KLASIFIKASI PENYAKIT PADA TANAMAN PADI MELALUI CITRA DAUN MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* BERBASIS *MOBILENETV2*

SKRIPSI

Disusun Sebagai Salah Satu Syarat
untuk Memperoleh Gelar Sarjana Komputer
pada Program Studi Informatika

Disusun oleh:
Sabdiel Tarigan
24060120120028

DEPARTEMEN INFORMATIKA
FAKULTAS SAINS DAN MATEMATIKA
UNIVERSITAS DIPONEGORO
2024

HALAMAN PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan di bawah ini :

Nama : Sabdiel Tarigan

NIM : 24060120120028

Judul : Klasifikasi Penyakit pada Tanaman Padi Melalui Citra Daun Menggunakan Algoritma CNN Berbasis *MobileNetV2*

Dengan ini saya menyatakan bahwa dalam skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan di dalam daftar pustaka.

Semarang, ...

Materai
Rp. 10.000,-

Sabdiel Tarigan

24060120120028

HALAMAN PENGESAHAN

Judul : Klasifikasi Penyakit pada Tanaman Padi Melalui Citra Daun Menggunakan Algoritma CNN Berbasis *MobileNetV2*

Nama : Sabdiel Tarigan

NIM : 24060120120028

Telah diujikan pada sidang tugas akhir dan dinyatakan lulus pada tanggal ……

Semarang, ...

Pembimbing 1,	Pembimbing 2,
Drs. Eko Adi Sarwoko M.Komp.	Khadijah, S.Kom., M.Cs.
NIP. 19651107 199203 1 003	NIP. 19890303 201504 2 002

Mengetahui,

Ketua	Ketua
Departemen Informatika	Panitia Penguji Tugas Akhir
Dr. Aris Sugiharto, S.Si., M.Kom.	Ragil Saputra, S.Si., M.Cs.
NIP. 19710811 199902 1 004	NIP. 19801021 200501 1003

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini dengan judul Klasifikasi Penyakit pada Tanaman Padi Menggunakan Algoritma *CNN* Berbasis *MobileNetV2*.

Laporan skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer di Departemen Informatika, Fakultas Sains dan Matematika, Universitas Diponegoro. Penulis menyadari bahwa laporan skripsi ini sulit untuk diselesaikan tanpa adanya bantuan, dukungan, dan bimbingan dari berbagai pihak. Oleh karena itu, dengan rendah hati, penulis mengucapkan terima kasih kepada :

1. Prof. Dr. Kusworo Adi, S.Si., M.T., selaku Dekan Fakultas Sains dan Matematika Universitas Diponegoro.
2. Dr. Aris Sugiharto, S.Si., M.Kom. selaku Ketua Departemen Informatika
3. Dr. Aris Puji Widodo, S.Si., M.T. selaku Dosen Wali Penulis.
4. Nurdin Bahtiar, S.Si., M.T., selaku Koordinator Skripsi di Departemen Informatika.
5. Drs. Eko Adi Sarwoko M.Komp. dan Khadijah, S.Kom., M.Cs., selaku dosen pembimbing yang telah membimbing dan mengarahkan penulis selama kegiatan skripsi berlangsung.
6. Kedua orang tua dan keluarga yang telah memberikan dukungan moral, materil, dan senantiasa mendoakan penulis selama kegiatan skripsi berlangsung.

Penulis menyadari bahwa laporan skripsi ini masih memiliki banyak kekurangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat membangun. Semoga laporan skripsi ini dapat menjadi manfaat bagi penulis dan juga para pembaca.

Semarang, ...

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai civitas akademik Universitas Diponegoro, saya yang bertanda tangan di bawah ini:

Nama	:	Sabdiel Tarigan
NIM	:	24060120120028
Program Studi	:	Informatika
Departemen	:	Informatika
Fakultas	:	Sains dan Matematika
Jenis Karya	:	Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Right)** kepada Universitas Diponegoro atas karya ilmiah saya yang berjudul:

Klasifikasi Penyakit Pada Tanaman Padi Melalui Citra Daun Menggunakan Convolutional Neural Network Berbasis MobileNetV2

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-eksklusif ini Universitas Diponegoro berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Semarang, ...

Materai
Rp. 10.000,-

Sabdiel Tarigan

24060120120028

ABSTRAK

Penyakit pada tanaman padi merupakan tantangan utama dalam sektor pertanian Indonesia yang dapat mengganggu kestabilan hasil panen. Deteksi penyakit secara tradisional membutuhkan waktu dan sumber daya yang signifikan. Penelitian ini mengembangkan model klasifikasi penyakit pada tanaman padi melalui citra daun menggunakan *Convolutional Neural Network* (CNN) berbasis arsitektur *MobileNetV2*. Teknologi ini menawarkan solusi untuk meningkatkan akurasi dan efisiensi dalam identifikasi penyakit tanaman padi. *MobileNetV2* dipilih karena kemampuannya dalam mengurangi kebutuhan komputasi dan memori, serta efektivitasnya yang telah terbukti dalam berbagai tugas klasifikasi citra dengan tingkat akurasi yang tinggi. *Dataset* yang digunakan terdiri dari citra daun padi dengan enam kelas penyakit, termasuk *blast*, *blight*, *tungro*, dan *brownspot*. Proses klasifikasi mencakup *preprocessing* citra, yaitu resizing, rotasi, dan *flipping*. *Dataset* dibagi menjadi 3 yaitu, data latih, data validasi, dan data uji. Hasil penelitian menunjukkan bahwa model CNN berbasis *MobileNetV2* mencapai akurasi dan *f1-score* hingga 99%, membuktikan efektivitasnya dalam klasifikasi penyakit pada tanaman padi. Diharapkan, hasil penelitian ini dapat berkontribusi dalam peningkatan produktivitas pertanian dan ketahanan pangan di Indonesia..

Kata Kunci : *Convolutional Neural Network (CNN),* deteksi penyakit tanaman padi, klasifikasi citra, *MobileNetV2*.

ABSTRACT

Diseases in rice plants are a major challenge in Indonesia's agricultural sector that can disrupt crop yield stability. Traditional disease detection requires significant time and resources. In this study, we developed a disease classification model for rice plants through leaf images using a Convolutional Neural Network (CNN) based on the MobileNetV2 architecture. This technology offers a solution to enhance accuracy and efficiency in identifying rice plant diseases. MobileNetV2 was chosen for its ability to reduce computational and memory requirements and its proven effectiveness in various image classification tasks with high accuracy. The dataset used consists of leaf images of rice plants with six disease classes, including blast, blight, tungro, and brownspot. The classification process includes image preprocessing, such as resizing, rotation, and flipping. The study results show that the CNN model based on MobileNetV2 achieved an accuracy and f1-score of up to 99%, demonstrating its effectiveness in classifying diseases in rice plants. It is hoped that the results of this study can contribute to increased agricultural productivity and food security in Indonesia.

Keywords ***:*** *Convolutional Neural Network (CNN), Rice plant disease detection, Image classification, MobileNetV2.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR PERSAMAAN

DAFTAR GAMBAR

DAFTAR SOURCE CODE

BAB I
PENDAHULUAN

Bab ini menyajikan latar belakang, rumusan masalah, tujuan dan manfaat, ruang lingkup, serta sistematika penulisan dalam penelitian ini yang berfokus kepada klasifikasi penyakit pada tanaman padi menggunakan algoritma CNN berbasis *MobileNetV2*.

1.1 Latar Belakang

Teknologi dapat berperan dalam menjaga kestabilan jumlah panen petani di Indonesia, hal ini juga didukung oleh sejumlah studi. Komalasari (2021) menekankan perlunya teknologi canggih yang dapat diterapkan pada praktik pertanian. Hal ini karena untuk mendeteksi penyakit padi membutuhkan waktu yang banyak. Deteksi penyakit padi dilakukan dengan memberikan sampel tanaman yang terkena penyakit kepada tenaga ahli, dari sampel tersebut tenaga ahli akan mengidentifikasi penyakit tersebut. Hal ini tentu akan memakan banyak waktu jika terimplementasikan pada pertanian besar. Penggunaan metode deteksi penyakit tanaman padi berbasis teknologi dapat membantu dalam mempercepat dan meningkatkan akurasi proses identifikasi penyakit tanaman padi secara efisien (Putra dkk, 2023).

Beberapa penelititan terkait klasifikasi penyakit pada tanaman padi telah banyak dilakukan. Menurut penelitian yang telah dilakukan oleh Ardiansyah & Nugroho (2023), *MobileNetV2* telah terbukti efektif dalam mengklasifikasikan penyakit pada tanaman kopi, dengan mencapai tingkat akurasi sebesar 99%. Arsitektur ini juga telah berhasil digunakan dalam mengklasifikasikan penyakit pada tanaman kentang melalui citra daunnya, dengan mencapai tingkat keberhasilan sebesar 97% (Ompusunggu, 2022). Penelitian Annur dkk. (2023) dalam mengklasifikasikan tingkat keparahan penyakit leaf*blast* pada tanaman padi mencapai tingkat akurasi sebesar 78%. Temuan dari penelitian-penelitian ini secara keseluruhan menunjukkan bahwa *MobileNetV2* merupakan pilihan yang sesuai untuk deteksi penyakit pada tanaman.

Penelitian ini bermaksud untuk memberikan kontribusi yang signifikan dalam pengembangan teknologi deteksi penyakit pada tanaman padi dengan menerapkan *Convolutional neural network* (CNN) berbasis arsitektur *MobileNetV2*. Metode

tersebut dipilih karena kemampuan CNN dalam mengekstrak fitur dari citra dan efisiensi *MobileNetV2* dalam penggunaan sumber daya komputasi. Dengan demikian, harapannya penelitian ini akan menghasilkan sistem yang efektif, efisien, dan dapat memberikan kontribusi positif dalam meningkatkan produktivitas pertanian serta ketahanan pangan.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang, maka dapat dirumuskan suatu permasalahan, yaitu bagaimana cara mengklasifikasikan penyakit pada tanaman padi melalui citra daun menggunakan algoritma CNN berbasis *MobileNetV2*?

1.3 Tujuan dan Manfaat

Tujuan dari penelitian ini adalah menghasilkan model klasifikasi penyakit pada tanaman padi yang akurat melalui citra daun menggunakan algoritma CNN berbasis *MobileNetV2*. Manfaat dari penelitian ini adalah memberikan metode baru bagi para peneliti dalam mengembangkan model klasifikasi penyakit pada tanaman padi. Selain itu, penelitian ini dapat menjadi pedoman pengetahuan bagi para peneliti tentang penyakit tanaman padi dan klasifikasi penyakit pada tanaman padi melalui citra daun menggunakan algoritma CNN berbasis *MobileNetV2*.

1.4 Ruang Lingkup

Batasan-batasan terhadap pembahasan dalam laporan skripsi ini diperlukan agar pembahasan menjadi lebih terarah dan tidak melebihi rumusan masalah dari penelitian ini. Batasan-batasan tersebut dapat dilihat di bawah ini :

1. *Dataset* yang digunakan dalam penelitian ini adalah kumpulan data (*dataset*) citra daun tanaman padi yang terkena penyakit *blast*, *blight*, *tungro*, dan *brown spot* yang diperoleh dari *Kaggle*.
2. Format *file* citra daun dalam *dataset* berupa *Joint Photographic Experts Group* (JPEG). Terdapat empat subfolder dalam folder *dataset* yang masing-masingnya menyimpan kumpulan gambar citra daun padi dengan penyakit *blast*, *blight*, *tungro*, dan *brown spot*.

3. Metrik evaluasi yang digunakan untuk mengevaluasi performa model CNN berbasis *MobileNetV2* adalah skor akurasi (*accuracy score*) dan *f1-score*.

1.5 Sistematika Penulisan

Sistematika penulisan disusun seperti di bawah ini, sebagai gambaran mengenai pembahasan-pembahasan dalam setiap bab yang ada dalam laporan skripsi ini :

BAB I PENDAHULUAN

Bab ini menyajikan latar belakang, rumusan masalah, tujuan dan manfaat, ruang lingkup, serta sistematika penulisan dalam penelitian ini yang berfokus kepada klasifikasi penyakit pada tanaman padi melalui citra daun menggunakan algoritma CNN berbasis *MobileNetV2*.

BAB II TINJAUAN PUSTAKA

Bab ini menyajikan landasan teori yang diimplementasikan dalam penelitian ini, yaitu kumpulan teori terkait klasifikasi penyakit pada tanaman padi melalui citra daun menggunakan algoritma CNN berbasis *MobileNetV2*.

BAB III METODE PENELITIAN

Bab ini menyajikan metode penelitian yang diterapkan dalam penelitian ini, yaitu metode klasifikasi penyakit pada tanaman padi melalui citra daun menggunakan algoritma CNN berbasis *MobileNetV2*.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisikan pembahasan mengenai hasil implementasi metode penelitian yang telah diterapkan dalam penelitian ini, yaitu klasifikasi penyakit pada tanaman padi melalui citra daun menggunakan algoritma CNN berbasis *MobileNetV2*.

BAB V KESIMPULAN DAN SARAN

Bab ini menyajikan kesimpulan dan saran terkait hasil implementasi metode penelitian yang telah diterapkan dalam penelitian ini, yaitu klasifikasi penyakit pada tanaman padi melalui citra daun menggunakan algoritma CNN berbasis *MobileNetV2*.

BAB II
TINJAUAN PUSTAKA

Bab ini menyajikan landasan teori yang diimplementasikan dalam penelitian ini, yaitu kumpulan teori terkait klasifikasi penyakit pada tanaman padi melalui citra daun menggunakan algoritma CNN berbasis *MobileNetV2*.

2.1 Penelitian Sebelumnya

Penyusunan laporan tugas akhir ini menggunakan beberapa referensi penelitian sebelumnya berupa jurnal-jurnal yang berhubungan dengan penelitian ini. Penelitian sebelumnya yang berkaitan dengan penelitian ini dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Terkait

Nama (Tahun)	Kasus Penelitian	Prosedur	Algoritma	Hasil
Ardiansyah & Nugroho (2023)	Klasifikasi penyakit daun kopi	*MobileNetV2* menggunakan proses pelatihan yang melibatkan pembelajaran berbasis data untuk mengidentifikasi pola visual dalam gambar daun kopi dan membedakan antar jenis penyakit daun kopi.	*MobileNet V2*	*MobileNetV2* menghasilkan akurasi rata-rata 0,99 dari 3 pengujian.
Annur dkk. (2023)	Klasifikasi tingkat keparahan penyakit *leafblast* tanaman Padi	*MobileNetV2* secara efisien menganalisis gambar daun padi yang terinfeksi *leafblast* untuk mengklasifikasi-kan tingkat keparahannya.	*MobileNet V2*	*MobileNetV2* menghasilkan akurasi sebesar 0,78.

Nama (Tahun)	Kasus Penelitian	Prosedur	Algoritma	Hasil
Purnama (2020)	Deteksi tanaman herbal	*MobileNetV2* menggunakan proses pelatihan yang melibatkan pembelajaran berbasis data untuk mengidentifikasi pola visual dalam gambar daun tanaman untuk membedakan tanaman herbal dan tidak herbal.	*MobileNetV2*	*MobileNetV2* menghasilkan akurasi sebesar 0,87.
Khaira dkk. (2024)	Deteksi penyakit tanaman jagung	*MobileNetV2* menggunakan proses pelatihan yang melibatkan pembelajaran berbasis data untuk mengidentifikasi pola visual dalam gambar daun kopi dan membedakan antar jenis penyakit tanaman jagung.	*MobileNetV2*	*MobileNetV2* menghasilkan akurasi sebesar 0,99.

Berdasarkan Tabel 2.1, dapat ditarik beberapa kesimpulan yang dapat mendukung penelitian ini tentang klasifikasi penyakit pada tanaman padi menggunakan *MobileNetV2*. Ardiansyah & Nugroho (2023) dan Annur dkk. (2023) menunjukkan bahwa *MobileNetV2* mampu mengidentifikasi dan mengklasifikasikan penyakit pada daun kopi dan tanaman padi dengan akurasi yang cukup tinggi, yaitu 0.99 pada daun kopi dan 0.78 pada penyakit leaf*blast* padi. Ini menunjukkan bahwa algoritma *MobileNetV2* cukup efektif dalam menganalisis gambar daun tanaman yang terinfeksi penyakit, mengidentifikasi pola visual, dan membedakan antar jenis penyakit atau tingkat keparahan. Selain itu, *MobileNetV2* dapat diaplikasikan pada berbagai jenis tanaman dan penyakit, seperti pada daun kopi, padi, dan tanaman herbal dengan akurasi 0.87, menunjukkan sifat generalitas dan adaptabilitas yang baik. Prosedur yang digunakan dalam penelitian tersebut konsisten dalam menggunakan *MobileNetV2*

untuk melatih model berdasarkan data pembelajaran, mengidentifikasi pola visual, dan melakukan klasifikasi.

2.2 Tanaman Padi

2.2.1 Pengertian Tanaman Padi

Tanaman padi merupakan bahan kebutuhan pokok masyarakat khususnya di Indonesia, petani padi di Indonesia berusaha agar mendapatkan hasil panen padi yang baik dengan banyak cara diantaranya penanggulangan hama dan penyakit, penanggulangan penyakit dan hama dilakukan dengaan cara dianosis, akan tetapi banyak petani yang kesulitan dalam mendiagnosis gejala-gejala yang timbul akibat penyakit dan hama, hal tersebut diakibatkan kurangnya pengetahuan beberapa petani dalam penanggulangan dini terhadap penyakit yang menyerang, maka dari itu diperlukan adanya seorang ahli atau pakar dalam hal tersebut (Asep, 2016).

