PERBANDINGAN ARSITEKTUR RESNET152V2 DAN ALEXNET DALAM MENDETEKSI PENYAKIT TANAMAN JAGUNG MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK* BERBASIS WEB

SKRIPSI

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Jenjang Strata Satu (S1)
Pada Program Studi Teknik Informatika Fakultas Teknik
Universitas Muhammadiyah Ponorogo

ADELYA ARMIAJI ILHAM
20533323

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PONOROGO
2024

HALAMAN PENGESAHAN

Nama	:	Adelya Armiaji Ilham
NIM	:	20533323
Program Studi	:	Teknik Informatika
Fakultas	:	Teknik
Judul Skripsi	:	Perbandingan Arsitektur Resnet152v2 Dan Alexnet Dalam Mendeteksi Penyakit Tanaman Jagung Menggunakan Metode *Convolutional Neural Network* Berbasis Web

Isi dan formatnya telah disetujui dan dinyatakan memenuhi syarat
Untuk mengikuti sidang skripsi Pada Program Studi Informatika
Fakultas Teknik Universitas Muhammadiyah Ponorogo

Ponorogo, 5 Agustus 2024

Menyetujui,

Dosen Pembimbing Utama,	Dosen Pembimbing Pendamping,
(Ghulam Asrofi Buntoro, S.T., M.Eng)	(Andy Triyanto Pujo Raharjo, S.T., M.Kom)
NIK. 19870723 202109 12	NIK. 19710521 201101 13

Mengetahui,

Dekan Fakultas Teknik,	Ketua Program Studi Teknik Informatika,
(Edy Kurniawan, ST., MT)	(Adi Fajaryanto C, S. Kom, M.Kom)
NIK. 19771026 200810 12	NIK. 19840924 201309 13

i

PERNYATAAN ORISINALITAS SKRIPSI

Yang bertanda tangan di bawah ini :

Nama : Adelya Armiaji Ilham

NIM : 20533323

Program Studi : Teknik Informatika

Dengan ini menyatakan bahwa Skripsi saya dengan judul : "Perbandingan Arsitektur Resnet152v2 Dan Alexnet Dalam Mendeteksi Penyakit Tanaman Jagung Menggunakan Metode *Convolutional Neural Network* Berbasis Web" bahwa berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang saya rancang/ teliti di dalam Naskah Skripsi ini adalah asli dari pemikiran saya. Tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam Naskah Skripsi ini dapat dibuktikan terdapat unsur-unsur plagiarisme, saya bersedia Ijazah saya dibatalkan, serta diproses sesuai dengan pengaturan perundangan-undangan yang berlaku

Demikian pernyataan ini dibuat dengan sesungguhnya dan dengan sebenar-benarnya.

Ponorogo, 5 Agustus 2024

Mahasiswa,

Adelya Armiaji Ilham

NIM.20533323

ii

HALAMAN BERITA ACARA UJIAN

Nama	:	Adelya Armiaji Ilham
NIM	:	20533323
Program Studi	:	Teknik Informatika
Fakultas	:	Teknik
Judul Skripsi	:	Perbandingan Arsitektur Resnet152v2 Dan Alexnet Dalam Mendeteksi Penyakit Tanaman Jagung Menggunakan Metode *Convolutional Neural Network* Berbasis Web

Telah diuji dan dipertahankan dihadapan

Dosen Penguji tugas akhir jenjang Strata Satu (S1) pada:

Hari : Senin

Tanggal : 5 Agustus 2024

Dosen Penguji

Ketua Penguji	Anggota Penguji I	Anggota Penguji II
(Ghulam Asrofi Buntoro, S.T., M.Eng)	(Dra. Ida Widaningrum, M.Kom)	(Yovi Litanianda, S.Pd, M.Kom)
NIK. 19870723 202109 12	NIK. 19660417 201101 13	NIK. 19810221 201309 13

Mengetahui

Dekan Fakultas Teknik	Ketua Program Studi Teknik Informatika
(Edy Kurniawan, ST., MT)	(Adi Fajaryanto Cobantoro, S. Kom, M.Kom)
NIK. 19771026 200810 12	NIK. 19840924 201309 13

iii

BERITA ACARA
BIMBINGAN SKRIPSI

Nama : Adelya Armiaji Ilham

NIM : 20533323

Judul Skripsi : Sistem Deteksi Penyakit Tanaman Jagung Menggunakan Algoritma Convolutional Neural Network Berbasis Web.

Dosen Pembimbing I : Ghulam Asrofi B, S.T., M.Eng

PROSES PEMBIMBINGAN

No	Tanggal	Materi Yang Dikonsultasikan	Saran Pembimbing / Hasil	Tanda Tangan
1	02/11 2023	Pengajuan Tema	Algoritma diperkuat dan cari referesi judul	
2	07/11 2023	Acc judul	Lanjut Bab 1	
3	15/12 2023	Bab I	Metode permasalahan di perjelas lagi dan banyak riset	
4	23/12 2023	BAB I	Acc bab I lanjut bab II untuk penulisan diedit lagi	

No	Tanggal	Materi Yang Dikonsultasikan	Saran Pembimbing / Hasil	Tanda Tangan
5	09/02 2024	BAB II	Penelitian terdahulu minimal 5 tahun terakhir Flowchart tidak terurut	
6	14/03 2024	BAB II	- Alur Interface dipanjelas lagi - Tidak ada penjelasan Algoritma - lanjut Bab III	
7	30/04 2024	ACC BAB III	Ditambahkan Reabel penelitian Pengambilan data di tambah	
8	20/05 2024	ACC BAB I dan III	Acc semua	
9	14/06 2024	BAB IV	Penekanan masalah perlu diperjelas	
10	17/06 2024	BAB IV	- Simulasi algoritma - kalimat di bab IV tekan sesuai	

No	Tanggal	Materi Yang Dikonsultasikan	Saran Pembimbing / Hasil	Tanda Tangan
11	26/06 2024	BAB IV	Abstrak belum runtut	
12	28/06 2024	BAB IV	Tambahkan menu algoritma sebagai pembanding	
13	04/07 2024	Demo program	Selesaikan ketemahan sistem/ masukkan ke saran	
14	09/07 2024	Program	Pengujian menggunakan partisipan manual	
15	21/07 2024	BAB V	Saran harus di penjelas	
16	30/07 2024		Bu sidang	

BERITA ACARA
BIMBINGAN SKRIPSI

Nama : Adelya Armiaji Ilham

NIM : 20533323

Judul Skripsi : Sistem Deteksi Penyakit Tanaman Jagung Menggunakan Algoritma Convolutional Neural Network Berbasis Web

Dosen Pembimbing II : Andy Triyanto Purjo R, S.T., M.Kom

PROSES PEMBIMBINGAN

No	Tanggal	Materi Yang Dikonsultasikan	Saran Pembimbing / Hasil	Tanda Tangan
1	04/11 2023	Pengajuan Tema	Penambahan algoritma yang spesifik dengan judul temal yang akan dibahas	
2	08/11 2023	ACC Judul	Lanjut bab I	
3	16/12 2023	BAB I	- Metode penelitian diperkuat lagi - Manfaat ditambah - Paragraf dirapikan - penulisan ada yang salah	
4	29/12 2023	BAB I	ACC BAB I lanjut BAB II penulisan di daftar isi diatur	

No	Tanggal	Materi Yang Dikonsultasikan	Saran Pembimbing / Hasil	Tanda Tangan
5	10/02 2024	BAB II	- Penomoran halaman harap diperhatikan - Alur penulisan sebuah subbab di tupikan	
6	16/04 2024	BAB II	Interface dan flowchart harap disesuaikan dengan pedoman buku	
7	30/04 2024	ACC BAB III	Lampiran tabel penelitian sesudah seminar	
8	30/05 2024	ACC BAB I II III	ACC Sempro	
9	04/06 2024	BAB IV	Daftar isi masih ada yang error Daftar tabel dan gambar	
10	07/06 2024	BAB IV	Penomoran dan spasi diurutkan	

No	Tanggal	Materi Yang Dikonsultasikan	Saran Pembimbing / Hasil	Tanda Tangan
11	10/06 2024	~~08/06~~ BAB ~~2024~~ 4	Penulisan bahasa inggris cetak miring	
12	15/06 2024	WEB	Demoo web pengujian penyakit tanaman jagung	
13	22/06 2024	WEB BAB IV	Syntax masih eror dan database belum tertata	
14	27/06 2024	BAB V	Kesimpulan harap diperbaiki	
15	24/07 2024	BAB IV V WEB	Pengujian web belum aturan paragraf belum tertata Daftar isi belum rapi	
16	01/08 2024	Full BAB & WEB	1/08 24. Acc Sidang!	

SURAT KETERANGAN HASIL PLAGIASI SKRIPSI

UNIVERSITAS MUHAMMADIYAH PONOROGO
LEMBAGA LAYANAN PERPUSTAKAAN
Jalan Budi Utomo No. 10 Ponorogo 63471 Jawa Timur Indonesia
Telp. (0352) 481124, Fax (0352) 461796, e-mail : lib@umpo.ac.id
website : www.library.umpo.ac.id
TERAKREDITASI A
(SK Nomor 000137/ LAP.PT/ III.2020)
NPP. 3502102D2014337

SURAT KETERANGAN
HASIL *SIMILARITY CHECK* KARYA ILMIAH MAHASISWA
UNIVERSITAS MUHAMMADIYAH PONOROGO

Dengan ini kami nyatakan bahwa karya ilmiah ilmiah dengan rincian sebagai berikut :

Nama : Adelya Armiaji Ilham
NIM : 20533323
Judul : Perbandingan Arsitektur Resnet152v2 Dan Alexnet Dalam Mendeteksi Penyakit Tanaman Jagung Menggunakan Metode Convolutional Neural Network Berbasis Web
Fakultas / Prodi : Teknik Informatika

Dosen pembimbing :
1. Ghulam Asrofi Buntoro, S.T., M.Eng
2. Andy Triyanto Pujo Raharjo, S.T., M.Kom

Telah dilakukan check plagiasi berupa **Skripsi** di Lembaga Layanan Perpustakaan Universitas Muhammadiyah Ponorogo dengan prosentase kesamaan sebesar **12 %**

Demikian surat keterangan dibuat untuk digunakan sebagaimana mestinya.

