IMPLEMENTASI MODEL *DENSENET* BERBASIS *DEEP LEARNING* UNTUK KLASIFIKASI GAMBAR HAMA PADA TANAMAN PADI

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro Fakultas Teknik Univeristas Islam Nusantara Bandung

Oleh:

INDRA SAPUTRA

41037002211020

UNINUS
UNIVERSITAS ISLAM NUSANTARA

PROGRAM STUDI TEKNIK ELEKTO

FAKULTAS TEKNIK

UNIVERSITAS ISLAM NUSANTARA

2025

LEMBAR KEASLIAN SKRIPSI

Yang bertanda tangan dibawah ini:

Nama : Indra Saputra

Nim : 41037002211020

Program studi : Teknik Elektro

Menyatakan bahwa skripsi yang berjudul :

IMPLEMENTASI MODEL DENSENET BERBASIS DEEP LEARNING UNTUK KLASIFIKASI GAMBAR HAMA PADA TANAMAN PADI dibuat dengan sebenar-benarnya dari penelitian, pemikiran, dan pemaparan hasil saya sendiri, untuk melengkapi sebagai pernyataan menjadi Sarjana (S1) pada jurusan Teknik Elektro Fakultas Teknik Universitas Islam Nusantara Bandung, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari buku Skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan jenjang Sarjana (S1) di lingkungan Teknik Elektro Fakultas Teknik Universitas Islam Nusantara Bandung maupun perguruan-perguruan tinggi atau instansi manapun kecuali bagian yang sumber informasi dicantumkan sebagaimana mestinya.

Bandung, 15 April 2025

Yang membuat pernyataan,

INDRA SAPUTRA

NIM. 41037002211020

LEMBAR PENGESAHAN

IMPLEMENTASI MODEL DENSENET BERBASIS DEEP LEARNING UNTUK KLASIFIKASI GAMBAR HAMA PADA TANAMAN PADI

Disusun dan diajukan oleh:

INDRA SAPUTRA

41037002211020

Disetujui dan disahkan pada Sidang Skripsi

pada tanggal:

Bandung, 15 Mei 2025

Pembimbing I	Pembimbing II
Ganis Sanhaji, S.Si., M.Sc	**Ir. Joko Haryatno, M.T**

Mengetahui

Dekan Fakultas Teknik	Ketua Prodi Teknik Elektro
Dr. Ricky Yoseptry, S.T., M.M.Pd	**Muhammad Zimamul Adli, M.Si**

LEMBAR PENGESAHAN

REVISIAN SKRIPSI

IMPLEMENTASI MODEL DENSENET BERBASIS DEEP LEARNING UNTUK KLASIFIKASI GAMBAR HAMA PADA TANAMAN PADI

Telah Direvisi

Oleh:

INDRA SAPUTRA

41037002211020

Bandung, 19 Mei 2025

Mengesahkan

Penguji I	Penguji II
Dr. Iksal Rachman, M.T	**Ryan Nur Iman, S.Si., M.Sc**

Ketua Sidang

Muhammad Zimamul Adli, M.Si

BIODATA PENULIS

Nama	: Indra Saputra
Tempat, Tanggal lahir	: Bandung, 24 Juli 2003
Telepon	: +6285603523458
Email	: indr4saputra247@gmail.com
Riwayat Pendidikan	: SD Negeri Sukajadi
	SMP Al-Qona'ah
	SMK Negeri 7 Baleendah

KATA PENGANTAR

Segala puja dan puji syukur kehadirat Tuhan Yang Maha Esa, yang telah melimpahkan rahmat dan kesehatan kepada kita semua sehingga penyusun dapat menyelesaikan penyusunan skripsi ini yang berjudul ***Implementasi Model DenseNet Berbasis Deep Learning untuk Klasifikasi Gambar Hama pada Tanaman Padi.***

Walaupun demikian penyusun berusaha dengan semaksimal mungkin demi kesempurnaan skripsi ini. Saran dan kritik yang bersifat membangun begitu diharapkan oleh penyusun demi kesempurnaan. Dalam kesempatan ini penyusun mengucapkan terimakasih kepada semua pihak yang telah membantu dalam penyusunan skripsi ini, diantaranya :

1. Kepada Tuhan yang telah memberikan kenikmatan dan kesehatan sampai saat ini.
2. Ibu Wiwin Winarsih dan Alm Bapak Pepen Ruspendi selaku orang tua yang selalu memberikan dorongan secara motivasi, doa dan semangat hingga dapat menyelesaikan skripsi ini.
3. Bapak Prof. Dr. Endang Komara, M.Si selaku Rektor Universitas Islam Nusantara.
4. Bapak Dr. Ricky Yoseptri, S.T., M.M. Pd selaku Dekan Fakultas Teknik, Universitas Islam Nusantara.
5. Bapak Muhammad Zimamul Adli, M.Si selaku Ketua Program Studi Teknik Elektro, yang senantiasa memberikan pengarahan dan nasihat kepada penyusun.
6. Bapak Ganis Sanhaji, S.Si., M.Sc selaku dosen pembimbing I, yang selalu memberi dukungan, bimbingan, arahan, masukan dan semangat kepada penyusun sehingga bisa menyelesaikan skripsi ini.
7. Bapak Ir. Joko Haryatno, M.T selaku dosen pembimbing II, yang selalu memberi dukungan, bimbingan, arahan, masukan dan semangat kepada penyusun sehingga bisa menyelesaikan skripsi ini.
8. Seluruh Dosen Fakultas Teknik yang telah memberikan ilmu yang bermanfaat bagi penyusun.

9. Teman-teman kelas teknik elektro angkatan 2021 yang selalu memberi semangat satu sama lain.
10. Semua pihak yang tidak dapat penyusun sebutkan satu persatu.

Semoga atas bantuan dengan ketulusan hati yang telah diberikan oleh semua pihak dibalas oleh Tuhan dan semoga langkah kita selalu dalam lindungan-Nya. Demikian, Semoga skripsi ini dapat diterima sebagai ide atau gagasan yang menambah kekayaan intelektual bangsa.

Bandung, 15 April 2025

INDRA SAPUTRA

NIM. 41037002211020

ABSTRAK

Nama : Indra Saputra

Program Studi : Teknik Elektro

Judul : IMPLEMENTASI MODEL DENSENET BERBASIS DEEP LEARNING UNTUK KLASIFIKASI GAMBAR HAMA PADA TANAMAN PADI

Serangan hama pada tanaman padi merupakan salah satu faktor utama yang menyebabkan penurunan produktivitas hasil pertanian di Indonesia. Deteksi dini terhadap jenis hama sangat penting agar penanganan dapat dilakukan secara cepat dan tepat. Penelitian ini bertujuan untuk mengembangkan sistem klasifikasi gambar hama tanaman padi menggunakan metode CNN dengan arsitektur DenseNet dan mengintegrasikannya ke dalam aplikasi berbasis web menggunakan Streamlit.

Dataset yang digunakan terdiri dari enam kelas hama utama pada tanaman padi, seperti belalang, ulat grayak, tikus sawah dan lainnya. Data kemudian melalui tahap preprocessing dan augmentasi untuk meningkatkan performa model. Model dilatih dengan beberapa rasio pembagian data pelatihan dan validasi, seperti 70:30, dan 80:20. Evaluasi dilakukan menggunakan metrik akurasi, presisi, recall, dan F1-score. Hasil pelatihan menunjukkan bahwa DenseNet mampu mengklasifikasikan jenis hama dengan tingkat akurasi tinggi, mencapai lebih dari 80% pada konfigurasi optimal.

Model yang telah dilatih kemudian diintegrasikan ke dalam aplikasi web interaktif, yang memungkinkan pengguna mengunggah gambar hama dan mendapatkan hasil klasifikasi beserta informasi solusi penanganannya secara otomatis. Sistem ini diharapkan dapat membantu petani dan penyuluh pertanian dalam melakukan deteksi cepat dan pengambilan keputusan yang tepat terhadap serangan hama di lapangan.

Kata kunci: Klasifikasi hama, tanaman padi, DenseNet, CNN

ABSTRACT

Name : Indra Saputra

Study Program : Electrical Engineering

Title : *IMPLEMENTATION OF DENSENET MODEL BASED ON DEEP LEARNING FOR CLASSIFICATION OF PEST IMAGES IN RICE PLANTS*

Pest attacks on rice plants are one of the main factors causing a decline in agricultural productivity in Indonesia. Early detection of pest types is crucial for prompt and accurate handling. This research aims to develop a pest image classification system for rice plants using the CNN method with DenseNet architecture and integrate it into a web-based application using Streamlit.

The dataset used consists of six main classes of pests on rice plants, such as locusts, armyworms, field mice, and others. The data then undergoes preprocessing and augmentation stages to enhance model performance. The model is trained with several training and validation data split ratios, such as 70:30 and 80:20. Evaluation is conducted using metrics of accuracy, precision, recall, and F1-score. The training results show that DenseNet is capable of classifying pest types with a high accuracy level, reaching over 80% in optimal configurations.

The trained model is then integrated into an interactive web application, allowing users to upload pest images and receive classification results along with information on handling solutions automatically. This system is expected to assist farmers and agricultural extension workers in making quick detections and accurate decisions regarding pest attacks in the field.

Keywords: Pest classification, rice plants, DenseNet, CNN.

DAFTAR ISI

LEMBAR KEASLIAN SKRIPSI i
LEMBAR PENGESAHAN ii
LEMBAR PENGESAHAN iii
BIODATA PENULIS iv
KATA PENGANTAR v
ABSTRAK vii
ABSTRACT viii
DAFTAR ISI ix
DAFTAR GAMBAR xi
DAFTAR TABEL xii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Perumusan masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan Penelitian 3
1.5 Manfaat Penelitian 3
1.6 Metode Penulisan 4
1.7 Metode Studi Pustaka 4
1.8 Sistematika Penulisan 4
BAB II LANDASAN TEORI 5
2.1 State of Arts 5
2.2 Klasifikasi Citra Digital 8
2.3 Hama Tanaman Padi 9
2.4 Solusi Penanganan Hama 11
2.5 Convolutional Neural Network (CNN) 12
2.6 DenseNet 13
2.7 Integrasi Model dalam Aplikasi 15
2.8 Evaluasi Model 15
2.9 Kerangka Berpikir 17
BAB III PERANCANGAN SISTEM & IMPLEMENTASI 18
3.1 Metode Penelitian 18

3.2 Perancangan Blok Diagram 21
BAB IV HASIL & PEMBAHASAN 28
4.1 Implementasi Model Klasifikasi Hama Tanaman Padi 28
4.2 Pengaruh Augmentasi 32
4.3 Perbandingan Kinerja Model DenseNet, CNN biasa dan VGG16 33
4.4 Evaluasi Model 34
4.5 Implementasi sistem 37
BAB V KESIMPULAN 40
5.1 Kesimpulan 40
5.2 Saran 40
REFRENSI 42
LAMPIRAN 45

DAFTAR GAMBAR

Gambar 1. 1 Hama Tanaman Padi (Penggerek Batang) 1
Gambar 2. 1 Walang Sangit 10
Gambar 2. 2 Arsitektur CNN 13
Gambar 2. 3 Arsitektur DenseNet 14
Gambar 3. 1 Flowchat Sistem 22
Gambar 3. 2 Sourcecode Preprocessing 23
Gambar 3. 3 Sourcode Data Augmentasi 24
Gambar 4. 1 Grafik Train & Valid Acc 30
Gambar 4. 2 Hasil Klasifikasi Hama 31
Gambar 4. 3 Contoh Augmentasi Gambar dari Kelas Penggerek Batang 32
Gambar 4. 4 Perbandingan CNN, VGG dan DenseNet201 34
Gambar 4. 5 Tampilan Website 38
Gambar 4. 6 Hasil Pengujian Model 39

DAFTAR TABEL

Tabel 2. 1 State of Arts 5
Tabel 2. 2 Solusi Penanganan Hama 11
Tabel 4. 1 Pembagian Data 29
Tabel 4. 2 Tabel Akurasi 100 Epoch 35
Tabel 4. 3 Tabel Confusion Matrix 36

BAB I
PENDAHULUAN

1.1 Latar Belakang

Tanaman padi merupakan sumber pangan utama bagi masyarakat Indonesia dan memiliki peran vital dalam menjaga ketahanan pangan nasional. Namun, produktivitas tanaman padi kerap terganggu oleh serangan berbagai jenis hama, seperti wereng, ulat grayak, dan penggerek batang. Serangan hama ini dapat menurunkan hasil panen secara signifikan, bahkan menyebabkan gagal panen jika tidak segera ditangani. Oleh karena itu, identifikasi dini terhadap jenis hama yang menyerang sangat penting dilakukan secara cepat dan akurat. (Sitompul et al., 2022)

Saat ini, sebagian besar petani masih mengandalkan metode manual untuk mengenali jenis hama, yaitu melalui pengamatan langsung atau bantuan dari tenaga penyuluh. Metode ini rentan terhadap kesalahan identifikasi, terutama bagi petani yang belum memiliki pengetahuan cukup tentang ciri-ciri hama. Selain itu, keterlambatan dalam proses identifikasi juga menghambat proses pengendalian hama secara efektif. Untuk itu, dibutuhkan solusi teknologi yang mampu mengotomatisasi proses klasifikasi hama dengan tingkat akurasi tinggi. (Faizin et al., 2022)

Gambar 1. 1 Hama Tanaman Padi (Penggerek Batang)

(Citra, 2023)

Kemajuan teknologi dalam bidang pengolahan citra dan pembelajaran mesin (machine learning) memberikan peluang besar untuk mengembangkan sistem klasifikasi hama secara otomatis berbasis gambar. Salah satu pendekatan yang populer dan terbukti efektif adalah penggunaan jaringan saraf tiruan dalam bentuk Convolutional Neural Network (CNN). CNN memiliki kemampuan mengenali pola-pola visual pada gambar dan telah digunakan secara luas dalam klasifikasi objek, termasuk citra daun atau hama tanaman. (Yuliany et al., 2022)

CNN dapat dipahami sebagai bentuk pembelajaran statistik yang memaksimalkan kemungkinan prediksi berdasarkan pola yang muncul dalam data. Konsep ini relevan dalam klasifikasi citra karena CNN mampu mempertahankan informasi spasial sambil mengurangi kompleksitas komputasi melalui penggunaan parameter berbagi (shared weights). Proses pelatihan dilakukan dengan algoritma *backpropagation* yang secara iteratif menyesuaikan bobot jaringan untuk meminimalkan kesalahan klasifikasi. Dengan pendekatan ini, CNN telah terbukti sangat efektif dalam berbagai aplikasi pengenalan pola, termasuk di bidang pertanian digital (Bishop, 2006)

Salah satu varian CNN yang menunjukkan performa tinggi dalam klasifikasi citra adalah DenseNet (Densely Connected Convolutional Networks). DenseNet dikenal memiliki keunggulan dalam efisiensi parameter dan mitigasi masalah vanishing gradient melalui koneksi antar layer yang padat. Dalam konteks klasifikasi hama tanaman padi, DenseNet mampu memproses informasi dari gambar daun atau tubuh hama dengan lebih mendalam dan presisi. (Pailus, 2022)

Melalui penelitian ini, penulis bertujuan untuk mengembangkan sistem klasifikasi otomatis gambar hama tanaman padi menggunakan arsitektur DenseNet. Sistem ini diharapkan dapat membantu petani dan pihak terkait dalam mengidentifikasi jenis hama secara cepat dan memberikan informasi penanganan yang sesuai. Dengan demikian, penggunaan teknologi ini dapat meningkatkan efektivitas pengendalian hama dan mendukung pertanian cerdas yang berkelanjutan.

