bobot pada setiap kelas dalam fungsi *loss* agar kelas minoritas mendapatkan penalti yang lebih besar saat terjadi kesalahan.

Pemberian bobot merupakan pendekatan berbasis fungsi loss yang tidak mengubah distribusi data asli, namun mengoreksi ketimpangan kontribusi antar kelas dalam proses pelatihan. Bobot untuk setiap kelas i dapat dihitung menggunakan rumus sebagai berikut:

$$w_i = \frac{N}{K \cdot N_i}, \tag{2.2}$$

dengan N adalah total sampel, K adalah jumlah kelas, dan N_i adalah jumlah sampel pada kelas ke-i. Dengan cara ini, kelas dengan jumlah data yang lebih sedikit akan memberikan kontribusi yang lebih besar terhadap nilai loss, sehingga model terdorong untuk mempelajari pola dari kelas tersebut secara lebih seimbang.

Pemberian bobot tidak bertujuan menggantikan teknik penyeimbangan data, melainkan sebagai alternatif atau pelengkap, terutama ketika modifikasi distribusi data dianggap tidak praktis atau berisiko menyebabkan overfitting. Oleh karena itu, pendekatan ini tetap relevan untuk mengatasi ketidakseimbangan kelas secara langsung dalam proses pelatihan model.

2.4 Pembagian Data

Sebelum melatih model klasifikasi, dataset perlu dibagi menjadi subset yang mencerminkan distribusi sebenarnya dari data secara proporsional. Proses ini penting untuk memastikan bahwa model tidak hanya belajar dari sebagian kecil pola, tetapi dapat menggeneralisasi ke seluruh populasi data. Dalam proyek ini, yang melibatkan deteksi dini penyakit *Powdery Mildew* pada daun labu menggunakan CNN, beberapa rasio pembagian data diuji untuk menilai kestabilan performa model.

Salah satu metode yang digunakan adalah *stratified sampling*, yaitu teknik pembagian data yang mempertahankan proporsi tiap kelas secara konsisten di setiap subset. Misalnya, jika kelas *Powdery Mildew* hanya mencakup 20% dari keseluruhan data, maka proporsi tersebut akan dijaga baik di *train set*, *validation set*, maupun *test set*.

Langkah umum dalam stratified hold-out:

1. Membagi dataset menjadi *test set* (misalnya 10–20%) untuk evaluasi akhir.
2. Sisanya dibagi menjadi *train set* dan *validation set*, umumnya dengan rasio 80:20 atau 85:15 tergantung skenario eksperimen.

Untuk menilai sensitivitas model terhadap ukuran subset, berikut adalah beberapa contoh skema yang dapat diuji:

- 10% test : 20% validation : 70% train
- 15% test : 15% validation : 70% train
- 20% test : 10% validation : 70% train

Sebagai alternatif dari pembagian hold-out, teknik *K-Fold Cross-Validation* juga dapat digunakan. Metode ini membagi dataset menjadi K bagian (misalnya $K = 5$), lalu secara bergiliran menggunakan satu bagian sebagai data uji dan sisanya sebagai data latih. Hasil dari K eksperimen tersebut dirata-ratakan untuk memperoleh metrik yang lebih stabil dan tidak bergantung pada satu pembagian acak saja.

Pendekatan ini memastikan evaluasi model dilakukan secara adil dan tidak bias, sekaligus mengeksplorasi seberapa sensitif arsitektur CNN terhadap variasi dalam proporsi data.

A Klasifikasi Biner

Dataset asli terdiri dari lima kelas citra, yaitu *Bacterial Leaf Spot*, *Downy Mildew*, *Healthy Leaf*, *Mosaic Disease*, dan *Powdery Mildew*. Untuk memfokuskan penelitian pada deteksi penyakit *Powdery Mildew*, seluruh label dalam dataset disederhanakan menjadi dua kelas utama:

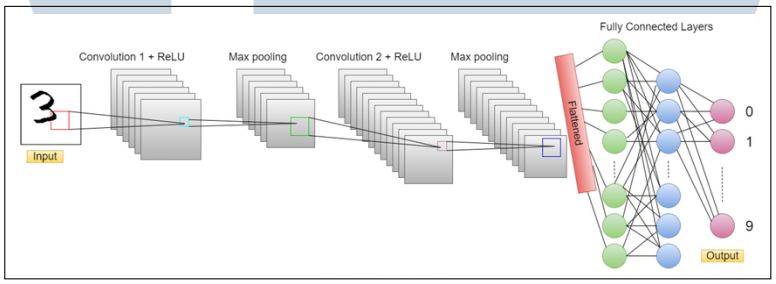
- **Powdery Mildew (label 1)**: Mewakili seluruh citra daun yang terinfeksi penyakit *Powdery Mildew*.
- **Not Powdery Mildew (label 0)**: Mewakili citra daun sehat (*Healthy Leaf*) serta daun yang terinfeksi penyakit lain seperti *Bacterial Leaf Spot*, *Downy Mildew*, dan *Mosaic Disease*.