Ponorogo, 22/Juli/2024
Kepala Lembaga Layanan Perpustakaan

Ayu Wulansari, S.Kom, M.A
NIK. 19760811 201111 21

NB: Dosen pembimbing dimohon untuk melakukan verifikasi ulang terhadap kelengkapan dan keaslian karya beserta hasil cek Turnitin yang telah dilakukan

KATA PERSEMBAHAN

Alhamdulillah, Puji Syukur kepada Allah SWT atas limpahan nikmat, rahmat serta kesehatan sehingga penulis dapat menyelesaikan skripsi dengan judul "Perbandingan Arsitektur Resnet152v2 Dan Alexnet Dalam Mendeteksi Penyakit Tanaman Jagung Menggunakan Metode Convolutional Neural Network Berbasis Web". Dalam penyusunan skripsi ini, saya banyak mendapatkan bimbingan, pengetahuan, serta dukungan dari banyak pihak yang selama ini membantu dalam menyelesaikan skripsi ini. Sebagai ungkapan rasa cinta dan terima kasih, karya tulis Tugas Akhir ini saya persembahkan kepada:

1. Allah SWT, telah memberikan kekuatan fisik dan jiwa hingga dapat menyelesaikan tugas akhir skripsi ini dengan baik.
2. Kedua orang tua tercinta, Bapak Hartono Tri Atmodjo dan Ibu Amin Syarifatul Hasanah selalu memberikan support dalam bentuk apapun dan doa dalam setiap penyusunan dan pengerjaan skripsi. Sangat Beruntung memiliki kedua orang tua yang selalu mendoakan dan memberikan support terbaik bagi anaknya. Serta seluruh keluarga yang telah memberikan semangat dalam proses saya.
3. Untuk keluarga besar tercinta. Terima kasih yang sebesar-besarnya atas segala bentuk dukungan, doa, dan kasih sayang yang terus mengiringi perjalanan ini. Tanpa kehadiran dan motivasi yang tak ternilai dari keluarga, pencapaian ini tidak akan mungkin tercapai. Semoga karya ini menjadi cerminan dari segala pengorbanan dan cinta yang telah diberikan.
4. Bapak Ghulam Asrofi Buntoro, S.T., M.Eng , selaku Dosen Pembimbing I yang tidak pernah lelah memberikan dukungan, arahan serta bimbingan penuh sehingga saya bisa terus termotivasi untuk menyelesaikan skripsi ini.
5. Bapak Andy Triyanto Pujo Raharjo, S.T., M.Kom, selaku Dosen Pembimbing II yang telah memberikan bimbingan dan arahan penuh untuk menyelesaikan skripsi ini.

6. Untuk rekan-rekan Teknik Informatika C, terima kasih atas setiap langkah yang kita lalui bersama. Kebersamaan dan dukungan kalian menjadi sumber kekuatan dalam menghadapi segala tantangan. Semoga kesuksesan senantiasa mengiringi kita ke depan.
7. Teman seperjuangan, Angkatan 2020 Teknik Informatika, terima kasih sudah menemani saya berproses dalam menghadapi dunia perkuliahan. Terima kasih untuk energi positif yang telah diberikan, sukses untuk kalian semua.
8. Teruntuk seorang yang tidak bisa sebutkan nama nya. Saya mengucapkan terima kasih atas jasa dan dukungan yang telah diberikan selama proses penyusunan skripsi ini. Terima kasih atas kesediaan merelakan waktu, tenaga, materi, dan pikiran dengan ikhlas untuk mendukung saya. Tanpa inspirasi, dorongan, dan dukungan yang telah diberikan, saya mungkin tidak akan mencapai titik ini. Terima kasih telah hadir dan setia mendampingi saya, terutama pada saat-saat terpuruk. Dukungan ini sangat berarti bagi saya.
9. Terima kasih untuk diri sendiri, karena telah mampu berusaha keras dan berjuang sejauh ini dari banyaknya hal dalam penulisan. Terima kasih untuk tidak menyerah sesulit apapun proses penyusunan skripsi ini dengan hasil sebaik dan semaksimal mungkin.

MOTTO

"Yang di katakan menang bukan berarti saya menang atas nama orang lain. Tapi saya menang atas diri saya sendiri. Rasullullah SAW mengatakan bahwa jihad yang terbesar adalah jihad melawan napsumu sendiri. Karena menang dalam hidup saya bukanlah sayang unggul atas anda. Menang bagi saya adalah Allah SWT ridho kepada saya, Allah SWT Mengakui kebenaran sikap hidup saya"

(Mbah Nun)

"Allah tidak membebani seseorang, kecuali menurut kesanggupannya"

(Al-Baqarah · Ayat 286)

"Orang lain nggak akan bisa faham dengan *Struggle* dan masa sulitnya kita yang mereka ingin tahu hanya bagian *Success Stories*. Berjuanglah Untuk Diri Sendiri Walaupun tidak ada yang bertepuk tangan. Kelak diri kita dimasa depan akan sangat bangga dengan apa yang kita perjuangan hari ini"

"Mulai dengan keyakinan, menjalankan dengan penuh keiklasan, menyelesaikan dengan penuh kebahagiaan"

PERBANDINGAN ARSITEKTUR RESNET152V2 DAN ALEXNET DALAM MENDETEKSI PENYAKIT TANAMAN JAGUNG MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK BERBASIS WEB

Adelya Armiaji Ilham[1], Ghulam Asrofi Buntoro [2], Andy Triyanto Pujo Raharjo[3]
Program Studi Teknik Informatika, Fakultas Teknik
Universitas Muhammadiyah Ponorogo
Gmail : armiaji123@gmail.com

Abstrak

Tanaman jagung (*Zea mays*) merupakan komoditas utama bagi petani di Indonesia. Namun, tanaman ini rentan terhadap berbagai penyakit seperti *Common Rust, Gray Leaf Spot*, dan *Northern Leaf Blight* yang dapat mengakibatkan penurunan hasil panen. Penelitian ini bertujuan untuk mengimplementasikan Metode *Convolutional Neural Network* (CNN) dalam Sistem Deteksi Penyakit Daun Jagung berbasis *web* yang mencakup dataset *Common Rust, Gray Leaf Spot, Healthy*, dan *Northern Leaf Blight*. Dataset yang digunakan terdiri dari 3852 citra daun jagung, termasuk kelas *Common Rust, Gray Leaf Spot, Healthy*, dan *Northern Leaf Blight*. Metode CNN yang diimplementasikan menggunakan model ResNet152V2 dan AlexNet. Hasil eksperimen menunjukkan bahwa ResNet152V2 mencapai akurasi 100% dalam mengidentifikasi semua kelas penyakit daun jagung, membuktikan efektivitas tinggi dalam deteksi penyakit. Di sisi lain, model AlexNet menunjukkan variasi akurasi yang lebih besar, dengan akurasi tertinggi 99.99% untuk *Northern Leaf Blight* dan terendah 44.22% *untuk Gray Leaf Spot*. Hal ini menegaskan bahwa ResNet152V2 lebih konsisten dan akurat dibandingkan AlexNet dalam mendeteksi penyakit pada tanaman jagung. Keunggulan ResNet152V2 terletak pada arsitektur yang lebih dalam dan kemampuannya mengatasi masalah vanishing gradient, sehingga dapat mengenali fitur-fitur spesifik dari berbagai kondisi daun jagung dengan sangat baik. Sementara itu, meskipun AlexNet mampu memberikan akurasi tinggi dalam beberapa kasus, model ini menunjukkan ketidakstabilan performa yang mengindikasikan perlunya penyesuaian atau data pelatihan yang lebih representatif. Secara keseluruhan, hasil ini menunjukkan bahwa ResNet152V2 adalah arsitektur yang lebih andal untuk identifikasi penyakit pada daun jagung dibandingkan dengan AlexNet.

Kata Kunci : AlexNet, Metode *Convolutional Neural Network*, ResNet152V2, Tanaman Jagung

KATA PENGANTAR

Assalamualaikum warahmatullahi wabarakatuh.

Puji Syukur saya panjatkan kepada Allah SWT yang senantiasa memberikan Rahmat serta hidayah-Nya, sehingga saya dapat menyelesaikan skripsi yang berjudul "Perbandingan Arsitektur Resnet152v2 Dan Alexnet Dalam Mendeteksi Penyakit Tanaman Jagung Menggunakan Metode Convolutional Neural Network Berbasis Web". Shalawat serta salam saya haturkan kepada Nabi Muhammad SAW, yang telah membawa manusia dari zaman jahiliyah menuju zaman Islamiyah seperti saat ini. Skripsi ini merupakan syarat untuk memperoleh Gelar Sarjana Strata Satu (S-1) Program Studi Teknik Informatika Fakultas Teknik Universitas Muhammadiyah Ponorogo. Penulisan skripsi ini, saya menyadari banyak mengalami kesulitan yang dihadapi, tetapi berkat bantuan, dukungan, dan doa dari berbagai pihak, skripsi ini dapat terselesaikan. Oleh karena itu, saya menyampaikan terimakasih tulus dan hormat kepada:

1. Allah SWT dengan segala Rahmat serta karunia-Nya yang telah memberikan kekuatan dan kemudahan dalam menyelesaikan skripsi ini.
2. Bapak Edy Kurniawan, ST., MT, selaku Dekan Fakultas Teknik Universitas Muhammadiyah Ponorogo.
3. Bapak Adi Fajaryanto Cobantoro, S. Kom, M.Kom, selaku ketua Program Studi Teknik Informatika Fakultas Teknik Universitas

Muhammadiyah Ponorogo.

4. Bapak Ghulam Asrofi Buntoro, S.T., M.Eng, selaku Dosen Pembimbing I yang telah memberikan bimbingan dan arahan penuh untuk menyelesaikan skripsi ini.
5. Bapak Andy Triyanto Pujo Raharjo, S.T., M.Kom, selaku Dosen Pembimbing II yang telah memberikan bimbingan dan arahan penuh untuk menyelesaikan skripsi ini.

Bapak dan Ibu Dosen Fakultas Teknik Universitas Muhammadiyah Ponorogo, beserta staff atas ilmu dan pengalaman yang telah diberikan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Tanaman jagung *(Zea mays)* adalah salah satu tanaman penting di Indonesia dan menjadi komoditas utama bagi petani di berbagai daerah. Jagung tumbuh baik di dataran rendah maupun dataran tinggi dan dapat ditemukan hampir di seluruh wilayah Indonesia. Tanaman jagung dapat memberikan hasil yang melimpah jika tanaman tersebut dalam keadaan sehat. Namun, seperti halnya tanaman lainnya jagung juga rentan terhadap serangan penyakit yang dapat menyebabkan kerusakan yang signifikan. Oleh karena itu, sangat penting bagi petani untuk mendeteksi penyakit pada tanaman jagung lebih awal guna mengontrol penyebarannya[1]. Beberapa jenis penyakit yang sering muncul dalam tanaman jagung adalah *Leaf Blight, Rust, Corn Smut, Maize Dwarf Mosaic Virus*, dan *Corn Stalk Rot*. Tanaman jagung rentan terhadap banyak penyakit yang menyerang daun, batang, dan bahkan bagian tajuknya. Penyakit-penyakit ini bisa merugikan pertumbuhan tanaman jagung, mengakibatkan penurunan hasil panen yang signifikan. Penyakit pada jagung dapat menyebar melalui berbagai cara seperti air hujan, angin, serangga, atau melalui tanah yang terkontaminasi. Akibatnya, tanaman jagung yang terinfeksi dapat menunjukkan gejala seperti bercak-bercak pada daun, pembusukan batang, atau bahkan kematian tanaman secara keseluruhan[2].

