BAB 3
METODOLOGI PENELITIAN

Dalam upaya mengembangkan sistem deteksi penyakit *Powdery Mildew* pada daun labu secara otomatis, penelitian ini dihadapkan pada sejumlah tantangan mendasar yang berkaitan dengan keterbatasan data dan kompleksitas pemodelan. Dataset yang tersedia memiliki jumlah data yang terbatas dan distribusi kelas yang tidak seimbang, yang berpotensi menurunkan kemampuan model dalam mengenali gejala penyakit secara akurat, terutama untuk kelas minoritas [38, 14].

Di sisi lain, meskipun arsitektur *Convolutional Neural Network (CNN)* telah banyak digunakan dalam klasifikasi citra daun, efektivitasnya sangat dipengaruhi oleh arsitektur yang digunakan serta kondisi dataset yang dilatih [28, 13]. Pemilihan antara membangun model CNN secara mandiri atau memanfaatkan arsitektur *pretrained* seperti VGG16 menjadi salah satu pertimbangan penting dalam penelitian ini [39, 40]. Untuk memperjelas posisi dan kontribusi penelitian ini, dilakukan telaah literatur terhadap studi-studi terdahulu yang relevan dengan konteks permasalahan yang diangkat.

3.1 Telaah Literatur

Perkembangan teknologi deteksi penyakit tanaman berbasis citra digital telah membuka peluang besar dalam dunia pertanian presisi. Deteksi dini penyakit pada tanaman hortikultura seperti labu (*Cucurbita spp.*) menjadi sangat penting untuk meminimalisir kerugian hasil panen akibat penyebaran patogen seperti *Powdery Mildew*. Namun, metode konvensional yang mengandalkan pengamatan visual sering kali terbatas karena subjektivitas dan keterbatasan tenaga ahli di lapangan. Oleh karena itu, penerapan algoritma *Convolutional Neural Network* (CNN) sebagai bagian dari deep learning menjadi solusi alternatif yang menjanjikan, dengan kemampuan untuk mengekstraksi fitur kompleks dari citra daun dan mengklasifikasikan penyakit secara otomatis.

Berikut ini adalah beberapa studi terdahulu yang relevan mengenai penerapan CNN dalam mendeteksi penyakit *Powdery Mildew*:

Tabel 3.1. Studi terdahulu menggunakan model deep learning untuk deteksi *Powdery Mildew* pada daun labu

Peneliti (Tahun)	**Model / Metode**	**Objek**	**Hasil utama**
Khaldi & Kalmoun [41]	DenseNet121, ResNet34, EfficientNetB7	Daun labu (Bangladesh) – 4 penyakit termasuk *Powdery Mildew*	DenseNet121: Akurasi 86.0%
Biswas et al. [17]	Transfer Learning (MobileNetV2, DenseNet121, Xception, InceptionV3)	Daun labu (Bangladesh) – 4 penyakit termasuk *Powdery Mildew*	ResNet50: 74.68%, EfficientNet: 79.74%, MobileNet: 95.44%, DenseNet: 97.82%
Khandaker et al. [42]	Explainable CNN (ResNet50 + Grad-CAM)	Daun labu (Bangladesh) – 4 penyakit termasuk *Powdery Mildew*	Akurasi 90.5%
Long et al. (2023) [43]	MobileNet, InceptionV3, VGG16, Xception, CerealConv konvensional	Gandum – 4 penyakit termasuk *Powdery Mildew*	CerealConv buatan: Akurasi 97.05%
Zhu et al. [44]	GCS-YOLOv8 (spora mikroskopis)	Mentimun – spora *Powdery Mildew*	Akurasi 92.6%

Tabel 3.1 merangkum berbagai penelitian sebelumnya yang memanfaatkan arsitektur deep learning untuk mendeteksi penyakit tanaman, khususnya *Powdery Mildew*. Penjelasan berikut menguraikan secara rinci setiap studi, termasuk dataset, metode yang digunakan, serta temuan utama yang relevan dengan penelitian ini. Khaldi & Kalmoun [41] melakukan analisis komprehensif terhadap beberapa arsitektur CNN terkemuka seperti DenseNet121, ResNet34, dan EfficientNetB7

pada *Pumpkin Leaf Disease Dataset*. Dataset ini mencakup lima kelas, yaitu daun sehat, *Downy Mildew*, *Powdery Mildew*, *Mosaic Disease*, dan *Bacterial Leaf Spot*. Mereka menemukan bahwa DenseNet121 memberikan kinerja optimal dengan akurasi keseluruhan 86,0%. Namun, mereka mencatat adanya kesulitan yang lebih besar dalam mendeteksi *Powdery Mildew* dibandingkan penyakit lainnya, sehingga mereka menyarankan pengembangan model lebih lanjut atau penggunaan fitur tambahan untuk meningkatkan performa.

Biswas et al. [17] mengembangkan dataset baru dengan 4.000 citra daun labu yang mengandung berbagai penyakit, termasuk *Powdery Mildew*. Penelitian ini melatih empat model CNN (ResNet50, EfficientNet, MobileNetV2, dan DenseNet121). DenseNet121 kembali menunjukkan akurasi tertinggi sebesar 97,82%, diikuti MobileNetV2 (95,44%), EfficientNet (79,74%), dan ResNet50 (74,68%). Penelitian ini berkontribusi besar dalam mendemonstrasikan keefektifan transfer learning untuk klasifikasi penyakit daun labu tetapi belum menguji model pada citra lapangan yang kompleks.