Penyederhanaan ini dilakukan untuk membentuk skenario klasifikasi biner yang menekankan kemampuan model dalam membedakan keberadaan atau

ketiadaan penyakit *Powdery Mildew*, tanpa membedakan jenis penyakit lain. Relabeling dilakukan pada tahap awal sebelum pembagian dataset dan pelatihan model.

2.5 Convolutional Neural Network (CNN)

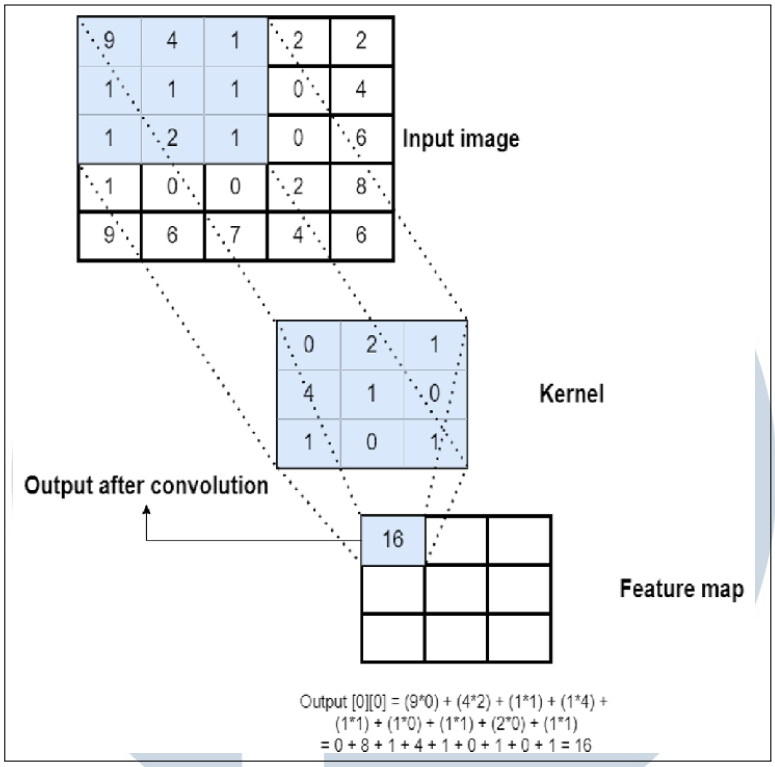
Convolutional Neural Network (CNN) adalah arsitektur deep learning yang dirancang khusus untuk memproses data bergambar. CNN meniru cara kerja sistem visual manusia dalam mengenali pola, dimulai dari fitur sederhana seperti tepi hingga fitur kompleks seperti tekstur dan bentuk bercak penyakit [33]. Dalam konteks penelitian ini, CNN digunakan untuk mendeteksi gejala awal *Powdery Mildew* pada daun labu secara otomatis dan efisien.



Gambar 2.3. Arsitektur umum CNN terdiri dari blok konvolusi, aktivasi, pooling, dan *fully connected layer*. Sumber: [33].

2.5.1 Convolutional Layer

Lapisan konvolusi merupakan komponen utama dalam arsitektur Convolutional Neural Network (CNN) yang bertujuan untuk mengekstraksi fitur lokal dari citra input, seperti tepi, sudut, atau tekstur. Proses ini dilakukan dengan menerapkan filter (kernel) berukuran tetap, misalnya $k \times k$, yang bergerak melintasi seluruh area citra [34].



Gambar 2.4. Ilustrasi operasi konvolusi dua dimensi (2D) pada citra menggunakan filter berukuran 3×3 untuk menghasilkan peta fitur. Sumber: [34].

Nilai keluaran di posisi (i, j) dari filter ke-f pada *feature map* hasil konvolusi dihitung dengan persamaan berikut:

$$y_{i,j}^{(f)} = \sum_{u=1}^{k} \sum_{v=1}^{k} x_{i+u-1,\, j+v-1} \cdot w_{u,v}^{(f)} + b^{(f)} \tag{2.3}$$

Di mana:

- $y_{i,j}^{(f)}$ adalah nilai output pada posisi (i, j) dari filter ke-f.
- $x_{i+u-1,\, j+v-1}$ adalah nilai piksel input pada lokasi yang dilalui kernel.
- $w_{u,v}^{(f)}$ adalah bobot kernel ke-f pada posisi (u, v).
- $b^{(f)}$ adalah bias untuk filter ke-f.
- k adalah ukuran sisi kernel (misalnya $k = 3$ untuk kernel 3×3).

Ketika konvolusi dilakukan dengan padding (P) dan stride (S), maka dimensi keluaran (tinggi dan lebar) dihitung menggunakan rumus berikut:

$$H_{\text{out}} = \frac{H_{\text{in}} - k + 2P}{S} + 1, \quad W_{\text{out}} = \frac{W_{\text{in}} - k + 2P}{S} + 1 \tag{2.4}$$

Dengan:

- $H_{\text{in}}, W_{\text{in}}$: tinggi dan lebar dari input image.
- $H_{\text{out}}, W_{\text{out}}$: tinggi dan lebar dari output feature map.
- P: jumlah padding di sekeliling citra input.
- S: langkah pergeseran (stride) dari kernel saat bergerak melintasi input.