Pemberian tindakan yang tepat dalam penanganan penyakit tanaman menjadi hal penting untuk menghindari masalah yang lebih serius. Metode tradisional untuk pengenalan penyakit pada tanaman bergantung pada interpretasi visual dari gejala yang muncul, memerlukan analisis laboratorium, dan tenaga ahli di bidang plantologi[3]. Tanaman yang terserang penyakit dapat dilihat dari kondisi daunnya, sehingga memantau tanaman secara rutin untuk mendeteksi gejala penyakit menjadi penting. Tindakan lain yang perlu dilakukan meliputi memastikan penempatan tanaman yang tepat untuk menjaga kondisi optimal, memotong dan membuang bagian tanaman yang terinfeksi,

serta memberikan perlakuan dengan pestisida atau fungisida. Demi menghindari kerugian yang disebabkan oleh penyakit-penyakit tersebut, petani jagung perlu mengadopsi praktik-praktik budidaya yang baik, termasuk pemilihan varietas yang tahan penyakit, rotasi tanaman, dan penggunaan metode pengendalian penyakit yang tepat. Upaya pencegahan yang efektif dapat membantu menjaga kesehatan tanaman jagung dan meningkatkan hasil panen yang optimal[1].

Penelitian ini menggunakan *dataset* yang diunggah ke Kaggle oleh Nafisha Moin. *Dataset* ini berisi citra daun yang terbagi menjadi empat kelas yaitu *Common Rust, Gray Leaf Spot, Healthy,* dan *Northern Leaf Blight* dengan total *dataset* sebanyak 3852 *images*. Data ini akan digunakan sebagai dasar untuk sistem deteksi penyakit pada daun jagung. Metode identifikasi penyakit pada daun jagung dapat mempercepat dan meningkatkan akurasi diagnosis. Proses ini melibatkan klasifikasi citra berdasarkan ciri-ciri yang dimilikinya. Menurut Isna Wulandari dkk., pendekatan ini meniru kemampuan manusia dalam memahami citra digital, memungkinkan komputer untuk mengenali objek seperti halnya manusia. Langkah awal melibatkan pengambilan dan pengolahan citra digital dari daun jagung terinfeksi. Kemudian, citra-citra ini digunakan sebagai data pelatihan untuk mengenali pola dan karakteristik penyakit. Dalam pembelajaran mesin, augmentasi data adalah kunci untuk meningkatkan kinerja model dan mencegah *overfitting* pada dataset terbatas. Teknik-teknik augmentasi seperti rotasi, skala, *flip, kontras*, dan perubahan warna digunakan untuk menciptakan variasi data yang lebih beragam. Hasilnya metode *Convolutional Neural Network* (CNN) dapat meningkatkan kemampuan generalisasi pada citra yang beragam. Dengan menerapkan teknik ini, dataset pelatihan menjadi lebih kaya dan model menjadi lebih adaptif terhadap variasi dalam data[4].

Penelitian yang dilakukan oleh Mochammad Kevin Santosa et al, membandingkan arsitektur ResNet152v2 dan AlexNet menggunakan metode *Convolutional Neural Network* (CNN) untuk klasifikasi penyakit pada daun kentang termasuk kategori *healthy*, *late blight*, dan *early blight*. Hasil penelitian menunjukkan bahwa arsitektur ResNet152v2 menghasilkan akurasi yang lebih tinggi dibandingkan AlexNet. Selain itu, penelitian lain juga mengungkapkan bahwa arsitektur ResNet152v2 lebih unggul dibandingkan AlexNet dalam klasifikasi citra penyakit daun kentang. ResNet152v2 mencapai akurasi tertinggi sebesar 99% setelah dilatih selama 16 *epoch* dengan *batch size* 14, serta menggunakan *learning rate* dan *weight decay* masing-masing sebesar 0,0001. Sebaliknya, AlexNet hanya mencapai akurasi sebesar 78% dengan konfigurasi pelatihan yang sama. Oleh karena itu, arsitektur ResNet152v2 menunjukkan performa yang akurat dalam hal akurasi dibandingkan AlexNet, sehingga ResNet152v2 lebih disarankan untuk aplikasi klasifikasi citra yang memerlukan tingkat akurasi yang tinggi[5]. Berdasarkan penelitian tersebut, peneliti akan menggunakan arsitektur ResNet152v2 dan AlexNet untuk membandingkan kinerja keduanya dalam mengklasifikasi penyakit daun jagung dengan menggunakan dataset sebanyak 3852 gambar. Diharapkan bahwa hasil dari perbandingan ini akan memberikan dampak yang signifikan dalam peningkatan kualitas tanaman jagung khususnya di daerah Ponorogo.

Berdasarkan uraian di atas dapat diperoleh bahwa penelitian ini akan disusun dengan Judul **"Perbandingan Arsitektur Resnet152v2 Dan Alexnet Dalam Mendeteksi Penyakit Tanaman Jagung Menggunakan Metode *Convolutional Neural Network* Berbasis Web".**

1.2 Rumusan Masalah

Rumusan masalah penelitian ini adalah Bagaimana hasil perbandingan performa model ResNet152V2 dan AlexNet dalam Sistem Deteksi Penyakit Daun Jagung berbasis web.

1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk mengetahui hasil perbandingan performa model ResNet152V2 dan AlexNet dalam Sistem Deteksi Penyakit Daun Jagung berbasis web.

1.4 Batasan Masalah

Adapun batasan masalah yang ada pada penelitian ini yaitu :

1) Penelitian ini membandingkan performa model ResNet152V2 dan AlexNet dalam Sistem Deteksi Penyakit Daun Jagung berbasis web
2) *Dataset* yang digunakan dalam penelitian ini yaitu *Common Rust, Gray Leaf Spot, Healthy,* dan *Northern Leaf Blight*.
3) Penelitian ini menggunakan dataset dengan jumlah yaitu 3852 citra.

1.5 Manfaat Penelitian

Manfaat penelitian ini terdapat beberapa aspek, sebagai berikut :

1) Bagi peneliti, penelitian ini dapat memberikan kontribusi dalam pengembangan teknologi informasi, khususnya dalam pengolahan citra digital dengan menggunakan Metode CNN. Penelitian ini juga dapat menjadi landasan bagi penelitian lanjutan dalam bidang pengolahan citra.
2) Bagi civitas akademika, penelitian ini dapat meningkatkan reputasi universitas atau institusi yang melakukan penelitian. Selain itu, hasil penelitian ini juga dapat digunakan sebagai materi pengajaran dalam mata kuliah pengolahan citra dan machine learning.
3) Bagi masyarakat, penelitian ini dapat membantu dalam meningkatkan kualitas dan hasil produksi tanaman jagung. Dengan mendeteksi penyakit pada daun jagung secara dini, dapat dilakukan tindakan pencegahan yang tepat untuk mengurangi kerugian dan meningkatkan hasil panen secara keseluruhan.

BAB 2
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian terdahulu memiliki peran penting dalam menjelaskan alasan dan kebutuhan penelitian baru. Informasi teknik-teknik dan kendala yang dihadapi dapat menjadi acuan dalam pengembangan Metode yang lebih efektif untuk mengidentifikasi citra daun jagung. Oleh karena itu, hasil penelitian terdahulu dapat memperluas pengetahuan dan pemahaman mengenai pengembangan CNN dalam mengidentifikasi jenis penyakit daun jagung. Referensi penelitian terdahulu yang relevan dapat ditemukan dalam tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No.	Penulis (Tahun)	Judul	Hasil Penelitian
1.	Rozaqi, Abdul Jalil (2021)	Deteksi Penyakit pada Daun Kentang Menggunakan Pengolahan Citra dengan Metode *Convolutional Neural Network*[5]	Penelitian ini menggunakan metode *Convolutional Neural Network* dalam mengidentifikasi penyakit daun kentang. Hasil dari penelitian ini menunjukkan bahwa penerapan metode CNN untuk deteksi penyakit daun kentang memberikan hasil yang positif dengan nilai akurasi 95% dan untuk akurasi validasi menghasilkan 94%.
2.	S. Fiviana & Sri Anardani (2023)	Aplikasi Deteksi Penyakit Tanaman Jagung Dengan	Penelitian ini menggunakan metode *Convolutional Neural Network* (CNN) dan

		Convolutional Neural Network dan *Support Vector Machine*[6].	*Support Vector Machine* (SVM) untuk deteksi penyakit pada daun jagung melalui analisis citra daun. Hasil dari penelitian ini menunjukkan bahwa metode *Convolutional Neural Network* (CNN) memiliki akurasi sebesar 98% dalam mendeteksi penyakit tanaman jagung melalui citra daun, sedangkan metode *Support Vector Machine* (SVM) memiliki akurasi sebesar 87%.
3.	Bima Widianto, dkk (2023)	Identifikasi Penyakit Tanaman Jagung Berdasarkan Citra Daun Menggunakan Convolutional Neural Network[7].	Penelitian ini menggunakan metode *Convolutional Neural Network* (CNN) untuk mengidentifikasi penyakit pada daun tanaman jagung. Hasil dari penelitian ini menunjukkan bahwa metode Convolutional Neural Network (CNN) mampu menghasilkan tingkat akurasi yang baik, dengan nilai mencapai 94%.

4.	Andhika Bagas Prakoso (2023)	Implementasi Model *Deep Learning Convolutional Neural Network* (CNN) Pada Citra Penyakit Daun Jagung Untuk Klasifikasi Penyakit Tanaman[8].	Penelitian ini menggunakan *Convolutional Neural Network* (CNN) untuk klasifikasi penyakit daun jagung. Hasil dari penelitian ini menunjukkan bahwa penerapan *Convolutional Neural Network* (CNN) pada citra penyakit daun jagung menghasilkan nilai akurasi sebesar 0.9990, nilai *precision* sebesar 0.9981, nilai *recall* sebesar 1, dan nilai F1 *Score* sebesar 0.9990.
5.	Ary Yoggyanto, dkk (2024)	Penerapan Metode *Convolutional Neural Network* (CNN) Dalam Klasifikasi Penyakit Tanaman Jagung[9].	Penelitian ini menggunakan metode *Convolutional Neural Network* (CNN) digunakan untuk mengklasifikasikan penyakit pada tanaman jagung. Hasil dari penelitian ini menunjukkan bahwa klasifikasi penyakit daun jagung menggunakan *Convolutional Neural Network* (CNN) menghasilkan akurasi sebesar 91.5% dan validasi sebesar 91% setelah proses *training* 10 *Epoch*.

Berdasarkan tabel 2.1 yang telah disajikan, dapat disimpulkan bahwa perbedaan utama antara penelitian sebelumnya dan penelitian yang akan dilakukan oleh penelitian terletak pada *dataset* yang digunakan. *Dataset* yang digunakan pada penelitian ini berisi citra daun yang terbagi menjadi empat kelas yaitu *Common Rust*, *Gray Leaf Spot*, *Healthy*, dan *Northern Leaf Blight* dengan total *dataset* sebanyak 3852 images. Penelitian yang akan diteliti mencakup tentang penggunaan *Convolutional Neural Network* (CNN) dalam mengidentifikasi dan mengklasifikasikan penyakit pada daun jagung dengan hasil yang akurat.

