

**ANALISIS PERBANDINGAN PERFORMA ARSITEKTUR VGG16 DAN
RESNET50 DALAM KLASIFIKASI MULTI-KELAS PADA PENYAKIT
TANAMAN PADI BERBASIS CONVOLUTIONAL NEURAL NETWORK
(CNN)**

SKRIPSI

DISUSUN OLEH

KRISNA ADITYA

NPM. 2109020048



UMSU

Unggul | Cerdas | Terpercaya

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA
MEDAN
2025**

**ANALISIS PERBANDINGAN PERFORMA ARSITEKTUR VGG16 DAN RESNET50
DALAM KLASIFIKASI MULTI-KELAS PADA PENYAKIT TANAMAN PADI
BERBASIS CONVOLUTIONAL NEURAL NETWORK (CNN)**

SKRIPSI

**Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer
(S.Kom) dalam Program Studi Teknologi Infomasi pada Fakultas Ilmu Komputer
dan Teknologi Informasi, Universitas Muhammadiyah Sumatera Utara**

**KRISNA ADITYA
NPM. 2109020048**

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA
MEDAN
2025**

LEMBAR PENGESAHAN

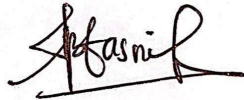
Judul Skripsi : ANALISIS PERBANDINGAN PERFORMA
ARSITEKTUR VGG16 DAN RESNET50 DALAM
KLASIFIKASI MULTI-KELAS PADA PENYAKIT
TANAMAN PADI BERBASIS CONVOLUTIONAL
NEURAL NETWORK (CNN)

Nama Mahasiswa : KRISNA ADITYA

NPM : 2109020048

Program Studi : TEKNOLOGI INFORMASI

Menyetujui
Komisi Pembimbing



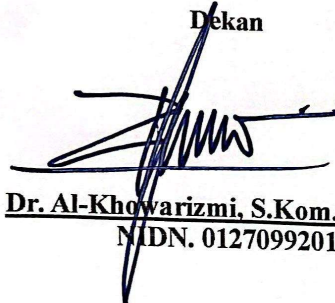
Mhd. Basri, S.Si., M.Kom.
NIDN. 0111078802

Ketua Program Studi



Fatma Sari Hutagalung, S.Kom., M.Kom.
NIDN. 0117019301

Dekan



Dr. Al-Khowarizmi, S.Kom., M.Kom.
NIDN. 0127099201

PERNYATAAN ORISINALITAS

ANALISIS PERBANDINGAN PERFORMA ARSITEKTUR VGG16 DAN RESNET50 DALAM KLASIFIKASI MULTI-KELAS PADA PENYAKIT TANAMAN PADI BERBASIS CONVOLUTIONAL NEURAL NETWORK (CNN)

SKRIPSI

Penulis menyatakan bahwa karya tulis ini adalah hasil karya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing disebutkan sumbernya.

Medan, 5 Mei 2025

Yang membuat pernyataan



Krisna Aditya

NPM. 210902048

**PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN
AKADEMIS**

Sebagai sivitas akademika Universitas Muhammadiyah Sumatera Utara, penulis bertanda tangan dibawah ini:

Nama : KRISNA ADITYA
NPM 2109020048
Program Studi : TEKNOLOGI INFORMASI
Karya Ilmiah : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Muhammadiyah Sumatera Utara Hak Bebas Royalti Non-Eksekutif (*Non-Exclusive Royalty free Right*) atas penelitian skripsi penulis yang berjudul:

**ANALISIS PERBANDINGAN PERFORMA ARSITEKTUR VGG16 DAN
RESNET50 DALAM KLASIFIKASI MULTI-KELAS PADA PENYAKIT
TANAMAN PADI BERBASIS CONVOLUTIONAL NEURAL NETWORK
(CNN)**

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-Eksekutif ini, Universitas Muhammadiyah Sumatera Utara berhak menyimpan, mengalih media, memformat, mengelola dalam bentuk database, merawat dan mempublikasikan Skripsi penulis ini tanpa meminta izin dari penulis selama tetap mencantumkan nama penulis sebagai penulis dan sebagai pemegang dan atau sebagai pemilik hak cipta.

Demikian pernyataan ini dibuat dengan sebenarnya.

Medan, 5 Mei 2025

Yang membuat pernyataan



Krisna Aditya

NPM. 2109020048

RIWAYAT HIDUP

DATA PRIBADI

Nama Lengkap : Krisna Aditya
Tempat dan Tanggal Lahir : Sarang Ginting, 17 Juli 2003
Alamat Rumah : Jl. Gunung Sibayak No.2
Telepon/Faks/HP : +6281264631850
E-mail : ozwellespencer49@gmail.com
Instansi Tempat Kerja : -
Alamat Kantor : -

DATA PENDIDIKAN

SD : Negeri 100707 Perkebunan Batang Toru TAMAT: 2015
SMP : SMP Negeri 1 Batang Toru TAMAT: 2018
SMA : SMA Negeri 1 Batang Toru TAMAT: 2021

KATA PENGANTAR



Assalamualaikum Warohmatullahi Wabarokatuh

Puji dan syukur penulis panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "ANALISIS PERBANDINGAN PERFORMA ARSITEKTUR VGG16 DAN RESNET50 DALAM KLASIFIKASI MULTI-KELAS PADA PENYAKIT TANAMAN PADI BERBASIS CONVOLUTIONAL NEURAL NETWORK (CNN)" dengan baik dan tepat waktu. Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Teknologi Informasi. Penulis menyadari bahwa penyelesaian skripsi ini tidak terlepas dari bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Terutama dan yang paling utama Kepada kedua orang tua tercinta ayahanda Saleh Wijaya dan Ibunda Soyem yang selalu memberikan ketulusan doanya dan juga memberikan dukungan baik secara moril maupun materil, juga motivasi sehingga penulis dapat menyelesaikan Skripsi ini. Semoga Allah SWT senantiasa melimpahkan kesehatan, keberkahan, dan kebahagiaan kepada kedua orang tua penulis, Amiin.
2. Bapak Prof. Dr. Agussani, M.AP., Rektor Universitas Muhammadiyah Sumatera Utara (UMSU), Semoga Allah SWT senantiasa melimpahkan kesehatan, keberkahan, dan kebahagiaan kepada beliau dan keluarga, Aamiin.
3. Bapak Dr. Al-Khowarizmi, S.Kom., M.Kom. Dekan Fakultas Ilmu Komputer dan Teknologi Informasi (FIKTI) UMSU, Semoga Allah SWT

senantiasa melimpahkan kesehatan, keberkahan, dan kebahagiaan kepada beliau dan keluarga, Aamiin.

4. Ibu Fatma Sari Hutagalung, S.Kom., M.Kom. Ketua Program Studi Teknologi Informasi, Semoga Allah SWT senantiasa melimpahkan kesehatan, keberkahan, dan kebahagiaan kepada beliau dan keluarga, Aamiin.
5. Bapak Mhd. Basri, S.Si., M.Kom. sebagai Sekretaris Program Studi Teknologi Informasi selaku dosen pembimbing yang tidak hanya telah membimbing penulis dalam penyusunan skripsi ini, tetapi juga telah menjadi sosok orang tua kedua bagi penulis selama menempuh pendidikan. Kesabaran, dedikasi, dan kebijaksanaan beliau dalam membimbing penulis melampaui tugas sebagai seorang dosen. Beliau tidak hanya mengarahkan penulis dalam aspek akademis tetapi juga memberikan nasihat kehidupan yang sangat berharga. Di saat-saat sulit, beliau selalu hadir memberikan motivasi dan kepercayaan yang membuat penulis tetap bertahan dan berjuang. Bimbingan dan kasih sayang yang beliau berikan akan selalu penulis kenang dan menjadi bekal berharga dalam perjalanan hidup penulis selanjutnya. Semoga Allah SWT senantiasa melimpahkan kesehatan, keberkahan, dan kebahagiaan kepada beliau dan keluarga, Aamiin.
6. Kepada Abang dan Adik penulis yang senantiasa selalu memberikan dukungan dan doa kepada penulis hingga saat ini.
7. Ucapan terima kasih yang istimewa penulis berikan kepada Nadia Aprilia Siregar, partner yang selalu setia mendampingi dalam setiap langkah perjalanan penulis selama menempuh pendidikan ini. Kehadirannya memberikan kekuatan dan inspirasi yang tak terhingga di saat penulis menghadapi berbagai tantangan dalam penyelesaian skripsi ini. Kesabaran, pengertian, dan dukungan moralnya menjadi sumber semangat yang tidak pernah padam. Di tengah kesibukan dan tekanan, yang selalu hadir memberikan motivasi, mendengarkan keluh kesah, serta memberikan sudut pandang baru yang mencerahkan. Terima kasih atas waktu yang dikorbankan, perhatian yang tulus, dan kasih sayang yang selalu diberikan.

Semoga apa yang telah kita lalui bersama menjadi awal dari perjalanan indah di masa depan, Aamiin.

8. Kepada teman – teman yang mungkin masih Bersama penulis terkhususnya Wahyu Purnomo terima kasih karna telah memberikan pengalaman selama penulis menempuh Pendidikan. Harapan penulis semoga kita dipertemukan di waktu yang Allah kehendaki nanti, Aamiin.
9. Semua pihak yang terlibat langsung atau tidak langsung yang tidak dapat penulis ucapkan satu per satu yang telah membantu dalam penyelesaian skripsi ini semoga Allah membalas dengan kebaikan, Aamiin.

“The Measure of A Man is What He Does With Power”

-Plato

ANALISIS PERBANDINGAN PERFORMA ARSITEKTUR VGG16 DAN RESNET50 DALAM KLASIFIKASI MULTI-KELAS PADA PENYAKIT TANAMAN PADI BERBASIS CONVOLUTIONAL NEURAL NETWORK (CNN)

ABSTRAK

Penelitian ini bertujuan untuk mengidentifikasi penyakit tanaman padi menggunakan metode *deep learning* dengan membandingkan arsitektur VGG16 dan ResNet50. *Dataset* yang digunakan terdiri dari 2042 gambar citra penyakit tanaman padi dari 4 kategori yaitu *Brown spot disease*, *Blast disease*, *False smut disease*, *Bacterial leaf blight disease*, dan *Healthy*. Metode penelitian meliputi pengumpulan dataset, pra-pemrosesan data, pembangunan model, pelatihan, dan evaluasi. Hasil penelitian menunjukkan bahwa arsitektur ResNet50 memiliki akurasi lebih tinggi sebesar 95,7% dibandingkan VGG16 yang mencapai 93,2%. Tingkat presisi dan *recall* untuk ResNet50 juga lebih unggul pada semua kategori penyakit. Penelitian ini berkontribusi pada pengembangan sistem deteksi dini penyakit tanaman padi yang dapat membantu petani mengurangi kerugian hasil panen.

Kata Kunci : *Deep learning*, klasifikasi penyakit padi, CNN, ResNet50, VGG16

COMPARATIVE ANALYSIS OF THE PERFORMANCE OF VGG16 AND RESNET50 ARCHITECTURES IN MULTI-CLASS CLASSIFICATION OF RICE PLANT DISEASES BASED ON CONVOLUTIONAL NEURAL NETWORKS (CNN)

ABSTRACT

This study aims to identify rice plant diseases using deep learning methods by comparing the VGG16 and ResNet50 architectures. The dataset used consists of 2042 images of rice plant diseases from 4 categories, namely brown spot disease, blast disease, false smut disease, bacterial leaf blight disease, and healthy. The research methods included dataset collection, data pre-processing, model development, training, and evaluation. The results showed that the ResNet50 architecture had a higher accuracy of 95.7% compared to VGG16, which achieved 93.2%. The precision and recall rates for ResNet50 were also superior in all disease categories. This research contributes to the development of an early detection system for rice plant diseases that can help farmers reduce crop yield losses.

Keywords: Deep learning, rice disease classification, CNN, ResNet50, VGG16

DAFTAR ISI

DAFTAR ISI.....	x
DAFTAR TABEL.....	xii
DAFTAR GAMBAR.....	xiii
BAB I.....	2
PENDAHULUAN.....	2
1.1 Latar Belakang Masalah.....	2
1.2 Rumusan Masalah	9
1.3 Batasan Masalah.....	9
1.4 Tujuan Penelitian.....	10
1.5 Manfaat Penelitian.....	10
BAB II LANDASAN TEORI.....	12
2.1 Deep Learning	12
2.1 Convolutional Neural Network (CNN).....	14
2.3 Visual Geometry Group (VGG16).....	27
2.4 Residual Network (ResNet50)	32
2.5 Transfer Learning.....	38
2.6 Perbandingan CNN, VGG16, dan ResNet50	40
2.7 Python	41
2.8 Tensorflow	42
2.9 Keras	43
2.10 Anaconda Python.....	44
2.11 Penelitian terdahulu	44
BAB III METODOLOGI PENELITIAN	60
3.1 Pendekatan Penelitian	60
3.2 Populasi dan sampel penelitian	60
3.3 Metode Pengumpulan Data	61
3.2.1 Data Sekunder	61
3.2.2 Studi Pustaka	61
3.4 Prosedur penelitian.....	61
3.5 Perangkat penelitian	64
3.6 Waktu penelitian	66
BAB IV HASIL DAN PEMBAHASAN	67

4.1	Hasil Penelitian	67
4.2	Pembahasan	96
4.2.1	Analisis Perbandingan Model	96
4.2.2	Analisis Kesalahan (Error Analysis).....	97
4.2.3	Interpretasi Hasil	97
BAB V.....		100
KESIMPULAN.....		100
5.1	Kesimpulan.....	100
5.2	Saran.....	101
DAFTAR PUSTAKA.....		102
LAMPIRAN		106
	Source Code	106
	SK Pembimbing Skripsi.....	112
	Bimbingan Proposal	113
	Bimbingan Skripsi.....	114
	LoA Jurnal.....	115

DAFTAR TABEL

Tabel 2.1 Algoritma Convolutional Neural Network.....	14
Tabel 2.2 Penelitian Terdahulu.....	50
Tabel 3.1 Kebutuhan Perangkat Keras	63
Tabel 3.2 Kebutuhan Perangkat Lunak	64
Tabel 3.3 Waktu Penelitian	65
Tabel 4.1 Perbandingan antara kedua model.....	83

DAFTAR GAMBAR

Gambar 2.1 Struktur Deep Learning	12
Gambar 2.2 Arsitektur Convolutional Neural Network	13
Gambar 2.3 Max Pooling	18
Gambar 2.4 Arsitektur VGG16	27
Gambar 2.5 Arsitektur ResNet50	32
Gambar 2.6 Transfer Learning	38
Gambar 2.7 Tensorflow API	42
Gambar 3.1 Prosedur Penelitian	61
Gambar 4.1.1 Import Library	67
Gambar 4.1.2 Pemrosesan Dataset	69
Gambar 4.1.3 Data Pre-processing	70
Gambar 4.1.4 Proses Encoding	71
Gambar 4.1.5 Normalisasi Data	71
Gambar 4.1.6 Mengkategorikan Gambar	71
Gambar 4.1.7 Menampilkan Gambar Berdasarkan Kelas	72
Gambar 4.1.8 Training test_split	73
Gambar 4.1.9 Augmentasi Data	73
Gambar 4.1.10 Model ResNet50	74
Gambar 4.1.11 Optimasi training model	75
Gambar 4.1.12 Hasil epochs.....	76
Gambar 4.1.13 Hasil epochs kedua	77
Gambar 4.1.14 Model VGG16	78
Gambar 4.1.15 Optimasi training VGG16.....	79
Gambar 4.1.16 Hasil epochs VGG16	80
Gambar 4.1.17 Hasil epochs kedua VGG16.....	81
Gambar 4.1.18 Hasil evaluasi model ResNet50.....	82
Gambar 4.1.19 Hasil Evaluasi model VGG16	83
Gambar 4.1.20 Waktu inferensi ResNet50.....	84
Gambar 4.1.21 Waktu inferensi VGG16.....	85
Gambar 4.1.22 Confusion matrix	85
Gambar 4.1.23 Hasil confusion matrix ResNet50.....	86

Gambar 4.1.24 Hasil confusion matrix VGG16.....	88
Gambar 4.1.25 Classification report	90
Gambar 4.1.26 Classification report ResNet50	91
Gambar 4.1.27 Grafik ResNet50.....	92
Gambar 4.1.28 Classification report VGG16.....	93
Gambar 4.1.29 Grafik VGG16.....	94

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Padi (*Oryza Sativa*) merupakan tanaman pangan yang memiliki peran vital dalam menjaga ketahanan pangan global, khususnya di kawasan Asia. Tanaman padi sangat penting untuk kehidupan orang di Indonesia dan sebagian besar negara Asia lainnya. Beras adalah biji-bijian penting yang secara signifikan berkontribusi pada ketahanan pangan global selama setengah abad terakhir (Udayananda et al., 2022). Beras adalah sumber utama karbohidrat. Nasi telah melekat dalam budaya Indonesia sehingga banyak orang percaya bahwa mereka belum makan jika tidak mengonsumsi nasi, meskipun mereka telah mengonsumsi makanan lain. Fenomena serupa juga terlihat di negara-negara Asia lainnya seperti Tiongkok, India, Jepang, dan Korea, di mana beras adalah bagian penting dari makanan sehari-hari mereka. Selain berkontribusi pada kehidupan masyarakat Asia dalam hal pangan, padi memiliki makna sosial, ekonomi, dan bahkan politik. Ketersediaan dan harga beras sering kali menunjukkan stabilitas sosial dan ekonomi di banyak negara Asia, termasuk Indonesia. Pemerintah di berbagai negara Asia secara aktif mengawasi produksi, distribusi, dan harga beras untuk menjamin ketahanan pangan nasional. Namun pertanian padi menghadapi berbagai tantangan, dan salah satu yang paling signifikan adalah serangan penyakit. Serangan penyakit pada tanaman padi dapat menyebabkan kerugian besar bagi petani dan mengancam ketahanan pangan.

Beras adalah makanan pokok di banyak negara, terutama di Asia, penurunan produksi beras akibat serangan penyakit dapat menyebabkan kelangkaan pasokan

di pasar, yang pada gilirannya menyebabkan kenaikan harga beras. Kenaikan harga ini dapat mempengaruhi daya beli masyarakat, terutama kelompok berpenghasilan rendah. Kelangkaan beras dapat menyebabkan konflik sosial dan ketidakstabilan politik dalam keadaan ekstrim. Dari perspektif kesehatan masyarakat, upaya pengendalian penyakit tanaman yang menggunakan bahan kimia seperti pestisida dapat menimbulkan risiko kesehatan jika tidak dikelola dengan baik. Residu pestisida pada beras dapat membahayakan konsumen, sementara paparan langsung terhadap pestisida dapat membahayakan petani dan pekerja pertanian. Menurut Badan Pusat Statistik luas panen padi pada 2024 diperkirakan sekitar 10,05 juta hektare, mengalami penurunan sebanyak 167,25 ribu hektare atau 1,64 persen dibandingkan luas panen padi di 2023 yang sebesar 10,21 juta hektare. Produksi padi pada 2024 diperkirakan sebesar 52,66 juta ton GKG, mengalami penurunan sebanyak 1,32 juta ton GKG atau 2,45 persen dibandingkan produksi padi di 2023 yang sebesar 53,98 juta ton GKG. Produksi beras pada 2024 untuk konsumsi pangan penduduk diperkirakan sekitar 30,34 juta ton, mengalami penurunan sebanyak 757,13 ribu ton atau 2,43 persen dibandingkan produksi beras di 2023 yang sebesar 31,10 juta ton .

Penyakit pada tanaman padi memiliki dampak yang sangat signifikan terhadap sektor pertanian, khususnya di negara-negara yang mengandalkan padi sebagai tanaman pangan utama. Aspek produksi bukanlah satu-satunya aspek yang terkena dampak ini, itu mencakup aspek ekonomi, sosial, dan bahkan politik. Penyakit pada tanaman padi dapat secara langsung berdampak pada penurunan hasil panen, atau dalam kasus terburuk, kegagalan panen total. Ini tentu saja berdampak langsung pada pendapatan petani, yang sering kali sudah berada dalam

kondisi ekonomi yang tidak stabil. Selain itu, karena beras merupakan makanan pokok bagi sebagian besar penduduk di banyak negara Asia, penurunan produksi padi akibat serangan penyakit dapat mengancam ketahanan pangan nasional.

Penyakit tanaman padi merupakan ancaman besar bagi mata pencaharian petani. Serangan penyakit dapat mengurangi hasil panen secara signifikan, bahkan dapat menyebabkan kegagalan panen total. Permasalahan yang sering muncul adalah banyak tanaman padi yang rentan terhadap serangan hama dan penyakit selama masa tanam (Tamara et al., 2024). Beberapa hama dan penyakit yang dapat menyerang tanaman padi antara lain : *Bacterial leaf blight* (BLB), *False smut*, *Blast disease* dan *Brown spot* . Hal ini berdampak langsung pada pendapatan petani, yang sudah berada dalam kondisi ekonomi yang tidak stabil. Penyakit tanaman menyebabkan kerugian finansial karena tidak hanya berkurangnya hasil panen tetapi juga biaya produksi untuk mengendalikan penyakit, seperti membeli pestisida atau fungisida. Situasi seperti ini dapat menyebabkan petani terlilit hutang atau bahkan meninggalkan pekerjaan mereka. Akibatnya, kelangsungan hidup sektor pertanian padi dapat terancam.

Implementasi *machine learning* merupakan solusi yang tepat untuk mengidentifikasi penyakit pada tanaman padi. Salah satu algoritma yang populer dalam *Deep Learning* adalah *Convolutional Neural Network* (CNN) (Mudzakir & Arifin, 2022). Untuk menganalisa penyakit pada tanaman padi tersebut digunakan algoritma *Convolutional Neural Network* (CNN) menggunakan Arsitektur VGG16 dengan Arsitektur ResNet50. Kedua arsitektur tersebut akan dibandingkan performanya dengan melihat nilai akurasi, *Precision*, *recall*, dan *f1-score* dari

masing – masing model arsitektur. VGG16 adalah sebuah arsitektur *deep learning* yang memiliki 16 layers (Windiawan et al., 2021). VGG16 telah berhasil mencapai tingkat akurasi yang tinggi dalam pengenalan gambar pada berbagai tugas, termasuk dataset *ImageNet*, yang terdiri dari jutaan gambar dari ribuan kelas yang berbeda (Rizki & Marina, 2019). Pada penelitian lain perbandingan performa Arsitektur VGG16 dan VGG19 yang dilakukan oleh (Weny Indah Kusumawati & Adisaputra Zidha Noorizki, 2023) Model VGG16 mencapai tingkat akurasi sebesar 98% dengan waktu *training* selama 73,405 detik, VGG16 menunjukkan performa yang lebih baik dalam hal recall dan f1-score dibandingkan dengan VGG19 yang mencapai akurasi sebesar 97% dengan waktu training selama 78,098 detik. Dalam penelitian ini VGG16 terbukti memiliki performa yang lebih unggul dibandingkan dengan VGG19. Hal ini di perkuat pada penelitian yang dilakukan oleh (Adiningsi & Saputra, 2023.) dengan menggunakan Arsitektur VGG16 Hasil *training* menunjukkan akurasi sebesar 81,61% dan validasi akurasinya 90,74%. Pada uji data 50 citra daun mendapatkan hasil akurasi 92% dengan 46 data yang terklasifikasi dan 4 data citra yang salah terklasifikasi. Dari hasil penelitian terdahulu yang telah dilakukan sebelumnya cukup untuk menunjukkan bahwa Arsitektur VGG16 diklaim baik dalam melakukan klasifikasi citra.

Sementara itu ResNet adalah arsitektur *Convolutional neural network* (CNN) yang menggunakan *skip connections* (koneksi lompatan) yang membantu menyelesaikan masalah gradien yang hilang, sehingga mampu meningkatkan proses pelatihan dan dan pengujian (Memon et al., 2024). ResNet50 adalah pemenang tantangan ImageNet pada tahun 2015 dan umumnya digunakan sebagai tulang punggung untuk berbagai tugas *computer vision*. Model ini digunakan untuk

tugas – tugas klasifikasi gambar, oleh karena itu dipilihnya model ResNet50 ini untuk mengidentifikasi penyakit pada tanaman padi yang diharapkan keandalannya dalam segi akurasi dan performa dapat dibandingkan dengan model VGG16 yang telah disebutkan diatas. Pada penelitian yang dilakukan oleh (Memon et al., 2024a) model ResNet50 yang dilatih mencapai akurasi pelatihan 99,49% dengan *precision* 96,6% dan *F1 score* 99,5% yang dilakukan untuk identifikasi penyakit daun yang berbeda pada tanaman. Hal ini juga diperkuat pada penelitian yang dilakukan oleh (Elpeltagy & Sallam, 2021) pada model yang dilatih untuk memprediksi *COVID-19* arsitektur ResNet50 mencapai nilai *accuracy* 97,7% pada dataset CT Scan dan 97,1% pada dataset X-Ray. Hasil ini menunjukkan bahwa modifikasi yang dilakukan oleh arsitektur ResNet50 kinerjanya meningkat dibandingkan dengan model aslinya.

Penelitian ini berfokus pada klasifikasi multi-kelas pada penyakit tanaman padi diantaranya yaitu bercak coklat (*brown spot disease*), bercak daun (*blast disease*), gosong palsu (*false smut disease*), hawar daun bakteri (*bacterial blast disease*) dan juga daun yang sehat. Terdapat perbedaan pada penelitian sebelumnya seperti pada penelitian yang dilakukan oleh (Habib Hawari et al., 2022) melakukan penelitian mengenai klasifikasi penyakit tanaman padi menggunakan Algoritma CNN menggunakan 3 jenis penyakit daun padi yaitu *Brown spot disease*, *Bacterial blast disease*, *leaf brown*. Adapaun penelitian yang dilakukan oleh (Mannepalli et al., 2024) melakukan penelitian untuk mendiagnosis penyakit pada tanaman padi menggunakan Arsitektur VGG16 dengan 3 kategori antara lain *Bacterial leaf blight*, *Rice blast*, *Brown spot*.

Penelitian ini memilih metode berbasis *Convolutional Neural Network* (CNN) dikutip dari (Habib Hawari et al., 2022) melalui (Oktaviana, et al., 2021) CNN telah dinilai menunjukkan keunggulan yang luar biasa dalam berbagai penerapan di dunia nyata dibandingkan dengan metode machine learning yang lain. Oleh karena itu CNN dipilih kemampuannya yang luar biasa untuk mengekstraksi fitur visual yang kompleks dari gambar daun tanaman padi yang terinfeksi penyakit. Berbeda dengan metode pembelajaran mesin konvensional seperti *Support Vector Machine* (SVM), *K-Nearest Neighbors* (KNN), atau *Random Forest* yang memerlukan ekstraksi fitur secara manual, tetapi CNN mampu belajar secara otomatis untuk mengidentifikasi pola visual yang terkait dengan berbagai jenis penyakit padi.

Arsitektur VGG16 dan ResNet50 dipilih untuk penelitian ini karena kedua model ini memenangkan tantangan *ImageNet* sebelumnya dengan memberikan hasil yang sama dalam hal akurasi prediksi (Dhankhar, n.d). Dengan desainnya yang sederhana, VGG16 menggunakan susunan lapisan konvolusi berurutan 3×3 kecil di seluruh jaringan untuk membangun representasi fitur yang kompleks (Mannepalli et al., 2024). Metode ini memungkinkan VGG16 dengan sangat baik menangkap detail tekstur dan pola lokal, yang sangat penting untuk menemukan gejala penyakit pada daun padi seperti bercak, perubahan warna, atau lesi.