Rumus ini memastikan ukuran output dapat diprediksi tergantung pada parameter yang digunakan dalam proses konvolusi.

2.5.2 Activation Function

Untuk memperkenalkan non-linearitas ke dalam jaringan, CNN menggunakan fungsi aktivasi yang diterapkan setelah operasi konvolusi. Tanpa fungsi aktivasi, jaringan akan bersifat linear dan tidak mampu memodelkan relasi kompleks dalam data.

Fungsi aktivasi yang paling umum digunakan dalam CNN adalah ReLU (Rectified Linear Unit), yang didefinisikan sebagai berikut:

$$\text{ReLU}(x) = \max(0, x) \tag{2.5}$$

Di mana:

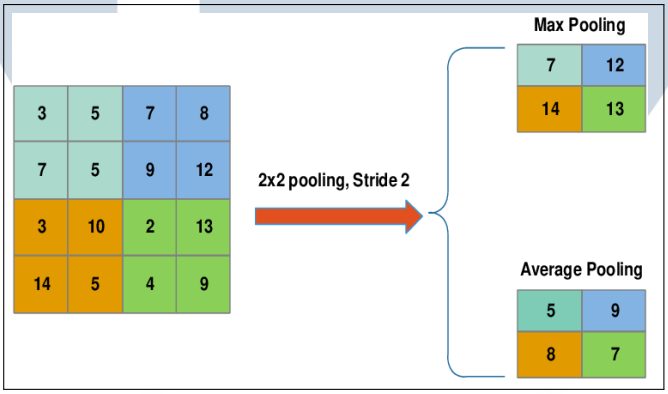
- x adalah nilai input dari hasil konvolusi (sebelum aktivasi).
- $\text{ReLU}(x)$ adalah output yang ditetapkan menjadi nol jika x negatif, dan tetap x jika x positif.

ReLU bersifat sederhana namun efektif dalam mengurangi masalah *vanishing gradient*, karena mempertahankan gradien untuk nilai positif. Namun, dalam beberapa kasus, ReLU dapat menyebabkan neuron menjadi tidak aktif secara permanen (dikenal sebagai *dying ReLU*). Untuk mengatasi hal ini, beberapa varian alternatif seperti Leaky ReLU atau ELU (Exponential Linear Unit) dapat digunakan, yang memungkinkan nilai output tetap kecil meskipun input negatif.

2.5.3 Pooling Layer

Pooling layer merupakan komponen penting dalam arsitektur *Convolutional Neural Network* (CNN) yang berfungsi untuk mereduksi dimensi spasial dari *feature map*. Dengan mengurangi resolusi data, pooling dapat menurunkan jumlah parameter dan komputasi, serta membantu mengurangi risiko *overfitting*. Selain itu, pooling memperkenalkan sifat invarian terhadap translasi kecil, sehingga fitur penting tetap dapat dikenali meskipun terjadi pergeseran posisi dalam citra.

Pooling bekerja dengan menerapkan jendela lokal berukuran tetap (misalnya 2×2 atau 3×3) yang digeser dengan *stride* tertentu, dan menghitung nilai agregasi dari setiap jendela tersebut. Dua jenis pooling yang paling umum adalah *max pooling* dan *average pooling* [35].



Gambar 2.5. Perbandingan hasil *max pooling* dan *average pooling* pada jendela berukuran 2×2. Sumber: .

Max Pooling

Max pooling memilih nilai maksimum dari setiap jendela lokal. Tujuannya adalah mempertahankan fitur yang paling dominan dalam wilayah tersebut, sehingga memperkuat keberadaan fitur penting dan mengurangi sensitivitas terhadap noise atau gangguan lokal.

Secara matematis, jika x adalah *feature map* hasil konvolusi, maka nilai keluaran $y_{i,j}$ dari operasi *max pooling* berukuran 2×2 dengan *stride* 2 dapat dituliskan sebagai:

$$y_{i,j} = \max_{0 \le u,v < 2} x_{2i+u,2j+v} \tag{2.6}$$

Dengan:

- $x_{2i+u,2j+v}$ merupakan nilai piksel pada posisi yang tercakup dalam jendela pooling berukuran 2×2 pada *feature map*.
- $y_{i,j}$ adalah nilai output pada posisi (i, j) setelah proses pooling.
- Indeks u dan v menjelajah elemen-elemen dalam jendela lokal, dan *stride* 2 berarti jendela bergeser dua piksel setiap langkahnya.

Operasi ini secara efektif mengurangi dimensi spasial dari *feature map* sambil mempertahankan fitur yang paling kuat di setiap area lokal.