2.2 Landasan Teori

Dalam penelitian "Sistem Deteksi Penyakit Tanaman Jagung Menggunakan Metode *Convolutional Neural Network* Berbasis Web", landasan teori memegang peran krusial dalam memberikan pondasi yang kuat bagi pengembangan Metode. Landasan teori membantu peneliti untuk memahami konsep-konsep dasar yang terkait dengan teknologi terbaru dalam pengolahan citra, khususnya dalam konteks identifikasi penyakit tanaman. Dengan memahami dasar-dasar tersebut, peneliti dapat merancang dan mengimplementasikan Metode *Convolutional Neural Network* (CNN) dengan lebih efektif, memastikan bahwa sistem deteksi penyakit tanaman jagung yang dikembangkan memiliki kualitas dan akurasi yang tinggi.

2.2.1 Jagung

Tanaman jagung, yang merupakan tanaman pangan utama urutan ketiga setelah padi dan terigu di dunia serta menempati posisi kedua setelah padi di Indonesia, merupakan salah satu tanaman yang populer. Hal ini menjadikan jagung penting untuk diperhatikan terkait penyakit yang menyerang daunnya karena gejala penyakit dapat mengindikasikan adanya infeksi pada tanaman jagung dan membantu petani untuk mengambil langkah-langkah pencegahan sejak dini agar menghindari

kerusakan yang lebih lanjut pada tanaman dan meningkatkan hasil produksi[10].

Jagung menjadi salah satu tanaman yang sangat rentan terhadap penyakit, baik itu disebabkan oleh bakteri, jamur, atau virus. Gejala penyakit pada tanaman jagung biasanya mempengaruhi daun, yang dapat dilihat dari perubahan warna dan bentuknya[11]. Berikut merupakan *dataset* pada daun jagung yang digunakan pada penelitian ini :

Citra	Parameter	Keterangan
 Gambar 2.1 Citra *Common Rust*	1) Citra Daun: Gambar daun yang terinfeksi jamur *Puccinia sorghi*. 2) Warna Bercak: Oranye hingga coklat. 3) Lokasi Bercak: Daun dan batang tanaman jagung. 4) Bentuk Bercak: Kecil, bulat atau oval. 5) Ukuran Bercak: Beberapa milimeter hingga sentimeter. 6) Kondisi Daun: Daun menguning, kering, dan terjatuh pada infeksi parah.	*Common rust* adalah penyakit tanaman yang disebabkan oleh jamur *Puccinia sorghi*. Penyakit ini merupakan salah satu penyakit penting pada tanaman jagung dan dapat menyebabkan kerugian ekonomi yang signifikan bagi petani. Gejala yang muncul biasanya berupa bercak-bercak kecil berwarna oranye hingga coklat pada daun dan batang tanaman jagung. Infeksi yang parah dapat menyebabkan daun menguning, kering, dan terjatuh, yang pada akhirnya dapat mengurangi kualitas dan kuantitas hasil panen jagung.

		Pengendalian penyakit ini menjadi krusial dalam budidaya jagung untuk menjaga produktivitas dan keberlanjutan usaha tani. Metode pengendalian yang umum dilakukan meliputi penggunaan varietas jagung yang tahan terhadap penyakit, penggunaan sumber benih yang bebas dari infeksi, dan penerapan praktik pertanian yang baik untuk meminimalkan kelembaban dan penyebaran spora jamur[11].
 Gambar 2.2 Citra *Gray Leaf Spot*	1) Citra Daun: Gambar daun yang terinfeksi jamur *Cercospora zeae-maydis.* 2) Warna Bercak: Abu-abu atau silver, tepi coklat. 3) Lokasi Bercak: Daun jagung. 4) Bentuk Lesi: Lesi besar dengan tepi coklat. 5) Ukuran Lesi: Lesi besar dengan panjang beberapa sentimeter.	*Gray leaf spot* adalah penyakit tanaman yang disebabkan oleh jamur *Cercospora zeae-maydis*. Penyakit ini sering menginfeksi tanaman jagung dan dapat menyebabkan kerugian yang signifikan pada hasil panen. Gejala utamanya adalah munculnya bercak-bercak berwarna abu-abu atau silver pada daun jagung, yang kemudian berkembang menjadi lesi besar dengan tepi coklat.

	6) Kondisi Daun: Daun mengering, mati, dan terjatuh pada infeksi parah.	Infeksi yang parah dapat menyebabkan daun mengering, mati, dan terjatuh, yang berpotensi mengurangi produktivitas dan kualitas tanaman jagung. Oleh karena itu, pengendalian penyakit ini merupakan bagian penting dalam manajemen hama dan penyakit pada budidaya jagung[12].
 Gambar 2.3 Citra *Healthy*	1) Citra Daun: Gambar daun jagung yang sehat. 2) Warna Daun: Hijau cerah. 3) Kondisi Daun: Tanpa bercak atau lesi. 4) Bentuk Daun: Bentuk normal tanpa deformasi. 5) Tekstur Daun: Halus dan tidak ada kerusakan. 6) Kondisi Batang: Batang kuat dan tegak.	Tanaman jagung yang sehat menunjukkan daun-daun yang berwarna hijau cerah, tanpa adanya bercak atau lesi yang mencolok. Pertumbuhan tanaman jagung yang sehat juga ditandai dengan batang yang kuat dan tegak serta akar yang berwarna putih dan beruas. Tanaman yang sehat cenderung memiliki produksi dan hasil panen yang optimal karena mampu menyerap nutrisi dengan baik dan mengalami pertumbuhan yang normal. Oleh karena itu, memahami karakteristik tanaman jagung

		yang sehat menjadi kunci dalam mendukung kesuksesan budidaya dan produksi jagung yang berkualitas[13].
 Gambar 2.4 Citra *Northern Leaf Blight*	*1)* Citra Daun: Gambar daun yang terinfeksi jamur *Exserohilum turcicum.* 2) Warna Bercak: Hijau keabu-abuan atau keputihan, berkembang menjadi coklat. 3) Lokasi Bercak: Daun jagung. 4) Bentuk Lesi: Lesi panjang berwarna coklat dengan tepi melengkung. 5) Ukuran Lesi: Lesi panjang dengan panjang beberapa sentimeter. 6) Kondisi Daun: Daun mengering dan permukaan fotosintesis berkurang.	*Northern Leaf Blight* adalah penyakit tanaman jagung yang disebabkan oleh jamur *Exserohilum turcicum*. Gejala awalnya adalah munculnya bercak-bercak hijau keabu-abuan atau keputihan pada daun, yang kemudian berkembang menjadi lesi panjang berwarna coklat dengan tepi yang sering kali melengkung. Penyakit ini dapat menyebar dengan cepat dalam kondisi yang lembap dan hangat, mengurangi luas permukaan daun yang berfungsi untuk fotosintesis, dan akhirnya menghambat pertumbuhan tanaman jagung serta menurunkan hasil panen. Pengendalian terhadap penyakit ini termasuk penggunaan varietas tanaman yang tahan, rotasi

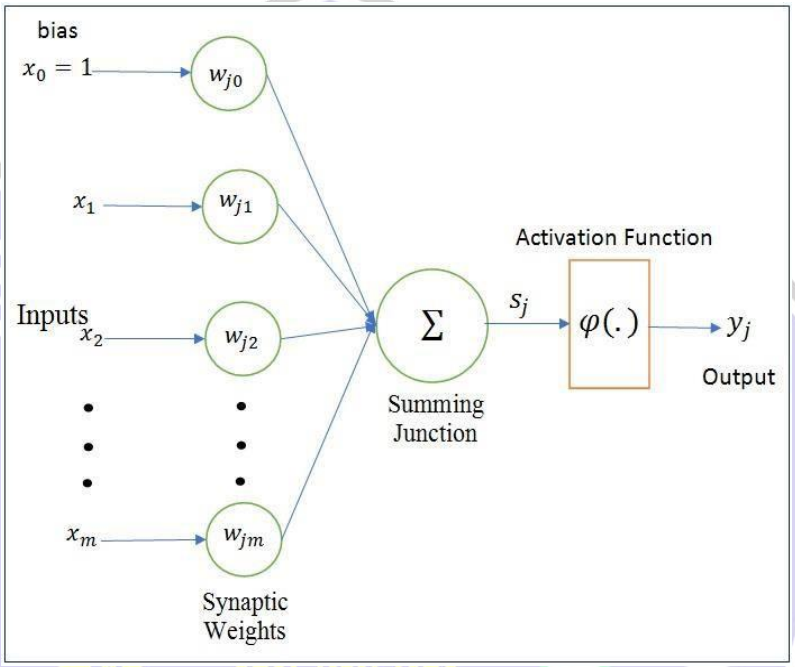
		tanaman, serta pengelolaan sisa tanaman yang terinfeksi untuk mengurangi penyebaran spora penyakit.

Namun, karena sulit melihat infeksi pada tanaman jagung secara langsung, petani perlu mengambil langkah-langkah awal untuk mendeteksinya sejak dini guna menghindari kesalahan dalam penanganan yang dapat mengurangi hasil panen dan meningkatkan risiko gagal panen. Penggunaan Jaringan Saraf Tiruan (*Neural Network*) menjadi sangat penting dalam mengklasifikasi daun jagung karena kemampuannya dalam mengolah informasi visual yang rumit pada gambar daun jagung dengan cepat dan tepat. Dengan *Neural Network*, model dapat mempelajari fitur-fitur penting pada gambar daun jagung, seperti bentuk, warna, dan tekstur daun, yang berperan penting dalam mengidentifikasi daun jagung ke dalam kategori yang berbeda berdasarkan ciri-cirinya.

2.2.2 *Neural Network*

Jaringan saraf, yang sering disebut sebagai *Neural Network* (NN), adalah sistem yang terdiri dari banyak unit pemrosesan sederhana yang disebut *neuron. Neuron-neuron* ini mirip dengan sel-sel saraf dalam otak manusia dan merupakan unit dasar dalam pengolahan informasi di dalam jaringan saraf. *Neural Network* telah digunakan secara luas dalam berbagai aplikasi seperti robotika, pengenalan suara, pengenalan wajah manusia, aplikasi medis, manufaktur, dan ekonomi. Struktur dasar dari jaringan saraf memungkinkan penggunaan yang fleksibel dalam merancang model-model yang sesuai dengan kebutuhan aplikasi tertentu.

Dengan kemampuannya untuk menyimpan pengetahuan dan membuatnya tersedia untuk digunakan, *Neural Network* telah menjadi alat yang sangat berguna dalam memecahkan berbagai masalah dalam berbagai bidang ilmu[12]. *Neuron* ini membentuk dasar untuk merancang keluarga jaringan saraf yang besar, berikut gambar 2.5 model *neuron* :



Gambar 2.4 model neuron non linear

Yang dapat dijelaskan pada persamaan 2.1 dan 2.2 berikut :

$$S_j = \sum_m^m = 0 W_{jm} X_m \qquad (2.1)$$

$$y_j = \varphi(sj) \qquad (2.2)$$

Di mana $x1$, $x2$, ..., xm adalah sinyal masukan; $x0$ = 1 adalah bias. $wj0$, $wj1$, ..., wjm adalah bobot sinaptik masing-masing neuron j. $\varphi(.)$ adalah fungsi aktiviti; and yj adalah sinyal keluaran dari neuron tersebut[12].