2.2.2 Penyakit pada Tanaman Padi

Penyakit pada tanaman padi yang dapat dideteksi melalui citra daun adalah *blast, blight*, *tungro*, dan *brown spot*. Penjelasan untuk masing-masing penyakit tanaman padi tersebut dapat dilihat di bawah ini (Dwi dkk, 2023) :

1. *Blast*

   Penyakit pada tanaman padi yang disebabkan oleh jamur *Pyricularia oryzae*. Penyakit ini biasanya mempengaruhi daun, malai, atau batang tanaman padi. Gejalanya termasuk munculnya bercak berwarna putih, coklat, atau hitam pada daun, yang kemudian berkembang menjadi bercak berbentuk oval atau panjang. *Blast* dapat menyebabkan penurunan hasil panen yang signifikan jika tidak dikendalikan dengan baik .

2. *Blight*

   Penyakit pada tanaman padi yang disebabkan oleh jamur *Magnaporthe grisea*. Gejalanya meliputi daun yang berubah warna menjadi kuning, kemudian menjadi coklat atau hitam, seringkali dengan tepi daun yang berwarna hijau. Penyakit ini dapat menyebar dengan cepat melalui udara dan tetesan air hujan, dan jika tidak dikelola dengan efektif, dapat menyebabkan kerugian yang besar dalam produksi padi

3. *Tungro*

   Penyakit pada tanaman padi yang disebabkan oleh dua virus, yaitu *Rice Tungro Bacilliform Virus* (RTBV) dan *Rice Tungro Spherical Virus* (RTSV), yang ditularkan oleh serangga wereng coklat. Gejalanya termasuk daun yang menguning, stunting tanaman, dan hingga kematian tanaman. Penyakit ini dapat mengurangi hasil panen secara signifikan dan seringkali menjadi masalah yang serius dalam pertanian padi di berbagai wilayah.

4. *Brown Spot*

   Penyakit pada tanaman padi yang disebabkan oleh jamur *Cochliobolus miyabeanus*. Gejala penyakit ini umumnya terlihat sebagai bercak-bercak berwarna coklat pada daun padi, terutama pada bagian atas daun. Tanaman yang terinfeksi biasanya mengalami penurunan dalam pertumbuhan dan hasil panen. Penyakit ini dapat menyebabkan kerugian ekonomi yang signifikan jika tidak dikelola dengan tepat.

2.3 Jaringan Konvolusional

2.3.1 Deskripsi Jaringan Konvolusional

Jaringan konvolusional (disebut juga jaringan saraf konvolusional atau CNN) adalah jenis jaringan saraf yang khusus dirancang untuk memproses data dengan topologi yang teratur (Goodfellow dkk, 2016). Konvolusi adalah jenis operasi linear yang khusus. Konvolusi adalah jenis operasi linear yang khusus digunakan untuk mengolah data spasial atau temporal, seperti gambar atau sinyal, dengan cara menggabungkan informasi dari lingkungan sekitar setiap elemen data menggunakan kernel atau filter tertentu yang bergerak melintasi data input. Jaringan konvolusional sebenarnya adalah jaringan saraf yang menggunakan konvolusi sebagai pengganti perkalian matriks umum setidaknya pada salah satu lapisannya.

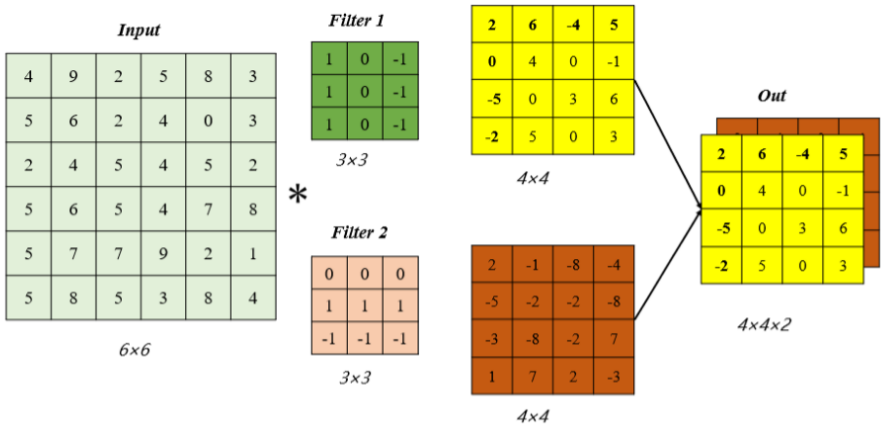
2.3.2 *Layer* Konvolusional

Lapisan konvolusional dalam jaringan saraf konvolusional (CNN) merupakan lapisan yang penting dalam pemrosesan gambar (citra). Citra direpresentasikan oleh matriks dengan informasi warna berupa kode warna *Red-Green-Blue* (RGB) serta

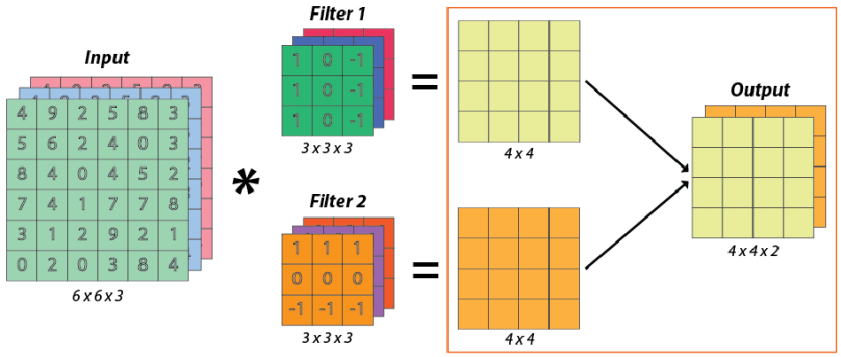
berdimensi $h \times w \times d$, di mana h dan w adalah tinggi dan lebar gambar, sedangkan d adalah kedalaman *channel* warna (dalam kasus RGB, $d = 3$).

Lapisan konvolusional menggunakan kernel atau filter sebagai komponen utama untuk melakukan operasi konvolusi. Misalkan K adalah sebuah kernel dengan x baris, y kolom, dan kedalaman d. Kernel dengan ukuran (Kx × Ky × d) bekerja pada area reseptif (Kx × Ky) pada gambar. Tinggi dan lebar kernel lebih kecil daripada tinggi dan lebar gambar input. Kernel "meluncur" atau bergerak (berkonvolusi) melintasi gambar, menghasilkan peta fitur. Konvolusi merupakan proses penghitungan di mana elemen-elemen dari kernel dikalikan secara elemen-ke-elemen dengan area yang sesuai pada gambar input, kemudian hasil perkalian tersebut dijumlahkan untuk menghasilkan satu nilai output pada peta fitur. Penting untuk dicatat bahwa kedalaman d dari kernel harus sama dengan kedalaman inputnya. Oleh karena itu, kedalaman tersebut dapat bervariasi di dalam jaringan. Biasanya, kedalaman gambar adalah jumlah channel warna, yaitu tiga channel RGB. Namun, pada gambar dengan channel warna HS (*hue-saturation),* kedalaman tersebut merupakan jumlah pita spektral. Ilustrasi konvolusi dengan input 6x6 (2 dimensi) serta menggunakan 2 kernel berukuran 3x3 dapat dilihat pada Gambar 2.1.

Langkah kernel (*kernel stride*) adalah parameter yang harus ditentukan sebelum pelatihan lapisan konvolusional. Langkah ini menentukan jumlah piksel yang digunakan oleh kernel untuk berpindah setiap kali melintasi gambar. Salah satu kekurangan penggunaan lapisan konvolusional adalah mengurangi ukuran peta fitur keluaran yang disebabkan oleh dua faktor utama, yaitu ukuran kernel dan langkah kernel (stride). Ketika kernel "meluncur" melintasi gambar input, ia hanya dapat mencakup sejumlah area tertentu berdasarkan dimensinya. Setiap pergerakan kernel (sesuai dengan langkah kernel) menghasilkan nilai tunggal pada peta fitur keluaran, yang merupakan hasil dari operasi konvolusi.. Semakin besar langkah kernel, maka ukuran peta keluaran akan menjadi lebih kecil. Ilustrasi konvolusi dengan Input 6 x 6 x 6 yang menggunakan 2 filter atau kernel berukuran 3x3 dapat dilihat pada Gambar 2.2.



Gambar 2.1 Ilustrasi Konvolusi dengan input 6x6 (2 dimensi) serta Menggunakan 2 Kernel Berukuran 3x3 (Goodfellow, 2016)

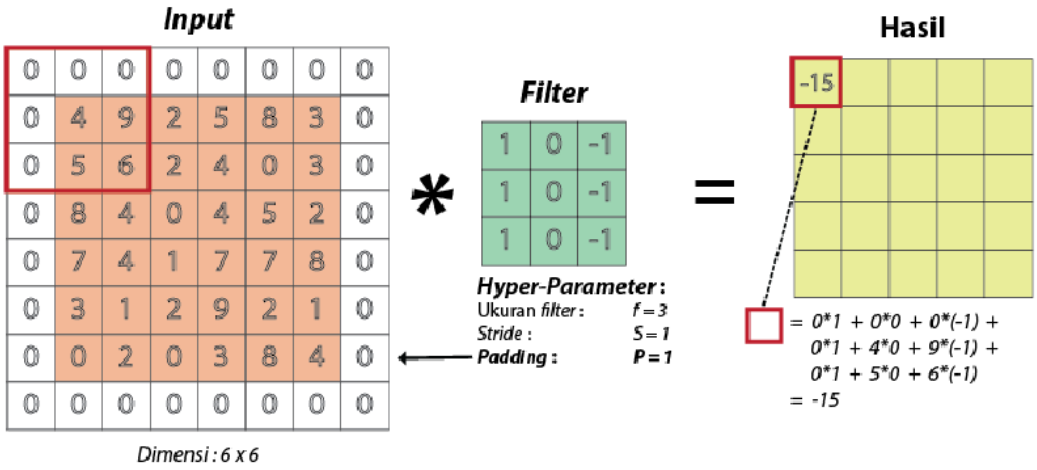


Gambar 2.2 Ilustrasi Konvolusi dengan Input 6x6x6 yang menggunakan dua Filter atau Kernel berukuran 3x3 (Goodfellow, 2016)

2.3.3 *Padding* dan *Stride*

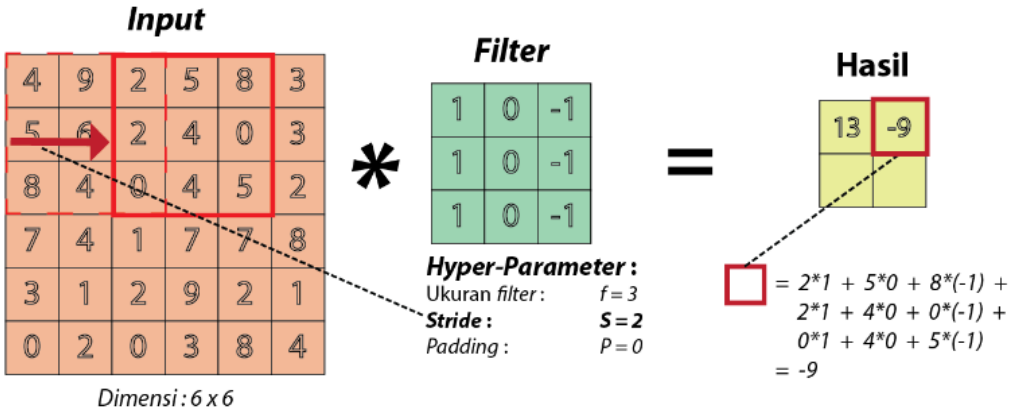
Padding merupakan metode yang diterapkan dalam pengolahan gambar, khususnya pada operasi konvolusi, dengan tujuan untuk mempertahankan informasi di sekitar tepi atau batas gambar sekaligus mencegah penyusutan dimensi gambar. Dalam konteks ini, padding dilakukan dengan menambahkan nilai di luar batas gambar. Zero-padding adalah salah satu jenis padding yang umum digunakan. Pada *zero-padding*, nilai-nol ditambahkan di sekitar tepi gambar. Hal ini dilakukan untuk memperluas ukuran gambar sehingga tepi gambar juga dapat tercakup dalam operasi konvolusi. Dengan menggunakan *zero-padding*, ukuran peta fitur output dari *layer* konvolusi akan memiliki ukuran yang sama dengan ukuran peta fitur input. Dalam *library* Keras, fungsi *layer* konvolusi secara default menggunakan *padding* "*same*" sebagai

parameter. *Padding* "*same*" akan menambahkan nilai-nol di sekeliling input sehingga menghasilkan ukuran peta fitur yang sama sebagai output *layer* konvolusi. Namun, parameter *padding* ini dapat diubah menjadi "valid" untuk melaksanakan operasi konvolusi tanpa menggunakan *padding*. Selain itu, *library* Keras juga menyediakan *layer Zero-padding* tersendiri yang memungkinkan pengaturan tepi mana yang akan diberi *padding*. Dengan demikian, penggunaan *padding* dalam pengolahan gambar, seperti *zero-padding*, dapat membantu mempertahankan informasi pada tepi gambar dan menghasilkan ukuran output yang diinginkan dalam operasi konvolusi (Snuverink, 2017). Konvolusi citra dengan menggunakan *zero-padding* dapat dilihat pada Gambar 2.3.



Gambar 2.3 Konvolusi Citra dengan Menggunakan *Zero-Padding* (Goodfellow, 2016)

Stride adalah sebuah parameter dalam *layer* konvolusional yang harus didefinisikan sebelum pelatihan dilakukan. *Stride* adalah jumlah piksel yang diperlukan kernel untuk berpindah pada satu waktu yang diilustrasikan oleh Gambar 2.4. *Stride* yang semakin besar akan menghasilkan output yang berukuran semakin kecil (Snuverink, 2017). Semakin besar nilai *stride* semakin banyak piksel yang dilewati oleh kernel, sehingga menghasilkan output yang berukuran lebih kecil.



Gambar 2.4 Konvolusi Citra dengan Menggunakan 2 *Stride* (Goodfellow, 2016)

2.3.4 Operasi Konvolusi

Operasi konvolusi adalah suatu operasi matematis yang melibatkan dua fungsi argumen bernilai nyata. Dalam konteks pembelajaran mesin, algoritma pembelajaran akan mempelajari nilai-nilai yang tepat dari kernel (filter) di tempat yang tepat. Dalam hal ini, kernel tidak perlu dibalik (*flipping*) seperti yang umumnya terjadi dalam operasi konvolusi pada pengolahan sinyal (Goodfellow dkk., 2016). Dengan mempelajari kernel, algoritma konvolusi dapat mengekstraksi fitur yang relevan dari data input, membantu dalam pemrosesan informasi visual, dan meningkatkan kemampuan model untuk memahami dan menggeneralisasi pola-pola yang kompleks dalam data.

Persamaan 2.1 digunakan untuk menentukan ukuran output dari *layer* konvolusi. Persamaan 2.2 menjelaskan detail bagaimana setiap nilai dalam output dihitung dari nilai input menggunakan filter konvolusi.. Ketika x_{out} dan y_{out} diketahui, x_{in} dan y_{in} dapat diperoleh dengan mengetahui hubungan antara ukuran *output* n_{out}, ukuran *input* n_{in}, ukuran *kernel* f, *stride* S, dan *padding* P. Ketika jumlah kedalaman *channel* d lebih dari satu, hasil konvolusi $I_{1\ldots n}$ dan $K_{1\ldots n}$ diselesaikan dengan menjumlahkan seluruh $O_{1\ldots n}(x_{out}, y_{out})$ sehingga menghasilkan $O_{total}(x_{out}, y_{out})$.

$$n_{out} = \frac{n_{in} - f + 2P}{S} + 1 \quad \text{..... (2.1)}$$

$$O(X_{out}, y_{out}) = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} I\,(X_{in} + a, y_{in} + b) \times K(a, b) \quad \text{..... (2.2)}$$

Keterangan :

n_{out} = Dimensi keluaran (panjang/lebar) fitur output

n_{in} = Dimensi masukan (panjang/lebar) fitur output

f = Ukuran kernel (filter) yang digunakan dalam konvolusi.

p = Padding, yaitu jumlah piksel yang ditambahkan di sekitar fitur input untuk mempertahankan dimensi tertentu.

s = Stride, yaitu langkah pergeseran kernel pada setiap operasi konvolusi.

n_{out} = Dimensi keluaran (panjang/lebar) fitur output.

$O(x_{out}, y_{out})$ = Representasi piksel citra *output* dari $(0,0)$ hingga $(n_{out} - 1, n_{out} - 1)$,dengan $n \times n$ menyatakan ukuran citra dalam piksel.

$I(x_{in}, y_{in})$ = Representasi piksel citra *input* dari $(0,0)$ hingga $(n_{in} - 1, n_{in} - 1)$,dengan $n \times n$ menyatakan ukuran citra dalam piksel.

K = *Kernel* berukuran $f \times f$.

Δ = Selisih jumlah sel antara pusat dengan tepi *Kernel* K,misal jika $f = 3$, maka $\Delta = 1$.