Sementara, ResNet50 menyediakan solusi untuk masalah *vanishing gradient* yang sering terjadi pada jaringan neural yang sangat dalam melalui inovasi *integration of residual connections* atau biasa dikenal sebagai *skip-connections* yang merupakan inti dari ResNet50 (Murimi, 2023). Kemampuan ini

memungkinkan ResNet50 untuk belajar representasi daun padi yang terinfeksi yang lebih abstrak dan kompleks. Ini mungkin menawarkan kinerja yang lebih baik untuk situasi di mana perbedaan antar jenis penyakit sangat kecil atau ketika diperlukan pemahaman kontekstual yang lebih luas dari citra secara keseluruhan.

Ketersediaan model *pre-trained* yang merupakan model yang telah dilatih sebelumnya yang sudah ada dalam kumpulan yang besar seperti *ImageNet* (Memon et al., 2024). *ImageNet* merupakan dataset yang biasa digunakan untuk melatih arsitektur deep learning yang terdiri dari jutaan gambar dari 1000 kelas yang berbeda (Rismiyanti & Luthfiarta, 2021). Mengingat keterbatasan jumlah data penyakit tanaman padi yang tersedia untuk pelatihan melalui *transfer learning*, kemampuan untuk mengidentifikasi pola dasar yang sudah dipelajari dari jutaan gambar umum dapat disesuaikan untuk kasus khusus identifikasi penyakit tanaman padi yang diharapkan dapat meningkatkan efisiensi pelatihan dan potensi performa model.

Perbandingan antara kedua arsitektur ini sangat penting untuk industri pertanian presisi teknologi yang semakin berkembang di Indonesia. Sangat penting untuk memiliki sistem deteksi penyakit otomatis yang tepat dan efektif mengingat pentingnya padi sebagai sumber makanan utama. Pada akhirnya, ini dapat menghasilkan peningkatan produktivitas pertanian padi di seluruh negara. Studi sebelumnya telah menunjukkan bahwa kedua arsitektur ini bekerja dengan baik dalam berbagai kasus klasifikasi citra.

Atas dasar pentingnya deteksi dini penyakit tanaman padi, maka penelitian ini membandingkan kinerja Arsitektur VGG16 dengan ResNet50 dapat membantu

memahami sejauh mana keunggulan masing-masing model dalam klasifikasi citra penyakit tanaman padi. Analisis ini bertujuan untuk memberikan wawasan yang signifikan tentang akurasi, efisiensi, dan waktu komputasi kedua Arsitektur tersebut, sehingga dapat membantu para peneliti, petani, dan praktisi teknologi. Dari fenomena tersebut maka penulis mengangkat topik penelitian yang berjudul **“ANALISIS PERBANDINGAN PERFORMA ARSITEKTUR VGG16 DAN RESNET50 DALAM KLASIFIKASI MULTI-KELAS PENYAKIT TANAMAN PADI BERBASIS CONVOLUTIONAL NEURAL NETWORK (CNN)”** Penelitian ini berkonsentrasi pada perbandingan dua algoritma yang populer di bidang *deep learning* untuk mengklasifikasikan gambar penyakit padi: VGG16 dan ResNet50.

1.2 Rumusan Masalah

Berdasarkan uraian dari latar belakang masalah, maka rumusan masalah yang menjadi pokok bahasan dalam penelitian ini adalah Bagaimana perbandingan performa arsitektur VGG16 dalam klasifikasi multi-kelas pada penyakit tanaman padi dengan arsitektur ResNet50 berbasis *Convolutional neural network* (CNN).

1.3 Batasan Masalah

Adapun batasan masalah dalam melakukan penelitian ini adalah :

1. Data citra penyakit tanaman padi bersumber dari *Kaggle* rentang tahun 2023 – 2024.
2. Jenis data citra penyakit padi yang digunakan yaitu bercak coklat (*brown spot disease*), bercak daun (*blast disease*), gosong palsu (*false smut*

disease), hawar daun bakteri (*bacterial blast disease*), dan daun sehat dengan pencahayaan normal.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk :

1. Mengetahui kinerja arsitektur VGG16 dan ResNet50 dalam mengklasifikasikan jenis-jenis penyakit pada tanaman padi menggunakan metode *Convolutional Neural Network* (CNN).
2. Membantu menemukan arsitektur yang paling efektif dan efisien dalam klasifikasi penyakit pada tanaman padi.
3. Mengetahui perbandingan metrik evaluasi seperti akurasi, presisi, recall, F1-score, dan waktu pemrosesan.
4. Membuat model *deep learning* yang lebih efisien untuk pengelolaan penyakit tanaman padi.

1.5 Manfaat Penelitian

Adapun manfaat dari penelitian ini yaitu :

1. Memberikan informasi tentang kinerja model VGG16 dan ResNet50 dalam mengenali jenis penyakit pada tanaman padi.
2. Membantu menemukan model CNN yang efektif dan efisien untuk mendeteksi penyakit pada tanaman padi.
3. Memberikan informasi perbandingan akurasi, presisi, recall, f1-score dan waktu pemrosesan sebagai dasar untuk memilih model yang paling baik dalam mendeteksi penyakit tanaman padi.

4. Menghasilkan model *deep learning* yang dapat membantu petani mengelola penyakit tanaman padi dengan lebih cepat, akurat dan efisien.

BAB II

LANDASAN TEORI

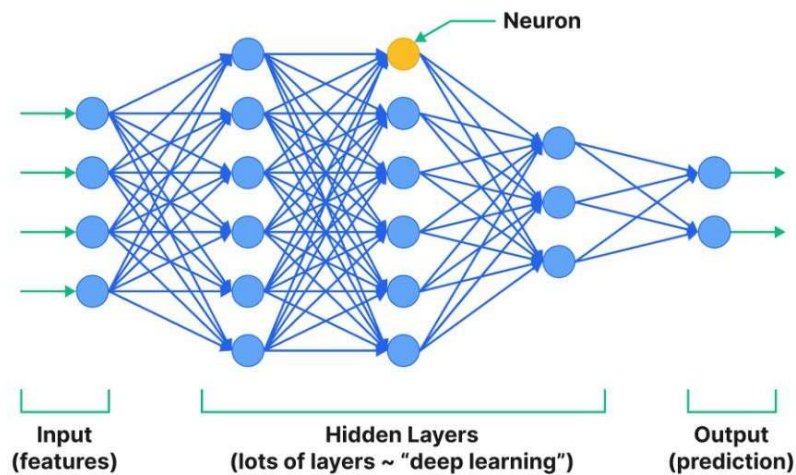
2.1 Deep Learning

Deep learning adalah sebuah cabang dari *machine learning* dan kecerdasan buatan yang saat ini dianggap sebagai teknologi inti dari Revolusi Industri Keempat (Industri 4.0) (Sarker, 2021). Jaringan saraf tiruan dalam *deep learning* terdiri dari banyak lapisan tersembunyi yang memungkinkan sistem untuk menangkap pola kompleks dalam data, baik itu gambar, suara, atau teks yang memungkinkan komputer untuk belajar dari banyak data dan membuat keputusan otomatis tanpa instruksi yang jelas dari manusia. Karena kemampuannya untuk belajar dari data, teknologi *Deep Learning* yang berasal dari *Artificial neural network* (ANN), telah menjadi topik hangat dalam konteks komputasi, dan diterapkan secara luas di berbagai bidang aplikasi seperti kesehatan, pengenalan visual, analisis teks, *cyber security*, dan masih banyak lagi.

Deep learning telah berkembang secara signifikan, terutama dalam menangani menangani *dataset* gambar dan mencapai akurasi yang tinggi. Dalam konteks *deep learning*, CNN memiliki kemampuan untuk mengidentifikasi fitur-fitur penting secara mandiri selama fase pelatihan, mengurangi ketergantungan pada campur tangan manusia untuk ekstraksi fitur (Sarker, 2021). *Deep Learning* telah menerima banyak perhatian dalam beberapa tahun terakhir yang lebih luas dalam kategori *Machine Learning*.

Tujuan *deep learning* adalah untuk memungkinkan pembelajaran dari ujung ke ujung, mencapai generalisasi yang kuat, meningkatkan akurasi, menskala kan *dataset* yang besar, dan mengotomatiskan ekstraksi fitur. Tujuan utamanya adalah

membuat sistem cerdas yang kuat yang dapat melakukan tugas-tugas dunia nyata yang kompleks dalam berbagai bidang seperti pemrosesan bahasa alami, pemrosesan gambar, pengenalan suara, dan *computer vision*. Secara umum *deep learning* memiliki tiga jenis struktur dari layer utama, berikut gambarannya :



Gambar 2.1 Struktur Deep Learning

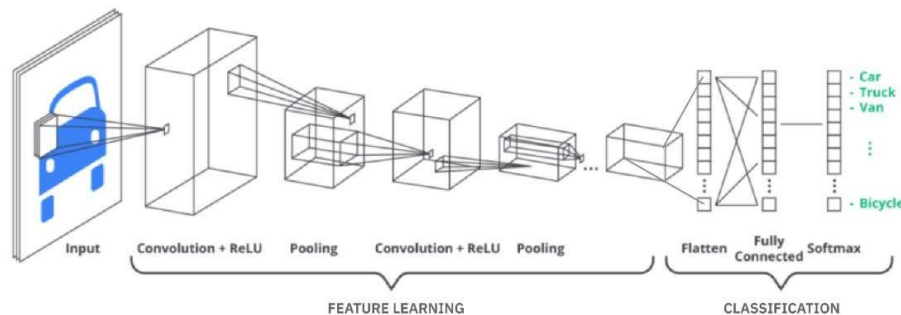
Sumber : <https://www.dicoding.com>

Struktur dari layer pertama dalam *deep learning* adalah *Input Layer* atau lapisan *input* yang berfungsi untuk menerima data atau *input* dari luar. Setiap neuron yang ada pada *input* layer merepresentasikan satu variabel atau fitur dari data yang masuk ke jaringan. Selanjutnya *Hidden Layer* atau lapisan tersembunyi terdiri dari neuron yang saling berhubungan. Neuron-neuron dalam *hidden layer* memproses input yang diterima dari *layer* sebelumnya dengan melakukan operasi matematika menggunakan bobot (*weights*) dan fungsi aktivasi tertentu. *Hidden layer* berfungsi untuk penyaring fitur yang membantu jaringan dalam mempelajari pola – pola yang kompleks pada data.

Output layer atau layer output adalah layer terakhir dalam *deep learning* yang berfungsi untuk menghasilkan *output* yang diproses sebelumnya oleh *hidden layer*. Jumlah neuron yang ada pada *output layer* bergantung pada tugas yang ingin diselesaikan. Dan untuk hasil akhir terdapat *loss-layer* yaitu komponen terakhir dalam jaringan syaraf yang berfungsi untuk menghitung kesalahan antara prediksi model dengan label yang sebenarnya.

2.1 Convolutional Neural Network (CNN)

Convolutional neural network (CNN) adalah jaringan syaraf yang dibuat untuk memproses data berbentuk gambar. CNN menggunakan lapisan konvolusi untuk mengekstrak fitur dari gambar. Menurut (Kairul Azmi, 2023), *Convolutional Neural Network* (CNN) adalah salah satu jenis arsitektur *neural network* yang biasa digunakan pada data *image*. CNN terdiri dari 2 lapisan arsitektur yaitu *Feature Learning* dan *Classification layer* berikut terlihat pada gambar dibawah :



Gambar 2.2 Arsitektur Convolutional Neural Network

Sumber : <https://medium.com>

Pada proses *feature learning* di mana *Convolutional neural network* (CNN) mengekstraksi pola-pola penting dari data masukan. Pada tahap ini terdapat

beberapa lapisan utama, termasuk *Convolution*, *ReLU*, dan *Pooling*. Lapisan *Convolution* mengidentifikasi pola sederhana, seperti garis atau tepi pada gambar, dengan fungsi aktivasi *ReLU* menghilangkan nilai negatif, sehingga jaringan hanya mempertahankan informasi yang relevan. Menggabungkan data juga mengurangi dimensi data, meningkatkan efisiensi jaringan, dan mengurangi sensitivitas terhadap perubahan posisi fitur dalam gambar. Proses ini diulang secara bertahap sehingga jaringan dapat mengenali pola yang lebih kompleks, seperti bentuk atau komponen tertentu dari suatu objek.

Tahap *classification* adalah tahap di mana fitur yang telah diekstraksi digunakan untuk menentukan kelas data *input*. Proses ini dimulai dengan lapisan *Flatten*, yang mengubah peta fitur tiga dimensi menjadi vektor satu dimensi yang dapat diproses lebih lanjut. Vektor ini kemudian dilanjutkan ke lapisan *Fully Connected*, yang menggabungkan data dari semua fitur untuk membantu membuat keputusan akhir. Akhirnya, lapisan *Softmax* menghasilkan probabilitas untuk setiap kelas. Kelas dengan probabilitas tertinggi digunakan sebagai hasil prediksi. Proses ini memastikan bahwa karakteristik yang telah dipelajari diolah menjadi klasifikasi yang tepat dan spesifik. Bagian dari lapisan CNN dijelaskan pada tabel 2.

Tabel 2.1 Algoritma Convolutional Neural Network

Tahap	Proses	Tujuan
<i>Input layer</i>	Data masukan berupa gambar dalam bentuk matriks piksel (contoh: RGB atau <i>grayscale</i>).	Memberikan data awal yang akan diproses oleh jaringan.

<i>Convolution layer</i>	Aplikasi filter (<i>kernels</i>) pada gambar untuk mengekstraksi fitur sederhana seperti tepi atau pola.	Mendapatkan peta fitur awal dari data masukan.
<i>Activation (ReLU)</i>	Fungsi aktivasi <i>Rectified Linear Unit</i> menghilangkan nilai negatif dalam peta fitur dengan mengubahnya menjadi nol.	Membuat jaringan lebih efisien dengan mempertahankan nilai-nilai positif.
<i>Pooling</i>	Mengurangi dimensi peta fitur melalui metode seperti <i>Max Pooling</i> (mengambil nilai maksimum dalam area kecil).	Mengurangi ukuran data sambil mempertahankan informasi penting.
<i>Iterasi</i>	Proses <i>Convolution</i> , <i>ReLU</i> , dan <i>Pooling</i> dilakukan beberapa kali untuk mengekstraksi fitur yang semakin kompleks.	Membentuk representasi fitur yang lebih tinggi (misalnya, objek lengkap).
<i>Flatten</i>	Peta fitur berdimensi tiga diubah menjadi vektor satu dimensi.	Menyiapkan data untuk tahap klasifikasi.
<i>Fully connected</i>	Lapisan neuron yang menghubungkan semua fitur yang telah diekstraksi, menggabungkan informasi untuk membuat prediksi.	Mengolah fitur menjadi representasi akhir untuk prediksi kelas.
<i>Softmax</i>	Fungsi aktivasi yang menghasilkan probabilitas untuk setiap kelas.	Menentukan kelas akhir berdasarkan probabilitas tertinggi.
<i>Output</i>	Keluaran berupa prediksi kelas (contoh: mobil, truk, sepeda).	Memberikan hasil akhir klasifikasi gambar.

a. Input Layer

Lapisan pertama jaringan saraf bertanggung jawab untuk menerima data, seperti teks atau gambar, dalam bentuk yang diatur. Data yang dimasukkan ke dalam CNN berupa matriks piksel; contohnya, matriks dua dimensi untuk gambar *grayscale* atau matriks tiga dimensi untuk gambar berwarna (dengan saluran RGB). Lapisan ini hanya mengirimkan data ke lapisan berikutnya, seperti lapisan *convolution*, tetapi tidak melakukan perhitungan atau pembelajaran.

b. Convolution layer

Pada lapisan ini, filter (atau kernel) digunakan untuk mengidentifikasi pola sederhana seperti tepi, garis, atau tekstur pada data yang dimasukkan. Proses dimulai dengan menggerakkan atau mengkonvolusi filter melalui gambar *input*. Setiap kali filter "melalui" bagian gambar, ia menghitung nilai titik produk antara filter dan area gambar yang sesuai. Nilai ini disimpan dalam peta fitur, juga dikenal sebagai *feature map*. *Feature Map* dapat yang akan dihasilkan didapat dari rumus berikut.

$$n_{out} = \binom{n_{in}-k+2p}{s} + 1 \quad (2.1)$$

Keterangan :

n_{out} : Ukuran dari *Feature Map*.

n_{in} : Ukuran dari matrix masukan.

k : Ukuran dari matrix filter.

p : Ukuran padding

s : *stride*

Berikut adalah rumus operasi konvolusi:

$$FM[i]_{j,k} = (\sum_m \sum_n N_{[j-m,k-n]} F_{[m,n]}) + bF \quad (2.2)$$

Keterangan :

$FM[i]_{j,k}$: Matrix dari *feature map* ke - i.

N : Matrix dari citra masukan.

F : Matrix dari filter konvolusi.

bF : Nilai bias pada filter.

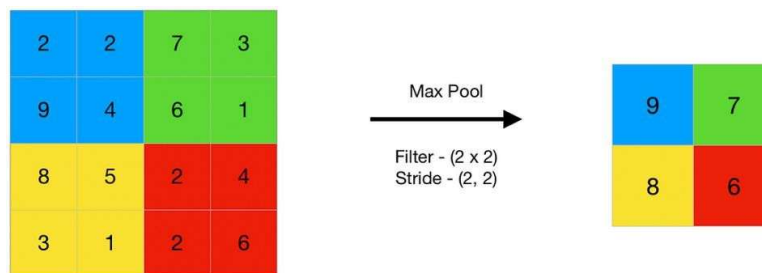
j, k : Posisi pixel pada matrix citra masukan.

m, n : Posisi pixel pada matrix filter konvolusi.

Setelah proses konvolusi selesai, langkah selanjutnya adalah menerapkan fungsi aktivasi menggunakan *Rectified Linear Unit* (ReLU). Pada tahap ini, fungsi ReLU akan memproses setiap piksel pada *feature map*, dengan mengubah nilai piksel yang kurang dari 0 menjadi 0, sementara nilai lainnya tetap dipertahankan. Fungsi ini dapat ditulis secara matematis sebagai $f(x) = \max(0, x)$. Perkenalan non-linearitas pada model adalah tujuan dari prosedur ini, yang akan memungkinkan jaringan untuk mempelajari representasi data yang lebih kompleks.

c. Pooling Layer

Fungsi dari *pooling layer* adalah untuk secara progresif mengurangi ukuran spasial representasi untuk mengurangi jumlah parameter dan komputasi dalam jaringan, dan karenanya juga mengontrol *overfitting*. Tidak ada pembelajaran yang terjadi pada lapisan penyatuan (LeChun, 1989). *Pooling Layer* digunakan untuk mengurangi ukuran *feature map*. Jenis *pooling* yang digunakan adalah *max pooling*, yang berarti memilih nilai maksimum pada suatu jendela tertentu. Proses ini mirip dengan *layer convolution*, yang menggeser jendela ke seluruh permukaan gambar, tetapi menggunakan jendela sebagai referensi untuk memilih nilai maksimum pada suatu area tertentu. *Feature map* dengan nilai maksimum yang dipilih akan dihasilkan sebagai *output* dari prosedur ini.



Gambar 2. 3 Max Pooling

d. Fully-connected layer

Fully-connected layer menggabungkan fitur yang diekstraksi dari lapisan sebelumnya, mengoptimalkan hubungan antar fitur, dan membuat keputusan akhir seperti regresi atau klasifikasi. Semua neuron di lapisan sebelumnya terhubung ke setiap neuron, memungkinkan model untuk

mempelajari pola yang kompleks. Selain itu, lapisan *fully-connected* berfungsi sebagai titik transisi ke lapisan *output*, di mana prediksi akhir dibuat berdasarkan data yang telah dipelajari.

e. Hidden layer

Proses ***hidden layer*** meliputi menerima *input* dari lapisan sebelumnya, menghitung kombinasi linear dengan bobot dan bias, menerapkan fungsi aktivasi untuk menangani non-linearitas, dan mengirimkan hasilnya ke lapisan berikutnya. *Hidden layer* mengolah data untuk mengekstraksi fitur kompleks dan menghasilkan representasi yang lebih informatif. Adapun rumus perhitungannya sebagai berikut :

$$z_in_i = \sum_{j=1}^n X_j * V_{j,i} + V_{0,i} \quad (2.3)$$

Dimana :

z_in_i : Masukan untuk node hidden layer ke-i dengan jumlah node n

X_j : Node X ke-j.

$V_{j,1}$: Bobot V untuk X_j dan node Z_i .

V_0 : Bias V untuk z_in_i .

Setelah perhitungan selesai, masukkan fungsi aktivasi ReLU untuk semua hasil perhitungan, dan nilai keluaran Z akan dihasilkan.

f. Output layer

Setelah proses di *hidden layer* selesai, *output* dikirim ke lapisan berikutnya atau ke lapisan *output* untuk menghasilkan prediksi. Adapun rumus perhitungannya sebagai berikut :

$$y_{in_i} = \sum_{j=1}^m Z_j * W_{j,i} + W_{o,i} \quad (2.4)$$

Dimana :

y_{in_i} :Masukan untuk node hidden layer Z ke-I dengan jumlah node m.

Z_j : node Z ke-j.

$W_{j,i}$: Bobot W untuk Z_j dan node Y_i .

W_0 : Bias W untuk y_{in_i} .

Setelah perhitungan selesai, masukkan fungsi aktivasi *Softmax* untuk semua hasil untuk mendapatkan nilai keluaran Y. Rumus yang digunakan untuk menghitung fungsi *softmax* adalah sebagai berikut :

$$Y_i = \frac{e^{y - in_i}}{\sum_{i=1}^m e^M} \quad (2.5)$$

Dimana :

Y_i : Keluaran untuk output layer ke-i.

$y - in_i$: Masukan untuk node layer ke-i.

M : Semua masukan untuk output layer sejumlah m .

2.2.1 Backpropagation

Dalam pelatihan jaringan saraf, *backpropagation* adalah algoritma yang digunakan untuk mengoptimalkan bobot dan bias dengan meminimalkan kesalahan prediksi. Algoritma ini bekerja dengan menghitung gradien dari fungsi kehilangan terhadap parameter model (bobot dan bias). Kemudian, menggunakan teknik penurunan gradien atau variasinya, parameter tersebut diperbarui.

a. Forward pass

Pada tahap forward pass, *input* X diteruskan melalui jaringan untuk menghasilkan output prediksi. Misalnya, jika X adalah vektor input dan z_i adalah output dari neuron ke- i di layer output, maka output dari neuron tersebut dapat dihitung sebagai:

$$Z_i = \sum_j W_{ji} x_j + b_i \quad (2.6)$$

Dimana :

W_{ji} : bobot yang menghubungkan neuron j di layer sebelumnya dengan neuron i di layer output,

b_i : bias pada neuron ke- i ,

x_j : output dari neuron sebelumnya.

Untuk mendapatkan probabilitas kelas, setelah menghitung z_i fungsi aktivasi softmax diterapkan pada output:

$$Y_i = \frac{e^{z_i}}{\sum_k e^{z_k}} \quad (2.7)$$

Di mana Y_i adalah kemungkinan untuk kelas ke- i , dan z_k adalah output logit untuk semua kelas. Pada tahap ini, jaringan memperkirakan kemungkinan untuk setiap kelas.

b. Menghitung loss (fungsi loss)

Setelah mendapatkan *output* probabilitas Y_i dari softmax, langkah berikutnya adalah menghitung loss atau kesalahan prediksi menggunakan fungsi categorical cross-entropy :

$$L = - \sum_i t_i \log(Y_i) \quad (2.8)$$

Dimana :

- Nilai target kelas ke- i adalah t_i , yang merupakan satu-hot encoding label kelas yang benar. Dengan kata lain, jika kelas benar adalah kelas 2, maka $t_2 = 1$, dan jika kelas lain adalah $t_i = 0$.
- Y_i adalah probabilitas prediksi untuk kelas ke- i .

c. Backward pass

Setelah melakukan forward pass dan menghitung *loss*, langkah berikutnya adalah backward pass. Algoritma *backpropagation* digunakan untuk menghitung gradien dari fungsi *loss* terhadap semua bobot dan bias dalam

jaringan. *Chain rule*, atau aturan rantai, digunakan dalam proses ini untuk menyebarkan gradien kesalahan dari lapisan *output* ke lapisan sebelumnya.

- Menghitung gradien *loss* (Softmax)

Gradien *loss* pada *output* neuron Y_i pada layer output yaitu :

$$\frac{\partial L}{\partial Y_i} = Y_i - t_i \quad (2.9)$$

Dimana :

L adalah nilai loss (categorical cross-entropy),

Y_i adalah output dari probabilitas softmax untuk kelas ke- i ,

t_i adalah label untuk target kelas ke- i (one-hot-encoding).

Turunan fungsi *categorical cross-entropy* menghasilkan rumus ini:

$$L = - \sum_i t_i \log(Y_i) \quad (2.10)$$

Ketika diturunkan terhadap Y_i hasilnya adalah:

$$\frac{\partial L}{\partial Y_i} = - \frac{t_i}{Y_i} \quad (2.11)$$

Namun, ketika *softmax* digunakan sebagai fungsi aktivasi output, turunan sederhana dihasilkan oleh kombinasi *softmax* dan *cross-entropy*:

$$\frac{\partial L}{\partial z_i} = Y_i - t_i \quad (2.12)$$

Di mana z_i adalah nilai sebelum softmax.

- Menghitung gradien terhadap *logit* (z_i)

Sebelum *softmax*, hasil perhitungan linear adalah *logit* (z_i) :

$$z_i = \sum_j W_{ij} x_j + b_i \quad (2.13)$$

Turunan *loss* L terhadap *logit* z_i yaitu :

$$\frac{\partial L}{\partial z_i} = Y_i - t_i \quad (2.14)$$

Gradien ini menunjukkan seberapa besar pengaruh kesalahan prediksi pada *logit*.

- Menghitung gradien terhadap bobot (W_{ji})

Untuk menghitung gradien *loss* terhadap bobot (W_{ji}), akan digunakan chain rule :

$$\frac{\partial L}{\partial W_{ji}} = \frac{\partial L}{\partial z_i} \cdot \frac{\partial z_i}{\partial W_{ji}} \quad (2.15)$$

Dimana :

Turunan loss L terhadap logit z_i adalah hasil dari langkah sebelumnya,

Output dari neuron dilayer sebelumnya x_j .

Substitusi kedua nilai akan menghasilkan :

$$\frac{\partial L}{\partial W_{ji}} = (Y_i - t_i) \cdot x_j \quad (2.16)$$

Artinya :

Mengalikan *error* pada *output* ($Y_i - t_i$) dengan nilai input neuron sebelumnya x_j untuk menghitung gradien loss terhadap bobot W_{ji} .

- Menghitung gradien Bias (b_i)

Metode yang serupa digunakan untuk menghitung gradien *loss* terhadap bias b_i :

$$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial z_i} \cdot \frac{\partial z_i}{\partial b_i} \quad (2.17)$$

Dimana :

$$\frac{\partial L}{\partial z_i} = Y_i - t_i,$$

$\frac{\partial z_i}{\partial b_i} = 1$, karena bias tidak bergantung dengan variabel yang lain.