Secara umum, istilah "*Neural Network*" merujuk pada model matematika yang terdiri dari sejumlah *neuron* yang terhubung satu sama lain. CNN (*Convolutional Neural Network*) bisa dianggap sebagai variasi

khusus dari *Neural Network* yang menggunakan lapisan konvolusi untuk mengolah data masukan. Terkadang, CNN juga dapat berperan sebagai komponen dalam *Neural Network* yang lebih besar, membantu dalam pemrosesan data input yang lebih kompleks dan beragam[12].

2.2.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan salah satu metode dalam *machine learning* yang digunakan untuk memproses data dua dimensi. Kedalaman arsitektur CNN dan aplikasinya yang luas dalam pengolahan data gambar membuatnya menjadi jenis jaringan saraf yang kompleks. Berikut adalah rumus akurasi[13] :

$$Akurasi : \frac{Jumlah\ citra\ yang\ teridentifikasi\ dengan\ benar}{Total\ jumlah\ citra\ yang\ diuji}\ x\ 100\% \quad (2.3)$$

Keterangan :

- Jumlah citra yang teridentifikasi dengan benar yaitu jumlah gambar dalam dataset pengujian yang diklasifikasikan dengan benar oleh model.
- Total jumlah citra yang diuji yaitu total jumlah gambar dalam dataset pengujian.

Teknik CNN sering diterapkan dalam proses identifikasi dan pengenalan pada data gambar. CNN, sebuah kelas jaringan saraf buatan yang dominan dalam berbagai tugas penglihatan komputer, menarik minat di berbagai bidang, termasuk radiologi. Arsitektur CNN dirancang untuk secara otomatis dan adaptif mempelajari hierarki fitur spasial melalui proses backpropagation dengan menggunakan blok pembangun seperti lapisan konvolusi, lapisan pooling, dan lapisan terhubung penuh. Arsitektur CNN yang mendalam dapat mengalami kendala dalam proses pelatihan, seperti masalah gradien yang hilang atau meledak. Untuk mengatasi masalah ini, arsitektur seperti *Residual Network* (ResNet) memanfaatkan blok residual, memungkinkan pelatihan CNN menjadi lebih efisien dan efektif. Langkah-langkah dalam Memproses Data

Gambar Menggunakan *Convolutional Neural Network* (CNN) sebagai berikut :

1) *Preprocessing* Data

   Langkah awal dalam memproses data gambar menggunakan *Convolutional Neural Network* (CNN) adalah *preprocessing*. Pertama, kumpulkan dataset gambar yang relevan dengan tugas yang ingin diselesaikan. Kemudian, normalisasi nilai piksel gambar menjadi rentang tertentu (biasanya 0 hingga 1 atau -1 hingga 1). Lakukan augmentasi data seperti rotasi, flip, zoom, dan perubahan kecerahan untuk meningkatkan variasi data dan mencegah *overfitting*[4].

2) Membangun Arsitektur CNN

   Setelah *preprocessing*, langkah selanjutnya adalah membangun arsitektur CNN. Tentukan dimensi input yang sesuai dengan ukuran gambar (misalnya, 224x224x3 untuk gambar RGB). Tambahkan beberapa lapisan konvolusi untuk mengekstraksi fitur dari gambar, dengan setiap lapisan menggunakan kernel (filter) untuk mendeteksi pola seperti tepi, tekstur, dan bentuk. Terapkan fungsi aktivasi (biasanya ReLU) setelah setiap lapisan konvolusi untuk menambahkan non-linearitas. Tambahkan lapisan *pooling* (biasanya *Max Pooling*) setelah beberapa lapisan konvolusi untuk mengurangi dimensi peta fitur, memperkenalkan invariansi terhadap translasi, dan mengurangi jumlah parameter. Setelah beberapa lapisan konvolusi dan pooling, tambahkan lapisan terhubung penuh (*fully connected*) untuk memetakan fitur yang diekstraksi ke keluaran akhir jaringan, dan tentukan lapisan output sesuai dengan tugas (misalnya, fungsi *softmax* untuk klasifikasi multikelas)[4].

3) *Training*

Langkah berikutnya adalah pelatihan model. *Propagasi input* melalui jaringan untuk menghasilkan output (*forward propagation*). Hitung nilai kerugian (*loss*) menggunakan fungsi kerugian yang sesuai (misalnya, *categorical cross-entropy* untuk klasifikasi multikelas). Hitung gradien dari kerugian terhadap parameter jaringan dan perbarui parameter menggunakan Metode optimasi seperti *Gradient Descent* atau Adam *(backward propagation*). Jalankan proses pelatihan dalam beberapa *Epoch* dan gunakan *mini-batch* untuk mempercepat pelatihan dan meningkatkan generalisasi[4].

4) *Validation*

Selama pelatihan, lakukan validasi untuk memonitor kinerja model. Bagi dataset menjadi set pelatihan, validasi, dan uji. Evaluasi kinerja model pada set validasi setelah setiap *Epoch* untuk memantau overfitting atau underfitting. Sesuaikan hiperparameter seperti ukuran kernel, jumlah lapisan, *learning rate*, dan ukuran *batch* untuk mengoptimalkan kinerja model[4].

5) *Testing*

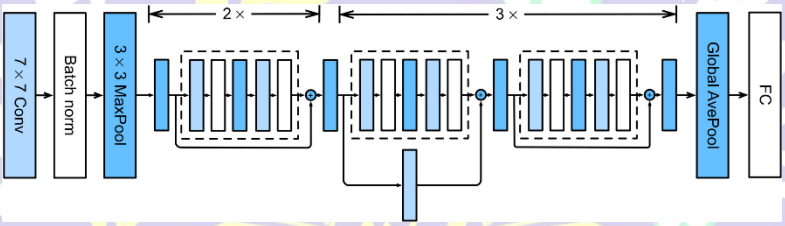
Setelah pelatihan selesai, uji model pada set uji untuk mengukur kinerja model pada data yang belum pernah dilihat sebelumnya. Gunakan metrik evaluasi seperti akurasi, *precision*, *recall*, dan F1-*score* untuk menilai kinerja model[4].

6) *Deployment*

Langkah terakhir adalah penyebaran model proses simpan model terlatih dalam format yang sesuai (misalnya, HDF5 atau *SavedModel*). Implementasikan model dalam aplikasi atau sistem produksi untuk melakukan prediksi pada data gambar baru. Monitor kinerja model secara berkala dan lakukan *retraining* jika diperlukan untuk menjaga akurasi dan keandalannya[4].

2.2.4 *ResNet*

ResNet adalah sebuah arsitektur *Convolutional Neural Network* (CNN) yang memperkenalkan konsep pembelajaran residual pada tahun 2015. Arsitektur ini dikenal memiliki lapisan yang sangat dalam dan telah memberikan peningkatan signifikan dalam akurasi pada berbagai tantangan *benchmarking* visi komputer. *ResNet* bahkan berhasil memenangkan kompetisi *ImageNet Large Scale Visual Recognition Challenge* 2015 (ILSVRC, 2015) dan *Microsoft Common Objects in Context* 2015 (MS COCO, 2015). Model *ResNet* dilatih menggunakan 1,28 juta gambar latihan yang terdiri dari 1000 kelas dan mencapai rata-rata kesalahan top-5 sebesar 5,25%[14].



Gambar 2.5 Lapisan *ResNet-18*

ResNet memiliki 4 lapisan konvolusional dalam setiap modul, tidak termasuk lapisan konvolusional 1 x 1. Ditambah dengan lapisan konvolusional pertama 7 x 7 dan lapisan terhubung penuh akhir, total ada 18 lapisan pada model ini, sehingga dikenal sebagai *ResNet*-18. Dengan memvariasikan jumlah channel dan blok residual di setiap modul, berbagai model *ResNet* dapat dibuat, misalnya *ResNet*-152 yang lebih dalam dengan 152 lapisan. Kombinasi ini telah membuat *ResNet* digunakan secara luas dan efektif dalam berbagai aplikasi.

2.2.5 *Pyhton*

Python adalah salah satu bahasa pemrograman paling terkenal dalam industri perangkat lunak dan pengembangan aplikasi. Diluncurkan pada tahun 1990 oleh Guido van Rossum di Belanda, Python awalnya dikembangkan sebagai hobi namun segera menjadi favorit berkat

sintaksnya yang sederhana dan mudah dipahami serta berbagai pustaka yang melimpah. *Python* digunakan luas dalam pengembangan aplikasi desktop, web, dan sistem, dan juga mendominasi dalam kecerdasan buatan (AI) dan *machine learning* dengan *TensorFlow* dan *PyTorch*. Dalam analisis data, *Python* populer berkat *Pandas* dan *NumPy* untuk manipulasi data yang kuat. Selain itu, *Python* digunakan untuk otomatisasi tugas-tugas rutin dan *prototyping* cepat, menunjukkan fleksibilitasnya dalam berbagai konteks pengembangan perangkat lunak modern. Dengan kombinasi keunggulan ini, *Python* menjadi pilihan utama bagi pengembang yang mencari produktivitas, kejelasan kode, dan kemampuan untuk menangani proyek-proyek kompleks di berbagai bidang[15].

Keberhasilan *Python* telah membuatnya menjadi bahasa yang sering dipelajari oleh mahasiswa di berbagai kampus IT. Mahasiswa menggunakan *Python* untuk menyelesaikan tugas kuliah, proyek akhir, dan penelitiannya. Pemahaman yang kuat tentang konsep dasar algoritma sangat penting dalam mempelajari *Python* atau bahasa pemrograman lainnya, karena pada dasarnya, pemrograman komputer adalah tentang menerapkan algoritma. Oleh karena itu, memahami konsep algoritma menjadi kunci keberhasilan bagi siapa pun yang ingin menjadi pengembang perangkat lunak yang kompeten. *Python* adalah bahasa pemrograman yang terkenal dengan sintaksnya yang mudah dipahami, yang membuatnya menjadi pilihan ideal untuk pembuatan program identifikasi jenis penyakit. Berbagai *library* populer seperti *TensorFlow*, *PyTorch*, dan Keras menyediakan alat yang efisien dan mudah digunakan untuk membangun model jaringan saraf tiruan. Dengan bantuan library-library ini, pengembang dapat dengan cepat mengimplementasikan dan menguji berbagai arsitektur jaringan saraf untuk aplikasi pengenalan penyakit tanaman[16].

2.2.6 *TensorFlow*

TensorFlow adalah sebuah platform sumber terbuka yang menyediakan solusi lengkap untuk pembelajaran mesin, mulai dari awal hingga akhir. *Platform* ini menawarkan ekosistem yang luas dan fleksibel, terdiri dari berbagai alat, perpustakaan, dan sumber daya komunitas. Hal ini memungkinkan para peneliti untuk mengembangkan teknologi pembelajaran mesin terbaru dan para pengembang untuk membuat dan menerapkan aplikasi berbasis pembelajaran mesin dengan mudah. Dengan menggunakan *TensorFlow*, pembuat model dapat dengan cepat membangun model identifikasi penyakit yang andal dan efisien[17].