2.4 *MobileNetV2*

2.4.1 Deskripsi *MobileNetV2*

MobileNetV2 adalah sebuah arsitektur jaringan saraf yang dirancang khusus untuk perangkat mobile dan lingkungan terbatas sumber daya. Arsitektur ini merupakan pengembangan dari *MobileNetV1* dan berhasil meningkatkan kinerja model-model mobile pada berbagai tugas dan benchmark, serta pada berbagai ukuran model yang berbeda. Salah satu komponen utama dari *MobileNetV2* adalah modul lapisan baru yang disebut *inverted residual* with linear *bottleneck*. Modul ini mengambil representasi yang terkompresi dalam dimensi rendah sebagai input, kemudian melakukan ekspansi menjadi dimensi yang lebih tinggi dan menyaring fitur menggunakan *depthwise convolution* yang ringan. Fitur-fitur tersebut kemudian diproyeksikan kembali menjadi representasi dimensi rendah menggunakan konvolusi linear. Implementasi resmi dari modul ini tersedia sebagai bagian dari pustaka model *TensorFlow-Slim*. Salah satu keunggulan dari *MobileNetV2* adalah kemampuannya untuk mengurangi kebutuhan operasi komputasi dan memori, sambil tetap mempertahankan tingkat akurasi yang sama dengan arsitektur sebelumnya. Hal ini dicapai melalui penggunaan *depthwise separable convolution*, di mana konvolusi

standar digantikan dengan versi yang terfaktorisasi menjadi dua lapisan terpisah: *depthwise convolution* dan konvolusi *pointwise*. Pendekatan ini mengurangi komputasi yang dibutuhkan hingga 8 hingga 9 kali lipat dibandingkan dengan konvolusi standar (Sandler dkk., 2018).

Arsitektur *MobileNetV2* juga memperkenalkan *bottleneck layer* dengan ekspansi *layer*. *Bottleneck layer* ini berguna untuk mengurangi dimensi lapisan dan ruang operasi, sehingga mengurangi beban komputasi dan memori yang dibutuhkan. Ekspansi *layer* digunakan untuk memperluas dimensi fitur sebelum dilakukan *depthwise convolution*, sehingga meningkatkan kapasitas representasi dan kinerja model. *MobileNetV2* telah berhasil mencapai *state-of-the-art* pada berbagai tugas pengenalan gambar dan deteksi objek untuk aplikasi mobile. Arsitektur ini memberikan keseimbangan yang baik antara akurasi dan kinerja komputasi, sehingga sangat cocok untuk digunakan pada perangkat mobile dengan sumber daya terbatas.

2.4.2 *Depthwise Separable Convolution*

Depthwise separable convolution merupakan blok *layer* konvolusi dalam arsitektur jaringan konvolusi. Ide dasarnya adalah menggantikan operator konvolusi penuh dengan versi yang terfaktorisasi menjadi dua lapisan terpisah. Lapisan pertama disebut *depthwise convolution*, yang melakukan filtrasi ringan dengan menerapkan satu filter konvolusi per channel input. Lapisan kedua adalah konvolusi 1×1, yang disebut konvolusi *pointwise*, bertanggung jawab untuk membangun fitur baru melalui penghitungan kombinasi linear dari channel input. Konvolusi standar mengambil tensor input L_i dengan dimensi $h_i \times w_i \times d_i$, dan menerapkan kernel konvolusi $K \in R^{k \times k \times d_i \times d_j}$ untuk menghasilkan tensor output L_j dengan dimensi $h_i \times w_i \times d_j$. Lapisan konvolusi standar memiliki biaya komputasi $h_i \cdot w_i \cdot d_i \cdot d_j \cdot k \cdot k$ (Sandler dkk., 2018).

Dalam *depthwise separable convolution*, prosesnya dibagi menjadi dua tahap. Pertama, *depthwise convolution* menerapkan filter konvolusi pada masing-masing channel input secara terpisah, menghasilkan tensor dengan dimensi $h_i \times w_i \times d_i$. Kemudian, *pointwise convolution* menggabungkan kembali channel-channel tersebut dengan menggunakan konvolusi 1×1, menghasilkan tensor dengan dimensi $h_i \times w_i \times d_j$. Dengan menggunakan *depthwise separable convolution*, beban komputasi dapat

dikurangi menjadi hi · wi · di · k · k + hi · wi · di · dj, yang jauh lebih efisien dibandingkan dengan konvolusi standar. Dengan menggunakan *depthwise separable convolution*, arsitektur jaringan saraf dapat menjadi lebih efisien dalam hal komputasi dan memori, sambil tetap mempertahankan kemampuan untuk mempelajari fitur-fitur yang kompleks. *Depthwise separable convolution* dapat digunakan sebagai pengganti langsung untuk lapisan konvolusi standar. Persamaan 2.3 merupakan rumus yang digunakan untuk menghitung jumlah operasi dalam *layer* konvolusi dari CNN. Dalam praktiknya, konvolusi ini memberikan kinerja yang hampir sama baiknya dengan konvolusi standar.

Jumlah operasi *layer* konvolusi $= h_i \cdot w_i \cdot d_i(k^2 + d_j)$ ..... (2.3)

Keterangan :

h_i = Dimensi tinggi fitur input pada lapisan ke-i

w_i = Dimensi lebar fitur input pada lapisan ke-i

d_i = Jumlah channel (kedalaman) fitur input.

k^2 = Luas kernel konvolusi

d_j = Jumlah channel (kedalaman) fitur output.

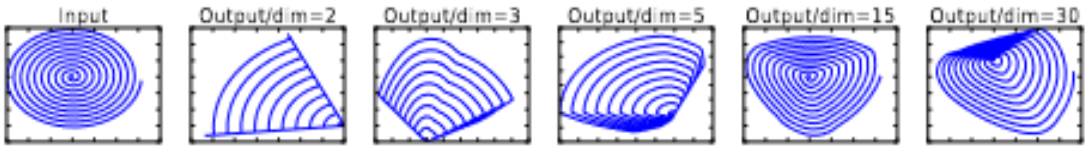
Depthwise separable convolution mengurangi beban komputasi yang dibutuhkan dibandingkan dengan lapisan konvolusi standar secara signifikan, hampir sebesar faktor k^2. Sebagai contoh, dalam *MobileNetV2*, digunakan *depthwise separable convolution* dengan ukuran k = 3 (*depthwise separable convolution* 3 × 3), sehingga biaya komputasinya menjadi 8 hingga 9 kali lebih kecil dibandingkan dengan konvolusi standar. Hal ini berarti *depthwise separable convolution* mampu menghasilkan hasil yang hampir setara dengan konvolusi standar namun dengan penggunaan sumber daya komputasi yang jauh lebih efisien. Meskipun terdapat pengurangan biaya komputasi yang signifikan, pengorbanan dalam hal akurasi model hanya sedikit, sehingga *depthwise separable convolution* menjadi pilihan yang sangat efisien dalam meningkatkan kinerja jaringan saraf dengan beban yang minimal.

2.4.3 *Linear Bottlenecks*

Linear bottlenecks merujuk pada pengurangan dimensionalitas lapisan dalam jaringan saraf untuk mengurangi dimensi ruang operasional. *MobileNetV2* adalah salah satu arsitektur jaringan saraf yang sukses memanfaatkan konsep ini. Pada dasarnya,

MobileNetV2 menggunakan pendekatan *width multiplier* yang memungkinkan pengurangan dimensi ruang aktivasi hingga *manifold of interest* mencakup seluruh ruang tersebut. *Width multiplier* adalah parameter yang mengontrol lebar (*dimensionality*) dari lapisan dalam jaringan. Dengan mengurangi lebar lapisan, jumlah fitur yang dihasilkan oleh lapisan tersebut berkurang, sehingga mengurangi kompleksitas komputasi dan mempercepat waktu pelatihan. Dalam *MobileNetV2*, pengurangan dimensionalitas lapisan dilakukan melalui penggunaan blok *linear bottleneck*. Blok ini terdiri dari beberapa lapisan konvolusi yang diikuti oleh lapisan *Batch Normalization* dan ReLU. Namun, pada lapisan terakhir dari blok ini, tidak ada aktivasi ReLU yang diterapkan. Dalam hal ini, keluaran lapisan tersebut adalah hasil dari operasi konvolusi linier, yang mengurangi dimensionalitas ruang aktivasi. Pendekatan linear *bottleneck* ini memungkinkan jaringan *MobileNetV2* untuk mencapai efisiensi komputasi yang tinggi dengan mengurangi jumlah parameter dan operasi komputasi yang diperlukan, sambil mempertahankan tingkat akurasi yang baik. Dalam praktiknya, pengurangan dimensionalitas ini membantu mempercepat waktu pelatihan dan inferensi pada perangkat dengan sumber daya terbatas, seperti perangkat mobile (Sandler dkk., 2018).

Berikut adalah spiral awal yang ditanamkan dalam ruang n-dimensi menggunakan matriks acak T diikuti oleh fungsi ReLU yang ditunjukkan pada Gambar 2.5, dan kemudian diproyeksikan kembali ke ruang 2D menggunakan invers matriks T. Ketika n = 2 atau 3, terjadi kehilangan informasi di mana beberapa titik pada *manifold* saling roboh, sementara untuk n = 15 hingga 30, transformasi menjadi sangat non-konveks.



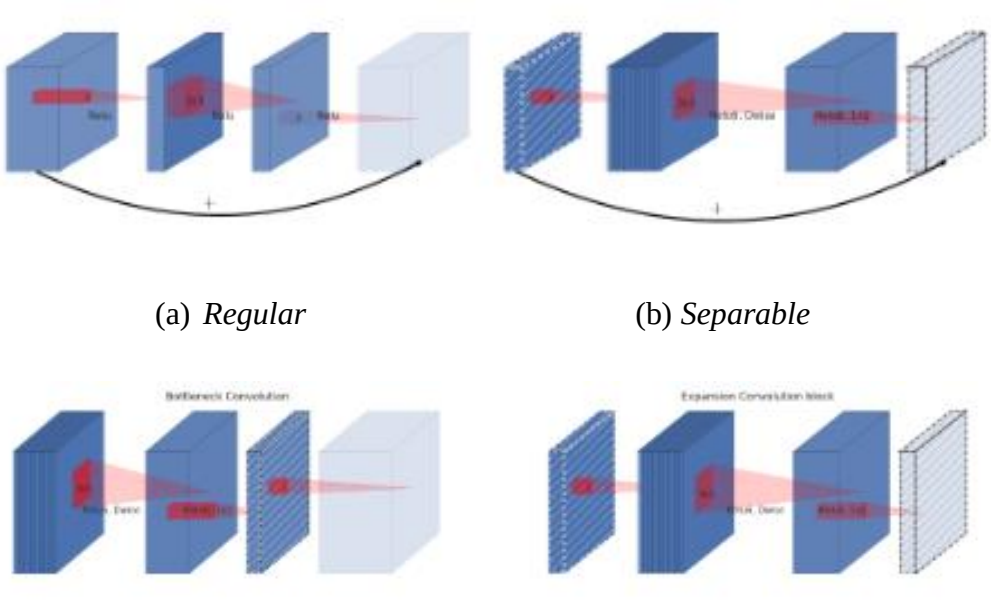
Gambar 2.5 Kasus transformasi ReLU dari *manifold* berdimensi rendah yang terbenam dalam ruang berdimensi tinggi (Sandler dkk., 2018)

Setiap sub-gambar menunjukkan bagaimana *manifold* awal, yang ditampilkan sebagai kurva tertutup pada gambar Input, berubah ketika dimensi output meningkat.

Manifold awal, yang dimulai dalam ruang dua dimensi (dataran dua dimensi), mengalami perubahan bentuk yang signifikan dan menjadi semakin kompleks saat dimensi output meningkat dari 2 hingga 30. Transformasi non-linear seperti ReLU (*Rectified Linear Unit*) memainkan peran penting dalam proses ini, karena menunjukkan bagaimana peningkatan dimensi output memperkaya representasi fitur dengan kompleksitas yang lebih tinggi. Saat *manifold* dua dimensi dipetakan ke ruang berdimensi lebih tinggi, bentuknya mulai menunjukkan lebih banyak variasi dan ketidakteraturan. Transformasi ini menggambarkan bagaimana data dapat diubah dan dipelajari dalam jaringan saraf. Dalam dimensi yang lebih tinggi, *manifold* memiliki kemampuan untuk menangkap lebih banyak detail dan pola dalam data, yang tidak mungkin terdeteksi dalam dimensi yang lebih rendah. Namun, meskipun *manifold* yang lebih tinggi dapat menangkap lebih banyak informasi dan kompleksitas, mereka juga menjadi lebih rumit dan sulit untuk divisualisasikan.

Proses ini mengilustrasikan kemampuan transformasi non-linear seperti ReLU untuk mengubah dan memperkaya representasi fitur melalui peningkatan dimensi ruang fitur. Dengan memanfaatkan dimensi output yang lebih tinggi, jaringan saraf dapat membedakan dan mengenali pola yang lebih kompleks, sehingga meningkatkan kinerja dalam berbagai tugas pembelajaran mesin. Transformasi ReLU membantu dalam menjaga sifat non-linear dari data, memungkinkan jaringan untuk belajar representasi yang lebih fleksibel dan kaya. Hal ini menunjukkan bagaimana *manifold* awal yang sederhana dapat berkembang menjadi struktur yang sangat kompleks dan kaya informasi ketika diproses melalui beberapa lapisan jaringan saraf dengan dimensi output yang meningkat. Dalam jaringan saraf, dimensi output yang lebih tinggi memungkinkan setiap lapisan untuk menghasilkan representasi fitur yang lebih detail dan spesifik. Ini penting karena tugas pembelajaran mesin, seperti klasifikasi gambar, sering kali melibatkan pola kompleks yang memerlukan pemahaman yang mendalam tentang data.

Berikut adalah beberapa jenis blok konvolusi yang digunakan dalam arsitektur *MobileNetV2* yang ditunjukkan pada Gambar 2.6.



(a) *Regular* (b) *Separable*

(c) *Separable With Bottleneck* (d) *Bottleneck with expansion layer*

Gambar 2.6 *Evolution of Separable Convolution Blocks* (Sandler dkk., 2018)

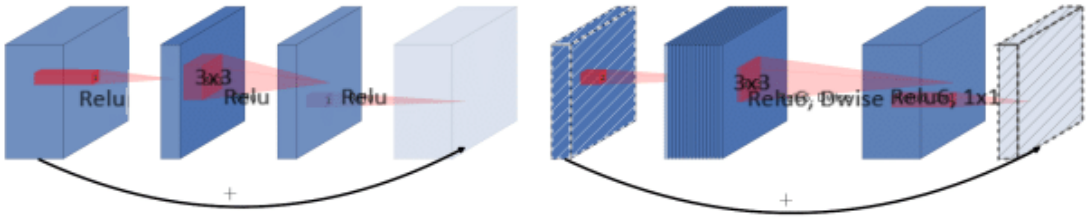
Pada Gambar 2.6 (a), konvolusi reguler diterapkan secara standar dengan setiap filter bekerja pada seluruh saluran input, menghasilkan keluaran dengan saluran yang sama banyaknya dengan jumlah filter. Gambar 2.6 (b) menunjukkan konvolusi terpisah (*separable convolution*), di mana konvolusi dibagi menjadi dua tahap: *depthwise* dan *pointwise*, mengurangi jumlah parameter dan komputasi. Gambar 2.6 (c) memperkenalkan "*linear bottleneck*" dalam konvolusi terpisah, menambahkan lapisan linear untuk menjaga informasi penting yang mungkin hilang akibat fungsi aktivasi non-linear. Terakhir, Gambar 2.6 (d) memperlihatkan "*bottleneck with expansion layer*", teknik yang memperluas dimensi fitur sebelum konvolusi terpisah dan kemudian menguranginya kembali dengan konvolusi linear, memungkinkan model menangkap fitur kompleks dengan efisiensi komputasi tinggi. Teknik ini dikenal sebagai "*inverted residuals*" dan "*linear bottleneck*s", yang secara keseluruhan menciptakan jaringan yang efisien dalam hal komputasi dan penggunaan memori, sambil mempertahankan akurasi tinggi. Pola garis miring menunjukkan lapisan-lapisan yang tidak mengandung non-linearitas. Lapisan terakhir (dengan warna yang lebih terang) menandakan awal blok berikutnya.

ReLU dapat menyebabkan kehilangan informasi dalam sebuah channel ketika channel tersebut roboh. Namun, dengan banyak channel dan *manifold* aktivasi yang terstruktur, informasi mungkin masih tetap terjaga di channel lainnya. Dalam bahan tambahan, ditunjukkan bahwa jika *manifold* input dapat disematkan ke dalam subruang dimensi yang lebih rendah dari ruang aktivasi, ReLU dapat mempertahankan informasi sambil memperkenalkan kompleksitas yang diperlukan. Secara ringkas, kebutuhan utamanya adalah *manifold* yang diminati harus berada dalam subruang dimensi yang lebih rendah dari ruang aktivasi yang berdimensi lebih tinggi.

Namun, penting untuk dicatat bahwa meskipun pengurangan dimensionalitas dapat memberikan manfaat komputasi, pendekatan ini mengabaikan sifat transformasi non-linear seperti ReLU dalam jaringan saraf konvolusi dalam. ReLU dapat menghasilkan transformasi non-linear yang kompleks, yang tidak dapat sepenuhnya direpresentasikan oleh transformasi linier. Oleh karena itu, meskipun linear *bottleneck* dapat mengurangi dimensi ruang operasional, mereka memiliki keterbatasan dalam merepresentasikan *manifold* of interest secara penuh dalam subruang berdimensi rendah. Jika setelah transformasi ReLU, *manifold* yang diinginkan masih memiliki volume yang tidak nol, maka itu menunjukkan bahwa transformasi tersebut bersifat linear. Dengan kata lain, meskipun pengurangan dimensi dapat mengurangi kompleksitas komputasi, tetapi tidak dapat sepenuhnya menggambarkan manifold yang lebih kaya atau struktur non-linear yang ada dalam data, yang tetap mempengaruhi representasi data dalam ruang berdimensi lebih rendah. Dengan kata lain, meskipun dilakukan pengurangan dimensi untuk mempercepat proses komputasi dan membuatnya lebih efisien dalam hal penggunaan memori dan waktu pemrosesan, transformasi non-linear yang diterapkan oleh ReLU di dalam jaringan saraf konvolusi dalam tidak dapat sepenuhnya digantikan oleh pendekatan linier.

