IMPLEMENTASI *IMAGE PROCESSING* PADA PENYAKIT DAUN TANAMAN PADI DENGAN METODE CNN (STUDI KASUS KABUPATEN TUBAN)

SKRIPSI

Oleh

Maulana Fiqri Nurul Fawaid

NIM E41200105

PROGRAM STUDI D4 TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI JEMBER

2024

IMPLEMENTASI *IMAGE PROCESSING* PADA PENYAKIT DAUN TANAMAN PADI DENGAN METODE CNN (STUDI KASUS KABUPATEN TUBAN)

SKRIPSI

sebagai salah satu syarat untuk memperoleh gelar Sarjana Sains Terapan Komputer (S.Tr.Kom)
di Program Studi Teknik Informatika
Jurusan Teknologi Informasi

Oleh

Maulana Fiqri Nurul Fawaid

NIM E41200105

PROGRAM STUDI D4 TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI JEMBER

2024

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI JEMBER
JURUSAN TEKNOLOGI INFORMASI

IMPLEMENTASI *IMAGE PROCESSING* PADA PENYAKIT DAUN TANAMAN PADI DENGAN METODE CNN (STUDI KASUS KABUPATEN TUBAN)

Maulana Fiqri Nurul Fawaid (E41200105)

Telah Diuji pada Tanggal 20 Maret 2024 dan Telah Dinyatakan Memenuhi Syarat

Ketua Penguji

Syamsul Arifin, S.Kom, M.Cs
NIP. 19810615 200604 1 002

Sekretaris Penguji

Trismayanti Dwi P, S.Kom, M.Cs
NIP. 19900227 201803 2 001

Anggota Penguji

Zilvanhisna Emka Fitri, ST. MT
NIP. 19920302 201803 2 001

Dosen Pembimbing

Trismayanti Dwi P, S.Kom, M.Cs
NIP. 19900227 201803 2 001

Mengesahkan
Ketua Jurusan Teknologi Informasi

Hendra Yufit Riskiawan, S.Kom, M.Cs
NIP. 19830203 200604 1 003

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Maulana Fiqri Nurul Fawaid

NIM : E41200105

Dengan tulus, saya menyatakan bahwa seluruh pernyataan dalam Laporan Akhir/Skripsi/Tesis berjudul "Implementasi *Image Processing* Pada Penyakit Daun Tanaman Padi Dengan Metode CNN (Studi Kasus Kabupaten Tuban)" adalah hasil dari pemikiran dan karya saya sendiri, dengan bimbingan dari komisi pembimbing, dan belum pernah diajukan dalam bentuk apapun di perguruan tinggi mana pun.

Segala data dan informasi yang digunakan telah diungkapkan secara jelas dan dapat diperiksa keakuratannya. Sumber informasi yang berasal dari atau dirujuk dari karya yang diterbitkan oleh penulis lain telah disebutkan dalam teks dan tercantum dalam daftar pustaka pada bagian akhir Skripsi ini.

Jember, 20 Maret 2024

Maulana Fiqri Nurul Fawaid
NIM E41200105

PERNYATAAN

PERSETUJUAN PUBLIKASI

KARYA ILMIAH UNTUK KEPENTINGAN

AKADEMIS

Yang bertanda tangan dibawah ini, saya:

Nama	**: Maulana Fiqri Nurul Fawaid**
NIM	**: E41200105**
Program Studi	**: Teknik Informatika**
Jurusan	**: Teknologi Informasi**

Demi pengembangan Ilmu Pengetahuan, saya menyetujui untuk memberikan kepada UPT. Perpustakaan Politeknik Negeri Jember, Hak Bebas Royalti Non-Eksklusif *(Non-Exclusive Royalty Free Right)* atas Karya Ilmiah **berupa Laporan Skripsi yang berjudul:**

IMPLEMENTASI *IMAGE PROCESSING* PADA PENYAKIT DAUN TANAMAN PADI DENGAN METODE CNN (STUDI KASUS KABUPATEN TUBAN)

Dengan Hak Bebas Royalti Non-Eksklusif ini UPT. Perpustakaan Politeknik Negeri Jember berhak menyimpan, mengalihkan media atau format, mengelola dalam bentuk Pangakan Data *(Database),* mendistribusikan karya dan menampilkan atau mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa pelu meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis atau pencipta.

Saya bersedia untuk menanggung secara pribadi tanpa melibatkan pihak Politeknik Negeri Jember, segala bentuk tuntutan hukum yang timbul atas Pelarangan Hak Cipta dalam Karya Ilmiah ini.

Demikian Pernyataan ini saya buat dengan sebenar benarnya.

Dibuat di : Jember

Pada Tanggal : 20 Maret 20204

Yang Menyatakan

Nama : Maulana Fiqri Nurul F.

Nim : E41200105

MOTTO

“Jangan khawatir tentang kegagalan, khawatirlah tentang peluang yang mungkin kamu lewatkan ketika kamu bahkan tidak mencoba”
(*Aditya Halindra Faridzky*)

“Satu-satunya batasan untuk meraih mimpi adalah keragu-raguan kita akan hari ini”
(*Aditya Halindra Faridzky*)

PERSEMBAHAN

Dengan mengucapkan Alhamdulillahirobil'alamin serta menyampaikan rasa syukur kepada Allah SWT atas kesehatan, rahmat, dan hidayah-Nya yang telah diberikan, penulis berhasil menyelesaikan skripsi ini. Skripsi ini dengan tulus saya persembahkan kepada:

1. Bapak dan Ibu saya yang telah senantiasa mendoakan saya dengan penuh kesungguhan, serta terima kasih atas segala kasih sayang, dukungan moral, dan motivasi yang telah diberikan.
2. Trismayanti Dwi P, S.Kom, M.Cs., selaku Dosen Pembimbing, terima kasih atas dukungan, bimbingan, motivasi, dan kesempatan pengalaman yang telah diberikan kepada saya selama menjalani masa studi di Politeknik Negeri Jember.
3. Para staf pengajar di Politeknik Negeri Jember, terutama Program Studi Teknik Informatika, yang telah membagikan berbagai ilmu, pengetahuan, dan nasehat yang sangat berharga bagi penulis.
4. Para staf Dinas Ketahanan Pangan, Pertanian dan Perikanan Kabupaten Tuban, terima kasih telah membantu dan mempermudah saya dalam melakukan penelitian.
5. Achmad Fadlori, SP, terima kasih telah berkenan menjadi pakar pada skripsi yang telah saya buat,
6. Diriku sendiri yang telah tegar dan gigih hingga bertahan hingga saat ini

Implementation of Image Processing on Disease
Rice Plant Leaf Disease with CNN Method
(Case Study of Tuban Regency)

Trismayanti Dwi P, S.Kom, M.Cs. as a chief counselor

Maulana Fiqri Nurul Faawaid
Study Program of Informatic engineering
Majoring of Information Technology

ABSTRACT

Rice plants have a very important role in Indonesia's agricultural sector as one of the main commodities in meeting food needs. However, rice production is often hampered by crop diseases, which can cause significant losses to farmers. The use of digital image technology, especially the Convolutional Neural Network (CNN) method, has opened up new opportunities to detect leaf diseases in rice plants more quickly and accurately. This research is focused on Tuban Regency, which is one of the largest rice producing regions in East Java. In this study, the steps taken included collecting image data of healthy rice leaves and contracting diseases, pre-processing image data, training CNN models to classify disease types in rice leaves, and evaluating the accuracy of CNN models. Research results show that CNN models with Xception architecture achieve an accuracy of 0.9853 or 98.53% which is capable of identifying certain types of images such as Xanthomonas oryzaepv. These include leafblight disease in leaves), Pyricularia oryzae Cav (blast disease in leaves), Brown Spot (Chocolate Scatter), Healthy (Healthy Leaves) and Random Data. The implementation of image processing technology using this method is expected to help officers and farmers quickly and accurately identify diseases in rice plants, thus reducing the impact of disease losses.

Keyword: *Image Processing, CNN, Rice Leaf Disease*

Implementasi *Image Processing* Pada Penyakit
Daun Tanaman Padi Dengan Metode CNN
(Studi Kasus Kabupaten Tuban)

Trismayanti Dwi P, S.Kom, M.Cs. selaku Dosen Pembimbing

Maulana Fiqri Nurul Fawaid
Program Studi Teknik Informatika
Jurusan Teknologi Informasi

ABSTRAK

Tanaman padi memiliki peran yang sangat penting dalam sektor pertanian Indonesia sebagai salah satu komoditas utama dalam pemenuhan kebutuhan pangan. Namun, produksi padi sering kali mengalami hambatan akibat penyakit tanaman, yang dapat menyebabkan kerugian besar bagi petani. Penggunaan teknologi citra digital, khususnya metode *Convolutional Neural Network* (CNN), telah membuka peluang baru dalam mendeteksi penyakit daun pada tanaman padi dengan lebih cepat dan akurat. Penelitian ini difokuskan pada Kabupaten Tuban, yang merupakan salah satu daerah penghasil padi terbesar di Jawa Timur. Dalam penelitian ini, langkah-langkah yang dilakukan meliputi pengumpulan data citra daun padi yang sehat dan terserang penyakit, melakukan *pre-processing* data citra, melatih model CNN untuk mengklasifikasikan jenis penyakit pada daun padi, serta mengevaluasi akurasi model CNN. Hasil penelitian menunjukkan bahwa model CNN dengan arsitektur *Xception* mencapai akurasi sebesar 0.9853 atau 98.53% yang mampu mengidentifikasi beberapa jenis citra seperti *Xanthomonas oryzae pv. oryzicola* (penyakit leaf blight pada daun), *Pyricularia oryzae Cav* (penyakit blast pada daun), *Brown Spot* (Bercak Coklat),*Healthy* (Daun Sehat) dan Data acak. Implementasi teknologi *image processing* dengan metode ini diharapkan dapat membantu petugas dan petani dalam mengidentifikasi penyakit pada tanaman padi secara cepat dan akurat, sehingga dapat mengurangi dampak kerugian akibat serangan penyakit.

Kata Kunci : *Image Processing*, CNN, Penyakit Daun Padi

RINGKASAN

IMPLEMENTASI *IMAGE PROCESSING* PADA PENYAKIT DAUN TANAMAN PADI DENGAN METODE CNN (STUDI KASUS KABUPATEN TUBAN), Maulana Fiqri Nurul Fawaid, NIM E41200105, Tahun 2024, 142 hlm.,Teknologi Informasi, Politeknik Negeri Jember, Trismayanti Dwi P, S.Kom, M.Cs. (Dosen Pembimbing).

Tanaman padi merupakan komoditas penting dalam sektor pertanian Indonesia. Namun, produksi padi sering terganggu oleh penyakit tanaman, yang dapat menyebabkan kerugian besar bagi petani. Penggunaan teknologi citra digital, khususnya metode *Convolutional Neural Network* (CNN), dapat mempercepat dan meningkatkan akurasi dalam mendeteksi penyakit daun pada tanaman padi. Kabupaten Tuban, sebagai daerah penghasil padi terbesar di Jawa Timur, menjadi fokus penelitian karena kebutuhan akan solusi teknologi untuk mengatasi penyakit daun padi.

Penelitian ini bertujuan untuk menerapkan teknik *image processing* pada penyakit daun tanaman padi dengan metode CNN pada studi kasus di Kabupaten Tuban. Untuk mencapai tujuan tersebut, peneliti melakukan beberapa langkah, yaitu mengumpulkan data citra daun padi yang sehat dan terserang penyakit,Melakukan *pre-processing* data citra,melatih model CNN untuk mengklasifikasikan jenis penyakit pada daun padi,Mengevaluasi akurasi model CNN serta mengimplementasikan model kedalam *website*.

Hasil penelitian menunjukkan bahwa akurasi model CNN dengan arsitektur *Xception* mencapai 98.53%. Model ini dapat mengidentifikasi beberapa citra seperti *Xanthomonas oryzae pv. oryzicola* (penyakit leaf blight pada daun), *Pyricularia oryzae Cav* (penyakit blast pada daun), *Brown Spot* (Bercak Coklat),*Healthy* (Daun Sehat) dan Data acak. Model ini dapat membantu petani dalam mengidentifikasi jenis penyakit pada daun tanaman padi dengan cepat dan akurat. Penerapan teknologi *image processing* pada penyakit daun tanaman padi dengan metode CNN

terbukti efektif,akurat serta dapat membantu petani dalam mengidentifikasi jenis penyakit.

PRAKATA

Puji syukur kami sampaikan atas kehadiran Allah SWT dan Nabi Besar Muhammad SAW, yang telah melimpahkan rahmat dan petunjuk-Nya, memungkinkan penulis menyelesaikan skripsi dengan judul " Implementasi *Image Processing* Pada Penyakit Daun Tanaman Padi Dengan Metode CNN (Studi Kasus Kabupaten Tuban)" ini dengan sukses dan dalam keadaan sehat wal-afiat.

Proses penulisan laporan ini tidak terlepas dari bantuan berbagai pihak, oleh karena itu, penulis ingin mengungkapkan rasa terima kasih kepada:

1. Orang tua penulis yang memberikan dukungan dan semangat sehingga penulis dapat menyelesaikan skripsi ini,
2. Saiful Anwar, S.TP, M.P. selaku Direktur Politeknik Negeri Jember,
3. Hendra Yufit Riskiawan, S.Kom., M.Cs. selaku Ketua Jurusan Teknologi Informasi,
4. Bety Etikasari, S.Pd, M.Pd. selaku Ketua Program Studi Teknik Informatika,
5. Trismayanti Dwi P, S.Kom, M.Cs. selaku Dosen Pembimbing,
6. Achmad Fadlori, SP, selaku Pakar serta Koordinator Wilayah Kerja TPH Proteksi Bojonegoro,
7. Serta kepada rekan-rekan dan semua pihak yang turut berkontribusi dalam memberikan semangat untuk penulis menyelesaikan skripsi ini.

Penulis sadar bahwa laporan magang ini masih memiliki banyak kekurangan. Oleh karean itu penulis dengan tulus mengharapkan kritik dan saran bermanfaat yang bersifat membangun guna perbaikan dimasa yang akan datang.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

DAFTAR KODE

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Pertanian merupakan salah satu sektor penting pada perekonomian Indonesia. Sektor pertanian khususnya tanaman padi menjadi salah satu tumbuhan yang paling krusial. Namun, produksi padi sering kali terganggu oleh berbagai penyakit tanaman yang dapat menyebabkan kerugian yang signifikan bagi petani. Penyakit daun di tanaman padi bisa ditimbulkan oleh berbagai faktor, seperti cuaca, kualitas tanah, kelembaban, dan serangan hama.

Salah satu permasalahan utama dalam produksi padi adalah hama dan penyakit tanaman padi. Penyakit pada tanaman padi dapat disebabkan oleh berbagai faktor, seperti jamur, bakteri, virus, dan faktor lingkungan. Contoh penyakit daun yang sering terjadi di Indonesia adalah penyakit blast (*Pyricularia oryzae*), hawar daun bakteri (*Xanthomonas oryzae pv. oryzae*), dan tungro (Rice tungro spherical virus), *Brown Spot*(bercak coklat). Hama dan penyakit dapat menyebabkan kerusakan pada daun, batang, dan malai padi, sehingga dapat menurunkan hasil panen (Pradana, 2022).

Kabupaten Tuban merupakan salah satu wilayah di Jawa Timur yang memiliki produksi padi yang cukup tinggi. Berdasarkan data yang diambil dari BPS (Badan Pusat Statistik) kabupaten Tuban menunjukkan rata-rata produksi padi di Tuban pada tahun 2022 mencapai 661.292 ton, namun turun menjadi 640.547 ton pada tahun 2023. Penurunan sebesar 20.745 ton ini disinyalir akibat faktor iklim dan hama penyakit tanaman padi (BPS, 2024). Saat ini, proses pendeteksian penyakit tanaman padi masih dilakukan secara manual oleh petani dan petugas setempat. Hal ini memiliki beberapa kelemahan, seperti: Akurasi rendah: Petani mungkin tidak memiliki pengetahuan yang cukup untuk mengidentifikasi penyakit secara akurat. Proses yang lama dan melelahkan: Memeriksa tanaman padi secara manual membutuhkan waktu dan tenaga yang banyak. Ketidakmampuan untuk mendeteksi penyakit pada tahap awal: Pada tahap awal, gejala penyakit tanaman padi sering kali tidak terlihat jelas, sehingga sulit dideteksi secara manual. Oleh

sebab itu, penelitian mengenai implementasi *image processing* pada penyakit daun tanaman padi menggunakan metode CNN di Kabupaten Tuban perlu dilakukan guna memberikan solusi dalam mengatasi persoalan tersebut.

Pada penelitian sebelumnya tentang CNN pada penyakit daun padi oleh Hawari dkk. (2022) menunjukkan hasil menjanjikan, namun masih terdapat beberapa kekurangan. Pertama, penelitian sebelumnya hanya fokus pada dua jenis penyakit, sehingga cakupannya terbatas. Kedua, penelitian tersebut belum berfokus pada implementasi praktis di lapangan. Ketiga, model CNN yang dikembangkan di satu lokasi belum tentu optimal di daerah lain dengan karakteristik berbeda. Penelitian ini perlu diperkuat dengan mengembangkan model CNN yang mampu mengidentifikasi berbagai jenis penyakit, membuat aplikasi mudah digunakan untuk identifikasi dan penanganan penyakit, serta menguji dan mengadaptasi model CNN di berbagai lokasi dengan karakteristik berbeda. Kerjasama dengan dinas pertanian dan pemangku kepentingan terkait juga diperlukan untuk implementasi model CNN di lapangan. Oleh karena itu, penelitian ini akan fokus pada 5 jenis dataset citra seperti *xanthomonas oryzae pv. oryzicola* (penyakit *leaf blight* pada daun), *pyricularia oryzae Cav* (penyakit blast pada daun), *brown Spot* (Bercak Coklat), *healthy* (daun sehat) dan data acak. Kemudian penelitian ini akan dibangun sebuah website guna untuk memudahkan pengguna dalam melakukan identifikasi penyakit.

Oleh karena itu, penelitian ini dilakukan dengan tujuan untuk mengembangkan sistem identifikasi penyakit daun padi pada tanaman padi di Kabupaten Tuban menggunakan metode *Convolutional Neural Network* (CNN). Hipotesis penelitian ini adalah bahwa sistem identifikasi penyakit daun padi yang dikembangkan dapat membantu petani di Kabupaten Tuban untuk meningkatkan hasil panen padi. Beberapa penelitian sebelumnya telah menunjukkan bahwa CNN dapat digunakan untuk mengidentifikasi penyakit tanaman dengan tingkat akurasi yang tinggi. Penelitian ini diharapkan dapat memberikan kontribusi pada pengembangan teknologi pertanian di Indonesia, khususnya di Kabupaten Tuban, dengan menyediakan sistem identifikasi penyakit daun padi yang akurat dan efisien.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam usulan ini antara lain:

a. Bagaimana penerapan teknologi *image processing* dengan metode CNN dapat membantu petani dalam mengidentifikasi jenis penyakit daun tanaman padi?
b. Apa saja faktor-faktor yang mempengaruhi akurasi metode CNN dalam mengidentifikasi jenis penyakit pada daun tanaman padi?
c. Bagaimana tingkat akurasi hasil dari sistem klasifikasi penyakit daun tanaman padi menggunakan algoritma *Convolutional Neural Network* (CNN)?

1.3 Tujuan

Berdasarkan latar belakang di atas, penelitian ini memiliki tujuan sebagai berikut:

a. Mengembangkan sistem klasifikasi penyakit daun padi menggunakan metode CNN.
b. Menganalisis faktor-faktor yang mempengaruhi akurasi deteksi penyakit daun padi dengan CNN.
c. Mengukur tingkat akurasi metode CNN dalam mendeteksi penyakit daun padi di Kabupaten Tuban.

1.4 Manfaat

Berdasarkan latar belakang di atas, penelitian ini memiliki manfaat sebagai berikut:

1.4.1 Manfaat bagi Peneliti:

1. Peneliti bisa memperdalam pengetahuan dan keterampilan di bidang *image processing* serta pengembangan model CNN.
2. Hasil penelitian ini dapat menjadi referensi serta menaikkan kredibilitas peneliti di dunia akademik.

3. Dapat menaikkan kemampuan peneliti dalam melakukan penelitian serta analisis data.

1.4.2 Manfaat bagi Pembaca:

1. Pembaca dapat memperoleh pengetahuan tentang pengembangan teknologi *image processing* dan pengembangan model CNN untuk identifikasi penyakit pada tanaman padi.
2. Hasil penelitian ini dapat membantu pembaca memahami lebih dalam tentang teknologi kecerdasan buatan dan penerapannya di bidang pertanian.
3. Pembaca dapat mengaplikasikan hasil penelitian ini pada bidang keahlian mereka sendiri.

1.4.3 Manfaat bagi Instansi terkait:

1. Penelitian ini dapat membantu Dinas Ketahanan Pangan, Pertanian dan Perikanan Kabupaten Tuban untuk meningkatkan kualitas dan efisiensi dalam pemantauan dan pengendalian penyakit pada tanaman padi.
2. Hasil penelitian ini dapat menjadi acuan dalam pengembangan teknologi dan kebijakan di bidang pertanian.
3. Dapat mempercepat pengambilan tindakan pengendalian penyakit pada tanaman padi dan meningkatkan produktivitas petani di Kabupaten Tuban.

1.5 Batasan Masalah

Untuk memastikan arah yang jelas dalam penelitian ini dan menghindari perluasan masalah yang dihadapi oleh Peneliti, telah ditetapkan batasan masalah. Beberapa batasan masalah yang berlaku dalam penelitian ini meliputi:

a. Fokus pada identifikasi penyakit daun pada tanaman padi di Kabupaten Tuban menggunakan teknologi *image processing* dan pengembangan model CNN.

b. Data yang digunakan dalam penelitian ini merupakan data yang diperoleh dari wilayah Kabupaten Tuban.

c. Penelitian ini difokuskan pada pengembangan model CNN untuk mengidentifikasi 3 jenis penyakit pada daun padi di Kabupaten Tuban. Penyakit yang akan diidentifikasi adalah *Xanthomonas oryzae pv. oryzicola* (penyakit leaf blight pada daun), *Pyricularia oryzae Cav* (penyakit blast pada daun), *Brown Spot* (Bercak Coklat).

d. Serta tanaman daun padi sehat akan dijadikan sebagai referensi dalam pengidentifikasian penyakit pada daun padi.

BAB 2. TINJAUAN PUSTAKA

2.1 *State of The Art*

Dalam penelitian ini, penting bagi peneliti untuk mencari referensi yang relevan dengan tema *image processing*. Selain itu, menemukan referensi yang tepat dapat memberikan para peneliti bahan belajar yang berguna untuk menjaga tema tetap dikembangkan. Serta, informasi tentang beberapa penelitian yang relevan ditunjukkan pada Tabel 2.1.

Tabel 2.1 *State Of The Art*

Peneliti	Judul	Metode	*Dataset*	Hasil
Alang Mulya Lesmana, Ronna Putri Fadhillah, Chaerur Rozikin	Identifikasi Penyakit pada Citra Daun Kentang Menggunakan *Convolutional Neural Network* (CNN)	*Convolutional Neural Network* (CNN)	5400 data citra akan dibagi menjadi 3 kelas yaitu 2 penyakit dan 1 sehat	Hasilnya menunjukkan bahwa dalam kasus ini tingkat akurasi 93%.
Rosalina, Ardi Wijaya	Pendeteksian Penyakit pada Daun Cabai dengan Menggunakan Metode *Deep Learning*	*Convolutional Neural Network* (CNN)	1000 data citra akan dibagi menjadi 2 kelas yaitu 500 data daun cabai sakit dan 500 data daun cabai sehat	Hasilnya menunjukkan bahwa dalam kasus ini tingkat akurasi 68,8%.

Peneliti	Judul	Metode	*Dataset*	Hasil
Qudsiah Nur Azizah	Klasifikasi Penyakit Daun Jagung Menggunakan Metode *Convolutional Neural Network* AlexNet	*Convolutional Neural Network* (CNN) Model AlexNet	4198 data citra akan dibagi menjadi 4 kelas yaitu 3 penyakit dan 1 sehat	Hasilnya menunjukkan bahwa dalam kasus ini tingkat akurasi 90%.
Rizal Amegia Saputra, Sri Wasiyanti, Adi Supriyatna, Dede Firmansyah Saefudin	Penerapan Algoritma *Convolutional Neural Network* Dan Arsitektur MobileNet Pada Aplikasi Deteksi Penyakit Daun Padi	*Convolutional Neural Network* (CNN) Model MobileNetVI	120 gambar daun padi yang terdiri dari 3 penyakit yaitu *blight, brown spot* dan *smut*	Hasilnya menunjukkan bahwa model Mobilenet menghasilkan hasil yang optimal dan akurat. Klasifikasi 83%
Rian Kurniawan, Yessi Mulyani, Puput Budi Wintoro, Muhamad Komarudin	IMPLEMENTASI ARSITEKTUR *XCEPTION* PADA MODEL *MACHINE LEARNING* KLASIFIKASI SAMPAH ANORGANIK	*Convolutional Neural Network* (CNN) Model *Xception*	1.850 data latih, 402 data validasi, dan 395 data uji.	Hasilnya menunjukkan bahwa tingkat akurasi sebesar 87,81%

Beberapa penelitian telah dilakukan untuk klasifikasi menggunakan metode *Convolutional Neural Network* (CNN). Penelitian ini menunjukkan hasil yang

beragam dalam hal akurasi. Penelitian Alang Mulya Lesmana dkk. menunjukkan hasil terbaik dengan tingkat akurasi 93% dalam klasifikasi penyakit daun kentang. Penelitian ini menggunakan *dataset* yang terdiri dari 5400 data citra daun kentang yang dikategorikan menjadi 2 kelas penyakit dan 1 kelas sehat.

Penelitian selanjutnya yaitu penelitian Rosalina dan Ardi Wijaya memiliki tingkat akurasi terendah yaitu 68,8% dalam klasifikasi penyakit daun cabai. *Dataset* yang digunakan terdiri dari 1000 data citra daun cabai, 500 data daun cabai sakit dan 500 data daun cabai sehat.

Adapun penelitian yang serupa dengan penelitian sebelumnya yaitu penelitian Qudsiah Nur Azizah menggunakan model AlexNet dan menghasilkan akurasi 90% dalam klasifikasi penyakit daun jagung. *Dataset* yang digunakan terdiri dari 4198 data citra daun jagung yang dikategorikan menjadi 3 kelas penyakit dan 1 kelas sehat. Penelitian Rian Kurniawan dkk. menunjukkan akurasi 87,81% dalam klasifikasi sampah anorganik menggunakan model *Xception*.

Penelitian yang serupa dengan penelitian yang akan saya lakukan yaitu penelitian Saputra dkk menggunakan model Mobilenet dan menghasilkan akurasi 83% dalam klasifikasi penyakit daun padi. *Dataset* yang digunakan terdiri dari 120 gambar daun padi yang terdiri dari 3 penyakit yaitu blight, brown spot dan smut.

Kemudian adapun penelitian yang saya gunakan juga yaitu penelitian Rian Kurniawan dkk. menunjukkan akurasi 87,81% dalam klasifikasi sampah anorganik menggunakan model *Xception*.

Akurasi model CNN dapat bervariasi tergantung pada beberapa faktor, seperti jenis dataset, metode pelatihan, dan arsitektur model. Penelitian ini hanya menunjukkan contoh kecil dari berbagai penelitian klasifikasi penyakit tanaman menggunakan CNN. Masih banyak penelitian lain yang perlu dilakukan untuk meningkatkan akurasi dan efisiensi model CNN dalam klasifikasi penyakit tanaman.

2.2 Tanaman Padi

Padi (*Oryza sativa*) merupakan tanaman pangan yang sangat penting bagi keberlangsungan hidup manusia. Tanaman ini merupakan makanan pokok bagi

hampir separuh penduduk dunia, terutama di Asia. Padi tumbuh pada lahan yang cukup basah serta memerlukan pengairan yang baik. Padi juga memerlukan perawatan yang tepat, seperti pemupukan, pengendalian hama serta penyakit, dan teknik budidaya yang baik, mirip sistem tanam jajar legowo, untuk mencapai hasil panen yang optimal (Husain, 2019).

Meskipun padi merupakan makanan pokok di banyak negara, produksinya tak jarang dipengaruhi oleh berbagai faktor seperti perubahan iklim dan perubahan penggunaan lahan. Untuk itu, dibutuhkan upaya-upaya yang serius buat meningkatkan produktivitas padi serta menjaga keberlangsungan produksinya. Salah satu upaya tadi ialah menggunakan penggunaan varietas unggul padi yang tahan terhadap kondisi lingkungan yang berubah-ubah. Beberapa varietas padi yang tahan terhadap kekeringan, agresi hama serta penyakit sudah dikembangkan dan mulai dipergunakan oleh petani di berbagai negara (Nurwijayo, 2022).Selain itu, perlu pula ditingkatkan pengelolaan lahan pertanian yang berkelanjutan, seperti memakai teknik pengolahan tanah yang baik, melakukan rotasi tanaman, serta penggunaan pupuk organik buat menjaga kesehatan tanah dan mengurangi penggunaan pestisida dan herbisida. Pengelolaan lahan pertanian yang berkelanjutan bisa membantu meningkatkan produktivitas padi secara jangka panjang dan menjaga keseimbangan ekosistem pada sekitar lahan pertanian (Parisa, 2021).