2.2.7 *Flask*

Flask adalah sebuah *mikro-framework* untuk pengembangan web yang ditulis dalam bahasa pemrograman *Python*. *Framework* ini digunakan sebagai kerangka kerja untuk membuat aplikasi dan tampilan web. Dengan menggunakan *Flask* dan *Python*, pengembang dapat membuat struktur web yang terstruktur dan mengatur perilaku situs web dengan lebih efektif. Penggunaan *Flask* dalam pengujian model bertujuan untuk menyederhanakan proses pengujian dan menyediakan antarmuka yang mudah digunakan bagi pengguna. Melalui *Flask*, pengguna dapat mengintegrasikan model mereka ke dalam layanan web yang dapat diakses melalui API. Dalam pengujian model, *Flask* dapat membantu pengguna dalam mempersiapkan data uji, menjalankan model, dan menampilkan hasil pengujian dengan sederhana dan efisien[18].

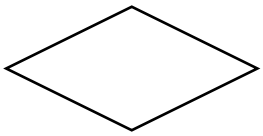
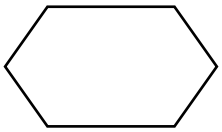
2.2.8 *Flowchart*

Flowchart atau sering disebut dengan diagram alir merupakan suatu jenis diagram yang merepresentasikan algoritma atau langkah-

langkah instruksi yang berurutan dalam sistem seorang analis sistem menggunakan *flowchart* sebagai bukti dokumentasi untuk menjelaskan gambaran logis sebuah sistem yang akan dibangun kepada programmer. Dengan begitu, *flowchart* dapat membantu untuk memberikan solusi terhadap masalah yang bisa saja terjadi dalam membangun sistem. Pada dasarnya, *flowchart* digambarkan dengan menggunakan simbol-simbol. Setiap simbol mewakili suatu proses tertentu. Sedangkan untuk menghubungkan satu proses ke proses selanjutnya digambarkan dengan menggunakan garis penghubung[19].

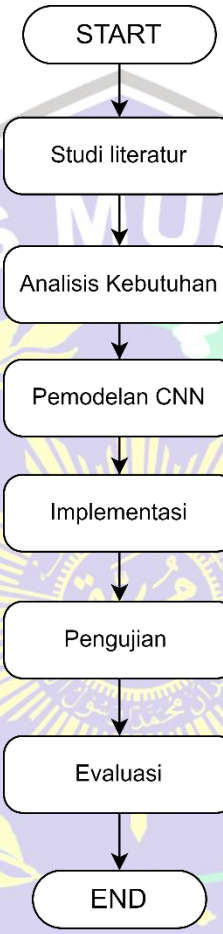
Tabel 2.2 Komponen *Flowchart*

Simbol	Nama	Fungsi
	*Terminator*	Menandai awal dan akhir dari suatu alur kerja atau proses
	Garis Alir *(Flow Line)*	Arah aliran atau urutan dari suatu proses dalam *Flowchart*
	*Input/Output* Data	Merepresentasikan masukan atau keluaran dari suatu alur kerja atau proses
	*Process*	Proses hitungan atau pengolahan data

	*Decision*	Menunjukkan suatu titik dalam proses dimana keputusan harus dibuat
	*Preparation*	Menunjukkan persiapan atau persiapan awal sebelum memulai suatu proses
	*Predefined Process* (Sub Program)	Menunjukkan suatu proses yang sudah ditentukan sebelumnya
	*OnPage Connector*	Menghubungkan dua atau lebih bagian *flowchart* yang terdapat pada halaman yang sama
	*OffPage Connector*	Menghubungkan bagian *flowchart* yang terdapat pada halaman yang berbeda

BAB 3
METODE PENELITIAN

3.1 Tahapan Penelitian



Gambar 3.1 Tahapan Penelitian

Penelitian ini merupakan sebuah studi tentang perbandingan arsitektur resnet152v2 dan alexnet dalam mendeteksi penyakit tanaman jagung menggunakan metode *convolutional neural network*. Tahapan penelitian yang dilakukan meliputi studi literatur, analisis, implementasi, dan pengujian. Studi literatur dilakukan untuk memahami konsep dasar identifikasi jenis penyakit dan Metode CNN. Selanjutnya, analisis dilakukan untuk menentukan parameter yang dibutuhkan dalam implementasi Metode CNN pada citra daun jagung.

Pemodelan bertujuan sebagai perancangan model CNN dengan arsitektur ResNet. Implementasi dilakukan dengan mengembangkan model identifikasi jenis penyakit daun jagung menggunakan Metode CNN dan dilakukan pada sebuah platform pengembangan perangkat lunak. Pengujian dilakukan dengan menggunakan dataset citra daun jagung dan menguji model yang telah dikembangkan. Hasil pengujian menunjukkan bahwa model identifikasi jenis penyakit daun jagung menggunakan Metode CNN memiliki performa yang baik.

3.1.1 Studi literatur

Studi literatur dalam penelitian melibatkan proses mengumpulkan, mengevaluasi, dan mengintegrasikan berbagai informasi yang relevan dengan topik penelitian. Sumber informasi yang digunakan termasuk jurnal, buku, makalah konferensi, dan sumber lain yang terkait dengan penggunaan Metode CNN dengan arsitektur ResNet untuk identifikasi jenis penyakit daun jagung. Tujuannya adalah untuk mendapatkan pemahaman yang komprehensif tentang topik penelitian, mengidentifikasi kekurangan pengetahuan dan kesenjangan dalam penelitian sebelumnya, serta memperoleh informasi yang diperlukan untuk merancang eksperimen yang efektif dan mengembangkan hipotesis yang tepat. Hasil dari studi literatur dapat digunakan untuk membentuk kerangka teori penelitian, memilih metode dan teknik yang sesuai, dan juga membantu dalam interpretasi hasil.

3.1.2 Analisis Kebutuhan

Analisis dalam penelitian melibatkan proses penyusunan, pemeriksaan, dan penafsiran data yang telah dikumpulkan selama eksperimen. Proses ini mencakup langkah-langkah pengecekan kualitas data serta penerapan teknik statistik atau pengolahan citra untuk mengidentifikasi pola atau hubungan antara variabel yang sedang diteliti. Melalui analisis data, hipotesis diuji, konsistensi dengan kerangka teori dievaluasi, dan kesimpulan tentang signifikansi hasil penelitian diambil.

1. *Dataset*

   Penelitian ini memanfaatkan sebuah *Dataset* yang tersedia di Kaggle yang dimiliki Nafisha Moin. *Dataset* ini berisi citra daun yang terbagi

menjadi empat kelas yaitu *Common Rust, Gray Leaf Spot, Healthy*, dan *Northern Leaf Blight* dengan total *dataset* sebanyak 3852 *images*. Data ini akan digunakan sebagai dasar untuk sistem deteksi penyakit pada daun jagung. Metode identifikasi penyakit pada daun jagung dapat mempercepat dan meningkatkan akurasi diagnosis. Proses ini melibatkan klasifikasi citra berdasarkan ciri-ciri yang dimilikinya.

Berikut citra yang digunakan pada Gambar 3.2 sampai Gambar 3.5 sebagai berikut :

Tabel 3.1 Citra Daun Jagung

Citra	Parameter	Keterangan
 Gambar 3.2 Citra *Common Rust*	1) Citra Daun: Gambar daun yang terinfeksi jamur *Puccinia sorghi*. 2) Warna Bercak: Oranye hingga coklat. 3) Lokasi Bercak: Daun dan batang tanaman jagung. 4) Bentuk Bercak: Kecil, bulat atau oval. 5) Ukuran Bercak: Beberapa milimeter hingga sentimeter. 6) Kondisi Daun: Daun menguning,	*Common rust* adalah penyakit tanaman yang disebabkan oleh jamur *Puccinia sorghi*. Penyakit ini merupakan salah satu penyakit penting pada tanaman jagung dan dapat menyebabkan kerugian ekonomi yang signifikan bagi petani. Gejala yang muncul biasanya berupa bercak-bercak kecil berwarna oranye hingga coklat pada daun dan batang tanaman jagung.

	kering, dan terjatuh pada infeksi parah.	
 Gambar 3.3 Citra *Gray Leaf Spot*	1) Citra Daun: Gambar daun yang terinfeksi jamur *Cercospora zeae-maydis*. 2) Warna Bercak: Abu-abu atau silver, tepi coklat. 3) Lokasi Bercak: Daun jagung. 4) Bentuk Lesi: Lesi besar dengan tepi coklat. 5) Ukuran Lesi: Lesi besar dengan panjang beberapa sentimeter. 6) Kondisi Daun: Daun mengering, mati, dan terjatuh pada infeksi parah.	Penyakit *Gray Leaf Spot* pada tanaman jagung disebabkan oleh jamur *Cercospora zeae-maydis* dan ditandai dengan adanya bercak-bercak abu-abu kecil atau besar pada daun, yang kemudian berkembang menjadi cincin berwarna gelap dengan tepi merah kecokelatan.

 Gambar 3.4 Citra *Healthy*	1) Citra Daun: Gambar daun jagung yang sehat. 2) Warna Daun: Hijau cerah. 3) Kondisi Daun: Tanpa bercak atau lesi. 4) Bentuk Daun: Bentuk normal tanpa deformasi. 5) Tekstur Daun: Halus dan tidak ada kerusakan. 6) Kondisi Batang: Batang kuat dan tegak.	*Healthy* adala kondisi di mana tanaman jagung tidak menunjukkan gejala penyakit atau gangguan kesehatan tertentu. Daun-daunnya tampak normal tanpa adanya bercak, lesi, atau perubahan warna yang tidak biasa yang menandakan adanya infeksi atau kerusakan. Tanaman jagung yang sehat cenderung memiliki pertumbuhan yang baik, warna daun yang hijau, dan tidak menunjukkan tanda-tanda stres atau kekurangan nutrisi.
 Gambar 3.5 Citra *Northern Leaf Blight*	1) Citra Daun: Gambar daun yang terinfeksi jamur *Exserohilum turcicum.* 2) Warna Bercak: Hijau keabu-abuan atau keputihan,	*Northern Leaf Blight* penyakit yang disebabkan oleh jamur *Exserohilum turcicum*, memunculkan lesi cokelat panjang pada daun, khususnya di bagian tengah dan bawah tanaman, yang

	berkembang menjadi coklat. 3) Lokasi Bercak: Daun jagung. 4) Bentuk Lesi: Lesi panjang berwarna coklat dengan tepi melengkung. 5) Ukuran Lesi: Lesi panjang dengan panjang beberapa sentimeter. 6) Kondisi Daun: Daun mengering dan permukaan fotosintesis berkurang.	dapat membesar dan menyebabkan kekuningan pada daun yang terinfeksi.