Residual block menghubungkan lapisan-lapisan dengan jumlah channel yang tinggi, sedangkan *inverted residual* menghubungkan *bottleneck*. Garis-garis diagonal menunjukkan lapisan-lapisan yang tidak menggunakan non-linearitas. Ketebalan setiap blok menunjukkan jumlah channel relatifnya. ReLU mampu mempertahankan informasi lengkap tentang *manifold* input, namun hanya jika *manifold* input berada dalam subruang dimensi rendah dari ruang input. Dengan asumsi *manifold* yang diminati memiliki dimensi yang rendah, kita dapat menangkapnya dengan

menyisipkan lapisan *bottleneck linear* ke dalam blok konvolusi. Perbedaan antara *residual block* dan *inverted residual block* dapat dilihat pada Gambar 2.7.



(a) *Residual Block* (b) *Inverted Residual Block*

Gambar 2.7 Perbedaan antara *residual block* dan *inverted residual block* (Sandler dkk., 2018)

Bukti eksperimental menunjukkan bahwa penggunaan lapisan linear sangat penting karena mencegah non-linearitas menghancurkan terlalu banyak informasi. Penggunaan lapisan non-linear pada *bottleneck* merugikan kinerja sebesar beberapa persen.

2.4.4 *Inverted Residual*

Bottleneck block tampak mirip dengan *residual block* di mana setiap blok berisi input diikuti oleh beberapa *bottleneck* kemudian diikuti oleh ekspansi. Namun, terinspirasi oleh intuisi bahwa *bottleneck* sebenarnya mengandung semua informasi yang diperlukan, sementara lapisan ekspansi bertindak hanya sebagai detail implementasi yang menyertai transformasi non-linear dari tensor, kami menggunakan shortcut langsung antara *bottleneck*. Struktur implementasi dasar diilustrasikan dalam Tabel 2.2. Untuk sebuah blok berukuran h × w, faktor ekspansi t, dan ukuran kernel k dengan d′ channel input dan d″ channel output, total jumlah perkalian dan penambahan yang diperlukan adalah h · w · d′ · t(d′ + k2 + d″). Dibandingkan dengan (1) ekspresi ini memiliki tambahan satu istilah, karena kita memang memiliki konvolusi 1 × 1 tambahan, namun sifat jaringan kita memungkinkan kita untuk menggunakan dimensi input dan output yang jauh lebih kecil.

Tabel 2.2 *Bottleneck residual block berubah* dari *k* ke *k′ channels*

Input	Operator	Output
h × w × k	1 × 1 conv2d, ReLU6	h × w × (tk)
h × w × tk	1 × 1 dwise = s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times (tk)$	Linear 1 × 1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

2.4.5 Arsitektur *MobileNetV2*

Arsitektur *MobileNetV2* mengandung lapisan konvolusi penuh awal dengan 32 filter, diikuti oleh 19 lapisan *bottleneck residual*. (Sandler dkk., 2018) melakukan eskperimen, ReLU6 digunakan sebagai fungsi non-linearitas karena keandalannya saat digunakan dengan komputasi presisi rendah. Ukuran kernel 3 × 3 selalu digunakan karena standar untuk jaringan modern, dan menggunakan *dropout* dan normalisasi *batch* selama pelatihan. Dengan pengecualian lapisan pertama, tingkat ekspansi konstan digunakan di seluruh jaringan. Ditemukan bahwa tingkat ekspansi antara 5 dan 10 menghasilkan kurva kinerja yang hampir identik, dengan jaringan yang lebih kecil lebih baik dengan tingkat ekspansi yang sedikit lebih kecil dan jaringan yang lebih besar memiliki kinerja yang sedikit lebih baik dengan tingkat ekspansi yang lebih besar. Faktor ekspansi 6 yang diterapkan pada ukuran *tensor input* digunakan untuk eksperimen utama. Sebagai contoh, untuk sebuah lapisan *bottleneck* yang mengambil tensor input dengan 64 channel dan menghasilkan tensor dengan 128 *channel*, lapisan ekspansi intermediat kemudian memiliki 64 × 6 = 384 *channel*.

MobileNetV2 menawarkan efisiensi komputasi tinggi dengan penggunaan *depthwise separable convolutions* dan blok *bottleneck inverted residual*, membuatnya cocok untuk perangkat dengan keterbatasan sumber daya, sambil tetap memberikan kinerja pengenalan gambar yang baik. Fleksibilitas arsitektur ini memungkinkan

adaptasi untuk berbagai aplikasi dengan penyesuaian kecil pada struktur jaringannya, menjadikannya pilihan yang efisien dan kuat untuk aplikasi pengenalan gambar pada perangkat dengan sumber daya terbatas. Konfigurasi *Linear Bottlenecks* dapat dilihat pada Tabel 2.3.

Tabel 2.3 Konfigurasi *Linear Bottlenecks MobileNetV2*

Input	**Operator**	**t**	**c**	**n**	**s**
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	*bottleneck*	1	16	1	1
$112^2 \times 16$	*bottleneck*	6	24	2	2
$56^2 \times 24$	*bottleneck*	6	32	3	2
$28^2 \times 32$	*bottleneck*	6	64	4	2
$14^2 \times 64$	*bottleneck*	6	96	3	1
$14^2 \times 96$	*bottleneck*	6	160	3	2
$7^2 \times 160$	*bottleneck*	6	320	1	1
$7^2 \times 320$	conv2d 1×1	-	1280	1	1
$7^2 \times 1280$	avgpool 7×7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1×1	-	k	-	-

Setiap baris menggambarkan urutan 1 atau lebih lapisan yang identik (modulus langkah), diulang n kali. Semua lapisan dalam urutan yang sama memiliki jumlah *output channel* yang sama, yaitu c. Lapisan pertama dari setiap urutan memiliki langkah s dan semua lapisan lain menggunakan langkah 1. Semua konvolusi spasial menggunakan kernel 3 × 3. Perbandingan ukuran model untuk *MobileNetV1, MobileNetV2,* dan *ShuffleNet* dapat dilihat pada Tabel 2.4.

Tabel 2.4 Perbandingan Ukuran Model untuk *MobileNetV1, MobileNetV2,* dan *ShuffleNet* (2x,g=3)

Size	***MobileNetV1***	***MobileNetV2***	***ShuffleNet* (2x,g=3)**
112×12	64/1600	$16/400$	$32/800$
56×56	128/800	32/200	48/300
28×28	256/400	64/100	400/600K
14×14	512/200	160/62	800/310
7×7	1024/199	320/32	1600/156
1×1	1024/2	1280/2	1600/3
max	1600K	400K	600K

Jumlah maksimum *channel*/memori (dalam Kb) yang perlu dimaterialisasikan pada setiap resolusi spasial untuk arsitektur yang berbeda. Kami mengasumsikan bilangan bulat 16-bit untuk aktivasi. Untuk *ShuffleNet*, 2x, g = 3 digunakan untuk perbandingan ini yang sesuai dengan kinerja *MobileNetV1* dan *MobileNetV2*.

2.5 Pembagian Data

Dalam penelitian ini, data akan dibagi menjadi data latih, validasi, dan uji. Pembagian data menjadi *train-validation-test* penting dalam klasifikasi penyakit pada tanaman padi. Pembagian data memungkinkan kita untuk memilih model terbaik berdasarkan kinerjanya pada data validasi, sehingga memastikan bahwa model yang dipilih memiliki kemampuan generalisasi yang baik dan mampu mengklasifikasikan berbagai jenis penyakit pada tanaman padi dengan akurat. Selain itu, dengan memisahkan data validasi dari data pelatihan, kita dapat mengendalikan risiko *overfitting*, di mana model hanya "menghafal" data pelatihan tetapi tidak mampu menggeneralisasi pada data baru (Vabalas dkk., 2019).

2.6 *Data augmentation*

Data augmentation adalah metode yang efektif dalam meningkatkan jumlah, kualitas, dan keragaman data pelatihan dalam domain visi komputer. Dengan memanfaatkan teknik augmentasi yang bervariasi, seperti menghasilkan sampel tambahan dari data pelatihan yang ada atau membuat data baru menggunakan berbagai strategi, *data augmentation* dapat membantu mengatasi keterbatasan data pelatihan yang ada. Pendekatan ini tidak hanya meningkatkan volume data pelatihan, tetapi juga membantu meningkatkan kinerja model pembelajaran mesin dengan memungkinkan arsitektur yang lebih sederhana mencapai kinerja generalisasi yang tinggi (Mumuni & Mumuni, 2022).

Terdapat beberapa proses spesifik *data augmentation* yang dapat digunakan dalam bidang visi komputer. Pertama, augmentasi berbasis transformasi seperti rotasi, pergeseran, pemotongan, pencerahan, dan penajaman dapat diterapkan pada gambar atau urutan video untuk menciptakan variasi yang lebih besar dalam data pelatihan.

Selain itu, augmentasi berbasis pemodelan grafis 3D memanfaatkan pembuatan data sintetis menggunakan pemodelan grafis 3D yang realistis untuk melengkapi atau memperluas data pelatihan yang ada. Pendekatan lainnya adalah augmentasi berbasis jaringan generatif *adversarial* (GAN), yang menggunakan dua jaringan saraf yang bersaing untuk menghasilkan data sintetis yang realistis. Terakhir, augmentasi berbasis pembelajaran meta memanfaatkan model pembelajaran mesin untuk mempelajari transformasi atau kebijakan augmentasi yang optimal secara otomatis dari data pelatihan yang ada. Dengan memanfaatkan berbagai pendekatan ini, *data augmentation* dapat meningkatkan kualitas dan keragaman data pelatihan serta membantu meningkatkan performa model dalam tugas visi komputer.

2.7 *Data normalization*

Data normalization adalah salah satu pendekatan *pra-processing* dimana data diubah skala atau ditransformasikan untuk memberikan kontribusi yang sama dari setiap fitur. Keberhasilan algoritma pembelajaran mesin bergantung pada kualitas data untuk mendapatkan model prediksi yang umum dari masalah klasifikasi. Pentingnya normalisasi data untuk meningkatkan kualitas data dan kinerja algoritma pembelajaran mesin telah dipresentasikan dalam banyak penelitian. Namun, penelitian ini kurang memperhatikan pendekatan pemilihan fitur dan pembobotan fitur, yang merupakan tren penelitian saat ini dalam pembelajaran mesin untuk meningkatkan kinerja. Oleh karena itu, penelitian ini bertujuan untuk menyelidiki dampak empat belas metode normalisasi data terhadap kinerja klasifikasi dengan mempertimbangkan kumpulan fitur lengkap, pemilihan fitur, dan pembobotan fitur.

Dalam penelitian (Singh & Singh, 2020), pentingnya normalisasi data untuk membangun model prediksi yang akurat telah ditinjau untuk berbagai algoritma pembelajaran mesin, seperti *k-Nearest Neighbors* (k-NN), *Artificial Neural networks* (ANN), dan *Support Vector Machines* (SVM). Banyak penulis telah memvalidasi dampak normalisasi data untuk meningkatkan kinerja klasifikasi dalam berbagai bidang, seperti klasifikasi data medis, sistem biometrik multimodal, klasifikasi kendaraan, deteksi motor yang rusak, prediksi pasar saham, klasifikasi daun, klasifikasi data persetujuan k*red*it, genomika, dan beberapa bidang aplikasi lainnya.

Dalam arsitektur *MobileNetV2*, normalisasi data adalah proses penting yang dilakukan sebelum gambar masuk ke dalam jaringan *neural*. Langkah pertama adalah *preprocessing* gambar, di mana gambar diubah ukurannya ke dimensi yang ditentukan, seperti 224 x 224 piksel, dan kadang-kadang dilakukan *cropping* acak untuk variasi tambahan dalam data pelatihan. Setelah itu, nilai piksel dari gambar dinormalisasi untuk memastikan setiap piksel memiliki rentang yang seragam dan stabil. Normalisasi ini umumnya dilakukan dengan mengurangi mean *global* dari setiap piksel dan membaginya dengan standar deviasi dari seluruh *dataset* pelatihan, sehingga menghasilkan data yang terstandarisasi dengan rentang umumnya berada di [-1, 1] atau [0, 1].

Setelah normalisasi, gambar-gambar yang telah diproses ini menjadi input untuk *MobileNetV2*. Arsitektur ini, dikenal karena efisiensinya, menggunakan teknik *depthwise separable convolution* yang memungkinkan pengurangan jumlah parameter dan operasi komputasi yang dibutuhkan. Dengan demikian, prosesnya menjadi lebih cepat dan lebih hemat daya, ideal untuk implementasi di perangkat mobile atau edge computing. Selama pelatihan, model diatur untuk mengoptimalkan parameter-parameter internalnya melalui proses *backpropagation*, yang menggunakan data latihan untuk menyesuaikan bobot-bobot dalam jaringan. Saat fase inferensi, *MobileNetV2* menggunakan pengetahuan yang diperoleh dari pelatihan untuk melakukan prediksi dengan cepat dan akurat terhadap gambar-gambar baru yang disajikan padanya, menjadikannya pilihan yang populer dalam aplikasi visi komputer di berbagai platform.

2.7 Aplikasi *Desktop*

Aplikasi *desktop* adalah perangkat lunak yang berjalan secara lokal pada komputer atau laptop pengguna. Keunggulan aplikasi *desktop* meliputi akses offline, respons cepat, dan pengalaman pengguna yang kaya (Bustamin , 2021). Aplikasi ini sering digunakan untuk berbagai keperluan seperti pengolahan data, klasifikasi, dan visualisasi hasil.

Pada penelitian ini, aplikasi *desktop* dikembangkan untuk mengintegrasikan model klasifikasi penyakit tanaman padi yang telah dilatih menggunakan *Convolutional Neural Network* (CNN) berbasis *MobileNetV2*. Aplikasi ini

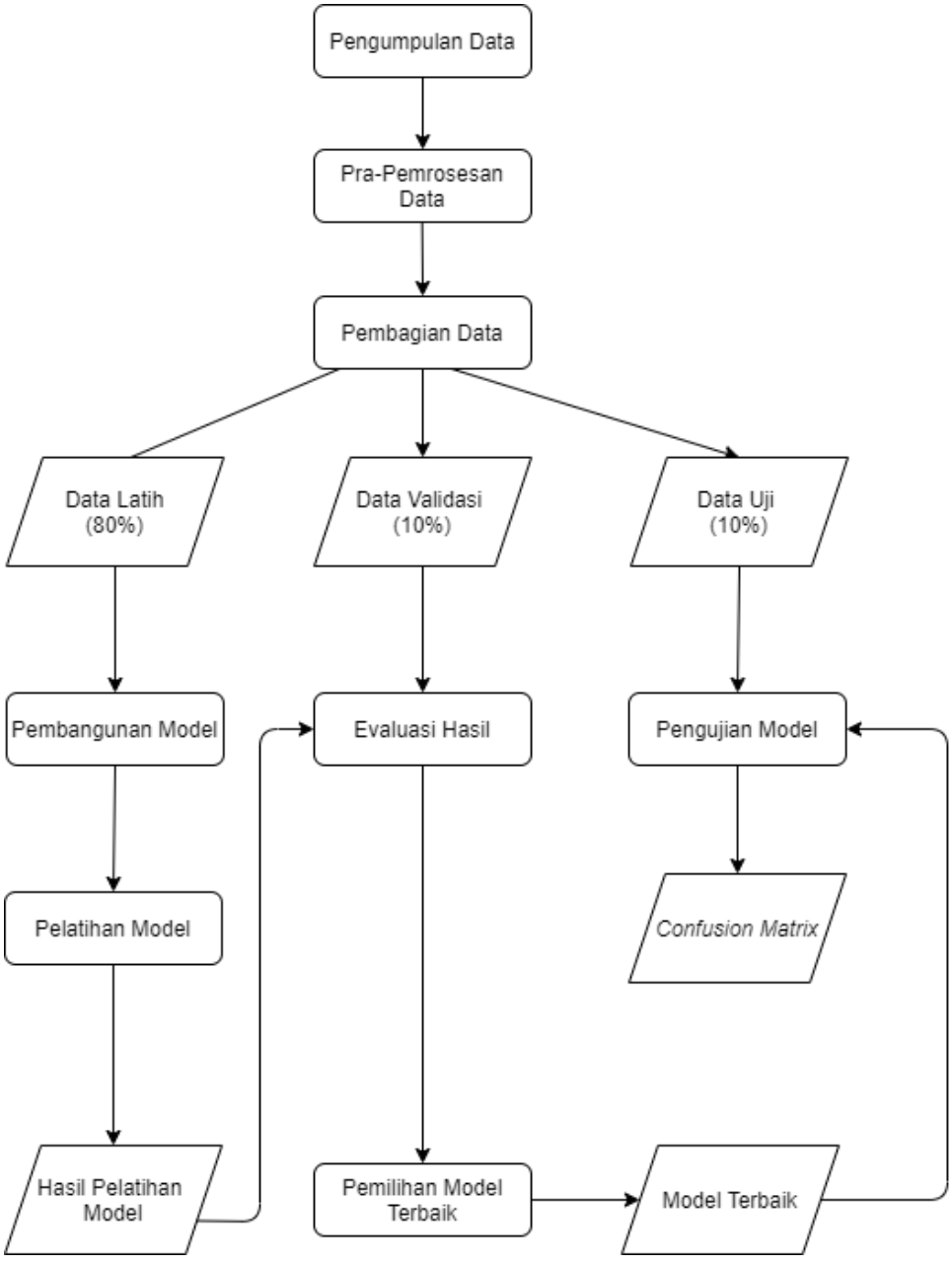
memungkinkan pengguna untuk memanfaatkan model tersebut secara lebih mudah dan efisien, tanpa perlu terhubung ke Google Colab atau platform cloud lainnya.