Hasilnya :

$$\frac{\partial L}{\partial x_j} = Y_i - t_i \quad (2.18)$$

d. Proses hidden layer

Dengan menggunakan *chain rule* atau aturan rantai, proses *backward pass* menyebarkan gradien ke lapisan sebelumnya jika terdapat *hidden layer* :

$$\frac{\partial L}{\partial z_j} = \sum_i \frac{\partial L}{\partial z_i} \cdot W_{ji} \cdot \sigma'(z_j) \quad (2.19)$$

Dimana :

$\frac{\partial L}{\partial z_i}$ adalah gradien pada layer output,

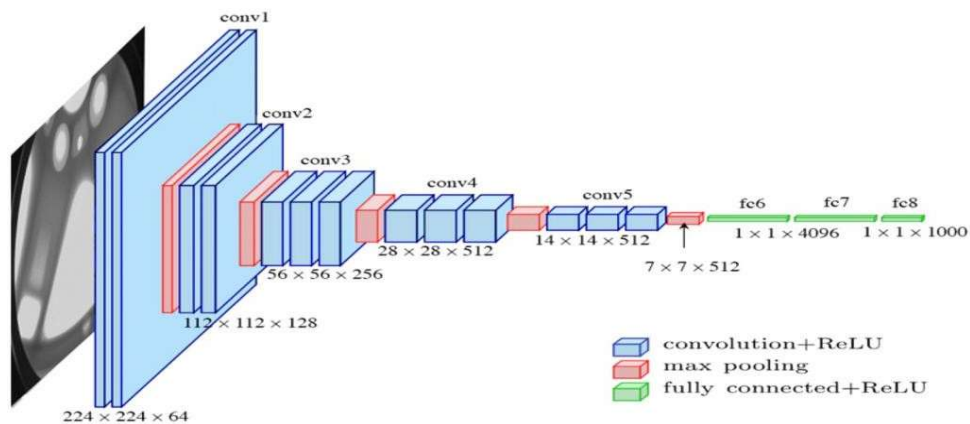
W_{ji} adalah bobot yang menghubungkan neuron j (hidden layer) ke neuron i (output layer),

$\sigma'(z_j)$ adalah turunan fungsi aktivasi di hidden layer.

2.3 Visual Geometry Group (VGG16)

VGG16 adalah salah satu arsitektur jaringan konvolusi dalam (Convolutional Neural Network atau CNN) yang dirancang untuk pengenalan gambar dan klasifikasi objek. Tim *Visual Geometry Group* di Universitas Oxford mengembangkan VGG16. Dipresentasikan pada tahun 2014 dalam makalah ilmiah berjudul "Very Deep Convolutional Networks for Large-Scale Image Recognition" (Simonyan et al., 2014). *Simoyan dan Zisserman* menciptakan model ini

mengevaluasi jaringan dan meningkatkan kedalamannya. Gambar dilewatkan melalui tumpukan *convolution layer* menggunakan filter yang sangat kecil (3x3) yang menunjukkan peningkatan yang signifikan pada konfigurasi sebelumnya (Simoyan et al., 2014). Berikut arsitektur dari VGG16 :



Gambar 2.4 Arsitektur VGG16

Arsitektur VGG16 terdiri dari lapisan konvolusi, *pooling*, dan seluruhnya terhubung yang dimaksudkan untuk pengenalan gambar. Pada langkah awal, jaringan menerima gambar berukuran $224 \times 224 \times 3$, yang menggambarkan gambar berwarna dengan tiga saluran RGB. *Input* ini kemudian diproses melalui lima blok utama, masing-masing terdiri dari beberapa lapisan konvolusi diikuti dengan satu lapisan *max pooling*. Lapisan konvolusi menggunakan kernel 3×3 dengan *padding* yang sama untuk mempertahankan dimensi spasial, sementara lapisan *pooling* menggunakan *kernel* 2×2 dengan *stride* 2 untuk menghasilkan gambar berwarna dengan tiga saluran RGB.

Dalam blok pertama (conv1), terdapat dua lapisan konvolusi dengan 64 filter, yang menghasilkan *output* $112 \times 112 \times 64$ setelah *pooling* maksimum. Di blok kedua (conv2), terdapat dua lapisan konvolusi dengan 128 filter, yang

menghasilkan *output* $56 \times 56 \times 128$. Di blok ketiga (conv3), terdapat tiga lapisan konvolusi dengan 256 filter, dan outputnya setelah *pooling* adalah $28 \times 28 \times 256$. Blok keempat, atau conv4, memiliki tiga lapisan konvolusi dengan 512 filter, yang menghasilkan *output* $14 \times 14 \times 512$. Blok kelima, atau conv5, juga memiliki tiga lapisan konvolusi dengan 512 filter, dan setelah digabungkan, menghasilkan *output* berukuran $7 \times 7 \times 512$.

Setelah melewati lima blok ini, hasilnya diratakan (flattening) menjadi vektor berukuran 1×25088 yang kemudian diproses oleh tiga lapisan yang benar-benar terhubung (FC). Lapisan pertama (FC6) dan kedua (FC7) memiliki 4096 neuron dengan fungsi aktivasi ReLU, dan lapisan terakhir (FC8) memiliki 1000 neuron dengan fungsi aktivasi Softmax, yang menghasilkan kemungkinan 1000 kelas, seperti yang terlihat pada dataset *ImageNet*.

Untuk memperkenalkan non-linearitas, fungsi aktivasi ReLU digunakan pada seluruh lapisan konvolusi dan penuh terhubung. Namun, untuk klasifikasi multi-kelas, lapisan *output* menggunakan *Softmax*. Arsitektur ini memungkinkan VGG16 mengekstraksi fitur gambar secara bertahap, mulai dari pola sederhana hingga kompleks, menjadikannya salah satu arsitektur yang paling populer untuk tugas pengenalan gambar.

a. Convolution layer

Pada lapisan konvolusi, mengalikan filter dengan *input* dan menjumlahkan hasilnya untuk setiap posisi piksel adalah tugas yang dilakukan. Rumus konvolusi adalah sebagai berikut:

$$Z(i, j, k) = \sum_{m=0}^{H-1} \sum_{n=0}^{W-1} \sum_{c=0}^{C-1} X(i+m, j+n, c) \cdot W(m, n, c, k) + b_k \quad (2.20)$$

Dimana :

- $Z(i, j, k)$ adalah nilai output pada posisi (i, j) untuk filter ke- k ,
- $X(i+m, j+n, c)$ adalah nilai pada posisi $(i+m, j+n)$ dari input gambar pada saluran ke- c ,
- $W(m, n, c, k)$ adalah bobot filter yang memiliki ukuran $H \times W \times C$, Dimana H adalah tinggi filter, W adalah lebar filter, dan C adalah jumlah saluran input
- b_k adalah bias untuk filter ke- k .
- H, W adalah ukuran kernel filter (VGG16, ini adalah 3×3).
- C adalah saluran pada input. Misalnya : 3 untuk gambar RGB.

Untuk memperkenalkan non-linearitas, sebuah aktivasi (misalnya fungsi ReLU) diterapkan setelah menghitung hasil konvolusi.

b. Max Pooling

Pada lapisan *max pooling*, tugas yang dilakukan adalah menentukan nilai maksimum dari area tertentu, biasanya berukuran 2×2 , untuk mengurangi dimensi spasial, yaitu lebar dan tinggi gambar. Rumus *max pooling* adalah:

$$Z(i, j, k) = \max_{m,n} \{X(s \cdot i + m, s \cdot j + n, k)\} \quad (2.21)$$

Dimana :

- $Z(i, j, k)$ adalah nilai output pooling pada posisi (i, j) untuk saluran ke- k .
- $X(s \cdot i + m, s \cdot j + n, k)$ adalah nilai input pada area pooling.
- s adalah stride pooling.

- m, n adalah indeks dalam area pooling.

Pooling membuat dimensi spasial lebih kecil, memungkinkan jaringan untuk lebih berkonsentrasi pada karakteristik utama daripada detail spasial yang sangat kecil.

c. Fully connected layer

Output dari lapisan konvolusi dan *pooling* akan diratakan menjadi vektor dan diproses oleh *fully connected layer*. Di sini, proses yang dilakukan adalah perkalian matriks antara vektor *input* dan bobot; kemudian, bias ditambahkan:

$$Z = W \cdot X + b \quad (2.22)$$

Dimana :

- Z adalah output dari lapisan fully connected.
- W adalah bobot matrix dengan dimensi $(N_{output} \times N_{input})$ dimana N_{output} adalah jumlah neuron dilapisan matrix tersebut dan N_{input} adalah panjang input.
- X adalah vektor input dari hasil lapisan sebelumnya yang telah diratakan.
- b adalah vektor bias.

d. Fungsi Softmax

Setelah skor dihitung oleh lapisan yang penuh terhubung, fungsi *Softmax* digunakan untuk mengkonversi skor menjadi probabilitas yang berkisar antara 0 dan 1. Rumusnya adalah sebagai berikut:

$$P(y = i | X) = \frac{\exp(Z_i)}{\sum_{j=1}^C \exp(Z_j)} \quad (2.23)$$

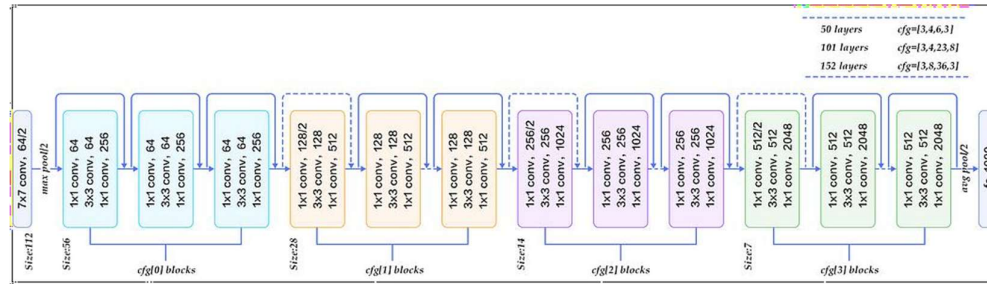
Dimana :

- $P(y = i | X)$ adalah probabilitas gambar X termasuk dalam kelas i .
- Z_i adalah skor untuk kelas i .
- C adalah jumlah total kelas (1000 untuk ImageNet).
- $\exp(Z_i)$ adalah fungsi eksponensial diterapkan pada skor untuk setiap kelas.

Fungsi *Softmax* memudahkan pemilihan kelas dengan prediksi probabilitas tertinggi karena memastikan bahwa hasil model adalah probabilitas totalnya adalah 1.

2.4 Residual Network (ResNet50)

ResNet50 adalah model pembelajaran mendalam yang menggunakan koneksi lompatan untuk mengatasi masalah gradien yang hilang, yang dapat menghambat pelatihan model pembelajaran mendalam (Suleman et al., 2024). ResNet50, versi 50 lapisan dari jaringan ResNet, diperkenalkan pada tahun 2015 oleh *Kaiming He* dan rekannya dalam paper mereka "Deep Residual Learning for Image Recognition". ResNet adalah salah satu arsitektur jaringan saraf dalam yang sangat populer dan kuat untuk tugas pengenalan gambar. Berikut arsitektur dari ResNet50 :



Gambar 2.5 Arsitektur ResNet50

ResNet50 adalah versi *Residual Networks* yang memiliki lima puluh lapisan. Dalam model ini, *bottleneck* terdiri dari blok *residual* yang terdiri dari tiga lapisan konvolusi: 1×1 *convolution* untuk mengurangi dimensi, 3×3 *convolution* untuk mengekstraksi fitur, dan 1×1 *convolution* untuk mengembalikan dimensi. Setiap blok memiliki *shortcut connection* yang memungkinkan input melompati beberapa lapisan, yang mempermudah pemahaman fungsi *residual* dan menghindari masalah *vanishing gradient*.

Arsitektur ResNet50 terdiri dari empat tahap utama. Tahap pertama memiliki tiga blok *residual*, tahap kedua memiliki empat blok, tahap ketiga memiliki enam blok, dan tahap keempat memiliki tiga blok. Setiap langkah dimulai dengan *convolution* atau *pooling*, yang mengurangi dimensi ruang. Untuk menghasilkan vektor dengan probabilitas 1000 kelas, penggabungan rata-rata global digunakan untuk mengubah ukuran *output* menjadi $1 \times 1 \times 2048$ $1 \times 1 \times 2048$. ResNet50 dibuat untuk klasifikasi gambar berskala besar dengan parameter yang efisien dan akurat, mirip dengan kumpulan ImageNet.

a. Residual Block

ResNet, menggunakan koneksi *residual* untuk mengatasi masalah ketika jaringan terlalu dalam. Dalam kasus seperti ini, *Vanishing Gradient* Gradien menjadi sangat kecil, sehingga sulit untuk memperbarui bobot selama pelatihan. Meskipun jumlah lapisan bertambah, akurasi model menurun.

$$H(x) = F(x) + x \quad (2.24)$$

Dimana :

- $H(x)$ adalah output akhir dari residual block.
- $F(x)$ adalah transformasi fitur hasil dari operasi seperti convolution, Batch Normalization, dan ReLU.
- x adalah input awal.

b. Bottleneck Block

ResNet50 menggunakan *Bottleneck block* dengan 3 lapisan konvolusi diantaranya :

1. *Convolution* 1 x 1 : memiliki fungsi untuk mengurangi jumlah *channel* atau dimensi fitur.
2. *Convolution* 3 x 3 : memiliki fungsi untuk mengekstrak fitur dari data.
3. *Convolution* 1 x 1 : memiliki fungsi untuk mengembalikan jumlah *channel* ke ukuran semula.

$$F(x) = W_3 \cdot \text{ReLU}(\text{BN}(W_2 \cdot \text{ReLU}(\text{BN}(W_1 \cdot X)))) \quad (2.25)$$

Dimana :

- W_1, W_2, W_3 adalah matrix bobot untuk konvolusi 1 x 1, 3 x 3, 1 x 1.
- BN adalah Batch Normalization.
- ReLU adalah fungsi aktivasi.

c. Dimensi Output

Dimensi *output* adalah ukuran data (atau tensor) yang dihasilkan setelah operasi jaringan saraf tertentu, seperti *convolution*, *pooling*, atau lapisan yang lengkap. Ini penting karena menunjukkan bagaimana ukuran data berubah saat melewati setiap lapisan jaringan neural.

$$O = \frac{I - K + 2P}{S} + 1 \quad (2.26)$$

Dimana :

- O adalah ukuran output (height/width).
- I adalah ukuran input.
- K adalah ukuran kernel.
- P adalah padding.
- S adalah stride.

Sebagai contoh :

Input 224 x 224, kernel 7 x 7, padding 3, stride 2

$$O = \frac{224 - 7 + 2(3)}{2} + 1 = 112 \quad (2.27)$$

Output : 112 x 112

d. Backpropagation

Selama *backpropagation*, menghentikan koneksi membantu menjaga aliran gradien. Gradien dihitung sebagai

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial H} \cdot \left(1 + \frac{\partial F(x)}{\partial x}\right) \quad (2.28)$$

- *Shortcut* Gradien (1-1): Memberikan jalur langsung untuk gradien tanpa melalui transformasi $F(x)$.
- Transformasi Gradien ($\frac{\partial F(x)}{\partial x}$) Gradien dari fungsi transformasi non-linear.

e. Parameter bottleneck

Parameter *bottleneck block* adalah jumlah bobot (weights) yang perlu dioptimalkan dalam sebuah *bottleneck block* dalam arsitektur ResNet50 tanpa mengurangi kemampuan jaringan untuk mengekstraksi fitur.

$$Total\ Parameter = (C_{in} \times C_{mid}) + (C_{mid} \times C_{mid} \times K^2) + (C_{mid} \times C_{out}) \quad (2.29)$$

Dimana :

- C_{in} adalah jumlah channel input.
- C_{mid} adalah jumlah channel intermediate.
- C_{out} adalah jumlah channel output.
- K^2 adalah tambahan untuk bias untuk setiap filter (misalnya 3 x 3).

Contoh : $C_{in} = 256, C_{mid} = 64, C_{out} = 256$:

$$\begin{aligned} Total\ Parameters &= (256 \times 64) + (64 \times 64 \times 9) + (64 \times 256) \\ &= 16,384 + 36,864 + 16,384 = 69,632 \end{aligned}$$

f. Fungsi Loss

ResNet50 dilatih dengan fungsi *loss categorical cross-entropy*. Berikut fungsinya :

$$L = - \sum_{i=1}^C y_i \log(\hat{y}_i) \quad (2.30)$$

Dimana :

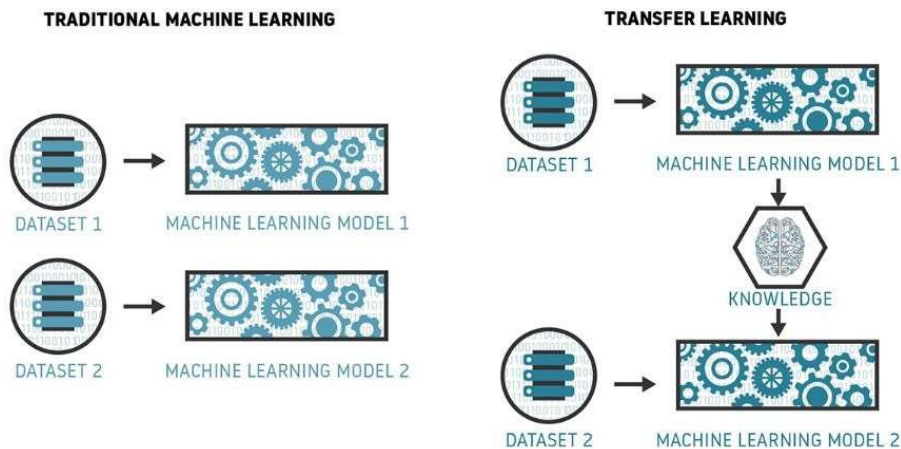
- C adalah jumlah kelas.
- y_i adalah label sebenarnya untuk kelas i (0 atau 1).
- $(\hat{y}_i)$ adalah probabilitas prediksi untuk kelas i .

2.5 Transfer Learning

Transfer learning adalah teknik *machine learning* yang sering digunakan dalam *deep learning*, khususnya untuk masalah dengan data terbatas. Metode *transfer learning* menggunakan model terlatih dari tugas-tugas terkait sebelumnya yang sebanding untuk melatih model saat ini (Asmaul, 2022). *Convolutional Neural Network* (CNN) yang telah dilatih sebelumnya yang digunakan dalam *transfer learning* dioptimalkan untuk satu tugas dan mentransfer pengetahuan ke mode yang berbeda (Nevavuori, 2019).

Transfer learning telah dibahas dengan beberapa nama dalam literatur sejak tahun 1990-an dimulai : *Learning to learn* (Caruana, 1995), *Life-long learning* (Silver, 2002), *Knowledge transfer* (Huang, 2013), *Inductive transfer* (Ruckert, 2008), *Multitask learning* (Caruana, 1997), *Knowledge consolidation* (Silver, 2013), *Context-sensitive learning* (Silver, 2007), *Knowledgebased inductive bias* (Caruana, 1993), *Meta learning* (Bengio, 1991), dan *Incremental or cumulative learning* (Thrun, 1991).

Model dalam *transfer learning* belajar dari satu tugas dan kemudian memecahkan masalah yang terkait dengan tugas lain. Misalnya, model yang sudah dilatih untuk mengidentifikasi objek dalam foto dapat digunakan untuk mengidentifikasi wajah atau emosi dalam foto.



Gambar 2.6 Transfer Learning

Sumber : <https://dqlab.id>

Transfer learning memiliki beberapa keuntungan yang sangat baik untuk *machine learning*. Pertama, *transfer learning* menghemat waktu dan sumber daya karena memungkinkan penggunaan *pre-trained* model yang sudah dilatih dengan *dataset* besar, yang mengurangi kebutuhan untuk komputasi dan waktu pelatihan. Kedua, karena *pre-trained* model telah mempelajari fitur umum yang relevan, teknik ini bekerja lebih baik dengan *dataset* kecil. Hal ini memungkinkan model memiliki akurasi yang tinggi dengan sedikit pelatihan ulang.

Ketiga, *transfer learning* mengurangi risiko *overfitting* karena model dapat memanfaatkan pengetahuan yang telah dipelajari sebelumnya, terutama dalam kasus *dataset* terbatas. Ini juga membantu model membuat generalisasi lebih mudah, sehingga mereka dapat menemukan pola umum yang dapat diterapkan pada berbagai tugas baru. Keuntungan tambahan adalah fleksibilitas *transfer learning*, yang dapat diterapkan pada berbagai bidang, seperti *natural language processing* dan *computer vision*.

2.6 Perbandingan CNN, VGG16, dan ResNet50

CNN dasar mudah dilatih dan hemat sumber daya komputasi, memiliki arsitektur sederhana dengan kurang dari sepuluh lapisan, dan tidak dapat menangkap fitur kompleks. Dengan filter konvolusi 3×3 yang konsisten (Mannepalli et al., 2024), struktur yang homogen dan mudah dipahami, dan 138 juta parameter, VGG16 meningkatkan kedalaman menjadi 16 lapisan. Dengan 50 lapisan, ResNet50 mengatasi vanishing gradient dengan skip connections (Murimi, 2023.), tetapi hanya dengan 25 juta parameter karena desain blok bottleneck yang efisien.

CNN dasar cepat dilatih dalam proses pelatihan tetapi cepat mencapai titik jenuh pada dataset kompleks. VGG16 membutuhkan waktu yang lebih lama untuk dilatih, rentan terhadap overfitting, dan konvergensi yang lebih lambat. Training yang lebih efektif, masalah vanishing gradient yang lebih sedikit, dan ketahanan terhadap overfitting diperoleh dengan ResNet50 dengan koneksi skip. Meskipun CNN dasar cukup baik untuk tugas sederhana dalam klasifikasi citra, ia hanya dapat digunakan pada dataset kompleks. ResNet50 biasanya mengungguli VGG16 dalam menangkap fitur detail, terutama tekstur dan pola lokal, tetapi VGG16 juga dapat mempelajari representasi hierarkis yang lebih dalam.

Untuk klasifikasi penyakit tanaman padi, CNN dasar sulit mengidentifikasi perbedaan kecil antara penyakit tetapi efektif dengan perangkat terbatas. Meskipun ukurannya besar, VGG16 berpotensi lebih baik dalam mendeteksi gejala penyakit tertentu seperti bercak atau perubahan warna. ResNet50 memungkinkan pengenalan pola kompleks dengan efisiensi parameter yang lebih baik, membuatnya lebih sesuai untuk penggunaan praktis.

Tabel 2.2 Perbandingan CNN, VGG16 dan ResNet50

Aspek	CNN	VGG16	ResNet50
Jumlah Lapisan	Biasanya <10	16	50
Jumlah Parameter	Relatif Sedikit <5 Juta	~138 Juta	~25 Juta
Arsitektur Utama	Konvolusi Sederhana + Pooling	Blok Konvolusi 3x3	Block Bottleneck dengan Skip connection
Keunggulan	Sederhana dan Mudah dilatih	Menangkap detail tekstur dengan baik	Menangkap representasi abstrak kompleks
Tantangan	Terbatas untuk fitur kompleks	Rentan overfitting, komputasi berat	Arsitektur yang lebih rumit
Kecepatan Inferensi	Cepat	Lambat	Moderat
Kebutuhan Memori	Rendah	Tinggi	Moderat
Top-5 Akurasi ImageNet	~80 – 85%	~90%	~92.9%
Aplikasi Optimal	Tugas klasifikasi sederhana	Tugas yang memerlukan detail visual	Klasifikasi objek kompleks

2.7 Python

Python adalah bahasa pemrograman tingkat tinggi yang dibuat untuk membuat penulisan dan membaca kode mudah. *Python* awalnya dibuat oleh *Guido van Rossum* pada akhir tahun 1980-an dan dirilis pertama kali pada tahun 1991. Itu dimaksudkan untuk menjadi penerus bahasa ABC yang memiliki kemampuan yang

lebih baik untuk menangani kesalahan dan mendukung pengembangan skala besar. Terinspirasi dari grup komedi Inggris "Monty Python's Flying Circus", nama *Python* sendiri mencerminkan etika bahasa yang menyenangkan dan santai.

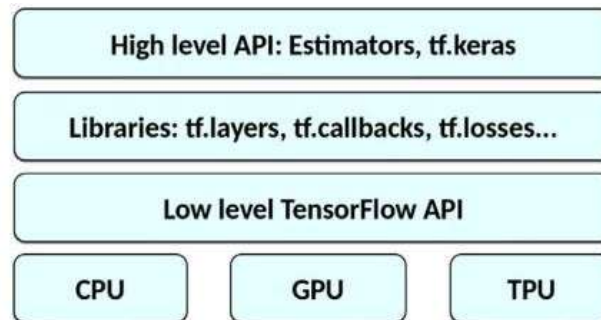
Python adalah bahasa pemrograman yang telah mendapatkan popularitas karena kesederhanaan, keterbacaan, dan fleksibilitasnya. Karena sintaksisnya yang sederhana, yang mirip dengan bahasa Inggris sehari-hari, *Python* menawarkan titik masuk pemrograman yang mudah diakses (Kovac, 2024). Selain itu, pustaka *Python* yang luas, termasuk *panda*, *NumPy*, dan *SciPy*, menawarkan sumber daya yang kuat untuk *machine learning*, visualisasi, dan analisis data, menjadikannya alat yang sangat baik untuk melakukan penelitian.

2.8 Tensorflow

TensorFlow adalah kerangka kerja *open source* yang digunakan untuk membangun dan melatih model pembelajaran mesin, terutama pembelajaran mendalam. *TensorFlow*, yang pertama kali dirilis pada tahun 2015, telah menjadi salah satu alat paling populer untuk pengembang kecerdasan buatan (AI). *TensorFlow* adalah platform *machine learning* yang kuat dengan *tools* yang mudah digunakan untuk mengembangkan dan menguji arsitektur *deep learning* (Akgun, 2022).

TensorFlow, sebuah pustaka perangkat lunak atau kerangka kerja yang dikembangkan oleh Google, memungkinkan untuk dengan cepat menerapkan teknik *machine learning* dan *deep learning* (Monica, 2022). *TensorFlow* memiliki banyak API yang dapat disesuaikan untuk memenuhi kebutuhan pengembangan untuk berbagai tingkat keahlian, dari pemula hingga tingkat lanjut. *TensorFlow*

memiliki banyak lapisan, dan API tingkat rendah berinteraksi dengan CPU, GPU, dan TPU seperti yang ditunjukkan pada gambar berikut :



Gambar 2. 7 Tensorflow API

Source : Akgun, 2022

TensorFlow memiliki API yang dibagi menjadi tiga bagian utama. API tingkat tinggi, seperti *tf.keras*, memberikan kemudahan penggunaan, *Libraries*, seperti *tf.layers*, dan API tingkat rendah memberikan kontrol penuh atas operasi komputasi. *TensorFlow* juga mendukung eksekusi pada CPU, GPU, dan TPU untuk meningkatkan kinerja pelatihan model.

2.9 Keras

Keras adalah *library open-source* yang dibuat dengan *Python* yang digunakan untuk membangun dan melatih model *deep learning*. Dengan antarmuka yang sederhana, *library* ini memungkinkan pengguna membuat model *neural network* dengan cepat tanpa mengetahui detail teknis yang rumit. Selain mendukung berbagai arsitektur *neural network*, seperti *Convolutional Neural Networks* (CNN) dan *Recurrent Neural Networks* (RNN), Keras memungkinkan pengguna untuk

membangun model secara fleksibel menggunakan komponen seperti lapisan (*layers*), *optimizers*, dan *loss functions*. Ini memungkinkan Keras untuk digunakan dalam berbagai aplikasi *machine learning*.

2.10 Anaconda Python

Anaconda Python adalah platform sumber terbuka gratis yang memungkinkan untuk menulis dan mengeksekusi kode dalam bahasa pemrograman *Python*. Platform ini dibuat oleh continuum.io, merupakan sebuah perusahaan yang mengkhususkan dalam pengembangan *Python*. *Anaconda Navigator* adalah antarmuka grafis pengguna (GUI) yang digunakan untuk mengelola lingkungan *Python* dan R, serta paket-paket terkait *data science*, *machine learning*, dan analisis data. Navigator memiliki berbagai fitur utama, seperti manajemen paket dan lingkungan yang memungkinkan pengguna untuk menginstal, memperbarui, dan menghapus paket-paket *Python* atau R dengan antarmuka yang mudah digunakan.

Navigator mendukung pembuatan lingkungan virtual untuk mengisolasi proyek, sehingga konflik antar-paket dapat dihindari. Selain itu pengguna juga dapat mengakses alat-alat populer seperti *Jupyter Notebook*, *Spyder*, *RStudio*, dan VS Code dengan cepat.