Khandaker et al. [42] memperkenalkan pendekatan Explainable AI (XAI) untuk memberikan transparansi pada proses klasifikasi penyakit. Mereka menggunakan ResNet50 yang mencapai akurasi 90,5% pada dataset yang sama. Selain itu, metode seperti Grad-CAM, Score-CAM, dan Layer-CAM digunakan untuk memvisualisasikan area pada citra yang memengaruhi prediksi model. Pendekatan ini penting untuk membangun kepercayaan pengguna dalam implementasi praktis di bidang pertanian.

Long et al. [43] mengambil pendekatan berbeda dengan membuat dataset gandum yang diambil dalam kondisi realistis di lapangan dan rumah kaca. Model CNN yang dikembangkan, yaitu CerealConv, berhasil mencapai akurasi 97,05% dan bahkan melampaui kinerja ahli patologi tanaman pada subset citra tertentu. Akan tetapi, mereka mencatat adanya tingkat kesalahan klasifikasi yang lebih tinggi pada gambar *Powdery Mildew*.

Penelitian [44] berfokus pada deteksi spora mikroskopis *Powdery Mildew* pada mentimun menggunakan model GCS-YOLOv8. Mereka memperkenalkan beberapa perbaikan arsitektur, seperti Global Context Attention dan Content-Aware Reassembly of Features (CARAFE), untuk meningkatkan deteksi target kecil. Dengan akurasi 92,6%, model ini sangat efektif dalam mendeteksi spora pada citra mikroskopis, meskipun konteksnya berbeda dengan penelitian ini yang menargetkan citra daun secara makro.

Berdasarkan kajian terhadap lima studi tersebut, dapat disimpulkan bahwa

meskipun arsitektur CNN telah menunjukkan efektivitas tinggi dalam klasifikasi penyakit tanaman, tantangan khusus tetap muncul pada deteksi *Powdery Mildew*, terutama karena kemiripan visualnya dengan daun sehat dan variabilitas kondisi pencitraan. Beberapa studi seperti [41] dan [43] secara eksplisit menyoroti kesulitan model dalam mengenali penyakit ini. Selain itu, sebagian besar penelitian sebelumnya hanya membandingkan performa model transfer learning tanpa membandingkan dengan model CNN konvensional yang belum dilatih. Oleh karena itu, penelitian ini berupaya mengisi celah tersebut dengan membandingkan performa model CNN konvensional dan VGG16 dalam mendeteksi *Powdery Mildew* pada daun labu. VGG16 dipilih karena strukturnya yang dalam namun sederhana, yang cenderung lebih tahan terhadap overfitting dibandingkan arsitektur yang lebih kompleks saat diaplikasikan pada dataset berukuran sedang. Selain itu, pendekatan ini diharapkan dapat meningkatkan ketepatan klasifikasi untuk penyakit dengan ciri visual halus dan memperkaya literatur deteksi penyakit tanaman melalui pendekatan berbasis CNN yang lebih terfokus dan adaptif.