Secara keseluruhan, padi adalah tanaman pangan penting yang memerlukan perawatan dan manajemen yang baik untuk menaikkan produktivitas dan menjaga keberlangsungan produksinya. Upaya-upaya buat menaikkan produktivitas padi melalui penggunaan varietas unggul dan pengelolaan lahan pertanian yang berkelanjutan sangat penting untuk memastikan ketersediaan pangan di masa depan.

2.3 Penyakit Tanaman Padi

Penyakit tanaman padi adalah salah satu masalah utama dalam produksi padi dan dapat menyebabkan kerugian ekonomi yang signifikan bagi petani. Beberapa jenis penyakit yang umum ditemukan pada tanaman padi antara lain *blast, leaf*

blight, hawar daun bakteri, dan karat daun (Safrullah, 2019). Penyebaran penyakit ini dapat dipengaruhi oleh faktor lingkungan seperti kelembaban dan suhu yang tinggi, serta faktor varietas padi yang rentan terhadap serangan penyakit. Oleh karena itu, langkah-langkah pencegahan dan pengendalian penyakit tanaman padi menjadi sangat penting untuk menjaga produktivitas dan ketersediaan pangan di masa depan (Nuryanto, 2018).

Untuk mengendalikan penyakit tanaman padi, ada beberapa strategi yang dapat digunakan, seperti pemilihan varietas padi yang tahan terhadap penyakit, sanitasi lahan yang baik, dan penggunaan pestisida yang tepat (Aeni, 2022). Selain itu, teknik pengelolaan tanah yang baik juga dapat membantu mengurangi serangan penyakit pada tanaman padi. Meskipun penggunaan pestisida dapat memberikan solusi cepat untuk mengendalikan penyakit, namun penggunaannya harus dilakukan dengan hati-hati dan sesuai dengan petunjuk penggunaan agar tidak membahayakan kesehatan manusia dan lingkungan (Kementerian Pertanian, Badan Penyuluhan dan Pengembangan Sumber Daya Manusia Pertanian, 2019).

2.3.1 *Xanthomonas Oryzae Pv. Oryzicola* (Penyakit *Leaf Blight*)

Xanthomonas oryzae pv. Oryzicola atau penyakit *leaf blight yang* menyerang daun tanaman padi dan merupakan salah satu penyebab utama kerugian ekonomi pada produksi padi di seluruh dunia. Penyakit ini sering disebut juga dengan nama "hawar daun" karena gejalanya yang menyerupai hawar pada daun tanaman padi. Penyakit ini ditandai dengan munculnya bercak berwarna kuning atau coklat pada daun, yang kemudian meluas dan menghitam. Bercak tersebut akhirnya membentuk lesi pada daun yang mengering dan rusak. Penyakit *leaf blight* dapat menyerang tanaman padi pada semua fase pertumbuhannya, mulai dari fase bibit hingga fase pematangan gabah. Kerusakan yang diakibatkan oleh penyakit ini dapat berdampak buruk pada produksi dan kualitas gabah, serta mengurangi hasil panen secara signifikan (D. Janse, 2022).

Gambar 2.1 Penyakit *Leaf Blight*

Sumber: https://www.cabidigitallibrary.org/doi/10.1079/cabicompendium.56977 (2021)

2.3.2 *Pyricularia Oryzae Cav* (Penyakit *Blast*)

Pyricularia oryzae Cav atau yang dikenal sebagai penyakit *blas*, merupakan salah satu penyakit tanaman padi yang paling merusak dan menyebar di seluruh dunia. Penyakit ini seringkali disebut juga sebagai "kudis padi" karena munculnya lesi pada bagian daun, batang, malai, dan gabah tanaman padi. Gejala awal dari penyakit ini adalah adanya bercak putih keabuan atau kelabu pada daun, kemudian berkembang menjadi bercak berbentuk bulat dengan pinggiran yang gelap dan tengah yang terang. Akhirnya, bercak tersebut melebar dan mengering membentuk lubang pada daun dan batang, yang mempengaruhi kemampuan tanaman padi dalam melakukan fotosintesis dan pertumbuhannya secara keseluruhan. *Pyricularia oryzae Cav* dapat menyerang tanaman padi pada semua fase pertumbuhan, mulai dari benih hingga panen. Karena tingkat kerusakannya yang tinggi, penyakit *blast*

menjadi salah satu ancaman utama bagi produksi padi di seluruh dunia (Nartymov dkk., 2021).

Gambar 2.2 Penyakit *Blast*

Sumber: https://nufarm.com/id/penyakit-blast/ (2022)

2.3.3 *Brown Spot* (bercak coklat)

Brown Spot atau bercak coklat adalah salah satu jenis penyakit pada tanaman padi yang disebabkan oleh jamur *Bipolaris oryzae*. Penyakit ini umumnya muncul pada fase vegetatif dan generatif tanaman padi dan dapat menyebabkan kerugian pada produksi dan kualitas hasil panen. Gejala awal dari penyakit ini adalah munculnya bercak kecil berwarna coklat pada daun, batang, dan malai padi. Bercak-bercak ini akan semakin melebar dan bergabung membentuk bercak-bercak besar yang dapat mengering dan menjadi gugur. Penyebaran *Brown Spot* pada tanaman padi dapat terjadi melalui serangan jamur yang menempel pada bibit atau melalui angin dan air hujan yang membawa spora jamur. Pengendalian penyakit ini dapat dilakukan dengan cara penggunaan benih yang bebas dari jamur penyebab, pemupukan yang tepat, sanitasi lingkungan, dan penggunaan fungisida (Ahmadpour dkk., 2017).

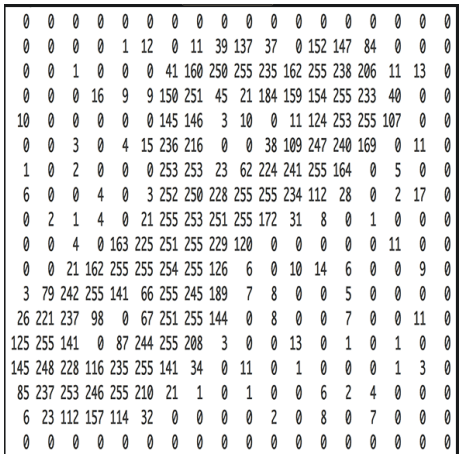
Gambar 2.3 Penyakit *Brown Spot*

Sumber: http://www.knowledgebank.irri.org/training/fact-sheets/pest-management/diseases/item/brown-spot (2021)

2.4 Citra

Gambar adalah gambar bidang dua dimensi yang terdiri dari banyak piksel, di mana piksel berdiri untuk Elemen Gambar dan dapat dianggap sebagai ratusan hingga jutaan titik kecil yang membentuk *snapshot* digital. Setiap *pixel* berisi informasi yang mengontrol warna (*hue*), intensitas warna (saturasi), dan kecerahan warna yang disajikan, sehingga pixel adalah bagian terkecil dari gambar. Secara umum, gambar terdiri dari persegi panjang dengan ruang *horizontal* dan vertikal yang identik antara piksel di seluruh gambar. Gambar digital adalah gambar yang dibuat sebagai konsekuensi dari pengolahan pada kamera, komputer, pemindai, atau peralatan elektronik lainnya (Hidayatullah, 2017).

Mesin melihat gambar sebagai kumpulan bilangan bulat dalam matriks, yang masing-masing menentukan intensitas cahaya di tempat itu, atau piksel.Serta contoh citra dalam pixel dapat dilihat seperti pada Gambar 2.4.

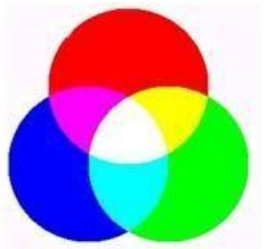


Gambar 2.4 Citra dalam *Pixel*

Sumber: Wilcox, How to use deep neural networks to forecast stock prices (2021)

2.4.1 Citra RGB

Citra RGB merupakan citra yang terdiri berasal 3 warna dasar, yaitu merah (R), hijau (G), serta biru (B), yang dikombinasikan untuk membuat gambar berwarna. Teknologi citra RGB dipergunakan secara luas dalam bidang fotografi, televisi, serta teknologi berita. Pada citra RGB, setiap piksel direpresentasikan oleh tiga nilai yang mewakili intensitas rona merah, hijau, serta biru. Kombinasi asal tiga nilai ini menghasilkan jutaan rona yang tidak sama yang dapat ditampilkan di layar. Serta contoh citra rgb dapat dilihat pada Gambar 2.5.



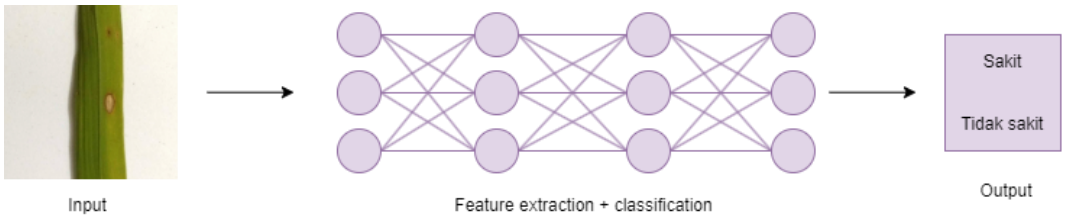
Gambar 2.5 Citra RGB

Dalam pemrosesan gambar digital, gambaran RGB seringkali dipergunakan menjadi representasi warna standar. Citra RGB juga sering dipergunakan pada teknik pengolahan citra seperti pemrosesan serta analisis citra, restorasi citra, serta

pengenalan citra sebagai model. Analisis intensitas warna pada citra RGB bisa dipergunakan untuk mengenali pola yang khas pada gambar, sehingga dapat dipergunakan buat mengenali objek serta karakteristik eksklusif pada gambar.

2.5 *Deep Learning*

Deep learning adalah algoritma pembelajaran mesin yang berusaha untuk belajar pada berbagai tingkatan, yaitu pada tingkat abstraksi yang berbeda. Jaringan saraf tiruan umumnya digunakan dalam *deep learning*. Ide *deep learning* bervariasi dalam bahwa konsep tingkat tinggi ditentukan oleh konsep tingkat rendah, dan gagasan tingkat rendah dapat berfungsi untuk mendefinisikan banyak konsep tingkat tinggi. *Deep learning* mengacu pada cabang pembelajaran mesin yang sekarang menjadi referensi penelitian yang paling umum untuk pengolahan citra digital. Algoritma *deep learning* tidak memerlukan pengetahuan tentang data yang akan dipelajari, dan mereka dapat secara sendiri melakukan pengolahan dan pemilihan model (Retnowardhani dan Ramdani, 2019). Serta contoh pemodelan *deep learning* dapat dilihat pada Gambar 2.6.

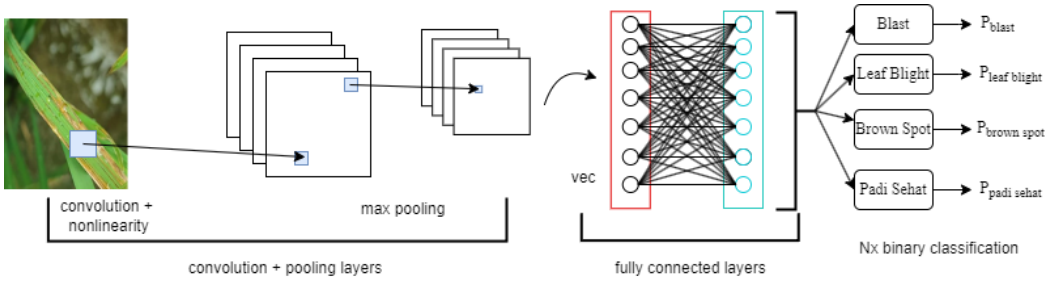


Gambar 2.6 Pemodelan *Deep Learning*

2.6 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan algoritma *deep learning* karena kedalaman jaringannya yang besar, *convolutional neural network* (CNN) adalah metode *deep learning*. Algoritma CNN adalah hasil dari pembuatan *Multilayer Perceptron* (MLP), yang memungkinkan pemrosesan data dua dimensi, memungkinkan metode CNN digunakan dalam pemrosesan data gambar atau audio. *Convolutional neural network* terdiri dari lapisan *input*, lapisan *output*, dan beberapa lapisan tersembunyi. *Convolution layer*, fungsi aktivasi, *rectified linear unit* (ReLU), *pooling layer, dropout layer dan fully connected layer* adalah contoh

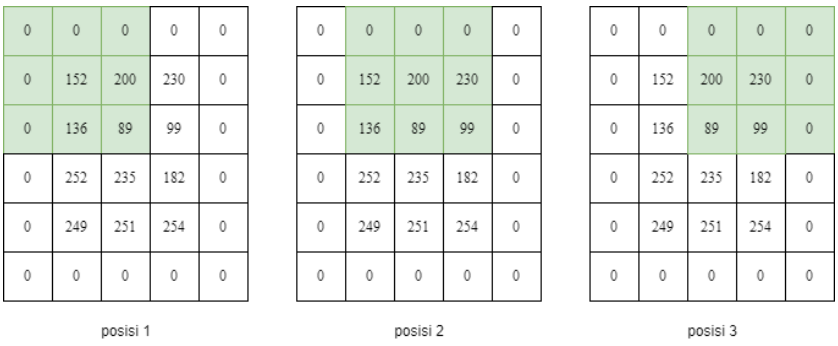
lapisan tersembunyi yang ditemukan di CNN. Lapisan ini digunakan untuk mempelajari karakteristik data gambar dengan membaca dan memproses nilai *pixel*. *Fully connected layer* adalah tingkat terakhir yang memberikan nilai probabilitas untuk setiap kelas (Mishra, 2020). Serta tahapan CNN dapat dilihat pada Gambar 2.7.



Gambar 2.7 Tahapan CNN

2.6.1 *Convolution Layer*

Convolution layer adalah sebuah matriks yang melakukan operasi matematika pada sebuah gambar. *Convolution layer* ini dapat untuk melakukan operasi tertentu pada data citra berturut-turut untuk memeriksa fitur tepi, warna, dan bentuk gambar masukan. Operasi *convolution* dilakukan pada dua matriks: matriks gambar masukan dan matriks kernel atau filter. Dalam *convolutional neural network*, filter adalah matriks nilai acak antara -1 hingga 1 yang digunakan untuk mempelajari karakteristik dalam gambar di wilayah kecil berdasarkan ukuran filter. Ukuran filter ditentukan oleh arsitektur. Filter bergeser dengan jumlah yang sama dengan nilai piksel pada gambar masukan (yanuar, 2018). Serta operasi *convolution* dengan *stride* 1 dapat dilihat pada Gambar 2.8.

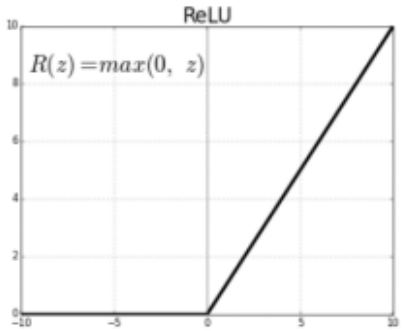


Gambar 2.8 Operasi *Convolution*

Neuron di organisasi ke dalam *feature map* di lapisan *convolution*. Setiap *neuron* dalam *feature map* adalah *receptive field*, yang sebelumnya dilatih dalam *neuron convolution layer* menggunakan satu set bobot yang dikenal sebagai *filter bank*.

2.6.2 *Rectified Linear Unit (ReLU)*

Dalam algoritma CNN, *rectified linear unit* (ReLU) adalah fungsi aktivasi yang paling banyak digunakan. ReLU memodifikasi nilai masukan dari *feature map neuron* yang dihasilkan oleh *convolution layer* dari 0 ke tak terhingga. Fungsi dari fungsi aktivasi ReLU ditentukan dalam persamaan berikut jika x adalah nilai masukan dari *neuron*. Serta aktivasi ReLU dapat dilihat pada Gambar 2.9.



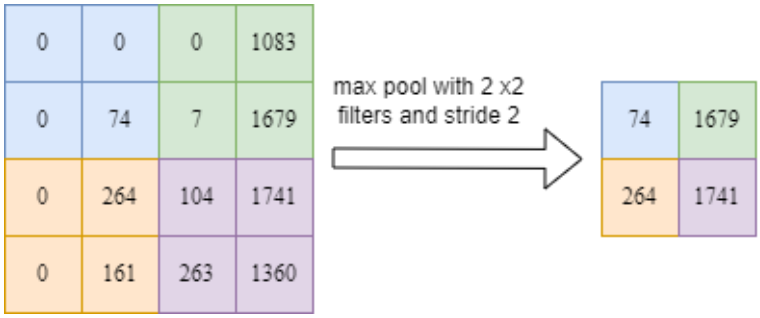
Gambar 2.9 *ReLU*

Dalam feature map, nilai 0 ditetapkan sebagai fungsi aktivasi sebagai pengganti nilai negatif, dan nilai masukan dari neuron feature map tetap jika bernilai lebih tinggi dari atau sama dengan 0 (Nurhikmat, 2018).

2.6.3 *Pooling Layer*

Pooling layer adalah lapisan dalam *convolutional neural network* yang mengambil fungsi *feature map* sebagai masukan dan memprosesnya dengan berbagai operasi statistik tergantung pada nilai piksel di sekitarnya. *Pooling layer*, yang ditambahkan di antara *convolution layer* berturut-turut dalam arsitektur model CNN, mempercepat proses komputasi dengan menurunkan volume setiap tumpukan *feature map* tanpa menghapus informasi penting. *Pooling layer*

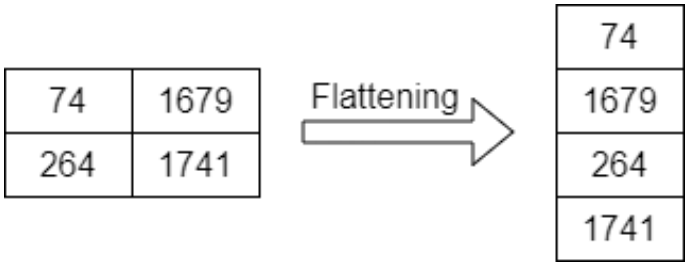
diklasifikasikan menjadi dua jenis: *pooling layer* maksimum dan *pooling layer* rata-rata. *Pooling layer* yang paling umum menggunakan filter 2x2, yang kemudian diterapkan dalam dua fase dan bertindak pada setiap irisan masukan. *Feature map* akan dikurangi menjadi 75% dari ukuran aslinya sebagai hasilnya (Edbert, 2021). Serta contoh operasi *max pooling* dapat dilihat pada Gambar 2.10.



Gambar 2.10 *Max-Pooling*

2.6.4 *Flatten*

Flatten digunakan untuk membentuk kembali *feature map* menjadi vektor yang kemudian dapat dimasukkan ke dalam *fully connected layer*. Serta contoh tahapan *flatten* dapat dilihat pada Gambar 2.11.



Gambar 2.11 *Flatten.*

2.6.5 *Dropout Layer*

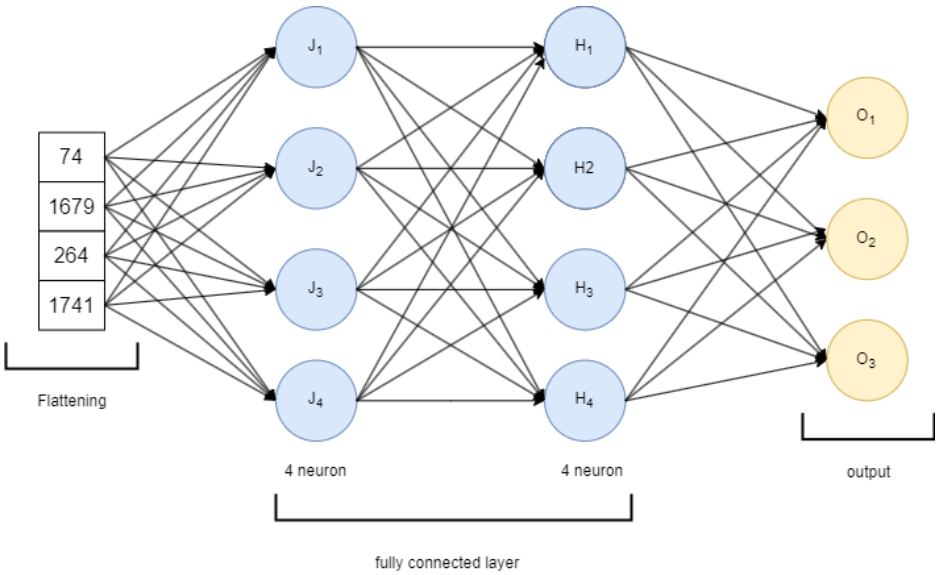
Algoritma CNN menampilkan lapisan tersembunyi yang sangat rumit, yang sering mengakibatkan *overfitting*. *Dropout layer* adalah salah satu metode untuk menyesuaikan diri. *Dropout layer* adalah pendekatan dasar yang dapat mengubah kinerja algoritma CNN selama pengujian model.

Dalam setiap iterasi model proses pelatihan, *dropout layer* secara acak menghilangkan *neuron* tertentu dengan probabilitas sebesar p, dan p dapat diatur

dengan eksperimen. Jaringan akan lebih tipis setelah *dropout layer* ditambahkan, yang menurun sesuai dengan model dan dapat mempercepat proses pelatihan.

2.6.6 *Fully Connected Layer*

Fully connected layer sepenuhnya mengkategorikan output dari *convolution layer* dan *pooling*. Setiap jaringan pada lapisan yang benar-benar terhubung dihubungkan ke jaringan pada tingkat sebelumnya. *Fully connected layer* membagi fitur ke dalam kelas berdasarkan dataset pelatihan. Kombinasi karakteristik yang paling kuat digunakan untuk menentukan kelas data gambar (Yanuar, 2018). Serta contoh yang mendefinisikan *fully connected layer* dapat dilihat pada Gambar 2.12.



Gambar 2.12 *Fully Connected Layer*

Dimana x adalah masukan pembelajaran fitur ke lapisan yang terhubung penuh, w adalah berat jaringan ixj dengan mengidentifikasi jumlah fitur dan j mewakili jumlah target kelas, b adalah bias, dan y adalah keluaran dari *fully connected layer*.

2.6.7 Aktifasi *Softmax*

Aktivasi *softmax* adalah jenis algoritma *logistic regression* yang dapat mengklasifikasikan lebih dari dua kelas. Klasifikasi standar algoritma *logistic*

regression adalah masalah mengkategorikan kelas biner. Serta perhitungan aktivasi *softmax* dapat dilihat pada Persamaan 2.1.

$$f_j(Z) = \frac{e^{zj}}{\sum_{k} e^{zk}} \quad \text{.....2.1}$$

Notasi fj mewakili hasil fungsi untuk setiap anggota j dalam vektor keluaran kelas. Argumen z adalah hipotesis model pelatihan yang akan dikategorikan menggunakan fungsi *softmax*. *Softmax* juga menghasilkan temuan yang lebih cerdas dan memiliki interpretasi probabilistik yang unggul jika dibandingkan dengan metode klasifikasi lainnya. *Softmax* dapat menghitung probabilitas untuk semua label. Sebuah vektor dengan nilai aktual akan diekstraksi dari label yang ada dan berubah menjadi vektor dengan nilai antara 0 dan 1, yang jika semua dijumlah akan menjadi satu.

2.7 *Confusion Matrix*

Confusion matrix adalah alat untuk evaluasi visual dalam sistem klasifikasi. *Confusion matrix* ini dapat digunakan untuk menilai kualitas model klasifikasi. Akurasi yang diperoleh dari nilai parameter spesifik, seperti *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), *dan False Negative* (FN), ditentukan oleh *confusion matrix* ukuran n, dimana n adalah jumlah kelas unik (Anggreany, 2020). Serta contoh yang mendefinisikan *tabel confusion matrix* dapat dilihat pada Tabel 2.1.

Tabel 2.2 *Confusion Matrix*

Kelas Sebenarnya	Kelas Hasil Klasifikasi	
	Predicted	*Predicted*
Actual	*True Positive (*TP*)*	*True Negative (*TN*)*
Actual	*False Positive (*FP*)*	*False Negative (*FN*)*

1. TP singkatan dari *True Positive*, dan mengacu pada jumlah data positif yang dikategorikan secara akurat oleh sistem.
2. TN singkatan dari *True Negatif*, dan mengacu pada jumlah data negatif yang dikategorikan dengan benar oleh sistem.

3. FN singkatan dari *False Negatif*, yaitu jumlah data negatif yang dikategorikan secara keliru oleh sistem.
4. FP singkatan dari *False Positive*, yaitu jumlah data positif yang dikategorikan secara keliru oleh sistem.

Penilaian klasifikasi dikaji dari berbagai indikator, termasuk indikator akurasi, spesifitas, dan sensitivitas, berdasarkan *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN). Akurasi didefinisikan sebagai rasio proyeksi jumlah aktual dari semua data. Spesifitas adalah metrik yang mencerminkan seberapa baik banyak data bernilai negatif dapat dikategorikan ke dalam klasifikasi negatif. Sensitivitas adalah metrik yang mencerminkan seberapa baik data bernilai positif dapat dikategorikan ke dalam kelas positif. Indikator dihitung menggunakan rumus berikut:

2.7.1 *Accuracy*

Akurasi digunakan sebagai parameter dengan cara yang sama seperti model menyelesaikan klasifikasi. Persamaan berikut dapat digunakan untuk menghitung tingkat akurasi prediksi. Serta perhitungan *confusion matrix* untuk *accuracy* dapat dilihat pada Persamaan 2.2.

$$Akurasi = (\frac{TP+TN}{TP+TN+FP+FN}) \times 100\% \quad \text{..... 2.2}$$

2.7.2 *Precision*

Presisi atau spesifitas menunjukkan akurasi di mana model memprediksi hasil positif dalam serangkaian tindakan prediktif. Nilai presisi dihitung sebagai berikut. Serta perhitungan *confusion matrix* untuk *precision* dapat dilihat pada Persamaan 2.3.

$$Spesifitas = (\frac{TP}{TP+FP}) \times 100\% \quad \text{2.3}$$

2.7.3 *Recall*

Sensifitas atau *recall* digunakan untuk memeriksa kinerja spesifik dari sistem atau kelas. Berikut merupakan perhitungan nilai *recall*. Serta perhitungan *confusion matrix* untuk *recall* dapat dilihat pada Persamaan 2.4.

$$Sensifitas = \left(\frac{TP}{TP+FN}\right) \times 100\% \quad \text{2.4}$$

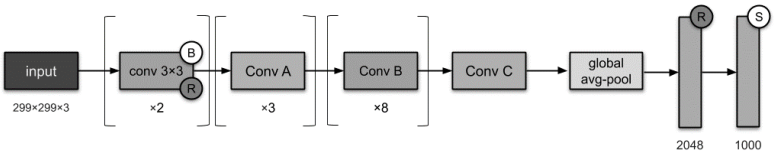
2.7.4 F1 *Score*

Perhitungan F1 *Score* melibatkan perhitungan *recall* dan presisi yang digabungkan untuk mendapatkan nilai F*score*.F1 *score* merupakan nilai rata-rata dari *Fscore*. Berikut ini adalah langkah-langkah perhitungan F1 *Score*. Serta persamaan *confusion matrix* untuk *F1 score* dapat dilihat pada Persamaan 4.6.

$$F1\ Score = 2\ \times \frac{Precission \times Recall}{Precission+Recall} \quad \text{2.4}$$

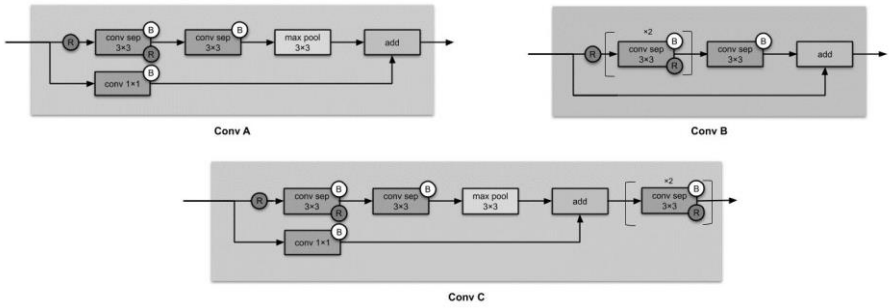
2.8 Arsitektur *Xception*

Xception adalah modifikasi *Inception* dimana modul *Inception* telah digantikan oleh konvolusi terpisah yang mendalam. Singkatnya, arsitektur *Xception* adalah tumpukan linier dari lapisan konvolusi diskrit yang dipisahkan oleh koneksi residual. Hal ini membuat arsitektur relatif sederhana untuk ditentukan dan beradaptasi. Istilah *Xception* adalah singkatan dari "*Extreme Inception*," dan menggunakan masukan gambar yang sama dengan arsitektur 299 x 299 (Chollet, 2016). Serta contoh arsitektur *xception* dapat dilihat pada Gambar 2.13.