2.11 Penelitian terdahulu

Penelitian terdahulu yang dilakukan oleh (Mannepalli et al., 2024) membahas tentang Diagnosis penyakit tanaman padi menggunakan VGG16. Tujuan utama dari penelitian ini yaitu mengembangkan metode baru untuk diagnosis penyakit hawar daun bakteri, hawar daun, dan bercak coklat pada daun padi dengan menggunakan jaringan syaraf tiruan (CNN) VGG16. Hasil yang diperoleh dari penelitian ini

menunjukkan bahwa setelah dilakukan pengujian, penerapan model VGG16 mendapatkan hasil akurasi yang sangat baik yaitu sebesar 97.77%.

Pada penelitian yang dilakukan oleh (Handayani et al., 2024) yang membahas tentang penggunaan algoritma VGG16 dan VGG19 untuk klasifikasi jenis kelamin berdasarkan radiografi *sefalometrik* pada orang dewasa di Indonesia. Tujuan dari penelitian ini adalah melakukan penyelidikan efektivitas model *deep learning* dalam memprediksi jenis kelamin secara akurat. Tantangan yang ada dalam penelitian ini adalah menyoroti Kumpulan data yang tidak seimbang dalam mengklasifikasikan ciri – ciri laki – laki secara akurat. Hasil yang diperoleh dari penelitian yang telah dilakukan menunjukkan model VGG16 memiliki akurasi identifikasi jenis kelamin sebesar 93% untuk Perempuan dan 73% untuk laki – laki yang menghasilkan akurasi rata – rata sebesar 89% untuk kedua jenis kelamin. Sedangkan pada model VGG19 mencapai akurasi sebesar 100% dalam mengidentifikasi ciri – ciri *sefalometrik* pada Perempuan secara akurat dalam 21 sampel contoh. Model VGG19 menunjukkan akurasi keseluruhan sebesar 93% dalam menganalisis kedua jenis kelamin. Untuk jenis kelamin Perempuan, akurasinya mencapai 95%, sedangkan untuk laki – laki, akurasinya mencapai 80%.

Penelitian yang dilakukan oleh (Swain & Tripathy, 2024) membahas tentang mendeteksi lubang dan penilaian kerusakan jalan menggunakan *machine learning* dan teknik *image processing* khususnya menggunakan metode *Convolutional neural network* (CNN) dan *transfer learning*, secara signifikan meningkatkan kecepatan dan akurasi . Tujuan utama dari penelitian ini adalah mengotomatisasi deteksi lubang dengan meningkatkan kualitas gambar dan pada akhirnya data

meningkatkan keselamatan di jalan raya. Hasil yang diperoleh dari penelitian ini mendapatkan akurasi sebesar 99.23% dengan *transfer learning* dan 98.46% tanpa menggunakan *transfer learning*. Penelitian yang dilakukan oleh (Xu et al., 2024) membahas tentang pengembangan jaringan saraf tiruan (CNN) berbasis VGG16 untuk *screening* kanker paru – paru yang tepat menggunakan CT Scan, Tujuan dari penelitian ini yaitu berfokus pada efektifitas algoritma *deep learning* dalam meningkatkan deteksi dan diagnosis kanker paru – paru dini dengan membandingkan kinerja model penelitian sebelumnya. Hasil yang didapatkan dari penelitian ini yaitu AUC (*Area Under the Curve*) pada model menunjukkan kemampuan diagnostik yang sangat baik dengan AUC sebesar 0.963 ± 0.004 . dengan akurasi klasifikasi mencapai nilai rata – rata sebesar $91.7\% \pm 0.004$, *positive predictive value* (PPV) sebesar $86.8\% \pm 0.015$, *negative predictive value* (NPV) sebesar $93.1\% \pm 0.003$, sensitivitas model sebesar $77.6\% \pm 0.01$, spesifisitas mencapai $96.2\% \pm 0.005$ dan *F1-Score* sebesar $81.9\% \pm 0.008$.

Pada penelitian yang dilakukan oleh (Basha et al., 2024) yang membahas tentang deteksi dan segmentasi pada tumor otak menggunakan Teknik *deep learning* dan *machine learning*, terutama berfokus pada gambar MRI. Tujuan dari penelitian ini untuk membantu para *professional* di bidang Kesehatan dalam membuat diagnosis yang tepat dan mengurangi Tingkat kematian yang terkait dengan tumor otak. Dari penelitian yang telah dilakukan menunjukkan bahwa model VGG16 mencapai akurasi yang tinggi yaitu 99.6% dan sensitivitas 99.61% dengan tingkat kesalahan yang rendah yaitu 3.2% dibandingkan dengan model lain seperti InceptionV3 dan ResNet50 yang memiliki tingkat kesalahan yang lebih tinggi yaitu sekitar 38% dan 22%.

Studi yang dilakukan oleh (Bella Dwi Mardiana et al., 2023) membahas klasifikasi tanaman herbal yang secara khusus berfokus pada kesalahan identifikasi karena kurangnya pengetahuan dikalangan Masyarakat umum. Tujuan dari penelitian yang dilakukan untuk mengklasifikasikan 10 (sepuluh) jenis tanaman herbal menggunakan model *deep learning* menggunakan arsitektur VGG16 dengan *Transfer Learning*. Hasil dari penelitian ini menghasilkan akurasi sebesar 96.73% dan validasi sebesar 95.63% dengan memanfaatkan *dataset* gambar sebanyak 3500. Ketika menggunakan Teknik *augmentasi* data, akurasi pengujian yang didapatkan sebesar 97%.

Setelah membahas beberapa penelitian terdahulu mengenai VGG16 , maka selanjutnya beralih pada bahasan penelitian terdahulu mengenai ResNet50 yang merupakan salah satu model yang populer dalam pengenalan gambar dan klasifikasi objek. ResNet50 dikenal sebagai arsitektur pembelajaran mendalam yang lebih baik karena relatif mudah dioptimalkan dan dapat mencapai akurasi yang lebih tinggi (Elpeltagy & Sallam, 2021). Penelitian terdahulu yang dilakukan oleh (Cambay et al., 2024) membahas pengembangan model *deep learning* yaitu ResNet50. Tujuan dari penelitian ini adalah mendeteksi otomatis dan klasifikasi penyakit pencernaan menggunakan gambar endoskopi yang berfokus pada peningkatan kinerja dan akurasi diagnostik pada kondisi seperti maag dan polip. Hasil dari penelitian ini yaitu model ResNet50 menunjukkan akurasi klasifikasi sebesar 92% untuk ketiga dataset, diantaranya *Kvasir dataset*, *Wireless capsule endoscopy* (WCE) dataset dan *Kvasir V2 dataset*. Sementara itu untuk dataset *Wireless capsule endoscopy* (WCE) mendapatkan akurasi sebesar 99,13%.

Pada penelitian yang dilakukan oleh (Wan et al., 2024) membahas tentang deteksi pertumbuhan pada ganggang biru dan hijau di badan air dengan meningkatkan model arsitektur ResNet50. Tujuan dari penelitian ini adalah untuk meningkatkan *feature extraction* atau ekstraksi fitur dan kinerja klasifikasi dalam mengatasi tantangan yang berkaitan dengan ekstraksi fitur yang tidak efektif dan kesalahan pada klasifikasi. Hasil dari peningkatan model arsitektur ResNet50 memperoleh *accuracy* 98.26%, *Precision* 95.81%, *F1-Score* 95.70% dan *Recall* 95.59%. jika dibandingkan dengan model lain seperti *MobileNetV2* mendapatkan *accuracy* sebesar 91.37% dan *Recall* 87.38%. model *ShuffleNetV2* mendapatkan *accuracy* sebesar 96.10% dan *Recall* 93.12%. model *DenseNet121* mendapatkan *accuracy* sebesar 96.41% dan *Recall* 93.37%. model ResNet50 yang di tingkatkan lebih unggul dibandingkan dengan model lainnya.

Penelitian yang dilakukan oleh (Li et al., 2024) yang membahas tentang klasifikasi dan deteksi cacat las menggunakan berbagai model *deep learning* seperti ResNet50. Tujuan dari penelitian ini antara lain mendeteksi dan klasifikasi cacat las alami menggunakan teknik pencitraan *Magneto-Optical* (MO) dengan fokus pada peningkatan akurasi dan kinerja klasifikasi melalui metode *Principal Component Analysis* (PCA) dan arsitektur *deep learning* yaitu ResNet50. Hasil yang diperoleh dari penelitian ini yaitu model ResNet50 mendapatkan akurasi keseluruhan yang tinggi sebesar 99% untuk *natural weld defects* dengan akurasi yang spesifik sebesar 99% untuk *cracks* dan 98% untuk *fores*. Akurasi dari keseluruhan model CNN sebesar 97%. Hasil ini menunjukkan bahwa model ResNet50 unggul dalam mengklasifikasikan cacat las.

Penelitian yang dilakukan oleh (Chen et al., 2024) menggunakan teknik pembelajaran mendalam, khususnya melalui analisis data MRI, untuk mengklasifikasikan penyakit Alzheimer (AD). Tujuan dari penelitian ini adalah untuk membedakan penyakit Alzheimer (AD) dari individu kognitif normal (CN), serta untuk meningkatkan akurasi dan keandalan deteksi AD dalam pencitraan medis. Hasil dari penelitian ini menunjukkan bahwa model ResNet50 memiliki keunggulan dalam klasifikasi penyakit Alzheimer karena mempertahankan akurasi tinggi dalam membedakan antara kelompok AD dan kelompok Mild Cognitive Impairment (MCI) dan MCI dan CN, masing-masing. Hasil ini menunjukkan bahwa model ResNet50 memiliki akurasi 98,91% dalam klasifikasi antara AD dan CN.

Studi yang dilakukan oleh (Pamungkas et al., 2023) menggunakan Jaringan Syaraf Tiruan (CNN) untuk mengklasifikasikan penyakit jagung berdasarkan gambar daun. Penelitian ini berfokus pada perbandingan kinerja dua model, EfficientNet-B0 dan ResNet-50, dalam menentukan empat klasifikasi penyakit jagung: *Blight*, *Common Rust*, *Grey Leaf Spot* dan *Healthy*. Tujuan dari penelitian ini adalah untuk meningkatkan akurasi identifikasi penyakit jagung untuk membantu petani menghindari kerugian panen. Hasil penelitian menunjukkan bahwa model ResNet50 mencapai *accuracy* sebesar 93%, *precision* 93%, *recall* 92% dan *F1-score* 92%. Sementara pada model EfficientNet-B0 mendapatkan *accuracy* sebesar 94%, *precision* 93%, *recall* 91% dan *F1-score* 92%. Model EfficientNet-B0 sedikit lebih unggul dibandingkan dengan model ResNet50 dalam mengklasifikasikan penyakit jagung.

Penelitian yang dilakukan oleh (Deng et al., 2022) menyelidiki efisiensi diagnostik model pembelajaran mesin yang berbeda, seperti ResNet50, forest random (RF), dan Deep Learning Ensemble (DS), dalam membedakan karsinoma tiroid papiler (PTC) dari nodul tiroid lainnya. Fokus penelitian adalah menilai potensi penggunaan data teks dan pencitraan untuk meningkatkan akurasi diagnosis tiroid karena kemiripan nodul tiroid menyebabkan kesulitan dalam diagnosis diferensial. Tujuan utama penelitian ini adalah untuk menilai kemampuan model diagnostik untuk menemukan PTC. Tujuannya adalah untuk meningkatkan akurasi diagnosa. Hasil menunjukkan bahwa model *deep learning* Ensemble (DS) berhasil mengobati 572 pasien dengan 38 kesalahan diagnosa. Ini termasuk dua belas kasus PTC yang salah dianggap sebagai jenis nodul lainnya, tiga belas kasus FTC, empat kasus MTC, dan sembilan kasus adenoma yang salah dianggap sebagai PTC dengan sensitivitas 85,87%, spesifisitas 97,18%, akurasi 93,77%, dan area di bawah kurva karakteristik operasi penerima (AUC) sebesar 0,982. Model random forest (RF) berhasil mendiagnosis 529 kasus karsinoma tiroid papiler (PTC) dengan 81 kesalahan diagnosa. Kesalahan tersebut mencakup 16 kasus PTC yang salah diidentifikasi sebagai nodul lainnya, 18 sebagai karsinoma tiroid folikuler (FTC), 26 sebagai karsinoma tiroid meduler (MTC), satu kasus gondok nodular, dan 26 kasus adenoma yang salah didiagnosis sebagai PTC. Model RF memiliki akurasi 86,72%, sensitivitas 64,67%, tingkat kesalahan diagnosis 13,28%, dan spesifisitas 96,24%. Sementara itu, model ResNet50 mendiagnosis 546 gambar dengan 64 kesalahan diagnosa, termasuk 35 kasus PTC yang salah dianggap sebagai jenis nodul lainnya, 16 kasus FTC, satu kasus MTC, satu kasus gondok nodular, dan 11 kasus adenoma yang salah dianggap sebagai PTC. dengan sensitivitas 84,24%,

spesifisitas 91,78%, akurasi 89,57%, dan AUC 0,955, tetapi mengalami 64 kesalahan diagnosa. Temuan ini menunjukkan keunggulan model ensemble DS dalam klasifikasi karsinoma tiroid papiler (PTC). Berikut adalah tabel penelitian terdahulu yang mendukung kerangka teoritis pada penelitian ini :

Tabel 2.3 Penelitian Terdahulu

No	Judul	Metode	Hasil
1.	Diagnosis of bacterial leaf blight, leaf smut, and brown spot in rice leafs using VGG16. (Praveen Kumar Mannepallia, Ayonija Pathreb, Gunjan Chhabrac, Priyanka Anup Ujjainkard, Shrutika Wanjarie, 2024).	Convolutional Neural Network ((CNN) , VGG16.	Hasil yang diperoleh yaitu penerapan model VGG16 mendapatkan hasil akurasi yang sangat baik yaitu sebesar 97.77%.
2.	Classification of Indonesian adult forensic gender using cephalometric radiography with VGG16 and VGG19: a Preliminary research. (Vitria Wuri Handayania,b , Ahmad	VGG16 dan VGG19	Hasil yang diperoleh dari penelitian yang telah dilakukan menunjukkan model VGG16 memiliki akurasi identifikasi jenis kelamin sebesar 93% untuk Perempuan dan 73% untuk laki – laki yang menghasilkan akurasi

	Yudiantoc,d , Mieke Sylvia M.A.Rd,e , Riries Rulaningtyasf,g and Muhammad Rasyad Caesarardhi, 2024).		rata – rata sebesar 89% untuk kedua jenis kelamin. Sedangkan pada model VGG19 mencapai akurasi sebesar 100% dalam mengidentifikasi ciri – ciri <i>sefalometrik</i> pada Perempuan secara akurat dalam 21 sampel contoh. Model VGG19 menunjukkan akurasi keseluruhan sebesar 93% dalam menganalisis kedua jenis kelamin. Untuk jenis kelamin Perempuan, akurasi mencapai 95%, sedangkan untuk laki – laki, akurasi mencapai 80%.
3.	Automatic detection of potholes using VGG-16 pre-trained network and Convolutional Neural Network. (Satyabrata Swain,	VGG16 Pre – Trained network dan Convolutional Neural Network (CNN).	Hasil yang diperoleh dari penelitian ini mendapatkan akurasi sebesar 99.23% dengan <i>transfer learning</i> dan 98.46% tanpa menggunakan <i>transfer learning</i>.

	Asis Kumar Tripathy, 2024).		
4.	Precision lung cancer screening from CT scans using a VGG16-based convolutional neural network. Hua Xu , Yuanyuan Yu , Jie Chang , Xifeng Hu , Zitong Tian and Ouwen Li, 2024).	VGG16 – Based Convolutional Neural Network (CNN).	Model VGG16 mencapai akurasi yang tinggi yaitu 99.6% dan sensitivitas 99.61% dengan tingkat kesalahan yang rendah yaitu 3.2% dibandingkan dengan model lain seperti InceptionV3 dan ResNet50 yang memiliki tingkat kesalahan yang lebih tinggi yaitu sekitar 38% dan 22%.
5.	Mask region-based convolutional neural network and VGG-16 inspired brain tumor segmentation. Niha Kamal Basha, Christo Ananth, K. Muthukumaran, Gadug Sudhamsu, Vikas Mittal ⁵ & Fikreselam Gared, 2024).	Convolutional Neural Network (CNN), VGG16 dan Segmentasi.	Model VGG16 mencapai akurasi yang tinggi yaitu 99.6% dan sensitivitas 99.61% dengan tingkat kesalahan yang rendah yaitu 3.2% dibandingkan dengan model lain seperti InceptionV3 dan ResNet50 yang memiliki tingkat kesalahan yang lebih

			tinggi yaitu sekitar 38% dan 22%.
6.	Herbal Leaves Classification Based on Leaf Image Using CNN Architecture Model VGG16. (Bella Dwi Mardiana, Wahyu Budi Utomo, Ulfah Nur Oktaviana, Galih Wasis Wicaksono, Agus Eko Minarno, 2023).	Convolutional Neural Network (CNN), Pre – Trained VGG16.	Hasil dari penelitian ini menghasilkan akurasi sebesar 96.73% dan validasi sebesar 95.63% dengan memanfaatkan <i>dataset</i> gambar sebanyak 3500. Ketika menggunakan Teknik <i>augmentasi</i> data, akurasi pengujian yang didapatkan sebesar 97%.
7.	Automated Detection of Gastrointestinal Diseases Using Resnet50*-Based Explainable Deep Feature Engineering Model with Endoscopy Images. (Veysel Yusuf Cambay, Prabal Datta Barua, Abdul Hafeez Baig, Sengul Dogan, Mehmet	ResNet50 Based Explainable Deep Feature Engineering Model.	Hasil dari penelitian ini yaitu model ResNet50 menunjukkan akurasi klasifikasi sebesar 92% untuk ketiga <i>dataset</i> , diantaranya <i>Kvasir dataset</i> , <i>Wireless capsule endoscopy</i> (WCE) <i>dataset</i> dan <i>Kvasir V2 dataset</i> . Sementara itu untuk <i>dataset Wireless capsule endoscopy</i>

	Baygin, Turker Tuncer and U. R. Acharya, 2024)		(WCE) mendapatkan akurasi sebesar 99,13%.
8.	An Improved ResNet50 for Environment Image Classification. (Jiacheng Wana, Bingchan Lib, Kun Wanga, Xixi Tenga, Tao Wanga , Bo Maoa, 2024).	ResNet50	Hasil dari peningkatan model arsitektur ResNet50 memperoleh <i>accuracy</i> <i>98.26%</i> , <i>Precision</i> <i>95.81%</i> , <i>F1-Score</i> <i>95.70%</i> dan <i>Recall</i> <i>95.59%</i> . jika dibandingkan dengan model lain seperti <i>MobileNetV2</i> mendapatkan <i>accuracy</i> sebesar <i>91.37%</i> dan <i>Recall</i> <i>87.38%</i> . model <i>ShuffleNetV2</i> mendapatkan <i>accuracy</i> sebesar <i>96.10%</i> dan <i>Recall</i> <i>93.12%</i> . model <i>DenseNet121</i> mendapatkan <i>accuracy</i> sebesar <i>96.41%</i> dan <i>Recall</i> <i>93.37%</i> . model ResNet50 yang di tingkatkan lebih

			unggul dibandingkan dengan model lainnya.
9.	Automatic Detection and Classification of Natural Weld Defects Using Alternating MagnetoOptical Imaging and ResNet50. (Yanfeng Li, Pengyu Gao, Yongbiao Luo, Xianghan Luo, Chunmei Xu, Jiecheng Chen, Yanxi Zhang, Genxiang Lin and Wei Xu, 2024).	Magneto – Optical Imaging dan ResNet50.	Hasil yang diperoleh dari penelitian ini yaitu model ResNet50 mendapatkan akurasi keseluruhan yang tinggi sebesar 99% untuk <i>natural weld defects</i> dengan akurasi yang spesifik sebesar 99% untuk <i>cracks</i> dan 98% untuk <i>fores</i> . Akurasi dari keseluruhan model CNN sebesar 97%. Hasil ini menunjukkan bahwa model ResNet50 unggul dalam mengklasifikasikan cacat las.
10.	Automated Alzheimer’s disease classification using deep learning models with Soft-NMS and improved ResNet50 integration. (Yusi Chen, Lizhen Wang	Soft-NMS dan Imporved ResNet50 integration.	Hasil ini menunjukkan bahwa model ResNet50 memiliki akurasi 98,91% dalam klasifikasi antara AD dan CN.

	f, Bijiao Ding e, Jianshe Shi e, Tingxi Wen d, Jianlong Huang, Yuguang Ya, 2024).		
11.	Leaf Image Identification: CNN with EfficientNet-B0 and ResNet-50 Used to Classified Corn Disease. (Wisnu Gilang Pamungkas, Muchammad Iqbal Putra Wardhana, Zamah Sari, Yufis Azhar, 2023).	Convolutional Neural Network (CNN), EfficientNet-B0 dan ResNet50.	Hasil penelitian menunjukkan bahwa model ResNet50 mencapai <i>accuracy</i> sebesar 93%, <i>precision</i> 93%, <i>recall</i> 92% dan <i>F1-score</i> 92%. Sementara pada model EfficientNet-B0 mendapatkan <i>accuracy</i> sebesar 94%, <i>precision</i> 93%, <i>recall</i> 91% dan <i>F1-score</i> 92%. Model EfficientNet-B0 sedikit lebih unggul dibandingkan dengan model ResNet50 dalam mengklasifikasikan penyakit jagung.
12.	Differential diagnostic value of the ResNet50, random forest, and DS ensemble models	ResNet50, Random Forest dan DS Ensemble models.	Hasil menunjukkan bahwa model <i>deep learning</i> Ensemble (DS) berhasil mengobati 572 pasien

	for papillary thyroid carcinoma and other thyroid nodules. (Chengwen Deng, Dongyan Han, Ming Feng, Zhongwei Lv and Dan Li. 2022).		dengan 38 kesalahan diagnosa. Ini termasuk dua belas kasus PTC yang salah dianggap sebagai jenis nodul lainnya, tiga belas kasus FTC, empat kasus MTC, dan sembilan kasus adenoma yang salah dianggap sebagai PTC dengan sensitivitas 85,87%, spesifisitas 97,18%, akurasi 93,77%, dan area di bawah kurva karakteristik operasi penerima (AUC) sebesar 0,982. Model random forest (RF) berhasil mendiagnosis 529 kasus karsinoma tiroid papiler (PTC) dengan 81 kesalahan diagnosa. Kesalahan tersebut mencakup 16 kasus PTC yang salah diidentifikasi sebagai nodul lainnya, 18 sebagai karsinoma tiroid folikuler (FTC), 26 sebagai karsinoma
--	---	--	---

			tiroid meduler (MTC), satu kasus gondok nodular, dan 26 kasus adenoma yang salah didiagnosis sebagai PTC. Model RF memiliki akurasi 86,72%, sensitivitas 64,67%, tingkat kesalahan diagnosis 13,28%, dan spesifisitas 96,24%. Sementara itu, model ResNet50 mendiagnosis 546 gambar dengan 64 kesalahan diagnosa, termasuk 35 kasus PTC yang salah dianggap sebagai jenis nodul lainnya, 16 kasus FTC, satu kasus MTC, satu kasus gondok nodular, dan 11 kasus adenoma yang salah dianggap sebagai PTC.
--	--	--	--

BAB III

METODOLOGI PENELITIAN

3.1 Pendekatan Penelitian

Penelitian ini menggunakan metode Convolutional Neural Network (CNN) untuk mengklasifikasikan berbagai kelas penyakit pada tanaman padi dengan melakukan analisis komparatif antara dua arsitektur jaringan syaraf tiruan, yaitu VGG16 dan ResNet50. Convolutional Neural Network (CNN) dikenal memiliki kemampuan unggul dalam memproses gambar dan mengenali pola. CNN sangat baik untuk tugas-tugas pengenalan citra karena dapat belajar dan mengekstraksi fitur dari data gambar secara otomatis, mengurangi ketergantungan pada ekstraksi fitur secara manual, yang sering kali memakan waktu dan tidak efisien.

3.2 Populasi dan sampel penelitian

Sampel penelitian ini diambil dari gambar-gambar yang menunjukkan penyakit pada tanaman padi yang dapat diakses melalui platform Kaggle. Dalam total, terdapat 2095 gambar daun padi yang dipilih, yang mencakup beberapa jenis penyakit, yaitu 300 Sampel Bercak coklat (*brown spot disease*), 634 Sampel Bercak daun (*blast disease*), 149 sampel Gosong palsu (*false smut disease*), 349 sampel Hawar daun bakteri (*bacterial blast disease*) dan juga 663 Sampel Daun yang sehat (*healthy*). Relevansi foto dengan tujuan penelitian, ketersediaan data yang memadai, dan kualitas foto yang memenuhi standar yang telah ditentukan adalah beberapa kriteria yang menentukan pemilihan sampel. Dengan cara ini, diharapkan bahwa penelitian dapat memberikan hasil yang lebih akurat dan dapat diandalkan tentang mengidentifikasi penyakit tanaman padi.

3.3 Metode Pengumpulan Data

3.2.1 Data Sekunder

Data sekunder berupa gambar yang diperoleh melalui platform *Kaggle* yang memiliki rentang waktu 11-12-2023 sampai dengan 07-07-2024 dan data gambar yang diperoleh sebanyak 2095. Data yang diambil dalam rentang waktu 7 bulan memiliki tujuan agar memberikan ruang untuk mengumpulkan data yang lebih beragam yang nantinya dapat meningkatkan akurasi dan hasil yang diharapkan untuk membandingkan kedua model yang digunakan.

3.2.2 Studi Pustaka

Studi Pustaka pada penelitian ini dilakukan untuk mengumpulkan, menganalisis, dan merangkum berbagai penelitian dan teori yang telah dilakukan sebelumnya. Penelitian ini memiliki beberapa tujuan penting untuk studi pustaka. Studi pustaka juga membantu menjelaskan metodologi yang digunakan dalam penelitian dengan mencari referensi seperti dari penelitian sejenis, artikel jurnal, skripsi maupun thesis baik berbentuk fisik maupun elektronik dalam upaya pemecahan masalah.

3.4 Prosedur penelitian

Prosedur penelitian adalah serangkaian langkah dan tahapan yang digunakan sebagai kerangka pemikiran untuk menyelesaikan masalah penelitian. Dalam prosedur ini, terdapat alat-alat yang digunakan untuk pengumpulan dan pengolahan

data, serta tahapan yang membantu dalam penyelesaian masalah dan penulisan laporan penelitian. Prosedur penelitian dapat digambarkan sebagai berikut :



Gambar 3.4 Prosedur Penelitian

Berikut adalah penjelasan rinci mengenai setiap langkah dalam prosedur penelitian:

1. Identifikasi masalah

Tahap ini dimulai dengan menemukan masalah pertanian yang signifikan, serangan penyakit pada tanaman padi. Masalah ini menjadi latar belakang penting yang menunjukkan betapa pentingnya mendeteksi dan mengklasifikasikan penyakit dengan cepat untuk meningkatkan produksi pertanian dan ketahanan pangan.

2. Studi literatur

Setelah mengidentifikasi masalah, pencarian referensi / literatur untuk mendapatkan informasi. Tujuan dari tinjauan ini adalah untuk membuat

dasar teori yang kuat, memahami konsep terkait seperti Deep Learning dan desain CNN, dan menemukan celah dalam penelitian yang ada.

3. Penentuan populasi dan sampel

Selanjutnya, membuat populasi yang terdiri dari semua gambar penyakit tanaman padi yang diambil dari platform Kaggle. Populasi tersebut terdiri dari 2042 gambar daun yang menampilkan berbagai jenis penyakit, seperti bercak coklat (*brown spot disease*), bercak daun (*blast disease*), gosong palsu (*false smut disease*), hawar daun bakteri (*bacterial blast disease*) dan juga daun yang sehat (*healthy*). Untuk memastikan bahwa data memenuhi standar kualitas yang diperlukan, pemilihan sampel dilakukan dengan teliti.