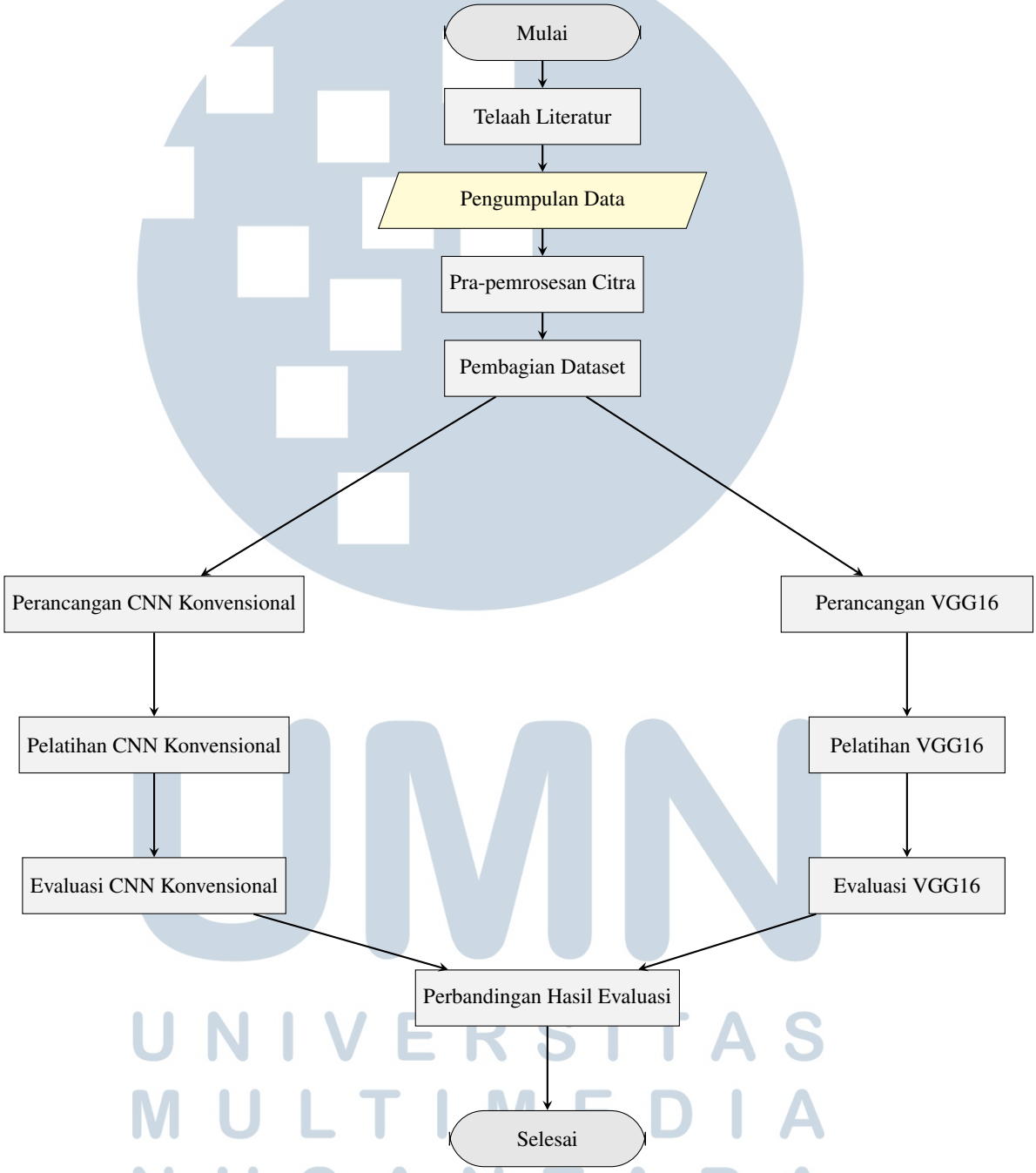
3.2 Perancangan Sistem

Penelitian ini dirancang secara terstruktur untuk mendeteksi penyakit *Powdery Mildew* pada daun labu menggunakan algoritma *Convolutional Neural Network* (CNN). Perencanaan ini mencakup berbagai tahapan mulai dari studi literatur, pemrosesan dataset, perancangan dan pelatihan model, hingga tahap evaluasi dan pengujian akhir. Dengan pendekatan yang sistematis ini, diharapkan hasil penelitian dapat diperoleh secara objektif dan replikatif.

Untuk memberikan gambaran umum mengenai alur pelaksanaan penelitian, Gambar 3.1 menyajikan diagram alir yang merangkum tahapan-tahapan utama dalam proses penelitian.

Diagram di atas menggambarkan tahapan-tahapan utama dalam Bab III yang membentuk metodologi penelitian ini. Proses dimulai dari telaah literatur sebagai dasar konseptual untuk memahami perkembangan terkini dalam deteksi penyakit tanaman menggunakan CNN. Kemudian dilanjutkan dengan pengumpulan data, yaitu dataset citra daun labu yang telah dikategorikan ke dalam lima kelas, termasuk kelas *Powdery Mildew*.

Tahap berikutnya adalah pra-pemrosesan, yang mencakup penyesuaian ukuran citra, normalisasi, dan penanganan saluran warna sesuai kebutuhan model. Setelah itu, dilakukan pembagian dataset secara stratifikasi menjadi subset



Gambar 3.1. Diagram alur tahapan dalam Bab III: Metodologi Penelitian.

pelatihan, validasi, dan pengujian untuk memastikan distribusi kelas yang seimbang di seluruh data.

Selanjutnya, dilakukan perancangan arsitektur model untuk dua pendekatan berbeda: model CNN konvensional yang dirancang dari awal, dan model berbasis VGG16 dengan teknik *fine-tuning*. Kedua model tersebut kemudian menjalani rangkaian pelatihan dengan konfigurasi yang seragam, termasuk skema augmentasi offline, penggunaan callback *EarlyStopping* dan *ReduceLROnPlateau*, serta strategi penanganan ketidakseimbangan kelas melalui *class weighting*.

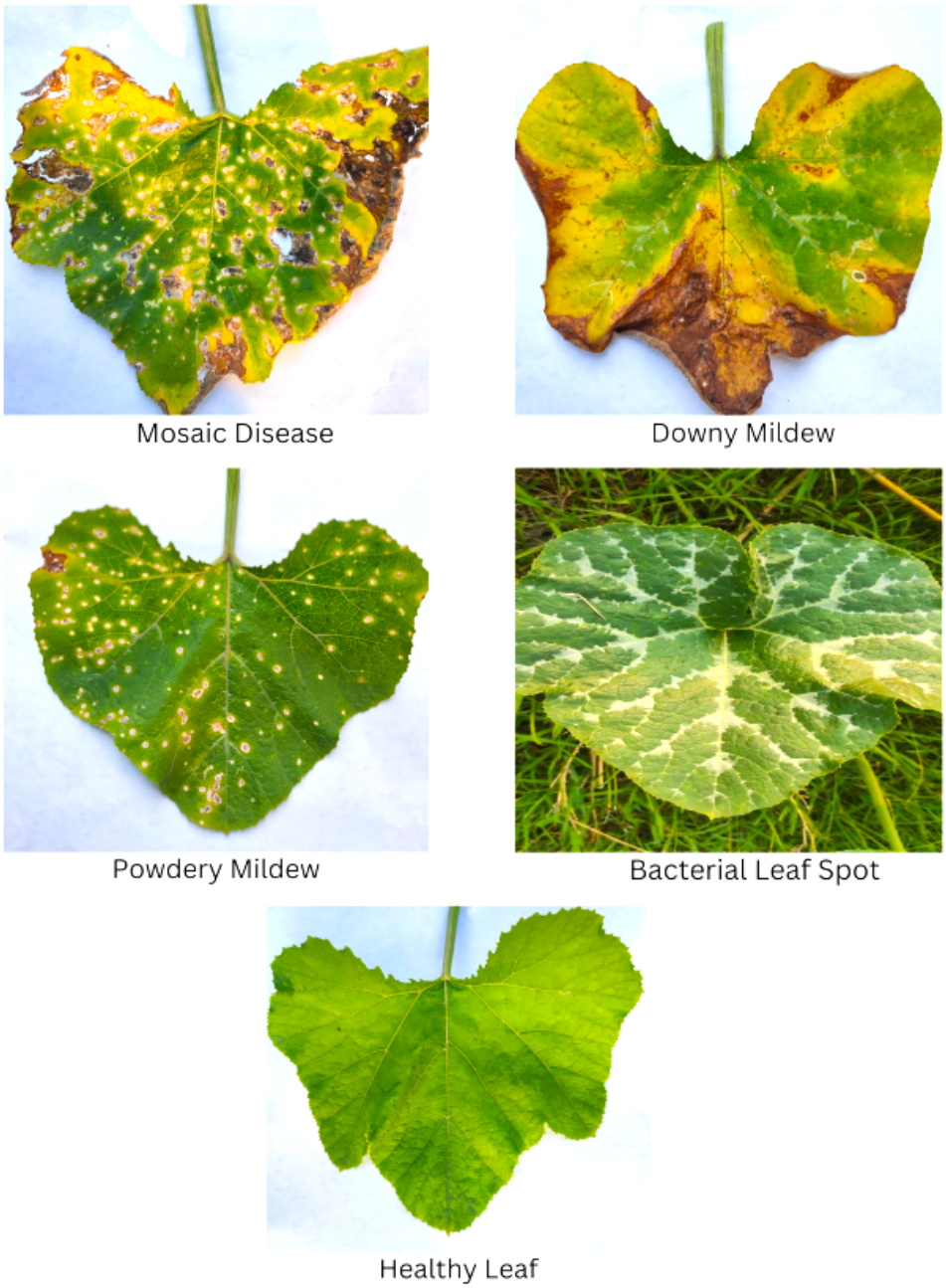
Terakhir, model dievaluasi dan divisualisasikan pada bagian Visualisasi dan Evaluasi Model menggunakan metrik performa seperti akurasi, presisi, recall, dan F1-score. Tahapan ini juga mencakup perbandingan hasil antar arsitektur dan antar skema pembagian data, sebagai dasar untuk menarik kesimpulan mengenai efektivitas pendekatan deteksi dini penyakit *Powdery Mildew*.