Gambar 2.13 Arsitektur *Xception*

Serta contoh *convolution* pada arsitektur xception dapat dilihat pada Gambar 2.14.



Gambar 2.14 Proses Convolusi *Xception*

Sumber: Chollet (2017)

2.9 *Python*

Python adalah bahasa komputer yang menggunakan orientasi objek untuk menjalankan beberapa instruksi secara langsung (interpretatif). *Python* adalah bahasa pemrograman yang paling mudah dipahami. Guido Van Rossum, seorang programmer Belanda, menemukan *python. Python* adalah bahasa pemrograman yang menggunakan semantik dinamis untuk menawarkan kemampuan membaca sintaktik dan dapat mengeksekusi beberapa instruksi secara langsung (interpretatif) menggunakan orientasi objek (*Object Oriented Programming*).*Python* sebagai bahasa yang mampu menggabungkan kemampuan dan sintaks kode yang sangat jelas sekaligus melengkapi fungsionalitas perpustakaan standar yang besar dan komprehensif. Meskipun *python* adalah bahasa pemrograman tingkat tinggi, *python* dirancang untuk menjadi sederhana untuk belajar dan memahami (Baktikominfo, 2019).

2.10 *Flask*

Flask merupakan sebuah *framework web open-source* yang ditulis dalam bahasa pemrograman *Python*, menawarkan kemudahan dan kecepatan dalam membangun aplikasi web. Dalam kategori *micro-framework*, *Flask* menyediakan fitur dasar yang diperlukan untuk membangun aplikasi web, memungkinkan pengembang untuk menciptakan aplikasi yang ringan dan responsif. Keunggulan *Flask* juga terletak pada fleksibilitasnya, yang memungkinkan penggunaan

komponen tambahan dengan mudah, termasuk komponen yang sudah tersedia secara *default* dalam *Flask* (Maulid, 2021).

Salah satu keunggulan utama *Flask* adalah kemudahan penggunaannya. Dokumentasi *Flask* yang lengkap dan mudah dipahami memungkinkan pengembang untuk dengan cepat mempelajari dan menggunakan *Flask*. Selain itu, *Flask* juga menyediakan berbagai ekstensi dan *plugin* yang mempermudah pengembang dalam membangun fitur-fitur kompleks seperti autentikasi pengguna dan integrasi dengan API pihak ketiga.

2.11 *Anaconda*

Anaconda adalah aplikasi yang berfungsi sebagai distribusi bahasa pemrograman *Python* dan *R* yang bersifat *open source*. *Python* sering digunakan untuk berbagai perhitungan ilmiah seperti *machine learning*, pengolahan data dalam skala besar, analisis prediksi, dan lain sebagainya. *Anaconda* bertujuan untuk menyederhanakan berbagai proses manajemen paket dan implementasi. *Anaconda* menyediakan lebih dari 1500 paket distribusi yang populer dan dapat diakses pada berbagai platform sistem operasi seperti *Windows, Linux,* dan *MacOS* (Efanntyo, 2021).

2.12 *Jupyter Notebook*

Jupyter Notebook dimaksudkan untuk memfasilitasi proses komputasi ilmiah yang mencakup dari ekspor sekarang interaktif ke publikasi rekaman. Kode *Jupyter Notebook* disusun ke dalam sel, yang dapat diedit dan dieksekusi secara independen. *Jupyter* mencoba membuat buku catatan lebih mudah diakses. *Jupyter Notebook* adalah proyek *open-source* yang dapat menangani kode dalam berbagai bahasa komputer. *Backend* Bahasa terpisah, yang dikenal sebagai kernel, terhubung dengan *Jupyter* melalui antarmuka bersama. Lebih dari 50 *backend* seperti itu sudah tersedia untuk bahasa mulai dari *C++* sampai *Bash*. *Jupyter Notebook* berevolusi dari proyek *IPython*, yang awalnya menyediakan antarmuka ini hanya untuk *Python*. *Jupyter Notebook* dapat diramban menggunakan peramban web. Berkas disimpan dalam format *JSON* dengan akhiran. ipynb (Kluyver dkk., 2016).

2.13 *TensorFlow*

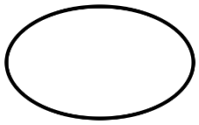
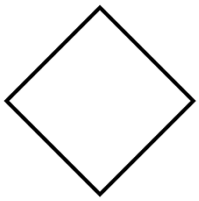
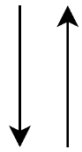
Tensorflow adalah kerangka pembelajaran mendalam yang populer yang dikembangkan oleh tim peneliti dari *Google*. Ini adalah sebuah pustaka sumber terbuka dan gratis untuk ilmu data yang menawarkan banyak kegunaan di berbagai bidang. Salah satunya adalah dalam deteksi objek, di mana *Tensorflow* menyediakan *TensorFlow Object Detection* API. API ini merupakan sebuah alat yang berguna untuk mempermudah proses pembuatan, pelatihan, dan implementasi model deteksi objek. Pengguna dapat menggunakan model deteksi objek yang sudah terlatih seperti Faster *R-CNN, SSD, RetinaNet, ResNet50*, dan masih banyak lagi. *TensorFlow Object Detection API* memberikan akses ke model-model terlatih ini sehingga pengguna dapat dengan mudah menerapkannya dalam proyek deteksi objek mereka (Manajang dkk., 2020).

TensorFlow memiliki kemampuan untuk melatih model menggunakan unit pemrosesan grafis (GPU). Selain itu, salah satu *library* yang digunakan dalam *deep learning* adalah *Keras*. *Keras* menggunakan beberapa fitur dari *TensorFlow* untuk membangun jaringan syaraf tiruan karena *Keras* menyediakan beberapa model dasar CNN yang sudah dioptimalkan untuk memudahkan *deep learning* (Arsal dkk., 2020).

2.14 *Flowchart*

Flowchart adalah representasi visual yang memuat simbol-simbol untuk menunjukkan arah aliran aktivitas dan data dalam eksekusi program. *flowchart* umumnya digunakan dalam pembangunan sistem untuk menetapkan langkah-langkah yang jelas dari input, proses, hingga output (Setiawan, 2021). Selain itu, informasi mengenai bentuk dan nama tanda yang sering digunakan dalam *flowchart* dapat dilihat pada Tabel 2.3.

Tabel 2.3 *Flowchart*

No	Nama	Bentuk	Keterangan
1	*Terminator*		Simbol yang menunjukkan awal atau akhir suatu program
2	*Proses*		Tanda yang mewakili prosedur yang dilakukan oleh computer
3	*Data*		Simbol yang menentukan proses *input* atau *output* tanpa mengandalkan peralatan.
4	*Decision*		Simbol yang menunjukkan kriteria spesifik yang akan menghasilkan salah satu dari dua jawaban potensial, ya atau tidak.
5	*Predefine Proses*		Simbol untuk melaksanakan segmen (sub-program) atau proses
6	*Direct data*		Data yang dapat diakses secara langsung, seperti data yang tersimpan di computer
7	Simbol data *Flow* (arus data)		Garis Penghubung adalah nama lain dari simbol yang digunakan untuk menghubungkan satu simbol dengan simbol lainnya.

2.15 *Unified Modeling Language* (UML)

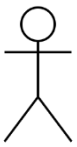
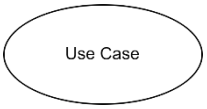
Unified Modeling Language (UML) merupakan sebuah metode pemodelan visual yang digunakan dalam perancangan sistem berorientasi objek. UML dirancang untuk mempermudah pengembangan perangkat lunak dan memenuhi semua kebutuhan pengguna secara efektif, lengkap, dan tepat. Faktor-faktor seperti

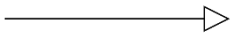
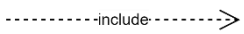
skalabilitas, ketangguhan, keamanan, dan lainnya juga menjadi bagian dari tujuan UML (Faulina, 2023). Berikut ini adalah beberapa contoh diagram UML yang sering digunakan.

2.15.1 *Use Case Diagram*

Use Case Diagram merupakan salah satu jenis diagram dalam U*nified Modeling Language* (UML) yang bertujuan mengilustrasikan hubungan interaktif antara sistem dan aktor. Diagram ini digunakan untuk menjelaskan berbagai jenis interaksi antara pengguna sistem dan sistem itu sendiri (Faulina, 2023). Selain itu, informasi mengenai simbol-simbol yang digunakan dalam *Use Case Diagram* dapat ditemukan pada Tabel 2.4.

Tabel 2.4 *Use Case Diagram* (UCD)

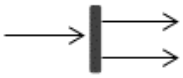
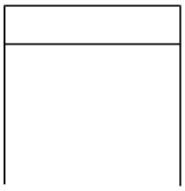
No	Nama	Bentuk	Keterangan
1	*Actor*		Orang proses atau sistem lain yang berinteraksi Dengan sistem informasi yang sedang dibangun sering kali berada di luar sistem informasi itu sendiri. Meskipun simbol yang digunakan untuk menggambarkan aktor adalah gambar orang, biasanya aktor tersebut dinamai dengan menggunakan kata benda di awal frasa untuk menggambarkan perannya.
2	*Use Case*		Fungsionalitas sistem yang disediakan dalam bentuk unit-unit yang saling bertukar pesan antara unit atau aktor sering kali dinyatakan dengan menggunakan kata kerja di awal frasa dalam nama *use case.*

No	Nama	Bentuk	Keterangan
3	*Association*		Komunikasi antara aktor dan *use case* yang berpartisipasi dalam use case atau memiliki interaksi dengan aktor tersebut.
4	*Extend*		Relasi *use case* tambahan mengacu pada sebuah *use case* di mana *usecase* tambahan tersebut dapat berdiri sendiri, meskipun memiliki nama depan yang sama dengan *use case* yang ditambahkannya.
5	*Generalization*		Hubungan generalisasi dan spesialisasi, juga dikenal sebagai hubungan umum-khusus, terjadi antara dua *use case* di mana satu *use case* memiliki fungsi yang lebih umum dibandingkan dengan yang lainnya.
6	*Include*		Relasi *use case* tambahan terjadi ketika sebuah *use case* membutuhkan *use case* lain untuk menjalankan fungsionalitasnya atau sebagai syarat untuk menjalankan *use case* tersebut.

2.15.2 *Activity Diagram*

Activity diagram adalah jenis diagram yang digunakan untuk memodelkan berbagai proses yang terjadi dalam suatu sistem. Diagram ini mengilustrasikan urutan proses yang berlangsung dalam sistem secara vertikal, mencerminkan jalannya proses dalam sistem (Faulina, 2023). Selain itu, informasi mengenai simbol-simbol yang digunakan dalam *activity diagram* dapat diakses pada Tabel 2.5.

Tabel 2.5 *Activity Diagram*

No	Nama	Bentuk	Keterangan
1	*Initial*		Status awal aktivitas sistem, yang terdapat dalam diagram aktivitas, ditunjukkan oleh sebuah state awal.
2	*Activity*		Aktivitas yang dilakukan oleh sistem biasanya diawali dengan sebuah kata kerja.
3	*Decision*		Asosiasi percabangan terjadi ketika lebih dari satu aktivitas digabungkan menjadi satu.
4	*Join*		Asosiasi penggabungan terjadi ketika lebih dari satu aktivitas digabungkan menjadi satu.
5	*Final*		Status akhir yang dilakukan oleh sistem, yang terdapat dalam diagram aktivitas, ditunjukkan oleh sebuah *state* akhir.
6	*Swimline*		Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

BAB 3. METODE PENELITIAN

3.1 Waktu dan Tempat Pelaksanaan

3.1.1 Waktu Pelaksanaan

Proses penelitian skripsi berjudul "*Implementasi Image Processing* Pada Penyakit Daun Tanaman Padi Dengan Metode CNN (Studi Kasus Kabupaten Tuban)" dilakukan dalam waktu 8 bulan, dimulai setelah seminar proposal dan berakhir pada bulan Januari tahun 2024. Jadwal kegiatan dapat dilihat pada Tabel 3.1.

Tabel 3.1 Waktu Pelaksanaan

No	kegiatan	Bulan 7	Bulan 8	Bulan 9	Bulan 10	Bulan 11	Bulan 12	Bulan 1	Bulan 2
1	Identifikasi Masalah								
2	Studi literatur dan studi pustaka								
3	Pengumpulan dataset								
4	Pengolahan data								
5	Implementasi sistem								
6	Evaluasi dan pembahasan								

3.1.2 Tempat Pelaksanaan

Pelaksanaan penelitian skripsi yang berjudul "Implementasi *Image Processing* Pada Penyakit Daun Tanaman Padi Dengan Metode CNN (Studi Kasus Kabupaten Tuban)" yaitu di Tuban.

3.2 Alat dan Bahan

3.2.1 Alat

Dalam penelitian ini, digunakan sebuah laptop dan *software* pendukung untuk mengembangkan dan mengeksekusi program. Berikut adalah rincian alat yang digunakan:

a. Perangkat Keras

Dalam penelitian "Implementasi *Image Processing* Pada Penyakit Daun Tanaman Padi Dengan Metode CNN (Studi Kasus Kabupaten Tuban)", perangkat keras yang dipakai adalah laptop Lenovo V330 dengan spesifikasi sebagai berikut:

1) Ram 8 GB
2) SSD 128 GB
3) SSD 512 GB
4) *Processor* AMD Ryzen 5 2500U *with* Radeon Vega Mobile Gfx

b. Perangkat Lunak

Dalam penelitian "Implementasi *Image Processing* Pada Penyakit Daun Tanaman Padi Dengan Metode CNN (Studi Kasus Kabupaten Tuban)", perangkat lunak yang dipakai sebagai berikut:

1) *Windows 10 Pro 64-bit*
2) *Visual Studio Code*
3) *Microsoft Word*
4) *Microsoft Excel*
5) *Jupyter Notebook*
6) *Microsoft Edge*

3.2.2 Bahan

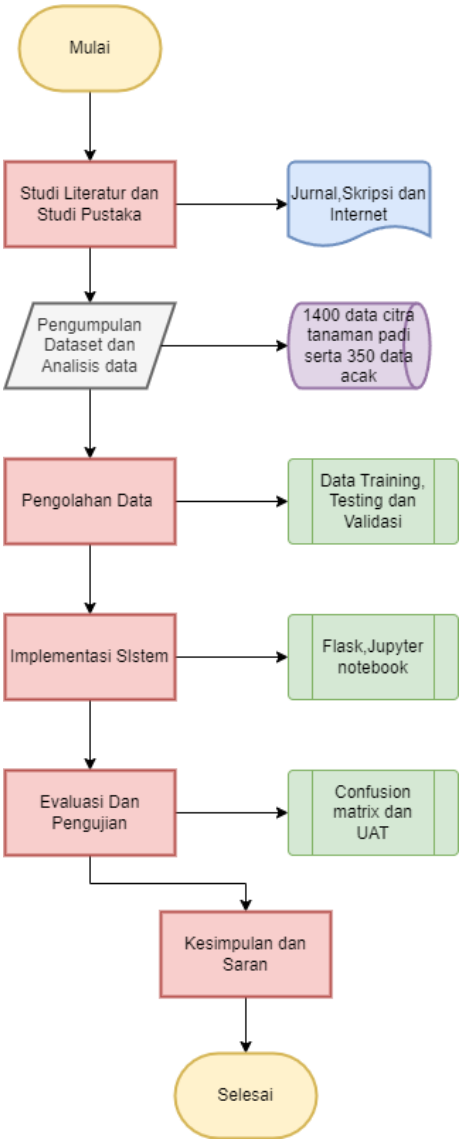
Dalam penelitian "Implementasi *Image Processing* Pada Penyakit Daun Tanaman Padi Dengan Metode CNN (Sudi Kasus Kabupaten Tuban)", Bahan-bahan yang diperlukan untuk melakukan penyelidikan ini adalah sebagai berikut:

a) Data citra tanaman Penyakit padi yang diperoleh dari Dinas Ketahanan Pangan, Pertanian dan Perikanan Kabupaten Tuban tahun 2021 - 2022.

b) Data citra tanaman Penyakit padi yang diperoleh dari observasi secara langsung ke lapangan.
c) Data citra tanaman padi sehat yang diperoleh dari *website kaggle* (Rice Leafs (kaggle.com)).
d) Data citra random yang digunakan untuk mencegah *overfitting* yang diambil dari *website kaggle*.
e) Data citra taman padi yang digunakan untuk melakukan uji aplikasi serta penentuan jenis penyakit bersama pakar yang diambil dari jurnal.

3.3 Alur Penelitian

Berikut ini adalah alur penelitian yang dilakukan pada penelitian yang berjudul "Implementasi *Image Processing* Pada Penyakit Daun Tanaman Padi Dengan Metode CNN (Sudi Kasus Kabupaten Tuban)". Serta alur penerlitian dapat dilihat pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

3.3.1 Studi Literatur

Pada tahap ini studi literatur digunakan untuk menyiapkan semua kebutuhan yang penting, seperti pengumpulan data atau sumber terkait, seperti permasalahan, landasan teori, metodologi penelitian, kebutuhan perangkat lunak, dan penelitian serupa. Berikut merupakan referensi yang dipelajari:

a. Jurnal atau *paper* penelitian sebelumnya terkait penerapan *image processing* menggunakan CNN.

3.3.2 Pengumpulan *Dataset* dan Analisis Data

Dalam penelitian, proses pengumpulan data harus menggunakan metode yang sesuai dengan sifat individualitas penelitian yang dilakukan. Data dapat dibagi menjadi dua jenis yaitu data primer dan data sekunder. Pada penelitian ini, akan menggunakan data sekunder dan data primer. Berikut penjelasan mengenai data primer dan sekunder:

a. Data Primer

Data primer adalah metode pengumpulan data yang dilakukan langsung oleh peneliti dari sumber data, seperti melalui pengisian angket, observasi, atau wawancara yang dilakukan secara langsung. Metode ini membutuhkan biaya dan waktu yang lebih banyak dibandingkan dengan pengumpulan data sekunder.

b. Data Sekunder

Data sekunder dilakukan dengan menggunakan data yang telah diperoleh dari penelitian lain dengan tujuan yang berbeda, seperti data yang tersedia di kantor pemerintah, perpustakaan, atau internet. Metode ini lebih cepat dan lebih murah dibandingkan dengan metode data primer, namun terkadang data yang diperoleh tidak sesuai dengan kebutuhan penelitian dan harus disesuaikan terlebih dahulu. Pemilihan metode pengumpulan data yang tepat akan sangat mempengaruhi kualitas hasil penelitian yang dilakukan.

Berikut beberapa alasan menggunakan data primer dan data sekunder pada penelitian ini:

1) Dengan menggunakan data primer, peneliti dapat memastikan bahwa data yang didapatkan sesuai dengan tujuan penelitian dan dapat dikumpulkan secara langsung dari sumber data.
2) Dengan menggunakan data primer, peneliti dapat mengumpulkan data yang lebih spesifik dan relevan dengan penelitian yang dilakukan.
3) Dengan menggunakan data sekunder, data akan menjadi lebih terstruktur karena umumnya sudah diproses dan diatur dengan baik, sehingga dapat menghemat waktu dan usaha dalam proses analisis data.

4) Dengan menggunakan data sekunder, tingkat akurasi data akan menjadi tinggi karena umumnya dikumpulkan oleh organisasi atau lembaga yang telah memastikan keakuratannya, sehingga dapat menghasilkan data yang lebih akurat dan dapat dipercaya.

3.3.3 Pengolahan Data

Pada tahap ini melibatkan pemisahan data *training*, data *testing* dan data validasi untuk menjalankan proses penelitian dan mencapai tujuan penelitian dengan metode yang telah ditentukan. Metode yang diusulkan dalam penelitian ini dengan menggunakan *Convolutional Neural Network* (CNN) dan menggunakan arsitektur *Xception*.

3.3.4 Implementasi Sistem

Pada tahap ini Implementasi merujuk pada tahap pelaksanaan atau penerapan desain yang telah dirancang, dan ini mencakup eksekusi penuh dari rencana yang telah dibuat. Tahap implementasi sistem dilakukan dengan melakukan pengkodean menggunakan bahasa pemrograman *python*.

3.3.5 Evaluasi dan Pengujian

Pada tahap ini tujuan dari pengujian ini adalah untuk menguji akurasi tingkat keberhasilan metode yang diusulkan. *Confusion matrix* digunakan sebagai metode pengujian dalam penelitian ini.

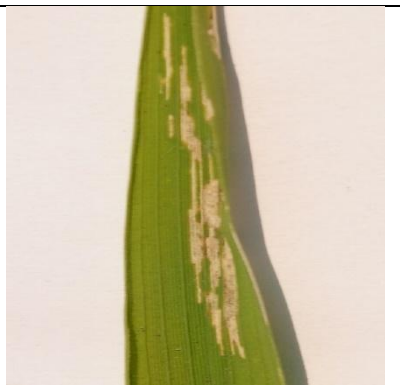
3.4 Tahap Pengumpulan Data

Penelitian ini menggunakan data citra penyakit daun padi yang diperoleh dari tiga sumber: Dinas Ketahanan Pangan, Pertanian, dan Perikanan Kabupaten Tuban, pengambilan data langsung di lapangan dengan metode observasi, dan dataset *Kaggle*. Sebanyak 1750 data terkumpul dan dibagi menjadi 5 kelas utama: *Xanthomonas oryzae pv. oryzicola* (penyakit leaf blight pada daun), *Pyricularia oryzae Cav* (penyakit blast pada daun), *Brown Spot* (Bercak Coklat),*Healthy* (Daun Sehat) dan Data acak juga ditambahkan untuk meningkatkan keakuratan model

dengan tujuan agar diagnosis penyakit daun padi dapat dilakukan secara cepat dan akurat. Detail sampel data citra dapat dilihat pada Tabel 3.2.

Tabel 3.2 Sampel *Dataset*

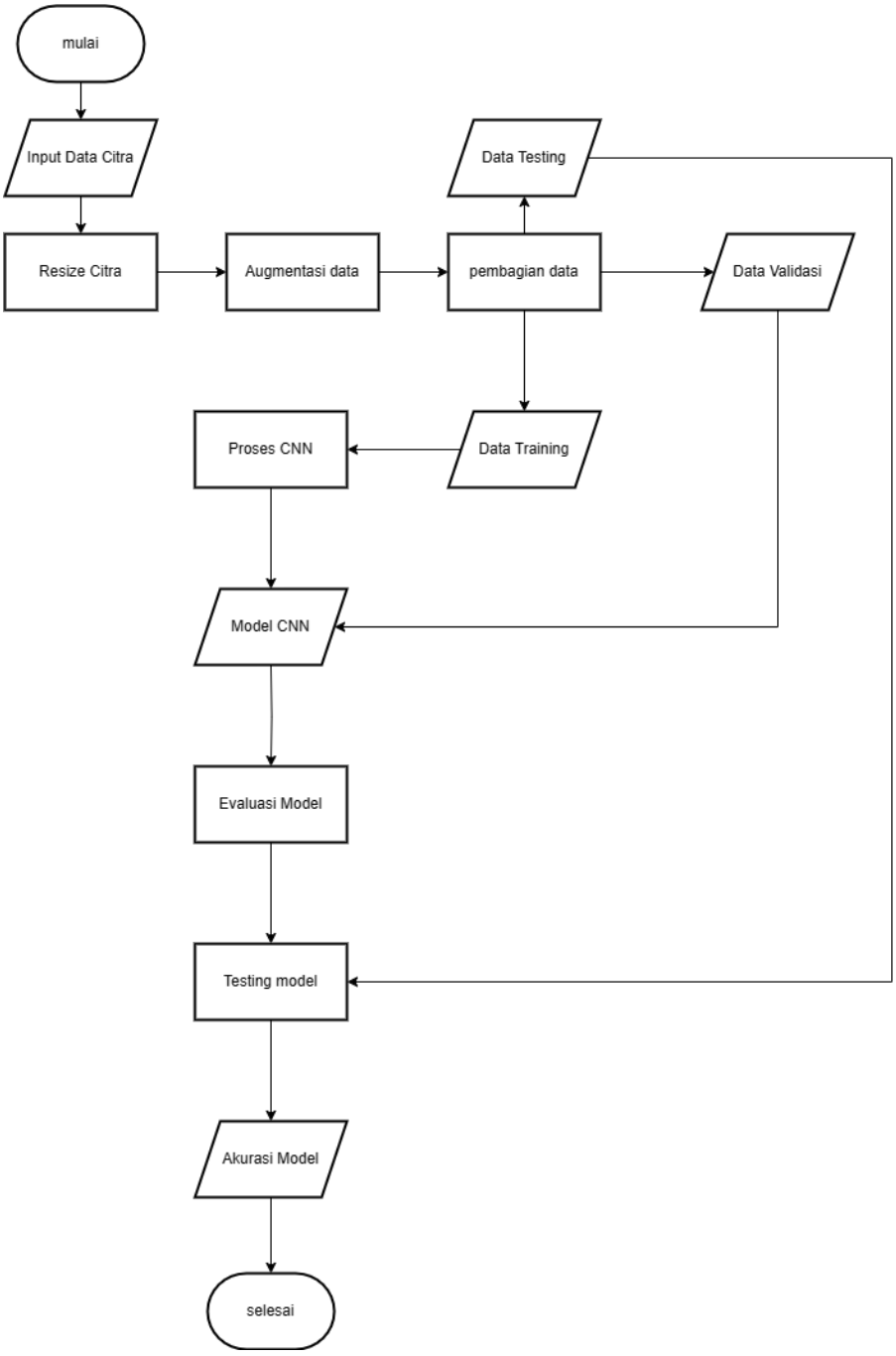
No	Nama Daun Padi	Gambar Daun Padi
1	Daun Padi Sehat	
2	Daun Padi *Brown Spot*	
3	Daun Padi Penyakit *Blast*	

No	Nama Daun Padi	Gambar Daun Padi
4	Daun Padi Penyakit *Leaf Blight*	

Tabel 3.3 menunjukkan jumlah data citra RGB yang digunakan untuk objek penelitian ini, yang terdiri dari 350 citra untuk kelas *Xanthomonas oryzae pv. oryzicola*, 350 citra untuk kelas *Pyricularia oryzae Cav*, 350 citra untuk kelas *Brown Spot*, 350 citra untuk kelas *Healthy* dan 350 citra untuk data acak.

3.5 Tahap Pengolahan Data

Identifikasi penyakit daun padi menggunakan CNN dengan arsitektur *Xception* berdasarkan klasifikasi data gambar. Serta, alur tahapan pengolahan data dapat dilihat pada Gambar 3.2.



Gambar 3.2 Proses Pengolahan Data

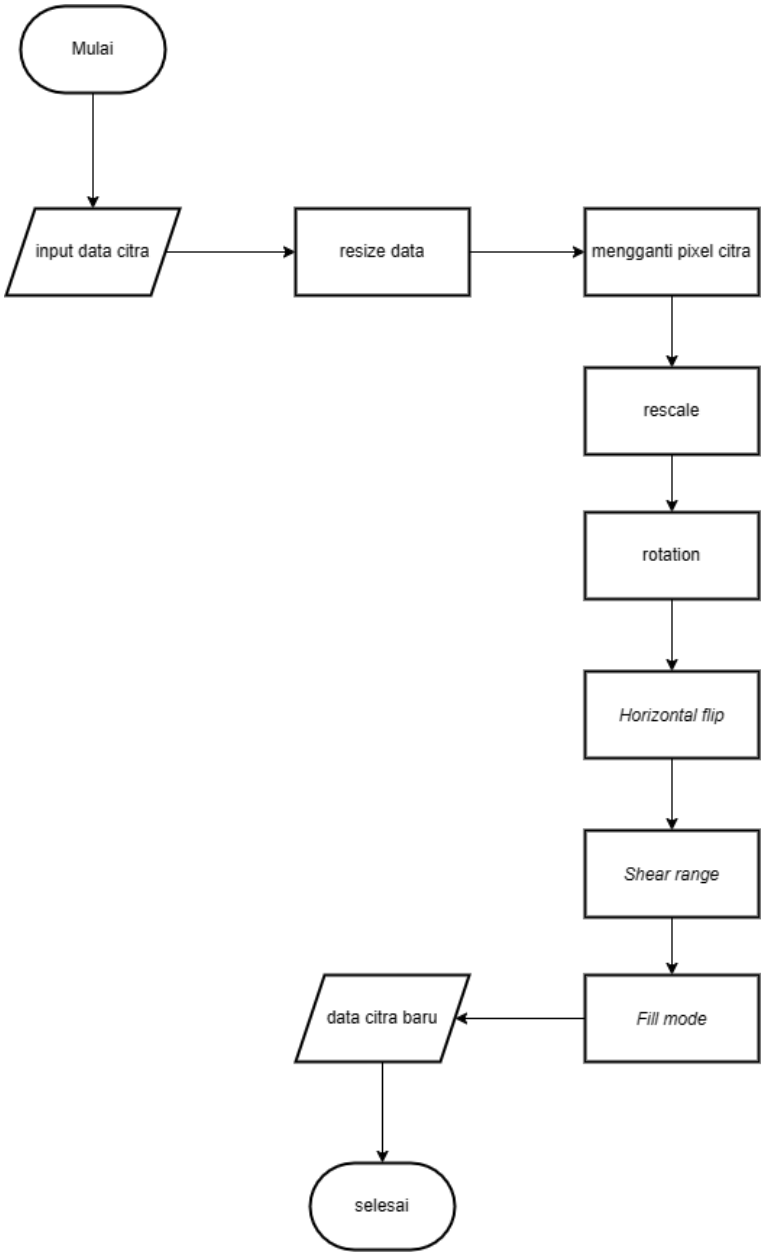
3.5.1 *Input* Data Citra

Untuk memperoleh data citra penyakit daun padi, digunakan dua metode pengambilan data. Pertama, data diperoleh dari Dinas Ketahanan Pangan, Pertanian dan Perikanan Kabupaten Tuban. Kedua, data diambil secara langsung di lapangan

menggunakan metode observasi lapangan. Data yang berhasil dikumpulkan berisi berbagai jenis citra penyakit daun padi, seperti citra *Xanthomonas oryzae pv*. *oryzicola* (penyakit leaf blight pada daun), *Pyricularia oryzae Cav* (penyakit blast pada daun), *Brown Spot* (Bercak Coklat),*Healthy* (Daun Sehat) dan Data acak.