1.2 Perumusan masalah

1. Bagaimana model klasifikasi berbasis DenseNet dapat mengenali jenis hama berdasarkan citra daun atau tubuh hama?
2. Bagaimana cara mengintegrasikan hasil klasifikasi dengan informasi solusi atau penanganan hama secara otomatis?
3. Seberapa akurat performa model dalam mengklasifikasikan berbagai jenis hama tanaman padi?

1.3 Batasan Masalah

1. Model yang digunakan adalah DenseNet, dan tidak dibandingkan dengan arsitektur lain secara mendalam.
2. Solusi atau penanganan hama hanya berupa informasi deskriptif yang ditampilkan berdasarkan hasil klasifikasi, tidak mencakup tindakan fisik otomatis.
3. Implementasi dilakukan dalam bentuk antarmuka sederhana berbasis web dan Google Colab, bukan dalam bentuk aplikasi mobile atau perangkat IoT.

1.4 Tujuan Penelitian

1. Mengembangkan model klasifikasi berbasis DenseNet yang mampu mengenali jenis hama berdasarkan citra daun atau tubuh hama?
2. Merancang sistem integrasi antara hasil klasifikasi model dengan informasi penanganan atau solusi pengendalian hama secara otomatis.
3. Mengevaluasi performa model DenseNet dalam mengklasifikasikan berbagai jenis hama tanaman padi dengan mengukur tingkat akurasi, presisi, recall, dan F1-score.

1.5 Manfaat Penelitian

1. Membantu petani dalam mengenali jenis hama dengan cepat untuk melakukan penanganan yang tepat.
2. Menjadi referensi dan dasar pengembangan sistem deteksi hama berbasis citra dan deep learning lebih lanjut.
3. Memberikan alat bantu berbasis teknologi untuk pemantauan dan mitigasi serangan hama secara efisien.

4. Menambah literatur dan implementasi nyata dari metode deep learning dalam bidang pertanian.

1.6 Metode Penulisan

Dalam penelitian ini penulis menggunakan metode observasi, Observasi diperlukan karena untuk menganalisa terhadap aspek-aspek lain dalam fungsi dan sistem alat.

1.7 Metode Studi Pustaka

Untuk menunjang metode wawancara dan observasi penulis melakukan metode studi pustaka dimana dilakukan denga mencari referensi-referensi yang berhubungan dengan judul penelitian.

1.8 Sistematika Penulisan

Sistematika dalam penulisan ini adalah sebagai berikut:

BAB I : PENDAHULUAN

Membahas latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan.

BAB II : LANDSAN TEORI

Menguraikan teori-teori dan penelitian terdahulu yang relevan dengan topik penelitian.

BAB III : ANALISIS & PERANCANGAN SISTEM

Menjelaskan metode penelitian yang digunakan, seperti rancangan sistem, alat dan bahan, serta prosedur eksperimen.

BAB IV : HASIL DAN PEMBAHASAN

Menyajikan hasil penelitian yang telah dilakukan, termasuk data eksperimen, analisis hasil, dan interpretasi temuan.

BAB V : KESIMPULAN

Berisi kesimpulan dari penelitian berdasarkan hasil dan pembahasan yang telah dilakukan serta saran untuk pengembangan penelitian selanjutnya.

BAB II
LANDASAN TEORI

2.1 State of Arts

Tabel 2. 1 State of Arts

No	Penulis	Judul	Jenis Penulisan	Hasil Pembahasan
1	Susi Yuliany et al. (2022)	Implementasi Deep Learning untuk Klasifikasi Hama Tanaman Padi (CNN)	Jurnal	Penelitian ini menggunakan CNN standar untuk klasifikasi 3 jenis hama padi. Akurasi pada data pelatihan adalah 83,02% dan data pengujian 69,33%. Penurunan akurasi menunjukkan kemungkinan overfitting.
2	Mursyid S. Nugroho & Eddy Nurraharjo (2023)	Klasifikasi Hama Padi Berdasarkan Citra Daun Menggunakan CNN	Jurnal	Model CNN dibuat menggunakan TensorFlow dengan akurasi awal 61%. Setelah dilakukan tuning, augmentasi, dan normalisasi data, akurasi meningkat hingga 99%. Hasil menunjukkan pentingnya preprocessing dan tuning dalam model klasifikasi.
3	Arif Faizin, A. T. (2022)	Deep Pre-Trained Model Menggunakan Arsitektur DenseNet untuk Identifikasi Penyakit Daun Padi	Jurnal	Penggunaan DenseNet efektif dan dapat membantu petani dalam mendeteksi penyakit secara cepat dan akurat, terutama jika diintegrasikan

No	Penulis	Judul	Jenis Penulisan	Hasil Pembahasan
				dalam aplikasi berbasis web.
4	Agung Pambudi et al. (2024)	Deteksi Hama Wereng Batang Cokelat (Nilaparvata Lugens) Pada Tanaman Padi Menggunakan Deep Learning.	Jurnal	Penelitian ini menggunakan algoritma YOLOv4 untuk mendeteksi hama wereng batang cokelat pada tanaman padi. Dataset terdiri dari 13.870 objek hama dari 986 citra
5	Bagus M Kusuma et al (2025)	Klasifikasi Jenis Penyakit Pada Tanaman Padi Menggunakan Algoritma Convolutional Neural Network	Jurnal	Pada penelitian ini dilakukan pembangunan dan pelatihan model CNN untuk mengenali kondisi kesehatan tanaman padi. Model dilatih dengan dataset citra daun padi yang berasal dari situs Kaggle,com dengan jumlah data sebanyak 53.300 data citra, dan terdiri dari 4 kategori yaitu Brownspot, Leafblast, Hispa, dan Healthy
6	Amilia Umi Astagina (2025)	Klasifikasi Hama dan Penyakit Tanaman Padi Menggunakan Algoritma Decision Tree	Jurnal	Dataset ini memiliki atribut seperti umur tanaman, tinggi tanaman, kondisi daun, serta tanda tanda hama dan penyakit. Proses klasifikasi dilaksanakan dengan menggunakan bahasa pemrograman Python, yang mencakup langkah-langkah

No	Penulis	Judul	Jenis Penulisan	Hasil Pembahasan
				prapemrosesan data, pemisahan antara data latih dan data uji, akurasi, presisi, recall, dan f1-score
7	Amanda Caecilia Milano1 et al (2024)	Klasifikasi Penyakit Daun Padi Menggunakan Model Deep Learning Efficientnet-B6	Jurnal	Pada penelitian ini menggunakan 3355 citra daun tanaman padi yang dibagi menjadi empat kelas yaitu Healthy sebanyak 1488 citra, LeafBlast sebanyak 779 citra, Hispa sebanyak 565 citra, dan BrownSpot sebanyak 523 citra
8	Gracia Yoel Christiawan et Al (2023)	Penerapan Metode Convolutional Neural Network (CNN) Dalam Mengklasifikasikan Penyakit Daun Tanaman Padi	Jurnal	Penelitian ini menggunakan dataset berjumlah 1630 data yang dibagi menjadi 3 kelaspenyakit.. Hasil dari penelitian ini menunjukan hasil yang sangat baik di angkat 98% dengan data yang tidak overfitting.
9	Nurul Istiqomah et Al (2024)	Klasifikasi Penyakit Tanaman Padi Berbasis Citra Daun Menggunakan Convolutional Neural Network (CNN)	Jurnal	Penelitian ini bertujuan untuk mengetahui hasil akurasi dan mempermudah petani dalam mengklasifikasikan jenis penyakit tanaman padi menggunakan convolutional neural network dengan arsitektur VGG16 yang telah dilatih sebelumnya pada dataset ImageNet

No	Penulis	Judul	Jenis Penulisan	Hasil Pembahasan
10	Deteksi Hama Penyakit daun padi Dengan Menggunakan Teknik Optimasi Deep Learning CNN	Panji Novantara (2025)	Jurnal	Model CNN yang dibangun diuji menggunakan 480 gambar sampel dan menghasilkan akurasi tinggi sebesar 97,75%
Perbedaan Penelitian terdahulu dengan penelitian yang tertulis lakukan ialah:				
11	Indra Saputra (2025)	Implementasi Model DenseNet Berbasis Deep Learning untuk Klasifikasi Gambar Hama pada Tanaman Padi.	Tugas akhir	Penelitian ini menggunakan model CNN arsitektur DenseNet untuk klasifikasi hama tanaman padi, Dataset sendiri terdiri 6 kelas dengan total gambar ada 1200. Hasil Evaluasi Model F1-score tertinggi 80%, precision 87%, dan recall 96%

2.2 Klasifikasi Citra Digital

Klasifikasi citra digital adalah proses mengelompokkan piksel dalam suatu gambar ke dalam kelas-kelas tertentu berdasarkan karakteristik spektral atau tekstural yang serupa. Dalam konteks ini, citra diolah secara numerik untuk menghasilkan label kelas yang merepresentasikan objek dalam gambar. Proses ini sangat penting dalam berbagai bidang, termasuk pertanian, kedokteran, dan penginderaan jauh. Salah satu tahapan utama dalam klasifikasi citra adalah preprocessing, yaitu tahap awal untuk meningkatkan kualitas gambar sebelum dilakukan klasifikasi. Langkah ini mencakup normalisasi, resizing, dan konversi ke format yang sesuai. Preprocessing yang baik akan meningkatkan akurasi dari hasil klasifikasi. (Wahyuni, 2021)

Deep learning, khususnya Convolutional Neural Network (CNN), telah banyak digunakan untuk klasifikasi citra karena kemampuannya dalam mengekstrak fitur secara otomatis dari gambar mentah. CNN bekerja dengan menerapkan filter ke citra dan membangun representasi hierarkis dari fitur. Pemilihan arsitektur model yang tepat, seperti DenseNet atau ResNet, dapat mempengaruhi hasil klasifikasi secara signifikan. DenseNet, misalnya, dikenal mampu mengatasi masalah degradasi pada jaringan dalam dan meningkatkan efisiensi parameter (Kurniawan, 2022)

Selain model, jumlah dan kualitas data latih juga sangat berpengaruh. Dataset yang besar dan beragam dapat membantu model mempelajari pola dengan lebih baik dan mengurangi kemungkinan overfitting. Evaluasi kinerja sistem klasifikasi dilakukan menggunakan metrik seperti akurasi, presisi, recall, dan F1-score. Metrik ini memberikan gambaran seberapa baik model dalam mengklasifikasikan data yang belum dikenal. (Siregar, 2020)

Dalam penelitian citra digital di bidang pertanian, klasifikasi digunakan untuk mendeteksi jenis tanaman, penyakit, dan hama. Teknologi ini sangat membantu petani dalam melakukan pengambilan keputusan yang tepat berbasis data visual. Klasifikasi citra digital tidak hanya bermanfaat dalam segmentasi objek, tetapi juga dalam pemantauan kondisi tanaman secara real-time ketika dikombinasikan dengan teknologi IoT. Hal ini meningkatkan efisiensi pemeliharaan tanaman di lahan pertanian modern. (Wijayanti, 2023)

2.3 Hama Tanaman Padi

Hama merupakan salah satu faktor utama yang menurunkan produktivitas tanaman padi. Hama adalah organisme pengganggu yang memakan, mengisap, atau merusak jaringan tanaman dan dapat menyebabkan kerusakan fisik maupun fisiologis. Serangan hama yang tidak dikendalikan dengan baik dapat menurunkan hasil panen hingga lebih dari 50% tergantung tingkat infestasi dan fase tanaman. (Sastrosiswojo, 2022) Di Indonesia, beberapa jenis hama utama tanaman padi yang sering ditemukan antara lain adalah wereng cokelat (*Nilaparvata lugens*), penggerek batang (*Scirpophaga incertulas*), ulat grayak (*Spodoptera litura*), walang sangit (*Leptocorisa oratorius*), dan tikus sawah (*Rattus argentiventer*).

Setiap jenis hama memiliki cara serang dan gejala yang berbeda, sehingga identifikasi yang cepat dan tepat sangat dibutuhkan (Muhdar, et al., 2025)

Wereng cokelat merupakan salah satu hama paling merusak karena selain mengisap cairan tanaman, juga menjadi vektor penyebaran virus tungro. Serangannya menyebabkan gejala “hopperburn” atau kekeringan pada tanaman. Pengendalian hama ini biasanya dilakukan dengan varietas tahan wereng dan penggunaan insektisida sistemik. (Syam et al 2021) Hama penggerek batang menyerang bagian dalam batang padi dengan cara meletakkan telur di pelepah daun, lalu larvanya menggerek masuk ke batang. Akibatnya, muncul gejala seperti “beluk” (pada fase vegetatif) dan “patah leher” (pada fase generatif), yang menyebabkan malai tidak muncul atau hampa (Supriyadi et al., 2022).

Ulat grayak (armyworm) menyerang daun tanaman padi secara kolektif, terutama pada malam hari. Hama ini sulit dikendalikan karena serangannya masif dan cepat menyebar. Identifikasi visual sangat penting karena gejalanya bisa mirip dengan serangan hama lain seperti penggulung daun. (Iskandar et al., 2020) Hama seperti walang sangit mengisap cairan bulir padi, sehingga gabah menjadi hampa atau bernas tidak sempurna. Kehadirannya paling merugikan pada fase pengisian bulir. Deteksi awal penting agar insektisida bisa diaplikasikan sebelum kerusakan parah terjadi (Rahmawati et al., 2022).

Gambar 2. 1 Walang Sangit

(BisaTani)

Tikus sawah adalah hama vertebrata yang menyerang padi sejak fase benih hingga panen. Mereka merusak batang muda dan memotong malai padi. Pengendalian tikus dilakukan secara terpadu melalui sanitasi lahan, perangkap, dan pemburuan kolektif (Sudarmaji et al, 2021) Seiring berkembangnya teknologi, identifikasi hama tidak lagi harus dilakukan manual. Pemanfaatan citra digital tanaman padi yang terserang hama dan klasifikasi berbasis machine learning terbukti mampu mempercepat proses identifikasi dan mengurangi kesalahan pengamatan subjektif (Yuliany et al., 2022).

Beberapa penelitian menggunakan pendekatan CNN dan DenseNet untuk mengklasifikasikan jenis hama berdasarkan gambar tubuh hama atau daun yang terserang. Misalnya, model CNN yang dikembangkan oleh Nugroho & Nurraharjo (2023) mampu mengenali 9 jenis hama padi dengan akurasi tinggi setelah dilakukan preprocessing dan augmentasi data. Dengan pendekatan klasifikasi citra, sistem tidak hanya mengenali hama secara otomatis, tetapi juga dapat diintegrasikan dengan informasi penanganan berbasis data. Hal ini sangat mendukung konsep pertanian presisi, di mana petani dapat mengambil keputusan cepat dan tepat berdasarkan hasil identifikasi berbasis teknologi (Asseweth, 2024).