4. Implementasi dan pelatihan model

Setelah data siap, proses implementasi model VGG16 dan ResNet50 dilakukan dengan menggunakan *framework* TensorFlow dan Keras. Parameter-parameter model disesuaikan sesuai dengan kebutuhan penelitian, untuk memastikan performa maksimal selama proses pelatihan. Model yang telah diimplementasikan kemudian dilatih menggunakan dataset yang telah diproses. Selama proses pelatihan, dilakukan pemantauan terhadap metrik evaluasi, seperti akurasi, kehilangan (loss), presisi, recall, dan F1-score, untuk menilai efektivitas model.

5. Evaluasi model dan analisis hasil

Setelah pelatihan, model diuji dengan *dataset* yang telah disiapkan sebelumnya. Hasil evaluasi ini mencakup akurasi, presisi, recall, skor F1, dan waktu pemrosesan. Evaluasi ini sangat penting untuk menentukan model mana yang lebih efisien dan efektif dalam mengklasifikasikan

penyakit tanaman padi. Hasil dari evaluasi kedua model dianalisis untuk memahami perbedaan performa antara VGG16 dan ResNet50. Analisis ini memberikan wawasan mengenai kelebihan dan kekurangan masing-masing arsitektur dalam konteks aplikasi di lapangan.

6. Penulisan laporan

seluruh hasil penelitian disusun dalam sebuah laporan. Laporan ini mencakup deskripsi metodologi, analisis data, kesimpulan, serta rekomendasi untuk penelitian lanjutan. Dengan demikian, laporan ini tidak hanya menyajikan hasil penelitian, tetapi juga menawarkan panduan yang berguna bagi peneliti lain dan praktisi di bidang pertanian.

3.5 Perangkat penelitian

Perangkat penelitian merupakan komponen penting dalam pelaksanaan suatu penelitian, karena berfungsi sebagai alat utama untuk mengumpulkan, menganalisis, dan menginterpretasikan data. Perangkat penelitian yang digunakan dalam penelitian ini meliputi perangkat keras (*hardware*) dan perangkat lunak (*software*) yang dipilih sesuai dengan kebutuhan penelitian. Berikut deskripsi tiap – tiap perangkat :

Tabel 3. 1 Kebutuhan Perangkat Keras

No.	Nama perangkat	Deskripsi perangkat
1	Laptop	Lenovo Legion Slim 5 16APH8
2	Processor	AMD Ryzen 7840HS
3	Graphical processing Unit (GPU)	NVIDIA GeForce RTX 4050

4	Random access memory (RAM)	16 GB
5	Storage	512 GB SSD

Tabel 3.2 Kebutuhan Perangkat Lunak

No.	Nama perangkat	Deskripsi perangkat
1	<i>Windows 11 64-bit</i>	Sistem operasi
2	<i>Anaconda Environment</i>	<i>Tools</i> yang digunakan untuk pengelolaan dan membangun sistem serta membuat dan melatih model yang akan digunakan.
3	<i>Python 3.9</i>	Bahasa pemrograman yang akan digunakan untuk membangun sistem
4	<i>OS</i>	Pustaka <i>python</i> untuk membaca dan menulis sistem berkas pada sistem operasi
5	<i>Numpy</i>	Pustaka <i>python</i> untuk memanipulasi array multidimensi
6	<i>Matplotlib</i> dan <i>Seaborn</i>	Pustaka <i>python</i> untuk visualisasi data
7	<i>OpenCV</i>	Pustaka untuk pemrosesan gambar
8	<i>Scikit-learn</i>	Pustaka <i>python</i> yang menyediakan algoritma dan fungsi untuk analisis data, termasuk dalam <i>clustering</i>

3.6 Waktu penelitian

Tabel 3.3 Waktu Penelitian

No.	Tahapan	2024	2025	2025	2025	2025
		Desember	Januari	Februari	Maret	April
1	Pengajuan Judul					
2	Pengumpulan data					
3	Penyusunan Proposal					
4	Seminar proposal					
5	Analisis data dan penelitian					
6	Penyusunan skripsi					
7	Sidang meja hijau					
8	Penyempurnaan skripsi dan penulisan artikel jurnal					

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil Penelitian

4.1.1 Dataset Pelatihan

Penelitian ini menggunakan dataset citra penyakit tanaman padi yang terdiri dari 4 kategori diantaranya *Brown spot disease* (Bercak coklat), *Blast disease* (Bercak daun), *False smut disease* (Gosong palsu), *Bacterial leaf blight disease* (Hawar daun bakteri) dan *Healthy* (Daun sehat). Dataset diambil melalui *platform Kaggle* dengan total sebanyak 2095 gambar. Data gambar yang diambil memiliki ukuran 244 x 244 dengan pencahayaan normal dan kualitas gambar yang perlu di perhatikan agar dapat meningkatkan kinerja pada proses pelatihan model.

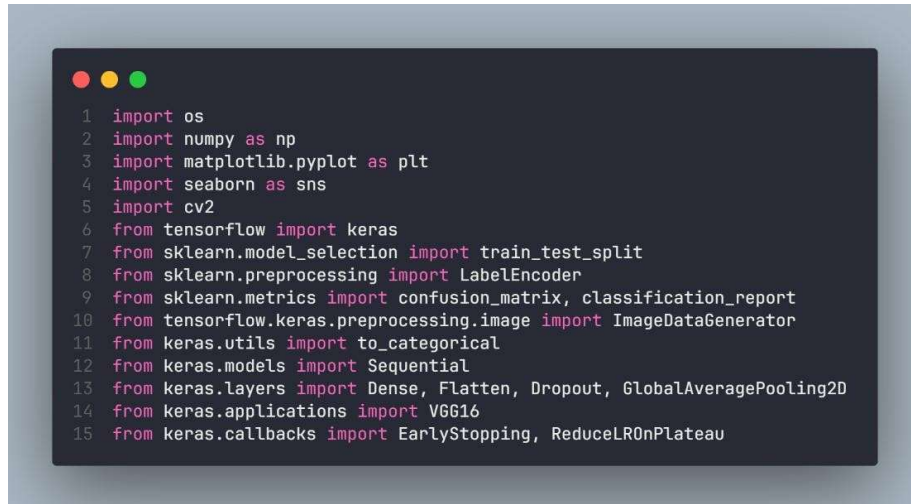
4.1.2 Pelatihan Model

Pelatihan model terdiri dari beberapa tahap yang dimana proses ini dilakukan setelah mengumpulkan dataset untuk pelatihan model yang telah dijelaskan sebelumnya. Penelitian ini akan membandingkan performa dari kedua arsitektur yaitu VGG16 dan ResNet50. Berikut adalah beberapa proses dan tahap yang dilalui untuk pelatihan model yaitu :

1. Import Library

Import library merupakan suatu proses dalam memuat pustaka eksternal ke dalam program yang berfungsi agar dapat menggunakan modul – modul yang sudah tersedia di dalamnya. Perintah ini dilakukan dengan mengetik perintah *import* agar bisa mengakses berbagai fitur tanpa perlu menulis

ulang kodenya dari 0 yang akan menghemat waktu. Berikut adalah library yang di import untuk memproses pelatihan ini :



```
1 import os
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import cv2
6 from tensorflow import keras
7 from sklearn.model_selection import train_test_split
8 from sklearn.preprocessing import LabelEncoder
9 from sklearn.metrics import confusion_matrix, classification_report
10 from tensorflow.keras.preprocessing.image import ImageDataGenerator
11 from keras.utils import to_categorical
12 from keras.models import Sequential
13 from keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D
14 from keras.applications import VGG16
15 from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

Gambar 4.1.1 Import Library

Adapun *library* yang di *import* antara lain yaitu *import os* yang berfungsi mengelola sistem file dan direktori sekaligus membaca file dataset dan mengatur *path file*. *Import numpy as np* berfungsi sebagai manipulasi *array* numerik dan digunakan untuk menyimpan dan mengolah data dalam bentuk *array*. *Import matplotlib.pyplot as plt* berfungsi untuk visualisasi data dan digunakan untuk menampilkan kurva *training* ataupun gambar dataset. *import seaborn as sns* berfungsi sebagai visualisasi data berbasis *matplotlib* tetapi dengan tampilan yang lebih menarik dan membantu memvisualisasikan distribusi data atau *heatmap confusion matrix* hasil evaluasi model. *Import cv2* digunakan untuk pengolahan citra dan digunakan untuk membaca, mengedit dan memproses gambar.

Berikutnya ada *from tensorflow import keras* berfungsi untuk menggunakan fitur Keras dalam *Tensorflow* yang digunakan untuk membangun, melatih, dan mengelola model *deep learning*. *from sklearn.model_selection import train_test_split* digunakan untuk membagi dataset menjadi data *training* dan testing misalnya 80% *training* 20% testing. *from sklearn.preprocessing import LabelEncoder* digunakan untuk mengkonversi label kategori menjadi angka agar bisa mengubah label dari kelas menjadi numerik agar bisa digunakan oleh model. *from sklearn.metrics import confusion_matrix, classification_report* digunakan untuk menghitung metrik evaluasi untuk model klasifikasi. *from tensorflow.keras.preprocessing.image import ImageDataGenerator* berfungsi untuk menggunakan augmentasi data gambar untuk memperbanyak variasi gambar saat *training*. *from keras.utils import to_categorical* digunakan untuk mengonversi label numerik menjadi *one-hot-encoding*. *from keras.models import Sequential* digunakan untuk membuat model *neural network* secara berurutan. *from keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D* berfungsi untuk menggunakan beberapa layer dalam *deep learning*. *from keras.applications import VGG16, ResNet50* menggunakan model VGG16 dan ResNet50 yang telah dilatih sebelumnya (Pre-Trained Model) tanpa melatih dari awal. *from keras.callbacks import EarlyStopping, ReduceLROnPlateau* memiliki fungsi *callbacks* untuk mengontrol proses pelatihan model.

2. Memuat dan memproses dataset

Langkah berikutnya adalah memuat dan memproses dataset yang sudah dikumpulkan sebelumnya. Berikut adalah pembahasannya secara menyeluruh :

```
1 #import path gambar
2 image_path = r"C:\Users\Lenovo\Downloads\Rice_Diseases\Rice_Diseases"
3 categories = ["Bacterial Blight Disease", "Blast Disease", "Brown Spot Disease", "False Smut Disease", "Healthy"]
4
5 def load_images(image_path, categories):
6     img_size = (128, 128)
7     images = []
8     labels = []
9
10    #iterasi kategori dan pengecekan data
11    for category in categories:
12        category_path = os.path.join(image_path, category)
13
14        # mengecek kategori apakah ada atau tidak
15        if not os.path.exists(category_path):
16            print(f"Path not found: {category_path}")
17            continue
18
19        #membaca dan memproses gambar
20        for filename in os.listdir(category_path):
21            if filename.endswith(".jpg") or filename.endswith(".jpeg") or filename.endswith(".png"):
22                img_path = os.path.join(category_path, filename)
23                img = cv2.imread(img_path)
24                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
25                img = cv2.resize(img, img_size)
26                images.append(img)
27                labels.append(category)
28
29        #konversi dari numpy ke Array
30    return np.array(images), np.array(labels)
31
32 #menampilkan Jumlah data
33 images, labels = load_images(image_path, categories)
34 print(len(images))
35 print(len(labels))
```

Gambar 4.1.2 Pemrosesan Dataset

Pada bagian *path dataset* terdapat *image_path* sebagai Path folder utama yang berisi dataset gambar. Kemudian ada *categories* merupakan daftar nama folder yang berisi gambar untuk setiap kategori penyakit padi. Kemudian fungsi *load_images* yang berfungsi untuk membaca, memproses, dan menyimpan gambar. Pada bagian membaca dan memproses gambar terjadi proses mengambil semua file yang ada dalam kategori dengan memastikan hanya file yang berupa gambar yang akan di proses yang kemudian akan di *Resize* menjadi 128 x 128 *pixel* agar ukuran gambar seragam yang kemudian akan gambar – gambar tersebut akan dikonversi

menjadi *Array Numpy* agar mudah digunakan dalam model. Terakhir menjalankan fungsi menampilkan jumlah data yang berhasil dimuat dan jumlah labelnya.

3. Datasets preprocessing

Langkah berikutnya adalah menampilkan sampel gambar dari dataset setelah *preprocessing* seperti pada gambar 4.1.3..

```
1 #menampilkan 9 gambar acak dari datasets
2 def display_sample_images(images, labels, class_names):
3     plt.figure(figsize=(10, 10))
4     for i in range(9):
5         idx = np.random.randint(len(images))
6         plt.subplot(3, 3, i + 1)
7         plt.imshow(images[idx])
8         plt.title("Pred: " + class_names[labels[idx]])
9         plt.axis('off')
10    plt.show()
```

Gambar 4.1.3 Data Pre-processing

Fungsi ini memiliki tujuan untuk memilih 9 gambar secara acak dari dataset yang telah di proses sebelumnya kemudian menampilkannya dalam *grid* 3x3 dengan memberikan judul gambar dengan nama kelas berdasarkan labelnya kemudian menghilangkan sumbunya agar tampilannya lebih rapi.

```
1 #Label encoding dan One-hot encoding
2 le = LabelEncoder()
3 labels_encoded = le.fit_transform(labels)
4 labels_encoded = to_categorical(labels_encoded)
```

Gambar 4.1.4 Proses Encoding

Pada gambar 4.1.4.. terjadi proses *Label encoding* dan *One-Hot encoding* yang dimana proses *LabelEncoder()* mengonversinya menjadi angka misalnya, “*Bacterial Blight Disease*” → 0, “*Blast Disease*” → 1 dan sampai seterusnya. Kemudian proses *to_categorical(labels_encoded)* mengonversi label numerik menjadi *One-Hot encoding* yang merupakan bagian yang penting untuk model *deep learning* yang menggunakan *Softmax* di *output layer*.



```
1 #Normalisasi Gambar
2 images = images / 255.0
```

Gambar 4.1.5 Normalisasi Data

Kemudian pada gambar 4.1.5 membagi setiap *pixel* gambar dengan 255 agar nilainya berada dalam rentang 0-1 yang membantu mempercepat konvergensi saat pelatihan model.



```
1 #Menampilkan gambar dan mendapatkan nama kelas
2 class_names = le.classes_
3 display_sample_images(images, labels_encoded.argmax(axis=1), class_names)
```

Gambar 4.1.6 Mengkategorikan Gambar

Pada gambar 4.1.6 mendapatkan nama kelas yang telah diencode terdapat fungsi *display_sample_images()* agar menampilkan gambar dengan nama kelasnya, Hasilnya sebagai berikut!



Gambar 4.1.7 Menampilkan Gambar Berdasarkan Kelas

Kemudian pada gambar 4.1.8 dilanjutkan dengan membagi dataset menjadi 80% data *training* dan 20% untuk data testing dengan pembagian data yang konsisten setiap kali dijalankan.

```

1 X_train, X_test, y_train, y_test = train_test_split(
2     images, labels_encoded, test_size=0.2, random_state=42)

```

Gambar 4.1.8 Training test_split

4. Data Augmentation

Langkah selanjutnya adalah Augmentasi data atau peningkatan variasi data yang dimana pada proses ini yang pertama adalah memutar gambar maksimal 40 derajat secara acak kemudian menggeser gambar horizontal dan vertikal sebanyak 20% dari ukuran aslinya. Kemudian ada *shear_range=0.2* yaitu menerapkan distorsi kemiringan (shear) hingga 20% memperbesar atau memperkecil gambar hingga 20% kemudian membalik gambar secara horizontal dan mengisi area kosong akibat transformasi dengan warna *pixel* terdekat dan fungsi *datagen.fit(x_train)* digunakan untuk menerapkan persiapan augmentasi pada dataset *training*.

```
1 # Data Augmentation
2 datagen = ImageDataGenerator(
3     rotation_range=40, width_shift_range=0.2,
4     height_shift_range=0.2, shear_range=0.2, zoom_range=0.2,
5     horizontal_flip=True, fill_mode='nearest'
6 )
7
8 datagen.fit(X_train)
```

Gambar 4.1.9 Augmentasi Data

5. Model ResNet50

```
1 resnet_base = ResNet50(weights='imagenet', include_top=False, input_shape=(128, 128, 3))
2
3 model = Sequential()
4 model.add(resnet_base)
5 model.add(GlobalAveragePooling2D())
6 model.add(Dense(512, activation='relu'))
7 model.add(Dropout(0.5))
8 model.add(Dense(256, activation='relu'))
9 model.add(Dropout(0.3))
10 model.add(Dense(len(categories), activation='softmax'))
11
12 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
13
```

Gambar 4.1.10 Model ResNet50

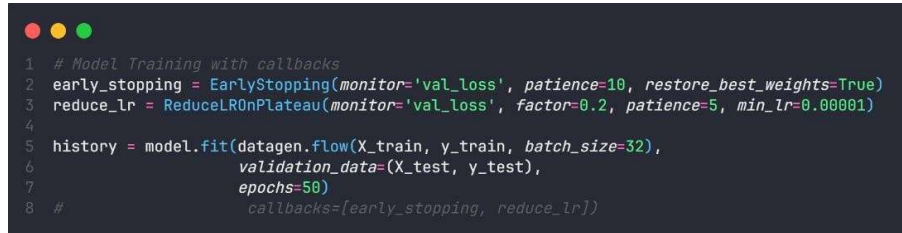
Pada gambar 4.1.10 menggunakan model ResNet50 yang telah dilatih di *ImageNet* sebagai *Feature extractor*, *weights="imagenet"* berfungsi memuat bobot *pretrained* dari dataset *ImageNet*. Kemudian, *include_top=False* menghapus *fully connected* bawaan ResNet50, agar bisa di gunakan untuk kasus dataset penyakit tanaman padi. *Input_shape=(128, 128, 3)* merupakan ukuran input gambar 128 x 128 pixel dengan 3 channel warna (RGB).

Tahap berikutnya yaitu membuat model dengan fungsi *sequential()* dengan menambahkan *resnet_base* sebagai lapisan pertama dan menambahkan lapisan klasifikasi yaitu *model.add(Global AveragePooling2D())* untuk menggantikan *fully connected layer* dari ResNet50 dengan mengubah fitur keluaran dari ResNet50 menjadi 1D vektor. Kemudian, menambahkan *fully connected layer* sebanyak 512 neuron yang dituliskan dalam fungsi *model.add(Dense(512, activation='relu'))* dan mencegah overfitting dengan membuang 50% neuron acak saat training *model.add(Dropout(0.5))*.

Kemudian, ditambahkan *fully connected layer* sebanyak 256 neuron dan dropout sebanyak 30%. Lalu dibuat layer output untuk klasifikasi *model.add(Dense(len(categories), activation='softmax'))* dengan jumlah kelas (kategori penyakit tanaman padi) dan *softmax* digunakan sebagai fungsi aktivasi karena penelitian ini merupakan *multi-class classification problem*. Terakhir, menggunakan *Adam Optimizer* untuk training dan *categorical_crossentropy* digunakan karena penelitian ini memiliki 4 kelas

dan menggunakan *one-hot encoding* kemudian menggunakan akurasi sebagai metrik evaluasi.

6. Optimasi Training ResNet50



```
1 # Model Training with callbacks
2 early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
3 reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.00001)
4
5 history = model.fit(datagen.flow(X_train, y_train, batch_size=32),
6                     validation_data=(X_test, y_test),
7                     epochs=50)
8 #
9 callbacks=[early_stopping, reduce_lr])
```

Gambar 4.1.11 Optimasi training model

Pada gambar 4.1.11 digunakan fungsi *Callbacks* untuk menghentikan *training* lebih awal jika model mulai *overfitting* dan menyesuaikan *learning rate* jika diperlukan. *Monitor_val*= 'val_loss' digunakan untuk memantau nilai *loss* pada data validasi yang dimana fungsi *patience*=10 digunakan jika nilai *loss* tidak membaik selama 10 *epoch* berturut – turut maka proses training akan dihentikan. Kemudian, setelah training berhenti model akan menggunakan bobot terbaik yang diperoleh selama training.

Jika model kesulitan dalam memperbaiki *loss*, *learning rate* diturunkan agar bisa belajar dengan lebih baik. *Factor*=0.2 akan mengurangi *learning rate* menjadi 20% dari nilai sebelumnya kemudian, jika dalam 5 *epoch* berturut – turut tidak membaik maka *learning rate* akan dikurangi dengan batas minimum *learning rate* yaitu 0.00001 agar model tetap bisa berjalan. Model akan dilatih dengan *Data Augmentation* dengan menggunakan 32 gambar per *batch* dalam satu iterasi dan data testing berupa *validation_data*=(*X_test*, *y_test*) tidak diaugmentasi yang akan digunakan untuk evaluasi model dengan model yang dilatih maksimal pada 50 *epochs*.

```

Epoch 1/50
S2/S2 [=====] - 26s 169ms/step - loss: 0.8950 - accuracy: 0.7183 - val_loss: 5882.4849 - val_accuracy: 0.3154
Epoch 2/50
S2/S2 [=====] - 7s 138ms/step - loss: 0.5541 - accuracy: 0.8469 - val_loss: 6065.5811 - val_accuracy: 0.3154
Epoch 3/50
S2/S2 [=====] - 7s 137ms/step - loss: 0.3377 - accuracy: 0.8861 - val_loss: 1.6255 - val_accuracy: 0.3252
Epoch 4/50
S2/S2 [=====] - 7s 140ms/step - loss: 0.3193 - accuracy: 0.9845 - val_loss: 2.0408 - val_accuracy: 0.3252
Epoch 5/50
S2/S2 [=====] - 7s 140ms/step - loss: 0.3083 - accuracy: 0.9241 - val_loss: 1.9671 - val_accuracy: 0.8782
Epoch 6/50
S2/S2 [=====] - 7s 139ms/step - loss: 0.9996 - accuracy: 0.8108 - val_loss: 2.5102 - val_accuracy: 0.1394
Epoch 7/50
S2/S2 [=====] - 7s 142ms/step - loss: 0.4164 - accuracy: 0.8702 - val_loss: 2.0974 - val_accuracy: 0.1394
Epoch 8/50
S2/S2 [=====] - 7s 140ms/step - loss: 0.3997 - accuracy: 0.8959 - val_loss: 5.0664 - val_accuracy: 0.1418
Epoch 9/50
S2/S2 [=====] - 7s 141ms/step - loss: 0.2758 - accuracy: 0.9118 - val_loss: 2.3478 - val_accuracy: 0.1418
Epoch 10/50
S2/S2 [=====] - 7s 140ms/step - loss: 0.2920 - accuracy: 0.9167 - val_loss: 5.8580 - val_accuracy: 0.1394

```

Gambar 4.1.12 Hasil epochs

Gambar 4.1.12 dapat dilihat bagaimana performanya pada data validasi hasil dari 10 *epochs* pertama. *Training loss & Accuracy* menunjukkan bahwa *loss* menurun yang dimana awalnya sangat tinggi dari 0.8950 menjadi 0.2930, artinya model semakin bisa menyesuaikan dengan data. Akurasi meningkat dari 71.83% menjadi 91.67% ini menunjukkan bahwa model mulai mengenali pola dalam dataset.

Pada *Validation loss & Accuracy* yang dimana awalnya *val_loss* sangat tinggi yaitu 5882.4849 dan tetap tinggi di *epochs* selanjutnya. Kemudian *val_accuracy* mengalami stagnan di angka 31-31%, lalu turun drastis menjadi 7.82% di *epoch 5* dan tetap buruk setelahnya yang dimana ini artinya adalah model gagal menggeneralisasi ke data validasi.

```

Epoch 40/50
S2/S2 [=====] - 8s 148ms/step - loss: 0.1353 - accuracy: 0.9602 - val_loss: 0.2283 - val_accuracy: 0.9169
Epoch 41/50
S2/S2 [=====] - 8s 146ms/step - loss: 0.2006 - accuracy: 0.9443 - val_loss: 1.1447 - val_accuracy: 0.6773
Epoch 42/50
S2/S2 [=====] - 8s 145ms/step - loss: 0.1329 - accuracy: 0.9528 - val_loss: 0.3055 - val_accuracy: 0.9193
Epoch 43/50
S2/S2 [=====] - 8s 145ms/step - loss: 0.1194 - accuracy: 0.9633 - val_loss: 0.1858 - val_accuracy: 0.9511
Epoch 44/50
S2/S2 [=====] - 8s 147ms/step - loss: 0.1138 - accuracy: 0.9639 - val_loss: 0.3092 - val_accuracy: 0.9438
Epoch 45/50
S2/S2 [=====] - 8s 146ms/step - loss: 0.1429 - accuracy: 0.9645 - val_loss: 48.0884 - val_accuracy: 0.3521
Epoch 46/50
S2/S2 [=====] - 8s 144ms/step - loss: 0.3206 - accuracy: 0.9088 - val_loss: 2.0755 - val_accuracy: 0.5379
Epoch 47/50
S2/S2 [=====] - 8s 146ms/step - loss: 0.2115 - accuracy: 0.9351 - val_loss: 0.5100 - val_accuracy: 0.8704
Epoch 48/50
S2/S2 [=====] - 8s 147ms/step - loss: 0.1489 - accuracy: 0.9528 - val_loss: 0.2366 - val_accuracy: 0.9389
Epoch 49/50
S2/S2 [=====] - 8s 145ms/step - loss: 0.1399 - accuracy: 0.9559 - val_loss: 0.1446 - val_accuracy: 0.9462
Epoch 50/50
S2/S2 [=====] - 8s 148ms/step - loss: 0.1174 - accuracy: 0.9614 - val_loss: 0.2125 - val_accuracy: 0.9413

```

Gambar 4.1.13 Hasil epochs kedua

Kemudian terlihat pada gambar 4.1.13 setelah selesai 50 *epochs* dapat dilihat performa model sudah membaik dari sebelumnya dimana *Loss* turun menjadi 0.1174 yang artinya model sudah cukup baik dalam mengenali data *training* dengan akurasi sebesar 96.14% yang menunjukkan bahwa model dengan sangat baik mengenali pola pada dataset *training*.

Validation loss naik menjadi 94.13% di akhir *epoch* ini menunjukkan bahwa model bisa mengenali data validasi dengan baik. Pada bagian *val_loss* awalnya stabil tetapi ada beberapa lonjakan yang cukup aneh yaitu pada *epoch* 45 *val_loss* tiba – tiba naik ke angka 48.0884 dan *accuracy* turun drastis menjadi 35.21%. *epoch* ke – 46 *val_loss* turun menjadi 2.0755 dan *val_accuracy* kembali naik menjadi 53.79% kemudian, pada *epoch* 47 – 50 *val_loss* kembali normal disekitar 0.1446 – 0.2366 dan *val_accuracy* meningkat kembali menjadi 94.13% yang dimana ini artinya model hampir sempurna di training set, tetapi ada fluktuasi aneh pada *validation loss* dengan waktu *training* rata – rata 7 – 8 detik per *epoch*.

7. Model VGG16

```
1 vgg_base = VGG16(weights='imagenet', include_top=False, input_shape=(128, 128, 3))
2
3 model = Sequential()
4 model.add(vgg_base)
5 model.add(GlobalAveragePooling2D())
6 model.add(Dense(512, activation='relu'))
7 model.add(Dropout(0.5))
8 model.add(Dense(len(categories), activation='softmax'))
9
10 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Gambar 4.1.14 Model VGG16

Model VGG16 yang akan dilatih sama seperti model ResNet50 diatas dimana model dilatih berdasarkan dengan yang telah dilatih pada *ImageNet*.

Dimana fungsi `weights='ImageNet'` memuat bobot pretrained dari *ImageNet*, kemudian menghilangkan fungsi *fully connected layer* dengan fungsi `include_top=False` bawaan agar bisa menambahkan lapisan yang baru.