3.5.2 *Preprocessing*

Preprocessing merupakan tahap awal dalam pengolahan data yang bertujuan untuk menghasilkan data yang optimal. Salah satu tindakan *preprocessing* yang dilakukan adalah me*resize* citra daun dalam *dataset* menjadi ukuran yang seragam, yaitu 224 x 224 piksel. Data yang telah dilakukan proses *resize* dilakukan proses augmentasi yang meliputi proses *rescale* yaitu Merubah skala piksel data RGB gambar dari rentang (0-255) ke rentang angka (0-1) akan mempermudah proses pelatihan data, *rotation* yaitu Memutar gambar ke kiri atau kanan sebesar 20 derajat. *Horizontal flip* yaitu Membalik gambar secara horizontal secara acak, *shear range* yaitu Melakukan penskalaan citra ke kanan dan kiri sebesar 0.2%. Data yang telah melalui *proses preprocessing* ini akan dijadikan input untuk proses klasifikasi. Tahapan *preprocessing* data citra mengikuti diagram alur yang ditunjukkan pada Gambar 3.3.



Gambar 3.3 Tahapan *Preprocessing*

3.5.3 Pembelajaran Fitur

Proses pembelajaran fitur dan klasifikasi pada penelitian ini dilakukan dalam satu arsitektur, yaitu CNN *Xception.* Arsitektur ini terdiri dari beberapa layer seperti *convolution layer, ReLU layer, max pooling*, dan *fully connected layer*. Pada CNN *Xception*, fitur-fitur data citra dipelajari menggunakan *convolution layer, ReLU layer,* dan *max-pooling*. Selanjutnya, hasil pembelajaran fitur data citra

diklasifikasi pada *fully connected layer* dengan menentukan kelas berdasarkan probabilitas kelas tertinggi pada *softmax layer*. Setelah proses pembelajaran, sistem klasifikasi akan menghasilkan model optimum yang akan diuji dengan *confusion matrix*.

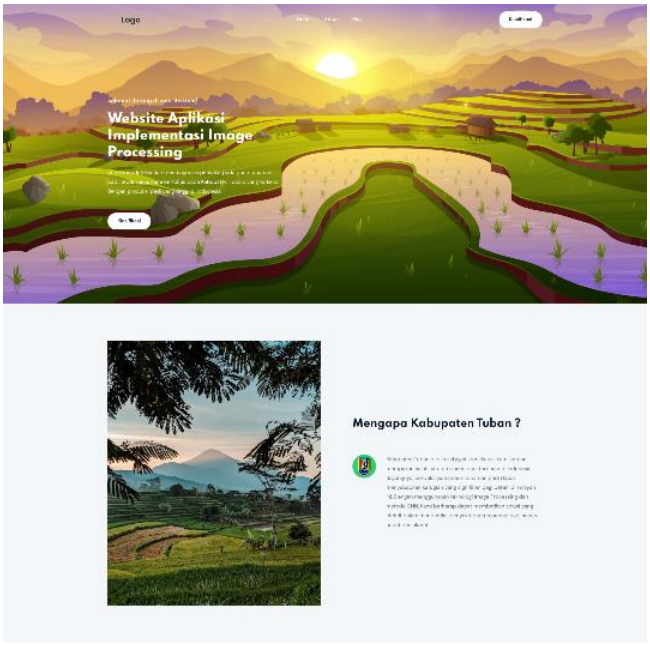
3.5.4 Analisa Hasil Klasifikasi

Untuk melakukan evaluasi terhadap hasil klasifikasi, digunakan *confusion matrix* yang berfungsi untuk mengevaluasi tingkat akurasi metode *convolutional neural network*. *Confusion matrix* tersebut berbentuk tabel yang memperlihatkan hasil perhitungan klasifikasi secara keseluruhan. Tujuan dari analisa hasil klasifikasi ini adalah untuk menentukan model jaringan yang paling optimal dalam melakukan klasifikasi penyakit daun padi. Selain itu, dari analisa hasil klasifikasi yang dilakukan juga dapat dilihat apakah model jaringan yang telah dibentuk dapat diterapkan dalam deteksi penyakit daun padi atau tidak. Hasil evaluasi data tersebut akan diukur dengan menggunakan akurasi untuk melihat seberapa besar tingkat ketepatan model dalam melakukan klasifikasi.

3.6 Desain Interface

3.6.1 Rancangan Desain *Interface* Halaman Beranda

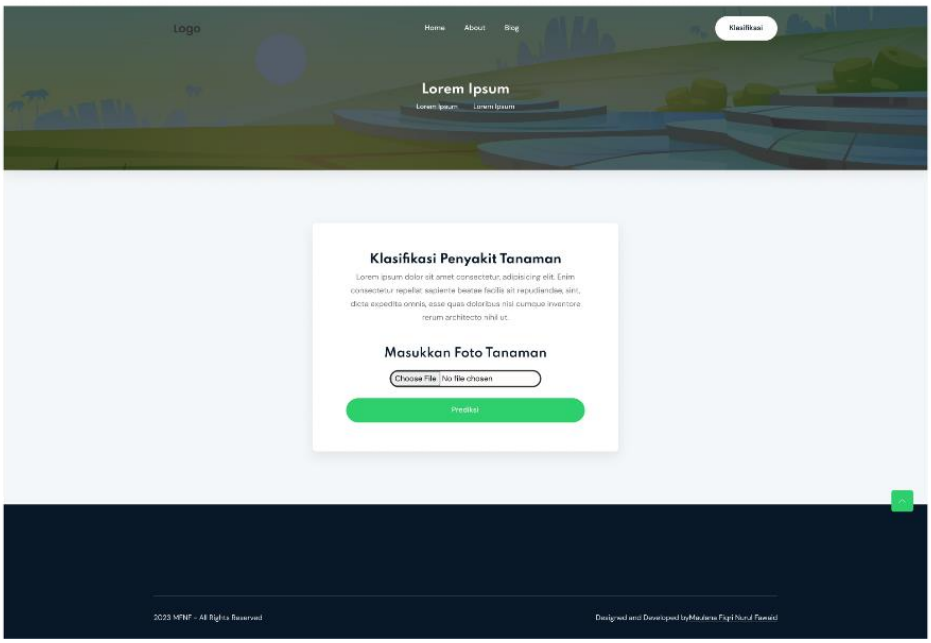
Halaman beranda merupakan halaman awal proses ketika pengguna masuk kedalam sistem. Pada halaman ini pengguna dapat melihat mengenai fitur-fitur yang tersedia pada aplikasi ini serta informasi-informasi tentang Penyakit tanaman padi. Serta, Rancangan desain interface dapat dilihat pada Gambar 3.4.



Gambar 3.4 Desain *Interface* Beranda

3.6.2 Rancangan Desain *Interface* Halaman Klasifikasi Penyakit

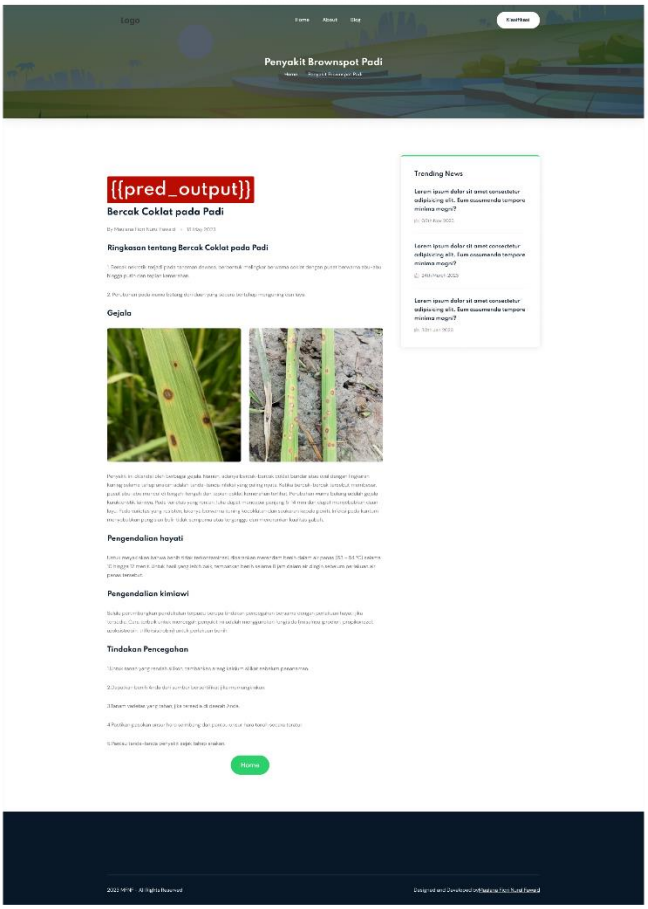
Halaman klasifikasi merupakan halaman bagi pengguna untuk melakukan klasifikasi daun tanaman padi. Pada halaman ini pengguna dapat memasukkan foto tanaman daun padi yang sehat maupun yang sakit tetapi pada aplikasi ini hanya fokus pada beberapa jenis citra seperti *Xanthomonas oryzae pv. oryzicola* (penyakit leaf blight pada daun), *Pyricularia oryzae Cav* (penyakit blast pada daun), *Brown Spot* (Bercak Coklat),*Healthy* (Daun Sehat) dan Data acak. Serta, Rancangan desain interface dapat dilihat pada Gambar 3.5.



Gambar 3.5 Desain Interface Klasifikasi Tanaman

3.6.3 Rancangan Desain *Interface* Halaman *Output* Klasifikasi

Halaman *Output* klasifikasi merupakan halaman bagi pengguna untuk menampilkan hasil klasifikasi daun tanaman padi. Pada halaman ini pengguna dapat menerima informasi tentang hasil tanaman yang dimasukkan pada halaman klasifikasi dan akan di prediksi oleh sistem foto tersebut terserang penyakit atau tidak. Pada halaman ini juga pengguna akan diberikan penjelasan tentang penyakit yang di prediksi, gejala-gejala serta cara untuk mengatasi penyakit yang diprediksi oleh sistem. Serta, Rancangan desain interface dapat dilihat pada Gambar 3.6.



Gambar 3.6 Desain *Interface Output* Klasifikasi

BAB 4. HASIL DAN PEMBAHASAN

4.1 Studi Literatur

Pada tahap ini, studi literatur menjadi landasan utama untuk menyusun dalam pengerjaan penelitian ini. Proses ini melibatkan beberapa tahap seperti mencari di berbagai sumber literatur, termasuk jurnal ilmiah, buku, dan artikel terkait untuk memperoleh pemahaman tentang permasalahan yang akan diteliti.

a. Jurnal atau paper penelitian sebelumnya terkait penerapan *image processing* menggunakan CNN.
b. Data tanaman Padi yang diperoleh dari Dinas Ketahanan Pangan, Pertanian, dan Perikanan Kabupaten Tuban
c. Data acak digunakan sebagai pembeda, sehingga tidak terjadi kesalahan dalam pembacaan data gambar. CNN merupakan model prediktif yang sangat kuat, namun tidak selalu dapat memprediksi kapan model tersebut akan melakukan kesalahan.

4.2 Deskripsi Data

Penelitian ini mengaplikasikan struktur *Xception* pada metode *Convolutional Neural Network* (CNN) untuk analisis citra daun tanaman padi. Sebanyak 1750 citra daun padi dikumpulkan dan dikelompokkan ke dalam lima kategori utama: *xanthomonas oryzae pv. oryzicola* (penyakit *leaf blight* pada daun), *pyricularia oryzae Cav* (penyakit blast pada daun), *brown Spot* (Bercak Coklat), *healthy* (daun sehat) dan data acak untuk meningkatkan stabilitas model. Contoh visual dari citra-citra yang digunakan dapat dilihat pada Gambar 4.1.

Gambar 4.1 Dataset yang Digunakan

4.2.1 Pemuatan *Dataset*

Pada langkah ini, *dataset* yang telah dimiliki dipanggil dan jumlah *dataset* di setiap kelasnya dihitung. Contoh implementasi kode untuk memuat *dataset* dapat dilihat pada Kode Program 4.1. Proses ini penting untuk memastikan integritas data dan memahami distribusi kelas dalam *dataset*. Pemanggilan *dataset* juga memungkinkan langkah-langkah berikutnya dalam analisis atau pemrosesan data dilakukan secara akurat.

Kode Program 4.1 Kode Memuat Dataset

```
1) sdir=r'E:\0.Tidak Hapus\SKRIPSI Fiqri\Dataset\dataset coba-
   coba'
2) filepaths=[]
3) labels=[]
4) classlist=os.listdir(sdir)
5)
6) for klass in classlist:
7)     classpath=os.path.join(sdir,klass)
8)     if os.path.isdir(classpath):
9)         flist=os.listdir(classpath)
10)            for f in flist:
11)                fpath=os.path.join(classpath,f)
12)                filepaths.append(fpath)
13)                labels.append(klass
14)
15)   Fseries= pd.Series(filepaths, name='filepaths')
16)   Lseries=pd.Series(labels, name='labels')
17)
18)   df=pd.concat([Fseries, Lseries], axis=1)
19)
20)   print (df['labels'].value_counts())
```

Penjelasan Kode Program:

1) `*sdir=r'E:\0.Tidak Hapus\SKRIPSI Fiqri\Dataset\dataset coba-coba'*`: Ini adalah variabel `*sdir*` yang menunjukkan jalur direktori yang berisi kumpulan data. Jalur ini digunakan untuk mencari berkas dalam skrip.
2) `*filepaths*=[]`:Variabel `*filepaths*` adalah daftar kosong yang akan digunakan untuk menyimpan path berkas.
3) `*labels*=[]`: Variabel `*labels*` adalah daftar kosong yang akan digunakan untuk menyimpan label kelas.
4) `*classlist=os.listdir(sdir)*`: Menggunakan fungsi `*os.listdir()*`, ini mendapatkan daftar direktori kelas dalam direktori yang ditentukan oleh `*sdir*` dan menyimpannya dalam variabel `classlist`.
5) `*for klass in classlist*`: Melakukan iterasi melalui setiap direktori kelas dalam `*classlist*`.
6) `*classpath=os.path.join(sdir,klass)*`: Ini menggabungkan jalur direktori `*sdir*` dengan nama kelas saat ini dalam iterasi untuk membentuk jalur lengkap ke direktori kelas tersebut.
7) `*if os.path.isdir(classpath)*`: Memeriksa apakah `*classpath*` adalah sebuah direktori. Jika ya, lanjutkan dengan iterasi.
8) `*flist=os.listdir(classpath)*`: Mendapatkan daftar berkas dalam direktori kelas saat ini dan menyimpannya dalam variabel `*flist*`.
9) `*for f in flist*`: Melakukan iterasi melalui setiap berkas dalam `*flist*`.
10) `*fpath=os.path.join(classpath,f)*`: Menggabungkan jalur direktori kelas dengan nama berkas untuk membentuk jalur lengkap ke berkas tersebut.
11) `*filepaths.append(fpath)*`: Menambahkan `*fpath*` ke dalam daftar `*filepaths*`.
12) `*labels.append(klass)*`: Menambahkan label kelas saat ini ke dalam daftar `*labels*`.
13) `*Fseries= pd.Series(filepaths, name='filepaths')*`: Membuat objek Series Pandas `*Fseries*` dari daftar `*filepaths*` dengan nama *'filepaths'*.
14) `*Lseries=pd.Series(labels, name='labels')*`: Membuat objek Series Pandas `*Lseries*` dari daftar `*labels*` dengan nama *'labels'*.

15) `*df=pd.concat([Fseries, Lseries], axis=1)*`: Menggabungkan dua Series `*Fseries*` dan `*Lseries*` menjadi satu DataFrame `df` dengan menggunakan fungsi `*pd.concat()*`. Mereka digabungkan secara berdampingan *(axis=1)*.
16) `*print (df['labels'].value_counts())*`:Mencetak jumlah kemunculan setiap label kelas dalam kolom *'labels'* dari DataFrame `*df*` dengan menggunakan metode `*value_counts()*` pada kolom *'labels'*. Ini akan memberikan distribusi jumlah kelas dalam dataset.

4.2.2 Pembagian *Dataset*

Dalam pembagian *dataset* merupakan langkah yang terpenting dalam penelitian ini dengan tujuan untuk memisahkan data menjadi *training*, *validation*, dan *test*. *Training* digunakan untuk melatih model, *validation* digunakan untuk penyesuaian parameter dan pencegahan *overfitting* serta *test* berfungsi untuk mengukur kinerja model pada data yang belum pernah dilihat sebelumnya, menguji generalisasi model.

1. Pembagian *dataset* 80%:20%

Pembagian *dataset* "80%:20%" mengacu pada pembagian proporsi data dalam dua set utama: *training* set (8 bagian), *validation set* dan *testing set* (2 bagian). Proses ini menggunakan *dataset* yang dibagi menjadi 80% untuk pelatihan dan 20% untuk validasi. Pembagian ini didasarkan pada prinsip Pareto yang umum ditemui dalam data mining, dimana 80% performa model didapatkan dari pelatihan terhadap 20% data. Serta contoh implementasi kode pembagian *dataset* dapat dilihat pada Kode Program 4.2.

Kode Program 4.2 Pembagian Dataset 80%:20%

```
1) train_split=.8
2) test_split=.1
3) dummy_split=test_split/(1-train_split)
4) train_df, dummy_df=train_test_split(df,
   train_size=train_split, shuffle=True, random_state=123)
5) test_df, valid_df=train_test_split(dummy_df,
   train_size=dummy_split, shuffle=True, random_state=123)
6) print ('train_df length: ', len(train_df), '  test_df length:
   ', len(test_df), '  valid_df length: ', len(valid_df))
```

Penjelasan Kode Program:

1) `*train_split=.8*`:Menetapkan proporsi data yang akan dialokasikan untuk set pelatihan (80%).
2) `*test_split=.1*`:Menetapkan proporsi data yang akan dialokasikan untuk set uji (10%).
3) `*dummy_split=test_split/(1-train_split)*`: Menghitung proporsi data yang akan digunakan untuk set validasi. Proporsi ini dihitung sebagai selisih antara proporsi data yang dialokasikan untuk set uji dan proporsi data yang dialokasikan untuk set pelatihan.
4) `*train_df, dummy_df=train_test_split(df, train_size=train_split, shuffle=True, random_state=123)*`: Memisahkan dataset `df` menjadi set pelatihan (`*train_df*`) dan sisa dataset (`*dummy_df*`) menggunakan fungsi `*train_test_split*` dari *scikit-learn*. Proporsi set pelatihan diatur sesuai dengan nilai `*train_split*`, dan data diacak sebelum pembagian dengan `*shuffle=True*`. `*random_state=123*` digunakan untuk memastikan hasil yang dapat direproduksi.
5) `*test_df, valid_df=train_test_split(dummy_df, train_size=dummy_split, shuffle=True, random_state=123)*`: Memisahkan sisa dataset *(`dummy_df`*) menjadi set uji (`*test_df*`) dan set validasi (`*valid_df*`). Proporsi set validasi diatur sesuai dengan nilai `*dummy_split*`, yang telah dihitung sebelumnya. Data diacak sebelum pembagian dengan `*shuffle=True*`. `*random_state=123*` digunakan untuk memastikan hasil yang konsisten.
6) `*print ('train_df length: ', len(train_df), ' test_df length: ', len(test_df), ' valid_df length: ', len(valid_df))*`: Mencetak panjang (jumlah baris) dari set pelatihan (`*train_df*`), set uji (`*test_df*`), dan set validasi (`*valid_df*`) untuk memberikan informasi tentang ukuran relatif dari masing-masing set dalam dataset yang telah dibagi.

2. Pembagian *dataset* 70%:30%

Pembagian *dataset* "70%:30%" mengacu pada pembagian proporsi data dalam dua set utama: *training* set (7 bagian), *validation set* dan *testing set* (3 bagian).

Proses ini menggunakan *dataset* yang dibagi menjadi 70% untuk pelatihan dan 30% untuk validasi serta *testing*. Pembagian ini didasarkan dengan menggunakan 70% untuk pelatihan dan 30% untuk pengujian serta validasi, karena memiliki jumlah data yang cukup besar untuk melatih model dengan baik, sambil tetap menyisakan sejumlah data yang signifikan untuk menguji kinerja model. Serta contoh implementasi kode pembagian *dataset* dapat dilihat pada Kode Program 4.3.

Kode Program 4.3 Pembagian Dataset 70%:30%

```
# Split the data into train, validation, and test sets
train_split = 0.7
valid_split = 0.2
test_split = 0.1

# Calculate the split sizes
train_size = int(train_split * len(df))
valid_size = int(valid_split * len(df))

# Split the data
train_df, remaining = train_test_split(df,
   train_size=train_size, shuffle=True, random_state=123)
valid_df, test_df = train_test_split(remaining,
   train_size=valid_size, shuffle=True, random_state=123)

print('train_df length:', len(train_df), '  valid_df
  length:',len(valid_df), '  test_df length:', len(test_df))
```

Penjelasan kode program:

1) *`train_split = 0.7`*, *`valid_split = 0.2`*, *`test_split = 0.1`*:Variabel ini menentukan pecahan dari dataset utuh yang akan dialokasikan untuk set pelatihan, validasi, dan uji.
2) *`train_size = int(train_split * len(df))`*, *`valid_size = int(valid_split * len(df))`*: Ini menghitung ukuran set pelatihan dan validasi dengan mengalikan pecahan masing-masing dengan panjang total DataFrame `df` (jumlah baris).
3) *`train_df, remaining = train_test_split(df, train_size=train_size, shuffle=True, random_state=123)`*: Fungsi *`train_test_split`* dari *scikit-learn* digunakan untuk membagi DataFrame *`df`* menjadi set pelatihan dan sisa data. Parameter *`train_size`* menentukan ukuran set pelatihan, *`shuffle=True`* mengacak data sebelum pembagian, dan *`random_state=123`* digunakan untuk memastikan hasil yang dapat direproduksi.

4) *`valid_df, test_df = train_test_split(remaining, train_size=valid_size, shuffle=True, random_state=123)`:* Sisa data dari pembagian sebelumnya dibagi menjadi set validasi dan uji dengan proporsi yang ditentukan oleh *`valid_size`*. Seperti sebelumnya, *`shuffle=True`* mengacak data sebelum pembagian, dan *`random_state=123`* digunakan untuk memastikan hasil yang konsisten.
5) *`print('Panjang train_df:', len(train_df), ' Panjang valid_df:', len(valid_df), ' Panjang test_df:', len(test_df))`:* Ini mencetak panjang (jumlah baris) dari set pelatihan *(`train_df`)*, set validasi *(`valid_df`)*, dan set uji *(`test_df`)*. Panjang ini memberikan informasi tentang ukuran relatif dari masing-masing set dalam dataset yang telah dibagi.

4.3 *Pre-Processing* data

Pada tahap ini dilakukan agar tidak terjadi *overfitting*, yaitu kondisi di mana model memiliki kinerja yang baik selama proses pelatihan tetapi buruk saat diuji dengan data baru, dilakukan proses augmentasi data citra daun saat tahap pelatihan. Berbagai teknik augmentasi data ini diuraikan dalam Tabel 4.1 di bawah ini:

Tabel 4.1 Teknik *Augmentasi*

Parameter	**Nilai**	**Keterangan**
Resize	224 x 224	Mengubah ukuran gambar menjadi 224 x 224 piksel.
Rescale	1./255	Menskalakan nilai piksel gambar antara 0 dan 1.
Rotation range	20	Memutar gambar secara acak dalam rentang -20 derajat hingga 20 derajat.
Horizontal flip	*True*	Membalik gambar secara horizontal secara acak.
Shear range	0.2	Memiringkan gambar secara acak dalam rentang -0.2 hingga 0.2.
Fill mode	*nearest*	Mengisi area kosong yang dihasilkan oleh transformasi gambar dengan nilai piksel terdekat.

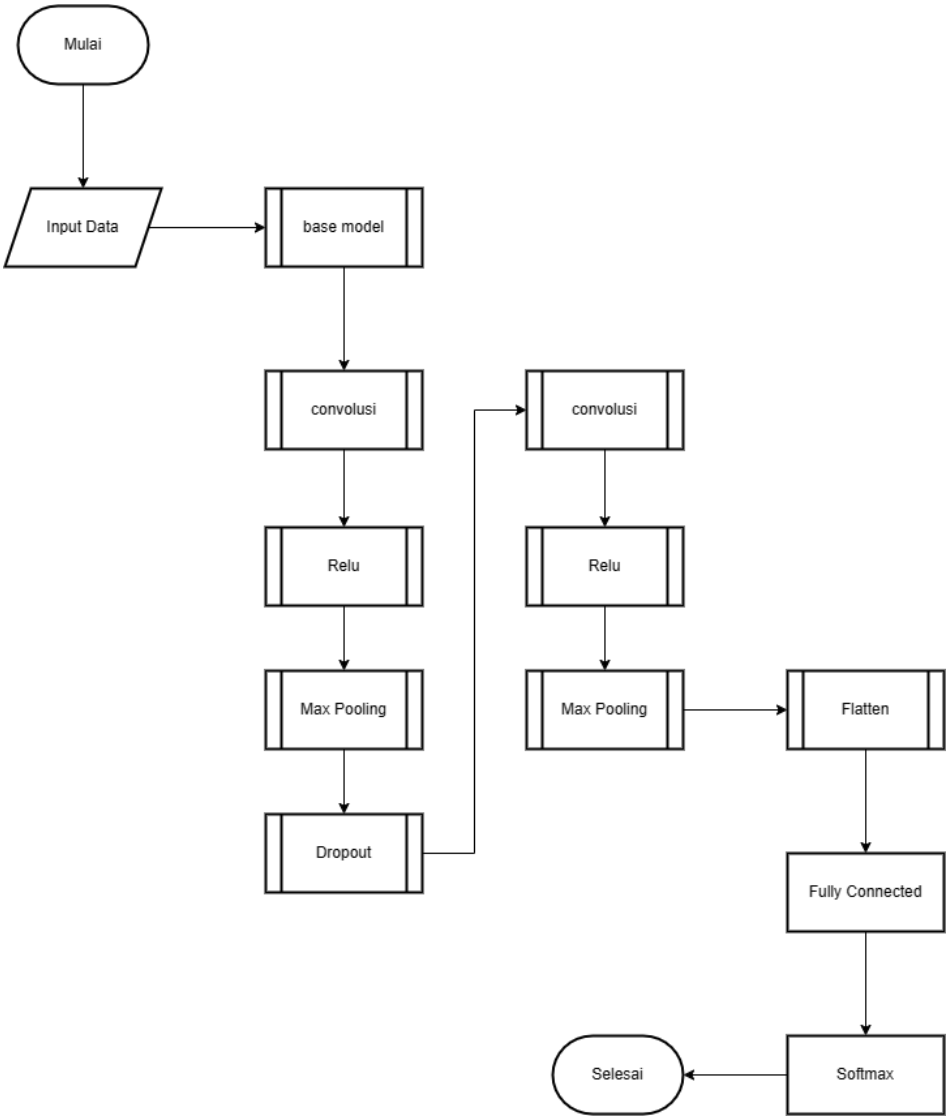
Hasil dari proses yang tercantum dalam proses *augmentasi* dapat dilihat dalam Gambar 4.2 berikut:

Gambar 4.2 Hasil *Augmentasi*

Gambar 4.2 di atas, terlihat perubahan yang terjadi pada daun padi setelah melalui proses augmentasi.