2.4 Solusi Penanganan Hama

Hama merupakan salah satu faktor utama yang menyebabkan penurunan hasil produksi tanaman padi secara signifikan. Pengendalian hama dilakukan dengan berbagai pendekatan, mulai dari pengendalian kimiawi, biologis, hingga kultur teknis, bergantung pada jenis hama dan tingkat serangannya. Pendekatan pengendalian kimiawi dilakukan dengan menggunakan insektisida atau pestisida yang disemprotkan langsung ke area tanaman. Misalnya, insektisida berbahan aktif imidakloprid dan klorpirifos terbukti efektif dalam mengendalikan aphids dan armyworm karena bekerja secara sistemik dan kontak terhadap serangga sasaran. (Muhdar, et al., 2025)

Berikut adalah solusi atau penanganan dari beberapa jenis hama tanaman padi:

Tabel 2. 2 Solusi Penanganan Hama

No	Hama	Solusi Atau Penangan

1	Wereng Coklat (*Nilaparvata lugens*)	Gunakan insektisida yang ditujukan untuk wereng, seperti insektisida berbahan aktif imidacloprid.
2	Hama Ulat Grayak (*Spodoptera litura*)	Gunakan insektisida yang efektif terhadap ulat, seperti insektisida berbahan aktif Bacillus thuringiensis (Bt).
3	Walang Sangit (*Leptocorisa oratorius*)	Penggunaan insektisida kontak saat fase bunting hingga pengisian bulir.
4	Tikus Sawah (*Rattus argentiventer*)	Buat perangkap tikus atau gunakan umpan racun tikus yang aman dan Gunakan predator alami seperti kucing atau Burung Hantu
5	Belalang (*Caelifera)*	Gunakan perangkap jaring. Pengasapan dan aplikasi insektisida kontak.
6	Penggerek Batang Padi (*Scirpophaga incertulas*)	Gunakan insektisida sistemik yang dapat diserap oleh tanaman serta lakukan pemangkasan pada bagian tanaman yang terinfeksi.

2.5 Convolutional Neural Network (CNN)

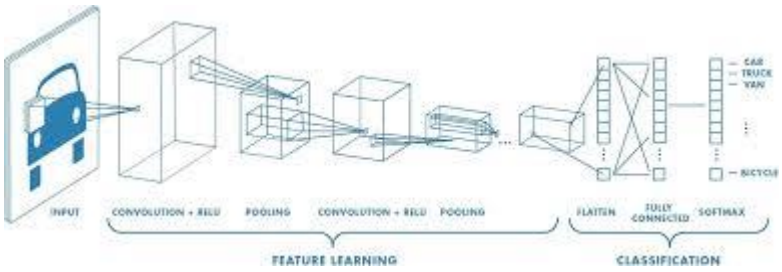
Convolutional Neural Network (CNN) merupakan salah satu jenis deep learning yang paling populer dalam bidang pengolahan citra. CNN dirancang untuk memproses data berbentuk grid, seperti gambar, dan secara otomatis mengekstraksi fitur spasial melalui proses konvolusi. CNN banyak digunakan dalam klasifikasi gambar, deteksi objek, dan pengenalan pola visual. CNN terdiri dari beberapa lapisan utama, yaitu convolution layer, pooling layer, dan fully connected layer. Lapisan konvolusi bertugas mengekstrak fitur lokal dari citra melalui penggunaan filter (kernel), sedangkan pooling layer berfungsi untuk mereduksi dimensi data dan menjaga fitur yang paling penting. (Rahmat et al, 2022)

Proses pelatihan CNN dilakukan dengan algoritma backpropagation dan optimasi menggunakan metode seperti stochastic gradient descent (SGD) atau Adam. CNN mampu belajar langsung dari data mentah tanpa perlu ekstraksi fitur secara manual, sehingga sangat efisien untuk tugas klasifikasi citra. Keunggulan CNN dibanding metode klasifikasi konvensional adalah kemampuannya dalam mengenali pola kompleks dan mendalam melalui penggunaan banyak layer.

Semakin banyak layer, maka semakin baik model dalam memahami fitur tingkat tinggi dari gambar, seperti bentuk objek atau tekstur spesifik. (Setyawan et al., 2021)

CNN juga memiliki kemampuan translasi invariant, yaitu kemampuan untuk mengenali objek meskipun posisi atau ukurannya berubah. Hal ini memungkinkan model tetap mampu mengklasifikasikan gambar dengan berbagai variasi, termasuk rotasi, pencahayaan, dan skala. Dalam pengembangan CNN, dropout dan batch normalization sering digunakan sebagai teknik regulasi untuk mencegah overfitting. Dropout akan menonaktifkan sebagian neuron secara acak selama pelatihan, sedangkan batch normalization membantu mempercepat konvergensi model. (Hidayat et al, 2020)

Dalam konteks klasifikasi hama tanaman padi, CNN digunakan untuk mengenali pola visual dari tubuh hama atau gejala yang ditimbulkan pada daun. Proses ini dilakukan dengan menginput gambar ke CNN, yang kemudian diklasifikasikan ke dalam kelas-kelas hama tertentu berdasarkan output probabilistic. (Ananda et al, 2022) CNN juga dapat diintegrasikan dengan antarmuka pengguna seperti aplikasi web, sistem IoT, atau perangkat Android, untuk menghasilkan sistem klasifikasi real-time berbasis gambar. Hal ini menjadikan CNN sangat relevan dalam pengembangan sistem cerdas berbasis citra digital di bidang pertanian presisi (Putra et al, 2023)



Gambar 2. 2 Arsitektur CNN

(Susi Yuliany et al, 2022)

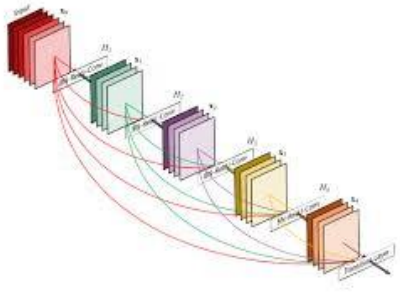
2.6 DenseNet

DenseNet merupakan salah satu arsitektur Convolutional Neural Network (CNN) yang diperkenalkan untuk mengatasi permasalahan vanishing gradient,

mempercepat pelatihan, dan meningkatkan akurasi klasifikasi. DenseNet menghubungkan setiap layer ke semua layer sebelumnya, bukan hanya ke layer di depannya saja. Dalam arsitektur DenseNet, setiap layer menerima input dari seluruh layer sebelumnya dan meneruskan fitur yang telah diekstraksi ke semua layer setelahnya. Konsep ini disebut dense connectivity, yang bertujuan untuk memaksimalkan penggunaan fitur dan mengurangi parameter (Fadillah et al., 2022).

Kelebihan DenseNet dibanding arsitektur CNN biasa adalah efisiensi dalam penggunaan parameter serta kemampuan untuk mencegah hilangnya informasi penting karena setiap layer memiliki akses langsung ke gradien dari loss function. DenseNet juga memiliki kemampuan untuk meningkatkan feature reuse, yaitu fitur dari layer sebelumnya tidak hilang, melainkan digunakan kembali oleh layer berikutnya. Hal ini menyebabkan proses pembelajaran menjadi lebih efisien dan akurat (Sari et al, 2020).

Dalam konteks klasifikasi gambar, DenseNet sering digunakan untuk mengenali objek-objek dengan ciri visual yang kompleks, termasuk dalam bidang medis, pertanian, maupun industri manufaktur karena kemampuannya membangun representasi fitur yang mendalam. Dalam penerapannya untuk klasifikasi hama tanaman padi, DenseNet mampu menangkap detail tekstur tubuh hama yang sering kali kompleks dan halus. Hal ini memberikan keunggulan dalam akurasi klasifikasi terhadap gambar resolusi rendah sekalipun (Agustina et al., 2022).



Gambar 2. 3 Arsitektur DenseNet

(Susi Yuliany et al, 2022)

2.7 Integrasi Model dalam Aplikasi

Integrasi model machine learning ke dalam aplikasi merupakan tahapan penting dalam menerapkan hasil pelatihan model ke dalam sistem nyata yang dapat digunakan oleh pengguna akhir. Proses ini mencakup pemanggilan model terlatih (file .h5 pada TensorFlow/Keras), pengolahan input (seperti gambar dari pengguna), serta penyajian hasil prediksi dalam bentuk antarmuka aplikasi. Tujuan dari integrasi ini adalah untuk meningkatkan aksesibilitas dan efektivitas model dalam menyelesaikan masalah dunia nyata (Utami et al, 2021).

Dalam pengembangan aplikasi, model biasanya diintegrasikan menggunakan backend framework seperti Flask untuk aplikasi web, atau menggunakan Streamlit yang lebih sederhana namun cukup kuat untuk prototipe dan aplikasi berbasis UI visual. Proses integrasi ini membutuhkan tahapan konversi data input ke format yang dapat dipahami model, pemanggilan prediksi model, serta pengolahan hasil keluaran ke dalam bentuk yang informatif (Maulana et al., 2020).

Keberhasilan integrasi juga ditentukan oleh kesesuaian antara arsitektur model dan kebutuhan aplikasi. Model DenseNet misalnya, dapat memberikan akurasi tinggi dalam klasifikasi gambar namun memerlukan sumber daya komputasi yang lebih tinggi dibandingkan model konvensional, sehingga penting mempertimbangkan trade-off antara akurasi dan efisiensi saat proses integrasi. (Setiawan , 2023) Akhirnya, dalam integrasi ke dalam aplikasi publik, perlu diperhatikan juga aspek user experience (UX) dan keamanan sistem, terutama dalam menangani data input dari pengguna. Sistem harus mampu menangani input tidak valid, menjaga performa aplikasi tetap stabil, serta memberikan informasi hasil klasifikasi yang jelas dan dapat ditindaklanjuti oleh pengguna (Nurhayati et al, 2019).

2.8 Evaluasi Model

Evaluasi model dalam machine learning merupakan tahap penting yang digunakan untuk mengukur performa dan keandalan model terhadap data yang belum pernah dilihat sebelumnya. Tujuannya adalah untuk mengetahui sejauh mana model mampu melakukan generalisasi terhadap data baru dan bukan hanya menghafal data pelatihan. Metode evaluasi umumnya melibatkan pengukuran

akurasi, presisi, recall, dan F1-score. (Putra , 2021) Akurasi adalah metrik paling umum digunakan, yaitu proporsi prediksi yang benar terhadap total prediksi. Namun, pada data yang tidak seimbang (imbalanced dataset), akurasi bisa menyesatkan karena model cenderung memprediksi kelas mayoritas. Oleh karena itu, digunakan metrik tambahan seperti precision (ketepatan) dan recall (kemampuan model dalam menemukan semua data relevan). (Sari , 2020)

Precision dihitung sebagai rasio antara true positive dan jumlah prediksi positif. Precision penting ketika biaya kesalahan pada prediksi positif cukup tinggi, seperti pada klasifikasi hama berbahaya. Sedangkan recall dihitung sebagai rasio antara true positive dan jumlah total aktual positif. Recall penting ketika kita tidak boleh melewatkan deteksi terhadap kelas tertentu. (Yuliani et al., 2022) Selain metrik-metrik tersebut, evaluasi model juga sering dilakukan melalui confusion matrix, yaitu tabel yang menampilkan perbandingan antara prediksi model dan nilai aktual untuk setiap kelas. Ini membantu melihat dengan jelas kelas mana yang sering salah diklasifikasikan, serta mengetahui pola kesalahan model (Wahyuni et al, 2019)

2.9 Kerangka Berpikir

BAB III
PERANCANGAN SISTEM & IMPLEMENTASI

3.1 Metode Penelitian

3.1.1 Pendekatan Penelitian

Metode penelitian yang digunakan adalah metode kuantitatif eksperimen. Peneliti membangun dan melatih model machine learning menggunakan arsitektur DenseNet121 untuk mengklasifikasikan enam jenis hama pada tanaman padi. Setiap eksperimen dilakukan untuk menguji efektivitas arsitektur tersebut dalam mengenali gambar hama. Metode ini dipilih karena memungkinkan peneliti melakukan pelatihan model klasifikasi secara terstruktur dan terukur berdasarkan dasaset hama tanaman padi.

Tahapan eksperimen mencakup pengumpulan dataset, preprocessing data, pelatihan model, evaluasi model, dan integrasi hasil ke dalam aplikasi berbasis web. Hasil eksperimen kemudian dianalisis menggunakan metrik evaluasi seperti akurasi, precision, recall, dan F1-score untuk mengetahui performa klasifikasi terhadap data uji yang telah dipisahkan sebelumnya. Model juga diuji data baru untuk mengamati konsistensi klasifikasi dalam kondisi riil.

3.1.2 Jenis Penelitian

Penelitian ini merupakan penelitian terapan yang bertujuan mengimplementasikan teknologi kecerdasan buatan dalam bidang pertanian, khususnya klasifikasi citra hama tanaman padi. Tujuan utama dari penelitian ini adalah menghasilkan sistem klasifikasi yang dapat mengenali jenis hama berdasarkan citra secara otomatis dan akurat menggunakan algoritma Deep Learning. Penelitian ini mengadopsi pendekatan kuantitatif karena melibatkan pengukuran numerik terhadap performa model klasifikasi.

Selain itu, penelitian ini bersifat eksperimental, karena melibatkan proses pelatihan dan pengujian model klasifikasi menggunakan data yang telah ditentukan. Dalam eksperimen ini, dilakukan beberapa pengujian terhadap model Deep

Learning, seperti DenseNet, untuk menilai performanya dalam mengklasifikasikan sembilan jenis hama tanaman padi. Perbandingan model atau parameter juga dapat dilakukan untuk menentukan konfigurasi terbaik dalam klasifikasi.

Penelitian ini juga menggunakan pendekatan desain rekayasa sistem. Artinya, peneliti tidak hanya berhenti pada analisis data, tetapi juga merancang sistem klasifikasi berbasis web atau aplikasi, sehingga dapat digunakan secara praktis oleh petani atau tenaga penyuluh. Pendekatan ini menggabungkan aspek teoritis dan praktis dalam bidang pertanian digital.

Secara keseluruhan, jenis penelitian ini bersifat interdisipliner, menggabungkan ilmu komputer, pertanian, dan teknik sistem informasi. Dengan demikian, hasil penelitian diharapkan mampu memberikan solusi nyata untuk mendeteksi hama tanaman secara lebih cepat dan akurat dibandingkan metode manual.

3.1.3 Populasi & Sampel Penelitian

Populasi dalam penelitian ini adalah seluruh gambar hama tanaman padi yang mencakup berbagai jenis hama yang umum ditemukan di wilayah pertanian Indonesia, seperti belalang, tikus, wereng coklat, walang sangit, penggerek batang dan ulat grayak. Populasi ini dapat diperoleh dari berbagai sumber, baik dari observasi langsung di lapangan, dataset publik, maupun publikasi penelitian terdahulu. Populasi ini sangat luas, karena hama dapat muncul dalam berbagai kondisi dan bentuk yang berbeda.

Sampel yang digunakan dalam penelitian ini adalah gambar digital dari enam jenis hama tanaman padi. Pengambilan sampel dilakukan secara purposive sampling, yaitu memilih gambar yang memenuhi kriteria tertentu, seperti resolusi tinggi, pencahayaan cukup, serta fokus objek yang jelas. Pemilihan dilakukan untuk memastikan bahwa model dilatih dengan gambar yang representatif dan informatif.

Ukuran sampel dalam penelitian ini yaitu 200 gambar per kelas. Beberapa teknik augmentasi gambar juga diterapkan untuk memperbanyak sampel secara sintetis, seperti rotasi, flip horizontal, atau penyesuaian brightness agar model lebih tahan terhadap variasi input. Dengan teknik sampling dan augmentasi yang tepat, diharapkan model dapat belajar secara efektif dari data dan memberikan akurasi tinggi pada saat klasifikasi. Penggunaan sampel yang berkualitas menjadi kunci keberhasilan dalam membangun sistem klasifikasi citra hama tanaman.