2. *Treatment*

Dalam perawatan daun pada tanaman jagung, *treatment* memiliki peran yang sangat penting dalam memastikan pertumbuhan yang sehat dan produktivitas yang optimal. Melalui *treatment*, kita dapat mengendalikan serangan hama dan penyakit yang berpotensi merusak daun serta mengganggu proses penting dalam fotosintesis tanaman jagung. *Treatment* juga memberikan kemungkinan untuk memberikan nutrisi yang tepat secara langsung ke daun, seperti pemupukan daun, yang membantu meningkatkan penyerapan nutrisi dan efisiensi dalam proses fotosintesis. Berikut *treatment* yang diperlukan dalam beberapa penyakit pada daun jagung:

Tabel 3.2 Citra Daun Jagung

No.	Nama Penyakit	*Treatment*
1.	*Common rust*	*Treatment* untuk penyakit *common rust* dapat dilakukan dengan penggunaan fungisida yang sesuai seperti *triazole* atau *strobilurin*, sesuai dengan rekomendasi dosis yang diberikan oleh produsen. Pemilihan waktu aplikasi yang tepat juga penting biasanya dilakukan saat gejala awal terlihat atau ketika kondisi lingkungan mendukung pertumbuhan penyakit. Selain itu, praktik pertanian yang baik seperti rotasi tanaman, pengelolaan sisa tanaman yang mati, dan penggunaan varietas jagung yang tahan terhadap karat umum juga dapat membantu mengurangi risiko infeksi.
2.	*Gray Leaf Spot*	*Treatment* untuk penyakit *gray leaf spot* dapat dilakukan dengan penggunaan fungisida dengan bahan aktif seperti *azoxystrobin* atau *propiconazole* dapat efektif. Pengaplikasian fungisida harus dilakukan sesuai dengan jadwal yang direkomendasikan dan dengan dosis yang tepat. Praktik manajemen tanaman yang baik, seperti pemantauan rutin, pemotongan daun yang terinfeksi, dan pengelolaan kelembaban tanaman, juga penting untuk mengurangi risiko penyebaran penyakit.
3.	*Healthy*	Tanaman jagung yang sehat tidak memerlukan *treatment* khusus terhadap

		penyakit. Namun, perawatan yang baik termasuk penyiraman yang cukup, pemupukan yang seimbang, dan pemantauan rutin kondisi tanaman untuk mendeteksi gejala penyakit atau gangguan lainnya yang mungkin timbul.
4.	*Northern Leaf Blight*	Treatment untuk penyakit *northen leaf blight* dapat dilakukan dengan pengaplikasian fungisida yang mengandung bahan aktif seperti *chlorothalonil* atau *azoxystrobin*. Penggunaan fungisida harus disesuaikan dengan jadwal aplikasi yang tepat dan dilakukan sejak awal gejala muncul. Selain itu, praktik manajemen tanaman yang baik seperti penghapusan sisa tanaman yang terinfeksi, rotasi tanaman, dan penggunaan varietas yang tahan terhadap penyakit ini juga dapat membantu mengurangi risiko infeksi.

3. Kebutuhan Perangkat Lunak

a) *Google Colab*

Platform berbasis *cloud* untuk menulis dan menjalankan kode *Python* dalam *notebook* interaktif. Berguna untuk mengembangkan, menguji, dan berbagi model machine learning dengan akses gratis ke GPU dan TPU, integrasi dengan *Google Drive*, serta mendukung kolaborasi tim.

b) *Visual Studio Code*

Editor kode sumber yang ringan namun kuat, mendukung berbagai bahasa pemrograman termasuk Python. Menyediakan ekstensi untuk *debug, linting,* dan integrasi Git, serta antarmuka yang dapat disesuaikan dan fitur kolaborasi langsung.

c) *Pyhton* 3.9

Versi Python yang diperlukan untuk mengembangkan model dan aplikasi menggunakan teknologi terbaru. Menawarkan sintaks yang lebih bersih, fitur-fitur baru untuk efisiensi pengkodean, dan dukungan untuk pustaka dan *framework* terbaru, termasuk Flask.

d) *Kaggle Notebook*

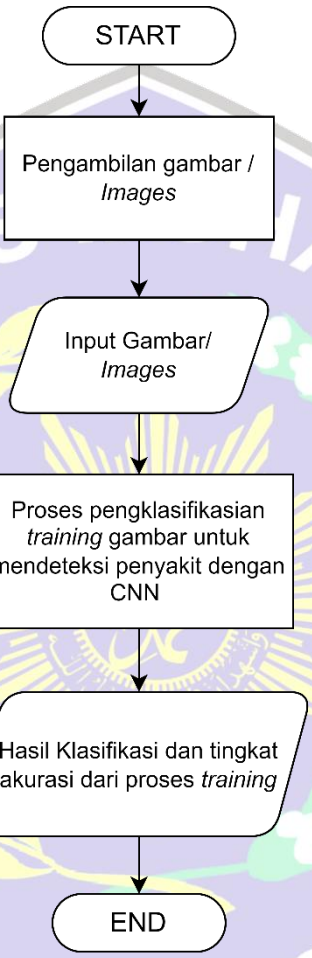
Platform online untuk pemrograman *Python,* khususnya dalam *data science* dan *machine learning*. Menyediakan akses ke berbagai *dataset*, fasilitas untuk menulis dan menjalankan kode dalam *notebook* interaktif, serta kemampuan berbagi proyek dan berkolaborasi dengan komunitas data *science* global.

Google Colab digunakan sebagai *platform* untuk membuat model dalam *notebook Kaggle*, sebuah *platform* pemrograman *Python* yang menyediakan lingkungan pelatihan menggunakan komputer *virtual*. *Kaggle* juga memberikan kemampuan untuk berbagi dan berkolaborasi dalam pengembangan model. Selain itu, untuk membuat sistem yang menggunakan *Flask*, sebuah *framework Python* berbasis web, *Python* perlu diinstal di lingkungan yang relevan. Hal ini diperlukan agar model yang telah diimplementasikan dan situs web yang dikembangkan menggunakan *Flask* dapat berjalan dengan lancar. Dengan menggunakan *Flask*, pengembang dapat dengan mudah membuat aplikasi web interaktif yang memungkinkan pengguna untuk berinteraksi dengan model secara langsung melalui antarmuka web.

Ini mempermudah distribusi dan penggunaan model dalam lingkungan praktis.

4. Perancangan Sistem

a) Flowchart Sistem

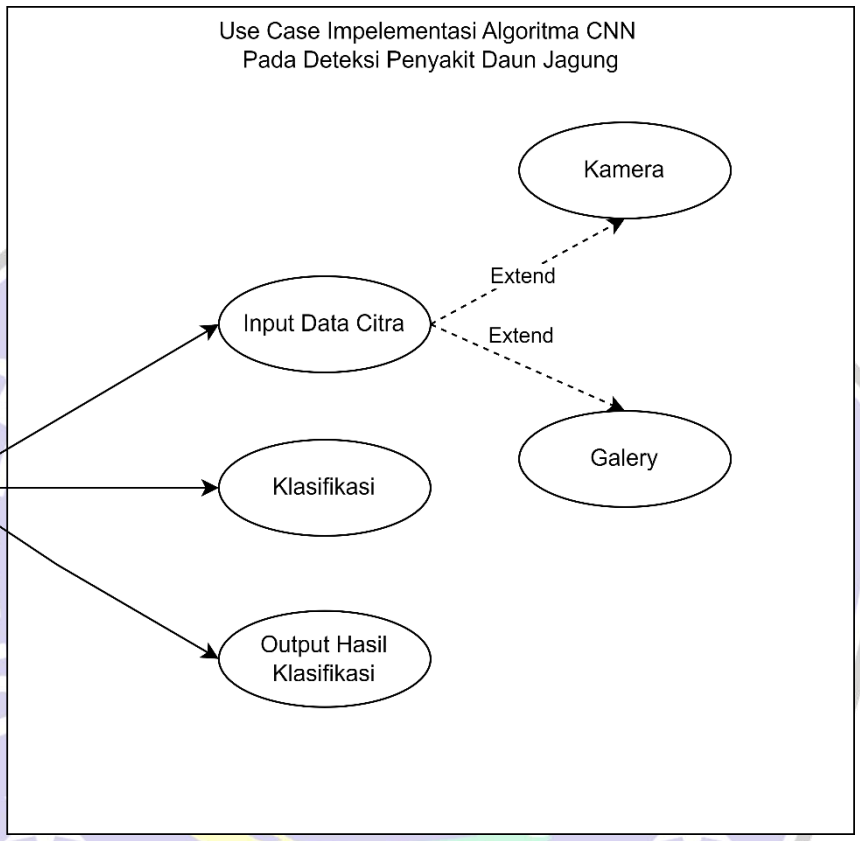


Gambar 3.6 *Flowchart* Sistem

Berdasarkan gambar 3.6, merupakan alur sistem deteksi penyakit pada daun jagung. Proses dimulai ketika pengguna mengambil gambar daun jagung melalui kamera atau galeri, dan gambar tersebut kemudian akan diidentifikasi oleh sistem. Setelah pengambilan gambar, pengguna mengunggah gambar tersebut ke sistem. Sistem merespons gambar yang diunggah dan melakukan proses klasifikasi menggunakan metode *Convolutional Neural*

Network (CNN) sesuai dengan gambar yang diuji untuk jenis penyakit pada daun jagung. Setelah proses pengujian selesai, hasilnya akan ditampilkan kepada pengguna, termasuk jenis penyakit yang terdeteksi dan akurasi data yang diperoleh.

b) Usecase Diagram



Gambar 3.7 *Usecase Diagram*

Pada gambar 3.7 terdapat use case diagram yang menampilkan interaksi antara pengguna, yang dalam hal ini hanya ada satu yaitu "user", dengan sistem deteksi penyakit pada daun jagung. User memiliki dua use case utama, yaitu "Input Data Citra" dan "Klasifikasi". Pertama, dalam use case "Input Data Citra", user memulai proses dengan menginputkan data citra daun jagung ke dalam sistem. Data citra tersebut dapat berasal dari berbagai sumber, seperti kamera atau galeri. Setelah data citra diinputkan, sistem akan melakukan proses klasifikasi untuk mengidentifikasi jenis penyakit

pada daun jagung. Kedua, dalam use case "Klasifikasi", setelah data citra diinputkan, sistem akan melakukan proses klasifikasi menggunakan metode yang telah ditentukan, seperti Convolutional Neural Network (CNN). Proses ini bertujuan untuk mengidentifikasi dan mengklasifikasikan jenis penyakit yang terdapat pada daun jagung berdasarkan citra yang diinputkan oleh pengguna.