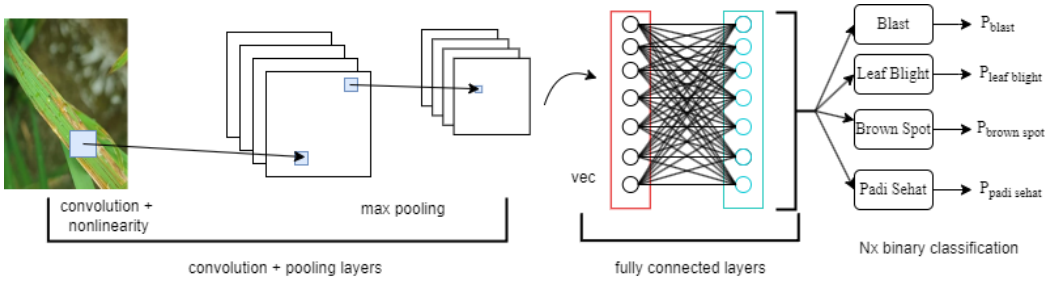
4.4 Perancangan dan Pemodelan Arsitektur CNN

Pembangunan model merupakan proses pembelajaran mesin yang bertujuan untuk menemukan pola tertentu dalam data. Proses ini dimulai dengan mempersiapkan data citra, kemudian dilatih menggunakan metode *Convolutional Neural Network* (CNN). Proses *training model* CNN memiliki beberapa tahapan, seperti *input data*, *convolution*, *pooling*, *fully connected*, dan *output*. Pada tahap *input data*, data citra dimasukkan ke dalam model CNN. Tahap *convolution* menggunakan filter untuk mengekstraksi fitur-fitur penting dari data citra. Tahap *pooling* menggabungkan hasil *convolution* untuk mengurangi dimensi data dan meningkatkan efisiensi komputasi. Tahap *fully connected* menghubungkan hasil *pooling* ke *output model*. Pada tahap ini, model CNN akan menghasilkan klasifikasi atau prediksi berdasarkan fitur-fitur yang telah diekstrak. Tahap *output* menghasilkan hasil klasifikasi atau prediksi dari model CNN. Berikut untuk tahapan dari *flowchart* CNN ditunjukkan pada Gambar 4.3.



Gambar 4.3 *Flowchart* CNN

Serta Contoh implementasi menggunakan arsitektur CNN ditunjukkan pada Gambar 4.4.



Gambar 4.4 Arsitektur CNN

Untuk cara perhitungan manual metode CNN akan dijelaskan mulai dari Input data hingga aktivasi softmax :

A. Input Data

Tahap pertama dalam melatih model adalah memasukkan data citra daunke dalam input layer. Pada layer ini,citra gambar diubah menjadi mastriks tiga dimensi dengan ukuran panjang x lebar x 3 channel RGB (*Red, Green, Blue*). Dalam penelitian ini, nilai RGB pada setiap piksel dinormalisasi menjadi 0-1 untuk mempermudah proses komputasi, seperti yang dijelaskan pada Gambar 4.4.

Nilai Pixel Layer Red

0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

Nilai Pixel Layer Green

0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

filter

0	-1	0
-1	4	-1
0	-1	0

Nilai Pixel Layer Blue

0	0	0	0	0	0
0	0.8824	0.5804	0.4353	0.898	0
0	0.8745	0.4941	0.3647	0.898	0
0	0.8549	0.4314	0.3176	0.8824	0
0	0.7686	0.2275	0.2941	0.8745	0
0	0	0	0	0	0

Gambar 4.5 Inputan Citra

Gambar 4.5 menunjukkan hasil citra yang telah dinormalisasi dari citra awal input. Peneliti menggunakan filter Sobel dikarenakan filter ini dapat mendeteksi tepi gambar serta dapat mendeteksi perubahan intensitas gambar secara horizontal dan vertikal.

B. Tahap *Convolusi*

Pada tahap ini, Proses *convolusi* berfungsi untuk mengekstraksi fitur-fitur (*feature map*) yang ada pada citra dengan menggunakan filter. Penelitian ini menggunakan filter berukuran 3x3 piksel, dengan stride 1 dan serta padding same yang bertujuan untuk mempertahankan ukuran output sama dengan ukuran input, yang menyederhanakan arsitektur CNN dan perhitungan selanjutnya. Dalam CNN, filter size juga disebut sebagai kernel atau *weight* yang nilainya diinisialisasi secara acak. Contoh prosesnya dapat dilihat pada Gambar 4.6 berikut.

0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

x

0	-1	0
-1	4	-1
0	-1	0

=

2.090196	0.71765	0.43529	2.1451
1.219608	-0.2314	-0.349	1.25098
1.258824	-0.149	-0.3804	1.25882
2.015686	0.05882	0.23529	2.23922

Gambar 4.6 Contoh Proses Convolusi

Pada Ilustrasi Gambar 4.5, terdapat input citra berukuran 6x6 yang akan dikonvolusi menggunakan filter berukuran 3x3. Hasilnya akan menghasilkan *feature map* berukuran 4x4. Operasi konvolusi dilakukan dengan menggeser kernel konvolusi piksel per piksel. Hasil konvolusi disimpan dalam matriks baru.Proses perhitungannya adalah sebagai berikut: (0 x 0) + (-1 x 0) + (0 x 0) + (-1 x 0) + (4 x 0.9529) + (-1 x 0.7686) + (0 x 0) + (-1 x 0.9529) + (0 x 0.702) = 2,090196

Dari proses tersebut, dihasilkan nilai 2,090196 yang akan menempati satu sel di citra baru. Setelah itu, filter digeser ke kanan dan ke bawah hingga terbentuk satu citra baru dan seterusnya.

1) Perhitungan pada layer *red*

Posisi 1

0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

2.0902

Posisi 2

0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

0.71765

Posisi 3

0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

0.43529

Posisi 4

0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

2.1451

Posisi 5

0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

1.21961

Posisi 6

0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

-0.2314

Posisi 7					
0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

-0.349

Posisi 8					
0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

1.25098

Posisi 9					
0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

1.25882

Posisi 10					
0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

-0.149

Posisi 11					
0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

-0.3804

Posisi 12					
0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

1.25882

Posisi 13					
0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

2.01569

Posisi 14					
0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

0.05882

Posisi 15					
0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

0.23529

Posisi 16					
0	0	0	0	0	0
0	0.9529	0.7686	0.702	0.949	0
0	0.9529	0.702	0.6549	0.949	0
0	0.9373	0.6627	0.6157	0.9412	0
0	0.8745	0.5451	0.5843	0.9412	0
0	0	0	0	0	0

2.23922

Gambar 4.7 Perhitungan Setiap Posisi *Convolusi Red*

Hasil			
2.0902	0.71765	0.43529	2.1451
1.21961	-0.2314	-0.349	1.25098
1.25882	-0.149	-0.3804	1.25882
2.01569	0.05882	0.23529	2.23922

Gambar 4.8 Hasil Perhitungan Proses *Convolusi Red*

2) Perhitungan pada layer *green*

Posisi 1					
0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

2.007843

Posisi 2

0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

0.643137

Posisi 3

0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

0.419608

Posisi 4

0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

2.043137

Posisi 5

0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

1.152941

Posisi 6

0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

-0.26275

Posisi 7

0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

-0.28235

Posisi 8					
0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

1.196078

Posisi 9					
0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

1.2

Posisi 10					
0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

-0.21961

Posisi 11					
0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

-0.30588

Posisi 12					
0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

1.184314

Posisi 13					
0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

1.870588

Posisi 14					
0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

0.082353

Posisi 15					
0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

0.329412

Posisi 16					
0	0	0	0	0	0
0	0.9059	0.7176	0.6706	0.9059	0
0	0.898	0.651	0.6392	0.9098	0
0	0.8824	0.6118	0.6078	0.898	0
0	0.8196	0.5255	0.5882	0.8902	0
0	0	0	0	0	0

2.07451

Gambar 4.9 Perhitungan Setiap Posisi *Convolusi Green*

Hasil			
2.00784	0.64314	0.41961	2.04314
1.15294	-0.2627	-0.2824	1.19608
1.2	-0.2196	-0.3059	1.18431
1.87059	0.08235	0.32941	2.07451

Gambar 4.10 Hasil Perhitungan Proses *Convolusi Green*

3) Perhitungan pada layer *blue*

Posisi 1					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

2.07451

Posisi 2					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

0.509804

Posisi 3					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

-0.10196

Posisi 4					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

2.258824

Posisi 5					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

1.266667

Posisi 6					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

-0.27451

Posisi 7					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

-0.68627

Posisi 8					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

1.447059

Posisi 9					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

1.345098

Posisi 10					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

-0.16863

Posisi 11					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

-0.70196

Posisi 12					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

1.439216

Posisi 13					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

1.992157

Posisi 14					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

-0.58431

Posisi 15					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

-0.24314

Posisi 16					
0	0	0	0	0	0
0	0.88235	0.58039	0.43529	0.89804	0
0	0.87451	0.49412	0.36471	0.89804	0
0	0.8549	0.43137	0.31765	0.88235	0
0	0.76863	0.22745	0.29412	0.87451	0
0	0	0	0	0	0

2.321569

Gambar 4.11 Perhitungan Setiap Posisi *Convolusi Blue*

Hasil			
2.07451	0.509804	-0.10196	2.258824
1.266667	-0.27451	-0.68627	1.447059
1.345098	-0.16863	-0.70196	1.439216
1.992157	-0.58431	-0.24314	2.321569

Gambar 4.12 Hasil Perhitungan Proses *Convolusi Blue*

4) Penjumlahan semua RGB

Pada tahap ini, Proses *penjumlahan* RGB digunakan untuk menentukan hasil citra yang baru.

Hasil Red

2.0902	0.71765	0.43529	2.1451
1.21961	-0.2314	-0.349	1.25098
1.25882	-0.149	-0.3804	1.25882
2.01569	0.05882	0.23529	2.23922

+

Hasil Green

2.00784	0.64314	0.41961	2.04314
1.15294	-0.2627	-0.2824	1.19608
1.2	-0.2196	-0.3059	1.18431
1.87059	0.08235	0.32941	2.07451

+

Hasil Blue

2.07451	0.509804	-0.102	2.25882
1.26667	-0.27451	-0.6863	1.44706
1.3451	-0.16863	-0.702	1.43922
1.99216	-0.58431	-0.2431	2.32157

Hasil Penjumlahan Setiap RGB

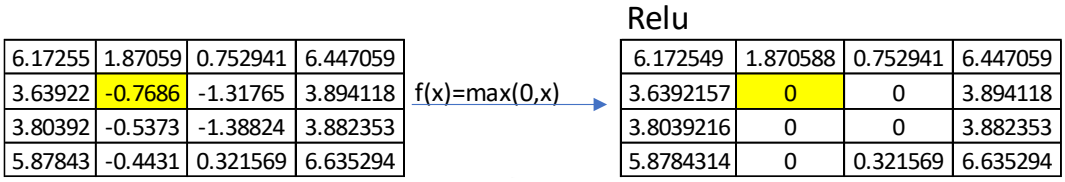
6.17255	1.87059	0.75294	6.44706
3.63922	-0.7686	-1.3176	3.89412
3.80392	-0.5373	-1.3882	3.88235
5.87843	-0.4431	0.32157	6.63529

Gambar 4.13 Hasil Penjumlahan RGB

Pada Ilustrasi Gambar 4.13, Proses perhitungannya adalah sebagai berikut: (2,0902+2,00784+2,07451) = 6,17255 . Dari proses tersebut, dihasilkan nilai 6,17255 yang akan menempati satu sel di citra baru. Setelah itu, dilakukan semua penjumlahan hingga terbentuk satu citra baru.

C. Perhitungan *ReLU*

Pada tahap ini, ReLU berfungsi untuk mengubah nilai masukan pada neuron feature map. Nilai masukan yang negatif akan diubah menjadi 0, sedangkan nilai yang tidak negatif akan tetap sama. Contoh implementasi perhitungan ReLU dapat dilihat pada Gambar 4.14. Contohnya pada koordinar (2,2) nilai -0,786 berubah menjadi angka 0 dikarenakan nilai bernilai negatif.



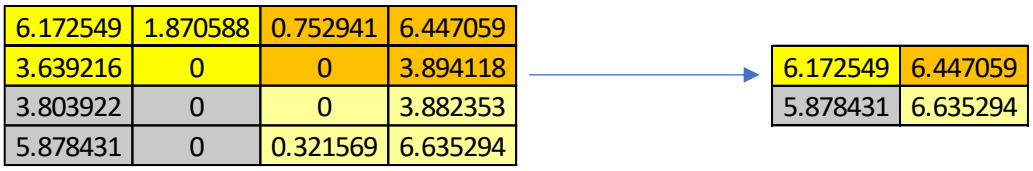
6.17255	1.87059	0.752941	6.447059
3.63922	-0.7686	-1.31765	3.894118
3.80392	-0.5373	-1.38824	3.882353
5.87843	-0.4431	0.321569	6.635294

6.172549	1.870588	0.752941	6.447059
3.6392157	0	0	3.894118
3.8039216	0	0	3.882353
5.8784314	0	0.321569	6.635294

Gambar 4.14 Hasil ReLU

D. Proses *Max-Pooling*

Lapisan pooling berfungsi untuk memperkecil ukuran spasial citra, serta mengurangi jumlah parameter dan perhitungan dalam jaringan saraf tiruan. Lapisan ini menerima masukan dari hasil *feature map* yang dihasilkan oleh lapisan konvolusi. Penelitian ini menggunakan *max pooling* dengan ukuran 2x2, sehingga menghasilkan citra yang lebih kecil. Gambar 4.8 menunjukkan contoh input *max pooling*. Proses ini menghasilkan *output* dengan nilai terbesar dari semua nilai input. Pada contoh ini, nilai terbesar adalah 6,172549, sehingga nilai tersebut yang diambil.



6.172549	1.870588	0.752941	6.447059
3.639216	0	0	3.894118
3.803922	0	0	3.882353
5.878431	0	0.321569	6.635294

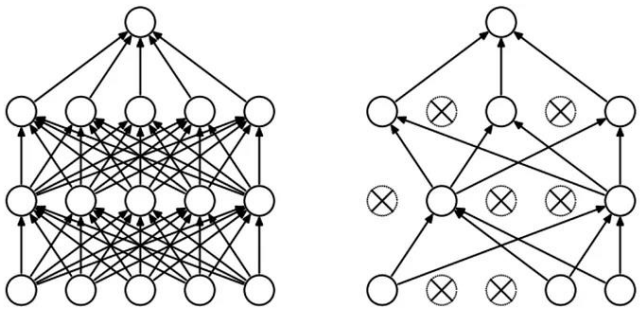
6.172549	6.447059
5.878431	6.635294

Gambar 4.15 Hasil *Max-Pooling*

Gambar 4.15 menunjukkan proses *max pooling*. Pada gambar tersebut, elemen berwarna kuning memiliki nilai terbesar 6,172549, sehingga nilai yang digunakan untuk input citra baru adalah 6,172549. Hal yang sama berlaku untuk elemen berwarna *orange* dengan nilai terbesar 6,447059, di mana nilai tersebut yang digunakan sebagai input citra baru.

E. *Dropout Layer*

Dropout layer adalah teknik regularisasi dalam jaringan saraf tiruan (JST) yang bertujuan untuk mencegah *overfitting*. *Overfitting* terjadi ketika JST belajar terlalu banyak detail dari data pelatihan dan tidak dapat menggeneralisasi dengan baik ke data baru. *Dropout layer* bekerja dengan menonaktifkan (*dropping out*) beberapa neuron secara acak pada jaringan saraf tiruan selama proses pelatihan. Hal ini memaksa JST untuk belajar dari berbagai kombinasi neuron, sehingga membuatnya lebih robust dan generalizable. Manfaat dropout layer: meningkatkan generalizability JST, mencegah *overfitting*, meningkatkan akurasi JST pada data baru, dan mempercepat proses pelatihan JST. Kekurangan *dropout layer*: meningkatkan kompleksitas model, membutuhkan waktu pelatihan yang lebih lama, dan dapat mengurangi akurasi JST pada data pelatihan. Berikut ilustrasi penggunaan droput layer pada Gambar 4.16.



Gambar 4.16 *Droput Layer*

F. *Flatten Layer*

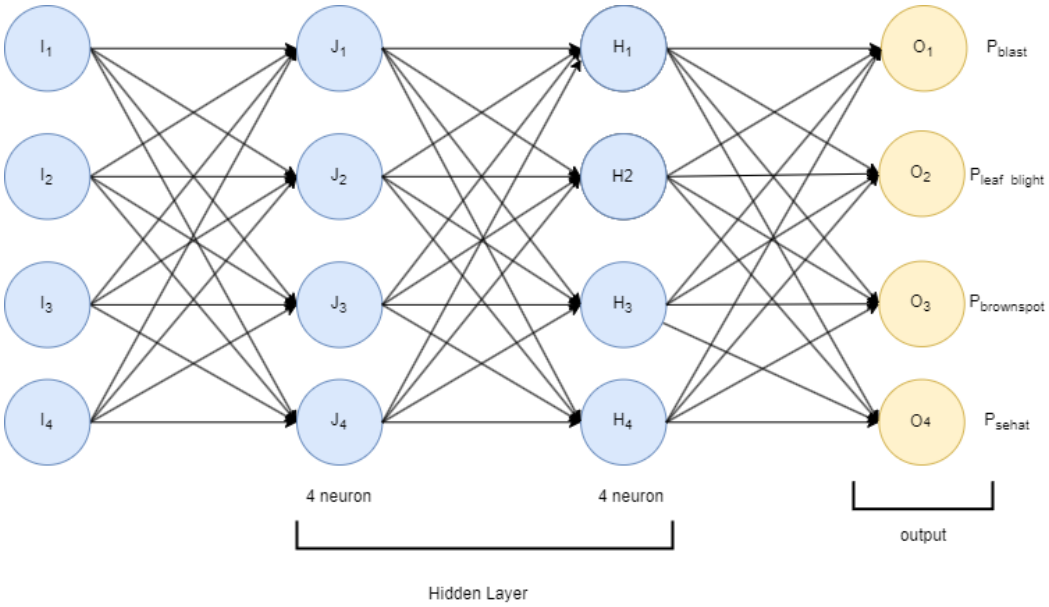
Pada tahap ini dilakukan pengubahan data matriks yang sebelumnya merupakan *matrix* 2 dimesi menjadi hanya satu dimensi vektor (*flatten*). Berikut ilustrasi proses *flatten layer* pada Tabel 4.2

Tabel 4.2 Proses *Flatten Layer*

Data *Matrix*			Data Vektor (x)			
6.17254902	6.447058824		6.17254902	6.447058824	5.878431373	6.635294118
5.878431373	6.635294118					

G. *Fully Connected Layer*

Tahap ini berfungsi untuk menghitung keluaran dari *Flatten Layer*, yaitu vektor satu dimensi. Vektor ini kemudian menjadi masukan untuk tahap selanjutnya, yaitu *Fully Connected Layer*. Berikut adalah ilustrasi dari *Fully Connected Layer*:



Gambar 4.17 Ilustrasi *Fully Connected Layer*

Pada ilustrasi 4.17 diatas merupakan contoh dari *fully connected layer* untuk I merupakan hasil dari *flatten* untuk J dan H merupakan *hidden layer* yang terdapat pada proses *fully connected layer* untuk O sendiri merupakan *output layer*.

Berikut contoh perhitungan dari *hidden layer* J untuk persamaan rumusnya ditunjukkan pada persamaan 4.1.

$$\sum_{i=1}^{N} I_1 * v_{ij} = J_i \quad \text{----------4.1}$$

I_1, I_2, I_3, dan I_4 adalah nilai dari tahap *flatten* sebelumnya untuk v_{ij} merupakan nilai bobot yang didapatkan secara acak pada tahap CNN.

Untuk menentukan bobot (*weight*) dapat dimisalkan:

a) Bobot awal W11 = 0.21, W12 = 0.45, W13 = 0.2, W14 =0.1
b) Bobot kedua W21 = 0.3, W22 = 0.50, W23 = 0.18, W24 =0.05
c) Bobot Ketiga W31 = 0.31, W32 = 0.43, W33 = 0.05, W34 =0.21

d) Bobot Keempat W41 = 0.2, W32 = 0.1, W33 = 0.13, W34 =0.3

Berikut untuk perhitungan menentukan hidden layer J:

1) $J1 = (I1 \times w11) + (I2 \times w12) + (I3 \times w13) + (I4 \times w14)$
$= (6.17254902 \times 0.21) + (6.447058824 \times 0.45) +$
$(5.878431373 \times 0.2) + (6.635294118 \times 0.1) = 6.036$
2) $J2 = (I1 \times w21) + (I2 \times w22) + (I3 \times w23) + (I4 \times w24)$
$= (6.17254902 \times 0.3) + (6.447058824 \times 0.50) +$
$(5.878431373 \times 0.18) + (6.635294118 \times 0.05) = 6.465$
3) $J3 = (I1 \times w31) + (I2 \times w32) + (I3 \times w33) + (I4 \times w34)$
$= (6.17254902 \times 0.31) + (6.447058824 \times 0.43) +$
$(5.878431373 \times 0.05) + (6.635294118 \times 0.21) = 6.373$
4) $J4 = (I1 \times w41) + (I2 \times w42) + (I3 \times w43) + (I4 \times w44)$
$= (6.17254902 \times 0.2) + (6.447058824 \times 0.1) +$
$(5.878431373 \times 0.13) + (6.635294118 \times 0.3) = 4.633$

Setelah melakukan perhitungan *hidden layer* J kemudian menghitung perhitungan dari hidden layer H untuk persamaan rumusnya ditunjukkan pada persamaan 4.2.

$$\sum_{i=1}^{N} J_1 * w_{ij} = H_i \quad \text{----------4.2}$$

J_1, J_2, J_3, dan J_4 adalah nilai dari perhitungan hidden layer J untuk w_{ij} merupakan nilai bobot yang didapatkan secara acak pada tahap CNN.

Untuk menentukan bobot (*weight*) dapat dimisalkan:

a) Bobot awal W11 = 0.11, W12 = 0.35, W13 = 0.23, W14 =0.21
b) Bobot kedua W21 = 0.35, W22 = 0.30, W23 = 0.45, W24 =0.35
c) Bobot Ketiga W31 = 0.11, W32 = 0.33, W33 = 0.25, W34 =0.1
d) Bobot Keempat W41 = 0.25, W32 = 0.11, W33 = 0.3, W34 =0.23

Berikut untuk perhitungan menentukan *hidden layer* H:

1) $H1 = (J1 \times w11) + (J2 \times w12) + (J3 \times w13) + (J4 \times w14)$
   $= (6.036 \times 0.11) + (6.465 \times 0.35) + (6.373 \times 0.23) +$
   $(4.633 \times 0.21) = 5.366$
2) $H2 = (J1 \times w21) + (J2 \times w22) + (J3 \times w23) + (J4 \times w24)$
   $= (6.036 \times 0.35) + (6.465 \times 0.30) + (6.373 \times 0.45) +$
   $(4.633 \times 0.35) = 8.542$
3) $H3 = (J1 \times w31) + (J2 \times w32) + (J3 \times w33) + (J4 \times w34)$
   $= (6.036 \times 0.11) + (6.465 \times 0.33) + (6.373 \times 0.25) +$
   $(4.633 \times 0.1) = 4.854$
4) $H4 = (J1 \times w41) + (J2 \times w42) + (J3 \times w43) + (J4 \times w44)$
   $= (6.036 \times 0.25) + (6.465 \times 0.11) + (6.373 \times 0.3) +$
   $(4.633 \times 0.23) = 5.198$

Setelah melakukan perhitungan *hidden layer* H kemudian menghitung perhitungan dari *output layer* O untuk persamaan rumusnya ditunjukkan pada persamaan 4.3.

$$\sum_{i=1}^{N} H_1 * x_{ij} = O_i$$ ----------4.3

H_1, H_2, H_3, dan H_4 adalah nilai dari perhitungan *hidden layer* H untuk x_{ij} merupakan nilai bobot yang didapatkan secara acak pada tahap CNN.

Untuk menentukan bobot (*weight*) dapat dimisalkan:

a) Bobot awal W11 = 0.1, W12 = 0.23, W13 = 0.13, W14 =0.21
b) Bobot kedua W21 = 0.2, W22 = 0.4, W23 = 0.3, W24 =0.37
c) Bobot Ketiga W31 = 0.11, W32 = 0.13, W33 = 0.25, W34 =0.3
d) Bobot Keempat W41 = 0.22, W32 = 0.11, W33 = 0.35, W34 =0.12

Berikut untuk perhitungan menentukan *output layer* O:

1) $O1 = (H1 \times w11) + (H2 \times w12) + (H3 \times w13) + (H4 \times w14)$

$$= (5.366 \times 0.1) + (8.542 \times 0.23) + (4.854 \times 0.13) + (5.198 \times 0.21) = 4.223$$

2) $O2 = (H1 \times w21) + (H2 \times w22) + (H3 \times w23) + (H4 \times w24)$

$$= (5.366 \times 0.2) + (8.542 \times 0.4) + (4.854 \times 0.3) + (5.198 \times 0.37) = 7.869$$

3) $O3 = (H1 \times w31) + (H2 \times w32) + (H3 \times w33) + (H4 \times w34)$

$$= (5.366 \times 0.11) + (8.542 \times 0.13) + (4.854 \times 0.25) + (5.198 \times 0.3) = 4.473$$

4) $04 = (H1 \times w41) + (H2 \times w42) + (H3 \times w43) + (H4 \times w44)$

$$= (5.366 \times 0.22) + (8.542 \times 0.11) + (4.854 \times 0.35) + (5.198 \times 0.12) = 4.442$$

H. Perhitungan Aktivasi *Softmax*

Tahap ini berfungsi untuk menghitung keluaran dari *Flatten Layer*, yaitu vektor satu dimensi. Vektor ini kemudian menjadi masukan untuk tahap selanjutnya, yaitu *Fully Connected Layer*. Berikut adalah ilustrasi dari *Fully Connected Layer*: dengan rumus dan ilustrasi sebagai berikut:

$$s(o_i) = \frac{e^{o_i}}{\sum_{j=1}^{n} e^{o_j}} \quad \text{----------4.7}$$

$$s(o_1) = \frac{e^{4.223}}{e^{4.223} + e^{7.869} + e^{4.473} + e^{4.442}} = 0.20$$

$$s(o_2) = \frac{e^{7.869}}{e^{4.223} + e^{7.869} + e^{4.473} + e^{4.442}} = 0{,}37$$

$$s(o_3) = \frac{e^{4.473}}{e^{4.223} + e^{7.869} + e^{4.473} + e^{4.442}} = 0{,}21$$

$$s(o_4) = \frac{e^{4.442}}{e^{4.223} + e^{7.869} + e^{4.473} + e^{4.442}} = 0{,}21$$

Sehingga didapatkan nilai bobot nilai probabilitas yang lebih besar yaitu 0,37 yang berarti input citra yang diprediksi terkena penyakit Hawar daun yang ditandakan dengan O_2.

4.4.1 Pemodelan CNN

Pemodelan ini bertujuan untuk membuat model yang akan digunakan pada sistem. Serta contoh implementasi kode pemodelan CNN dapat dilihat pada Kode Program 4.4.

Kode Program 4.4 Pemodelan CNN

```
1) model_name='test'
2) print("Building model with", base_model)
3) model = tf.keras.Sequential([
4)       base_model,
5)       tf.keras.layers.Conv2D(filters=32, padding='same',
6)       kernel_size=3, activation='relu', strides=1),
7)       tf.keras.layers.MaxPool2D(pool_size=2, strides=2),
8)       tf.keras.layers.Dropout(rate=0.5),
9)       tf.keras.layers.Conv2D(filters=64, padding='same',
10)      kernel_size=3, activation='relu', strides=1),
11)      tf.keras.layers.MaxPool2D(pool_size=2, strides=2),
12)      tf.keras.layers.Flatten(),
13)      tf.keras.layers.Dense(5, activation='softmax')
14)     ])
15) model.compile(optimizer=tf.keras.optimizers.Adam(learning_ra
16) te=.001), loss='categorical_crossentropy',
17) metrics='accuracy')
```

Penjelasan Kode Program:

1) `*model_name = 'test'*`:Variabel `*model_name*` menyimpan nama model.
2) `*print("Building model with", base_model)*`: Ini mencetak pesan yang berisi informasi tentang model dasar yang digunakan.
3) `*model = tf.keras.Sequential([ ... ])*`: Membuat objek model menggunakan *Sequential API* dari *Keras*.
4) `*base_model*`: Model dasar (*pre-trained*) yang digunakan sebagai lapisan pertama dalam model.
5) `*tf.keras.layers.Conv2D(...)*`,`*tf.keras.layers.MaxPool2D(...)*`,`*tf.keras.layers.Dropout(...)*`, `*tf.keras.layers.Flatten()*`, `*tf.keras.layers.Dense(...)*`: Ini adalah lapisan-lapisan yang ditambahkan ke model: `*Conv2D*`: Lapisan konvolusi untuk mengekstraksi fitur dari gambar. `*MaxPool2D*`: Lapisan *max pooling* untuk mereduksi dimensi gambar. `*Dropout*`: Lapisan *dropout* untuk mencegah *overfitting*. `*Flatten*`: Lapisan untuk meratakan *output* menjadi vektor satu dimensi. `*Dense*`: Lapisan dense (sepenuhnya

terhubung) untuk klasifikasi, dengan aktivasi *softmax* sebagai lapisan *output*.