3.1.4 Sumber Data

Data yang digunakan dalam penelitian ini bersumber dari data primer dan data sekunder. Data primer diperoleh melalui dokumentasi langsung dari lapangan, seperti lahan pertanian, sawah, atau kebun percobaan, menggunakan kamera digital. Foto-foto ini kemudian dikategorikan berdasarkan jenis hama dengan bantuan ahli pertanian.

Sementara itu, data sekunder diperoleh dari berbagai dataset terbuka yang tersedia di internet, seperti dari situs Kaggle, dan GitHub. Dataset tersebut biasanya sudah disusun dalam bentuk terstruktur dan dilabeli dengan baik, sehingga dapat langsung digunakan untuk pelatihan model setelah proses normalisasi dan preprocessing dilakukan. Kedua sumber data ini dikombinasikan untuk menciptakan dataset yang komprehensif, mencakup berbagai kondisi pencahayaan, sudut pandang, dan ukuran objek. Dengan demikian, model dapat mengenali hama dengan tingkat keakuratan yang lebih tinggi pada situasi dunia nyata.

Sebelum digunakan dalam proses pelatihan, semua data gambar melewati proses validasi untuk memastikan tidak ada duplikasi, kesalahan label, atau kualitas gambar yang buruk. Langkah ini penting untuk menjaga integritas data agar tidak menyebabkan overfitting atau kesalahan prediksi pada model akhir.

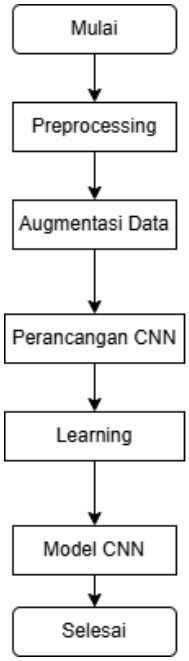
3.1.5 Metode Pengumpulan Data

Pengumpulan data primer dilakukan dengan metode observasi langsung, yaitu dengan mendatangi lahan pertanian dan mengambil gambar hama yang ditemukan pada tanaman padi. Dokumentasi dilakukan secara sistematis dengan mencatat waktu, lokasi, dan kondisi lingkungan saat pengambilan gambar, agar dapat dianalisis lebih lanjut.

Untuk data sekunder, metode yang digunakan adalah penelusuran pustaka dan eksplorasi dataset daring. Beberapa sumber dataset seperti “Pest Images” di Kaggle dimanfaatkan karena sudah memiliki anotasi label yang jelas. Selain itu, jurnal-jurnal ilmiah juga dijadikan rujukan untuk mengetahui klasifikasi visual hama yang umum digunakan dalam penelitian sebelumnya.

Setelah data terkumpul, dilakukan proses preprocessing, seperti resize ke ukuran 200x200 piksel, konversi ke array numerik, serta normalisasi. Beberapa gambar juga diberikan proses augmentasi seperti rotasi, zoom, dan flip horizontal untuk meningkatkan keragaman data. Proses pengumpulan data ini disesuaikan dengan kebutuhan model klasifikasi, yaitu untuk memastikan model mendapatkan input data yang konsisten dan berkualitas tinggi. Proses ini sangat krusial karena kualitas data akan sangat menentukan performa akhir dari sistem klasifikasi yang dibangun.

3.2 Perancangan Blok Diagram



Gambar 3. 1 Flowchat Sistem

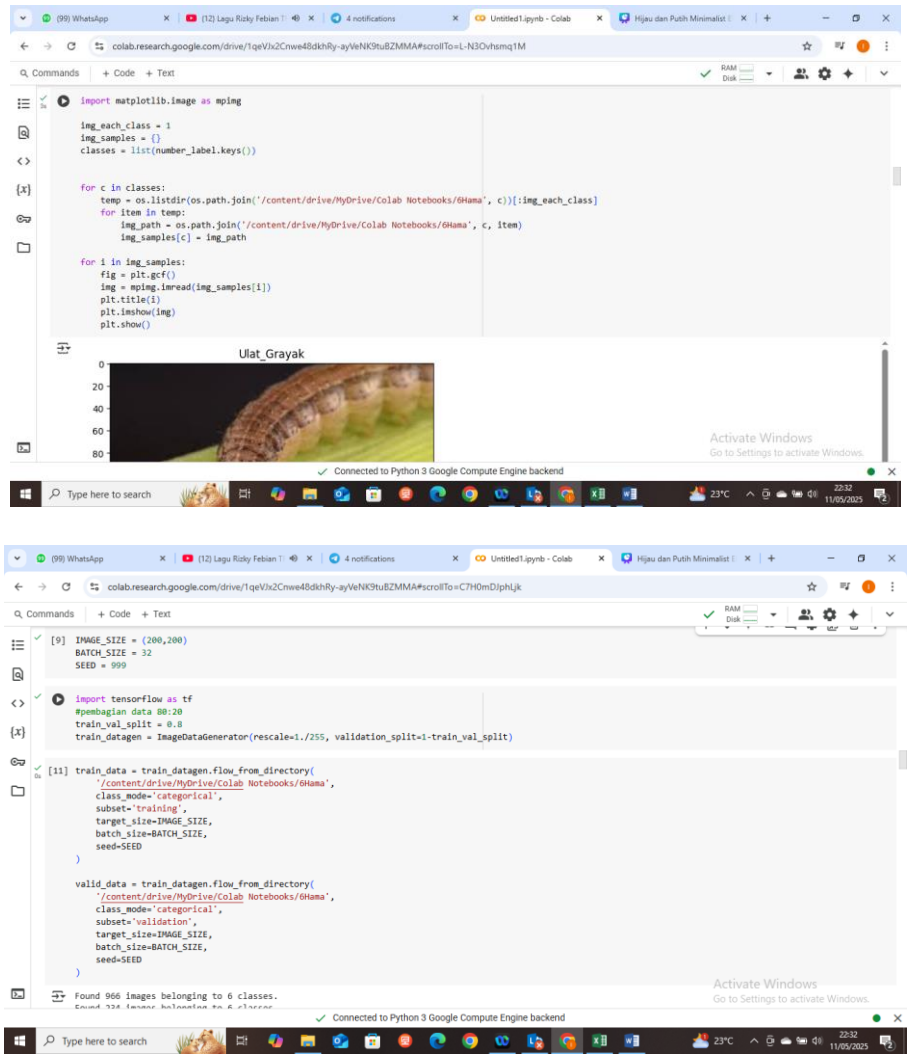
1. Mulai

Penelitian ini dimulai dengan inisiasi proyek. Pada tahap ini, dilakukan penetapan tujuan analisis, termasuk menentukan apakah proses bertujuan untuk mendeteksi akurasi, presisi, dan recall. Di sinilah pengguna atau sistem menyiapkan data mentah berupa gambar-gambar hama tanaman padi yang akan digunakan sebagai dataset. Tahap ini penting karena keberhasilan semua proses berikutnya bergantung pada kualitas awal data dan kesiapan lingkungan sistem.

2. Preprocessing

Tahap preprocessing adalah fondasi awal dalam pemrosesan citra digital sebelum data dilatih dalam jaringan saraf. Dalam konteks klasifikasi hama tanaman padi, preprocessing bertujuan untuk memastikan bahwa seluruh gambar yang digunakan memiliki format dan ukuran yang seragam serta berkualitas baik. Langkah pertama dalam preprocessing adalah resizing gambar, di mana semua gambar diubah ukurannya menjadi dimensi tetap, seperti 200x200 piksel. Hal ini penting karena model CNN hanya menerima input dengan dimensi tetap agar

struktur jaringan bisa berfungsi secara optimal. Selanjutnya, dilakukan normalisasi nilai piksel, yakni mengubah skala piksel dari 0–255 menjadi 0–1. Tujuannya adalah mempercepat proses pelatihan model dan meningkatkan konvergensi selama training karena nilai yang lebih kecil membuat perhitungan gradien lebih stabil.



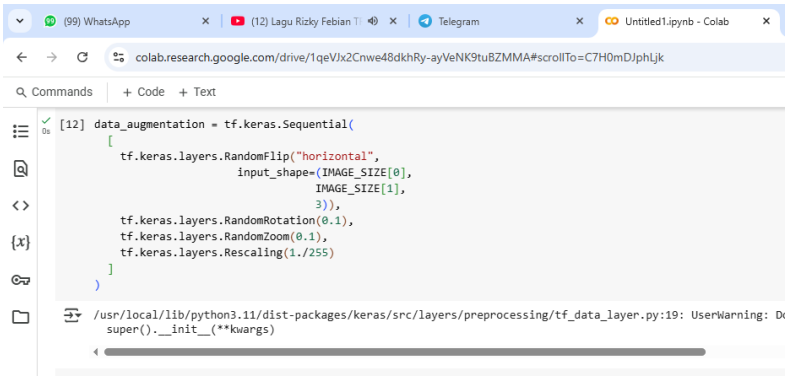
Gambar 3. 2 Sourcecode Preprocessing

Selain itu, dilakukan pembersihan data, yakni proses untuk menghapus data citra yang tidak relevan, seperti gambar buram, overexposed, atau bahkan duplikat data yang bisa menyebabkan model belajar secara tidak akurat. Proses ini juga meliputi pengecekan apakah setiap gambar sudah sesuai dengan label kelasnya.

Misalnya, gambar yang seharusnya diklasifikasikan sebagai "belalang" namun menampilkan serangga lain perlu disingkirkan untuk menjaga kualitas dataset. Tahap ini krusial karena data yang bersih dan seragam akan sangat memengaruhi hasil akhir dari pelatihan model, terutama dalam konteks klasifikasi citra yang sangat bergantung pada kualitas visual inputnya.

3. Augmentasi Data

Augmentasi data merupakan proses penting dalam memperkaya variasi dataset yang terbatas. Dalam proyek klasifikasi hama tanaman padi, jumlah data untuk masing-masing kelas bisa jadi tidak merata atau terlalu sedikit, sehingga diperlukan augmentasi untuk menciptakan variasi citra buatan yang tetap merepresentasikan kelas aslinya. Salah satu teknik augmentasi yang paling umum adalah rotasi gambar, di mana gambar diputar dengan sudut tertentu, seperti 15 atau 30 derajat, untuk meniru kondisi nyata saat gambar diambil dari sudut yang berbeda. Teknik lain yang digunakan adalah flip horizontal dan vertikal, yaitu membalik gambar ke kanan-kiri atau atas-bawah, yang sangat efektif untuk meniru tampilan hama dari berbagai orientasi.



Gambar 3. 3 Sourcode Data Augmentasi

Selain itu, teknik zooming (perbesaran) dan shifting (pergeseran) citra digunakan untuk membantu model belajar mengenali objek dengan ukuran dan posisi yang bervariasi. Augmentasi juga dapat mencakup perubahan kontras, pencahayaan, hingga distorsi perspektif. Proses ini dilakukan secara otomatis saat

pelatihan menggunakan fungsi generator gambar dari pustaka seperti ImageDataGenerator milik Keras. Augmentasi sangat penting dalam membangun model CNN yang robust, yaitu model yang tidak hanya belajar mengenali pola dari contoh yang sama, tetapi juga dari variasi bentuk dan kondisi lingkungan sekitar objek. Ini membuat model lebih siap menghadapi gambar yang berbeda saat pengujian atau implementasi di lapangan.

4. Perancangan CNN

Perancangan arsitektur CNN (Convolutional Neural Network) menjadi jantung dari sistem klasifikasi. CNN dirancang untuk mengenali pola visual dalam gambar melalui lapisan-lapisan yang meniru cara kerja sistem visual manusia. Pada tahap awal, citra yang telah melalui preprocessing akan diproses oleh lapisan konvolusi (Convolutional Layer). Di sini, filter kecil (kernel) digunakan untuk mengekstraksi fitur penting dari gambar, seperti tepi, warna, atau tekstur tubuh hama. Setiap filter akan menghasilkan feature map, yaitu representasi spasial dari fitur yang dikenali. Kemudian hasil dari convolusi dilewatkan ke fungsi aktivasi ReLU (Rectified Linear Unit) untuk memperkenalkan non-linearitas dalam jaringan sehingga model mampu mengenali hubungan kompleks antar fitur.

Setelah itu, hasil feature map dikompresi menggunakan Pooling Layer, biasanya MaxPooling, yang mengambil nilai maksimum dari area tertentu di feature map. Ini dilakukan untuk mengurangi dimensi data (downsampling) dan menghindari overfitting tanpa kehilangan informasi penting. Beberapa kombinasi dari Convolutional + Pooling layer dilakukan secara bertahap agar model semakin memahami pola yang kompleks. Pada bagian akhir jaringan, digunakan Fully Connected Layer (Dense) yang menggabungkan seluruh fitur hasil ekstraksi menjadi satu vektor, dan akhirnya diproses oleh layer output yang menggunakan fungsi aktivasi softmax. Fungsi ini mengubah output menjadi nilai probabilitas dari setiap kelas hama, sehingga dapat ditentukan kelas mana yang paling mungkin berdasarkan input gambar..

5. Learning

Setelah arsitektur model CNN selesai, tahap selanjutnya adalah pelatihan (training) dan pembelajaran (learning). Dataset dibagi menjadi dua bagian: data pelatihan (training data) dan data validasi (validation data), misalnya dengan rasio 80:20. Data pelatihan digunakan untuk melatih model, sementara data validasi digunakan untuk menguji kinerja model pada data yang belum pernah dilihat sebelumnya. Pada tahap ini, model mencoba menyesuaikan bobotnya secara bertahap melalui proses forward propagation dan backpropagation, di mana model menghitung kesalahan (loss) dan memperbarui bobotnya untuk memperbaiki prediksi. Fungsi loss yang umum digunakan adalah categorical crossentropy untuk klasifikasi multi-kelas.

Proses training dilakukan dalam sejumlah epoch, yaitu berapa kali seluruh data pelatihan digunakan untuk melatih model. Biasanya 25–50 epoch digunakan tergantung dari ukuran dataset dan kompleksitas model. Parameter lain seperti batch size (jumlah data yang diproses sekaligus) juga memengaruhi hasil pelatihan. Optimizer seperti Adam digunakan untuk mempercepat konvergensi model dalam menemukan nilai bobot terbaik. Selama training, akurasi dan nilai loss dari data pelatihan dan validasi dicatat untuk memantau performa model. Bila akurasi meningkat dan loss menurun, maka model berada pada arah pelatihan yang benar. Jika tidak, bisa dilakukan teknik seperti early stopping atau penyesuaian arsitektur/model untuk menghindari overfitting.

6. Model CNN

Setelah proses training selesai dan model mencapai performa terbaiknya, model akan disimpan dalam format .h5 (HDF5) yang memuat struktur jaringan dan bobotnya. Model ini dapat langsung digunakan untuk proses inferensi, yaitu memprediksi kelas hama dari gambar baru yang tidak dikenal sebelumnya. Sebelum implementasi, dilakukan pengujian dengan data uji untuk melihat bagaimana model berperforma di dunia nyata. Bila hasil pengujian memuaskan, model dapat

dilanjutkan ke tahap deployment atau integrasi ke dalam aplikasi, misalnya web-based app menggunakan Streamlit atau Flask, di mana pengguna cukup mengunggah gambar hama dan sistem akan menampilkan hasil klasifikasi serta solusi penanganannya.