2.7 *Tkinter*

Tkinter adalah pustaka standar *Python* yang digunakan untuk membuat aplikasi desktop dengan antarmuka grafis (GUI) (Hutama, 2021). Pustaka ini menawarkan komponen GUI dasar seperti tombol, label, menu, dan kotak input yang dapat dengan mudah diintegrasikan dengan aplikasi machine learning. *Tkinter* memiliki kelebihan berupa kemudahan dalam implementasi dan sudah terintegrasi langsung dengan *Python*, sehingga tidak memerlukan instalasi tambahan. Dalam konteks penelitian ini, *Tkinter* dapat digunakan untuk membangun antarmuka sederhana yang memungkinkan pengguna mengunggah gambar daun padi, menjalankan model klasifikasi, dan menampilkan hasil deteksi penyakit secara real-time.

BAB III
METODE PENELITIAN

Bab ini menyajikan metode penelitian yang diterapkan dalam penelitian ini, yaitu metode klasifikasi penyakit pada tanaman padi menggunakan algoritma CNN berbasis *MobileNetV2*. Pembahasan diawali dengan garis besar penyelesaian masalah, seperti yang dapat dilihat pada Gambar 3.1.



Gambar 3.1 *Flowchart* Garis Besar Penyelesaian Masalah

Flowchart tersebut merupakan gambaran umum dari proses penyelesaian masalah untuk klasifikasi penyakit pada tanaman padi menggunakan algoitma CNN berbasis *MobileNetV2*. Proses pengumpulan data citra digital tanaman padi yang terinfeksi penyakit menjadi kunci keberhasilan dalam proyek ini. Data citra dapat diperoleh melalui *website kaggle*. Data citra tersebut kemudian melalui tahap pra-pemrosesan, di mana citra-citra tersebut dibersihkan, diubah ukurannya, dan dinormalisasi agar sesuai untuk diproses lebih lanjut oleh model CNN. Pada tahap pra-pemrosesan, selain teknik augmentasi data yang telah disebutkan sebelumnya, *preprocessing* lainnya seperti segmentasi daun, ekstrasi fitur warna dan tekstur, serta transformasi citra ke dalam ruang fitur yang lebih informatif (misalnya, transformasi *Fourier* atau *wavelet*) dapat dipertimbangkan untuk meningkatkan kemampuan model dalam mendeteksi pola visual penyakit secara lebih akurat.

Teknik augmentasi data merupakan metode yang digunakan untuk memperkaya dan memperbanyak jumlah data latih tanpa harus mengumpulkan data baru secara langsung. Dalam konteks klasifikasi penyakit pada tanaman padi menggunakan CNN berbasis MobileNetV2, augmentasi data memainkan peran penting dalam meningkatkan generalisasi model dan mencegah *overfitting*. Beberapa teknik augmentasi yang umum digunakan meliputi rotasi citra, seperti mengubah sudut citra secara acak (misalnya 90°, 180°, atau 270°), yang memberi model berbagai sudut pandang terhadap objek yang sama. Pembalikan citra secara horizontal atau vertikal juga berguna untuk mengenali objek dari sisi yang berlawanan, mensimulasikan variasi posisi yang mungkin terjadi pada tanaman. Selain itu, perubahan ukuran dan pemotongan citra (*cropping*) memberikan variasi pada data latih dengan mengubah bagian-bagian citra yang diproses, membantu model belajar dari citra yang lebih fleksibel. Modifikasi kecerahan, kontras, dan saturasi warna juga dapat diterapkan untuk mensimulasikan kondisi pencahayaan yang berbeda, yang sangat berguna mengingat perbedaan pencahayaan saat pengambilan gambar tanaman. Teknik lainnya, seperti translasi (*shifting*), memungkinkan citra digeser pada grid citra untuk memberikan variasi posisi objek, sementara penambahan noise pada citra mensimulasikan gangguan yang mungkin terjadi pada data asli, seperti noise sensor kamera. Selain itu, pemotongan objek (*shearing*) juga bisa digunakan untuk memiringkan bentuk objek dalam citra, memberikan tantangan ekstra bagi model dalam mengenali pola penyakit. Setelah itu, data diklasifikasikan menjadi tiga set: data latih, data uji, dan data validasi, dengan proporsi yang

seimbang untuk menjamin akurasi dan generalisasi model. Pembagian data ini dilakukan secara acak untuk menghindari bias dalam pelatihan dan evaluasi model.

Selanjutnya, dilakukan pemodelan dengan menggunakan metode pelatihan model, pengujian model, dan evaluasi hasil. Pada tahap pelatihan model, algoritma CNN berbasis *MobileNetV2* dilatih menggunakan data latih untuk membangun model klasifikasi yang mampu membedakan berbagai jenis penyakit pada tanaman padi. Tahap ini melibatkan proses fine-tuning *hyperparameter* model, seperti *learning rate*, dan ukuran *batch*, untuk mengoptimalkan performa model. Tahap *pengujian* model melibatkan pengujian model pada data uji untuk mengevaluasi performa model dalam mengidentifikasi penyakit secara akurat. pengukuran metrik seperti akurasi, presisi, *recall*, dan *f1-score* digunakan untuk menilai kinerja model. Sementara itu, Evaluasi Hasil menganalisis kinerja model menggunakan data validasi untuk memastikan model dapat bergeneralalisasi dengan baik dan tidak mengalami *overfitting*. Pada tahap pengujian, selain evaluasi metrik kinerja standar, analisis kesalahan prediksi model juga perlu dilakukan untuk mengidentifikasi kasus-kasus sulit, pemahaman pola kesalahan prediksi, serta analisis sensitivitas model terhadap variasi data. visualisasi aktivasi fitur pada lapisan konvolusi dapat membantu memahami proses pengambilan keputusan model, serta mengidentifikasi fitur visual yang paling informatif untuk klasifikasi penyakit tanaman padi.

3.1 Pengumpulan Data

Dalam penelitian ini, data yang dikumpulkan adalah gambar daun tanaman padi yang terkena penyakit yang diambil dari situs *kaggle* dengan url *https://www.kaggle.com/datasets/rahmi21/rice-datasets* yang dibuat oleh Rahmi21. Data ini menjadi *dataset* (kumpulan data) utama dan dasar bagi tahapan-tahapan selanjutnya dalam penelitian yang dilakukan dalam kegiatan skripsi ini. Data citra tersebut digunakan sebagai input dalam pelatihan dan pengujian. *Dataset* ini berjumlah 3.321 gambar dengan 4 jenis penyakit, yaitu *blast* (778), *blight* (820), *tungro* (821), *brownspot* (802). Tipe gambarnya adalah *Joint Photographic Experts Group* **(**JPG). Ukuran gambar bervariasi mulai dari 300 x 300 sampai 359 x 539. Contoh dari masing-masing jenis penyakit ditunjukkan pada Gambar 3.2.

(a)

(b)

(c)

(d)

Gambar 3.3 (a) Contoh citra penyakit *tungro* (b) Contoh citra penyakit *blast,* (c) Contoh citra penyakit *blight* (d) Contoh citra penyakit *brownspot.*

3.2 Pra-pemrosesan Data

Tahapan setelah pengumpulan data adalah melakukan pra-pemrosesan data. Pertama, gambar dimuat dari direktori sesuai dengan kategori penyakit tanaman padi. Kemudian, setiap gambar diubah ukurannya menjadi 224 x 224 piksel untuk memenuhi kebutuhan model. Ukuran gambar yang seragam diperlukan agar model dapat menerima input dengan dimensi yang tetap saat dilatih dan saat digunakan untuk membuat prediksi. Dalam konteks ini, gambar yang diubah ukurannya menjadi 224 x 224 piksel bertujuan untuk memenuhi persyaratan masukan model CNN berbasis *MobileNetV2*. Selanjutnya, gambar-gambar tersebut dikonversi menjadi *array numpy* untuk mengubah representasi gambar menjadi format yang dapat dimengerti oleh model *machine learning*, setiap gambar dalam *dataset* direpresentasikan sebagai larik

nilai piksel, yang memungkinkan model untuk mempelajari pola dan fitur dari data gambar. Setelah itu, array gambar dimasukkan ke dalam *flat_data_arr*, sebagai kumpulan data gambar yang sudah diproses, sementara indeks kategori penyakit tanaman padi ditambahkan ke dalam target_arr sebagai label kelas.

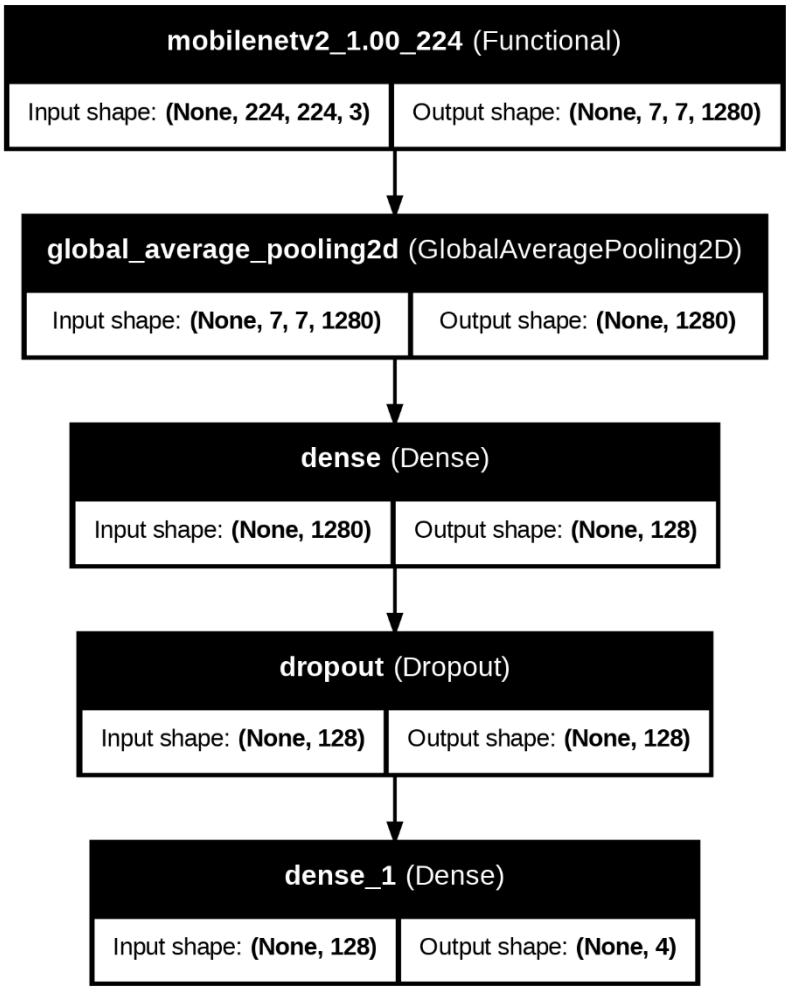
3.2 Pembagian Data

Dalam tahap ini, *dataset* akan dibagi menjadi data latih, data validasi, dan data uji. Pertama, data dibagi menjadi data latih dan data uji. Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk menguji kinerja model setelah pelatihan. Setelah itu, data latih dibagi menjadi dua bagian: data latih sebenarnya dan data validasi. Data validasi digunakan untuk mengukur kinerja model secara periodik selama pelatihan, memastikan bahwa model tidak terlalu "menghafal" data latih dan dapat menggeneralisasi dengan baik ke data yang belum pernah dilihat sebelumnya. Dengan demikian, proses ini memungkinkan kita untuk mengoptimalkan dan mengevaluasi model dengan memastikan kinerjanya baik pada data latih maupun data yang belum pernah dilihat sebelumnya. Persentase data latih dan data uji dari keseluruhan *dataset* adalah 80% data latih, 10% data validasi, dan 10% data uji (Goodfellow dkk., 2016).

3.3 Pembangunan Model

Dalam pembangunan model *MobileNetV2* untuk klasifikasi penyakit pada tanaman padi, langkah pertama yang dilakukan adalah memuat model tersebut dari *TensorFlow* Keras sebagai *base model*. *Base model* ini sudah dilatih sebelumnya pada *dataset ImageNet* yang besar. Kemudian, model baru dibentuk dengan menambahkan beberapa lapisan di atas base model. Pertama, ada lapisan *GlobalAveragePooling2D* yang bertujuan untuk meratakan output dari lapisan sebelumnya. Selanjutnya, ditambahkan *dense layer* dengan 128 unit dan fungsi aktivasi ReLU untuk memperkuat fitur-fitur yang ditemukan oleh model. Untuk menghindari *overfitting*, dilakukan penambahan *Dropout layer* dengan nilai *dropout* sebesar 0.5. Terakhir, ditambahkan *Dense layer* terakhir dengan jumlah unit yang sesuai dengan jumlah kategori penyakit pada tanaman padi, yang menggunakan fungsi aktivasi *softmax* untuk menghasilkan probabilitas distribusi kelas.

Setelah pembentukan model, langkah selanjutnya adalah membekukan (*freeze*) lapisan-lapisan pada base model agar tidak terpengaruh selama proses pelatihan. Hal ini dilakukan dengan tujuan agar fitur-fitur yang sudah dipelajari oleh base model tidak berubah. Setelah membebekukan lapisan, model dikompilasi dengan menggunakan Adam Optimizer, fungsi loss *sparse_categorical_crossentropy*, dan metrik akurasi. Dengan demikian, model siap untuk dilatih menggunakan data latih dan data validasi yang sudah dipersiapkan sebelumnya. Proses ini memungkinkan model *MobileNetV2* untuk belajar dan menyesuaikan diri dengan data pelatihan, sehingga dapat digunakan untuk melakukan klasifikasi penyakit pada tanaman padi. Gambar arsitektur model *MobileNetV2* dapat dilihat pada Gambar 3.3.



Gambar 3.3 Gambar Arsitektur Model *MobileNetV2*

3.4 Pelatihan Model

Pelatihan model *MobileNetV2* dilakukan dengan menggunakan data pelatihan yang telah dimuat dan dipersiapkan sebelumnya. Proses pelatihan dimulai dengan menginisialisasi model *MobileNetV2* yang telah disesuaikan dengan lapisan-lapisan yang sesuai untuk tugas klasifikasi penyakit pada tanaman padi. Model kemudian disusun dengan menambahkan lapisan-lapisan klasifikasi kustom, termasuk lapisan *global average pooling*, lapisan dense dengan aktivasi ReLU, lapisan *dropout* untuk mengurangi *overfitting*, dan lapisan output *softmax* dengan jumlah neuron yang sesuai dengan jumlah kategori penyakit yang diinginkan. Setelah model disusun, proses kompilasi dilakukan dengan menggunakan *optimizer* Adam, fungsi *loss sparse* categorical *crossentropy*, dan metrik evaluasi akurasi. Selanjutnya, model dilatih menggunakan data latih dan divalidasi menggunakan data validasi. Proses pelatihan dilakukan dengan menentukan jumlah *epochs* yang diinginkan, yaitu jumlah iterasi pembelajaran pada seluruh data latih. Selama pelatihan, setiap *epoch* model akan melalui proses pembelajaran maju dan mundur, di mana pembelajaran maju dilakukan untuk memperbarui bobot model berdasarkan nilai loss yang dihasilkan pada setiap *batch* data latih. Proses validasi dilakukan setelah menyelesaikan pembelajaran setiap *epoch*, di mana kinerja model dievaluasi menggunakan data validasi. Nilai loss dan metrik evaluasi lainnya, seperti akurasi, ditampilkan dan dimonitor untuk setiap *epoch* selama proses pelatihan. Proses pelatihan berlangsung hingga mencapai jumlah *epochs* yang telah ditentukan atau hingga kriteria penghentian diterapkan, seperti mencapai tingkat akurasi yang memuaskan.