Penjelasan lebih rinci dari setiap tahapan pada diagram akan diuraikan dalam subbagian-subbagian berikutnya dalam bab ini.

3.3 Pengumpulan Data

Penelitian ini menggunakan *Pumpkin Leaf Disease Dataset* yang dikembangkan oleh Biswas et al. [17]. Dataset ini dirancang khusus untuk klasifikasi penyakit daun labu dan telah digunakan dalam berbagai penelitian terdahulu terkait deteksi penyakit tanaman berbasis deep learning. Terdapat total 2.000 citra RGB resolusi tinggi yang terbagi secara merata ke dalam lima kategori: *Powdery Mildew*, *Mosaic Disease*, *Downy Mildew*, *Bacterial Leaf Spot*, dan daun sehat, masing-masing terdiri dari 400 gambar.

Seluruh citra dikumpulkan langsung dari lahan pertanian di berbagai wilayah menggunakan kamera ponsel seperti Redmi Note 10 Pro Max, Redmi Note 9 Pro Max, dan POCO X5, guna merepresentasikan kondisi lapangan yang bervariasi secara visual. Proses pengambilan gambar dilakukan dengan pencahayaan alami tanpa teknik pra-pemrosesan tambahan, sehingga mencerminkan gejala visual yang umum ditemukan oleh petani di lapangan. Setiap citra berformat RGB dengan resolusi dan orientasi yang beragam. Pemilihan dataset ini mempertimbangkan ketersediaannya yang terbuka serta distribusi kelas yang seimbang, yang sangat penting untuk pelatihan model CNN yang adil dan representatif [17].



Gambar 3.2. Contoh citra lima kelas pada *Pumpkin Leaf Disease Dataset*: *Powdery Mildew*, *Mosaic Disease*, *Downy Mildew*, *Bacterial Leaf Spot*, dan daun sehat. Sumber: [17].

Penggunaan dataset ini juga memungkinkan penelitian ini untuk dibandingkan secara langsung dengan studi sebelumnya yang menggunakan arsitektur CNN seperti DenseNet, ResNet, dan EfficientNet, sekaligus menguji pendekatan alternatif berbasis VGG16 serta model CNN konvensional yang

dirancang khusus untuk mendeteksi *Powdery Mildew*.

3.4 Pra-pemrosesan Data

Pra-pemrosesan merupakan tahap penting dalam pipeline pembelajaran mesin, terutama dalam konteks pengolahan citra. Tujuan utamanya adalah untuk standarisasi format input, mengurangi noise, memperluas keragaman data, dan memastikan data berada dalam skala numerik yang optimal bagi jaringan saraf konvolusional (CNN). Proses ini disesuaikan berdasarkan arsitektur model yang digunakan—baik CNN konvensional maupun model berbasis VGG16.

3.4.1 Pra-pemrosesan untuk Model CNN Konvensional

Pra-pemrosesan bertujuan untuk menyiapkan data citra agar sesuai dengan kebutuhan arsitektur CNN konvensional. Seluruh citra dalam dataset dikonversi ke ukuran 256×256 piksel serta dinormalisasi ke dalam rentang nilai `[0, 1]`. Proses ini dilakukan menggunakan pendekatan sebagai berikut:

- Membaca setiap citra berformat RGB dari lokasi penyimpanan dataset.
- Mengubah ukuran setiap citra menjadi 256×256 piksel untuk menyamakan dimensi input.
- Melakukan konversi tipe data ke `float32` untuk memastikan presisi numerik dalam komputasi.
- Melakukan normalisasi nilai piksel dengan membaginya terhadap 255, sehingga setiap nilai piksel berada dalam rentang kontinu antara 0 dan 1.