Kemudian untuk model *sequential* ditambahkan *vgg.base* sebagai dasar model VGG16 dan fungsi *GlobalAveragePooling2D()* digunakan untuk meratakan fitur dari CNN agar tidak memiliki terlalu banyak parameter, dan ditambahkan dengan 512 neuron dengan aktivasi *ReLU* agar model dapat mempelajari pola yang kompleks dengan fungsi `model.add(Dense(512, activation='relu'))` lalu untuk mencegah *overfitting* maka membuang 50% dari lapisan neuron secara acak saat *training* dan menambahkan fungsi `model.add(Dense(len(categories), activation='softmax'))` untuk lapisan output sesuai dengan jumlah kelas pada dataset menggunakan *softmax* untuk klasifikasi.

Untuk kompilasi model digunakan `optimizer='adam'` yang cepat dan efisien untuk model *deep learning* dan *categorical_crossentropy* digunakan untuk klasifikasi multi kelas dan untuk mengevaluasi modelnya berdasarkan akurasi.

8. Optimasi Training VGG16

```
1 # Model Training with callbacks
2 early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
3 reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.00001)
4
5 history = model.fit(datagen.flow(X_train, y_train, batch_size=32),
6                       validation_data=(X_test, y_test),
7                       epochs=50)
8 #
9 callbacks=[early_stopping, reduce_lr]
```

Gambar 4.1.15 Optimasi training VGG16

Pada gambar 4.1.15 digunakan fungsi *Callbacks* untuk menghentikan training lebih awal jika model mulai *overfitting* dan menyesuaikan *learning rate* jika diperlukan. *Monitor_val='var_loss'* digunakan untuk memantau nilai loss pada data validasi yang dimana fungsi *patience=10* digunakan jika nilai loss tidak membaik selama 10 *epoch* berturut – turut maka proses training akan dihentikan. Kemudian, setelah training berhenti model akan menggunakan bobot terbaik yang diperoleh selama training.

Jika model kesulitan dalam memperbaiki loss, *learning rate* diturunkan agar bisa belajar dengan lebih baik. *Factor=0.2* akan mengurangi *learning rate* menjadi 20% dari nilai sebelumnya kemudian, jika dalam 5 *epoch* berturut – turut tidak membaik maka *learning rate* akan dikurangi dengan batas minimum *learning rate* yaitu 0.00001 agar model tetap bisa berjalan. Model akan dilatih dengan *Data Augmentation* dengan menggunakan 32 gambar per *batch* dalam satu iterasi dan data testing berupa *validation_data=(X_test, y_test)* tidak diaugmentasi yang akan digunakan untuk evaluasi model dengan model yang dilatih maksimal pada 50 *epochs*.

```
Epoch 1/50
52/52 [=====] - 34s 319ms/step - loss: 2.2536 - accuracy: 0.3007 - val_loss: 1.4759 - val_accuracy: 0.3252
Epoch 2/50
52/52 [=====] - 13s 251ms/step - loss: 1.4041 - accuracy: 0.3613 - val_loss: 1.3147 - val_accuracy: 0.3325
Epoch 3/50
52/52 [=====] - 13s 252ms/step - loss: 1.2283 - accuracy: 0.4081 - val_loss: 1.1334 - val_accuracy: 0.5452
Epoch 4/50
52/52 [=====] - 13s 252ms/step - loss: 0.9466 - accuracy: 0.6246 - val_loss: 0.8109 - val_accuracy: 0.6968
Epoch 5/50
52/52 [=====] - 14s 258ms/step - loss: 1.0519 - accuracy: 0.5977 - val_loss: 1.2459 - val_accuracy: 0.5061
Epoch 6/50
52/52 [=====] - 13s 251ms/step - loss: 1.0251 - accuracy: 0.5707 - val_loss: 0.8584 - val_accuracy: 0.6504
Epoch 7/50
52/52 [=====] - 13s 255ms/step - loss: 0.9049 - accuracy: 0.6320 - val_loss: 0.9492 - val_accuracy: 0.6333
Epoch 8/50
52/52 [=====] - 13s 254ms/step - loss: 0.9426 - accuracy: 0.6405 - val_loss: 0.7035 - val_accuracy: 0.7237
Epoch 9/50
52/52 [=====] - 13s 254ms/step - loss: 0.7204 - accuracy: 0.7128 - val_loss: 0.7199 - val_accuracy: 0.7384
Epoch 10/50
52/52 [=====] - 13s 253ms/step - loss: 0.7247 - accuracy: 0.7263 - val_loss: 0.5270 - val_accuracy: 0.8166
```

Gambar 4.1.16 Hasil epochs VGG16

Pada gambar 4.1.16 adalah hasil dari 10 *epochs* pertama dalam *training* model VGG16 pertama, *loss* awalnya cukup tinggi berkisar di angka 2.2536

di *epoch* 1 yang dimana ini menunjukkan bahwa model masih kesulitan dalam memahami pola pada data latih dan pada *epoch* 10 secara bertahap *loss* menurun hingga 0.7247 yang menandakan model mulai belajar dengan baik, Kemudian akurasi meningkat dari 30.07% pada *epoch* 1 menjadi 72.63% pada *epoch* 10 yang dimana ini menunjukkan bahwa model sudah mulai mengenali pola dengan lebih baik karena menunjukkan akurasi yang signifikan.

Pada bagian *validation accuracy* naik dari 32.52% pada *epoch* 1 menjadi 81.66% pada *epoch* 10 yang menandakan bahwa model tidak hanya belajar dari data latih , tetapi juga cukup baik dalam mengenali data yang belum pernah dilihat. Untuk *validation loss* turun dari 1.4752 menjadi 0.5270 pada *epoch* 10 yang menunjukkan model semakin akurat dalam memprediksi data validasi , Namun terjadi fluktuasi pada *val_loss* terutama pada *epoch* 5 dan 7, dimana *val_loss* sempat naik menjadi 1.2459 pada *epoch* 5 kemudian turun menjadi 0.7035 pada *epoch* 8.

```
Epoch 40/50
S2/S2 [=====] - 14s 261ms/step - loss: 0.3532 - accuracy: 0.8745 - val_loss: 0.3864 - val_accuracy: 0.8582
Epoch 41/50
S2/S2 [=====] - 14s 261ms/step - loss: 0.4054 - accuracy: 0.8530 - val_loss: 0.3949 - val_accuracy: 0.8802
Epoch 42/50
S2/S2 [=====] - 14s 262ms/step - loss: 0.3804 - accuracy: 0.8775 - val_loss: 0.4168 - val_accuracy: 0.8688
Epoch 43/50
S2/S2 [=====] - 14s 261ms/step - loss: 0.4551 - accuracy: 0.8610 - val_loss: 0.3873 - val_accuracy: 0.8784
Epoch 44/50
S2/S2 [=====] - 14s 262ms/step - loss: 0.3897 - accuracy: 0.8788 - val_loss: 0.3999 - val_accuracy: 0.8688
Epoch 45/50
S2/S2 [=====] - 14s 261ms/step - loss: 0.3280 - accuracy: 0.8922 - val_loss: 0.4094 - val_accuracy: 0.8688
Epoch 46/50
S2/S2 [=====] - 14s 264ms/step - loss: 0.3713 - accuracy: 0.8788 - val_loss: 0.3272 - val_accuracy: 0.9022
Epoch 47/50
S2/S2 [=====] - 14s 264ms/step - loss: 0.3105 - accuracy: 0.8953 - val_loss: 0.4171 - val_accuracy: 0.8688
Epoch 48/50
S2/S2 [=====] - 14s 264ms/step - loss: 0.2678 - accuracy: 0.9869 - val_loss: 0.4615 - val_accuracy: 0.8289
Epoch 49/50
S2/S2 [=====] - 14s 262ms/step - loss: 0.2703 - accuracy: 0.9118 - val_loss: 0.2822 - val_accuracy: 0.9218
Epoch 50/50
S2/S2 [=====] - 14s 264ms/step - loss: 0.2598 - accuracy: 0.9143 - val_loss: 0.3108 - val_accuracy: 0.9095
```

Gambar 4.1.17 Hasil epochs kedua VGG16

Kemudian pada gambar 4.1.17 hasil dari keseluruhan *epochs* 50 dapat dilihat bahwa *loss* menurun dari 0.3532 menjadi 0.2598 yang menandakan model terus belajar dan menyempurnakan kemampuannya. Pada *accuracy*

meningkat dari 87.45% menjadi 91.43%. Kemudian pada bagian *val_loss* terjadi fluktuasi tetapi menurun setelahnya, dari angka 0.3864 menjadi 0.3108.

Terdapat beberapa fluktuasi aneh ada *epoch* 47 – 48, dimana *val_loss* naik menjadi 0.4615 dan *val_accuracy* turun menjadi 82.89% tetapi kembali membaik di *epoch* berikutnya yaitu pada *epoch* 49 *val_loss* turun menjadi 0.2822 dan *val_accuracy* naik menjadi 0.9218, Hasilnya model masih generalisasi dengan baik dan hanya ada sedikit inkonsistensi dalam validasi dan untuk waktu *training* per *epoch* rata – rata sekitar 13 – 14 detik.

4.1.3 Evaluasi Model

1. Testing Accuracy ResNet50

Setelah training selesai, selanjutnya adalah mengevaluasi seberapa baik model bekerja pada data yaitu dengan uji testing seperti yang terlihat pada gambar 4.1.18



```
1 # Evaluate Model
2 test_loss, test_accuracy = model.evaluate(X_test, y_test)
3 print(f'Test Accuracy: {test_accuracy * 100:.2f}%')
```

Proses ini menerapkan fungsi *test_loss*, *test_accuracy* = *model.evaluate(X_test, y_test)* dengan memasukkan data uji (*X_test*, *y_test*) ke dalam model yang dimana model akan menghitung *loss* dan *accuracy* pada data uji tanpa melakukan proses training lagi proses ini mirip dengan validasi, tetapi menggunakan data yang belum pernah dipakai sebelumnya. Kemudian ada *test_loss* yaitu mengukur seberapa besar

kesalahan pada data uji dan *test_accuracy* yaitu mengukur presentase prediksi yang benar pada data uji.

```
13/13 [=====] - 1s 38ms/step - loss: 0.2125 - accuracy: 0.9413  
Test Accuracy: 94.13%
```

Gambar 4.1.18 Hasil evaluasi model ResNet50

Berikut adalah hasil dari testing dari model ResNet50 mendapatkan akurasi yang tinggi dengan angka 94.13% artinya model mampu mengenali pola dengan sangat baik pada data uji dan tidak ada *overfitting* yang signifikan karena *accuracy* tetap tinggi, untuk *loss* mendapatkan nilai yang rendah yaitu 0.2125 yang menunjukkan bahwa prediksi model tidak jauh melenceng dari label aslinya, meskipun tidak sempurna tetapi model cukup stabil dalam membuat keputusan.

Model ResNet50 menunjukkan performa yang sangat baik dalam mengenali data training dengan akurasi 96.13% dan loss sebesar 0.1174, tanpa indikasi overfitting yang ekstrem. *validation loss* dan *test loss* yang hampir sama yaitu 0.2125 mengonfirmasi bahwa model tetap stabil saat diuji dengan data baru. Jika terjadi overfitting, maka *test loss* jauh lebih tinggi dibanding *training loss*, namun dalam kasus ini, keduanya seimbang. Selain itu, akurasi pada *validation* dan *test set* yang sama, yaitu 94.13%, menunjukkan bahwa model tidak hanya menghafal data *training*, tetapi juga mampu mengenali pola dengan baik pada data.

```
13/13 [=====] - 1s 55ms/step - loss: 0.3108 - accuracy: 0.9095  
Test Accuracy: 90.95%
```

Gambar 4.1.19 Hasil Evaluasi model VGG16

Berikutnya adalah hasil dari *testing* pada model VGG16 yang terdapat pada gambar 4.1.19 mendapatkan akurasi yang juga cukup tinggi yaitu 90.95% dari seluruh sampel dalam dataset uji. Dengan ini menunjukkan bahwa model dapat mengenali pola dalam gambar dengan baik dan memiliki kemampuan generalisasi yang baik. Nilai *loss* masih dalam batas wajar yang menunjukkan bahwa kesalahan prediksi yang terjadi tidak terlalu signifikan. Tetapi kemungkinan masih ada beberapa kategori yang sulit dibedakan oleh model. Berikut tabel perbandingan dari hasil testing kedua model seperti yang ditunjukkan pada tabel 4.1:

Tabel 4.1 Perbandingan antara kedua model

Model	Test loss	Test accuracy
ResNet50	0.2125	94.13%
VGG16	0.3108	90.95%

Analisis perbandingan hasil testing dari kedua model yaitu dalam hal akurasi model ResNet50 lebih unggul jika dibandingkan dengan VGG16 dimana model ResNet50 mendapatkan akurasi sebesar 94.13% sedangkan model VGG16 mendapatkan akurasi sebesar 90.95% dalam hal ini artinya model ResNet50 lebih baik dalam mengenali pola dan membuat prediksi yang benar pada data uji. Kemudian, untuk *loss* model ResNet50 memiliki nilai yang rendah yaitu dengan angka 0.2125 dibandingkan dengan model VGG16 mendapatkan angka 0.3108, untuk hal ini nilai *loss* yang lebih kecil menunjukkan bahwa model lebih baik dalam meminimalkan kesalahan selama prediksi.

Kemudian pada bagian *generate prediction* yang di tunjukkan pada gambar... model yang sudah dilatih digunakan untuk memprediksi data uji yaitu *X_test*, mengubah hasil prediksi probabilitas menjadi kelas dengan fungsi *argmax* dan mengubah label yang sebenarnya yaitu *y_test* menjadi format yang sama untuk perbandingan.

```
1 # Generate predictions
2 y_pred = model.predict(X_test)
3 y_pred_classes = np.argmax(y_pred, axis=1)
4 y_true = np.argmax(y_test, axis=1)
```

Gambar 4.1.20 menunjukkan *output* pada model ResNet50 yang menunjukkan waktu *interfensi*/prediksi yang dimana model telah memproses 13 dari 13 *batch* data dengan total waktu yang dibutuhkan adalah 1 detik dan waktu rata – rata yang dibutuhkan per *batch* adalah 33 milidetik.

```
13/13 [=====] - 1s 33ms/step
```

Gambar 4.1.20 Waktu interfensi ResNet50

Selanjutnya pada model VGG16 menunjukkan *output* waktu interfensi/prediksi yaitu model telah memproses 13 dari 13 *batch* data dengan total waktu yang dibutuhkan adalah 1 detik dengan waktu rata – rata yang dibutuhkan per *batch* adalah 56 milidetik.

```
13/13 [=====] - 1s 56ms/step
```

Gambar 4.1.21 Waktu interfensi VGG16

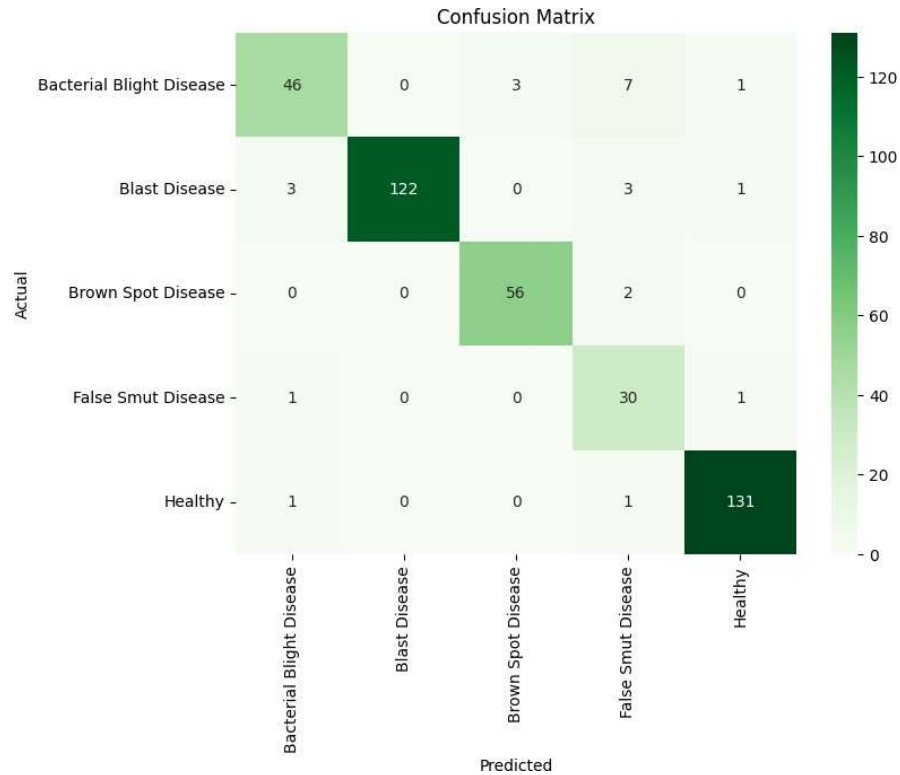
2. Confusion Matrix ResNet50

Selanjutnya adalah menganalisis performa model menggunakan *confusion matrix* yaitu *matriks* yang berfungsi untuk menunjukkan jumlah prediksi benar dan prediksi salah untuk setiap kelas

```
1 # Confusion Matrix
2 conf_matrix = confusion_matrix(y_true, y_pred_classes)
3 plt.figure(figsize=(8, 6))
4 sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Greens',
5             xticklabels=class_names, yticklabels=class_names)
6 plt.xlabel('Predicted')
7 plt.ylabel('Actual')
8 plt.title('Confusion Matrix')
9 plt.show()
```

Gambar 4.1.22 Confusion matrix

Proses ini mengambil label asli dari data uji yaitu *y_true* dan prediksi model *y_pred_class* yang hasilnya adalah matriks $N \times N$ yang dimana N adalah jumlah kelas. Kemudian, untuk membuat tampilan yang lebih jelas digunakan *sns.heatmap()* dan fungsi *annot=True* untuk menampilkan angka didalam kotak heatmap dan menggunakan skema warna hijau diberikan nama kelas agar lebih informatif yaitu *xticklabels* dan *yticklabels*. Kemudian penambahan label dan judul pada sumbu X adalah kelas yang memprediksi model dan sumbu Y adalah kelas yang sebenarnya.

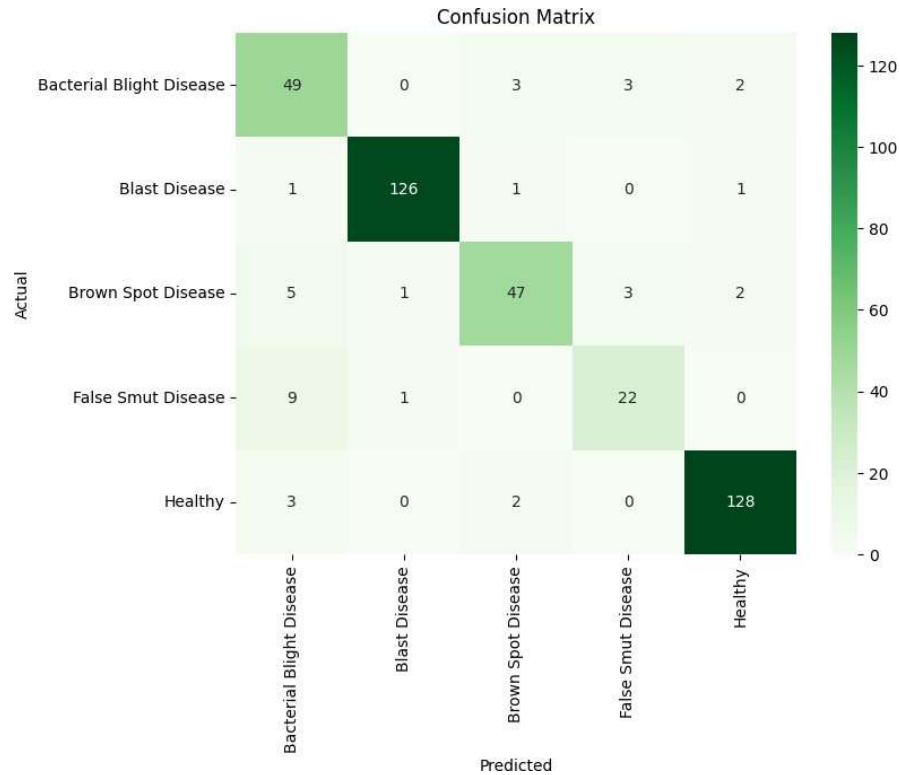


Gambar 4.1.23 Hasil confusion matrix ResNet50

Gambar 4.1.23 merupakan hasil dari *confusion matrix* dari model ResNet50 terlihat bahwa untuk *Bacterial Blight Disease*, model berhasil mengklasifikasikan dengan benar sebanyak 46 sampel, meskipun masih terjadi beberapa kesalahan dengan 3 sampel dikira *Brown Spot Disease*, 7 sampel diklasifikasikan sebagai *False Smut Disease*, dan 1 sampel dianggap sebagai *Healthy*. Sementara itu, *Blast Disease* memiliki performa paling akurat dengan 122 sampel diklasifikasikan dengan benar. Kesalahan yang terjadi relatif kecil, dengan masing-masing 3 sampel yang salah diklasifikasikan sebagai *Bacterial Blight Disease*, *False Smut Disease*, dan *Healthy*. Model menunjukkan kemampuan yang hampir sempurna dalam mengenali penyakit ini.

Pada *Brown Spot Disease*, model berhasil mengklasifikasikan 56 sampel dengan benar, hanya dengan 2 kasus yang salah dikategorikan sebagai *False Smut Disease*. Ini menunjukkan bahwa model memiliki akurasi tinggi dalam mendeteksi penyakit ini. *False Smut Disease* sendiri juga memiliki akurasi yang cukup baik dengan 30 sampel yang benar terdeteksi. Namun, masih ada 1 kasus yang salah dikategorikan sebagai *Bacterial Blight Disease* dan 1 kasus sebagai *Healthy*. Kategori *Healthy* memiliki akurasi yang hampir sempurna dengan 131 sampel diklasifikasikan dengan benar, hanya dengan 1 kesalahan yang salah dianggap sebagai *False Smut Disease*. Secara keseluruhan, model memiliki tingkat akurasi yang sangat tinggi dengan mayoritas penyakit diklasifikasikan dengan benar.

Sementara itu pada gambar 4.1.24 dibawah merupakan hasil dari *confusion matrix* dari model VGG16 yang terlihat bahwa model menunjukkan performa yang baik dalam mengenali *Blast Disease*, dengan 126 kasus yang diklasifikasikan dengan benar, hanya terdapat 1 kasus yang salah diklasifikasikan sebagai *Bacterial Blight Disease*, 1 kasus sebagai *Brown Spot Disease*, dan 1 kasus sebagai *Healthy*. Ini menunjukkan bahwa model mampu mengenali *Blast Disease* dengan tingkat keakuratan yang tinggi.



Gambar 4.1.24 Hasil confusion matrix VGG16

Pada bagian *Bacterial Blight Disease*, model berhasil mengklasifikasikan 49 kasus dengan benar, namun masih terdapat 3 kasus yang salah diklasifikasikan sebagai *Brown Spot Disease*, 3 kasus sebagai *False Smut Disease*, dan 2 kasus sebagai *Healthy*. Hal ini menunjukkan bahwa model terkadang mengalami kesulitan dalam membedakan *Bacterial Blight Disease* dengan *Brown Spot Disease* dan *False Smut Disease*. *Brown Spot Disease*, terdapat 47 kasus yang diklasifikasikan dengan benar, sementara 5 kasus salah diklasifikasikan sebagai *Bacterial Blight Disease*, 1 kasus sebagai *Blast Disease*, 3 kasus sebagai *False Smut Disease*, dan 2 kasus sebagai *Healthy*. Ini mengindikasikan bahwa *Brown*

Spot Disease memiliki beberapa kemiripan dengan kelas lainnya, terutama dengan *Bacterial Blight Disease*.

Kemudian, *False Smut Disease* diklasifikasikan dengan benar sebanyak 22 kasus, namun terdapat 9 kasus yang salah diklasifikasikan sebagai *Bacterial Blight Disease*, 1 kasus sebagai *Blast Disease*, dan 0 kasus salah sebagai *Brown Spot Disease* atau *Healthy*. Ini menunjukkan bahwa model mengalami kesulitan dalam membedakan *False Smut Disease* dari *Bacterial Blight Disease*. Terakhir, untuk tanaman sehat (*Healthy*), model berhasil mengklasifikasikan 128 kasus dengan benar, tetapi masih terdapat 3 kasus yang salah diklasifikasikan sebagai *Bacterial Blight Disease*, dan 2 kasus sebagai *Brown Spot Disease*. Ini menunjukkan bahwa model dapat mengenali tanaman sehat dengan baik, tetapi ada beberapa kasus di mana tanaman sehat dikira sebagai tanaman yang terkena penyakit tertentu.

Secara keseluruhan, model VGG16 menunjukkan performa yang cukup baik dengan mayoritas kelas memiliki tingkat klasifikasi yang akurat. Kesalahan klasifikasi yang terjadi terutama pada penyakit yang memiliki karakteristik visual yang mirip, seperti antara *Bacterial Blight Disease* dan *False Smut Disease*, serta *Brown Spot Disease* dan *Bacterial Blight Disease*.

4.1.4 Classification Report

Setelah model dilatih dan diuji untuk mengenali berbagai jenis penyakit pada tanaman padi berdasarkan citra yang diberikan Evaluasi model dilakukan dengan menggunakan *classification report* yang memberikan gambaran

mengenai tingkat akurasi, presisi, recall, serta F1-score untuk setiap kategori. Selain itu, *confusion matrix* digunakan untuk menganalisis distribusi prediksi model terhadap data sebenarnya, yang membantu dalam memahami pola kesalahan dan keakuratan klasifikasi. Berikut terlihat pada gambar 4.1.25

```
1 # Classification Report
2 class_report = classification_report(y_true, y_pred_classes, target_names=class_names)
3 print('Classification Report:')
4 print(class_report)
5
6 plt.figure(figsize=(12, 4))
7 plt.subplot(1, 2, 1)
8 plt.plot(history.history['loss'], label='Train Loss')
9 plt.plot(history.history['val_loss'], label='Validation Loss')
10 plt.legend()
11 plt.title('Loss')
12
13 plt.subplot(1, 2, 2)
14 plt.plot(history.history['accuracy'], label='Train Accuracy')
15 plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
16 plt.legend()
17 plt.title('Accuracy')
18
19 plt.show()
```

Gambar 4.1.25 Classification report

Untuk melihat bagaimana model belajar selama proses *training*, visualisasi *grafik loss* dan *accuracy* juga disertakan. Grafik ini menunjukkan perubahan nilai *loss* dan akurasi dari tahap pelatihan hingga validasi, yang dapat digunakan untuk mengevaluasi stabilitas serta generalisasi model terhadap data baru. Dengan hasil evaluasi ini, kita dapat menilai seberapa efektif model dalam melakukan prediksi dan apakah masih terdapat area yang perlu diperbaiki untuk meningkatkan performa klasifikasi. Berikut hasilnya tampak pada gambar 4.1.26

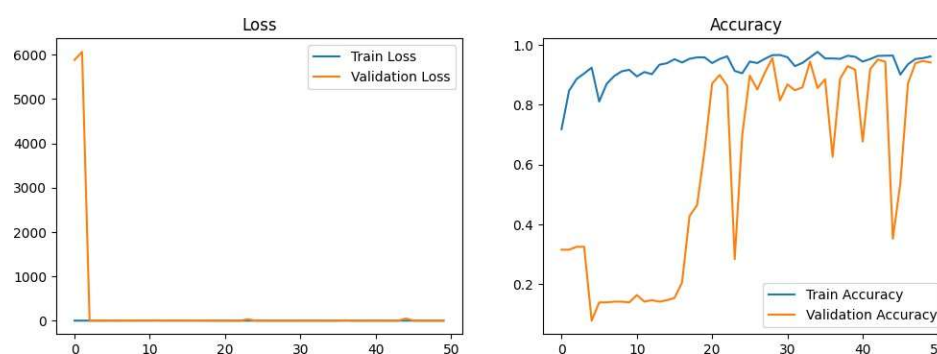
Classification Report:				
	precision	recall	f1-score	support
Bacterial Blight Disease	0.90	0.81	0.85	57
Blast Disease	1.00	0.95	0.97	129
Brown Spot Disease	0.95	0.97	0.96	58
False Smut Disease	0.70	0.94	0.80	32
Healthy	0.98	0.98	0.98	133
accuracy			0.94	409
macro avg	0.91	0.93	0.91	409
weighted avg	0.95	0.94	0.94	409

Gambar 4.1.26 Classification report ResNet50

Hasil *classification report* untuk model ResNet50 menunjukkan bahwa model memiliki akurasi tinggi, yaitu 94%, dengan performa yang sangat baik dalam mengklasifikasikan penyakit tanaman padi dan tanaman sehat. *Precision* tertinggi dicapai oleh *Blast Disease* (1.00), yang berarti setiap prediksi penyakit ini selalu benar, sedangkan *False Smut Disease* memiliki *precision* terendah (0.70), menandakan masih ada beberapa kasus yang diklasifikasikan salah. Dari sisi *recall*, *False Smut Disease* memiliki nilai tinggi (0.94), menunjukkan bahwa model jarang melewatkan kasus penyakit ini, meskipun *precision*-nya lebih rendah. *F1-score* tertinggi dimiliki oleh *Healthy* (0.98) dan *Blast Disease* (0.97), yang menandakan bahwa model sangat andal dalam mendeteksi kedua kategori tersebut.