Implementasi model ke dalam aplikasi akan sangat mempermudah petani atau penyuluh pertanian dalam mengenali jenis hama secara cepat dan tepat. Model juga bisa dilengkapi dengan fitur tambahan seperti visualisasi probabilitas, penjelasan singkat mengenai hama yang terdeteksi, serta solusi penanganan yang direkomendasikan. Pada tahap ini, dilakukan juga dokumentasi model, termasuk evaluasi performa menggunakan metrik seperti akurasi, precision, recall, dan confusion matrix, agar pengguna tahu seberapa baik model bekerja. Dengan demikian, model CNN yang telah dilatih tidak hanya berhenti pada fase akademik, tetapi benar-benar digunakan secara praktis dalam dunia pertanian.

7. Selesai

Pada tahapan akhir sistem sudah memiliki model yang siap digunakan dan bisa diintegrasikan ke dalam platform seperti web, aplikasi mobile, atau sistem IoT. Model ini bisa digunakan berkali-kali untuk mendeteksi hama secara otomatis dari gambar baru yang dimasukkan ke dalam sistem.

BAB IV
HASIL & PEMBAHASAN

4.1 Implementasi Model Klasifikasi Hama Tanaman Padi

Implementasi model klasifikasi hama tanaman padi pada penelitian ini dilakukan dengan memanfaatkan arsitektur Convolutional Neural Network (CNN) yang dikembangkan berdasarkan model DenseNet121. Model ini dipilih karena kemampuannya dalam mengekstraksi fitur citra yang kompleks melalui koneksi langsung dari setiap layer ke semua layer berikutnya, sehingga informasi dari fitur tidak mudah hilang saat propagasi ke layer yang lebih dalam. Proses implementasi mencakup beberapa tahap utama, yaitu preprocessing data, augmentasi data, pelatihan model (training), dan pengujian model menggunakan data validasi atau data baru.

Pada tahap awal, data citra yang digunakan merupakan kumpulan gambar hama tanaman padi dari berbagai kelas, seperti Ulat grayak, Belalang, Walang sangit, Wereng coklat, Penggerek batang, Tikus sawah. Setiap citra memiliki resolusi berbeda, sehingga perlu dilakukan normalisasi ukuran (resizing) agar sesuai dengan input layer model DenseNet, yaitu 200x200 piksel. Gambar yang tidak berwarna juga dikonversi ke format RGB untuk keseragaman. Selanjutnya, dilakukan proses augmentasi data seperti rotasi, flipping horizontal, zoom, dan brightness adjustment untuk meningkatkan keragaman data dan mengurangi risiko overfitting. Hal ini sangat penting mengingat banyak gambar yang berasal dari sumber serupa dan berpotensi membuat model bias terhadap pola tertentu.

Setelah preprocessing dan augmentasi selesai, data dibagi menjadi data latih (training) dan data validasi (validation).

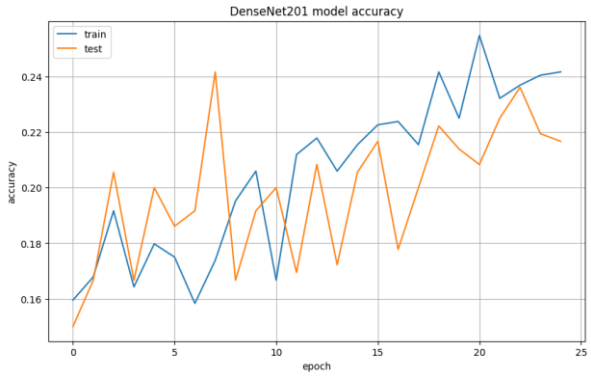
Tabel 4. 1 Pembagian Data

Epoch		Pembagian Data	
		70%:30%	80%:20%
25	Training Acc	0.2242	0.2352
	Training Loss	17.332	17.366
	Validation Acc	0.2417	0.1752
	Validation Loss	17.704	17.967
	Time	222	217
50	Training Acc	0.2452	0.2690
	Training Loss	17.187	16.932
	Validation Acc	0.2639	0.1923
	Validation Loss	17.313	17.899
	Time	264	220
75	Training Acc	0.3160	0.3067
	Training Loss	16.430	16.612
	Validation Acc	0.2472	0.1923
	Validation Loss	17.053	17.596
	Time	224	220
100	Training Acc	0.3510	0.3070
	Training Loss	16.085	16.195
	Validation Acc	0.3083	0.2821
	Validation Loss	16.684	17.417
	Time	225	223

Pada skenario pertama digunakan rasio 70% untuk pelatihan dan 30% untuk validasi. Model kemudian dilatih menggunakan data latih selama beberapa epoch 25 epoch dengan menggunakan optimizer Adam dan fungsi loss categorical_crossentropy, karena masalah ini merupakan klasifikasi multi-kelas. Proses pelatihan dilakukan di Google Colab.

Hasil pelatihan menunjukkan peningkatan akurasi model secara bertahap pada data latih dan validasi. Misalnya, pada epoch ke-10, akurasi validasi mencapai 0.17, dan meningkat menjadi 0.241 pada epoch ke-25. Ini menunjukkan bahwa model mampu mengenali pola-pola visual dari berbagai jenis hama dengan cukup

baik. Grafik training accuracy dan validation accuracy ditampilkan untuk memvisualisasikan proses konvergensi model.



Gambar 4. 1 Grafik Train & Valid Acc

Setelah model selesai dilatih, dilakukan proses implementasi pengujian secara real-time terhadap gambar hama baru yang diunggah oleh pengguna. Implementasi ini menggunakan antarmuka berbasis web atau melalui Google Colab, dengan mekanisme unggah gambar dan prediksi label hama. Sebagai contoh, ketika pengguna mengunggah gambar berformat .jpg yang menampilkan serangga wereng coklat, sistem akan memproses gambar tersebut, mengubah ukurannya menjadi 200x200 piksel, dan menjalankannya ke dalam model untuk prediksi. Model kemudian mengembalikan vektor probabilitas untuk setiap kelas, dan memilih kelas dengan probabilitas tertinggi sebagai hasil klasifikasi.

Untuk setiap hasil klasifikasi, sistem juga menampilkan probabilitas dalam bentuk persentase agar pengguna dapat mengetahui tingkat keyakinan model terhadap prediksi tersebut. Sebagai contoh:

Gambar 4. 2 Hasil Klasifikasi Hama

Klasifikasi: Penggerek Batang (Probabilitas: 97%)

Ini berarti bahwa model yakin sebesar 97% bahwa gambar tersebut merupakan gambar Penggerek Batang. Hasil implementasi lainnya menunjukkan bahwa beberapa kelas hama seperti belalang dan tikus sawah cenderung memiliki akurasi prediksi yang lebih tinggi, karena tekstur dan bentuk tubuhnya cukup unik dan mudah dikenali oleh model. Namun, untuk kelas seperti ulat grayak dan penggerek batang, prediksi terkadang keliru akibat kemiripan visual, terutama jika latar belakang gambar tidak bersih atau pencahayaannya buruk.

Sebagai bagian dari antarmuka yang dikembangkan, pengguna hanya perlu mengunggah gambar hama dan menunggu hasil klasifikasi tampil di layar bersama dengan informasi solusi yang sesuai. Implementasi ini memberikan kemudahan bagi pengguna, khususnya petani atau praktisi pertanian, untuk mendeteksi jenis hama secara otomatis tanpa memerlukan pengetahuan teknis mendalam tentang jenis-jenis hama.

Dengan demikian, implementasi model DenseNet dalam penelitian ini membuktikan bahwa pendekatan deep learning sangat efektif dalam menangani permasalahan klasifikasi visual di bidang pertanian, khususnya dalam identifikasi

jenis hama tanaman padi. Sistem ini dapat digunakan sebagai alat bantu awal dalam proses monitoring dan pengambilan keputusan terkait penanganan hama yang menyerang tanaman.

4.2 Pengaruh Augmentasi

Dalam proses pelatihan model klasifikasi citra hama tanaman padi, salah satu faktor penting yang diuji adalah penggunaan augmentasi data. Teknik ini diterapkan untuk menambah variasi citra pelatihan dengan melakukan transformasi tertentu seperti rotasi, zooming, flipping horizontal, shifting, dan brightness adjustment. Tujuan dari augmentasi adalah untuk meningkatkan generalisasi model dan mengurangi risiko overfitting, terutama ketika dataset asli terbatas jumlahnya atau tidak merata antara kelas satu dan lainnya.

Augmentasi data dilakukan menggunakan library *ImageDataGenerator* dari Keras. Contoh augmentasi yang digunakan mencakup rotasi gambar hingga 36 derajat, zoom range 0.1, dan horizontal flip. Dengan augmentasi ini, satu gambar asli dapat menghasilkan banyak varian baru yang tetap relevan secara visual namun berbeda dalam orientasi maupun ukuran. Sebagai contoh, Gambar 4.2 berikut menunjukkan perbandingan antara gambar asli dan hasil augmentasi dari kelas hama Wereng Coklat. Terlihat bahwa gambar hasil augmentasi tetap mempertahankan struktur visual utama dari objek, namun mengalami perubahan posisi dan bentuk secara minor:

Gambar 4. 3 Contoh Augmentasi Gambar dari Kelas Penggerek Batang

Hasil eksperimen menunjukkan bahwa model yang dilatih dengan dataset yang telah diaugmentasi memiliki akurasi validasi lebih tinggi dibandingkan dengan model yang dilatih pada data mentah tanpa augmentasi. Selain itu, grafik

loss validasi pada model dengan augmentasi juga menunjukkan penurunan yang lebih stabil, mengindikasikan bahwa model lebih tahan terhadap overfitting. Ini menguatkan temuan dalam studi-studi sebelumnya bahwa augmentasi data memainkan peran penting dalam meningkatkan ketahanan model terhadap data yang bervariasi di dunia nyata (Fadillah, 2022; Kurniawan, 2022).

Dengan demikian, penggunaan augmentasi data dapat dianggap sebagai strategi penting dalam pelatihan model klasifikasi citra, terutama ketika bekerja dengan dataset yang terbatas. Hal ini juga membuktikan bahwa model DenseNet mampu memanfaatkan variasi visual tambahan dengan baik untuk meningkatkan performa prediksi.

4.3 Perbandingan Kinerja Model DenseNet, CNN biasa dan VGG16

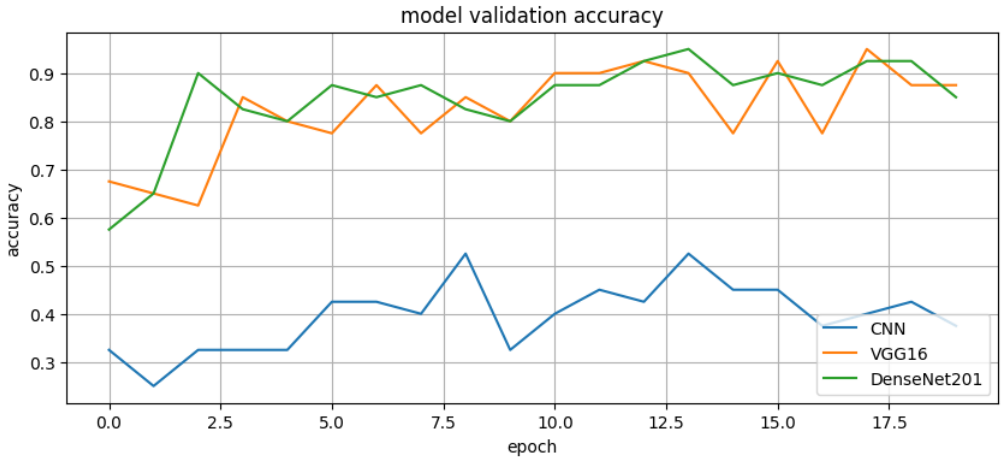
Sebagai bagian dari evaluasi model klasifikasi citra hama tanaman padi, dilakukan perbandingan performa antara model DenseNet yang menjadi model utama dalam penelitian ini, dengan model CNN biasa dan VGG16 sebagai pembanding. Ketiga model ini diuji pada dataset yang sama, dengan proporsi data pelatihan dan validasi sebesar 80%:20%.

Model VGG16 merupakan salah satu arsitektur convolutional neural network (CNN) yang populer, yang terdiri dari susunan lapisan konvolusi dan pooling secara berurutan tanpa koneksi langsung seperti pada DenseNet. Arsitektur ini memiliki keunggulan pada kesederhanaannya dan telah banyak digunakan dalam berbagai tugas klasifikasi gambar. Namun, dibandingkan dengan DenseNet, VGG16 memiliki jumlah parameter yang jauh lebih besar, sehingga memerlukan waktu pelatihan yang lebih lama dan sumber daya komputasi yang lebih tinggi.

Hasil pengujian menunjukkan bahwa model DenseNet menghasilkan akurasi yang lebih stabil dibandingkan VGG16 dan CNN biasa. Selain itu, nilai F1-score, presisi, dan recall dari DenseNet juga lebih baik secara konsisten. Hal ini menunjukkan bahwa konektivitas langsung antar layer pada DenseNet

memungkinkan fitur dipelajari secara lebih mendalam dan efisien, dibandingkan arsitektur berlapis standar seperti pada VGG16.

Secara keseluruhan, DenseNet lebih unggul dalam hal akurasi dan efisiensi, terutama pada dataset dengan ukuran relatif kecil seperti pada penelitian ini. VGG16, meskipun cukup baik, menunjukkan kecenderungan overfitting yang lebih tinggi dan membutuhkan waktu pelatihan yang lebih lama. Berdasarkan hasil ini, DenseNet tetap dipilih sebagai model utama dalam implementasi sistem klasifikasi hama tanaman padi.



Gambar 4. 4 Perbandingan CNN, VGG dan DenseNet201

4.4 Evaluasi Model

Evaluasi model klasifikasi sangat penting untuk mengetahui sejauh mana model yang dibangun mampu mengidentifikasi jenis hama tanaman padi dengan benar. Dalam penelitian ini, evaluasi dilakukan dengan mengukur beberapa metrik utama seperti akurasi (accuracy), presisi (precision), recall, dan F1-score. Metrik-metrik ini digunakan karena klasifikasi multi-kelas membutuhkan lebih dari sekadar pengukuran akurasi global, terutama saat ada kemungkinan data tidak seimbang di tiap kelas hama.

4.4.1 Akurasi

Akurasi diukur berdasarkan pembagian data 70:30 dan 80:20. Dari 180 gambar hama yang diuji, model berhasil mengklasifikasikan 115 gambar dengan benar, maka akurasinya adalah 66,6%. Namun, akurasi saja tidak cukup untuk menggambarkan performa model secara keseluruhan, karena pada data dengan distribusi kelas yang tidak seimbang, model bisa jadi hanya memprediksi kelas dominan dengan benar dan tetap memiliki akurasi tinggi.

Tabel 4. 2 Tabel Akurasi 100 Epoch

Pembagian Data	Epoch	Kelas	Hasil Klasifikasi		Akurasi
			Benar	Salah	
70%:30%	100	Belalang	9	6	64%
		Penggerek_Batang	12	3	
		Tikus_Sawah	10	5	
		Ulat_Grayak	7	8	
		Walang_Sangit	12	3	
		Wereng_Coklat	8	7	
Total			58	32	
Pembagian Data	Epoch	Kelas	Hasil Klasifikasi		Akurasi
			Benar	Salah	
80%:20%	100	Belalang	8	7	68%
		Penggerek_Batang	13	2	
		Tikus_Sawah	11	4	
		Ulat_Grayak	8	7	
		Walang_Sangit	13	2	
		Wereng_Coklat	9	6	
Total			62	28	

4.4.2 Presisi, Recall, dan F1-Score

Presisi menunjukkan seberapa tepat prediksi model untuk satu kelas hama—yakni dari semua gambar yang diprediksi sebagai belalang, berapa yang benar-

benar belalang. Sementara recall menunjukkan kemampuan model dalam menemukan semua contoh dari suatu kelas dalam dataset—yakni dari semua gambar belalang yang ada, berapa yang berhasil dikenali oleh model. Nilai presisi dan recall yang tinggi menunjukkan bahwa model tidak hanya akurat secara keseluruhan, tetapi juga handal untuk setiap kelas hama secara individual.