Berikut adalah contoh perhitungan masing-masing bagian *layer* model *MobileNetV2* dari *layer input* hingga beberapa langkah pembaruan bobot dengan metode *back propagation*:

1. *Layer* input: Citra yang digunakan memiliki *channel* BGR yang memiliki dimensi 224 x 224 x 3. Urutan *channel* BGR diperoleh berdasarkan fungsi yang ada pada *library* cv2. Berikut adalah matriks input yang digunakan:

$$B = \begin{bmatrix} 134 & 135 & 136 & \cdots & 155 & 155 & 155 \\ 131 & 132 & 134 & \cdots & 155 & 155 & 155 \\ 127 & 129 & 133 & \cdots & 155 & 155 & 155 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 164 & 165 & 165 & \cdots & 219 & 219 & 219 \\ 164 & 165 & 165 & & 219 & 219 & 219 \\ 165 & 166 & 165 & \cdots & 219 & 219 & 219 \end{bmatrix}$$

$$G = \begin{bmatrix} 138 & 139 & 140 & \cdots & 162 & 162 & 162 \\ 135 & 136 & 138 & \cdots & 162 & 162 & 162 \\ 131 & 133 & 138 & \cdots & 162 & 162 & 162 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 168 & 169 & 169 & \cdots & 222 & 222 & 222 \\ 168 & 169 & 169 & & 222 & 222 & 222 \\ 169 & 170 & 169 & \cdots & 222 & 222 & 222 \end{bmatrix}$$

$$R = \begin{bmatrix} 149 & 150 & 151 & \cdots & 171 & 171 & 171 \\ 146 & 147 & 149 & \cdots & 171 & 171 & 171 \\ 142 & 144 & 147 & \cdots & 171 & 171 & 171 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 173 & 174 & 174 & \cdots & 227 & 227 & 227 \\ 173 & 174 & 174 & & 227 & 227 & 227 \\ 174 & 175 & 174 & \cdots & 227 & 227 & 227 \end{bmatrix}$$

2. *Zero-Padding*: *Padding* ((0,1),(0,1)) dilakukan dengan menambah nilai nol pada sisi kanan dan bawah citra, sehingga dimensinya menjadi 225 × 225 × 3. Berikut ini adalah *output* dari *zero-padding*:

$$B = \begin{bmatrix} 134 & 135 & 136 & \cdots & 155 & 155 & 0 \\ 131 & 132 & 134 & \cdots & 155 & 155 & 0 \\ 127 & 129 & 133 & \cdots & 155 & 155 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 164 & 165 & 165 & \cdots & 219 & 219 & 0 \\ 164 & 165 & 165 & & 219 & 219 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}$$

$$G = \begin{bmatrix} 138 & 139 & 140 & \cdots & 162 & 162 & 0 \\ 135 & 136 & 138 & \cdots & 162 & 162 & 0 \\ 131 & 133 & 138 & \cdots & 162 & 162 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 168 & 169 & 169 & \cdots & 222 & 222 & 0 \\ 168 & 169 & 169 & & 222 & 222 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}$$

$$R=\begin{bmatrix} 149 & 150 & 151 & \cdots & 171 & 171 & 0 \\ 146 & 147 & 149 & \cdots & 171 & 171 & 0 \\ 142 & 144 & 147 & \cdots & 171 & 171 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 173 & 174 & 174 & \cdots & 227 & 227 & 0 \\ 173 & 174 & 174 & & 227 & 227 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}$$

3. Konvolusi standar 3 × 3 dengan 2 *stride*. Ukuran citra pada *output layer zero-padding* sebelumnya adalah 225 × 225, hal itu dilakukan untuk menghasilkan citra *output* dengan ukuran setengah dari ukuran citra *input* sebelum dilakukan operasi *zero-padding*. Maka, konvolusi ini dilakukan dengan menggunakan *padding valid* atau tanpa *padding* sehingga akan menghasilkan citra dengan ukuran 112 × 112. Berikut contoh *kernel* yang digunakan untuk operasi konvolusi pada *channel* B', G', dan R':

$$K1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

$$K2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix},$$

$$K2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Sehingga menghasilkan matriks *output* $O_{1,total}$ secara keseluruhan yang merupakan hasil operasi konvolusi pada *channel output* pertama O_1. Persamaan 2.1 digunakan untuk mencari x_{out} dan y_{out} ketika perhitungan konvolusi menghasilkan piksel $O(0,0)$ dengan mengetahui hubungan antara ukuran *output* n_{out}, ukuran*input* n_{in}, ukuran *kernel* f, *stride* S, dan *padding* P seperti contoh berikut.

$$n_{out} = \frac{225 - 3 + 2.0}{2} + 1$$

$$n_{out} = 111 + 1$$

$$n_{out} = 112$$

$$O_{1,c1}(0,0) = \sum_{a=-1}^{1} \sum_{b=-1}^{1} I_1(1+a, 1+b) \times K1(a,b)$$

$$= (134 \times 0) + (135 \times 1) + (136 \times 0) + (131 \times 1) +$$

$$(132 \times 0) + (134 \times 1) + (127 \times 0) + (129 \times 1) + (133 \times 0)$$

$$= 254$$

$$O_{1,c1}(0,0) = \sum_{a=-1}^{1} \sum_{b=-1}^{1} I_1(1+a, 1+b) \times K1(a,b)$$

$$= (138 \times 0) + (139 \times 1) + (140 \times 0) + (135 \times 1) + (136 \times 0) + (138 \times 1) + (131 \times 0) + (133 \times 1) + (138 \times 0)$$

$$= 270$$

$$O_{1,c1}(0,0) = \sum_{a=-1}^{1} \sum_{b=-1}^{1} I_1(1+a, 1+b) \times K1(a,b)$$

$$= (149 \times 0) + (150 \times 1) + (151 \times 0) + (146 \times 1) + (147 \times 0) + (149 \times 1) + (142 \times 0) + (144 \times 1) + (147 \times 0)$$

$$= 312$$

$$c_{2out} = \begin{bmatrix} 836 & 1275 & 1293 & \cdots & 1464 & 1464 & 1464 \\ 1215 & 1669 & 1702 & \cdots & 1952 & 1952 & 1952 \\ 1227 & 1684 & 1711 & \cdots & 1952 & 1952 & 1952 \\ \cdots & \cdots & \cdots & & \cdots & \cdots & \cdots \\ 998 & 2011 & 2008 & \cdots & 2672 & 2672 & 2672 \\ 1007 & 2023 & 2008 & & 2672 & 2672 & 2672 \\ 1016 & 2032 & 2008 & \cdots & 2672 & 2672 & 2672 \end{bmatrix}$$

4. *Batch Normalization*. Normalisasi $BN_{\gamma,\beta}(x_i)$ dilakukan untuk setiap x_i dalam sebuah *mini- batch* pada satu *channel*. Pelatihan ini menggunakan *batch size* 8, namun, pada contoh perhitungan ini digunakan satu *mini-batch* berukuran sesuai dengan ukuran citra. Pertama, menghitung nilai *mean*/ rata-rata.

   $n_{out}(Weight) = 730$

   $$\mu = \frac{1}{64 \times 64} \sum_{i=1}^{64} \sum_{j=1}^{64} C_{1in_{i,j}}$$

   $$\mu = \frac{836 + 824 + 812 + \cdots - 688}{4096}$$

μ =2,312

$$\mu = \frac{1}{64 \times 64}\sum_{i=1}^{64}\sum_{j=1}^{64}(C_{1in_{i,j}} - 2{,}312)^2$$

$$= \frac{(836 - 2{,}312)^2 + (824 - 2{,}312)^2 + (812 - 2{,}312)^2 + \cdots (-688 - 2{,}312)^2}{4096}$$

$$= 5614{,}749$$

$$\mu = \frac{836 - 2{,}312}{\sqrt{5614{,}749 + 0{,}001}}$$

$$= 1{,}4487$$

$$y = \begin{bmatrix} -1{,}4887 & -1{,}0898 & -1{,}0751 & \cdots & -0{,}9353 & -0{,}9353 & -0{,}9353 \\ -1{,}1388 & -0{,}7677 & -0{,}7407 & \cdots & -0{,}5365 & -0{,}5365 & -0{,}5365 \\ -1{,}1290 & -0{,}7554 & -0{,}7334 & \cdots & -0{,}5365 & -0{,}5365 & -0{,}5365 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -1{,}3126 & -0{,}4863 & -0{,}4954 & \cdots & 0{,}0529 & 0{,}0529 & 0{,}0529 \\ -1{,}3053 & -0{,}4765 & -0{,}4887 & \cdots & 0{,}0529 & 0{,}0529 & 0{,}0529 \\ -1{,}2979 & -0{,}4692 & -0{,}4887 & \cdots & 0{,}0529 & 0{,}0529 & 0{,}0529 \end{bmatrix}$$

$$y = \begin{bmatrix} -2{,}8974 & -2{,}1796 & -2{,}1502 & \cdots & -1{,}8706 & -1{,}8706 & -1{,}8706 \\ -2{,}2777 & -1{,}5354 & -1{,}4815 & \cdots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ -2{,}2581 & -1{,}5109 & -1{,}4668 & \cdots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -2{,}6253 & -0{,}9726 & -0{,}9909 & \cdots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -2{,}6106 & -0{,}9530 & -0{,}9775 & \cdots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -2{,}5959 & -0{,}9384 & -0{,}9775 & \cdots & 0{,}1058 & 0{,}1058 & 0{,}1058 \end{bmatrix}$$

$$c_{2out} = \begin{bmatrix} -2{,}8974 & -2{,}1796 & -2{,}1502 & \cdots & -1{,}8706 & -1{,}8706 & -1{,}8706 \\ -2{,}2777 & -1{,}5354 & -1{,}4815 & \cdots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ -2{,}2581 & -1{,}5109 & -1{,}4668 & \cdots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -3{,}5814 & -0{,}9726 & -0{,}9775 & \cdots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -2{,}6106 & -0{,}9530 & -0{,}9775 & \cdots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -2{,}5959 & -0{,}9384 & -0{,}9775 & \cdots & 0{,}1058 & 0{,}1058 & 0{,}1058 \end{bmatrix}$$

5. Fungsi aktivasi *ReLU6*. Persamaan dibawah ini digunakan untuk menunjukkan contoh fungsi aktivasi *ReLU6* yang dilakukan pada dua *channel* hasil *Batch Normalization* yang menjadi *channel input* c_{1in}.

$$f\left(c_{1in_{0,0}}\right) = \min(\max(0, -2{,}8974), 6)$$

$$f(c_{1in}) = \begin{bmatrix} \min(\max(0, -2{,}8974), 6) & \dots & \min(\max(0, -1.8706), 6) \\ \vdots & \ddots & \vdots \\ \min(\max(0, -1.7774), 6) & \dots & \min(\max(0{,}0.1045), 6) \end{bmatrix}$$

$$f(c_{1in}) = \begin{bmatrix} -2{,}8974 & -2{,}1796 & -2{,}1502 & \dots & -1{,}8706 & -1{,}8706 & -1{,}8706 \\ -2{,}2777 & -1{,}5354 & -1{,}4815 & \dots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ -2{,}2581 & -1{,}5109 & -1{,}4668 & \dots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ -3{,}5814 & -0{,}9726 & -0{,}9775 & \dots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -2{,}6106 & -0{,}9530 & -0{,}9775 & \dots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -2{,}5959 & -0{,}9384 & -0{,}9775 & \dots & 0{,}1058 & 0{,}1058 & 0{,}1058 \end{bmatrix}$$

Hasil *output* dari fungsi aktivasi *ReLU6* yang dilakukan pada kedua *channel* c_{1in} dan c_{2in} adalah sebagai berikut:

$$C_{1out} = \begin{bmatrix} -2{,}8974 & -2{,}1796 & -2{,}1502 & \dots & -1{,}8706 & -1{,}8706 & -1{,}8706 \\ -2{,}2777 & -1{,}5354 & -1{,}4815 & \dots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ -2{,}2581 & -1{,}5109 & -1{,}4668 & \dots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ -3{,}5814 & -0{,}9726 & -0{,}9775 & \dots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -2{,}6106 & -0{,}9530 & -0{,}9775 & \dots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -2{,}5959 & -0{,}9384 & -0{,}9775 & \dots & 0{,}1058 & 0{,}1058 & 0{,}1058 \end{bmatrix}$$

$$C_{2out} = \begin{bmatrix} -2{,}8974 & -2{,}1796 & -2{,}1502 & \dots & -1{,}8706 & -1{,}8706 & -1{,}8706 \\ -2{,}2777 & -1{,}5354 & -1{,}4815 & \dots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ -2{,}2581 & -1{,}5109 & -1{,}4668 & \dots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ -3{,}5814 & -0{,}9726 & -0{,}9775 & \dots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -2{,}6106 & -0{,}9530 & -0{,}9775 & \dots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -2{,}5959 & -0{,}9384 & -0{,}9775 & \dots & 0{,}1058 & 0{,}1058 & 0{,}1058 \end{bmatrix}$$

6. Konvolusi *Depthwise* dengan ukuran kernel 3 x 3 dengan *stride* 1 dan *padding same*. Pada *layer* konvolusi *depthwise*, *channel* hasil *ReLU6* menjadi *channel input* I_i. Berikut adalah contoh dua kernel K_1 dan K_2 yang digunakan pada contoh operasi konvolusi *depthwise*:

$$n_{out} = \frac{112 - 3 + 2}{1} + 1$$

$$n_{out} = 112$$

$$O_{1,c1}(0,0) = \sum_{a=-1}^{1} \sum_{b=-1}^{1} I_1(0 + a, 0 + b) \times K1(a, b)$$

$$= (0 \times 0) + (1 \times 0) + (0 \times 0) + (1 \times 0) +$$

$$(-4 \times 2{,}8974) + (1 \times -2{,}1796) + (0 \times 0) + (1 \times 2{,}2777)$$

$$= 7{,}1323$$

$$O_1 = \begin{bmatrix} 7{,}1323 & 2{,}1355 & 2{,}7748 & \cdots & -1{,}464 & -1{,}8706 & -1{,}8706 \\ 2{,}4200 & -1{,}3080 & -1{,}3080 & \cdots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ 3{,}0495 & -7{,}3741 & -1{,}0000 & \cdots & -1{,}0727 & -1{,}0727 & -1{,}0727 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 4{,}2976 & -1{,}6380 & -0{,}9775 & \cdots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -4{,}2682 & -1{,}6896 & -0{,}9775 & \cdots & 0{,}1058 & 0{,}1058 & 0{,}1058 \\ -6{,}8348 & -7{,}7306 & -0{,}9775 & \cdots & 0{,}1058 & 0{,}1058 & 0{,}1058 \end{bmatrix}$$

7. Konvolusi *Pointwise*. Konvolusi *pointwise* menggunakan operasi konvolusi biasa dengan ukuran *kernel* 1 × 1 dengan *stride* 1 dan tanpa *padding*. Pada *layer* konvolusi *pointwise*, *channel output* hasil konvolusi *depthwise* menjadi *channel input* I_i. Berikut adalah contoh dua kernel K_1 dan K_2 yang digunakan pada contoh operasi konvolusi *pointwise.*

$K_1 = (1), \qquad K_2 = (2),$

$$n_{out} = \frac{112 - 1 + 0}{1} + 1$$

$$n_{out} = 112$$

Contoh operasi konvolusi *pointwise* yang dilakukan pada *channel input* I_1 dengan *kernel* K_1

$$O_{1,c1}(0,0) = I_1(0,0) \times K_1$$

$$= 7{,}3123 \times 1$$

$$= 7{,}3123$$

Contoh operasi konvolusi *pointwise* yang dilakukan pada *channel input* I_1 dengan *kernel* K_2

$$O_{1,c2}(0,0) = I_1(0,0) \times K_2$$

$$= 7{,}3123 \times 2$$

$$= 14{,}6246$$

Kemudian hasil konvolusi dijumlahkan sesuai dengan posisi matriks. Contoh penjumlahan tersebut dijabarkan sebagai berikut:

$$O_{1total}(0,0) = 7{,}3123 + 14{,}6246$$

$$= 23{,}728$$

$$O_1 = \begin{bmatrix} 23{,}728 & 6{,}7744 & 9{,}1923 & \cdots & 8{,}9856 & 8{,}9483 & 15{,}028 \\ 9{,}4035 & -5{,}3768 & -2{,}7834 & \cdots & -2{,}8236 & -2{,}8686 & 3{,}4245 \\ 11{,}187 & -3{,}2381 & -4{,}2624 & \cdots & 0{,}0450 & 0 & 3{,}211 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 10{,}623 & -3{,}7339 & 0{,}2665 & \cdots & 0{,}0252 & 0 & -0{,}3161 \\ 10{,}049 & -3{,}8914 & 0{,}2048 & \cdots & 0{,}0252 & 0 & -0{,}3161 \\ -17{,}039 & -1{,}1585 & 3{,}1464 & \cdots & -0{,}2936 & 0{,}3161 & 0{,}6327 \end{bmatrix}$$

8. *Inverted Residual Block.* Menggunakan konvolusi 1x1, lalu menggunakan konvolusi 3x3, kemudian menggnakan konvolusi 1x1 lagi untuk mengurangi (*project*) dimensi *channel* kembali ke jumlah awal, lalu *skip connection* menghubungkan input langsung ke *output* jika jumlah *channel input* dan *output* sama dan *stride* = 1.

Sehingga menghasilkan matriks *output* $O_{1,total}$ secara keseluruhan yang merupakan hasil operasi konvolusi pada *channel output* pertama O_1:

$$O_1 = \begin{bmatrix} 20{,}823 & 8{,}6143 & 2{,}5455 & \cdots & 4{,}5426 & 1{,}3062 & 10{,}355 \\ 2{,}5723 & -4{,}6086 & -9{,}5417 & \cdots & -7{,}8289 & -9{,}9504 & 0{,}0985 \\ 9{,}9086 & 1{,}0508 & -1{,}9296 & \cdots & -1{,}4936 & -4{,}0368 & -4{,}7138 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 11{,}8209 & -2{,}4284 & 2{,}1176 & \cdots & 0{,}3573 & 0{,}5959 & -0{,}3672 \\ 11{,}0860 & -4{,}6339 & 2{,}7993 & \cdots & 0{,}4931 & 1{,}3427 & -0{,}0654 \\ 19{,}6975 & 2{,}3865 & 12{,}9156 & \cdots & -0{,}1084 & -0{,}0782 & -0{,}4211 \end{bmatrix}$$

3.5 Pengembangan Aplikasi Desktop

Setelah model klasifikasi selesai dilatih, langkah selanjutnya adalah mengintegrasikannya ke dalam aplikasi desktop berbasis **Tkinter**. Aplikasi ini dirancang untuk memudahkan pengguna dalam mengunggah gambar daun padi dan mendapatkan hasil klasifikasi penyakit secara langsung.:

Langkah-langkah dalam pengembangan aplikasi desktop:

1. **Membangun Antarmuka Pengguna (GUI) dengan Tkinter**

   Tkinter digunakan untuk membangun antarmuka yang sederhana dan intuitif. Antarmuka ini memiliki beberapa komponen utama, yaitu:

   a) ***Button*** untuk mengunggah gambar daun padi.

   b) ***Label*** untuk menampilkan hasil klasifikasi penyakit.

   c) ***Canvas*** untuk menampilkan gambar yang diunggah.

   d) ***Textbox*** untuk menampilkan informasi tentang penyakit yang terdeteksi.

2. **Memuat Model CNN**

   Setelah aplikasi berjalan, model *MobileNetV2* yang telah dilatih dimuat menggunakan fungsi load_model() dari Keras. Model ini akan digunakan untuk melakukan prediksi pada citra daun yang diunggah oleh pengguna.

Source Code 3.1 Memuat Model CNN

```
from tensorflow.keras.models import load_model
model = load_model('model_penyakit_padi.h5')
```

3. **Proses Klasifikasi Gambar**

   Ketika pengguna mengunggah gambar daun padi, aplikasi akan melakukan preprocessing pada gambar, termasuk mengubah ukuran gambar menjadi 224 x 224 piksel dan normalisasi nilai piksel. Setelah itu, gambar tersebut diberikan sebagai input ke model untuk diprediksi.