Setelah tahap normalisasi, dilakukan augmentasi citra secara *offline* untuk meningkatkan keragaman data latih tanpa memperbesar kompleksitas pelatihan model. Teknik augmentasi meliputi:

- Rotasi acak dengan sudut maksimum 15 derajat.
- Pergeseran posisi horizontal dan vertikal maksimum sebesar 10% dari dimensi citra.
- Transformasi *shear* untuk menambahkan variasi geometri.

- Zoom in atau zoom out hingga 10%.
- Pembalikan citra secara horizontal (*horizontal flip*).
- Pengisian area kosong dengan piksel terdekat menggunakan metode `nearest neighbor`.

Proses augmentasi dilakukan terhadap seluruh data latih secara terpisah sebelum dimulainya pelatihan. Data hasil augmentasi kemudian digabungkan dengan data asli untuk membentuk dataset pelatihan akhir, sedangkan label masing-masing citra diperluas sesuai urutan penggabungan.

Pseudocode: Pra-pemrosesan dan Augmentasi untuk CNN Konvensional

Algorithm 1 Pra-pemrosesan dan Augmentasi Dataset

for setiap *gambar* dalam dataset **do**
    Baca citra dalam format RGB
    Ubah ukuran menjadi 256×256 piksel
    Normalisasi piksel ke rentang [0, 1]
    Tambahkan citra ke dalam list data
end for
Terapkan augmentasi pada seluruh data latih:
- rotasi acak (max 15°)
- pergeseran horizontal dan vertikal (max 10%)
- shear, zoom, dan pembalikan horizontal

Gabungkan data asli dengan data hasil augmentasi
Gabungkan label asli dengan label hasil augmentasi

3.4.2 Pra-pemrosesan untuk Model VGG16

Ukuran citra juga diubah menjadi `256×256` piksel, dan augmentasi dilakukan dengan parameter yang lebih agresif dibandingkan model CNN konvensional. Tujuannya adalah memperluas kemampuan generalisasi model terhadap variasi visual daun labu yang tidak ditemukan dalam dataset ImageNet.

Augmentasi mencakup rotasi acak hingga 30°, pergeseran horizontal dan vertikal maksimum 20%, distorsi *shear*, pembesaran atau pengecilan (zoom), serta

pembalikan horizontal. Semua augmentasi dilakukan dengan pengisian area kosong menggunakan metode *nearest fill*.

Setelah data asli dan hasil augmentasi digabungkan, seluruh data pelatihan dan validasi diproses lebih lanjut menggunakan fungsi `preprocess_input` dari pustaka `tensorflow.keras.applications.vgg16`. Fungsi ini akan menyesuaikan citra agar selaras dengan distribusi data pada ImageNet, termasuk konversi warna dari RGB ke BGR, pengurangan nilai rata-rata per channel, dan penskalaan piksel. Proses ini penting agar bobot pralatih yang dimiliki VGG16 dapat bekerja secara optimal pada domain baru.

Langkah-langkah pra-pemrosesan untuk model VGG16 dapat dirangkum dalam pseudocode berikut:

Algorithm 2 Pra-pemrosesan Data untuk Model VGG16

1: Ubah ukuran semua citra menjadi `256×256` piksel.
2: Normalisasi nilai piksel ke dalam skala [0, 255].
3: Lakukan augmentasi data secara *offline* dengan parameter agresif: rotasi hingga 30°, pergeseran horizontal dan vertikal 20%, *shear* 20%, *zoom* 20%, dan pembalikan horizontal.
4: Gabungkan data asli dan data hasil augmentasi.
5: Terapkan fungsi `preprocess_input()` dari VGG16 pada seluruh citra hasil gabungan untuk menyesuaikan dengan standar ImageNet.

Langkah ini krusial untuk memastikan bahwa bobot pra-latih VGG16 dapat digunakan secara efektif dalam mendeteksi fitur pada domain baru, yaitu daun labu.

3.5 Pembagian Dataset

Pembagian dataset dilakukan untuk memisahkan data pelatihan, validasi, dan pengujian guna memastikan proses pelatihan model tidak mengalami kebocoran data dan mampu dievaluasi secara objektif. Dalam penelitian ini, digunakan tiga skema pembagian data untuk keperluan evaluasi model secara menyeluruh: *stratified split* (sebagai pendekatan utama), *direct 80:20 split*, dan *direct 70:30 split*. Rincian dari masing-masing skema ditampilkan pada Tabel 3.2.

Tabel 3.2. Perbandingan skema pembagian data

Skema	Train (%)	Validasi (%)	Uji (%)	Jumlah Citra (4000)
Stratified Split	72%	18%	10%	2880 : 720 : 400
Direct Split 80:20	80%	10%	10%	3200 : 400 : 400
Direct Split 70:30	70%	15%	15%	2800 : 600 : 600

3.5.1 Stratified Split (72% : 18% : 10%)

Algorithm 3 Pembagian Data dengan Stratified Split

1: Sisihkan 10% data dari keseluruhan dataset sebagai data uji.
2: Ambil sisa 90% untuk diproses lebih lanjut.
3: Bagi sisa 90% tersebut menjadi 80% data pelatihan dan 20% data validasi.
4: Pastikan seluruh pembagian menggunakan stratifikasi label untuk menjaga proporsi kelas.

Pada skema ini, sebanyak 10% dari dataset awal disisihkan terlebih dahulu sebagai data uji untuk memastikan independensi data pengujian. Sisanya (90%) kemudian dibagi secara stratifikasi dengan rasio 80:20 untuk menghasilkan data pelatihan dan validasi. Hasil akhir pembagian ini adalah 72% data pelatihan, 18% data validasi, dan 10% data pengujian.