6) `*model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=.001), loss='categorical_crossentropy', metrics='accuracy')*`: Mengompilasi model dengan konfigurasi: Pengoptimal Adam dengan laju pembelajaran (*learning rate*) 0.001. Fungsi kerugian *categorical crossentropy*, sesuai dengan masalah klasifikasi multikelas. Metrik akurasi untuk mengevaluasi kinerja model selama pelatihan dan evaluasi.

Arsitektur model pada Kode Program 4.4 akan menghasilkan *output* sesuai dengan format yang tercantum dalam Tabel 4.3.

Tabel 4.3 Model CNN

Layer (type)	***Fungsi***	***Output Shape***	***Param #***
xception (Functional)	Ekstraksi fitur tingkat tinggi (pre-trained)	(*None*, 7, 7, 2048)	20861480
conv2d_10 (Conv2D)	Ekstraksi fitur spasial	(*None*, 7, 7, 32)	589856
max_pooling2d_2 (MaxPooling2D)	Pengurangan dimensi spasial	(*None*, 3, 3, 32)	0
dropout_1 (Dropout)	Pencegahan *overfitting*	(*None*, 3, 3, 32)	0
conv2d_11 (Conv2D)	Ekstraksi fitur spasial	(*None*, 3, 3, 64)	18496
max_pooling2d_3 (MaxPooling2D	Pengurangan dimensi spasial	(*None*, 1, 1, 64)	0
flatten_1 (Flatten)	Konversi data spasial menjadi vektor	(*None*, 64)	0

Layer (type)	***Fungsi***	***Output Shape***	***Param #***
dense_1 (Dense)	Klasifikasi gambar	(*None*, 5)	325
Total params: **21470157 (81.90 MB)**			
Trainable params: **608677 (2.32 MB)**			
Non-trainable params: **20861480 (79.58 MB)**			

Pada Tabel 4.3 menunjukkan model yang digunakan dalam penelitian "Implementasi Pengolahan Citra pada Penyakit Daun Tanaman Padi dengan Metode CNN (Studi Kasus Kabupaten Tuban)". Berikut adalah rumus untuk menghitung *parameter*:

Ketengaran Rumus *parameter:*

a) F untuk ukuran *filter* (kernel),
b) C untuk jumlah *channel*,
c) H untuk tinggi (*height*) gambar,
d) W untuk lebar (*width*) gambar.
e) IC untuk *input channel*
f) OC untuk untuk *output channel*
g) IN unruk *input neurons*
h) ON unruk *output neuron*

Perhitungan *Parameter:*

1. *Xception* (*Functional*) *Layer*:

a) *Output Shape*: (H, W, C) = (7,7,2048)

b) Parameter: $F \times F \times IC \times OC + OC$

$$3 \times 3 \times 2048 \times 2048 + 2048 = 20{,}861{,}840$$

2. *Layer* (conv2d_10):

a) *Output Shape*: (H, W, C) = (7,7,32)

b) Parameter: $F \times F \times IC \times OC + OC$

$$3 \times 3 \times 2048 \times 32 + 32 = 589{,}856$$

3. *MaxPooling2D Layer* (max_pooling2d_2):

a) Tidak ada parameter yang dapat diubah.

4. ***Dropout Layer*** **(dropout_1):**
    a) Tidak ada parameter yang dapat diubah.
5. **Conv2D *Layer* (conv2d_11):**
    a) *Output Shape*: (H, W, C) = (3,3,64)
    b) Parameter: $F \times F \times IC \times OC + OC$

$$3 \times 3 \times 32 \times 64 + 64 = 18{,}496$$

6. **MaxPooling2D *Layer* (max_pooling2d_3):**
    a) Tidak ada parameter yang dapat diubah.
7. ***Flatten Layer*** **(flatten_1):**
    a) Tidak ada parameter yang dapat diubah.
8. ***Dense Layer*** **(dense_1):**
    a) *Output Shape*:(5,)
    b) Parameter: $IN \times ON + ON$

$$64 \times 5 + 5 = 325$$

Total Params**:** $20{,}861{,}840 + 589{,}856 + 18{,}496 + 325 = 21{,}470{,}157$

Trainable Params**:** $589{,}856 + 18{,}496 + 325 = 608{,}677$

Non-trainable params**:** $20{,}861{,}840$

4.1 Pelatihan dan Pengujian Model

4.5.1 Pelatihan Model

A. Pelatihan Model Dataset 80%:20%

Dalam tahap ini, model CNN yang telah dibuat akan menjalani pelatihan dan evaluasi. Proses ini menggunakan *dataset* yang dibagi menjadi 80% untuk pelatihan dan 20% untuk validasi. Pembagian ini didasarkan pada prinsip Pareto yang umum ditemui dalam data mining, dimana 80% performa model didapatkan dari pelatihan terhadap 20% data. Jumlah *epoch* pelatihan, dalam hal ini 25 *epoch*, dipilih berdasarkan eksperimen dan referensi untuk mencapai keseimbangan antara durasi pelatihan dan performa akhir model. *Dataset* validasi yang terpisah ini membantu memantau kemampuan generalisasi model, yaitu kemampuan model untuk mengenali penyakit pada data baru yang belum pernah dilihatnya sebelumnya. Kode program untuk proses pelatihan ini dapat ditemukan pada Kode Program 4.5.

Kode Program 4.5 Pelatihan Model

```
1) epochs =25
2) history=model.fit (x=train_gen, epochs=epochs,
3) validation_data=valid_gen)
```

Penjelasan Kode Program:

1) `*history = model.fit()*`digunakan untuk memulai proses pelatihan model.
2) `*x=train_gen,*`merupakan parameter untuk data pelatihan yang digunakan untuk melatih model. `*train_gen*` adalah generator yang menyediakan batch data pelatihan.
3) `*epochs=epochs,*` merupakan parameter untuk menentukan jumlah *epochs* yang akan digunakan dalam pelatihan. Nilainya diambil dari variabel `*epochs*` yang telah ditentukan sebelumnya.
4) `*validation_data=valid_gen*` merupakan untuk parameter untuk data validasi yang digunakan untuk mengevaluasi kinerja model. `*valid_gen*` adalah *generator* yang menyediakan *batch* data validasi.

Pelatihan model pada Kode Program 4.5 akan menghasilkan hasil pelatihan sesuai dalam Tabel 4.4.

Tabel 4.4 Hasil Pelatihan Model

Epoch	***Time***	***Accuracy***	***loss***	***Val Accuracy***	***Val Loss***
1/25	205 s	0.7136	0.7342	0.4472	0.8514
2/25	203 s	0.8579	0.3708	0.8971	0.3411
3/25	210 s	0.9064	0.2509	0.8971	0.2760
4/25	200 s	0.9357	0.2018	0.8971	0.3182
5/25	201 s	0.9264	0.1854	0.9200	0.2263
6/25	203 s	0.9450	0.1460	0.9371	0.1806
7/25	202 s	0.9543	0.1272	0.9086	0.2274
8/25	198 s	0.9514	0.1285	0.9429	0.1990
9/25	217 s	0.9679	0.0992	0.9314	0.2361
10/25	198 s	0.9550	0.1032	0.9657	0.1811
11/25	201 s	0.9657	0.1046	0.9486	0.1700
12/25	208 s	0.9636	0.0957	0.9714	0.1392

Epoch	*Time*	*Accuracy*	*loss*	*Val Accuracy*	*Val Loss*
13/25	205 s	0.9564	0.1062	0.9543	0.1838
14/25	215 s	0.9636	0.0914	0.9429	0.2101
15/25	205 s	0.9650	0.0995	0.9371	0.2545
16/25	208 s	0.9650	0.1005	0.9486	0.1483
17/25	210 s	0.9671	0.0835	0.9429	0.1779
18/25	200 s	0.9707	0.0885	0.9371	0.2131
19/25	209 s	0.9650	0.0877	0.9543	0.1605
20/25	201 s	0.9679	0.0779	0.9429	0.1743
21/25	204 s	0.9736	0.0831	0.9200	0.3267
22/25	212 s	0.9693	0.1018	0.9486	0.1226
23/25	206 s	0.9757	0.0749	0.9486	0.1460
24/25	212 s	0.9721	0.0767	0.9371	0.1602
25/25	207 s	0.9821	0.0572	0.9371	0.1633

Keterangan:

1) *Epoch* : Satu putaran lengkap saat model belajar dari semua data yang ada.
2) *Time* : Waktu yang dibutuhkan untuk melatih model dalam satu putaran.
3) *Accuracy* : Seberapa banyak prediksi yang benar dibandingkan dengan total prediksi yang dilakukan.
4) *Loss* : Seberapa besar kesalahan model dalam memprediksi.
5) *Val Accuracy* : Seberapa baik model dapat memprediksi data baru yang belum pernah dilihat.
6) *Val Loss* : Seberapa buruk kesalahan model saat memprediksi data baru yang belum pernah dilihat.

Hasil pelatihan model pada Tabel 4.4 akan menghasilkan *output* visualisai grafik sesuai dengan Gambar 4.18.

Gambar 4.18 Visualisasi Grafik Pelatihan Model

Hasil visualisasi pelatihan (*training*) pada Gambar 4.4 dijelaskan sebagai berikut:

1) Diagram x menunjukkan jumlah *epochs* yang diterapkan pada model.
2) Diagram y menunjukkan nilai *loss* atau akurasi yang terjadi selama setiap *epochs* berlangsung.
3) Garis merah melambangkan kurva hasil pelatihan model. Dalam diagram akurasi, semakin tinggi nilai akurasi, semakin baik kualitas model tersebut. Sebaliknya, pada diagram *training loss*, semakin rendah nilai kehilangan, semakin baik kualitas model tersebut.
4) Garis hijau menunjukkan kurva hasil validasi model. Pada diagram tersebut, jika perbedaan nilai antara kurva pelatihan dan validasi tidak signifikan, dan nilai berubah secara stabil, maka model tersebut dapat dikatakan memiliki kualitas yang baik, tidak mengalami overfitting atau underfitting. Model yang baik cenderung memiliki kehilangan yang rendah dan akurasi yang tinggi (Allaam dan Wibowo, 2021). Dari grafik terlihat bahwa model yang dihasilkan cukup baik karena memiliki kehilangan yang rendah dan akurasi yang tinggi, yaitu mencapai 97,14%.
5) Titik biru pada diagram atas menunjukkan *epochs* terbaik untuk model. Pada titik biru ini menunjukkan epochs yang mana memiliki kerugian pelatihan terendah serta memiliki epoch yang mana memiliki akurasi pelatihan tertinggi.

B. Pelatihan Model Dataset 70%:30%

Dalam tahap ini, model CNN yang telah dibuat akan menjalani pelatihan dan evaluasi. Proses ini menggunakan *dataset* yang dibagi menjadi 70% untuk pelatihan dan 30% untuk validasi serta *testing*. Pembagian ini didasarkan dengan menggunakan 70% untuk pelatihan dan 30% untuk pengujian serta validasi, karena memiliki jumlah data yang cukup besar untuk melatih model dengan baik, sambil tetap menyisakan sejumlah data yang signifikan untuk menguji kinerja model. Jumlah *epochs* pelatihan, dalam hal ini 25 *epochs*, dipilih berdasarkan eksperimen dan referensi untuk mencapai keseimbangan antara durasi pelatihan dan performa akhir model. Dataset validasi yang terpisah ini membantu memantau kemampuan generalisasi model, yaitu kemampuan model untuk mengenali penyakit pada data baru yang belum pernah dilihatnya sebelumnya. Kode program untuk proses pelatihan ini dapat ditemukan pada Kode Program 4.6.

Kode Program 4.6 Pelatihan Model dengan data 70%:30%

```
1) epochs =25
2) history=model.fit (x=train_gen, epochs=epochs,
3) validation_data=valid_gen)
```

Penjelasan Kode Program:

1) `*history = model.fit()*`digunakan untuk memulai proses pelatihan model.
2) `*x=train_gen*,`merupakan parameter untuk data pelatihan yang digunakan untuk melatih model. `*train_gen*` adalah generator yang menyediakan batch data pelatihan.
3) `*epochs=epochs*,` merupakan parameter untuk menentukan jumlah *epochs* yang akan digunakan dalam pelatihan. Nilainya diambil dari variabel `*epochs*` yang telah ditentukan sebelumnya.
4) `*validation_data=valid_gen*` merupakan untuk parameter untuk data validasi yang digunakan untuk mengevaluasi kinerja model. `*valid_gen*` adalah *generator* yang menyediakan *batch* data validasi.

Pelatihan model pada Kode Program 4.6 akan menghasilkan hasil pelatihan sesuai dalam Tabel 4.5.

Tabel 4.5 Hasil Pelatihan Model

Epoch	*Time*	*Accuracy*	*loss*	*Val Accuracy*	*Val Loss*
1/25	214s	0.6661	0.7998	0.7857	0.5424
2/25	200s	0.8359	0.4152	0.9000	0.3262
3/25	203s	0.8645	0.3257	0.8971	0.2926
4/25	197s	0.9004	0.2559	0.9029	0.3251
5/25	198s	0.9167	0.2004	0.9143	0.2375
6/25	210s	0.9249	0.1907	0.8943	0.2955
7/25	213s	0.9420	0.1691	0.9400	0.1769
8/25	201s	0.9388	0.1600	0.9257	0.2632
9/25	213s	0.9380	0.1744	0.9257	0.1845
10/25	208s	0.9535	0.1286	0.9057	0.2482
11/25	202s	0.9469	0.1291	0.9314	0.2174
12/25	198s	0.9690	0.1003	0.9400	0.1736
13/25	211s	0.9478	0.1368	0.9343	0.2144
14/25	208s	0.9559	0.1194	0.9257	0.1970
15/25	205s	0.9698	0.0830	0.9429	0.2193
16/25	202s	0.9600	0.1037	0.9314	0.2662
17/25	208s	0.9739	0.0721	0.9371	0.2300
18/25	206s	0.9682	0.0752	0.9057	0.4297
19/25	196s	0.9641	0.0936	0.9314	0.2190
20/25	213s	0.9649	0.0913	0.9200	0.2451
21/25	204 s	0.9698	0.0863	0.9543	0.1848
22/25	212 s	0.9739	0.0735	0.9371	0.2226
23/25	206 s	0.9714	0.0813	0.9343	0.2216
24/25	212 s	0.9755	0.0644	0.9400	0.2018
25/25	207 s	0.9853	0.0470	0.9486	0.1957

Keterangan:

1) *Epoch* : Satu putaran lengkap saat model belajar dari semua data yang ada.
2) *Time* : Waktu yang dibutuhkan untuk melatih model dalam satu putaran.
3) *Accuracy* : Seberapa banyak prediksi yang benar dibandingkan dengan total prediksi yang dilakukan.
4) *Loss* : Seberapa besar kesalahan model dalam memprediksi.

5) *Val Accuracy* : Seberapa baik model dapat memprediksi data baru yang belum pernah dilihat.
6) *Val Loss* : Seberapa buruk kesalahan model saat memprediksi data baru yang belum pernah dilihat.

Hasil pelatihan model pada Tabel 4.5 akan menghasilkan *output* visualisai grafik sesuai dengan Gambar 4.19.

Gambar 4.19 Visualisasi Grafik Pelatihan Model

Hasil visualisasi pelatihan (*training*) pada Gambar 4.5 dijelaskan sebagai berikut:

1) Diagram x menunjukkan jumlah *epochs* yang diterapkan pada model.
2) Diagram y menunjukkan nilai *loss* atau akurasi yang terjadi selama setiap *epochs* berlangsung.
3) Garis merah melambangkan kurva hasil pelatihan model. Dalam diagram akurasi, semakin tinggi nilai akurasi, semakin baik kualitas model tersebut. Sebaliknya, pada diagram *training loss*, semakin rendah nilai kehilangan, semakin baik kualitas model tersebut.
4) Garis hijau menunjukkan kurva hasil validasi model. Pada diagram tersebut, jika perbedaan nilai antara kurva pelatihan dan validasi tidak signifikan, dan nilai berubah secara stabil, maka model tersebut dapat dikatakan memiliki kualitas yang baik, tidak mengalami overfitting atau underfitting. Model yang baik cenderung memiliki kehilangan yang rendah dan akurasi yang tinggi (Allaam dan Wibowo, 2021). Dari grafik terlihat bahwa model yang

dihasilkan cukup baik karena memiliki kehilangan yang rendah dan akurasi yang tinggi, yaitu mencapai 98,53%.

5) Titik biru pada diagram atas menunjukkan *epochs* terbaik untuk model. Pada titik biru ini menunjukkan *epochs* yang mana memiliki kerugian pelatihan terendah serta memiliki *epoch* yang mana memiliki akurasi pelatihan tertinggi.

4.5.2 Membandingkan *Scenario* pembagian *Dataset*

Pada tahap ini perbandingan skenario pembagian dataset merupakan langkah penting dalam membangun model machine learning yang akan digunakan dalam tahap *deployment website*. Tahap ini melibatkan perbandingan dua skenario pembagian dataset, yaitu 80%:20% dan 70%:30%. Proses ini dilakukan dengan menggunakan nilai epoch sebanyak 25. Serta hasil perbandingan kedua dataset dapat dilihat Tabel 4.6.

Tabel 4.6 Perbandingan Hasil Pelatihan

Skenario *Dataset*	*Train*		*Vall*		*Test*	
	Accuracy	*Loss*	*Accuracy*	*Loss*	*Accuracy*	*Loss*
80%:20%	0.9821	0.0572	0.9371	0.1633	0.9657	0.1407
70%:30%	0.9853	0.0470	0.9486	0.1957	0.9657	0.0968

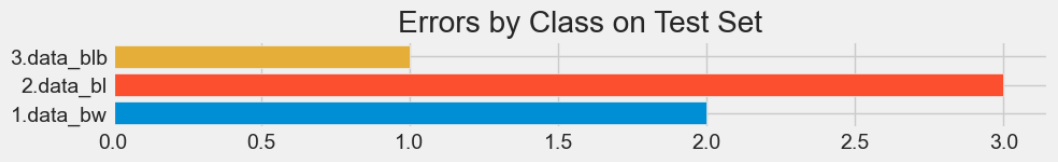
Keterangan:

1) *Train*: Persentase data yang digunakan untuk melatih model.
2) *Vall*: Persentase data yang digunakan untuk memvalidasi model selama pelatihan.
3) *Test*: Persentase data yang digunakan untuk menguji performa model setelah pelatihan.
4) Akurasi: Metrik yang mengukur seberapa baik model memprediksi hasil yang benar. Nilai akurasi yang lebih tinggi menunjukkan kinerja model yang lebih baik.
5) *Loss*: Metrik yang mengukur seberapa besar kesalahan model dalam memprediksi hasil yang benar. Nilai *loss* yang lebih rendah menunjukkan kinerja model yang lebih baik.

Berdasarkan hasil percobaan, model yang dilatih dengan skenario dataset 70%:30% menunjukkan performa yang lebih baik dibandingkan dengan model yang dilatih dengan skenario dataset 80%:20%. Model 70%:30% memiliki akurasi yang lebih tinggi pada data validasi dan data tes, menunjukkan kemampuannya yang lebih baik dalam memprediksi hasil yang benar. Selain itu, model 70%:30% juga memiliki loss yang lebih rendah pada kedua jenis data, menandakan kesalahan prediksi yang lebih kecil. Kesimpulannya, skenario dataset 70%:30% terbukti lebih optimal untuk model ini dalam menghasilkan prediksi yang lebih akurat dan andal.

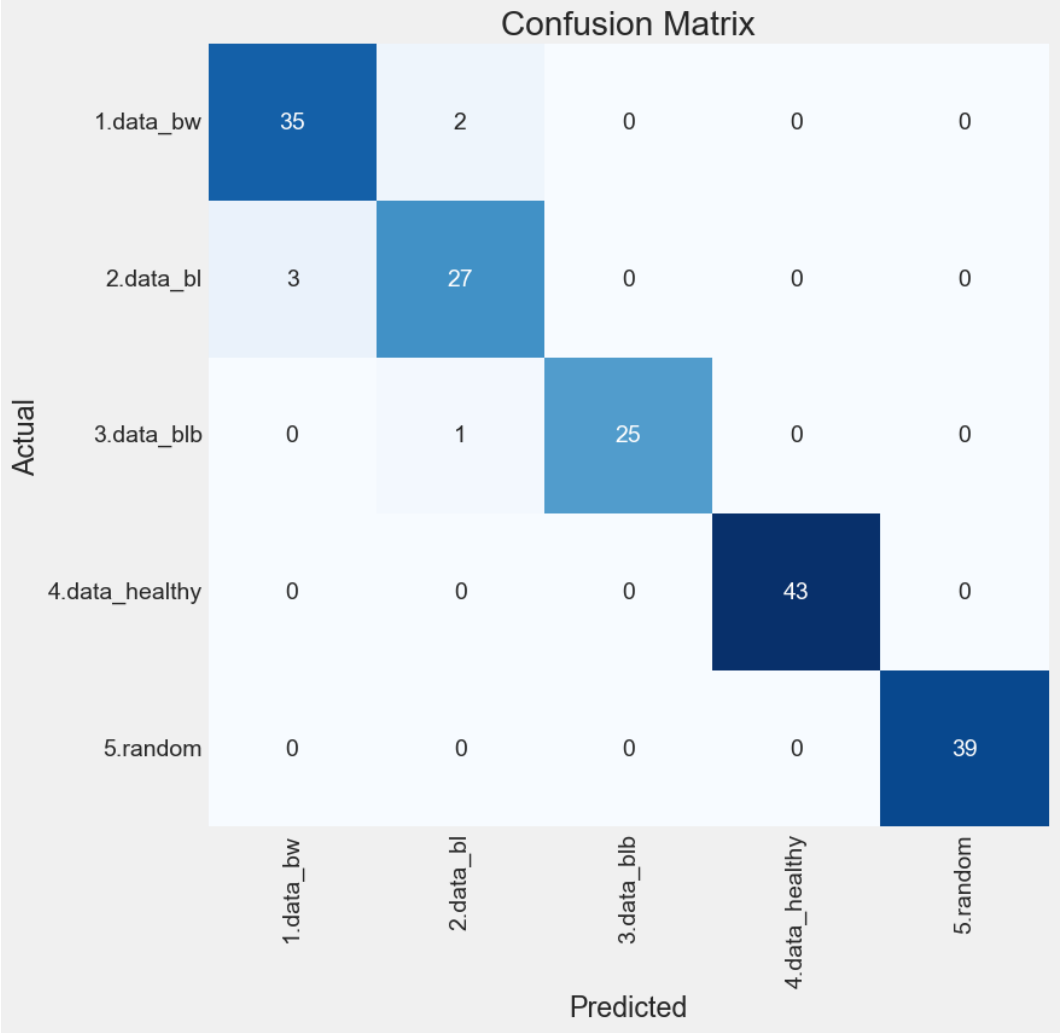
4.5.3 Pengujian Model

Setelah melatih model *Convolutional Neural Network* (CNN), langkah selanjutnya adalah mengevaluasi kinerjanya menggunakan *confusion matrix*. *Confusion matrix* memberikan gambaran detail tentang kemampuan model dalam mengklasifikasikan berbagai kelas pada dataset uji. Pada Gambar 4.20 menunjukkan hasil *error test* pada dataset yang digunakan.



Gambar 4.20 *error test*

Gambar 4.6 merupakan hasil error pada data test yang telah diujuikan dengan model yang telah dibuat sebelumnya. Pada diagram batang diatas terdapat beberapa error ata kesalahan membaca karena tidak sesuai dengan kelas sebelumnya. Pada diagram diatas menunjukkan bahwa data_bw atau (*Brown Spot)* menunjukkan kesalahan dalam proses klasifikasi data sebanyak 2, data_bl atau (*Pyricularia oryzae Cav)* menunjukkan kesalahan dalam proses klasifikasi data sebanyak 3, dan data_blb atau (*Xanthomonas oryzae pv. Oryzicola)* menunjukkan kesalahan dalam proses klasifikasi data sebanyak 1. Untuk hasil dari *confusion matrix* ditunjukkan oleh Gambar 4.21.



Gambar 4.21 Hasil dari *confusion matrix*

Keterangan:

1) Data_bw merupakan data penyakit *Brown Spot* atau bercak coklat.
2) Data_bl merupakan data penyakit *Pyricularia oryzae Cav* atau penyakit blast pada daun.
3) Data_blb merupakan data penyakit *Xanthomonas oryzae pv. oryzicola* atau penyakit leaf blight pada daun.
4) Data_*healthy* merupakan data tanaman padi sehat.
5) Data_*random* merupakan data gambar acak selain daun tanaman padi.

Berikut merupakan hasil dari *confusion matrix* sesuai dengan Gambar 4.7:

Tabel 4.7 Hasil *Confusion Matrix*

	Precision	*Recall*	*F1-Score*	*Support*
Data_bw	0.92	0.95	0.93	37
Data_bl	0.90	0.90	0.90	30
Data_blb	1.00	0.96	0.98	26
Data_*healthy*	1.00	1.00	1.00	43
Data_*random*	1.00	1.00	1.00	39

Keterangan:

1) ***Precision***: Mengukur ketepatan model dalam mengklasifikasikan sampel positif. Nilai yang lebih tinggi menunjukkan ketepatan yang lebih baik.
2) ***Recall***: Mengukur kemampuan model dalam menemukan semua sampel positif yang sebenarnya. Nilai yang lebih tinggi menunjukkan kemampuan yang lebih baik dalam menemukan semua sampel positif.
3) ***F1-score:*** Merupakan gabungan dari precision dan recall, memberikan gambaran keseimbangan antara kedua metrik tersebut. Nilai yang lebih tinggi menunjukkan keseimbangan yang lebih baik dan performa model yang lebih optimal.
4) ***Support***: Menunjukkan jumlah sampel pada setiap kelas.
5) **Data_bw**: Kelas gambar yang data penyakit *Brown Spot* atau bercak coklat.
6) **Data_bl**: Kelas gambar yang *Pyricularia oryzae Cav* atau penyakit blast pada daun.
7) **Data_blb**: Kelas gambar yang menunjukkan *Xanthomonas oryzae pv. oryzicola* atau penyakit leaf blight pada daun.
8) **Data_*healthy***: Kelas gambar yang menunjukkan data tanaman padi sehat.
9) **Data_*random***: Kelas gambar acak selain daun tanaman padi.

Berikut merupakan perhitungan *confusion matrix* sesuai dengan Tabel 4.7:

1. BW (*Brown Spot*)

a) *Precision*

$$Precision = \left(\frac{35}{35+3}\right) \times 100\% = 92\ \%$$

b) *Recall*

$$Recall = \left(\frac{35}{35+2}\right) \times 100\% = 95\%$$

c) *F1-score*

$$F1\ Score = 2\ \times \frac{0.92 \times 0.95}{0.92 + 0.95} \times 100\% = 93\%$$

2. PO (*Pyricularia Oryzae Cav*)

a) *Precision*

$$Precision = \left(\frac{27}{27+3}\right) \times 100\% = 90\ \%$$

b) *Recall*

$$Recall = \left(\frac{27}{27+3}\right) \times 100\% = 90\%$$

c) F1-score

$$F1\ Score = 2\ \times \frac{0.90\ \times 0.90}{0.90 + 0.90} \times 100\% = 90\%$$

3. X.O (*Xanthomonas Oryzae Pv. Oryzicola*)

a) Precision

$$Precision = \left(\frac{25}{25+0}\right) \times 100\% = 100\ \%$$

b) Recall

$$Recall = \left(\frac{25}{25+1}\right) \times 100\% = 96\%$$

c) F1-score

$$F1\ Score = 2\ \times \frac{1\ \times 0.96}{1 + 0.96} \times 100\% = 98\%$$

4. Healthy

a) *Precision*

$$Precision = \left(\frac{43}{43+0}\right) \times 100\% = 100\ \%$$

b) *Recall*

$$Recall = \left(\frac{43}{43+0}\right) \times 100\% = 100\%$$

c) *F1-score*

$$F1\ Score = 2\ \times \frac{1\ \times 1}{1+1}\ \times 100\% = 100\%$$

5. *Random*

a) *Precision*

$$Precision = \left(\frac{39}{39+0}\right) \times 100\% = 100\ \%$$

b) *Recall*

$$Recall = \left(\frac{39}{39+0}\right) \times 100\% = 100\%$$

c) *F1-score*

$$F1\ Score = 2\ \times \frac{1\ \times 1}{1+1}\ \times 100\% = 100\%$$

Berdasarkan Tabel 4.7,model menunjukkan hasil klasifikasi yang baik. Dari 175 data citra. Hanya pada 6 citra yang memiliki kesalahan dalam pengklasifikasian penyakit daun padi. Perhitungan akurasi dari keseluruhan matriks menggunakan persamaan 4.4.

$$Akurasi: \frac{Jumlah\ data\ benar}{jumlah\ data\ total} \times 100\% = \frac{169}{175}\ \times 100\% = 96{,}57\% \text{ -------- 4.4}$$

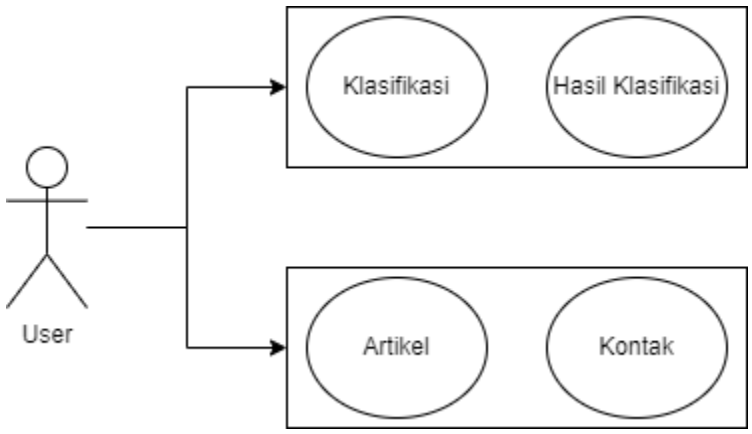
4.2 Implementasi *Website*

Website SIKAPADI merupakan *website* untuk klasifikasi dan deteksi penyakit pada tanaman padi. Melalui antarmuka yang *user-friendly*, pengguna dapat mengunggah gambar tanaman padi yang potensial terkena penyakit, dan sistem ini memberikan hasil klasifikasi dengan tingkat akurasi tinggi serta rekomendasi pengobatan yang sesuai.