Average Pooling

Average pooling menghitung nilai rata-rata dari elemen-elemen dalam jendela lokal. Berbeda dengan max pooling yang fokus pada nilai ekstrem, average pooling mempertahankan informasi global dengan cara meredam dominasi nilai lokal tertentu. Jenis pooling ini menghasilkan representasi yang lebih halus dan stabil, serta dapat berguna ketika informasi menyeluruh lebih penting dibandingkan fitur dominan.

Secara matematis, average pooling dengan ukuran jendela 2×2 dan *stride* 2 dapat dinyatakan sebagai:

$$y_{i,j} = \frac{1}{4} \sum_{0 \leq u,v < 2} x_{2i+u,2j+v} \tag{2.7}$$

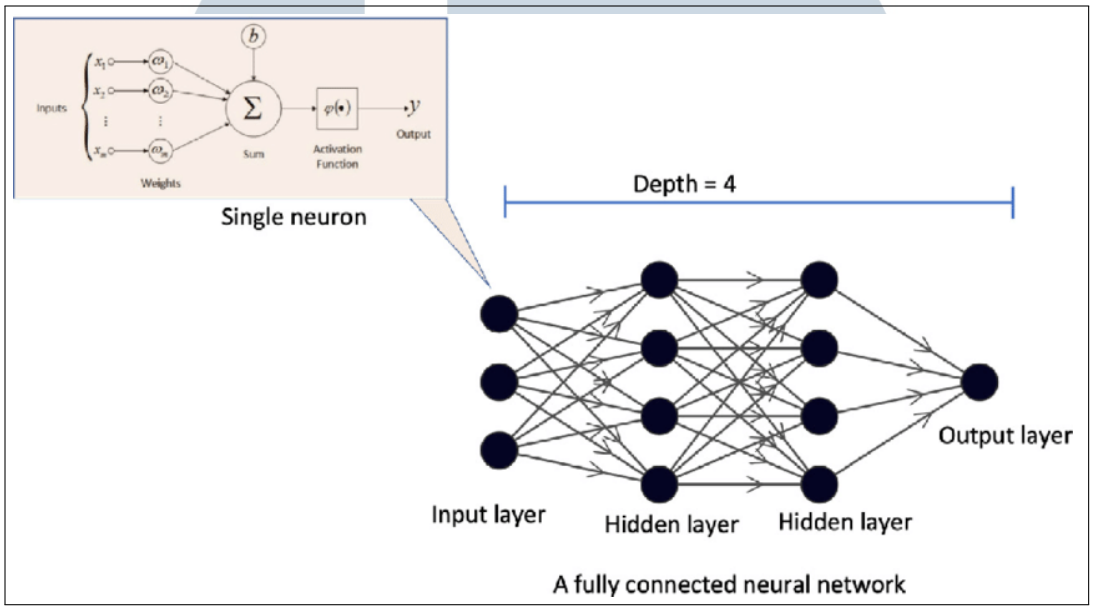
Dengan:

- $x_{2i+u,2j+v}$ merupakan nilai-nilai dalam jendela lokal berukuran 2×2 pada *feature map*.
- $\sum_{0 \leq u,v < 2}$ berarti menjumlahkan seluruh piksel dalam jendela tersebut.
- Faktor $\frac{1}{4}$ digunakan untuk menghitung rata-rata dari empat elemen dalam jendela 2×2.
- $y_{i,j}$ adalah nilai rata-rata hasil pooling yang ditempatkan di posisi (i, j) pada output map.

Kedua teknik pooling tersebut memiliki peran penting dalam proses ekstraksi fitur, dan pemilihannya tergantung pada kebutuhan arsitektur model serta karakteristik data yang digunakan.

2.5.4 Fully Connected Layer

Setelah beberapa blok konvolusi dan pooling, peta fitur yang dihasilkan akan diratakan (flatten) menjadi vektor satu dimensi sebelum dikirim ke lapisan *fully connected* (FC), yang berfungsi sebagai pengambil keputusan akhir dalam klasifikasi [36].



Gambar 2.6. Ilustrasi lapisan *fully connected* yang menghubungkan seluruh neuron hasil proses *flatten* ke setiap kelas output. Sumber: [36].

Secara matematis, transformasi linear dalam lapisan fully connected dapat dinyatakan sebagai:

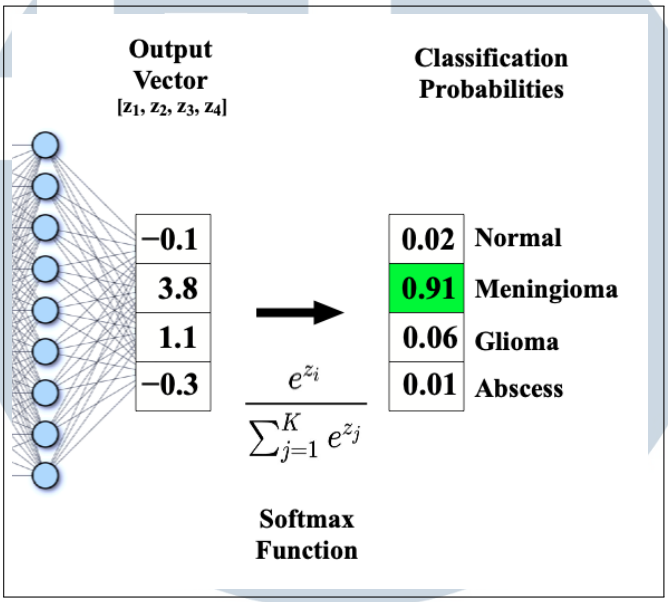
$$z = W_{\text{fc}} h + b_{\text{fc}} \tag{2.8}$$