Source Code 3.2 Proses Klasifikasi Gambar

```
def klasifikasikan_gambar(gambar):
    # Ubah ukuran gambar
    gambar = gambar.resize((224, 224))
    # Konversi ke array
    img_array = np.array(gambar)
    # Normalisasi
    img_array = img_array / 255.0
    # Tambahkan dimensi batch
    img_array = np.expand_dims(img_array, axis=0)
    # Prediksi dengan model
    hasil_prediksi = model.predict(img_array)
    return hasil_prediksi
```

4. **Menampilkan Hasil Klasifikasi**

   Hasil prediksi yang diberikan oleh model berupa probabilitas untuk setiap kelas penyakit. Aplikasi akan menampilkan kelas dengan probabilitas tertinggi sebagai hasil klasifikasi kepada pengguna.

3.5 Pengujian Model

Proses pengujian model *MobileNetV2* dilakukan pada platform *web Google Colaboratory*. Setelah meneyelesaikan pelatihan di setiap skenario, model dilakukan pengujian untuk mengetahui kemampuan hasil klasifikasi dari model tersebut. Pengukuran skor yang digunakan untuk mengevaluasi kemampuan model *MobileNetV2* adalah *f1-score* dan akurasi pelatihan. Langkah – langkah pengujian dilakukan sebagai berikut:

1. Memuat model *MobileNetV2* yang telah dilatih dan data uji.
2. Melakukan pembelajaran maju model *MobileNetV2*
3. Menghitung *f1-score,* akurasi pelatihan, *precision, recall,* dan *confustion matrix* pada citra
4. Menampilkan rata-rata keseluruhan *f1-score* dan akurasi pelatihan

3.5 Pengujian Aplikasi *Desktop*

Pengujian aplikasi desktop dilakukan untuk memastikan bahwa aplikasi yang telah dikembangkan dapat bekerja dengan baik dalam berbagai kondisi penggunaan. Pengujian ini mencakup:

a. Fungsionalitas

   Pengujian fungsionalitas bertujuan untuk memastikan bahwa semua fitur aplikasi

berfungsi dengan baik sesuai dengan tujuan. Pengujian dilakukan dengan beberapa skenario, seperti:

a. Mengunggah gambar daun padi yang terinfeksi penyakit (blast, blight, tungro, atau brown spot).
b. Memastikan hasil klasifikasi muncul setelah gambar diunggah.
c. Memastikan bahwa aplikasi tidak mengalami error saat dijalankan.

b. Pengujian GUI

Pengujian GUI dilakukan untuk memastikan antarmuka pengguna bekerja dengan baik dan mudah digunakan. Beberapa aspek yang diuji antara lain:

a. Kesesuaian tata letak komponen GUI (tombol, label, canvas) dengan tampilan aplikasi.
b. Responsivitas aplikasi saat penggun mengunggah gambar atau mengklik tombol.
c. Keterbacaan teks dan hasil klasifikasi yang ditampilkan pada layer.

3.7 Evaluasi

Evaluasi model *MobileNetV2* dilakukan setelah mendapatkan *hyper-parameter* pada hasil pengujian. Evaluasi dilakukan untuk mengetahui bagaimana kemampuan model *MobileNetV2* dengan *hyper-parameter* terbaik di dalam melakukan klasifikasi penyakit pada tanaman padi terhadap *dataset*. Proses evaluasi model *MobileNetV2* dengan penentuan *hyper-parameter* terbaik akan dijabarkan pada Bab IV.

BAB IV
HASIL DAN PEMBAHASAN

Bab ini berisikan pembahasan mengenai hasil implementasi metode penelitian yang telah diterapkan dalam penelitian ini, yaitu klasifikasi penyakit pada tanaman padi melalui citra daun menggunakan algoritma CNN berbasis *MobileNetV2*.

4.1 Lingkungan dan Perangkat yang Digunakan untuk Penelitian

Penelitian ini dilakukan di lingkungan *Jupyter Notebook* pada platform *web Google Colaboratory*. Spesifikasi perangkat keras pada layanan yang disediakan *Google Colaboratory* secarasingkat dapat dilihat pada Tabel 4.1. Spesifikasi layanan tersebut merupakan jenis layanan perangkat keras virtual yang dirancang untuk mendukung berbagai kebutuhan komputasi di platform berbasis cloud. Layanan ini menyediakan akses ke sumber daya komputasi seperti CPU, GPU, dan RAM yang dapat digunakan secara efisien oleh pengguna untuk menjalankan berbagai aplikasi, termasuk analisis data, pelatihan model machine learning, dan eksperimen komputasi lainnya. Meskipun layanan ini sangat bermanfaat bagi pengguna yang membutuhkan sumber daya komputasi yang tinggi, seperti dalam pelatihan jaringan saraf atau pengolahan citra, layanan ini juga memiliki batasan waktu tertentu per sesi. Setelah sesi berakhir, semua data yang tidak disimpan akan terhapus, dan pengguna harus memulai sesi baru untuk melanjutkan pekerjaan mereka. Setiap pengguna dapat menggunakan layanan tersebut dengan dibatasi waktu tertentu per sesi, setelah itu akan dihapus dan harus dimulai ulang dengan sesi yang baru.

Tabel 4.1. Spesifikasi Singkat Perangkat Keras Layanan yang Disediakan *Google Colaboratory*

Kode Perolehan Informasi	Nama Perangkat	Spesifikasi Perangkat
`!nvidia-smi`	GPU	1 × Nvidia Tesla K80, 2496 core CUDA, 12GB(11,439GB) GDDR5 VRAM
`!nvidia-smi`	Versi Driver	410,79
`!nvidia-smi`	Versi CUDA	10,0
`!lscpu`	CPU	Intel® Xeon® CPU @ 2,30GHz
`!cat /proc/meminfo`	RAM	12,6GB
`!df -h`	Disk	320GB

Dataset pada penelitian ini disimpan di penyimpanan awan *Google Drive.* Contoh data citra untuk *testing* disimpan di dalam penyimpanan awan tersebut. Layanan penyimpanan awan *Google Drive* disediakan secara gratis oleh *Google* dengan kapasistas penyimpanan sebesar 15 GB. Penyimpanan awan tersebut dapat dimuat pada *Google Colaboratory.*

Adapun perangkat keras yang digunakan untuk menjalankan lingkungan *Google Colaboratory* pada penelitian ini adalah satu buah Laptop dengan spesifikasi sebagai berikut:

1. Merek : HP Laptop 14s – dk0xxx
2. Layar : 14 Inch
3. Prosesor : AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx (8CPUs), ~2.1GHz
4. RAM : 8GB
5. Hard Drive : 1TB HDD 5400 rpm
6. Sistem Operasi : Windows 11

4.2 Pra-Pemrosesan Data

Tahapan setelah pengumpulan data adalah melakukan pra-pemrosesan data. Pertama, gambar dimuat dari direktori sesuai dengan kategori penyakit tanaman padi. Kemudian, setiap gambar diubah ukurannya menjadi 224 x 224 piksel untuk memenuhi kebutuhan model. Ukuran gambar yang seragam diperlukan agar model dapat menerima input dengan dimensi yang tetap, baik saat proses pelatihan maupun saat digunakan untuk membuat prediksi. Hal ini sangat penting karena model deep learning, seperti CNN (Convolutional Neural Networks), memerlukan input dengan ukuran yang konsisten untuk dapat memproses data secara efisien. Jika gambar yang dimasukkan memiliki ukuran yang bervariasi, model akan kesulitan dalam memproses data dan mengidentifikasi pola-pola yang ada dalam gambar. Dengan memastikan bahwa semua gambar diubah ukurannya menjadi 224 x 224 piksel, kita menciptakan struktur input yang seragam, yang memungkinkan model untuk belajar fitur yang lebih baik dan lebih konsisten dari data latih. Selain itu, ukuran gambar yang seragam juga memudahkan dalam pengaturan batch input selama proses pelatihan, di mana setiap

batch akan memiliki dimensi yang sama, memungkinkan pemrosesan paralel yang lebih efisien dan penghematan waktu komputasi.

Ukuran gambar yang seragam juga mengoptimalkan penggunaan sumber daya komputasi, karena model dapat mengalokasikan memori dan waktu pemrosesan dengan lebih efisien. Dalam konteks ini, ukuran gambar yang umumnya digunakan adalah 224 x 224 piksel, yang merupakan ukuran standar untuk banyak arsitektur CNN populer, seperti MobileNetV2. Dengan mengubah ukuran gambar ke dimensi yang seragam ini, model dapat memproses data lebih efisien, karena input yang lebih kecil mengurangi jumlah parameter yang harus diproses pada setiap lapisan jaringan. Ini memungkinkan model untuk memanfaatkan sumber daya komputasi secara lebih maksimal, mengurangi waktu pelatihan, serta meningkatkan efisiensi saat menerapkan model pada perangkat dengan kemampuan komputasi terbatas. Selain perubahan ukuran gambar, proses selanjutnya adalah normalisasi nilai piksel gambar. Normalisasi ini penting untuk memastikan bahwa rentang nilai piksel gambar berada dalam rentang yang seragam, biasanya antara 0 hingga 1. Ini dicapai dengan membagi nilai setiap piksel dengan 255 (nilai maksimum piksel dalam citra RGB).

4.3 Skenario 1: Menggunakan Kombinasi *Image Augmentation*, *Batch Size*, dan E*pochs*

Skenario pertama dalam menentukan *hyperparameter* terbaik menggunakan *augmentation* dan *batch size*. Nilai setiap *hyperparameter* diambil secara manual, nilai *epochs* diambil dari yang terendah hingga tertinggi, yaitu 50, 100, 200, nilai *batch size* yang diambil dari yang terendah hingga tertinggi, yaitu 32, 64, 128, dan jenis *image augmentation* yang dipakai dalam skenario ini adalah rotasi, *zoom, scaling, width shift, height shift,* dan *flip*. Semua jenis *image augmentation* dikali sesuai angka yang terdapat dalam kolom *image augmentation*.

4.4 Analisis dan Pembahasan Hasil Skenario 1

Hasil pelatihan model *MobileNetV2* menggunakan kombinasi *image augmentation* dan *epochs* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil Skenario 1

Eksperimen	*Image Augmentation*	*Epochs*	*Batch Size*	*F1-Score*	Akurasi
1	1	10	32	0,57	0,99
			64	0,62	0,99
			128	0,75	0,98
		20	32	0,25	0,98
			64	0,34	0,98
			128	0,46	0,99
		30	32	0,84	0,98
			64	0,78	0,98
			128	0,91	0,98
2	2	10	32	0,99	0,99
			64	0,97	0,98
			128	0,95	0,99
		20	32	0,91	0,99
			64	0,93	0,98
			128	0,95	0,99
		30	32	0,95	0,99
			64	0,96	0,99
			128	0,99	0,98
3	3	10	32	0,67	0,88
			64	0,75	0,88
			128	0,66	0,85
		20	32	0,69	0,86
			64	0,66	0,85
			128	0,72	0,79
		30	32	0,99	0,99
			64	0,99	0,98
			128	0,99	0,98

Berdasarkan tabel hasil eksperimen di atas, dapat dianalisis hubungan antara *image augmentation*, jumlah *epochs*, *f1-score*, dan akurasi dalam skenario yang berbeda. Pengaruh jumlah *epochs* pada akurasi dapat dilihat bahwa tidak ada perubahan yang signifikan ketika jumlah *epochs* diubah dan pengaruh jumlah *epochs* pada *f1-score* dapat dilihat bahwa ada peerubahan, semakin tinggi jumlah *epochs*, maka semakin tinggi *f1-score*nya. Pengaruh jumlah *image augmentation* dan *batch size* pada akurasi dapat dilihat juga bahwa tidak ada perubahan yang signifikan ketika jumlah *image augmentation* dan *batch size* ditambah. Pengaruh jumlah *image*

augmentation pada *f1-score* dapat dilihat ketika *image augmentation* bernilai 2, rata-rata nilai *f1-score* lebih tinggi dari rata-rata nilai *f1-score* ketika *image augmentation* bernilai 1 dan 3. Pengaruh jumlah *batch size* pada *f1-score* dapat dilihat bahwa semakin tinggi nilai *batch size,* maka semakin tinggi nilai *f1-score*nya. *F1-score* dan akurasi tertinggi tercapai dengan *image augmentation* bernilai 3 , *epochs* bernilai 30 dan nilai *batch size* sebesar 32.

4.5 Skenario 2: Menggunakan Kombinasi *Batch Size* dan *Epochs* Tanpa *image Augmentation*

Skenario kedua dalam menentukan *hyperparameter* terbaik menggunakan *augmentation* dan *batch size*. Nilai setiap *hyperparameter* diambil secara manual, nilai *epochs* diambil dari yang terendah hingga tertinggi, yaitu 50, 100, 200, dan nilai *batch size* diambil dari yang terendah hingga tertinggi, yaitu 32, 64, 128.

4.6 Analisis dan Pembahasan Hasil Skenario 2

Hasil pelatihan model *MobileNetV2* menggunakan kombinasi *Batch size* dan *epochs* dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil Skenario 2

Eksperimen	*Epochs*	*Batch Size*	*F1-Score*	Akurasi
1	10	32	0,16	0,98
		64	0,50	0,98
		128	0,37	0,98
2	20	32	0,56	0,98
		64	0,17	0,97
		128	0,20	0,99
3	30	32	0,66	0,99
		64	0,61	0,99
		128	0,34	0,99

Berdasarkan tabel hasil eksperimen di atas, dapat dianalisis hubungan antara *batch size*, jumlah *epochs*, *f1-score*, dan Akurasi dalam skenario yang berbeda. Pengaruh jumlah *epochs* pada akurasi dapat dilihat bahwa tidak ada perubahan yang signifikan ketika jumlah *epochs* diubah dan pengaruh jumlah *epochs* pada *f1-score* dapat dilihat di eksperimen 1 dan 2 bahwa semakin tinggi jumlah *epochs,* maka semakin tinggi *f1-score*nya. Pengaruh jumlah *batch size* pada akurasi dapat dilihat

juga bahwa tidak ada perubahan yang signifikan ketika *batch size* ditambah. Pengaruh jumlah *batch size* pada *f1-score* dapat dilihat bahwa semakin tinggi nilai *batch size,* maka semakin rendah rata-rata nilai *f1-score*nya. *f1-score* dan akurasi tertinggi tercapai dengan *batch size* bernilai 32 dan *epochs* bernilai 30.

4.7 Model Terbaik

Berdasarkan analisis yang dilakukan pada dua skenario yang berbeda, model terbaik diperoleh pada skenario pertama dengan kombinasi *image augmentation* sebesar 3, *epochs* sebanyak 30, dan *batch size* sebesar 32. Kombinasi ini menghasilkan *f1-score* dan akurasi tertinggi sebesar 0.99 Pengaruh dari penambahan *image augmentation* menunjukkan bahwa ketika *image augmentation* bernilai 3, model dapat mencapai *f1-score* dan akurasi yang optimal. Hal ini menunjukkan bahwa variasi data melalui augmentasi gambar membantu model dalam menggeneralisasi dan meningkatkan performa. Sementara itu, jumlah *epochs* yang lebih tinggi juga memberikan kontribusi positif terhadap peningkatan *f1-score,* menunjukkan bahwa model memiliki waktu yang cukup untuk belajar dari pola data yang lebih dalam skenario kedua, yang memanfaatkan kombinasi *batch size* dan *epochs* tanpa *image augmentation,* menunjukkan bahwa model terbaik dengan *batch size* 32 dan *epochs* 30 memberikan *f1-score* dan akurasi yang mendekati hasil dari skenario pertama. Namun, tanpa *image augmentation*, variasi dalam data pelatihan tidak seoptimal skenario pertama.

4.8 Pengujian Pada Data Uji

Setelah menentukan model terbaik menggunakan algoritma CNN berbasis *MobileNetV2,* model tersebut diuji pada data uji untuk mengevaluasi kemampuan generalisasinya. Model ini memuat data uji yang merupakan 10% dari keseluruhan *dataset* yang belum pernah dilihat selama pelatihan. Proses pengujian melibatkan prediksi klasifikasi penyakit pada tanaman padi dan evaluasi kinerja model dengan menghitung metrik akurasi dan *f1-score*. Hasil pengujian menunjukkan bahwa model mencapai akurasi 100% dan *f1-score* sebesar 1, mengindikasikan kemampuan model yang sangat baik dalam mengklasifikasikan penyakit bahkan pada data baru. Tingginya nilai akurasi dan *f1-score* menunjukkan bahwa model ini memiliki kemampuan

generalisasi yang kuat, menjadikannya alat yang andal untuk mendeteksi penyakit pada tanaman padi di lapangan.

4.9 *Confusion Matrix*

Berikut adalah *confusion matrix* dari hasil pengujian pada data uji:

$$\begin{bmatrix} 183 & 0 & 0 & 0 \\ 0 & 162 & 0 & 0 \\ 0 & 0 & 148 & 0 \\ 0 & 0 & 0 & 152 \end{bmatrix}$$

Confusion matrix yang ditampilkan dalam hasil pengujian model *MobileNetV2* untuk identifikasi penyakit pada tanaman padi mencerminkan empat kelas penyakit yang berbeda, yaitu *blast*, *blight*, *tungro*, dan *brownspot*. Matriks berformat 4x4 ini memperlihatkan kinerja model yang optimal dalam klasifikasi penyakit. Nilai-nilai pada diagonal utama matriks, yang berurutan adalah 183 untuk *blast*, 162 untuk *blight*, 148 untuk *tungro*, dan 152 untuk *brownspot*, menunjukkan jumlah sampel yang secara akurat diklasifikasikan sesuai kelasnya. Elemen non-diagonal, yang semuanya bernilai 0, menunjukkan bahwa tidak terdapat kesalahan dalam mengklasifikasikan sampel antar kelas. Hasil ini mengindikasikan bahwa model memiliki kemampuan prediksi yang sempurna dengan tingkat akurasi 100% dalam pengujian.