Untuk mendapatkan gambaran menyeluruh dari keseimbangan antara presisi dan recall, digunakan juga F1-score, yang merupakan rata-rata harmonik dari keduanya. Nilai F1 yang mendekati 1 menunjukkan keseimbangan yang baik antara jumlah hama yang terdeteksi dan ketepatan deteksi tersebut. Misalnya, untuk kelas penggerek batang, didapatkan nilai presisi 0.80, recall 0.80, dan F1-score 0.80, yang berarti model sangat baik dalam mengenali kelas ini.

4.4.3 Confusion matrix

Tabel 4. 3 Tabel Confusion Matrix

	Belalang	Penggerek_Batang	Tikus_Sawah	Ulat_Grayak	Walang Sangit	Wereng_Coklat
Belalang	8	1	0	0	4	2
Penggerek_Batang	0	13	0	1	0	1
Tikus_Sawah	2	0	11	0	1	1
Ulat_Grayak	0	4	0	8	2	1
Walang_Sangit	2	0	0	0	13	0
Wereng_Coklat	0	0	0	3	3	9

Confusion matrix memberikan gambaran visual mengenai kesalahan model dalam memprediksi, misalnya jika gambar ulat grayak sering diklasifikasikan sebagai penggerek batang, hal ini akan terlihat dari angka tinggi di sel yang sesuai. Pada confusion matrix, terdapat 4 gambar ulat grayak yang salah diklasifikasikan sebagai penggerek batang, ini menandakan bahwa fitur visual kedua hama ini

cenderung mirip, dan mungkin perlu augmentasi atau penambahan data untuk membedakannya dengan lebih baik.

Model juga diuji terhadap gambar baru dari luar dataset pelatihan untuk mengevaluasi kemampuan generalisasi. Gambar-gambar ini diambil dari berbagai sumber terbuka dan diproses serupa dengan gambar pelatihan. Hasilnya, model masih mampu memberikan klasifikasi yang tepat dengan akurasi 68%, yang menunjukkan bahwa model tidak overfitting dan masih bekerja baik pada data baru yang belum pernah dilihat sebelumnya.

Melalui evaluasi ini, dapat disimpulkan bahwa model klasifikasi hama berbasis DenseNet menunjukkan performa yang sangat baik dalam mengidentifikasi berbagai jenis hama tanaman padi. Namun, tetap ada beberapa kelas yang perlu perhatian lebih, seperti beetle dan bollworm, yang sering tertukar. Solusinya dapat berupa penambahan data lebih banyak untuk kelas tersebut atau penggunaan teknik pembobotan kelas selama pelatihan.

4.5 Implementasi sistem

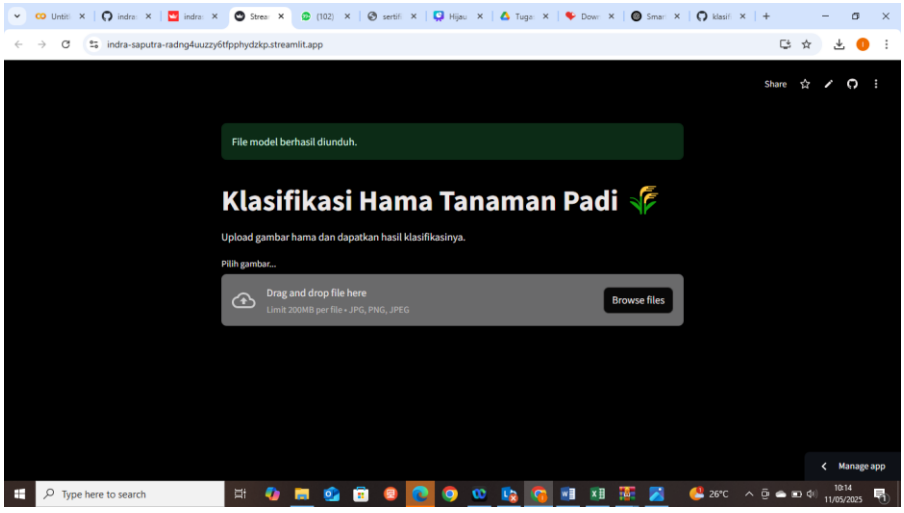
4.5.1 Arsitektur Sistem

Arsitektur sistem terdiri dari beberapa komponen utama, yaitu: model klasifikasi yang telah dilatih menggunakan DenseNet, antarmuka web dengan Streamlit, serta fungsi tambahan untuk menampilkan hasil klasifikasi beserta solusi atau penanganan hama berdasarkan output model. Model yang disimpan dalam format .h5 diimpor kembali ke dalam program, kemudian digunakan untuk memproses input berupa gambar yang diunggah oleh pengguna. Output dari model berupa label kelas (jenis hama) dan probabilitasnya, kemudian diterjemahkan ke dalam informasi yang lebih deskriptif yang mencakup nama hama, tingkat serangan, dan solusi penanganan.

4.5.2 Antarmuka Aplikasi dengan Streamlit

Antarmuka pengguna dikembangkan menggunakan Streamlit karena framework ini ringan, mudah di-deploy, dan mendukung integrasi langsung dengan model deep learning berbasis Keras atau TensorFlow. Tampilan antarmuka Streamlit dirancang sesederhana mungkin, terdiri dari komponen-komponen berikut:

1. Judul aplikasi dan deskripsi singkat
2. Komponen untuk mengunggah gambar (.jpg/.png)
3. Tampilan visual gambar yang diunggah
4. Hasil klasifikasi berupa nama hama dan probabilitas
5. Teks informasi penanganan berdasarkan jenis hama
6. Tombol untuk memuat ulang prediksi atau mengunggah gambar baru

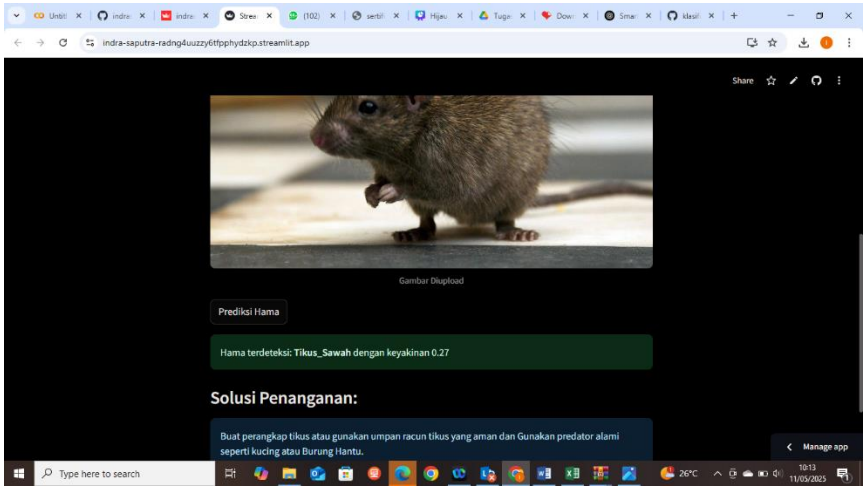


Gambar 4. 5 Tampilan Website

4.5.3 Integrasi dan Pengujian Aplikasi

Setelah antarmuka selesai dikembangkan, aplikasi diuji menggunakan berbagai contoh gambar hama dari dataset dan juga gambar yang diperoleh secara acak dari internet. Tujuan pengujian ini adalah untuk memastikan bahwa aplikasi

dapat mengenali citra dengan baik di luar data pelatihan (uji generalisasi), dan menampilkan solusi yang sesuai dengan prediksi yang diberikan oleh model.



Gambar 4. 6 Hasil Pengujian Model

Dengan adanya sistem berbasis web ini, pengguna tidak perlu memahami teknis machine learning untuk dapat menggunakan hasil penelitian ini. Cukup dengan mengunggah gambar, pengguna dapat segera mengetahui jenis hama serta rekomendasi penanganannya. Implementasi ini menunjukkan penerapan nyata dari teknologi deep learning dalam dunia pertanian yang inklusif dan praktis. Sistem ini dapat dikembangkan lebih lanjut menjadi aplikasi mobile atau bahkan terhubung dengan perangkat IoT untuk mendeteksi hama secara real-time di lapangan.

BAB V
KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil penelitian dan implementasi yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut:

1. Model klasifikasi berbasis DenseNet mampu mengenali jenis hama tanaman padi melalui citra digital dengan cukup baik. Model dilatih menggunakan dataset gambar berbagai jenis hama seperti belalang, ulat grayak, tikus sawah, dan lainnya. Proses pelatihan dan validasi menunjukkan bahwa model dapat membedakan karakteristik visual masing-masing hama, bahkan dalam kondisi pencahayaan dan bentuk yang bervariasi.
2. Integrasi hasil klasifikasi dengan informasi penanganan hama dapat dilakukan secara otomatis dalam sistem, di mana setelah model melakukan prediksi, sistem secara langsung menampilkan solusi penanganan sesuai jenis hama. Solusi ini disajikan dalam bentuk deskriptif dan informatif agar dapat dimengerti oleh pengguna akhir seperti petani atau teknisi lapangan.
3. Performa model cukup memuaskan, dengan nilai akurasi, presisi, dan recall yang menunjukkan tingkat keberhasilan tinggi sebesar 80% dalam klasifikasi multi-kelas. Evaluasi dilakukan dengan mengukur confusion matrix, nilai F1-score, serta akurasi pada data uji. Model dengan perbandingan data pelatihan dan validasi 80:20 memberikan performa terbaik secara konsisten.

5.2 Saran

Dari penelitian ini, beberapa saran dapat diberikan untuk pengembangan lebih lanjut:

1. Perluasan dan peningkatan kualitas dataset sangat disarankan, baik dalam jumlah maupun keragaman gambar. Dataset yang lebih besar dan representatif akan meningkatkan generalisasi model serta akurasi klasifikasi pada kondisi nyata di lapangan.

2. Penambahan fitur deteksi dini dan klasifikasi multi-objek (jika dalam satu gambar terdapat lebih dari satu hama) akan membuat sistem lebih robust dan relevan dengan kondisi nyata. Hal ini dapat diwujudkan dengan menggabungkan model klasifikasi dengan object detection seperti YOLO atau Faster R-CNN.
3. Implementasi model pada platform mobile atau terintegrasi dengan IoT akan lebih bermanfaat bagi petani di lapangan. Dengan sensor kamera otomatis atau drone, sistem ini dapat mendeteksi dan memberikan rekomendasi penanganan secara real-time.
4. Kolaborasi dengan pakar pertanian atau lembaga penyuluh dapat memperkaya informasi solusi penanganan hama dan menjamin bahwa saran yang diberikan sesuai dengan kondisi lokal dan kebijakan pertanian yang berlaku.

REFRENSI

Ananda, R. &. (2022). Implementasi CNN untuk identifikasi hama tanaman padi. *Jurnal Teknologi Agroindustri*.

Anggraini, W. S. (2023). Implementasi Model CNN untuk Klasifikasi Gambar di Perangkat Edge. *Jurnal Teknologi dan Aplikasi Komputer*, 33-35.

Arif Faizin, A. T. (2022). Deep Pre-Trained Model Menggunakan Arsitektur DenseNet untuk Identifikasi Penyakit Daun Padi. *Jurnal Mahasiswa Teknik Informatika).*

Asseweth, M. Y. (2024). Klasifikasi Hama dan Penyakit Padi Menggunakan DenseNet dan Grid Search. *Repositori UMA.*

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Spinger.

Christiawan, G., Putra, R., Sulaiman, A., & Poerbaningtyas, E. (2023). Penerapan Metode Convolutional Neural Network (CNN) Dalam Mengklasifikasikan Penyakit Daun Tanaman Padi. *journal of information and technology*.

Fadillah, R. N. (2022). Optimasi klasifikasi gambar menggunakan DenseNet. *Jurnal Teknologi Informasi dan Ilmu Komputer*.

Fahmi, R. &. (2021). Evaluasi Model CNN untuk Klasifikasi Gambar Hama Tanaman Menggunakan Dataset Lokal. *Jurnal Teknologi Informasi*.

Hidayat, A. &. (2020). Konsep translasi invarian dalam CNN dan implementasinya. *Jurnal Komputasi dan Teknologi*, 23-25.

Iskandar, B. S. (2021). Penerapan DenseNet pada citra daun tanaman. *Jurnal Rekayasa dan Teknologi Sistem Informasi*.

Istiqomah, N. (2024). Klasifikasi Penyakit Tanaman Padi Berbasis Citra Daun Menggunakan Convolutional Neural Network (CNN). *Jurnal Sarjana Teknik Informatika*.

Kurniawan, F. P. (2022). Penerapan DenseNet untuk klasifikasi gambar penyakit tanaman. *Jurnal Teknologi Informasi dan Ilmu Komputer*, 112-119.

Kusuma, b. M., Hermanto, T. I., & Lestari, C. D. (2025). Klasifikasi Jenis Penyakit Pada Tanaman Padi Menggunakan Algoritma Convolutional Neural Network. *Jurnal Informatika dan Komputer*.

Maulana, A. S. (2020). Implementasi Model Klasifikasi Gambar pada Aplikasi Menggunakan Flask. *Jurnal Ilmiah Teknologi Informasi*, 33-35.

Milano, A., & Yasid, A. (2024). Klasifikasi Penyakit Daun Padi Menggunakan Model Deep Learning Efficientnet-B6. *Jurnal Informatika dan Teknik Elektro Terapan*.

Muhdar, F., Fhitra, Safrial, Fatmawati, Amelia, R., & Nurlina. (2025). PENGENALAN DAN PENGENDALIAN ORGANISME PENGGANGGU TANAMAN (OPT) PADA TANAMAN PADI DAN JAGUNG DI DESA TACIPONG. *Jurnal Pengabdian Masyarakat Kita Semua*, 2-3.

Nugroho, D. L. (2021). Deteksi Hama Tanaman Menggunakan Deep Learning dengan Arsitektur DenseNet. *Jurnal Ilmu Komputer Terapan*.

Nugroho, M., & Nurraharjo, E. (2003). KLASIFIKASI HAMA TANAMAN PADI BERDASARKAN CITRA DAUN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK. *Jurnal Pendidikan Biologi dan Sains*, 2-3.

Nurhayati, R. &. (2019). Desain UX dan Keamanan pada Aplikasi Berbasis AI. . *Jurnal Teknologi Informasi dan Komputerisasi*.

Nurhidayati, S. &. (2020). Ekstraksi fitur tekstur pada klasifikasi citra daun. *Jurnal Teknologi dan Sistem Komputer*, 34-40.

Putra, A. &. (2020). Penerapan Metode CNN pada Klasifikasi Gambar Hama dan Penyakit Tanaman. *Jurnal Sains Komputer dan Informatika*.

Putra, M. Y. (2023). Integrasi CNN dan Streamlit dalam sistem klasifikasi daun. *Jurnal Teknologi dan Sistem Informasi*.

Rachman, D. &. (2021). Efisiensi parameter dalam DenseNet.

Rahmat, M. &. (2022). Struktur dasar CNN dalam pengolahan citra digital. *Jurnal Informatika UIN*, 45-52.