Dengan:

- h adalah vektor input hasil flatten dari peta fitur konvolusi.
- W_{fc} adalah matriks bobot dari lapisan fully connected.
- b_{fc} adalah vektor bias.
- z adalah hasil output linear sebelum diberi aktivasi (misalnya Softmax).

2.5.5 Softmax Output

Lapisan output dalam jaringan CNN untuk klasifikasi biasanya menggunakan fungsi aktivasi Softmax. Fungsi ini mengubah output linear dari lapisan fully connected menjadi distribusi probabilitas atas seluruh kelas [37].



Gambar 2.7. Distribusi probabilitas kelas yang dihasilkan oleh fungsi aktivasi *Softmax*. Sumber: .

Secara matematis, fungsi Softmax didefinisikan sebagai:

$$P(y = k \mid x) = \frac{e^{z_k}}{\sum_j e^{z_j}} \tag{2.9}$$

Dengan:

- z_k adalah skor (logit) dari kelas ke-k yang dihasilkan oleh lapisan fully connected.
- $\sum_j e^{z_j}$ adalah penjumlahan eksponensial dari semua skor kelas, untuk normalisasi.
- $P(y = k \mid x)$ adalah probabilitas prediksi bahwa input x termasuk ke dalam kelas k.

Fungsi Softmax memastikan bahwa seluruh output berada dalam rentang [0, 1] dan total probabilitas dari semua kelas berjumlah 1, sehingga cocok digunakan pada tugas klasifikasi multikelas.

2.5.6 Regularisasi dan Optimisasi

Agar pelatihan CNN berjalan stabil dan menghindari overfitting, digunakan beberapa teknik regulasi dan optimisasi yang terbukti efektif dalam meningkatkan generalisasi model serta mempercepat proses konvergensi.

Batch Normalization

Teknik ini menstabilkan distribusi input antar lapisan dengan menormalkan aktivasi dalam mini-batch agar memiliki rata-rata nol dan varians satu. Operasi ini diikuti dengan transformasi linier menggunakan parameter yang dapat dilatih.

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \quad y = \gamma\hat{x} + \beta \tag{2.10}$$

Dengan:

- x: nilai input sebelum normalisasi.
- μ_B, σ_B^2: rata-rata dan varians dari mini-batch.
- ε: nilai kecil untuk menjaga stabilitas numerik.
- $\hat{x}$: hasil normalisasi.
- γ, β: parameter skala dan pergeseran yang dapat dilatih.
- y: output akhir setelah normalisasi.

Dropout

Dropout adalah teknik regularisasi yang bekerja dengan menonaktifkan neuron secara acak selama pelatihan. Tujuannya adalah untuk mencegah co-adaptation antar neuron dan mendorong jaringan untuk belajar representasi yang lebih robust.

$$y_i = z_i \cdot r_i, \quad r_i \sim \text{Bernoulli}(p) \tag{2.11}$$

Dengan:

- z_i: nilai aktivasi dari neuron ke-i sebelum dropout.

- r_i: variabel acak yang mengikuti distribusi Bernoulli, dengan probabilitas p untuk aktif (1) dan $1 - p$ untuk dinonaktifkan (0).
- y_i: output neuron ke-i setelah diterapkan dropout.

Optimizer

Optimisasi bertujuan memperbarui bobot model untuk meminimalkan fungsi loss berdasarkan gradien. Beberapa algoritma yang umum digunakan meliputi:

- **SGD (Stochastic Gradient Descent)**: metode dasar berbasis penurunan gradien dengan pembaruan iteratif.
- **RMSProp**: menyimpan rata-rata kuadrat gradien untuk setiap parameter dan menyesuaikan pembaruan secara adaptif.
- **Adam**: menggabungkan momentum dan rata-rata kuadrat gradien, memungkinkan pembaruan yang stabil dan konvergensi cepat.

Dalam penelitian ini digunakan Adam karena efisiensinya dalam mengatasi dinamika gradien yang tidak stabil dan memberikan konvergensi yang cepat pada jaringan dalam.

A ReduceLROnPlateau

ReduceLROnPlateau merupakan salah satu metode penjadwalan laju pembelajaran (*learning rate scheduler*) yang digunakan untuk menyesuaikan nilai *learning rate* secara dinamis selama proses pelatihan jaringan saraf. Teknik ini dirancang untuk mengatasi permasalahan stagnasi pembelajaran yang terjadi ketika metrik validasi, seperti *loss* atau *accuracy*, tidak mengalami perbaikan dalam sejumlah epoch berturut-turut.