4.6.1 *Use Case Diagram Website* SIKAPADI

Use case adalah desain yang dibuat untuk mengidentifikasi fungsi-fungsi dari sebuah aplikasi yang dapat digunakan dan diakses oleh pengguna. Fungsi-fungsi ini kemudian akan diuraikan menjadi fitur-fitur yang ada dalam aplikasi

tersebut. Gambaran visual dari *use case* untuk aplikasi yang digunakan dalam penelitian ini dapat dilihat pada Gambar 4.22 di bawah ini.



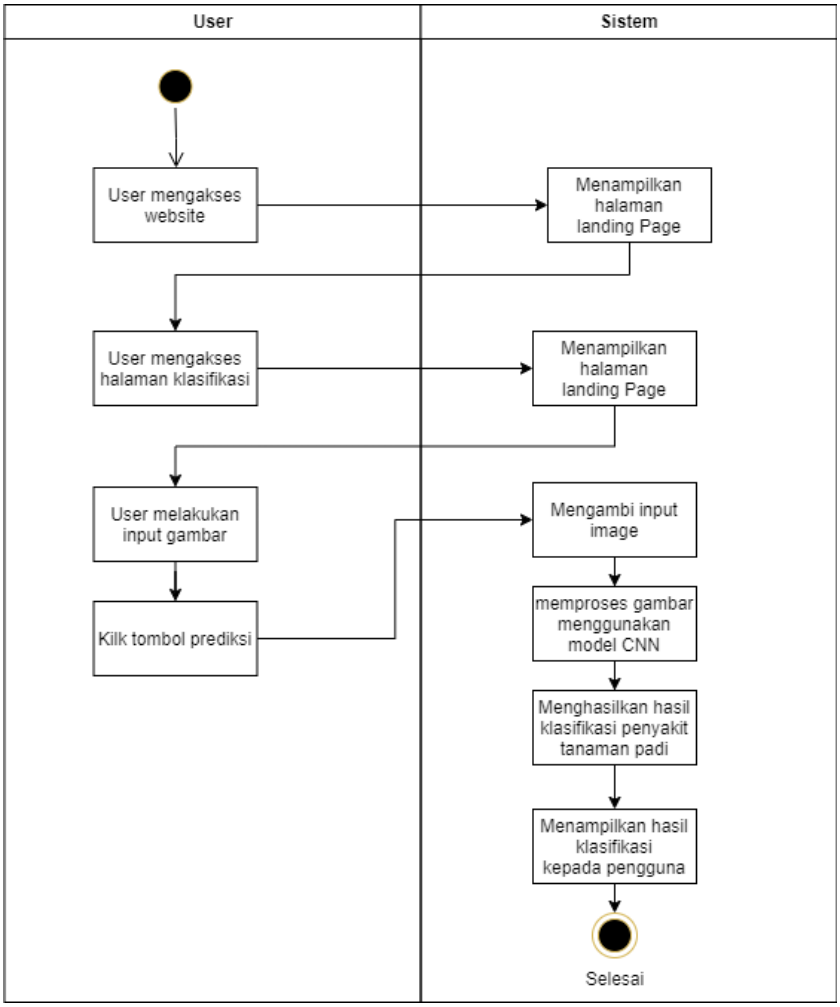
Gambar 4.22 UCD SIKAPADI

Pada Gambar 4.22 merupakan *use case diagram website* SIKAPADI yang menunjukkan cara pengguna berinteraksi dengan *website*. Diagram ini terdiri dari tiga elemen utama, yaitu:

1. Aktor, yang mewakili pengguna atau pihak lain yang berinteraksi dengan sistem. Dalam hal ini, aktornya adalah *User*.
2. *Use case*, yang mewakili fungsionalitas yang disediakan oleh sistem. Dalam hal ini, terdapat tiga *use case*, yaitu:
   a) Klasifikasi, yang memungkinkan pengguna untuk mengupload atau memfoto penyakit tanaman padi .
   b) Artikel, yang menyediakan informasi tentang tanaman padi.
   c) Kontak, yang memungkinkan pengguna untuk berkonsultasi dengan *Admin* SIKAPADI.
3. Asosiasi, yang menunjukkan hubungan antara aktor dan *use case*. Dalam hal ini, terdapat dua asosiasi, yaitu:
   1) Asosiasi antara *User* dan Klasifikasi, yang menunjukkan bahwa *User* dapat menggunakan halaman klasifikasi untuk mendeteksi penyakit daun tanaman padi.
   2) Asosiasi antara *User* dan Artikel, yang menunjukkan bahwa *User* dapat mengakses informasi yang memuat tentang tanaman padi.

3) Asosiasi antara *User* dan Kontak, yang menunjukkan bahwa *User* dapat berkonsultasi dengan *Admin* SIKAPADI secara langsung melalui *whatsapp*.

4.6.2 *Activity Diagram Website* SIKAPADI



Gambar 4.23 *Activity Diagram*

Pada Gambar 4.23 menunjukkan proses pembuatan halaman arahan untuk sistem klasifikasi penyakit padi. Pengguna memulai dengan mengakses situs web dan kemudian ditampilkan halaman arahan. Pengguna kemudian dapat mengakses halaman klasifikasi di mana mereka dapat memasukkan gambar tanaman padi. Sistem kemudian memproses gambar menggunakan model CNN dan menghasilkan

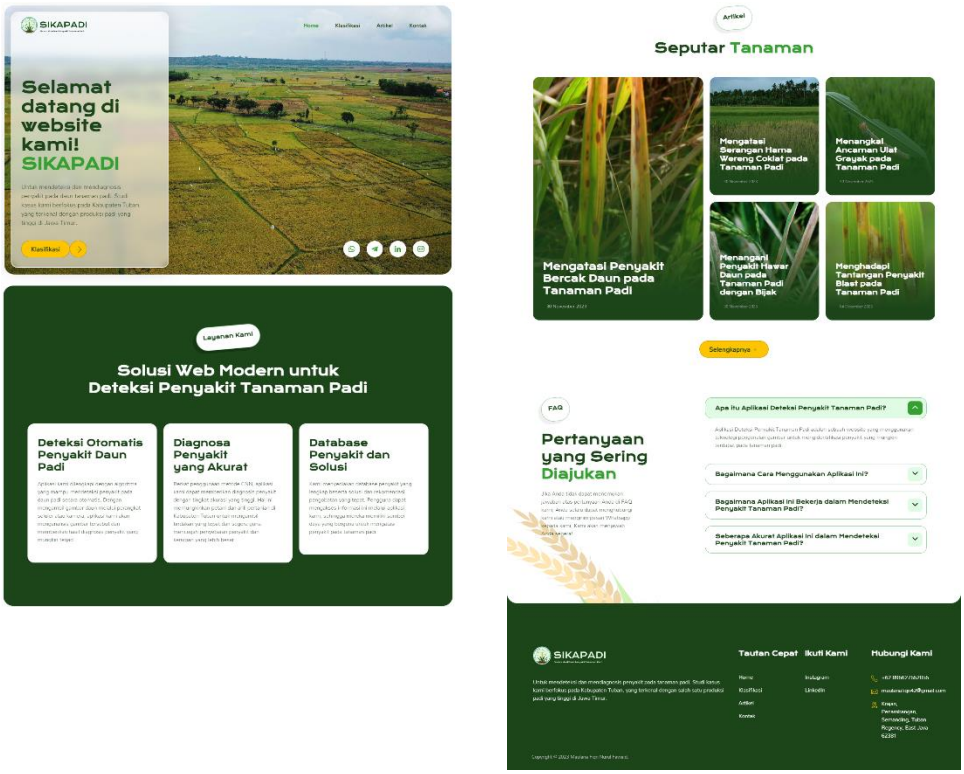
klasifikasi penyakit. Hasil klasifikasi kemudian ditampilkan ke pengguna.Berikut Penjelasan lebih rinci untuk setiap langkah:

1. Langkah 1: Pengguna mengakses situs web dengan memasukkan alamat situs web ke *browser* web.
2. Langkah 2: Sistem menampilkan halaman arahan yang berisi informasi dasar tentang sistem klasifikasi penyakit padi.
3. Langkah 3: Pengguna mengklik tautan ke halaman klasifikasi.
4. Langkah 4: Pengguna mengunggah gambar tanaman padi ke sistem.
5. Langkah 5: Sistem menggunakan model CNN untuk memproses gambar dan menghasilkan klasifikasi penyakit.
6. Langkah 6: Sistem menampilkan klasifikasi penyakit kepada pengguna.

4.6.3 Halaman *User*

A. Halaman *Home*

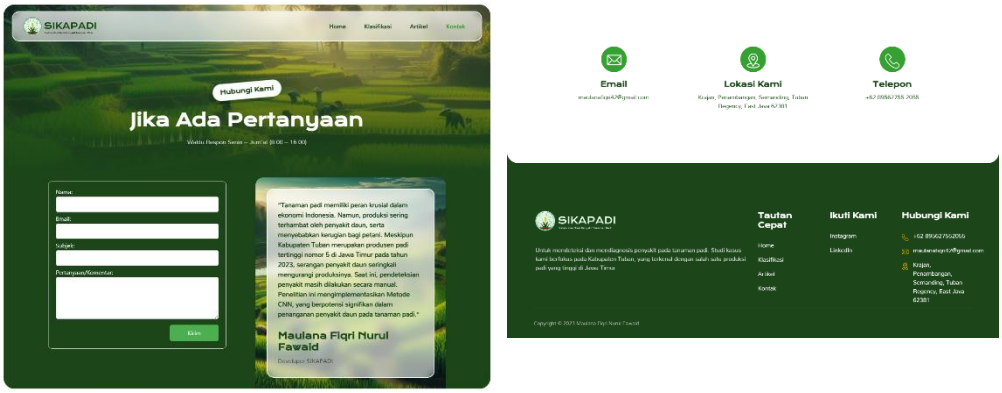
Halaman *Home* merupakan tampilan awal pengguna dalam aplikasi. Di sini, pengguna disuguhkan berbagai fitur dan gambaran fungsionalitas aplikasi seperti pengetahuan tentang penyakit tanaman padi. Informasi yang disajikan membantu pengguna memahami dan mengelola kondisi tanaman padi mereka. Serta, hasil impementasi desain *interface* dapat dilihat pada Gambar 4.24.



Gambar 4.24 Halaman *Home*

B. Halaman Kontak

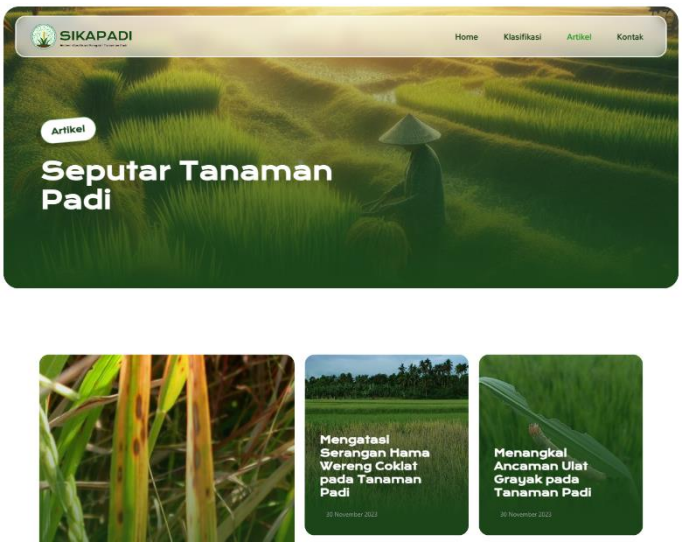
Halaman Kontak merupakan tampilan untuk komunikasi antara pengguna dan admin aplikasi SIKAPADI. Pengguna dapat dengan mudah mengajukan pertanyaan, mendapatkan informasi, dan bantuan melalui formulir yang disediakan. Formulir ini terhubung langsung dengan WhatsApp yang memungkinkan respon cepat dan efisien. Serta, hasil impementasi desain *interface* dapat dilihat pada Gambar 4.25.



Gambar 4.25 Halaman Kontak

C. Halaman Artikel

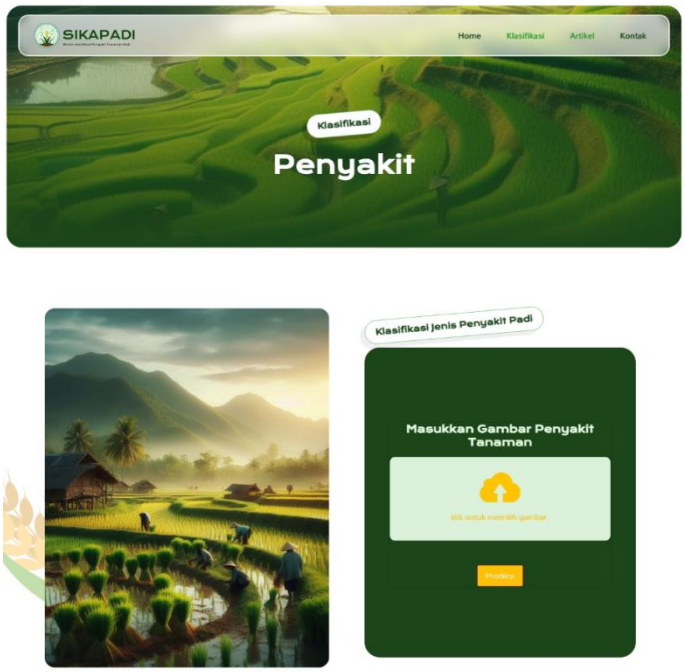
Halaman Artikel adalah tampilan informatif bagi pengguna yang ingin memahami dan mengatasi penyakit tanaman padi. Halaman ini menyediakan informasi tentang berbagai aspek penyakit, seperti deskripsi, gejala, rekomendasi penanganan, dan tindakan pencegahan. Serta, hasil impementasi desain *interface* dapat dilihat pada Gambar 4.26.



Gambar 4.26 Halaman Artikel

D. Halaman Klasifikasi

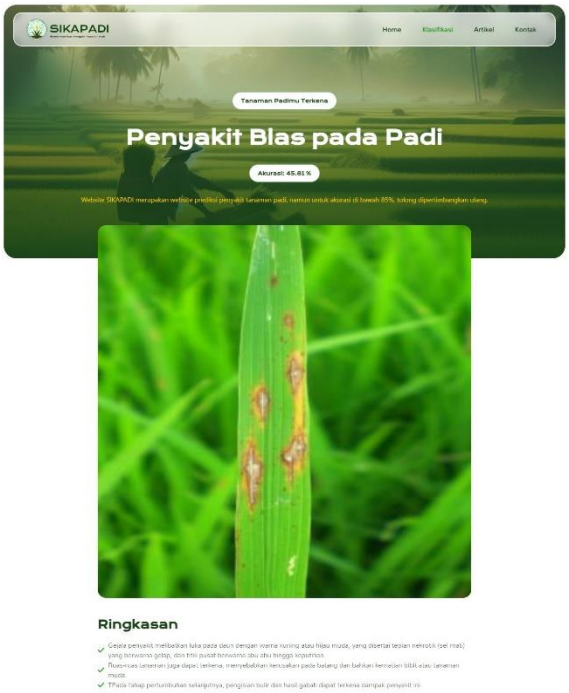
Halaman klasifikasi merupakan tampilan yang memungkinkan pengguna untuk melakukan input data atau mengambil foto daun tanaman padi yang ingin dideteksi. Pada halaman klasifikasi ini, pengguna dieberikan keleluasaan dalam memberikan data input. Dengan mengizinkan pengguna untuk memfoto atau mengunggah gambar, aplikasi ini memperluas ruang lingkup analisis, memungkinkan identifikasi lebih akurat terhadap kondisi tanaman padi. Serta, hasil impementasi desain *interface* dapat dilihat pada Gambar 4.27.



Gambar 4.27 Halaman Klasifikasi

E. Halaman *Output*

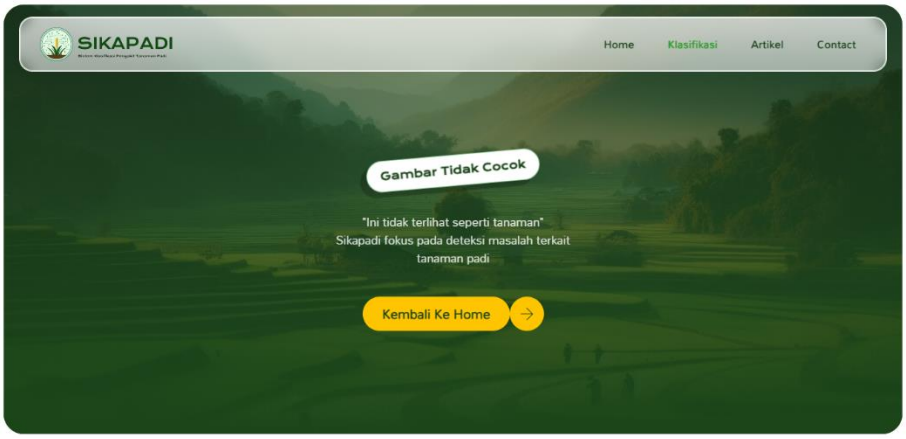
Halaman *Output* menampilkan tampilan yang memungkinkan pengguna untuk melihat hasil klasifikasi gambar yang telah dimasukkan. Pada halaman ini, terdapat informasi tentang nama penyakit, tingkat akurasi klasifikasi, gambar yang telah diunggah oleh pengguna, ringkasan penyakit, gejala yang mungkin muncul, dan cara menanganinya. Serta, hasil impementasi desain *interface* dapat dilihat pada Gambar 4.28.



Gambar 4.28 Halaman *Output*

F. Halaman *Output* Pada *Random Object*

Halaman *Output* menampilkan tampilan yang memungkinkan pengguna untuk melihat hasil klasifikasi gambar data objek acak yang telah dimasukkan. Pada halaman ini, terdapat informasi yang menyatakan bahwa gambar tersebut bukan merupakan tanaman padi. Fungsi utama dari halaman ini adalah memberikan gambaran menyeluruh kepada pengguna mengenai hasil klasifikasi data acak. Namun, pada deteksi objek acak ini, masih terdapat beberapa data yang mendapatkan prediksi yang tidak tepat, hal ini disebabkan karena model ini memiliki tingkat kekuatan prediktif yang tinggi namun tidak dapat mengidentifikasi kapan prediksi dari model tersebut melakukan kesalahan (DeVries dan Taylor, 2018). Serta, hasil impementasi desain *interface* dapat dilihat pada Gambar 4.29.



Gambar 4.29 Halaman *Output* Pada *Random Object*

4.3 Pengujian Aplikasi dengan Pakar

Pengujian aplikasi di website SIKAPADI adalah proses kolaboratif antara teknologi deteksi penyakit pada tanaman padi dan pengetahuan ahli pertanian. Di sini, seorang ahli pertanian melakukan serangkaian tes dan analisis untuk memeriksa keakuratan aplikasi dalam mengenali berbagai penyakit daun pada tanaman padi. Ahli pertanian yang melakukan pengujian terhadap *website* SIKAPADI bernama Bapak Achmad Fadlori, SP yaitu seorang Kepala Koordinator Wilayah Kerja TPH (Tanaman Pangan dan Hortikultura) Proteksi Bojonegoro yang bertugas di 3 wilayah Jawa Timur, yaitu Bojonegoro, Tuban dan Lamongan.

Beliau selaku Kepala Koordinator TPH melakukan uji coba yang mensimulasikan kondisi di lapangan dan memvalidasi hasil deteksi dengan data yang telah mereka verifikasi sebelumnya. Melalui kerja sama ini, SIKAPADI dapat diperbaiki dan ditingkatkan kehandalannya sehingga bisa lebih efektif membantu para petani dalam merawat tanaman padi mereka. Hasil dari pengujian bersama pakar Tanaman Pangan dan Hortikultura dapat dilihat pada Gambar 4.30 hingga Gambar 4.35.

Hal: Permohonan Validasi Akurasi Aplikasi Kepada Pakar

Kepada Yth.

Bapak. ACHMAD FADLORI, SP

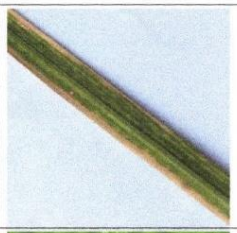
Di tempat

Dengan Hormat,

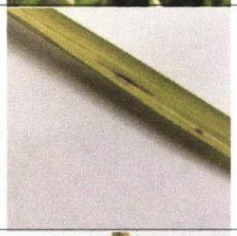
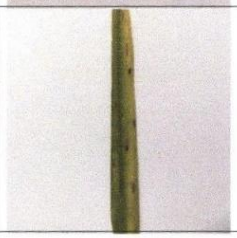
Dalam rangka pengujian validasi akurasi aplikasi Tugas Akhir Skripsi dengan Judul "Implementasi *Image Processing* Pada Penyakit Daun Tanaman Padi Dengan Metode CNN (Studi Kasus Kabupaten Tuban)", Saya memohon kesediaan bapak untuk memvalidasi jenis penyakit tanaman padi guna untuk keperluan penelitan tersebut.

Atas perhatian dan bantuan bapak saya ucapkan terimakasih

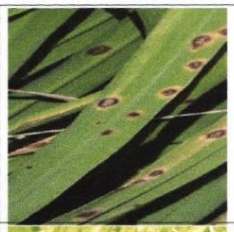
Validasi Jenis Penyakit tanaman padi pada Aplikasi SIKAPADI Serta Pakar

No	Gambar	Jenis Penyakit	Klasifikasi Pakar		Klasifikasi Aplikasi		Kesimpulan
			Ya	Tidak	Ya	Tidak	
1		Blast	V		V		
2		Blast	V		V		

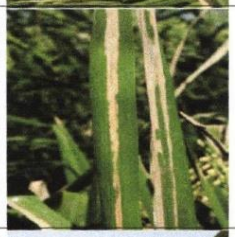
Gambar 4.30 Pengujian dengan Pakar

3		Blast	✓		✓		
4		Blast	✓		✓		
5		Blast	✓		✓		
6		Brown Spot	✓		✓		

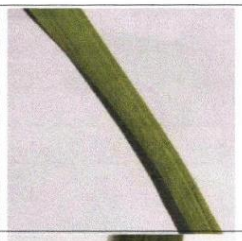
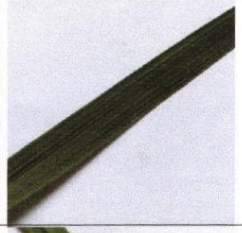
Gambar 4.31 Pengujian dengan Pakar

7		Brown Spot	√		√		
8		Brown Spot	√		√		
9		Brown Spot	√		√		
10		Brown Spot	√		√		
11		Hawar Daun (BLB)	√		√		

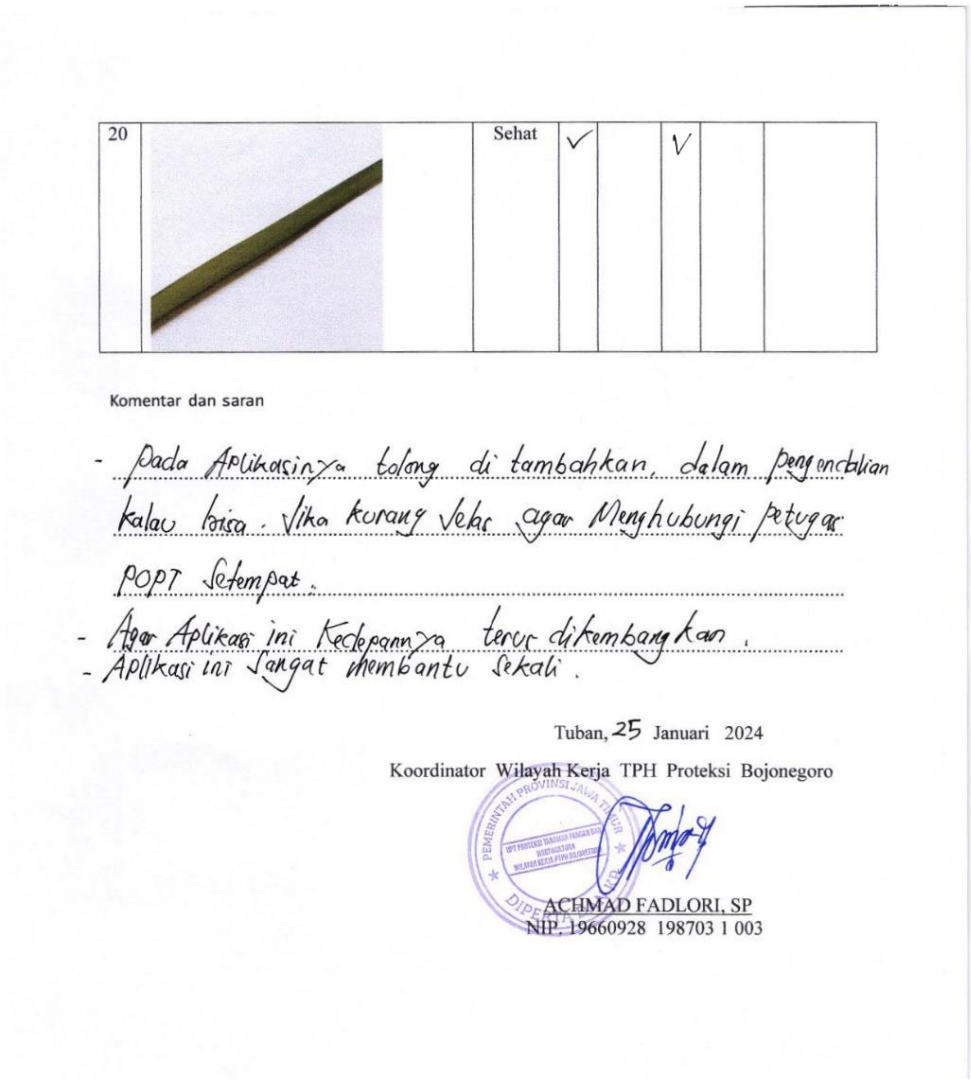
Gambar 4.32 Pengujian dengan Pakar

12		Hawar Daun (BLB)	√		√		
13		Hawar Daun (BLB)	√		√		
14		Hawar Daun (BLB)	√		√		
15		Hawar Daun (BLB)	√		√		

Gambar 4.33 Pengujian dengan Pakar

16		Sehat	V		V		
17		Sehat	V		V		
18		Sehat	V		V		
19		Sehat	V		V		

Gambar 4.34 Pengujian dengan Pakar



20		Sehat	✓		V		

Komentar dan saran

- Pada Aplikasinya tolong di tambahkan, dalam pengambilan kalau bisa. Jika kurang jelas agar Menghubungi petugas POPT Setempat.
- Agar Aplikasi ini Kedepannya terus dikembangkan.
- Aplikasi ini Sangat membantu Sekali.