3.5.2 Direct Split 80:20 (Train : Val : Test = 80% : 10% : 10%)

Algorithm 4 Pembagian Data dengan Direct Split 80:20

1: Bagi dataset menjadi 80% data pelatihan dan 20% sisanya sebagai data validasi dan uji.
2: Bagi sisa 20% tersebut secara merata menjadi 10% data validasi dan 10% data uji.
3: Gunakan stratifikasi label pada setiap tahap pembagian untuk menjaga distribusi kelas.

Skema ini membagi dataset secara langsung menjadi 80% data pelatihan dan sisa 20% dibagi merata menjadi validasi dan pengujian. Dengan distribusi

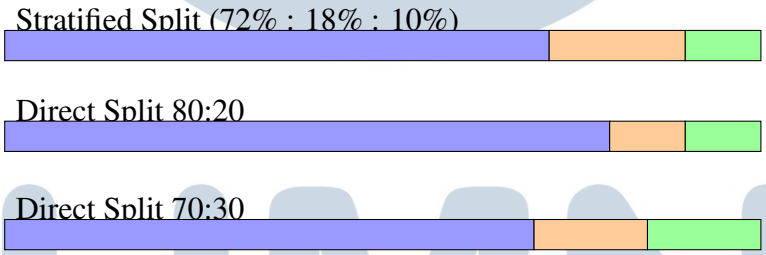
stratifikasi, skema ini cocok untuk evaluasi ketika model membutuhkan lebih banyak data pelatihan.

3.5.3 Direct Split 70:30 (Train : Val : Test = 70% : 15% : 15%)

Algorithm 5 Pembagian Data dengan Direct Split 70:30

1: Bagi dataset menjadi 70% data pelatihan dan 30% sisanya sebagai data validasi dan uji.
2: Bagi sisa 30% tersebut secara merata menjadi 15% data validasi dan 15% data uji.
3: Gunakan stratifikasi label pada setiap tahap pembagian untuk menjaga distribusi kelas.

Skema ini memberi porsi data pengujian dan validasi yang lebih besar (15%) untuk mengukur generalisasi model secara lebih kuat, meskipun harus mengorbankan sedikit data pelatihan.



Gambar 3.3. Ilustrasi visual proporsi pembagian data. Warna biru: pelatihan, oranye: validasi, hijau: pengujian.

Keterangan warna:

- **Pelatihan** – digunakan untuk melatih model.
- **Validasi** – mengevaluasi performa model selama pelatihan.
- **Pengujian** – mengukur performa akhir model.

Stratifikasi dipertahankan di seluruh strategi pemisahan untuk menjaga proporsi kelas biner (*Powdery_Mildew* dan *Not_Powdery_Mildew*) agar tetap seimbang. Ketiga skema ini digunakan untuk memastikan evaluasi model tidak bergantung pada satu konfigurasi data saja dan memberikan wawasan yang lebih menyeluruh tentang kemampuan generalisasi model.

3.6 Perancangan Arsitektur Model

Penelitian ini menggunakan dua pendekatan arsitektur Convolutional Neural Network (CNN) untuk mendeteksi penyakit *Powdery Mildew* pada daun labu, yaitu model CNN konvensional dan model berbasis VGG16. Model CNN konvensional dirancang khusus berdasarkan karakteristik dataset, sedangkan VGG16 digunakan sebagai pendekatan transfer learning dengan bobot pralatih dari ImageNet.

3.6.1 Model CNN Konvensional

Arsitektur model CNN konvensional terdiri dari empat blok konvolusi yang diikuti oleh bagian klasifikasi. Jumlah filter pada tiap blok meningkat secara progresif: 32, 64, 128, dan 256. Setiap blok terdiri dari konvolusi, normalisasi batch, dan max pooling.

Pseudocode: Arsitektur CNN Konvensional

Algorithm 6 Perancangan Arsitektur CNN Konvensional

1: Inisialisasi model sebagai *Sequential*
2: Tambahkan blok konvolusi sebanyak 4 kali:
3: **for** setiap blok **do**
4: Tambahkan `Conv2D` dengan kernel 3×3 dan `ReLU`
5: Tambahkan `BatchNormalization`
6: Tambahkan `MaxPooling2D` dengan ukuran 2×2
7: **end for**
8: Tambahkan `Flatten` untuk meratakan output
9: Tambahkan `Dense` dengan 256 unit dan aktivasi `ReLU`
10: Tambahkan `Dropout` sebesar 30%
11: Tambahkan `Dense` akhir dengan 2 unit dan aktivasi `Softmax`
12: Kompilasi model dengan:
- Optimizer: Adam, learning rate = 0.0001
- Loss: `sparse_categorical_crossentropy`
- Metrik: Akurasi

Arsitektur CNN dirancang dengan empat blok konvolusi yang terdiri dari lapisan `Conv2D`, `BatchNormalization`, dan `MaxPooling2D`. Jumlah filter meningkat secara progresif untuk menangkap fitur visual dari tingkat dasar hingga

kompleks. Setelah tahap ekstraksi fitur, digunakan lapisan `Flatten` dan `Dense` dengan dropout untuk membentuk klasifikasi akhir. Struktur ini bertujuan untuk menghasilkan model ringan yang tetap mampu mengenali pola visual khas penyakit *Powdery Mildew* secara efektif, terutama ketika dilatih dari awal tanpa bobot pralatih.

3.6.2 Model VGG16 dengan Fine-tuning

Model kedua dibangun menggunakan arsitektur VGG16 dengan bobot pralatih dari ImageNet. Lapisan klasifikasi default dihapus dan digantikan dengan klasifikasi kustom. Hanya 12 lapisan terakhir dari VGG16 yang diaktifkan untuk pelatihan (fine-tuning).