Saputra, R. Y. (2022). Analisis Confusion Matrix dalam Evaluasi Kinerja Model Deep Learning. *Jurnal Teknik Informatika dan Sistem Informasi*, 27–35.

Sari, E. &. (2020). Feature reuse pada DenseNet untuk klasifikasi citra digital. *Jurnal Informatika Polinema*.

Sastrosiswojo, S. (2022). Pengendalian Hama dan Penyakit Tanaman Padi. *Balai Penelitian Tanaman Pangan*.

Setiawan, R. &. (2023). Perbandingan Kinerja Model CNN dalam Aplikasi Klasifikasi Gambar. *Jurnal Teknologi dan Sistem Komputer*, 11-15.

Setyawan, D. P. (2021). Pelatihan CNN untuk klasifikasi daun menggunakan dataset lokal. *Jurnal Teknologi Informasi*, 78-85.

Siregar, A. M. (2020). Evaluasi performa klasifikasi citra menggunakan metrik evaluasi. *Jurnal Sains dan Teknologi*, 44-50.

Sitompul, P. (2021). Identifikasi Penyakit Tanaman Padi Melalui Citra Daun Menggunakan DenseNet 201. *Journal of Machine Learning andArtificial Intelligence*.

Susi, Y., Aradea, & Rachman, A. N. (2022, April). Implementasi Deep Learning pada Sistem Klasifikasi Hama Tanaman. *Jurnal Buana Informatika*, 4-7.

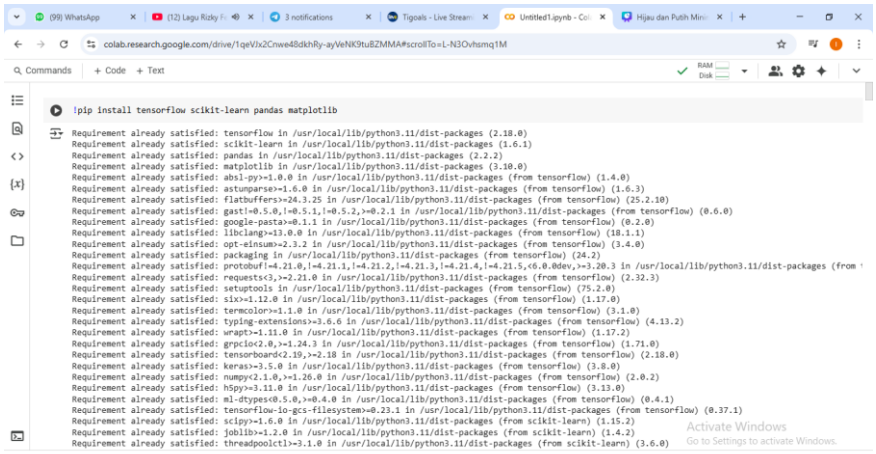
Syam, T. A., & Nur, A. (2021). Kendala dan Solusi dalam Pengendalian Wereng Cokelat. . *Jurnal Proteksi Tanaman*, 32–41.

Wahyuni, S. &. (2021). Pengaruh preprocessing terhadap akurasi klasifikasi citra daun. *Jurnal Teknologi Informasi*, 45-52.

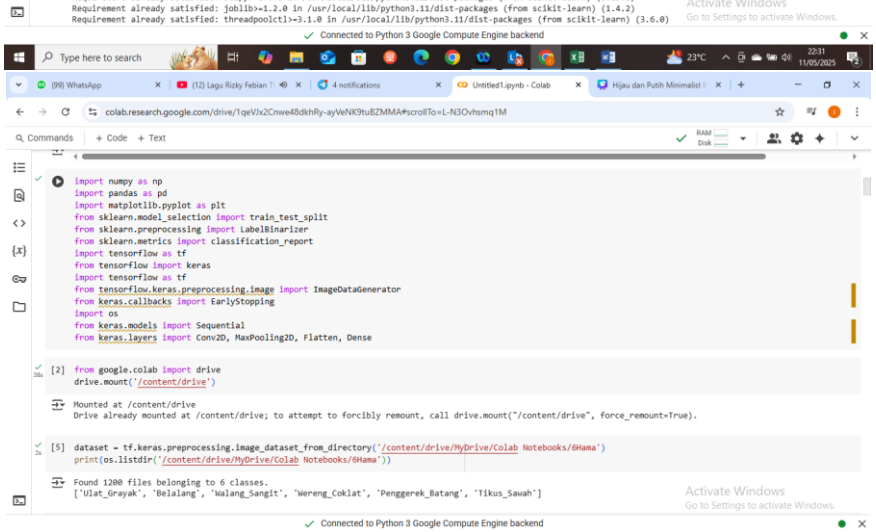
Wijayanti, R. S. (2023). Integrasi klasifikasi citra dengan IoT untuk monitoring tanaman. *Jurnal Teknologi Pertanian Cerdas*, 10-18.

LAMPIRAN

Foto Pembuatan Machine Leaning



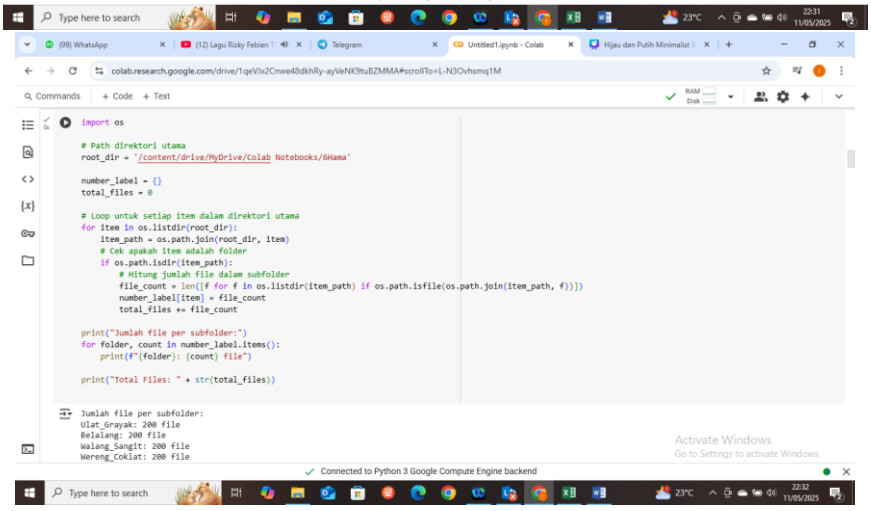
```
!pip install tensorflow scikit-learn pandas matplotlib

Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.13.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.13.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
```



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import classification_report
import tensorflow as tf
from tensorflow import keras
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping
import os
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
```

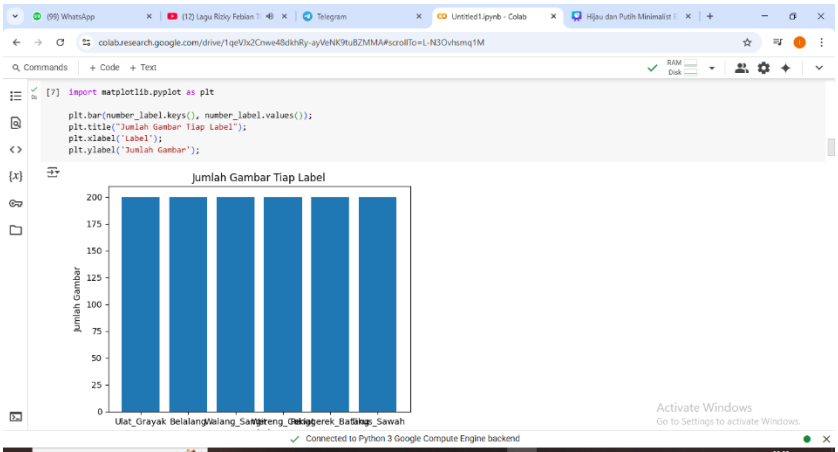
```
[2] from google.colab import drive
    drive.mount('/content/drive')

Mounted at /content/drive
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

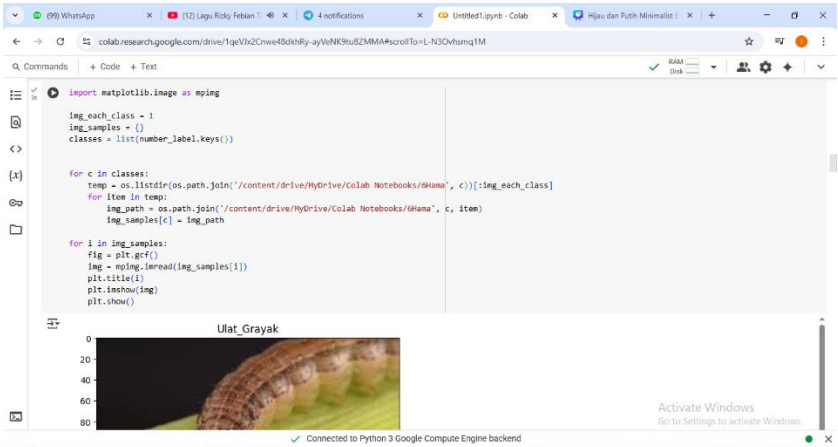
```
[5] dataset = tf.keras.preprocessing.image_dataset_from_directory('/content/drive/MyDrive/Colab Notebooks/6Hama')
    print(os.listdir('/content/drive/MyDrive/Colab Notebooks/6Hama'))

Found 1200 files belonging to 6 classes.
['Ulat_Grayak', 'Belalang', 'Walang_Sangit', 'Wereng_Coklat', 'Penggerek_Batang', 'Tikus_Sawah']
```

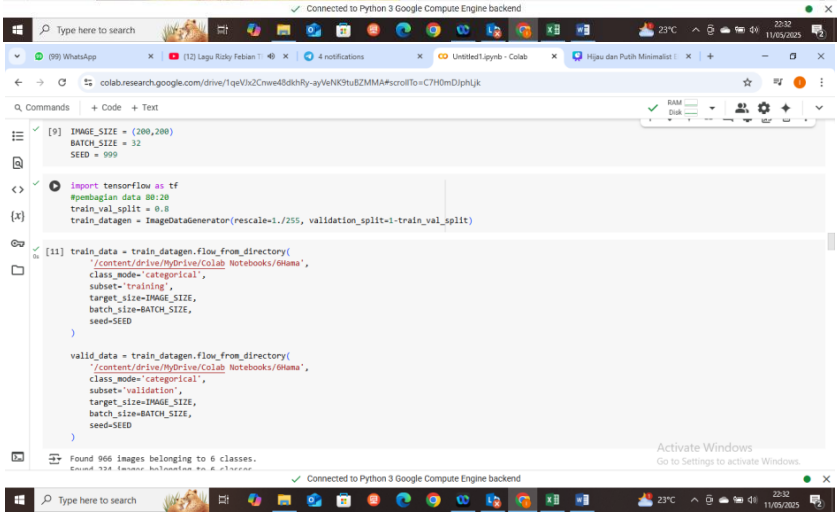


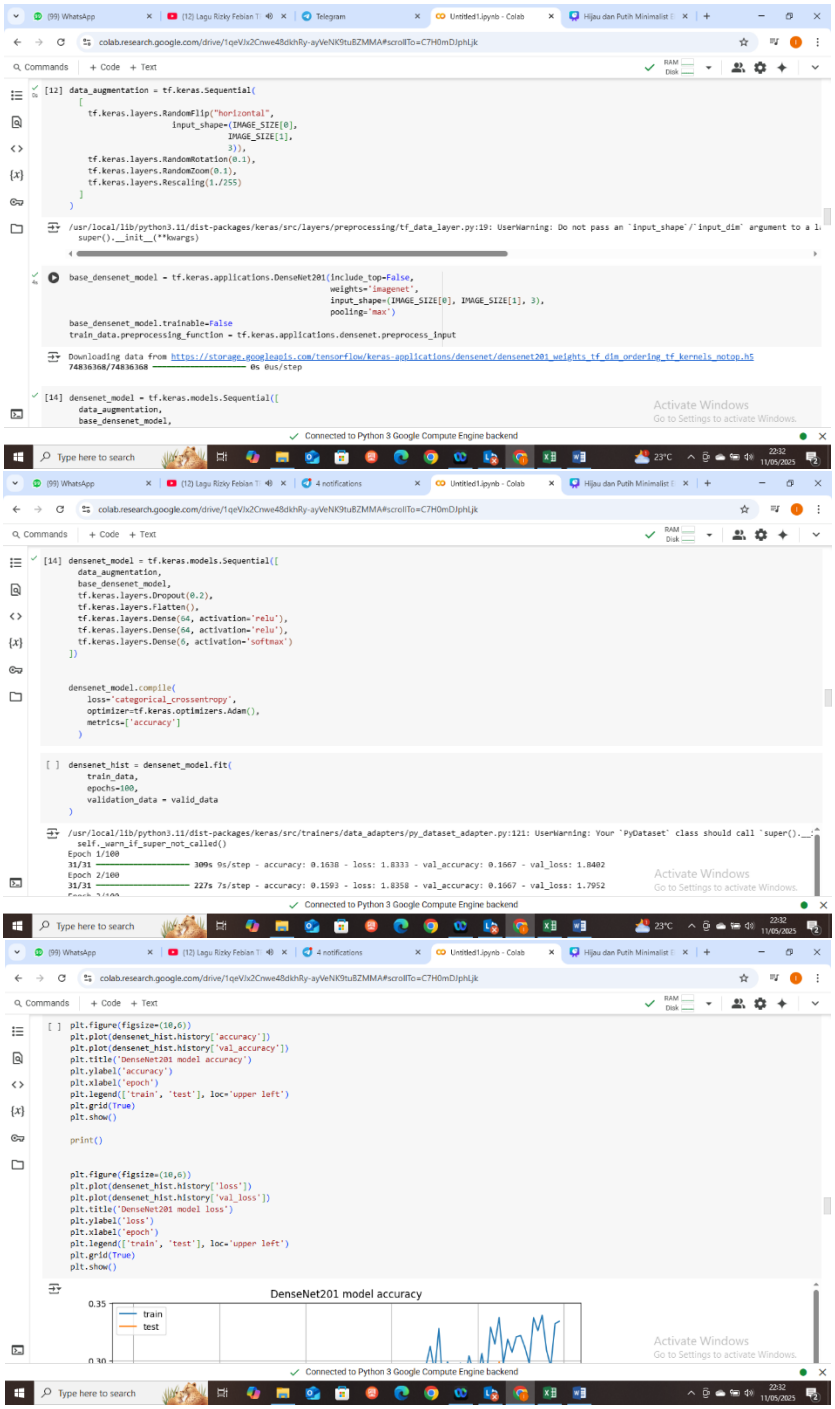
```
import os

# Path direktori utama
root_dir = '/content/drive/MyDrive/Colab Notebooks/6Hama'

number_label = {}
total_files = 0

# Loop untuk setiap item dalam direktori utama
for item in os.listdir(root_dir):
    item_path = os.path.join(root_dir, item)
    # Cek apakah item adalah folder
    if os.path.isdir(item_path):
        # Hitung jumlah file dalam subfolder
        file_count = len([f for f in os.listdir(item_path) if os.path.isfile(os.path.join(item_path, f))])
        number_label[item] = file_count
        total_files += file_count

print("Jumlah file per subfolder:")
for folder, count in number_label.items():
    print(f"{folder}: {count} file")

print("Total Files: " + str(total_files))
```