Pendekatan ini bekerja dengan cara menurunkan nilai *learning rate* sebesar faktor tertentu apabila tidak terjadi peningkatan performa pada metrik yang dimonitor setelah melewati jumlah epoch yang telah ditentukan sebelumnya (*patience*). Dengan penurunan nilai *learning rate*, proses pembaruan bobot menjadi lebih halus, sehingga dapat membantu model mencapai konvergensi yang lebih stabil dan menghindari kemungkinan overshooting terhadap minimum lokal dari fungsi *loss*.

Beberapa parameter utama yang digunakan dalam *ReduceLROnPlateau* meliputi:

- **monitor**: metrik evaluasi yang diamati, seperti `val_loss` atau `val_accuracy`,
- **patience**: jumlah epoch tanpa peningkatan performa sebelum *learning rate* diturunkan,
- **factor**: nilai pengurang terhadap *learning rate* saat ini,
- **min_lr**: nilai minimum *learning rate* yang diperbolehkan.

Dengan demikian, *ReduceLROnPlateau* berperan sebagai mekanisme adaptif yang dapat meningkatkan efisiensi dan efektivitas proses pelatihan, terutama dalam tahap-tahap akhir pembelajaran ketika model mulai mendekati titik konvergensi.

2.5.7 Desain Hyperparameter

Kinerja model CNN sangat bergantung pada pemilihan hyperparameter yang tepat. Hyperparameter adalah parameter yang tidak dipelajari secara langsung dari data, melainkan ditentukan sebelum proses pelatihan dimulai. Pemilihan nilai-nilai ini dilakukan melalui serangkaian eksperimen bertahap dan validasi performa pada data uji silang (cross-validation). Berikut adalah kelompok hyperparameter utama yang memengaruhi arsitektur dan proses pelatihan CNN:

Ukuran dan Jumlah Filter Konvolusi

Ukuran filter (*kernel size*), seperti 3×3 atau 5×5, menentukan seberapa besar area lokal pada citra yang dianalisis oleh masing-masing neuron. Semakin besar ukuran filter, semakin banyak konteks spasial yang diperhitungkan, namun juga meningkatkan jumlah parameter.

Jumlah filter per lapisan konvolusi menentukan jumlah fitur yang dapat diekstraksi pada level tersebut. CNN umumnya menggunakan jumlah filter yang meningkat secara bertahap (misalnya $32 \rightarrow 64 \rightarrow 128$) agar fitur yang lebih kompleks dapat dipelajari pada lapisan yang lebih dalam.

Stride dan Padding

Stride menentukan seberapa jauh filter bergeser saat melakukan konvolusi. Nilai stride yang lebih besar menghasilkan output yang lebih kecil dan agresif dalam mereduksi dimensi, namun dapat menghilangkan detail penting.

Padding digunakan untuk mengontrol ukuran output feature map, biasanya dilakukan dengan menambahkan nol di sekitar citra input. Padding yang tepat (*same padding*) mempertahankan dimensi spasial dan penting untuk mempertahankan informasi di tepi citra.

Jumlah Blok Konvolusi dan Neuron Fully Connected

Jumlah blok konvolusi yang digunakan menentukan kedalaman arsitektur. Blok-blok ini biasanya terdiri dari satu atau dua lapisan konvolusi yang diikuti oleh pooling. Penambahan blok memungkinkan model menangkap fitur kompleks, namun juga meningkatkan risiko overfitting dan waktu pelatihan.

Di akhir jaringan, lapisan fully connected menggabungkan fitur-fitur hasil konvolusi untuk membuat prediksi akhir. Jumlah neuron dalam lapisan ini juga menjadi hyperparameter penting—jumlah terlalu kecil dapat membatasi kapasitas representasi, sedangkan terlalu besar dapat meningkatkan risiko overfitting.

Hyperparameter Pelatihan: Learning Rate, Batch Size, dan Epoch

- **Learning rate** mengatur seberapa besar langkah pembaruan parameter saat proses optimisasi. Nilai yang terlalu besar dapat menyebabkan model gagal konvergen, sementara nilai terlalu kecil dapat memperlambat pelatihan.
- **Batch size** menentukan jumlah sampel yang digunakan dalam satu iterasi pelatihan. Ukuran batch kecil membuat pelatihan lebih cepat tetapi kurang stabil, sedangkan batch besar lebih stabil tetapi memerlukan memori lebih besar.
- **Epoch** adalah jumlah keseluruhan iterasi di mana seluruh dataset dilalui oleh model. Jumlah epoch yang tepat bergantung pada kompleksitas data dan risiko overfitting, sehingga biasanya dikombinasikan dengan *early stopping*.