Pseudocode: Arsitektur VGG16 Hybrid

Algorithm 7 Perancangan Arsitektur VGG16 dengan Fine-tuning

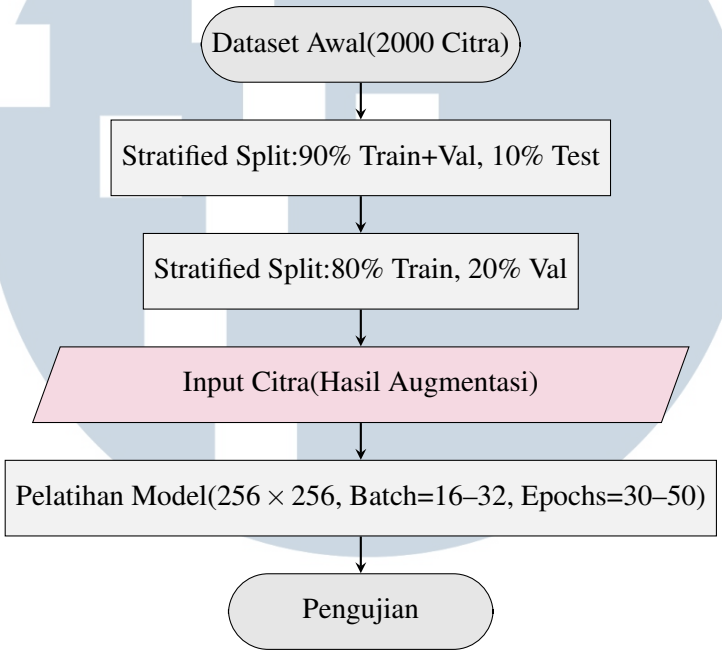
1: Muat model VGG16 dengan bobot ImageNet, tanpa top layer
2: Nonaktifkan semua lapisan kecuali 12 lapisan terakhir
3: Bangun model akhir sebagai *Sequential*:
- Tambahkan model dasar VGG16
- Tambahkan `Flatten`
- Tambahkan `Dense` dengan 256 unit dan `ReLU`
- Tambahkan `Dropout` sebesar 50%
- Tambahkan `Dense` akhir dengan 2 unit dan `Softmax`

4: Kompilasi model dengan konfigurasi identik dengan model CNN konvensional

Arsitektur VGG16 digunakan tanpa lapisan klasifikasi default, dengan hanya 12 lapisan terakhir yang diaktifkan untuk pelatihan ulang. Ini memungkinkan model mempertahankan fitur umum dari dataset ImageNet sambil menyesuaikan bagian akhir terhadap karakteristik domain daun labu. Penambahan lapisan klasifikasi baru disesuaikan dengan struktur yang setara dengan CNN konvensional, namun dengan dropout yang lebih tinggi untuk mengatasi risiko overfitting akibat kedalaman arsitektur. Strategi ini bertujuan menggabungkan keunggulan fitur pralatih dan adaptasi terhadap domain target.

3.7 Rangkaian Proses Pelatihan Model

Untuk memastikan pelatihan yang efisien dan evaluasi yang adil, dataset dibagi menggunakan pendekatan stratifikasi untuk mempertahankan proporsi kelas pada setiap subset. Diagram alur pada Gambar 3.4 menunjukkan keseluruhan proses mulai dari pembagian data, augmentasi, hingga evaluasi model.



Gambar 3.4. Diagram alur skema pembagian data, augmentasi, pelatihan, dan evaluasi model dalam penelitian ini.

Sebanyak 10% dari dataset awal akan disisihkan terlebih dahulu sebagai data uji untuk memastikan bahwa data tersebut tidak tersentuh selama proses pelatihan maupun validasi. Sisanya, yaitu 90% dari dataset, kemudian akan dibagi lebih lanjut secara stratifikasi dengan rasio 80:20 untuk mendapatkan data pelatihan dan validasi. Dengan demikian, komposisi akhir dataset menjadi 72% untuk pelatihan, 18% untuk validasi, dan 10% untuk pengujian.

Augmentasi citra direncanakan secara *offline*, yaitu sebelum proses pelatihan dimulai. Pendekatan ini bertujuan untuk memperkaya variasi data latih tanpa membebani proses pelatihan dengan augmentasi dinamis selama epoch. Teknik augmentasi yang digunakan mencakup rotasi acak, translasi, flipping horizontal, zoom, dan perubahan tingkat kecerahan (*brightness*). Semua citra hasil augmentasi dan non-augmentasi akan disesuaikan ke dalam ukuran 256×256 piksel.

Model akan dilatih dengan ukuran *batch* yang divariasikan antara 16

hingga 32 dan jumlah *epoch* antara 30 hingga 50, tergantung pada stabilitas performa selama proses pelatihan. Selain itu, *learning rate* awal yang digunakan direncanakan berada pada rentang 1×10^{-5} hingga 1×10^{-3}, dengan penyesuaian adaptif selama pelatihan menggunakan *callback* ReduceLROnPlateau.

Sebelum memulai pelatihan, beberapa fungsi *callback* akan dikonfigurasi untuk mengontrol proses pelatihan secara otomatis:

Inisialisasi ModelCheckpoint: Simpan model terbaik berdasarkan val_accuracy

Inisialisasi EarlyStopping: Monitor val_loss Patience = 8 Pulihkan bobot terbaik

Inisialisasi ReduceLROnPlateau: Monitor val_loss Patience = 4 Factor = 0.2 Tampilkan informasi saat learning rate dikurangi

- **ModelCheckpoint**: menyimpan model dengan nilai `val_accuracy` tertinggi.
- **EarlyStopping**: menghentikan pelatihan dini jika tidak ada peningkatan nilai validasi selama 8 epoch.
- **ReduceLROnPlateau**: mengurangi `learning rate` sebanyak 0,2 kali lipat jika validasi tidak membaik dalam 4 epoch berturut-turut.