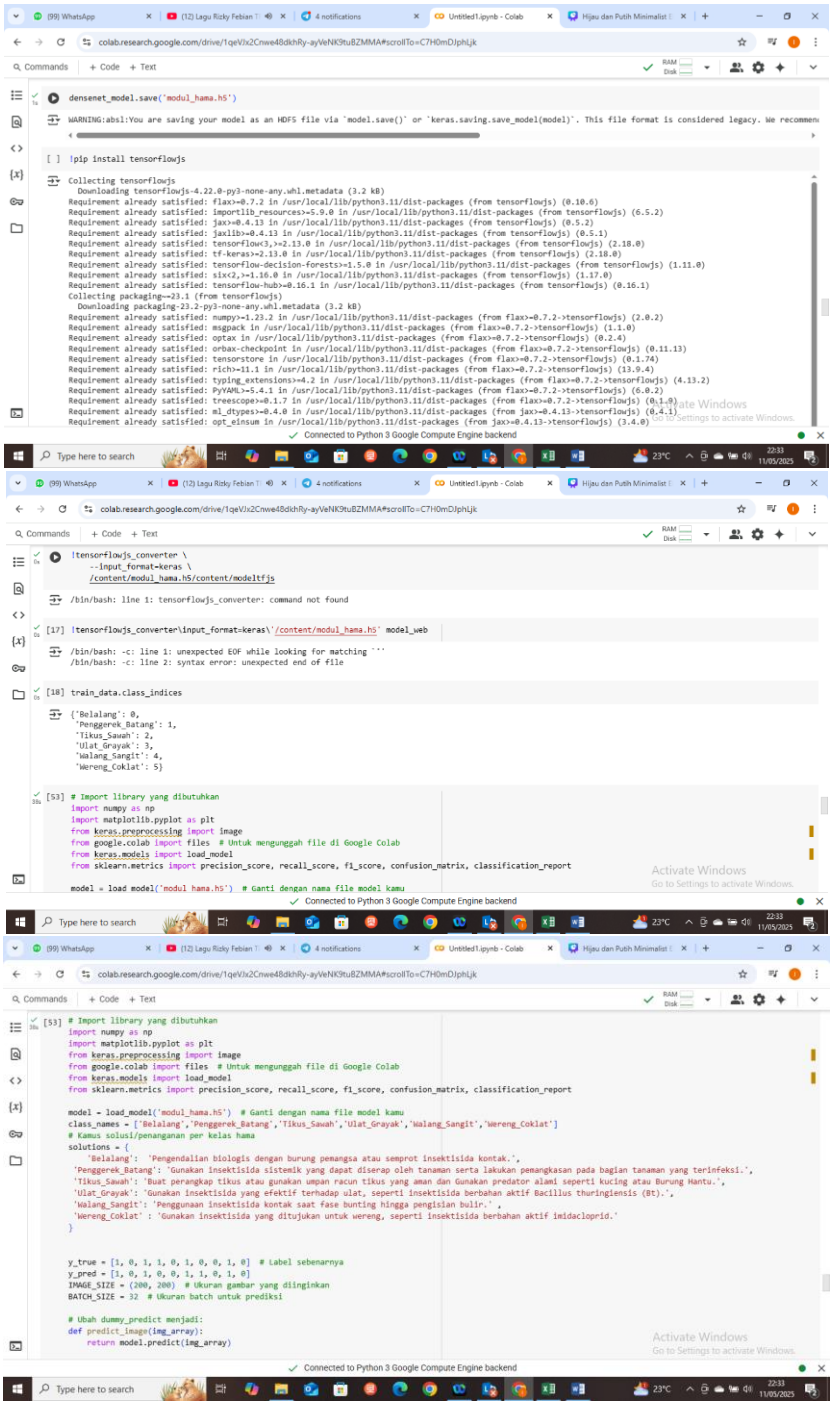
```
Jumlah file per subfolder:
Ulat_Grayak: 200 file
Belalang: 200 file
Walang_Sangit: 200 file
Wereng_Coklat: 200 file
```


[7] import matplotlib.pyplot as plt
plt.bar(number_label.keys(), number_label.values());
plt.title("Jumlah Gambar Tiap Label");
plt.xlabel('Label');
plt.ylabel('Jumlah Gambar');
Jumlah Gambar Tiap Label
Jumlah Gambar
Ulat_Grayak
Connected to Python 3 Google Compute Engine backend


import matplotlib.image as mpimg
img_each_class = 1
img_samples = []
classes = list(number_label.keys())
for c in classes:
temp = os.listdir(os.path.join('/content/drive/MyDrive/Colab Notebooks/6Hama', c))[:img_each_class]
for item in temp:
img_path = os.path.join('/content/drive/MyDrive/Colab Notebooks/6Hama', c, item)
img_samples[c] = img_path
for i in img_samples:
fig = plt.gcf()
img = mpimg.imread(img_samples[i])
plt.title(i)
plt.imshow(img)
plt.show()
Ulat_Grayak
Connected to Python 3 Google Compute Engine backend


[9] IMAGE_SIZE = (200,200)
BATCH_SIZE = 32
SEED = 999
import tensorflow as tf
#pembagian data 80:20
train_val_split = 0.8
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=1-train_val_split)
[11] train_data = train_datagen.flow_from_directory(
'/content/drive/MyDrive/Colab Notebooks/6Hama',
class_mode='categorical',
subset='training',
target_size=IMAGE_SIZE,
batch_size=BATCH_SIZE,
seed=SEED
)
valid_data = train_datagen.flow_from_directory(
'/content/drive/MyDrive/Colab Notebooks/6Hama',
class_mode='categorical',
subset='validation',
target_size=IMAGE_SIZE,
batch_size=BATCH_SIZE,
seed=SEED
)
Found 966 images belonging to 6 classes.
Connected to Python 3 Google Compute Engine backend


[12] data_augmentation = tf.keras.Sequential(
[
tf.keras.layers.RandomFlip("horizontal",
input_shape=(IMAGE_SIZE[0],
IMAGE_SIZE[1],
3)),
tf.keras.layers.RandomRotation(0.1),
tf.keras.layers.RandomZoom(0.1),
tf.keras.layers.Rescaling(1./255)
]
)
/usr/local/lib/python3.11/dist-packages/keras/src/layers/preprocessing/tf_data_layer.py:19: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a l
super().__init__(**kwargs)
base_densenet_model = tf.keras.applications.DenseNet201(include_top=False,
weights='imagenet',
input_shape=(IMAGE_SIZE[0], IMAGE_SIZE[1], 3),
pooling='max')
base_densenet_model.trainable=False
train_data.preprocessing_function = tf.keras.applications.densenet.preprocess_input
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet201_weights_tf_dim_ordering_tf_kernels_notop.h5
74836368/74836368 0s 0us/step
[14] densenet_model = tf.keras.models.Sequential([
data_augmentation,
base_densenet_model,
Activate Windows
Go to Settings to activate Windows.
Connected to Python 3 Google Compute Engine backend
[14] densenet_model = tf.keras.models.Sequential([
data_augmentation,
base_densenet_model,
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(64, activation='relu'),
tf.keras.layers.Dense(64, activation='relu'),
tf.keras.layers.Dense(6, activation='softmax')
])
densenet_model.compile(
loss='categorical_crossentropy',
optimizer=tf.keras.optimizers.Adam(),
metrics=['accuracy']
)
densenet_hist = densenet_model.fit(
train_data,
epochs=100,
validation_data = valid_data
)
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__
self._warn_if_super_not_called()
Epoch 1/100
31/31 309s 9s/step - accuracy: 0.1638 - loss: 1.8333 - val_accuracy: 0.1667 - val_loss: 1.8402
Epoch 2/100
31/31 227s 7s/step - accuracy: 0.1593 - loss: 1.8358 - val_accuracy: 0.1667 - val_loss: 1.7952
Activate Windows
Go to Settings to activate Windows.
Connected to Python 3 Google Compute Engine backend
[ ] plt.figure(figsize=(10,6))
plt.plot(densenet_hist.history['accuracy'])
plt.plot(densenet_hist.history['val_accuracy'])
plt.title('DenseNet201 model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()
print()
plt.figure(figsize=(10,6))
plt.plot(densenet_hist.history['loss'])
plt.plot(densenet_hist.history['val_loss'])
plt.title('DenseNet201 model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()
DenseNet201 model accuracy
0.35
0.30
train
test
Activate Windows
Go to Settings to activate Windows.
Connected to Python 3 Google Compute Engine backend



Coding Machine Learning

```
!pip install tensorflow scikit-learn pandas matplotlib

import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import classification_report
import tensorflow as tf
from tensorflow import keras
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping
import os
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from google.colab import drive
drive.mount('/content/drive')
dataset=
tf.keras.preprocessing.image_dataset_from_directory('/content/drive/MyDrive/Col
ab Notebooks/6Hama')
print(os.listdir('/content/drive/MyDrive/Colab Notebooks/6Hama'))
import os
# Path direktori utama
root_dir = '/content/drive/MyDrive/Colab Notebooks/6Hama'
number_label = {}
total_files = 0
for item in os.listdir(root_dir):
    item_path = os.path.join(root_dir, item)
    if os.path.isdir(item_path):
```

```
file_count = len([f for f in os.listdir(item_path
if os.path.isfile(os.path.join(item_path, f))])
        number_label[item] = file_count
        total_files += file_count
print("Jumlah file per subfolder:")
for folder, count in number_label.items():
    print(f"{folder}: {count} file")
print("Total Files: " + str(total_files))
import matplotlib.pyplot as plt
plt.bar(number_label.keys(), number_label.values());
plt.title("Jumlah Gambar Tiap Label");
plt.xlabel('Label');
plt.ylabel('Jumlah Gambar');
import matplotlib.image as mpimg
img_each_class = 1
img_samples = {}
classes = list(number_label.keys())
for c in classes:
    temp = os.listdir(os.path.join('/content/drive/MyDrive/Colab
Notebooks/6Hama', c))[:img_each_class]
    for item in temp:
        img_path = os.path.join('/content/drive/MyDrive/Colab Notebooks/6Hama',
c, item)
        img_samples[c] = img_path
for i in img_samples:
    fig = plt.gcf()
    img = mpimg.imread(img_samples[i])
```

```
    plt.title(i)
    plt.imshow(img)
    plt.show()
IMAGE_SIZE = (200,200)
BATCH_SIZE = 32
SEED = 999
import tensorflow as tf
#pembagian data 80:20
train_val_split = 0.8
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=1-
train_val_split)
train_data = train_datagen.flow_from_directory(
    '/content/drive/MyDrive/Colab Notebooks/6Hama',
    class_mode='categorical',
    subset='training',
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    seed=SEED
)
valid_data = train_datagen.flow_from_directory(
    '/content/drive/MyDrive/Colab Notebooks/6Hama',
    class_mode='categorical',
    subset='validation',
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    seed=SEED
)
```

```
data_augmentation = tf.keras.Sequential(
 [
  tf.keras.layers.RandomFlip("horizontal",
                input_shape=(IMAGE_SIZE[0],
                      IMAGE_SIZE[1],
                      3)),
  tf.keras.layers.RandomRotation(0.1),
  tf.keras.layers.RandomZoom(0.1),
  tf.keras.layers.Rescaling(1./255)
 ]
)
base_densenet_model = tf.keras.applications.DenseNet201(include_top=False,
                                    weights='imagenet',
                                    input_shape=(IMAGE_SIZE[0],
IMAGE_SIZE[1], 3),
                                    pooling='max')
base_densenet_model.trainable=False
train_data.preprocessing_function =
tf.keras.applications.densenet.preprocess_input
densenet_model = tf.keras.models.Sequential([
 data_augmentation,
 base_densenet_model,
 tf.keras.layers.Dropout(0.2),
 tf.keras.layers.Flatten(),
 tf.keras.layers.Dense(64, activation='relu'),
 tf.keras.layers.Dense(64, activation='relu'),
 tf.keras.layers.Dense(6, activation='softmax')
```

```
])
densenet_model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adam(),
    metrics=['accuracy']
  )
densenet_hist = densenet_model.fit(
    train_data,
    epochs=100,
    validation_data = valid_data
)
plt.figure(figsize=(10,6))
plt.plot(densenet_hist.history['accuracy'])
plt.plot(densenet_hist.history['val_accuracy'])
plt.title('DenseNet201 model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()
print()
plt.figure(figsize=(10,6))
plt.plot(densenet_hist.history['loss'])
plt.plot(densenet_hist.history['val_loss'])
plt.title('DenseNet201 model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
```

```
plt.legend(['train', 'test'], loc='upper left')

plt.grid(True)

plt.show()

densenet_model.save('modul_hama.h5')

# Import library yang dibutuhkan

import numpy as np

import matplotlib.pyplot as plt

from keras.preprocessing import image

from google.colab import files  # Untuk mengunggah file di Google Colab

from keras.models import load_model

from sklearn.metrics import precision_score, recall_score, f1_score,
confusion_matrix, classification_report

model = load_model('modul_hama.h5')  # Ganti dengan nama file model kamu

class_names =
['Belalang','Penggerek_Batang','Tikus_Sawah','Ulat_Grayak','Walang_Sangit','We
reng_Coklat']

solutions = {

    'Belalang':  'Pengendalian biologis dengan burung pemangsa atau semprot
insektisida kontak.',

 'Penggerek_Batang': 'Gunakan insektisida sistemik yang dapat diserap oleh
tanaman serta lakukan pemangkasan pada bagian tanaman yang terinfeksi.',

 'Tikus_Sawah': 'Buat perangkap tikus atau gunakan umpan racun tikus yang aman
dan Gunakan predator alami seperti kucing atau Burung Hantu.',

 'Ulat_Grayak': 'Gunakan insektisida yang efektif terhadap ulat, seperti insektisida
berbahan aktif Bacillus thuringiensis (Bt).',

 'Walang_Sangit': 'Penggunaan insektisida kontak saat fase bunting hingga
pengisian bulir.' ,

 'Wereng_Coklat' : 'Gunakan insektisida yang ditujukan untuk wereng, seperti
insektisida berbahan aktif imidacloprid.'

}
```

```
y_true = [1, 0, 1, 1, 0, 1, 0, 0, 1, 0]  # Label sebenarnya
y_pred = [1, 0, 1, 0, 0, 1, 1, 0, 1, 0]
IMAGE_SIZE = (200, 200)  # Ukuran gambar yang diinginkan
BATCH_SIZE = 32  # Ukuran batch untuk prediksi
# Ubah dummy_predict menjadi:
def predict_image(img_array):
    return model.predict(img_array)
# Upload file gambar
uploaded = files.upload()  # Ini akan memunculkan dialog untuk mengunggah file
for fn in uploaded.keys():
    # Load gambar dan resize
    img = image.load_img(fn, target_size=IMAGE_SIZE)  # Memuat gambar dengan ukuran yang ditentukan
    plt.imshow(img)  # Menampilkan gambar
    plt.axis('off')  # Menyembunyikan sumbu
    plt.show()  # Menampilkan gambar
    # Konversi gambar ke array dan beri tambahan dimensi batch
    x = image.img_to_array(img)  # Mengubah gambar menjadi array
    x = np.expand_dims(x, axis=0)  # Menambahkan dimensi untuk batch
    # Simulasi prediksi menggunakan dummy model
    prediction = model.predict(x)
    class_index = np.argmax(prediction)
    class_label = class_names[class_index]  # Mengambil indeks kelas dengan probabilitas tertinggi
    print(f"Nama file: {fn}")
    print(f"Klasifikasi: {class_label} (Probabilitas: {prediction[0][class_index]:.2f})")
```

```
    if class_label in solutions:
        print(f"Solusi: {solutions[class_label]}")
    else:
        print("Solusi belum tersedia untuk kelas ini.")
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)
print(f'\nPresisi: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
```