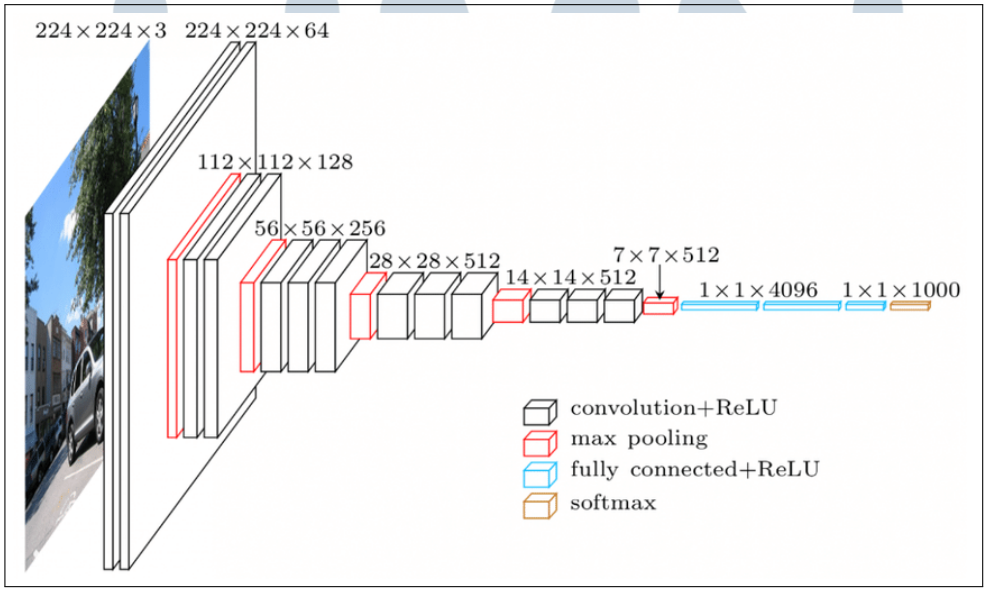
Pemilihan Hyperparameter

Pemilihan hyperparameter umumnya dilakukan melalui eksperimen bertahap dan validasi performa model. Strategi yang sering digunakan mencakup pencarian grid (grid search), pencarian acak (random search), atau pendekatan adaptif seperti Bayesian optimization. Evaluasi hyperparameter biasanya didasarkan pada metrik kinerja seperti akurasi, loss, atau nilai F1-score pada data validasi.

2.6 Arsitektur VGG-16

VGG-16 adalah salah satu arsitektur CNN paling populer dan banyak digunakan dalam tugas klasifikasi citra. Arsitektur ini diperkenalkan oleh Simonyan dan Zisserman dari Visual Geometry Group, University of Oxford, dalam kompetisi ILSVRC tahun 2014 [19]. Ciri khas utama dari VGG-16 adalah penggunaan banyak lapisan konvolusi kecil (3×3) secara bertumpuk, dengan jumlah total 16 lapisan yang memiliki parameter dapat dilatih (13 lapisan konvolusi dan 3 fully connected).

Model ini dirancang untuk menerima input berupa citra RGB berukuran $224 \times 224 \times 3$ piksel (tinggi $\times$ lebar $\times$ kanal), sehingga seluruh citra input perlu diubah ukurannya agar sesuai dengan dimensi tersebut sebelum dimasukkan ke jaringan.



Gambar 2.8. Ilustrasi arsitektur bertingkat dari VGG-16, yang terdiri atas blok konvolusi berulang dan lapisan klasifikasi.

2.6.1 Struktur VGG-16

Struktur arsitektur VGG-16 terdiri dari lima blok konvolusi, masing-masing diikuti oleh lapisan *max pooling*, kemudian dilanjutkan dengan tiga lapisan fully connected. Rinciannya adalah sebagai berikut:

- **Blok 1–2:** Masing-masing terdiri dari dua lapisan konvolusi berukuran 3×3, dengan 64 dan 128 filter.
- **Blok 3–5:** Masing-masing terdiri dari tiga lapisan konvolusi 3×3, dengan jumlah filter berturut-turut 256, 512, dan 512.
- **Pooling:** Setiap blok diakhiri dengan lapisan *max pooling* berukuran 2×2 dengan *stride* 2.
- **Fully Connected:** Tiga lapisan FC, terdiri dari dua lapisan dengan 4096 neuron, dan satu lapisan output dengan 1000 neuron (untuk klasifikasi ImageNet).
- **Output:** Fungsi aktivasi Softmax digunakan untuk menghasilkan distribusi probabilitas terhadap 1000 kelas.

2.6.2 Keunggulan VGG-16 dalam Transfer Learning

Transfer learning memungkinkan penggunaan bobot dari model yang telah dilatih sebelumnya pada dataset berskala besar seperti ImageNet untuk diterapkan pada tugas klasifikasi baru dengan data terbatas. VGG-16 sangat cocok untuk pendekatan ini karena:

- **Bobot Pretrained yang Stabil:** Telah dilatih pada lebih dari satu juta gambar ImageNet.
- **Lapisan Konvolusi yang Dalam:** Efektif dalam mengekstraksi fitur tekstur dan pola visual kompleks seperti gejala penyakit daun.
- **Arsitektur Sederhana:** Konsistensi dalam penggunaan kernel 3×3 memudahkan adaptasi, interpretasi, dan implementasi ulang.
- **Kompatibel dengan Fine-tuning:** Lapisan awal dapat dibekukan (*frozen*) untuk mempertahankan pengetahuan visual umum, sedangkan lapisan akhir dapat dilatih ulang sesuai kebutuhan domain.