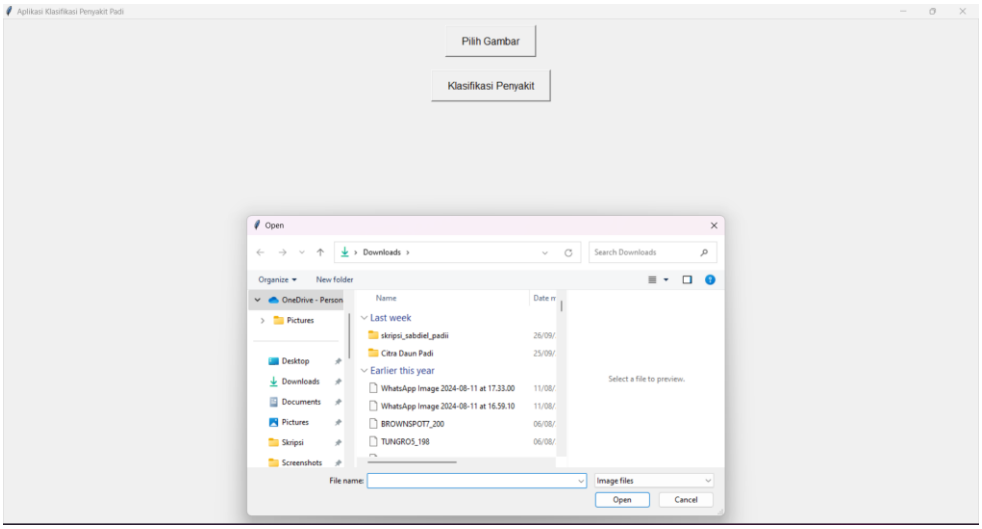
4.10 Deskripsi Fitur Aplikasi Desktop

Aplikasi desktop yang dikembangkan dalam penelitian ini dirancang untuk membantu pengguna dalam mengklasifikasikan penyakit pada tanaman padi melalui citra daun padi. Aplikasi ini mengintegrasikan model CNN berbasis *MobileNetV2* yang telah dilatih dan diimplementasikan dalam antarmuka pengguna grafis (GUI) menggunakan ***Tkinter***. Berikut adalah deskripsi fitur utama yang terdapat dalam aplikasi desktop ini:

1. Unggah Gambar

   Fitur ini memungkinkan pengguna untuk mengunggah citra daun padi yang akan diklasifikasikan. Setelah tombol "Pilih Gambar" diklik, pengguna dapat memilih file gambar dari perangkat lokal mereka. Gambar yang diunggah akan otomatis ditampilkan di canvas aplikasi.

1. Komponen: Tombol "Pilih Gambar".
2. Fungsi: Membuka jendela dialog untuk memilih gambar daun padi dari penyimpanan lokal.

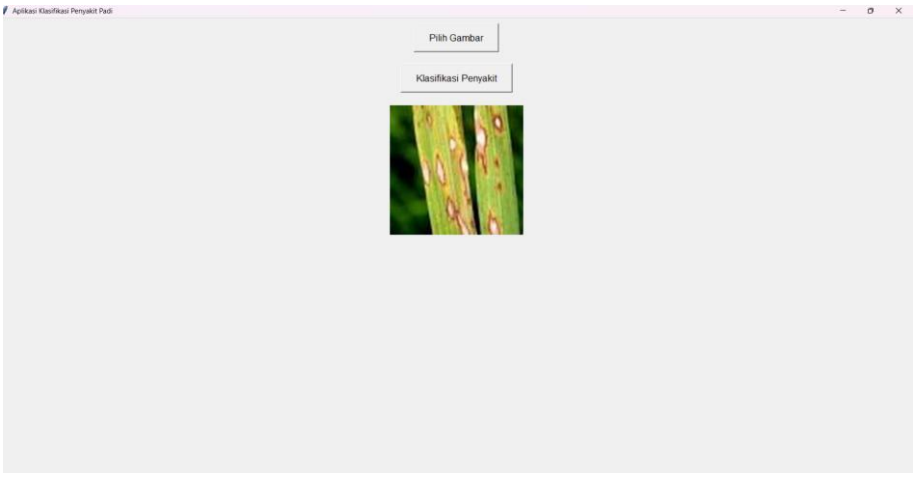


Gambar 4.1 Tampilan Unggah Gambar

2. Tampilan Gambar yang Diunggah

   Setelah gambar diunggah, aplikasi menampilkan gambar tersebut pada bagian ***Canvas***. Ini membantu pengguna memastikan bahwa gambar yang mereka unggah benar dan siap untuk diproses oleh model.

   1. Komponen: *Canvas*.
   2. Fungsi: Menampilkan citra daun padi yang telah diunggah oleh pengguna.



Gambar 4.2 Tampilan Gambar Yang Diunggah

3. Klasifikasi Penyakit

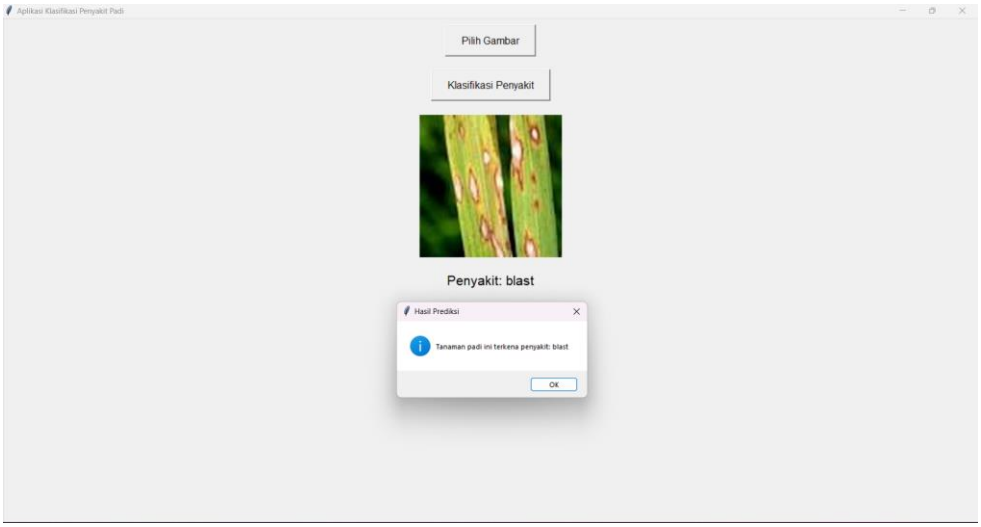
   Setelah gambar diunggah, aplikasi secara otomatis menjalankan model klasifikasi CNN MobileNetV2 untuk mendeteksi penyakit pada daun padi. Hasil klasifikasi berupa nama penyakit yang terdeteksi akan ditampilkan pada layar, beserta probabilitas atau kepercayaan model terhadap prediksi tersebut.

   1. Komponen: Tombol “Klasifikasi Penyakit”.
   2. Fungsi: Menampilkan nama penyakit yang terdeteksi berdasarkan hasil prediksi model.

4. Informasi Penyakit

   Setelah klasifikasi selesai, aplikasi memberikan informasi tambahan tentang penyakit yang terdeteksi. Informasi ini mencakup penjelasan singkat mengenai penyakit tersebut dan gejala-gejalanya pada tanaman padi.

   1. Komponen: *Textbox* informasi penyakit.
   2. Fungsi: Memberikan penjelasan mengenai penyakit yang terdeteksi untuk memberikan pemahaman lebih lanjut kepada pengguna.



Gambar 4.3 Tampilan Informasi Penyakit

4.11 Hasil Pengujian Aplikasi Desktop

Setelah pengembangan aplikasi desktop selesai, dilakukan pengujian untuk memastikan bahwa aplikasi dapat berfungsi dengan baik dan memberikan hasil yang

akurat serta sesuai dengan tujuan penelitian. Berikut adalah hasil pengujian aplikasi desktop berdasarkan beberapa skenario pengujian yang dilakukan:

4.11.1 Pengujian Fungsionalitas

Pengujian ini dilakukan untuk memverifikasi bahwa seluruh fitur dalam aplikasi berjalan dengan baik sesuai dengan yang diharapkan. Hasil pengujian menunjukkan bahwa semua fitur utama aplikasi, termasuk fitur unggah gambar, tampilan gambar, klasifikasi, dan penampilan informasi penyakit, berfungsi dengan baik. Ketika pengguna mengunggah gambar daun padi yang terinfeksi penyakit, model CNN berhasil melakukan klasifikasi dan menampilkan hasil dengan akurasi yang baik.

4.11.2 Pengujian GUI (*Graphical User Interface*)

Pengujian GUI dilakukan untuk memastikan bahwa antarmuka aplikasi mudah digunakan dan komponen GUI terlihat jelas serta berfungsi dengan baik. Hasil pengujian menunjukkan bahwa komponen - komponen seperti tombol unggah, canvas untuk menampilkan gambar, dan hasil klasifikasi ditata dengan baik, sehingga pengguna dapat dengan mudah berinteraksi dengan aplikasi.

4.11.3 Pengujian Kinerja

Pengujian kinerja dilakukan untuk mengukur seberapa cepat aplikasi dapat melakukan klasifikasi setelah pengguna mengunggah gambar daun padi. Hasil pengujian menunjukkan bahwa aplikasi dapat melakukan proses klasifikasi dengan waktu respon kurang dari 2 detik per gambar.

BAB V
KESIMPULAN DAN SARAN

Bab ini menyajikan kesimpulan dan saran terkait hasil implementasi metode penelitian yang telah diterapkan dalam penelitian ini, yaitu klasifikasi penyakit pada tanaman padi menggunakan algoritma CNN berbasis *MobileNetV2*

5.1 Kesimpulan

Dari penelitian ini dapat disimpulkan beberapa hal penting terkait kombinasi *hyperparameter* dan performa model. Pada skenario 1, yang menggabungkan *image augmentation* dan *epochs*, kombinasi terbaik ditemukan pada *image augmentation* sebesar 3, 10 *epochs* dan 128 *batch size*, menghasilkan *f1-score* dan akurasi masing-masing sebesar 0,99. Peningkatan jumlah *epochs* dalam eksperimen ini cenderung meningkatkan *f1-score*, terutama dalam eksperimen dengan *image augmentation* sebesar 1 dan 3, meskipun ada penurunan performa pada beberapa titik. Dalam skenario 2, yang mengkombinasikan *Batch size* dan *epochs* tanpa *image augmentation*. Secara keseluruhan, skenario 1 memberikan hasil terbaik dengan *f1-score* dan akurasi tertinggi masing - masing sebesar 0,99, menjadikannya lebih efektif dalam meningkatkan performa model dibandingkan dengan skenario 2. Penyesuaian *hyperparameter* yang tepat dalam skenario 1 menunjukkan bahwa kombinasi *image augmentation*. *epochs*, dan *batch size* yang optimal dapat meningkatkan kinerja model secara signifikan.

5.2 Saran

Saran-saran yang dapat dilaksanakan terkait klasifikasi penyakit pada tanaman padi menggunakan algoritma CNN berbasis *MobileNetV2*

1. Perluasan dataset dengan menambahkan lebih banyak citra daun dari berbagai kondisi pencahayaan, sudut, dan lingkungan.
2. Menerapkan teknik augmentasi data untuk meningkatkan performa model tanpa menambah dataset.
3. Aplikasi model ke perangkat mobile sehingga lebih mudah digunakan oleh petani di lapangan.

DAFTAR PUSTAKA

Ardiansyah, A. S., & Nugroho, A. (2023). Klasifikasi Penyakit Daun Kopi Dengan Arsitektur MobileNetV2. *Jurnal Ilmu Komputer Dan Bisnis*, *14*(1), 66–73. https://doi.org/10.47927/jikb.v14i1.622

Bustamin Prodi Teknik Mesin, S., & Teknologi Industri Dewantara Palopo, A. (2021). Aplikasi Dekstop Multi Platform untuk Redis Client Framework Electron JS dan React JS. *DEWANTARA. J. Tech*, *02*(01).

Cahya Hutama, R., & Titi Komalasari, R. (2021). *STRING (Satuan Tulisan Riset dan Inovasi Teknologi) APLIKASI CHATBOT BERBASIS TEKS MENGGUNAKAN ALGORITMA NAIVE BAYES CLASSIFIER FAQ GRABADS*.

Dwi, A., Alwy, P., Syahid, M., Wahid, N., Naufal, B., Ag, N., & Fakhri, M. M. (2023). Klasifikasi Penyakit Pada Padi Dengan Ekstraksi Fitur LBP dan GLCM. *Journal of Deep Learning, Computer Vision and Digital Image Processing*, *1*(1), 1–10. https://journal.diginus.id/index.php/decoding

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Cambridge, MA: MIT Press, hlm. 17-50

Mumuni, A., & Mumuni, F. (2022). Data augmentation: A comprehensive survey of modern approaches. In *Array*, 16-22, (Vol. 16). Elsevier B.V. https://doi.org/10.1016/j.array.2022.100258

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. https://arxiv.org/abs/1801.04381

Singh, D., & Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, *97*. https://doi.org/10.1016/j.asoc.2019.105524

Teresia Ompusunggu, P. (2022). Klasifikasi Penyakit Tanaman Pada Daun Kentang Dengan Metode Convolutional Neural Network Arsitektur Mobilenet. *Jurnal Syntax Fusion*, *2*(09), 740–751. https://doi.org/10.54543/fusion.v2i09.217

Vabalas, A., Gowen, E., Poliakoff, E., & Casson, A. J. (2019). Machine learning algorithm validation with a limited sample size. *PLoS ONE*, *14*(11). https://doi.org/10.1371/journal.pone.0224365

Virgantara Putra, O., Zaim Mustaqim, M., Muriatmoko, D., Teknik Informatika, J., & Sains dan Teknologi, F. (2023). *Transfer Learning untuk Klasifikasi*

Penyakit dan Hama Padi Menggunakan MobileNetV2 Transfer Learning for Rice Disease and Pest Classification using MobileNetV2 (Vol. 22, pp. 20-34).

LAMPIRAN-LAMPIRAN

LAMPIRAN 1. *Source Code*

1. *Source Code* 1: Pra-pemrosesan Data

Source Code 1:
Pra-pemrosesan Data

```
import os
import numpy as np
from tensorflow.keras.preprocessing import image

# Mendefinisikan kategori penyakit padi
Categories = ['blast', 'blight', 'tungro', 'brownspot']
flat_data_arr = []
target_arr = []

datadir = '/content/drive/MyDrive/Skripsi/Citra Daun Padi'

for category in Categories:
    print(f'memuat... kategori: {category}')
    path = os.path.join(datadir, category)
    category_count = 0
    for img in os.listdir(path):
        img_array = image.load_img(os.path.join(path, img),
target_size=(224, 224)) # Mengubah ukuran gambar
        img_array = image.img_to_array(img_array) # Mengubah gambar
menjadi array
        flat_data_arr.append(img_array)
        target_arr.append(Categories.index(category))
        category_count += 1
    print(f'loaded category: {category} BERHASIL, Jumlah Data:
{category_count}')

# Mengonversi data menjadi array numpy
flat_data = np.array(flat_data_arr)
target = np.array(target_arr)
```

2. *Source Code* 2: Pembagian Data

Source Code 2:
Pembagian Data

```
from sklearn.model_selection import train_test_split

# Bagi data menjadi data pelatihan, validasi, dan pengujian
x_train, x_test, y_train, y_test = train_test_split(flat_data,
target, test_size=0.2, random_state=42)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
test_size=0.1, random_state=42)
```

3. *Source Code* 3: Pembangunan Model

Source Code 3:
Pembangunan Model

```
from tensorflow.keras.applications import MobileNetV2
import tensorflow as tf

# Load MobileNetV2 model pre-trained pada ImageNet
base_model = MobileNetV2(input_shape=(224, 224, 3), include_top=False,
weights='imagenet')

# Adding custom classification head
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(len(Categories), activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

4. *Source Code* 4: Proses Pelatihan Model

Source Code 4:
Proses Pelatihan Model

```
# Definisikan augmentasi data
datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.3,
    height_shift_range=0.3,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    fill_mode='nearest'
)


# Proses augmentasi data pada dataset pelatihan
datagen.fit(x_train)

# Latih model dengan augmented data dan callback
history = model.fit(datagen.flow(x_train, y_train, batch_size=32),
                    epochs=30,
                    validation_data=(x_val, y_val)
```

5. *Source Code* 5: Pengujian Model

```
# Prediksi untuk satu contoh gambar
img_path = '/content/drive/MyDrive/Skripsi/Test/blasttest.jpeg'
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
predictions = model.predict(img_array)
predicted_class = np.argmax(predictions)
predicted_label = Categories[predicted_class]

# Display the image
plt.imshow(img)
plt.title("Predicted: " + predicted_label)
plt.axis('off')  # Turn off axis
plt.show()

print("Tanaman padi ini terkena penyakit", predicted_label)


# Mendapatkan nilai akurasi pelatihan dan validasi terakhir
train_accuracy = history.history['accuracy'][-1]
val_accuracy = history.history['val_accuracy'][-1]
print('Train Accuracy:', train_accuracy)
print('Validation Accuracy:', val_accuracy)

# Plot akurasi pelatihan dan validasi
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()

# Menampilkan laporan klasifikasi dan matriks kebingungan
y_pred = np.argmax(model.predict(x_test), axis=-1)
print(classification_report(y_test, y_pred, target_names=Categories,
zero_division=1))
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(conf_matrix)
```