Sementara itu, *Bacterial Blight Disease* memiliki *recall* 0.81, yang berarti masih ada kasus yang salah diklasifikasikan ke kelas lain. Dengan nilai *macro average precision* 0.91, *recall* 0.93, dan *F1-score* 0.91, serta *weighted average* yang lebih tinggi (*precision* 0.95, *recall* 0.94, *F1-score* 0.94), model ini menunjukkan performa yang sangat baik secara keseluruhan. Namun, tantangan

utama terletak pada klasifikasi *False Smut Disease*, yang masih memiliki *precision* lebih rendah dibanding kelas lain. Meskipun begitu, model ini sudah cukup andal untuk digunakan dalam deteksi penyakit tanaman padi dengan hanya sedikit kesalahan klasifikasi.



Gambar 4.1.27 Grafik ResNet50

Grafik *Loss* dan *Accuracy* menunjukkan tren pelatihan dan validasi model selama 50 *epoch*. Pada grafik *Loss*, terlihat bahwa *Train Loss* tetap rendah sepanjang proses pelatihan, sedangkan *Validation Loss* mengalami lonjakan ekstrem pada awalnya sebelum turun mendekati nol. Ini bisa mengindikasikan adanya *outlier* atau masalah dalam normalisasi data pada awal pelatihan. Sementara itu, pada grafik *Accuracy*, *Train Accuracy* meningkat dengan cepat hingga mendekati 1, menunjukkan bahwa model berhasil belajar dari data pelatihan.

Namun, *Validation Accuracy* mengalami *fluktuasi* besar, terutama pada *epoch* awal hingga pertengahan, sebelum akhirnya stabil mendekati akurasi pelatihan. *Fluktuasi* ini bisa menunjukkan adanya *overfitting* atau masalah dalam stabilitas proses pelatihan. Meskipun pada akhirnya model tampaknya mencapai akurasi yang tinggi, perbedaan tren antara pelatihan dan validasi menunjukkan

bahwa model mungkin terlalu menyesuaikan diri dengan data latih dan kurang mampu melakukan generalisasi terhadap data baru.

Demikian dengan model VGG16 setelah di evaluasi menggunakan *classification report* dan juga grafik *loss* dan *accuracy* akan di jelaskan seperti gambar berikut :

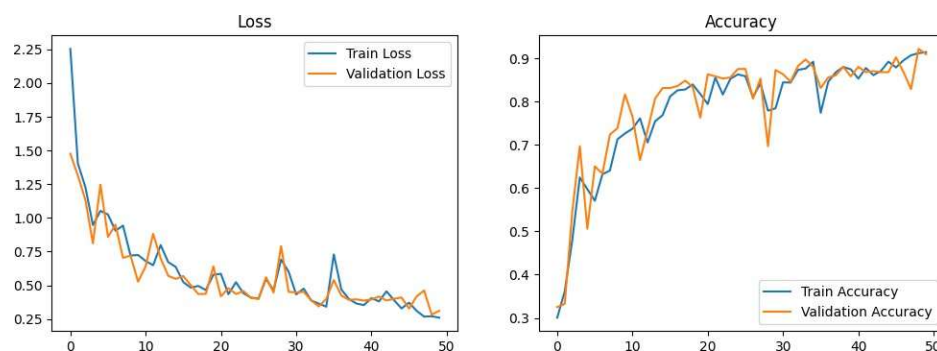
Classification Report:				
	precision	recall	f1-score	support
Bacterial Blight Disease	0.73	0.86	0.79	57
Blast Disease	0.98	0.98	0.98	129
Brown Spot Disease	0.89	0.81	0.85	58
False Smut Disease	0.79	0.69	0.73	32
Healthy	0.96	0.96	0.96	133
accuracy			0.91	409
macro avg	0.87	0.86	0.86	409
weighted avg	0.91	0.91	0.91	409

Gambar 4.1.28 Classification report VGG16

Hasil *Classification Report* dari model VGG16 menunjukkan bahwa model memiliki akurasi keseluruhan sebesar 91%, dengan nilai *precision*, *recall*, dan *f1-score* yang cukup baik di sebagian besar kelas. *Blast Disease* dan *Healthy* memiliki performa terbaik dengan *precision* dan *recall* masing-masing 0.98 dan 0.96, menunjukkan model dapat mengenali kedua kelas ini dengan sangat baik. *Bacterial Blight Disease* memiliki *recall* yang tinggi (0.86), tetapi *precision* yang lebih rendah (0.73), menandakan bahwa model terkadang salah mengklasifikasikan kategori lain sebagai penyakit ini.

Brown Spot Disease memiliki *precision* 0.89 dan *recall* 0.81, menunjukkan performa yang cukup baik meskipun ada sedikit kesalahan dalam

mendeteksi kelas ini. *False Smut Disease* memiliki performa yang paling lemah dengan *recall* 0.69 dan *f1-score* 0.73, menandakan bahwa model sering gagal mengenali penyakit ini dengan baik. Nilai *macro average* sebesar 0.87 *precision* dan 0.86 *recall* menunjukkan bahwa model cukup seimbang dalam mengenali setiap kelas, sementara *weighted average* sebesar 0.91 menunjukkan bahwa kelas dengan lebih banyak sampel mendominasi performa keseluruhan model. Secara umum, model VGG16 menunjukkan kinerja yang baik dalam mendeteksi penyakit tanaman padi, meskipun ada beberapa kelemahan terutama dalam mengenali *False Smut Disease*.



Gambar 4.1.29 Grafik VGG16

Grafik *Loss* dan *Accuracy* dari model VGG16 menunjukkan proses pelatihan yang stabil dengan peningkatan performa seiring waktu. Pada grafik *Loss*, baik *train loss* maupun *validation loss* mengalami penurunan secara konsisten, menunjukkan bahwa model semakin baik dalam meminimalkan kesalahan prediksi. Pada awal pelatihan, *loss* cukup tinggi, tetapi dengan meningkatnya jumlah *epoch*, nilai *loss* turun dan akhirnya mendekati nilai minimum yang stabil. Grafik *Accuracy* menunjukkan peningkatan akurasi yang

signifikan di awal pelatihan, dengan *train accuracy* dan *validation accuracy* yang terus meningkat hingga mencapai sekitar 90%.

Perbedaan antara *train accuracy* dan *validation accuracy* relatif kecil, menunjukkan bahwa model tidak mengalami *overfitting* yang signifikan. Meskipun terdapat sedikit fluktuasi pada *validation loss* dan *accuracy*, tren keseluruhan menunjukkan bahwa model telah berhasil belajar dengan baik dan dapat melakukan klasifikasi dengan cukup akurat pada data validasi.

4.2 Pembahasan

4.2.1 Analisis Perbandingan Model

Hasil penelitian menunjukkan bahwa model ResNet50 menunjukkan kinerja yang lebih baik daripada VGG16, dengan akurasi pengujian sebesar 94.13% dibandingkan dengan 90.95% pada VGG16. Beberapa faktor mendukung hasil ini adalah sebagai berikut:

1. Berdasarkan kedalaman arsitektur ResNet50 memiliki lebih banyak lapisan daripada VGG16, yang memungkinkan model ini mengekstrak fitur yang lebih kompleks dan mendalam.
2. Adanya *Residual Connections* yang merupakan keunggulan besar ResNet50 karena membantu mengatasi masalah *vanishing gradient*, yang memungkinkan model untuk belajar dengan lebih efisien.
3. *Generalization Power* Berdasarkan hasil *confusion matrix*, model ResNet50 memiliki kemampuan generalisasi yang lebih baik

daripada VGG16, seperti yang ditunjukkan oleh jumlah kesalahan yang lebih sedikit dalam klasifikasi kelas tertentu.

4.2.2 Analisis Kesalahan (Error Analysis)

Hasil dari *confusion matrix* yang beberapa kelas masih mengalami kesalahan klasifikasi yang cukup signifikan. Kelas *false smut disease* memiliki tingkat kesalahan tertinggi, sering kali di klasifikasikan sebagai kelas lain. Ada beberapa kemungkinan yang menjadi penyebab kesalahan ini antara lain :

1. Terdapat kemiripan visual *false smut disease* memiliki pola bercak yang mirip dengan *brown spot disease* dan *blast disease*, sehingga model mengalami kesulitan dalam membedakannya.
2. Adanya ketidakseimbangan data jika jumlah sampel untuk kelas *false smut disease* lebih sedikit dibandingkan dengan kelas lainnya, model cenderung lebih sulit untuk mengenali pola kelas tersebut.
3. Penambahan variasi dalam dataset dengan *factor* pencahayaan atau sudut pengambilan gambar yang mungkin dapat berkontribusi terhadap kesalahan klasifikasi.

4.2.3 Interpretasi Hasil

Berdasarkan hasil pengujian yang telah dilakukan, perbandingan performa arsitektur VGG16 dan ResNet50 dalam klasifikasi multi-kelas pada penyakit tanaman padi berbasis *Convolutional Neural Network* (CNN) dapat dianalisis dari beberapa aspek utama, yaitu akurasi, *loss function*, kecepatan konvergensi, dan kompleksitas model.

1. Akurasi Model

Dari hasil evaluasi, model ResNet50 menunjukkan hasil akurasi yang lebih tinggi jika dibandingkan dengan VGG16. Hal ini mengindikasikan bahwa ResNet50 memiliki kemampuan generalisasi yang lebih baik dalam mengklasifikasikan penyakit tanaman padi.

2. Loss Function

Nilai *loss function* selama proses *training* dan validasi juga dapat digunakan untuk mengukur kinerja model. Hasil pengujian model ResNet50 menunjukkan nilai *loss* yang lebih rendah jika dibandingkan dengan VGG16 yang berarti bahwa model ini mampu meminimalkan kesalahan dalam klasifikasi penyakit tanaman padi. Nilai *loss* yang lebih rendah pada model ResNet50 menandakan bahwa model ini lebih stabil dalam proses *training* dan memiliki Tingkat *overfitting* yang lebih rendah jika dibandingkan dengan VGG16.

3. Kecepatan konvergensi

Dalam proses *training*, ResNet50 membutuhkan waktu yang lebih sedikit untuk mencapai performa optimal dibandingkan dengan VGG16. Hal ini disebabkan oleh mekanisme *residual learning* yang memungkinkan model belajar lebih cepat dan lebih efisien. Sebaliknya, VGG16 memerlukan lebih banyak *epoch* untuk mencapai tingkat akurasi yang optimal, yang menunjukkan bahwa model ini lebih lambat dalam memahami pola data dibandingkan dengan ResNet50.

4. Kompleksitas Model

Jika dilihat dari segi kompleksitas, VGG16 memiliki arsitektur yang lebih sederhana dengan layer konvolusi yang berulang, sedangkan ResNet50 yang memiliki struktur lebih dalam dengan adanya *residual blocks*. Walaupun ResNet50 lebih kompleks, penggunaan *residual connections* memungkinkan model ini lebih efisien dalam menangani jaringan yang dalam, sedangkan VGG16 rentan terhadap permasalahan degradasi performa pada jaringan yang mendalam.

Berdasarkan hasil evaluasi yang telah dilakukan, dapat disimpulkan bahwa arsitektur ResNet50 memiliki keunggulan dibandingkan dengan VGG16 dalam hal akurasi, *loss function*, waktu yang dibutuhkan model untuk belajar, dan efisiensi model. Dengan demikian, dalam kasus klasifikasi multi-kelas penyakit tanaman padi, arsitektur ResNet50 lebih direkomendasikan dibandingkan dengan VGG16 karena mampu memberikan performa yang lebih baik dan lebih stabil dalam proses klasifikasi.

BAB V

KESIMPULAN

5.1 Kesimpulan

Dari penelitian yang telah dilakukan dapat diambil kesimpulan yaitu :

1. Berdasarkan kinerja dari kedua arsitektur yang diukur melalui nilai *accuracy*, *precision*, *recall*, dan *f1-score* maka didapatkan bahwa model arsitektur ResNet50 lebih unggul jika dibandingkan dengan dengan model arsitektur VGG16. Model arsitektur ResNet50 mendapatkan nilai *accuracy* sebesar 94,13% dengan nilai *precision*, *recall*, dan *f1-score* masing – masing 0.95, 0.94 dan 0.94. sedangkan model arsitektur VGG16 mendapatkan nilai *accuracy* sebesar 91,95% , dengan nilai *precision*, *recall*, dan *f1-score* masing – masing 0.91. Keduanya efektif dalam mengidentifikasi berbagai penyakit tanaman padi termasuk *Bacterial Blight*, *Blast Disease*, *Brown Spot*, *False Smut*, dan tanaman sehat.
2. Berdasarkan metrik evaluasi, model ResNet50 terbukti lebih efektif dengan akurasi 94,13% dibandingkan dengan model VGG16 dengan akurasi sebesar 91,95%. Model ResNet50 juga menunjukkan performa yang sangat baik dalam hal *precision* dan *recall* untuk hamper semua kategori penyakit, terutama untuk *Blast disease* dengan *f1-score* 0,97 dan tanaman sehat dengan *f1-score* 0,98.
3. Hasil perbandingan melalui metrik evaluasi yaitu *accuracy*, *precision*, *recall*, dan *f1-score* pada model ResNet50 di dapatkan *accuracy* sebesar 94% *precision weighted average* sebesar 95%, *recall*, dan *f1-score* masing-masing memperoleh *weighted average* sebesar 94%, dengan waktu

pemrosesan membutuhkan waktu 38ms/step untuk evaluasi dan 7 – 8 detik untuk waktu rata – rata training per *epoch* menunjukkan kemampuan inferensi yang cepat Dimana ini penting untuk aplikasi praktis di lapangan. Berikutnya pada model VGG16 mendapatkan *accuracy* sebesar 91% begitu pula dengan *precision*, *recall*, dan *f1-score* mendapatkan *weighted average* sebesar 91% dengan waktu pemrosesan membutuhkan waktu 56ms/step untuk evaluasi dan 13 – 14 detik untuk waktu rata – rata training per *epoch*.

4. Hasil dari penelitian ini telah berhasil mengembangkan model deep learning yang efisien untuk mengelola penyakit pada tanaman padi. Model terbaik yaitu model ResNet50 mencapai akurasi 94,13% dengan *loss* hanya 0.2125 dan waktu pemrosesan 38ms/step, menjadikan Solusi yang sangat layak untuk implementasi praktis.

5.2 Saran

1. Adapun saran yang diberikan kepada penelitian lebih lanjut yaitu diharapkan kedepannya mengembangkan model yang dapat mengidentifikasi tahap perkembangan penyakit yang kemungkinan *inferensi* yang lebih tepat waktu.
2. Mampu meningkatkan kemampuan model untuk mendeteksi infeksi ganda pada tanaman yang sama.
3. Memperbanyak *dataset* tambahan untuk penyakit dengan jumlah sampel rendah seperti *False Smut Disease* dan menambahkan varietas penyakit padi lainnya.

DAFTAR PUSTAKA

- Adiningsi, S., & Saputra, R. A. (n.d.). *JIP (Jurnal Informatika Polinema) IDENTIFIKASI JENIS DAUN TANAMAN OBAT MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN) DENGAN MODEL VGG16*.
- Basha, N. K., Ananth, C., Muthukumaran, K., Sudhamsu, G., Mittal, V., & Gared, F. (2024). Mask region-based convolutional neural network and VGG-16 inspired brain tumor segmentation. *Scientific Reports*, 14(1). <https://doi.org/10.1038/s41598-024-66554-4>
- Bella Dwi Mardiana, Wahyu Budi Utomo, Ulfah Nur Oktaviana, Galih Wasis Wicaksono, & Agus Eko Minarno. (2023). Herbal Leaves Classification Based on Leaf Image Using CNN Architecture Model VGG16. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 7(1), 20–26. <https://doi.org/10.29207/resti.v7i1.4550>
- Cambay, V. Y., Barua, P. D., Hafeez Baig, A., Dogan, S., Baygin, M., Tuncer, T., & Acharya, U. R. (2024). Automated Detection of Gastrointestinal Diseases Using Resnet50*-Based Explainable Deep Feature Engineering Model with Endoscopy Images. *Sensors*, 24(23). <https://doi.org/10.3390/s24237710>
- Chen, Y., Wang, L., Ding, B., Shi, J., Wen, T., Huang, J., & Ye, Y. (2024). Automated Alzheimer's disease classification using deep learning models with Soft-NMS and improved ResNet50 integration. *Journal of Radiation Research and Applied Sciences*, 17(1), 100782. <https://doi.org/10.1016/j.jrras.2023.100782>
- Deng, C., Han, D., Feng, M., Lv, Z., & Li, D. (2022). Differential diagnostic value of the ResNet50, random forest, and DS ensemble models for papillary thyroid carcinoma and other thyroid nodules. *Journal of International Medical Research*, 50(4). <https://doi.org/10.1177/03000605221094276>
- Dhankhar, P. (n.d.). ResNet-50 and VGG-16 for recognizing Facial Emotions. *International Journal of Innovations in Engineering and Technology*. <https://doi.org/10.21172/ijiet.134.18>
- Elpeltagy, M., & Sallam, H. (2021a). Automatic prediction of COVID– 19 from chest images using modified ResNet50. *Multimedia Tools and Applications*, 80(17), 26451–26463. <https://doi.org/10.1007/s11042-021-10783-6>

- Elpeltagy, M., & Sallam, H. (2021b). Automatic prediction of COVID-19 from chest images using modified ResNet50. *Multimedia Tools and Applications*, 80(17), 26451–26463. <https://doi.org/10.1007/s11042-021-10783-6>
- Habib Hawari, F., Fadillah, F., Rifqi Alviandi, M., & Arifin, T. (2022). KLASIFIKASI PENYAKIT PADI MENGGUNAKAN ALGORITMA CNN (CONVOLUTIONAL NEURAL NETWORK). *JURNAL RESPONSIF*, 4(2), 184–189. <https://ejurnal.ars.ac.id/index.php/jti>
- Handayani, V. W., Yudianto, A., Mieke Sylvia, M. A. R., Rulaningtyas, R., & Caesarardhi, M. R. S. (2024). Classification of Indonesian adult forensic gender using cephalometric radiography with VGG16 and VGG19: a Preliminary research. *Acta Odontologica Scandinavica*, 83, 308–316. <https://doi.org/10.2340/aos.v83.40476>
- Kualitas Buah Salak dengan Transfer Learning Arsitektur VGG, K., & Luthfiarta, A. (2021). VGG16 Transfer Learning Architecture for Salak Fruit Quality Classification. *Jurnal Informatika Dan Teknologi Informasi*, 18(1), 37–48. <https://doi.org/10.31515/telematika.v18i1.4025>
- Li, Y., Gao, P., Luo, Y., Luo, X., Xu, C., Chen, J., Zhang, Y., Lin, G., & Xu, W. (2024). Automatic Detection and Classification of Natural Weld Defects Using Alternating Magneto-Optical Imaging and ResNet50. *Sensors*, 24(23). <https://doi.org/10.3390/s24237649>
- Mannepal, P. K., Pathre, A., Chhabra, G., Ujjainkar, P. A., & Wanjari, S. (2024). Diagnosis of bacterial leaf blight, leaf smut, and brown spot in rice leaves using VGG16. *Procedia Computer Science*, 235, 193–200. <https://doi.org/10.1016/j.procs.2024.04.022>
- Memon, M. S., Qabulio, M., Kumar, P., Soomro, A. K., & Memon, S. (2024a). Identification of Leaf Diseases of Different Crops using modified ResNet50. *The Asian Bulletin of Big Data Management*, 4(02). <https://doi.org/10.62019/abbdm.v4i02.166>
- Memon, M. S., Qabulio, M., Kumar, P., Soomro, A. K., & Memon, S. (2024b). Identification of Leaf Diseases of Different Crops using modified ResNet50. *The Asian Bulletin of Big Data Management*, 4(02). <https://doi.org/10.62019/abbdm.v4i02.166>

- Mudzakir, I., & Arifin, T. (2022). Klasifikasi Penggunaan Masker dengan Convolutional Neural Network Menggunakan Arsitektur MobileNetv2. *EXPERT: Jurnal Manajemen Sistem Informasi Dan Teknologi*, 12(1), 76. <https://doi.org/10.36448/expert.v12i1.2466>
- Murimi, M. M. (n.d.). *IDENTIFICATION OF MAIZE LEAF DISEASES USING SUPPORT VECTOR MACHINE AND CONVOLUTIONAL NEURAL NETWORKS ALEXNET AND RESNET50*.
- Pamungkas, W. G., Wardhana, M. I. P., Sari, Z., & Azhar, Y. (2023). Leaf Image Identification: CNN with EfficientNet-B0 and ResNet-50 Used to Classified Corn Disease. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 7(2), 326–333. <https://doi.org/10.29207/resti.v7i2.4736>
- Rizki, A. M., & Marina, N. (2019). KLASIFIKASI KERUSAKAN BANGUNAN SEKOLAH MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK DENGAN PRE-TRAINED MODEL VGG-16. *Jurnal Ilmiah Teknologi Dan Rekayasa*, 24(3), 197–206. <https://doi.org/10.35760/tr.2019.v24i3.2396>
- Sarker, I. H. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. In *SN Computer Science* (Vol. 2, Issue 6). Springer. <https://doi.org/10.1007/s42979-021-00815-1>
- Swain, S., & Tripathy, A. K. (2024). Automatic detection of potholes using VGG-16 pre-trained network and Convolutional Neural Network. *Heliyon*, 10(10). <https://doi.org/10.1016/j.heliyon.2024.e30957>
- Tamara, W., Khair, H., Informasi, S., & Kaputama Binjai, S. (2024). Jurnal JISIILKOM (Jurnal Inovasi Sistem Informasi & Ilmu Komputer) Penerapan Metode Case Based Reasoning Mendiagnosa Penyakit Tanaman Padi. In *Jurnal JISIILKOM* (Vol. 2, Issue 2). Online.
- Udayananda, G. K. V. L., Shyalika, C., & Kumara, P. P. N. V. (2022). Rice plant disease diagnosing using machine learning techniques: a comprehensive review. In *SN Applied Sciences* (Vol. 4, Issue 11). Springer Nature. <https://doi.org/10.1007/s42452-022-05194-7>

- Wan, J., Li, B., Wang, K., Teng, X., Wang, T., & Mao, B. (2024). An Improved ResNet50 for Environment Image Classification. *Procedia Computer Science*, 242, 1000–1007. <https://doi.org/10.1016/j.procs.2024.08.246>
- Weny Indah Kusumawati, & Adisaputra Zidha Noorizki. (2023). Perbandingan Performa Algoritma VGG16 Dan VGG19 Melalui Metode CNN Untuk Klasifikasi Varietas Beras. *Journal of Computer, Electronic, and Telecommunication*, 4(2). <https://doi.org/10.52435/complete.v4i2.387>
- Windiawan, R., Suharso, A., & Artikel, S. (n.d.). *Identifikasi Penyakit pada Daun Kopi Menggunakan Metode Deep Learning VGG16 INFO ARTIKEL ABSTRAK*. <https://doi.org/10.35891/explorit>
- Xu, H., Yu, Y., Chang, J., Hu, X., Tian, Z., & Li, O. (2024). Precision lung cancer screening from CT scans using a VGG16-based convolutional neural network. *Frontiers in Oncology*, 14. <https://doi.org/10.3389/fonc.2024.1424546>

LAMPIRAN

Source Code

ResNet50

```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D
from keras.applications import ResNet50
from keras.callbacks import EarlyStopping, ReduceLROnPlateau

#import path gambar
image_path = r"C:\Users\Lenovo\Downloads\Rice_Diseases\Rice_Diseases"
categories = ["Bacterial Blight Disease", "Blast Disease", "Brown Spot Disease", "False Smut Disease", "Healthy"]

def load_images(image_path, categories):
    img_size = (128, 128)
    images = []
    labels = []

    #iterasi kategori dan pengecekan data
    for category in categories:
        category_path = os.path.join(image_path, category)

        # mengecek kategori apakah ada atau tidak
        if not os.path.exists(category_path):
            print(f"Path not found: {category_path}")
            continue

        #membaca dan memproses gambar
        for filename in os.listdir(category_path):
            if filename.endswith(".jpg") or filename.endswith(".jpeg") or filename.endswith(".png"):
                img_path = os.path.join(category_path, filename)
                img = cv2.imread(img_path)
                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                img = cv2.resize(img, img_size)
                images.append(img)
                labels.append(category)

    #konversi dari numpy ke Array
    return np.array(images), np.array(labels)
```

```

#menampilkan Jumlah data
images, labels = load_images(image_path, categories)
print(len(images))
print(len(labels))

#menampilkan 9 gambar acak dari datasets
def display_sample_images(images, labels, class_names):
    plt.figure(figsize=(10, 10))
    for i in range(9):
        idx = np.random.randint(len(images))
        plt.subplot(3, 3, i + 1)
        plt.imshow(images[idx])
        plt.title("Pred: " + class_names[labels[idx]])
        plt.axis('off')
    plt.show()

#Label encoding dan One-hot encoding
le = LabelEncoder()
labels_encoded = le.fit_transform(labels)
labels_encoded = to_categorical(labels_encoded)

#Normalisasi Gambar
images = images / 255.0

#Menampilkan gambar dan mendapatkan nama kelas
class_names = le.classes_
display_sample_images(images, labels_encoded.argmax(axis=1), class_names)

X_train, X_test, y_train, y_test = train_test_split(
    images, labels_encoded, test_size=0.2, random_state=42)

# Data Augmentation
datagen = ImageDataGenerator(
    rotation_range=40, width_shift_range=0.2,
    height_shift_range=0.2, shear_range=0.2, zoom_range=0.2,
    horizontal_flip=True, fill_mode='nearest'
)

datagen.fit(X_train)

resnet_base = ResNet50(weights='imagenet', include_top=False,
input_shape=(128, 128, 3))

model = Sequential()
model.add(resnet_base)
model.add(GlobalAveragePooling2D())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(len(categories), activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

```

```

# Model Training with callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
                                restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5,
                                min_lr=0.00001)

history = model.fit(datagen.flow(X_train, y_train, batch_size=32),
                    validation_data=(X_test, y_test),
                    epochs=50)
#                    callbacks=[early_stopping, reduce_lr])

# Evaluate Model
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {test_accuracy * 100:.2f}%')

# Generate predictions
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

# Confusion Matrix
conf_matrix = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Greens',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Classification Report
class_report = classification_report(y_true, y_pred_classes,
                                     target_names=class_names)
print('Classification Report:')
print(class_report)

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Loss')

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Accuracy')

plt.show()