2.6.3 Penerapan dalam Penelitian

Dalam penelitian ini, arsitektur VGG-16 digunakan sebagai basis klasifikasi citra daun labu. Lapisan akhir dimodifikasi untuk mengklasifikasikan lima kelas penyakit: *Powdery Mildew*, *Downy Mildew*, *Bacterial Leaf Spot*, *Mosaic Disease*, dan daun sehat. Hanya bagian classifier (Dense dan Dropout) yang dilatih ulang, sedangkan bobot pretrained dari lapisan konvolusi digunakan kembali sebagai fitur extractor.

2.6.4 Fine-tuning dan Freezing

Teknik *fine-tuning* digunakan untuk menyesuaikan lapisan akhir model dengan domain klasifikasi baru. Lapisan awal—yang bertanggung jawab mendeteksi fitur visual dasar seperti tepi, sudut, dan tekstur—dibekukan agar pengetahuan umum yang telah dipelajari dari ImageNet tetap terjaga.

Proses pembekuan parameter dituliskan sebagai:

$$\text{layer.trainable} = \text{False} \tag{2.12}$$

Setelah pembekuan, hanya lapisan baru yang ditambahkan ke bagian akhir model—seperti *Flatten*, *Dense*, dan *Dropout*—yang diatur sebagai *trainable* dan dilibatkan dalam proses pelatihan ulang.

2.7 Metrik Evaluasi Klasifikasi

Untuk mengevaluasi performa model dalam mengklasifikasikan daun labu yang sehat dan terinfeksi *Powdery Mildew*, digunakan sejumlah metrik evaluasi klasifikasi. Metrik ini memberikan gambaran menyeluruh tidak hanya tentang seberapa akurat model dalam mengenali pola, tetapi juga bagaimana model memperlakukan kelas minoritas seperti penyakit yang menjadi fokus deteksi dini. Metrik-metrik ini sangat penting mengingat dataset memiliki kelas yang tidak seimbang dan tugas klasifikasi bersifat biner.

2.7.1 Confusion Matrix

Confusion matrix merupakan tabel berukuran 2×2 yang mencatat hasil klasifikasi model ke dalam empat kategori:

- TP (True Positive): Kasus *Powdery Mildew* yang diprediksi benar sebagai positif.
- FP (*False Positive*): Kasus sehat yang salah diprediksi sebagai positif.
- FN (*False Negative*): Kasus *Powdery Mildew* yang tidak terdeteksi.
- TN (True Negative): Kasus sehat yang diprediksi benar sebagai negatif.

2.7.2 Accuracy

Akurasi adalah proporsi prediksi benar terhadap seluruh jumlah prediksi:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.13}$$

Metrik ini cocok digunakan apabila distribusi data antar kelas bersifat seimbang.

2.7.3 Precision

Precision mengukur seberapa tepat model dalam memprediksi kelas positif:

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2.14}$$

Nilai *precision* tinggi berarti sebagian besar prediksi positif memang benar adanya, dan berguna dalam konteks di mana kesalahan tipe I (*false positive*) perlu dihindari.

2.7.4 Recall (Sensitivity)

Recall atau sensitivitas mengukur seberapa banyak kasus positif yang berhasil terdeteksi oleh model:

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2.15}$$

Metrik ini penting dalam konteks deteksi dini penyakit, karena berfokus pada menghindari kesalahan tipe II (*false negative*) yang dapat menyebabkan penyakit tidak terdeteksi.

2.7.5 F1-Score

F1-score adalah rata-rata harmonik antara *precision* dan *Recall*, yang berguna ketika diperlukan keseimbangan antara keduanya:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.16}$$

F1-score menjadi penting dalam konteks klasifikasi dengan distribusi kelas tidak seimbang, seperti pada kasus deteksi penyakit tanaman. Metrik ini memastikan bahwa model tidak hanya fokus pada ketepatan (*precision*), tetapi juga pada kemampuan untuk mendeteksi kasus penyakit sebanyak mungkin (*Recall*).

2.7.6 Fokus pada Recall dalam Deteksi Powdery Mildew

Dalam konteks deteksi dini penyakit *Powdery Mildew*, kesalahan tipe II (*false negative*) memiliki dampak yang lebih berbahaya dibanding kesalahan tipe I. Jika model gagal mendeteksi daun yang sebenarnya terinfeksi, penyakit dapat menyebar lebih luas sebelum dilakukan tindakan. Oleh karena itu, **recall menjadi metrik yang sangat krusial** dalam penelitian ini, karena mewakili kemampuan model dalam mengidentifikasi semua kasus penyakit secara menyeluruh.

F1-score juga dipertimbangkan sebagai metrik pendukung untuk memastikan performa model tidak berat sebelah antara *precision* dan *recall*, terutama saat mengevaluasi generalisasi model terhadap data uji.