Tuban, 25 Januari 2024

Koordinator Wilayah Kerja TPH Proteksi Bojonegoro

ACHMAD FADLORI, SP
NIP. 19660928 198703 1 003

Gambar 4.35 Pengujian dengan Pakar

Pakar juga memberikan rekomendasi terhadap aplikasi yang telah dikembangkan, menyarankan untuk menambahkan fitur tertentu dalam pengambilan data jika terdapat kekurangan atau ketidakjelasan dapat berkomunikasi dengan petugas POPT setempat jika diperlukan klarifikasi. Upaya ini diharapkan dapat meningkatkan kualitas dan efektivitas aplikasi, mendukung pengembangan berkelanjutan, dan memastikan bahwa aplikasi dapat terus relevan dengan kebutuhan penggunanya. Secara umum, para pakar mengakui bahwa aplikasi ini memiliki kontribusi yang sangat penting dalam membantu penanganan

penyakit tanaman padi. Berikut akan ditunjukkan tentang perhitungan akurasi aplikasi dengan pakar :

$$\frac{20}{20} \times 100\% = 100\%$$

Pada perhitungan jenis penyakit tanaman padi yang divalidasi oleh pakar, hasilnya sesuai dengan yang diharapkan. Namun, dalam aplikasi SIKAPADI, terdapat beberapa kasus di mana akurasi atau nilai kepercayaan terhadap klasifikasi gambar menurun di bawah 85%. Hal ini disebabkan oleh kurangnya variasi data yang tersedia untuk melatih model. Sebagai contoh, pada Gambar 4.36 dapat dilihat bahwa terdapat hasil klasifikasi dengan akurasi rendah.

Gambar 4.36 Hasil Klasifikasi yang dibawah 85%

Untuk mengatasi masalah ini, diperlukan upaya untuk memperbanyak variasi data latih yang digunakan dalam pelatihan model. Dengan dataset yang lebih beragam, model akan menjadi lebih baik dalam mengenali berbagai jenis penyakit tanaman padi dengan lebih akurat. Langkah-langkah lain yang dapat diambil termasuk melakukan penyesuaian pada algoritma atau metode pembelajaran mesin yang digunakan, serta melakukan pengujian dan validasi yang lebih ketat terhadap model untuk memastikan kinerjanya secara konsisten di berbagai kondisi.

4.4 Revisi Hasil Implementasi *Website*

Revisi hasil implementasi *website* oleh pakar merupakan langkah yang penting untuk memastikan keberlanjutan aplikasi. Dengan melibatkan pakar, baik dalam pengalaman pengguna, klasifikasi tanaman maupun teknis, revisi dapat dilakukan dengan memperhatikan aspek keamanan, kinerja, dan kegunaan. Proses ini tidak hanya meningkatkan fungsionalitas *website*, tetapi juga memperbaiki masalah yang mungkin akan timbul, sehingga dapat menghasilkan pengalaman pengguna yang lebih baik dan meningkatkan daya saing aplikasi di pasar yang terus berubah. Untuk hasil yang belum direvisi seperti pada Gambar 4.37.

Gambar 4.37 Hasil Sebelum Direvisi

Hasil revisi *website* dapat disajikan dalam bentuk visual untuk memudahkan pemahaman dan identifikasi perubahan yang dilakukan. Salah satu contohnya adalah Gambar 4.38, yang menunjukkan perubahan pada website tersebut.

Gambar 4.38 Hasil yang telah direvisi

4.5 UAT (*User Acceptance Testing) Website* SIKAPADI

User Acceptance Testing (UAT) pada aplikasi SIKAPADI merupakan langkah krusial dalam memvalidasi keberhasilan fungsionalitas deteksi penyakit daun tanaman padi. UAT melibatkan partisipasi langsung pakar sebagai pengguna akhir untuk menguji kegunaan dan keandalan aplikasi.Berbagai skenario pemakaian diberikan kepada para pakar untuk memperoleh umpan balik terkait antarmuka pengguna, kecepatan respons, dan akurasi deteksi penyakit. Hasil UAT menjadi tolak ukur penting dalam mengevaluasi kesesuaian aplikasi SIKAPADI dengan kebutuhan dan harapan pengguna sesungguhnya.

Tabel 4.8 UAT *Website* SIKAPADI

ID	*Test Case Description*	*Test Case Procedure*	*Test Data*	*Expected Output*	*Test Date*	*Result*
1.*Home*						
TC 1	Konten *Home*	1.Buka website SIKAPADI	*User*	1. Dapat menampilkan halaman *Home* 2. Dapat menampilkan konten Layanan Kami 3. Dapat menampilkan konten Artikel 4. Dapat menampilkan FAQ (*Frequently Ask Question*) 5. Dapat menampilan konten pada Footer	18 Jan 2024	*Pass*
2. Klasifikasi						
TC 2	Klasifikasi Penyakit Tanaman Padi	1. Buka website SIKAPADI 2. Klik menu Klasifikasi	*User*	1. Terdapat halaman Klasifikasi Penyakit 2. Dapat *input* gambar 3. Dapat melakukan prediksi penyakit	18 Jan 2024	*Pass*

ID	*Test Case Description*	*Test Case Procedure*	*Test Data*	*Expected Output*	*Test Date*	*Result*
		3.Klik area dengan tulisan "Klik untuk memilih gambar" 4. Pilih Gambar Tanaman Padi 5. Klik button *Open*/Buka/Enter 6. Klik *button* Prediksi		4. Terdapat halaman hasil klasifikasi penyakit		
TC 3	Klasifikasi Menggunakan Gambar Lain	1. Buka website SIKAPADI 2. Klik menu Klasifikasi 3. Klik area dengan tulisan "Klik untuk memilih gambar" 4. Pilih Gambar Acak 5. Klik button Open/Buka/Enter 6. Klik button Prediksi	User	1. Terdapat halaman Klasifikasi Penyakit 2. Dapat input gambar 3. Tidak dapat melakukan prediksi penyakit 4. Terdapat halaman hasil klasifikasi penyakit	18 Jan 2024	Pass
3. Artikel						
TC 4	View Artikel	1. Buka website SIKAPADI 2. Klik menu Artikel 3. Klik salah satu Artikel	User	1. Terdapat halaman Artikel 2. Terdapat membuka Artikel 3. Dapat menampilkan konten Artikel	18 Jan 2024	Pass
4. Kirim Pertanyaan						
TC 5	Kirim Pertanyaan	1. Buka website SIKAPADI 2. Klik menu Kontak 3. Input Nama 4. Input Email 5. Input Subjek 6. Input Pertanyaan/Komentar 7. Klik button Kirim	User	1. Terdapat halaman Kontak 2. Dapat input Nama 3. Dapat input Email 4. Dapat input Subjek 5. Dapat input Pertanyaan/Komentar 6. Dapat menampilkan halaman nomor WhatsApp 7. Berhasil mengirimkan Pertanyaan/Komentar	18 Jan 2024	Pass

Berdasarkan data yang tersaji dalam Tabel 4.57, dapat disimpulkan bahwa semua fitur atau komponen yang diuji dalam *User Acceptance Testing* (UAT) telah memenuhi persyaratan yang telah ditetapkan. Hal ini menunjukkan bahwa

aplikasi/sistem yang diuji berfungsi dengan baik dan sesuai dengan kebutuhan pengguna.

BAB 5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil penelitian skripsi yang berjudul "Implementasi *Image Processing* Pada Penyakit Daun Tanaman Padi Dengan Metode CNN (Studi Kasus Kabupaten Tuban)," terdapat beberapa kesimpulan yang dapat diambil:

1. Implementasi teknologi pengolahan citra dan metode *Convolutional Neural Network* (CNN) dalam mendeteksi penyakit daun pada tanaman padi di Kabupaten Tuban melibatkan beberapa langkah, termasuk *preprocessing* data dengan teknik *resize* untuk mempercepat komputasi model. Algoritma CNN *Xception* digunakan dengan lapisan-lapisan yang telah ditentukan sesuai arsitektur untuk pembelajaran fitur dan klasifikasi.
2. Faktor-faktor yang mempengaruhi akurasi metode CNN dalam mengidentifikasi jenis penyakit pada daun tanaman padi termasuk persentase pembagian data *training*, ukuran/resolusi citra, penyebaran data serta normalisasi data. Selain itu, hasil prediksi juga dipengaruhi oleh variasi yang terjadi setiap iterasi serta data yang kurang bervariasi.
3. Tingkat akurasi sistem klasifikasi penyakit daun tanaman padi menggunakan algoritma *Convolutional Neural Network* (CNN) telah mencapai puncaknya pada 98,53%. Pengujian dilakukan dengan dua pembagian dataset menggunakan arsitektur *Xception* selama 25 *epoch*, dengan pembagian data untuk pelatihan, pengujian, dan validasi masing-masing sebesar 70%, 30% serta 80%, 20%. Meskipun beberapa *epoch* mengalami penurunan akurasi karena variasi hasil prediksi, namun secara keseluruhan, metode ini memberikan kontribusi positif dengan tingkat akurasi yang signifikan.

Oleh karena itu penelitian ini menunjukkan potensi besar dalam deteksi otomatis penyakit daun padi dengan tingkat akurasi tinggi. Penelitian ini membuka peluang untuk pengembangan sistem yang lebih handal dan efisien untuk membantu petani di Kabupaten Tuban dan wilayah lainnya.

5.2 Saran

Penelitian ini menunjukkan beberapa kekurangan yang dapat diperbaiki untuk penelitian mendatang. Sebagai langkah perbaikan, disarankan untuk memperluas dataset dengan menambah lebih banyak data dari sumber-sumber yang berbeda. Dengan menambahkan variasi data yang lebih besar, sistem dapat mempelajari pola penyakit daun tanaman padi dengan lebih baik dan dapat meningkatkan kinerja model.Dengan demikian, saran-saran ini dapat menjadi pedoman untuk pengembangan penelitian lebih lanjut dalam bidang ini

DAFTAR PUSTAKA

Aeni , S. N. 2022. 5 Cara Pengendalian hama Dan Penyakit Pada Tanaman padi. KOMPAS.com. 2023, from https://agri.kompas.com/read/2022/09/04/195544684/5-cara-pengendalian-hama-dan-penyakit-pada-tanaman-padi

Ahmadpour, A., Castell-Miller, C., Javan-Nikkhah, M., Naghavi, M. R., Dehkaei, F. P., Leng, Y., Puri, K. D., dan Zhong, S. 2017. *Population structure, genetic diversity, and sexual state of the rice brown spot pathogen bipolaris oryzae* from three Asian countries. *Plant Pathology*, *67*(1), 181–192. https://doi.org/10.1111/ppa.12714

Allaam, M. R. R., & Wibowo, A. T. 2021. Klasifikasi Genus Tanaman Anggrek Menggunakan Convolutional Neural Network. E-Proceeding of Engineering, 8(2), 1153–1189.

Anggreany, Dr. M. S. 2020. *Confusion matrix*. *School of Computer Science*. https://socs.binus.ac.id/2020/11/01/confusion-matrix/

Arsal, M., Agus Wardijono, B., & Anggraini, D. 2020. Face recognition Untuk Akses pegawai bank menggunakan deep learning Dengan Metode CNN. *Jurnal Nasional Teknologi Dan Sistem Informasi*, *6*(1), 55–63. https://doi.org/10.25077/teknosi.v6i1.2020.55-63

Azizah, Q. N. 2023. Klasifikasi penyakit daun jagung menggunakan metode convolutional neural network Alexnet. *Sudo Jurnal Teknik Informatika*, *2*(1), 28–33. https://doi.org/10.56211/sudo.v2i1.227

Baktikominfo. 2019. Badan Aksesibilitas telekomunikasi Dan Informasi. BAKTI.https://www.baktikominfo.id/id/informasi/pengetahuan/bahasa_pemrograman_python_pengertian_sejarah_kelebihan_dan_kekurangannya-954

Biro Humas Pemprov Jatim. 2022. Jawa Timur Kembali sebagai penghasil padi terbesar di Indonesia. Bappeda Provinsi Jawa Timur RSS. Retrieved April 14, 2023, from https://bappeda.jatimprov.go.id/2022/01/26/jawa-timur-kembali-sebagai-penghasil-padi-terbesar-di-indonesia/

BPS (Badan Pusat Statistik) Kabupaten Tuban. 2024. Kabupaten Tuban Dalam Angka 2024

Chollet, F. 2016. Xception: Deep Learning with Depthwise Separable Convolutions. CoRR, abs/1610.0. http://arxiv.org/abs/1610.02357

D. Janse, D. J. 2022. *Eppo Global Database*. *Xanthomonas oryzae pv. oryzae (XANTOR)[Datasheet]| EPPO Global Database*. Retrieved April 15, 2023, from https://gd.eppo.int/taxon/XANTOR/datasheet

DeVries, T., dan Taylor, G. W. 2018. Learning confidence for out-of-distribution detection in neural networks. ArXiv Preprint ArXiv:1802.04865.

Edbert, I. S. 2021. *Pooling layer. School of Computer Science*. Retrieved May 3, 2023, from https://socs.binus.ac.id/2021/10/07/pooling-layer/

Efanntyo, Mitra, A.R. 2021. Perancangan Aplikasi Sistem Pengenalan Wajah Dengan Metode *Convolutional Neural Network* (CNN) Untuk Pencatatan Kehadiran Karyawan. *Jurnal Instrumentasi dan Teknologi Informatika (JITI)* vol.3 no.1.

Faulina, A. R. 2023. *Apa Itu Uml? Ini Pengertian, Fungsi, Dan Contohnya*. Software House dan System Integrator di Malang, Indonesia. https://www.sekawanmedia.co.id/blog/apa-itu-uml/

Hawari, F. H., Fadillah, F., Alviandi, M. R., dan Arifin, T. 2022. Klasifikasi Penyakit tanaman padi Menggunakan algoritma CNN (*Convolutional Neural Network*). Jurnal Responsif : Riset Sains Dan Informatika, *4*(2), 184–189. https://doi.org/10.51977/jti.v4i2.856

Hidayatullah, P. 2017. Pengolahan citra digital : teori dan aplikasi nyata /penyusun, Priyanto Hidayatullah. Bandung : *Penerbit Informatika*, 2017 https://opac.perpusnas.go.id/DetailOpac.aspx?id=1059250#

Husain, S.ST. 2019. Budidaya Tanaman Padi *(Oryza sativa)*. Cyber extension. Retrieved April 14, 2023, from http://cybex.pertanian.go.id/mobile/artikel/84581/Budidaya-Tanaman-Padi---Oryza-Sativa-/

Kementerian Pertanian, Badan Penyuluhan dan Pengembangan Sumber Daya Manusia Pertanian. 2019. Pengendalian Hama Dan Penyakit padi sawah.

Cyber extension. Retrieved April 14, 2023, from http://www.cybex.pertanian.go.id/artikel/87628/pengendalian-hama-dan-penyakit-padi-sawah/

Khoiruddin, M., Junaidi, A., dan Saputra, W. A. 2022. Klasifikasi penyakit Daun padi Menggunakan *convolutional neural network*. *Journal of Dinda: Data Science, Information Technology, and Data Analytics*, *2*(1), 37–45. https://doi.org/10.20895/dinda.v2i1.341

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K. 2016. *Jupyter Notebooks-a publishing format for reproducible computational workflows*. ELPUB .https://doi.org/10.3233/978-1-61499-649-1-87

Lesmana, A. M., Fadhillah, R. P., dan Rozikin, C. 2022. Identifikasi penyakit pada citra daun kentang Menggunakan *convolutional neural network* (CNN). *Jurnal Sains Dan Informatika*, *8*(1), 21–30. https://doi.org/10.34128/jsi.v8i1.377

Manajang, D. J. P., Sompie, S. R. U. A., Jacobus, A. 2020. Implementasi *Framework Tensorflow Object Detection* Dalam Mengklasifikasi Jenis Kendaraan Bermotor. Jurnal Teknik Informatika vol.15 no.3. https://doi.org/10.35793/jti.15.3.2020.29775

Maulid, R. 2021. *Mengenal Flask, library machine learning python idaman developer*. Mengenal Flask, Library Machine Learning Python Idaman Developer. https://dqlab.id/mengenal-flask-library-machine-learning-python-idaman-developer

Mishra, M. 2020. *Convolutional Neural Networks, explained*. Medium. Retrieved May 8, 2023, from https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

Nartymov, D., Kharitonov, E., Dubina, E., Garkusha, S., Ruban, M., Istomin, N., dan Kostylev, P. 2021. *Studying of cultural properties of pyricularia oryzae cav. strains in the south of Russia*. *Microbiology Research*, *12*(1), 21–28. https://doi.org/10.3390/microbiolres12010003

Nisa', C., Puspaningrum, E. Y., dan Maulana, H. 2020. Penerapan metode *convolutional neural network* untuk Klasifikasi Penyakit Daun Apel Pada imbalanced data. *Prosiding Seminar Nasional Informatika Bela Negara*, *1*, 169–175. https://doi.org/10.33005/santika.v1i0.46

Nurhikmat, T. 2018. Implementasi Deep Learning Untuk *Image Classification* Menggunakan Algoritma *Convolutional Neural Network* (Cnn) Pada Citra Wayang Golek. 10.13140/Rg.2.2.10880.53768.

Nurwijayo, W. 2022. 14 Pilihan varietas padi unggul tahan kekeringan. Pupuk Organik GDM dan Suplemen Organik Cair GDM. Retrieved April 14, 2023, from https://gdm.id/varietas-padi-tahan-kering

Nuryanto, B. 2018. Pengendalian Penyakit tanaman padi berwawasan lingkungan melalui pengelolaan komponen epidemik. *Jurnal Penelitian Dan Pengembangan Pertanian, 37* (1), 1. https://doi.org/10.21082/jp3.v37n1.2018.p1-8 /

Parisa, N. 2021. Sistem Pertanian Berkelanjutan. Cyber extension. Retrieved April 14, 2023, from http://www.cybex.pertanian.go.id/artikel/98578/sistem-pertanian-berkelanjutan/

Pothen, M. E., dan Pai, D. M. 2020. *Detection of rice leaf diseases using image processing. 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*. https://doi.org/10.1109/iccmc48092.2020.iccmc-00080

Pradana, R. G. 2022. *Produksi padi di kabupaten Semarang Menurun Akibat terserang Hama*. Tribunjateng.com. https://jateng.tribunnews.com/2022/09/15/produksi-padi-di-kabupaten-semarang-menurun-akibat-terserang-hama

Retnowardhani, A., dan Ramdani, T. 2019. *Apakah deep learning ?* MMSI BINUS University. Retrieved May 2, 2023, from https://mmsi.binus.ac.id/2019/11/26/apakah-deep-learning/

Rosalina, R., dan Wijaya, A. 2020. Pendeteksian Penyakit Pada Daun cabai dengan menggunakan metode deep learning. *Jurnal Teknik Informatika Dan Sistem Informasi*, *6*(3). https://doi.org/10.28932/jutisi.v6i3.2857

Safrullah, S.P. 2019. Jenis Penyakit Utama Pada Tanaman padi. Cyber extension. Retrieved April 14, 2023, from http://cybex.pertanian.go.id/mobile/artikel/84516/Jenis-Penyakit-Utama-Pada-Tanaman-Padi-/

Saputra, R. A., Wasiyanti, S., Supriyatna, A., dan Saefudin, D. F. 2021. Penerapan algoritma convolutional neural network Dan Arsitektur Mobilenet Pada aplikasi Deteksi Penyakit Daun padi. *Swabumi*, *9*(2), 184–188. https://doi.org/10.31294/swabumi.v9i2.11678

Setiawan , R. 2021. Flowchart Adalah: Fungsi, Jenis, Simbol, Dan Contohnya. Dicoding Blog. Retrieved May 3, 2023, from https://www.dicoding.com/blog/flowchart-adalah/

Tiwari, D., Ashish, M., Gangwar, N., Sharma, A., Patel, S., dan Bhardwaj, S. 2020. *Potato leaf diseases detection using Deep Learning*. *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*. https://doi.org/10.1109/iciccs48265.2020.9121067

Wilcox, P. 2021. *How to use deep neural networks to forecast stock prices*. Neuravest. https://www.neuravest.net/deep-neural-networks-to-forecast-stock-prices/

Yanuar, A. 2018. *Fully-connected layer CNN* Dan Implementasinya. Universitas Gadjah Mada. Retrieved May 3, 2023, from https://machinelearning.mipa.ugm.ac.id/2018/06/25/fully-connected-layer-cnn-dan-implementasinya/

LAMPIRAN

Lampiran 1 Kode Untuk *Main.py* pada *flask*

```
# Inisialisasi Flask
app = Flask(__name__)


# Konfigurasi model dan definisi kelas prediksi
MODEL_ARCHITECTURE = 'model/clean721.json'
MODEL_WEIGHTS = 'model/clean721.h5'


   PREDICTION_CLASSES = {
      0:('Gambar Tidak Cocok', 'klasifikasi
     noklasifikasi.html'),
   1:('Tanaman Padimu Terkena', 'klasifikasi-bl.html'),
   2:('Tanaman Padimu Terkena', 'klasifikasi-blb.html'),
   3:('Tanaman Padimu Terkena', 'klasifikasi-bw.html'),
   4:('Tanaman', 'klasifikasi-sh.html'),
   }


# Fungsi untuk memuat model dari file
def load_model_from_file():
   try:
       with open(MODEL_ARCHITECTURE, 'r') as json_file:
           loaded_model_json = json_file.read()
           model = model_from_json(loaded_model_json)
           model.load_weights(MODEL_WEIGHTS)
           return model
       except Exception as e:
           logging.error(f"Error loading model: {str(e)}")
           return None

# Memuat model saat aplikasi dimulai
model = load_model_from_file()

# Fungsi untuk melakukan prediksi menggunakan model
def model_predict(img_path, model):
    try:
        TARGET_IMAGE_SIZE = (224, 224)
        test_image = load_img(img_path,
                              target_size=TARGET_IMAGE_SIZE)
        logging.info("@@ Got Image for prediction")

        test_image = img_to_array(test_image) / 255
        test_image = np.expand_dims(test_image, axis=0)

        result = model.predict(test_image)
        pred_class = np.argmax(result, axis=1)[0]
        pred_prob = result[0][pred_class] * 100
```

```
49)
50)             return PREDICTION_CLASSES[pred_class][0],
51)                    PREDICTION_CLASSES[pred_class][1],
52)                    pred_prob
53)         except Exception as e:
54)             logging.error(f"Error predicting: {str(e)}")
55)             return 'Error', 'error.html', 0
56)
57)
58) # Routing untuk halaman utama
59) @app.route("/", methods=['GET'])
60) def home():
61)     return render_template('index.html')
62)
63)
64) # Routing untuk halaman Blog
65) @app.route("/blog", methods=['GET'])
66) def blog():
67)     return render_template('blog.html')
68)
69)
70) # Routing untuk halaman Blog
71) @app.route("/contact", methods=['GET'])
72) def contact():
73)     return render_template('contact.html')
74)
75)
76) # Routing untuk halaman klasifikasi
77) @app.route("/klasifikasi", methods=['GET', 'POST'])
78) def klasifikasi():
79)     return render_template('klasifikasi.html')
80)
81)
82) # Routing untuk halaman prediksi
83) @app.route("/predict", methods=['GET', 'POST'])
84) def predict():
85)     if request.method == 'POST':
86)         file = request.files['image']
87)         filename = secure_filename(file.filename)
88)         logging.info("@@ Input posted =", filename)
89)
90)         file_path = os.path.join('static', 'user_uploaded',
91)                                  filename)
92)         file.save(file_path)
93)
94)         logging.info("@@ Predicting class...")
95)         pred_class, output_page, pred_prob =
96)         model_predict(file_path, model)
97)
98)         return render_template(output_page,
99)                                pred_output=pred_class,
100)                               pred_prob=pred_prob,
101)                               user_image=file_path)
102)
103)
104)if __name__ == "__main__":
```

```
105)     app.run(debug=True)
```

Lampiran 2 Kode Agar *Dataset* Tidak Terjadi Penumpukan Data

```
1) from PIL import Image
2) import imagehash
3) import os
4)
5) def calculate_hash(image_path):
6)     # Menghitung hash dari gambar
7)     with Image.open(image_path) as img:
8)         hash_value = imagehash.phash(img)
9)     return hash_value
10)
11)   def clean_duplicate_images(input_folder, output_folder):
12)       # Membuat direktori output jika belum ada
13)       if not os.path.exists(output_folder):
14)           os.makedirs(output_folder)
15)
16)       # Membuat kamus untuk menyimpan hash gambar yang telah
  ditemukan
17)       hash_dict = {}
18)
19)       # Loop melalui setiap file dalam direktori input
20)       for filename in os.listdir(input_folder):
21)           file_path = os.path.join(input_folder, filename)
22)
23)           # Memeriksa apakah file adalah file gambar
24)           if os.path.isfile(file_path) and
  any(filename.lower().endswith(ext) for ext in ['.jpg','.JPG',
  '.jpeg', '.png']):
25)               # Menghitung hash gambar
26)               hash_value = calculate_hash(file_path)
27)
28)               # Memeriksa apakah hash telah ada dalam kamus
  (duplikat)
29)               if hash_value in hash_dict:
30)                   print(f"Duplicate found: {filename}")
31)               else:
32)                   # Menyimpan hash gambar dan menyimpan
  gambar ke direktori output
33)                   hash_dict[hash_value] = filename
34)                   output_path = os.path.join(output_folder,
  filename)
35)                   os.rename(file_path, output_path)
36)
37)       print("Pembersihan dataset selesai.")
38)
39)   # Contoh penggunaan:
40)   input_folder = r"E:\1"
41)   output_folder = r"E:\1\CLEAN"
42)   clean_duplicate_images(input_folder, output_folder)
```

Lampiran 3 Kode Untuk *Resize* Data

```
import cv2
import os

def preprocess_image(image_path, target_size=(1600, 1600)):
    img = cv2.imread(image_path)

    img = cv2.resize(img, target_size)

    return img

# Path direktori dataset
dataset_dir = r"E:\SKRIPSI Fiqri\dataset coba baru\PO"

# Path direktori untuk menyimpan gambar yang sudah diproses
  output_dir = r"E:\SKRIPSI Fiqri\dataset 1600\padi bl"

# Buat direktori output jika belum ada
os.makedirs(output_dir, exist_ok=True)

# List semua file gambar dalam direktori
image_files = [f for f in os.listdir(dataset_dir) if
  f.endswith(('.jpg','.JPG' ,'.jpeg', '.png'))]

# Membersihkan dan menyamakan ukuran semua gambar dalam
  dataset
for image_file in image_files:
    image_path = os.path.join(dataset_dir, image_file)
    cleaned_image = preprocess_image(image_path)

    # Simpan gambar yang sudah diproses ke direktori baru
    output_path = os.path.join(output_dir, image_file)
    cv2.imwrite(output_path, cleaned_image)

print("Proses selesai. Gambar yang sudah diproses tersimpan
  di:", output_dir)
```

Lampiran 4 Kode Program Mengambil Nilai RGB

```
from PIL import Image
import numpy as np

# Membaca gambar dan meresize menjadi 5x5
image_path = 'Picture1.jpg'  # Ganti dengan path gambar
  Anda
original_image = Image.open(image_path)
resized_image = original_image.resize((4, 4))


      # Mendapatkan nilai pixel untuk setiap channel (BGR)
  dari gambar yang diresize
      pixel_values = np.array(resized_image)

      blue_values = pixel_values[:, :, 2]
```

```
    green_values = pixel_values[:, :, 1]
    red_values = pixel_values[:, :, 0]

    # Menampilkan nilai pixel
    print("Blue values:")
    print(blue_values)

    print("\nGreen values:")
    print(green_values)

    print("\nRed values:")
print(red_values)
```

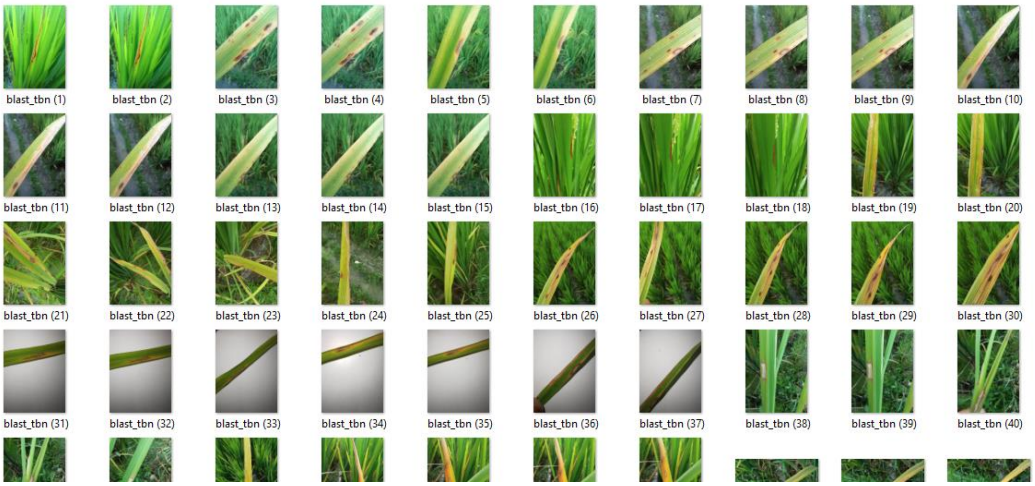
Lampiran 5 Foto bersama Kepala Seksi Pengendalian Hama dan Penyakit

Lampiran 6 Observasi Secara Langsung Tanamam Padi di Lapangan

Lampiran 7 Citra Tanaman Padi Sehat

Lampiran 8 *Xanthomonas oryzae pv. Oryzicola* (Penyakit *Leaf Blight*)

Lampiran 9 *Pyricularia oryzae Cav* (Penyakit *Blast*)



Lampiran 10 *Brown Spot* (Bercak Coklat)