```

VGG16

```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D
from keras.applications import ResNet50
from keras.callbacks import EarlyStopping, ReduceLROnPlateau

#import path gambar
image_path = r"C:\Users\Lenovo\Downloads\Rice_Diseases\Rice_Diseases"
categories = ["Bacterial Blight Disease", "Blast Disease", "Brown Spot Disease", "False Smut Disease", "Healthy"]

def load_images(image_path, categories):
    img_size = (128, 128)
    images = []
    labels = []

    #iterasi kategori dan pengecekan data
    for category in categories:
        category_path = os.path.join(image_path, category)

        # mengecek kategori apakah ada atau tidak
        if not os.path.exists(category_path):
            print(f"Path not found: {category_path}")
            continue

        #membaca dan memproses gambar
        for filename in os.listdir(category_path):
            if filename.endswith(".jpg") or filename.endswith(".jpeg") or filename.endswith(".png"):
                img_path = os.path.join(category_path, filename)
                img = cv2.imread(img_path)
                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                img = cv2.resize(img, img_size)
                images.append(img)
                labels.append(category)

    #konversi dari numpy ke Array
    return np.array(images), np.array(labels)

#menampilkan Jumlah data
images, labels = load_images(image_path, categories)
print(len(images))
print(len(labels))
```

```

#menampilkan 9 gambar acak dari datasets
def display_sample_images(images, labels, class_names):
    plt.figure(figsize=(10, 10))
    for i in range(9):
        idx = np.random.randint(len(images))
        plt.subplot(3, 3, i + 1)
        plt.imshow(images[idx])
        plt.title("Pred: " + class_names[labels[idx]])
        plt.axis('off')
    plt.show()

#Label encoding dan One-hot encoding
le = LabelEncoder()
labels_encoded = le.fit_transform(labels)
labels_encoded = to_categorical(labels_encoded)

#Normalisasi Gambar
images = images / 255.0

#Menampilkan gambar dan mendapatkan nama kelas
class_names = le.classes_
display_sample_images(images, labels_encoded.argmax(axis=1), class_names)

X_train, X_test, y_train, y_test = train_test_split(
    images, labels_encoded, test_size=0.2, random_state=42)

# Data Augmentation
datagen = ImageDataGenerator(
    rotation_range=40,width_shift_range=0.2,
    height_shift_range=0.2, shear_range=0.2, zoom_range=0.2,
    horizontal_flip=True, fill_mode='nearest'
)

datagen.fit(X_train)

vgg_base = VGG16(weights='imagenet', include_top=False, input_shape=(128,
128, 3))
model = Sequential()
model.add(vgg_base)
model.add(GlobalAveragePooling2D())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(categories), activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Model Training with callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5,
min_lr=0.00001)

history = model.fit(datagen.flow(X_train, y_train, batch_size=32),
                    validation_data=(X_test, y_test),
                    epochs=50)
#                    callbacks=[early_stopping, reduce_lr])

```

```

# Evaluate Model
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {test_accuracy * 100:.2f}%')

# Generate predictions
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

# Confusion Matrix
conf_matrix = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Greens',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Classification Report
class_report = classification_report(y_true, y_pred_classes,
                                     target_names=class_names)
print('Classification Report:')
print(class_report)

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Loss')

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Accuracy')

plt.show()

```

SK Pembimbing Skripsi



UMSU
Unggul | Cerdas | Terpercaya
Diklaimasi sebagai universitas nomor satu di Sumatera Utara

MAJELIS PENDIDIKAN TINGGI PENELITIAN & PENGEMBANGAN PIMPINAN PUSAT MUHAMMADIYAH

UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UMSU Terakreditasi A Berdasarkan Keputusan Badan Akreditasi Nasional Perguruan Tinggi No. 89/SK/BAN-PT/Akred/PT/III/2019
Pusat Administrasi: Jalan Mukhtar Basri No. 3 Medan 20238 Telp. (061) 6622400 - 66224567 Fax. (061) 6625474 - 6631003
<https://fkit.umsu.ac.id> fkit@umsu.ac.id [fumsu](#) [umsu](#) [umsu](#) [umsu](#)

**PENETAPAN DOSEN PEMBIMBING
PROPOSAL/SKRIPSI MAHASISWA
NOMOR : 939/IL3-AU/UMSU-09/F/2024**

Assalamu'alaikum Warahmatullahi Wabarakatuh

Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Muhammadiyah Sumatera Utara, berdasarkan Persetujuan permohonan judul penelitian Proposal / Skripsi dari Ketua / Sekretaris.

Program Studi : Teknologi Informasi
Pada tanggal : 04 Desember 2024

Dengan ini menetapkan Dosen Pembimbing Proposal / Skripsi Mahasiswa.

Nama : Krisna Aditya
NPM : 2109020048
Semester : VII (Tujuh)
Program studi : Teknologi Informasi
Judul Proposal / Skripsi : Analisis Perbandingan Performa Arsitektur VGG16 Dan RESNET50 Dalam Klasifikasi Multi-Kelas Pada Penyakit Tanaman Padi Berbasis Convolutional Neural Network (CNN)

Dosen Pembimbing : Mhd. Basri, S.Si.,M.Kom.

Dengan demikian di izinkan menulis Proposal / Skripsi dengan ketentuan

1. Penulisan berpedoman pada buku panduan penulisan Proposal / Skripsi Fakultas Ilmu Komputer dan Teknologi Informasi UMSU
2. Pelaksanaan Sidang Skripsi harus berjarak 3 bulan setelah dikeluarkannya Surat Penetapan Dosen Pembimbing Skripsi.
3. **Proyek Proposal / Skripsi dinyatakan " BATAL " bila tidak selesai sebelum Masa Kadaluaarsa tanggal : 04 Desember 2025**
4. Revisi judul.....

Wassalamu'alaikum Warahmatullahi Wabarakatuh.

Ditetapkan di : Medan
Pada Tanggal : 03 Jumadil Akhir 1446 H
04 Desember 2024 M

Dekan



Dr. M. Khawarizmi, S.Kom., M.Kom
NIDN : 0127099201



Cc. File



112

Bimbingan Proposal



UMSU
Unggul | Cerdas | Terpercaya

Bila mahasiswa salah, ia juga disalahkan
Nahar dan bimbingan

MAJELIS PENDIDIKAN TINGGI PENELITIAN & PENGEMBANGAN PIMPINAN PUSAT MUHAMMADIYAH

UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UMSU Terakreditasi A Berdasarkan Keputusan Badan Akreditasi Nasional Perguruan Tinggi No. 89/SK/BAN-PT/Akred/PT/III/2019

Pusat Administrasi: Jalan Mukhtar Basri No. 3 Medan 20238 Telp. (061) 6622400 - 66224567 Fax. (061) 6625474 - 6631003

https://faksi.umsu.ac.id faksi@umsu.ac.id umsumedan umsumedan umsumedan umsumedan

Berita Acara Pembimbingan Proposal

Nama Mahasiswa : Krisna Aditya

NPM : 2109020048

Nama Dosen Pembimbing : Mhd. Basri, S.Si., M.Kom.

Program Studi : Teknologi Informasi

Konsentrasi : Deep Learning

Judul Penelitian : Analisis Perbandingan Performa Arsitektur Vgg16 Dan Resnet50 Dalam Klasifikasi Multi-Kelas Pada Penyakit Tanaman Padi Berbasis Convolutional Neural Network (CNN)

Tanggal Bimbingan	Hasil Evaluasi	Paraf Dosen
20/2024/12	Revisi Bab I, Revisi: Latar belakang, tujuan dan manfaat	
02/2025/01	Revisi Bab II, Peningkat kajian pustaka, bentuk tabel Peneliti terdahulu	
03/2025/01	Revisi bab II, Jelaskan prosedur penelitian	
30/2025/01	Revisi Bab III, Ganti dan typo, dan Daftar Pustaka	
11/2025/02	Acc Proposal	

Diketahui oleh :

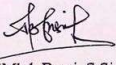
Ketua Program Studi
Teknologi Informasi

(Fatma Sari Hutagalung, S.Kom., M.Kom)

Medan,

Disetujui oleh :

Dosen Pembimbing



(Mhd. Basri, S.Si., M.Kom)





Bimbingan Skripsi



MAJELIS PENDIDIKAN TINGGI PENELITIAN & PENGEMBANGAN PIMPINAN PUSAT MUHAMMADIYAH

UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UMSU Terakreditasi A Berdasarkan Keputusan Badan Akreditasi Nasional Perguruan Tinggi No. 89/SK/BAN-PT/Akred/PT/IIK/2019
 Pusat Administrasi: Jalan Mukhtar Basri No. 3 Medan 20238 Telp. (061) 6622400 - 66224567 Fax. (061) 6625474 - 6631003
<https://fkip.umhu.ac.id> Email: fkip@umhu.ac.id [umsumedan](https://www.facebook.com/umsumedan) [umsumedan](https://www.instagram.com/umsumedan) [umsumedan](https://www.youtube.com/umsumedan) [umsumedan](https://www.tiktok.com/umsumedan)

Berita Acara Pembimbingan Skripsi

Nama Mahasiswa : Krisna Aditya Program Studi : Teknologi Informasi
 NPM : 2109020048 Konsentrasi : Deep Learning
 Nama Dosen Pembimbing : Mhd. Basri, S.Si., M.Kom. Judul Penelitian : Analisis Perbandingan Performa Arsitektur Vgg16 Dan Resnet50 Dalam Klasifikasi Multi-Kelas Pada Penyakit Tanaman Padi Berbasis Convolutional Neural Network (CNN)

Tanggal Bimbingan	Hasil Evaluasi	Paraf Dosen
17/3 2025	Perbaikan Bab I sesuai pengujian	<i>[Signature]</i>
17/3 2025	Perbaikan Bab II sesuai pengujian	<i>[Signature]</i>
04/4 2025	Perbaikan Bab III sesuai pengujian	<i>[Signature]</i>
24/4 2025	Kesimpulan dan saran sempurna	<i>[Signature]</i>
05/5 2025	Acc Sidang	<i>[Signature]</i>

Diketahui oleh :

Ketua Program Studi
Teknologi Informasi

[Signature]
(Fatma Sari Hutagalung, S.Kom., M.Kom.)

Medan, 24 Mei 2025

Disetujui oleh :

Dosen Pembimbing

[Signature]
(Mhd. Basri, S.Si., M.Kom.)





LETTER OF ACCEPTANCE (LoA)

Dear Mr/Ms/Br

Krisna Aditya, Mhd Basri

In

Place

We hereby inform you that the manuscript with the following details has been declared accepted for publication in the Tsabit Journal of Computer Science, ISSN: 3062-8504 (Online), Volume 1 Number 2 December 2025 Edition.

Title	Comparative Analysis Of The Performance Of VGG16 And ResNet50 Architectures In Multi-Class Classification Of Rice Plant Diseases Based On Convolutional Neural Networks (CNN)
Author	Krisna Aditya, Mhd Basri
Correspondent Email	mhd.basri@umsu.ac.id

Thus we have created this certificate so that it can be used as necessary.



Medan, 09 September 2025

Editor in Chief

Oris Krianto Sulaiman, S.T., M.Kom.



ORIGINALITY REPORT

19%	15%	9%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.umsu.ac.id Internet Source	1%
2	elibrary.unikom.ac.id Internet Source	1%
3	kupangkota.bps.go.id Internet Source	1%
4	dicoding.com Internet Source	1%
5	jip.polinema.ac.id Internet Source	<1%
6	ejurnal.seminar-id.com Internet Source	<1%
7	Submitted to Universitas Muhammadiyah Sumatera Utara Student Paper	<1%
8	repository.adzkia.ac.id Internet Source	<1%
9	123dok.com Internet Source	<1%
10	docplayer.info Internet Source	<1%
11	doaj.org Internet Source	<1%

Submitted to Fakultas Teknik

12

<1 %

13

Praveen Kumar Mannepalli, Ayonija Pathre, Gunjan Chhabra, Priyanka Anup Ujjainkar, Shrutika Wanjari. "Diagnosis of bacterial leaf blight, leaf smut, and brown spot in rice leafs using VGG16", Procedia Computer Science, 2024

Publication

<1 %

14

journal.ittelkom-sby.ac.id

Internet Source

<1 %

15

Indah Clara Sari. "INTEGRASI MODEL DEEP LEARNING EFFICIENTNET-B0 UNTUK DETEKSI PENYAKIT DAUN TOMAT PADA APLIKASI SELULER BERBASIS FLUTTER", Djtechno: Jurnal Teknologi Informasi, 2024

Publication

<1 %

16

Submitted to LPPM

Student Paper

<1 %

17

Yusi Chen, Lizhen Wang, Bijiao Ding, Jianshe Shi, Tingxi Wen, Jianlong Huang, Yuguang Ye. "Automated Alzheimer's disease classification using deep learning models with Soft-NMS and improved ResNet50 integration", Journal of Radiation Research and Applied Sciences, 2024

Publication

<1 %

18

ijcs.stmikindonesia.ac.id

Internet Source

<1 %

19

Yanfeng Li, Pengyu Gao, Yongbiao Luo, Xianghan Luo, Chunmei Xu, Jiecheng Chen, Yanxi Zhang, Genxiang Lin, Wei Xu.

<1 %

"Automatic Detection and Classification of Natural Weld Defects Using Alternating Magneto-Optical Imaging and ResNet50",
Sensors, 2024

Publication

20	medicaljournalssweden.se Internet Source	<1 %
21	Muhammad Ulfi, Muhammad Ibadurrahman Arrasyid Supriyanto, Chalsi Mala Sari, Setio Ardy Nuswantoro. "Identifikasi Penyakit pada Tanaman Padi Menggunakan Convolutional Neural Network (CNN)", Jurnal Sains Komputer dan Teknologi Informasi, 2025 Publication	<1 %
22	ejournal.stmikgici.ac.id Internet Source	<1 %
23	www.coursehero.com Internet Source	<1 %
24	Submitted to Telkom University Student Paper	<1 %
25	Submitted to Universitas Gunadarma Student Paper	<1 %
26	digilib.uns.ac.id Internet Source	<1 %
27	teknologipintar.org Internet Source	<1 %
28	www.frontiersin.org Internet Source	<1 %
29	www.scilit.net Internet Source	<1 %

30	Rayhan Dzikri Fauzi, Muhammad Fachrie. "SISTEM PREDIKSI TINGKAT KEMATANGAN BUAH TOMAT MENGGUAKAN ARSITEKTUR VGG16", Djtechno: Jurnal Teknologi Informasi, 2024 Publication	<1%
31	Submitted to STT PLN Student Paper	<1%
32	eprints.walisongo.ac.id Internet Source	<1%
33	jurnal.iaii.or.id Internet Source	<1%
34	www.researchgate.net Internet Source	<1%
35	Anda Putra. "Klasifikasi Penyakit Daun Cabai Dengan Metode CNN Untuk Deteksi Awal.", Jurnal Profesi Insinyur Universitas Lampung, 2025 Publication	<1%
36	Chengwen Deng, Dongyan Han, Ming Feng, Zhongwei Lv, Dan Li. "Differential diagnostic value of the ResNet50, random forest, and DS ensemble models for papillary thyroid carcinoma and other thyroid nodules", Journal of International Medical Research, 2022 Publication	<1%
37	jutif.if.unsoed.ac.id Internet Source	<1%
38	pemrogramanmatlab.com Internet Source	<1%

39

repository.its.ac.id

Internet Source

<1 %

40

www.nature.com

Internet Source

<1 %

41

jurnal.unw.ac.id

Internet Source

<1 %

42

Arimbi Puspitasari, Diana Sava Salsabila, Dwi Roliawati. "Penerapan ResNet-50 CNN untuk Optimalisasi Klasifikasi pada Data Fashion", *INDONESIAN JOURNAL ON DATA SCIENCE*, 2025

Publication

<1 %

43

Submitted to Binus University International

Student Paper

<1 %

Ryan Ali Mas'ud, Junta Zeniarja. "Optimasi Convolutional Neural Networks untuk Deteksi Kanker Payudara menggunakan Arsitektur DenseNet", *Edumatic: Jurnal Pendidikan Informatika*, 2024

Publication

<1 %

Erdina Turnip, Anief Fauzan Rozi. "Analisis Perbandingan Arsitektur Convolutional Neural Network pada Klasifikasi Jenis Penyakit Daun Padi", *ProTekInfo(Pengembangan Riset dan Observasi Teknik Informatika)*, 2024

Publication

<1 %

Ines Heidiani Ikasari, Perani Rosyani, Resti Amalia. "Klasifikasi Jenis Buah Menggunakan Metode CNN", *RIGGS: Journal of Artificial Intelligence and Digital Business*, 2025

Publication

<1 %

Submitted to Universitas Jember

Jiacheng Wan, Bingchan Li, Kun Wang, Xixi Teng, Tao Wang, Bo Mao. "An Improved ResNet50 for Environment Image Classification", Procedia Computer Science, 2024

Publication

<1 %

Submitted to The Islamia University Bahawalpur

Student Paper

<1 %

Dhyanna Lisa Rahmadona Putri, Supatman . "KLASIFIKASI CITRA JENIS HIJAB

MENGGUNAKAN DENSENET-121", Jurnal Informatika dan Teknik Elektro Terapan, 2025

Publication

<1 %

51

kc.umh.ac.id

Internet Source

<1 %

52

machinelearning.mipa.ugm.ac.id

Internet Source

<1 %

53

ojs.uma.ac.id

Internet Source

<1 %

54

repository.ittelkom-pwt.ac.id

Internet Source

<1 %

55

e-journal.uajy.ac.id

Internet Source

<1 %

56

santika.upnjatim.ac.id

Internet Source

<1 %

57

Submitted to University of Muhammadiyah Malang

Student Paper

<1 %

58

ejournal.unisbablitar.ac.id

Internet Source

<1 %

59

journal.unublitar.ac.id

Internet Source

<1 %

60

repository.nurulfikri.ac.id

Internet Source

<1 %

61

repository.uin-suska.ac.id

Internet Source

<1 %

Muhammad Khatama Insani, Dwi Budi Santoso. "Perbandingan Kinerja Model Pre-Trained CNN (VGG16, RESNET, dan INCEPTIONV3) untuk Aplikasi Pengenalan Wajah pada Sistem Absensi Karyawan", Jurnal Indonesia : Manajemen Informatika dan Komunikasi, 2024

Publication

<1 %

Muhammad Kholilurrahman, Wahyul Amien Syafei, Oky Dwi Nurhayati. "Klasifikasi Image Processing Pada Citra Warna Daun Padi Menggunakan Metode Convolutional Neural Network", Jurnal Ilmiah Sains, 2023

Publication

<1 %

SATRIA SATRIA, Sumijan, Billy Hendrik. "Implementasi Convolutional Neural Network Untuk Klasifikasi Citra KTP-EI", Jurnal CoSciTech (Computer Science and Information Technology), 2024

Publication

<1 %

65

Submitted to Syiah Kuala University

Student Paper

<1 %

66

Submitted to Universitas Muslim Indonesia

Student Paper

<1 %

67

jurnal.stkipppgritulungagung.ac.id

Internet Source

<1%

68

Galih Hermawan. "Klasifikasi Pengemudi Terganggu Berdasarkan Citra Menggunakan ResNet-50", Jurnal Informatika Komputer, Bisnis dan Manajemen, 2024

Publication

<1%

69

Tommy Sarawan, Kusrini Kusrini. "KLASIFIKASI PENGENALAN WAJAH SISWA PADA SISTEM KEHADIRAN DENGAN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN)", Djtechno: Jurnal Teknologi Informasi, 2025

Publication

<1%

70

text-id.123dok.com

Internet Source

<1%

71

Submitted to Konsorsium Turnitin Relawan Jurnal Indonesia

Student Paper

<1%

72

Submitted to Padjadjaran University

Student Paper

<1%

73

Submitted to Universitas Muhammadiyah Makassar

Student Paper

<1%

74

Veysel Yusuf Cambay, Prabal Datta Barua, Abdul Hafeez Baig, Sengul Dogan, Mehmet Baygin, Turker Tuncer, U. R. Acharya. "Automated Detection of Gastrointestinal Diseases Using Resnet50*-Based Explainable Deep Feature Engineering Model with Endoscopy Images", Sensors, 2024

Publication

<1%

75	ejurnal.ars.ac.id Internet Source	<1 %
76	jurnal.umk.ac.id Internet Source	<1 %
77	simantik.panca-sakti.ac.id Internet Source	<1 %
78	Meddy Nurpratama, Nana Supriatna Sonjaya, Agus Yudianto, Ismail Agung. "Studi Orientasi Pasar dan Inovasi Produk Terhadap Kinerja Pemasaran UMKM di Kabupaten Indramayu", Jurnal EMT KITA, 2024 Publication	<1 %
79	ejournal.nusamandiri.ac.id Internet Source	<1 %
80	jurnal.upnyk.ac.id Internet Source	<1 %
81	ojs.unud.ac.id Internet Source	<1 %
82	pubmed.ncbi.nlm.nih.gov Internet Source	<1 %
83	Muh Ikbal Fauzi. "KLASIFIKASI IMAGE TINGGI TANAMAN JAGUNG DENGAN MENGGUNAKAN ALGORITMA CONVOLUTION NEURAL NETWORK (CNN)"KLASIFIKASI IMAGE TINGGI TANAMAN JAGUNG DENGAN MENGGUNAKAN ALGORITMA CONVOLUTION NEURAL NETWORK (CNN)", Jurnal Informatika dan Teknik Elektro Terapan, 2024 Publication	<1 %
84	blog.csdn.net Internet Source	<1 %

85

Anasthasya Averina, Helen Hadi, Joko Siswanto. "Analisis Sentimen Multi-Kelas Untuk Film Berbasis Teks Ulasan Menggunakan Model Regresi Logistik", Teknik, 2022

Publication

<1%

86

docs.google.com

Internet Source

<1%

87

jurnal.atmaluhur.ac.id

Internet Source

<1%

88

mlflow.org

Internet Source

<1%

89

www.mdpi.com

Internet Source

<1%

Muhammad Albani, Rahmat Rizal Andhi. "Klasifikasi Uang Rupiah Kertas Tidak Layak Edar Menggunakan CNN Xception Transfer Learning Berbasis Website", INOVTEK Polbeng - Seri Informatika, 2023

Publication

<1%

Sarifah Agustiani, Yoseph Tajul Arifin, Agus Junaidi, Siti Khotimatul Wildah, Ali Mustopa. "Klasifikasi Penyakit Daun Padi menggunakan Random Forest dan Color Histogram", Jurnal Komputasi, 2022

Publication

<1%

Siti Hajar, Murinto Murinto, Anton Yudhana. "Identifikasi Jenis Daun untuk Ecoprint Menggunakan Metode Convolutional Neural Network", Jurnal Sains dan Informatika, 2025

Publication

<1%

93	Yefta Tolla, Kusrini. "Deteksi Stres dan Depresi Unggahan Media Sosial dengan Machine Learning", JURNAL FASILKOM, 2025 Publication	<1 %
94	ind.acousticbiotech.com Internet Source	<1 %
95	journal.artei.or.id Internet Source	<1 %
96	jurnal.itscience.org Internet Source	<1 %
97	library.binus.ac.id Internet Source	<1 %
98	repo.stimata.ac.id Internet Source	<1 %
99	repositorio.unican.es Internet Source	<1 %
100	repository.ibs.ac.id Internet Source	<1 %
101	repository.trisakti.ac.id Internet Source	<1 %
102	snhrp.unipasby.ac.id Internet Source	<1 %
103	www.contohjurnal.net Internet Source	<1 %
104	Dhini Septhya, Rahmaddeni, Susanti, Agustin. "Penerapan Algoritma Convolutional Neural Network Untuk Klasifikasi Penyakit Kanker Kulit", The Indonesian Journal of Computer Science, 2024 Publication	<1 %

105 Erin Ermawati, Anik Sri Wahyuningsih.
"PERANCANGAN SISTEM INFORMASI
PENGOLAHAN DATA NILAI SISWA BERBASIS
WEB", Jurnal Informasi dan Komputer, 2021
Publication

106 Gaurav Aggarwal, Ashutosh Tripathi, Himani
Goyal Sharma, Tripti Sharma, Rishabh Dev
Shukla. "Integrated Technologies in Electrical,
Electronics and Biotechnology Engineering",
CRC Press, 2025
Publication

107 Panji Novantara, Risteruw Leonardo
Firmansyah, Marrilyn Arismawati. "Deteksi
Hama Penyakit Daun Padi Dengan
Menggunakan Teknik Optimasi Deep Learning
Convolutional Neural Network", bit-Tech,
2025
Publication

108 Sariah Sariah, Nana Suarna, Irfan Ali, Dodi
Solihudin. "Penerapan Convolutional Neural
Network (CNN) Untuk Prediksi Penyakit
Tanaman Padi Melalui Citra Daun", Jurnal
Komtika (Komputasi dan Informatika), 2025
Publication

109 Sayyid Ahmad Wisak, Nindy Aulia Safirah,
Yampi R Kaesmetan. "IDENTIFIKASI JENIS
MANGGA BERDASARKAN CIRI DAUN
MENGUNAKAN METODE CNN", Jurnal
Sistem Informasi dan Informatika (Simika),
2024

110 Submitted to Sriwijaya University
Student Paper

111	doku.pub Internet Source	<1 %
112	ejournal.akakom.ac.id Internet Source	<1 %
113	ejournal.unida.gontor.ac.id Internet Source	<1 %
114	ejurnal.teknokrat.ac.id Internet Source	<1 %
115	eprints.umsb.ac.id Internet Source	<1 %
116	erlinasitumeang.wordpress.com Internet Source	<1 %
117	idus.us.es Internet Source	<1 %
118	journal.mahecenter.org Internet Source	<1 %
119	jurnal.goretanpena.com Internet Source	<1 %
120	repositori.uma.ac.id Internet Source	<1 %
121	repositori.usu.ac.id Internet Source	<1 %
122	repository.radenintan.ac.id Internet Source	<1 %
123	repository.uisi.ac.id Internet Source	<1 %
124	repository.uksw.edu Internet Source	<1 %

125

Internet Source

<1 %

126

I Wayan Suardi. "Perbandingan Nilai Akurasi Analisa Sentiment Pada Kata Kunci Pemilu 2024", The Indonesian Journal of Computer Science, 2025

Publication

<1 %

127

Alfitriana Riska, Purnawansyah, Herdianti Darwis, Wistiani Astuti. "Studi Perbandingan Kombinasi GMI, HSV, KNN, dan CNN pada Klasifikasi Daun Herbal", The Indonesian Journal of Computer Science, 2023

Publication

<1 %

128

Lo Mei Ly Vebriyanti, Shantika Martha, Wirda Andani, Setyo Wira Rizki. "Analisis Kelayakan Kredit Menggunakan Classification Tree dengan Teknik Random Oversampling", Euler : Jurnal Ilmiah Matematika, Sains dan Teknologi, 2024

Publication

<1 %

129

Raihan Raihan, Cecep Nurul Alam, Wildan Budiawan Zulfikar. "Deteksi Pneumonia pada Citra Akhir X – Ray Dada Menggunakan Convolutional Neural Networks Berdasarkan Fitur Prewitt Operator", INTERNAL (Information System Journal), 2025

Publication

<1 %

130

T. Mariprasath, Kumar Reddy Cheepati, Marco Rivera. "Practical Guide to Machine Learning, NLP, and Generative AI: Libraries, Algorithms, and Applications", River Publishers, 2024

Publication

<1 %

131

dokumen.pub

Internet Source

<1%

132

id.scribd.com
Internet Source

<1%

133

www.slideshare.net
Internet Source

<1%

Exclude quotes On

Exclude matches Off

Exclude bibliography On