Proses pelatihan model dilakukan dengan fungsi pelatihan utama sebagai berikut:

Latih model menggunakan: - Data latih (X_train, y_train) - Data validasi (X_val, y_val) - Batch size = 16–32 - Epochs = 30–50 - Class weighting aktif - Callbacks aktif (checkpoint, earlystop, reduce_lr)

- `batch_size`: ukuran batch mini yang divariasikan antara 16 dan 32.
- `epochs`: jumlah maksimum epoch, direncanakan antara 30 hingga 50.
- `validation_data`: tuple data validasi untuk memantau performa tiap epoch.
- `class_weight`: bobot kelas untuk mengatasi ketidakseimbangan kelas.
- `callbacks`: daftar fungsi pengontrol pelatihan.

Evaluasi kinerja model akan dilakukan terhadap data uji dengan menggunakan metrik akurasi, precision, recall, dan F1-score. Karena terdapat ketidakseimbangan antara jumlah kelas *Powdery Mildew* dan kelas lainnya, skema

class weighting akan diterapkan untuk memberikan bobot lebih besar pada kelas minoritas dan menghindari bias terhadap kelas mayoritas.

Seluruh proses pelatihan akan dilaksanakan di lingkungan Jupyter Notebook menggunakan framework TensorFlow 2.14 dan Keras, pada perangkat keras yang dijelaskan lebih lanjut pada Bab berikutnya.

3.8 Visualisasi dan Evaluasi Model

Untuk mengevaluasi kinerja model dan memahami dinamika pelatihan, dilakukan visualisasi terhadap metrik pelatihan serta analisis hasil klasifikasi pada data uji. Pendekatan ini bertujuan untuk memberikan gambaran menyeluruh terhadap proses pelatihan dan efektivitas model dalam mendeteksi penyakit *Powdery Mildew*.

3.8.1 Visualisasi Kurva Pelatihan

Selama proses pelatihan, dilakukan pencatatan nilai akurasi dan loss pada data pelatihan dan validasi untuk setiap epoch. Visualisasi dua kurva berikut digunakan untuk mengevaluasi konvergensi dan mendeteksi potensi *overfitting*:

- **Kurva Akurasi:** Menampilkan perbandingan antara *training accuracy* dan *validation accuracy* terhadap jumlah epoch.
- **Kurva Loss:** Menampilkan *training loss* dan *validation loss* terhadap jumlah epoch.

Kedua kurva ini digunakan untuk memantau stabilitas pelatihan, mendeteksi gejala *underfitting* atau *overfitting*, serta menilai efektivitas penggunaan callback seperti *EarlyStopping* dan *ReduceLROnPlateau*. Kurva ini juga membantu mengidentifikasi kapan pelatihan dihentikan lebih awal akibat tidak adanya peningkatan metrik validasi, serta kapan penurunan *learning rate* terjadi selama pelatihan.

3.8.2 Evaluasi Kinerja Model

Setelah pelatihan selesai, model dievaluasi menggunakan data uji yang telah dipisahkan sebelumnya. Evaluasi dilakukan dengan dua pendekatan utama:

- **Confusion Matrix:** Digunakan untuk memvisualisasikan performa klasifikasi model dalam bentuk matriks 2×2 antara label sebenarnya dan hasil prediksi. Matriks ini memudahkan identifikasi jumlah *true positives*, *false positives*, *true negatives*, dan *false negatives*.
- **Classification Report:** Laporan ini mencakup metrik evaluasi berupa *accuracy*, *precision*, *recall*, dan *F1-score* untuk masing-masing kelas, serta nilai rata-ratanya secara makro dan mikro.

Pemilihan metrik evaluasi dilakukan dengan mempertimbangkan karakteristik dataset yang tidak seimbang serta pentingnya deteksi yang akurat terhadap penyakit. Selain akurasi sebagai metrik umum, digunakan pula metrik tambahan seperti *precision*, *recall*, dan *F1-score* untuk memberikan gambaran yang lebih menyeluruh mengenai performa model. Dalam konteks deteksi penyakit seperti *Powdery Mildew*, nilai *recall* menjadi penting karena berkaitan dengan kemampuan model untuk mendeteksi kasus yang benar-benar positif, sehingga kegagalan deteksi (*false negative*) dapat diminimalkan.

Selain itu, penelitian ini juga membandingkan performa dua arsitektur model, yaitu CNN konvensional dan model VGG16 dengan fine-tuning. Perbandingan dilakukan berdasarkan hasil pelatihan dan evaluasi dari masing-masing model terhadap data uji yang sama. Untuk menguji konsistensi performa model terhadap variasi skema pembagian data, eksperimen dilakukan dengan beberapa rasio split seperti 90:10, 80:20, dan 70:30. Pada setiap skenario, model dijalankan beberapa kali, dan hasil terbaik dari masing-masing model—berdasarkan akurasi atau F1-score—digunakan sebagai dasar pembandingan. Hasil dari setiap konfigurasi kemudian dianalisis menggunakan metrik evaluasi yang telah dijelaskan sebelumnya.

Seluruh proses evaluasi dilakukan menggunakan fungsi evaluasi dari pustaka `scikit-learn`, serta divisualisasikan menggunakan pustaka `matplotlib` dan `seaborn` dalam lingkungan Jupyter Notebook.