Setelah proses klasifikasi selesai, sistem akan menghasilkan output berupa hasil dari klasifikasi tersebut. Output ini akan ditampilkan kepada pengguna, sehingga pengguna dapat melihat hasil identifikasi jenis penyakit pada daun jagung yang telah dilakukan oleh sistem. Dengan demikian, use case diagram ini mencakup seluruh langkah-langkah interaksi antara pengguna dan sistem deteksi penyakit pada daun jagung.

c) *User Interface*

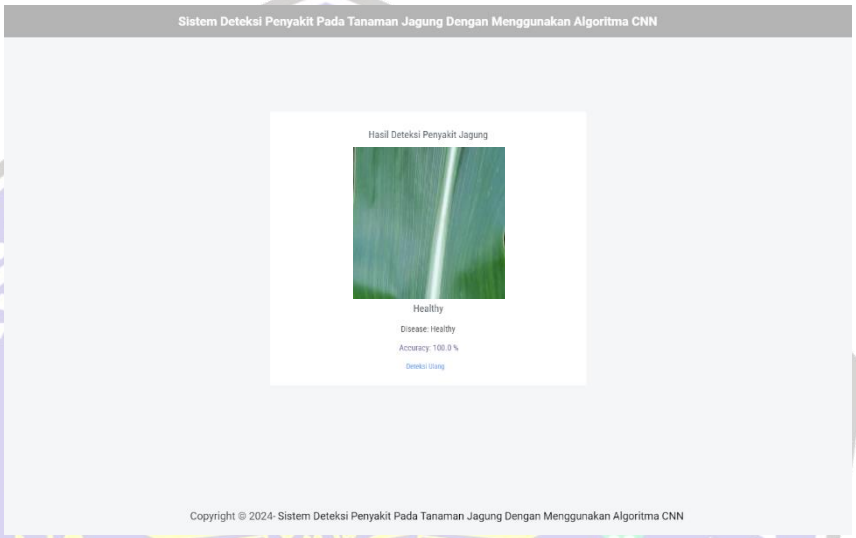
(1) *Input* Citra

Gambar 3.8 *Input* Citra

Gambar 3.8 menampilkan antarmuka *input* citra pada sistem deteksi penyakit pada daun jagung, di mana pengguna dapat mengunggah gambar yang akan diklasifikasikan. Antarmuka ini dirancang untuk memungkinkan pengguna dengan mudah mengunggah gambar tanpa mengalami kesulitan, serta memberikan

umpan balik yang jelas mengenai proses pengunggahan dan hasil klasifikasi yang diperoleh. Dengan antarmuka ini, pengguna dapat dengan cepat dan mudah melakukan klasifikasi penyakit pada daun jagung tanpa memerlukan pengetahuan teknis yang mendalam dalam penggunaan sistem deteksi penyakit daun jagung.

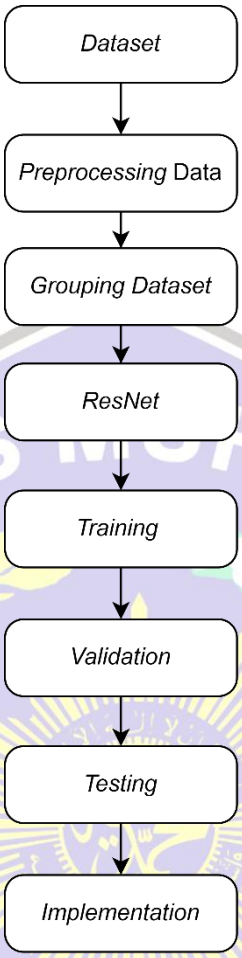
(2) Hasil Klasifikasi



Gambar 3.9 Hasil Klasifikasi

Gambar 3.9 menampilkan hasil klasifikasi dari sistem deteksi penyakit pada daun jagung. Hasil ini adalah hasil akhir dari proses identifikasi yang dilakukan oleh sistem terhadap citra daun jagung yang diunggah oleh pengguna. Dalam gambar tersebut, mungkin terdapat informasi tentang jenis penyakit yang terdeteksi pada daun jagung, serta tingkat kepercayaan atau akurasi klasifikasi yang diberikan oleh sistem. Informasi ini sangat penting bagi pengguna karena dapat memberikan gambaran tentang kondisi kesehatan tanaman jagung mereka, sehingga mereka dapat mengambil tindakan yang diperlukan untuk mengatasi penyakit tersebut.

3.1.3 Pemodelan CNN



Gambar 3.10 Pemodelan CNN

Berikut penjelasan tahapan pemodelan pada gambar 3.6, sebagai berikut :

1) *Dataset* : *Dataset* merupakan kumpulan gambar yang digunakan untuk melatih, menguji, dan memvalidasi model *Convolutional Neural Network* (CNN). Dalam dataset yang digunakan, perlu ada gambar-gambar yang mewakili semua kategori yang ingin dikenali oleh model.
2) *Preprocessing* : Sebelum dataset diolah oleh model, dilakukan pra-pemrosesan data. Proses ini dapat meliputi pemotongan citra, normalisasi, augmentasi data, dan segmentasi untuk mempersiapkan dataset dengan baik sebelum digunakan dalam pelatihan model.
3) *Grouping Dataset* : *dataset* dibagi menjadi dua bagian yaitu set latih dan set validasi. Set latih digunakan untuk melatih model, sedangkan

set validasi digunakan untuk mengevaluasi performa model selama proses pelatihan. Hal ini membantu memastikan bahwa model memiliki kemampuan yang baik untuk mengenali berbagai kelas gambar yang ada.

4) Membangun model ResNet : Model ResNet terdiri dari berbagai komponen seperti lapisan *konvolusi, Batch Normalization, Activation Layer, Maxpooling Layer, Residual Block*, dan *fully connected layer*.
5) *Training* : Proses pelatihan model, *dataset* latih digunakan sebagai panduan untuk menyesuaikan parameter model secara bertahap. Tujuannya adalah meningkatkan akurasi model seiring dengan berjalannya waktu.
6) *Validation* : Setelah tahap pelatihan, kinerja model diuji menggunakan dataset validasi untuk memastikan bahwa model dapat menggeneralisasi dengan baik ke data baru yang tidak pernah dilihat sebelumnya. Dataset validasi digunakan sebagai tolak ukur independen untuk mengevaluasi kemampuan model dalam mengeneralisasi pola dari data latih ke data yang belum pernah dilihat.
7) *Testing* : Testing adalah tahap terakhir dalam proses pembangunan model machine learning, di mana model yang telah dilatih dan divalidasi diuji dengan menggunakan data yang belum pernah dilihat sebelumnya, yang disebut sebagai data pengujian. Tujuan dari tahap ini adalah untuk mengukur kinerja model secara objektif di luar dataset yang digunakan selama pelatihan dan validasi, serta untuk mengevaluasi kemampuan model untuk menggeneralisasi ke data baru.
8) *Implementation* : Setelah berhasil diuji, model yang telah teruji dapat diimplementasikan dalam *framework Flask* untuk penggunaan praktis. Proses implementasi ini melibatkan integrasi model ke dalam aplikasi web yang dapat diakses secara publik atau privat. Dalam tahap ini, model diuji kembali menggunakan data baru yang mungkin lebih representatif dari situasi nyata.

3.1.4 Implementasi

Tahap implementasi merupakan langkah dalam proses pengembangan yang bertujuan untuk menerapkan atau menjalankan solusi yang telah dirancang atau dikembangkan. Dalam konteks pengembangan model *Convolutional Neural Network* (CNN) untuk identifikasi jenis penyakit pada daun jagung, tahap implementasi melibatkan integrasi model CNN ke dalam *platform web* menggunakan *Framework Flask*. Proses ini melibatkan penggunaan bahasa pemrograman *Python* untuk menghubungkan model CNN yang telah dilatih sebelumnya dengan aplikasi *Flask*. Selanjutnya, model tersebut diakses melalui antarmuka web yang telah dibangun menggunakan *Flask*. Tahap implementasi ini memungkinkan pengguna untuk dengan mudah menggunakan model untuk mengidentifikasi jenis penyakit pada daun jagung melalui aplikasi web yang telah disediakan.

3.1.5 Pengujian

Pada tahap pengujian, citra-citra yang akan diidentifikasi penyakitnya dimasukkan ke dalam aplikasi web Flask yang telah diimplementasikan dengan model Convolutional Neural Network (CNN) yang telah dilatih sebelumnya. Proses identifikasi jenis penyakit pada setiap citra dilakukan oleh model CNN, dan hasil identifikasi kemudian dievaluasi untuk mengukur akurasi model. Proses pengujian bertujuan untuk mengevaluasi seberapa baik model CNN dapat mengidentifikasi citra daun jagung menjadi kategori penyakit yang tepat. Hasil pengujian akan menjadi dasar untuk menilai performa model dan untuk memutuskan apakah model tersebut dapat digunakan secara praktis atau memerlukan penyempurnaan lebih lanjut.

Untuk menghitung akurasi, label sebenarnya dari setiap citra akan dibandingkan dengan label yang dihasilkan oleh model CNN. Akurasi dihitung dengan membagi jumlah citra yang teridentifikasi dengan benar oleh model dengan jumlah total citra yang diuji. Proses pengujian

dilakukan untuk mengevaluasi seberapa baik model CNN dapat mengidentifikasi citra daun jagung menjadi kategori penyakit yang tepat.

Hasil pengujian akan menjadi dasar untuk mengevaluasi performa model dan untuk memutuskan apakah model sudah siap untuk digunakan secara praktis atau memerlukan penyempurnaan lebih lanjut. Rumus akurasi yang digunakan adalah:

$$Akurasi : \frac{Jumlah\ citra\ yang\ teridentifikasi\ dengan\ benar}{Total\ jumlah\ citra\ yang\ diuji}\ x\ 100\% \quad (2.3)$$

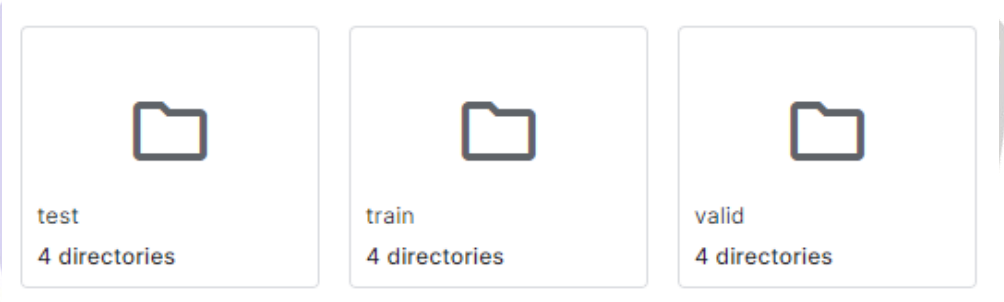
3.1.6 Evaluasi

Evaluasi sistem deteksi penyakit tanaman jagung berbasis CNN dilakukan untuk menentukan kelayakan sistem sebelum dideploy kepada petani. Evaluasi melibatkan pengujian akurasi deteksi pada empat dataset yang berbeda secara berulang untuk mencatat hasil terbaik dari setiap iterasi. Standar akurasi tertinggi telah ditetapkan sebelumnya, dan hasil terbaik dari seluruh pengujian akan dibandingkan untuk menilai apakah sistem memenuhi standar yang telah ditetapkan. Selain fokus pada akurasi, evaluasi juga mempertimbangkan kehandalan sistem, kecepatan respons, dan kemudahan penggunaan agar dapat memberikan manfaat optimal bagi para petani di lapangan.

BAB 4
HASIL DAN PEMBAHASAN

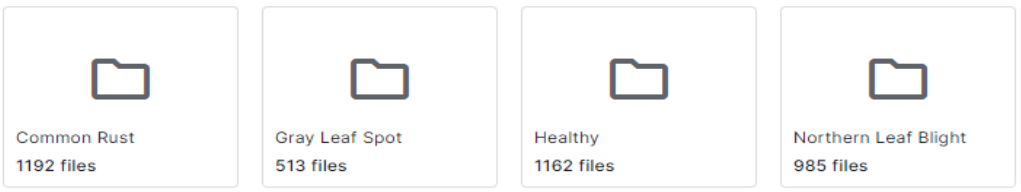
Hasil penelitian ini bertujuan dalam mengidentifikasi berbagai jenis penyakit daun jagung, sehingga memfasilitasi petani dan ahli pertanian untuk dengan segera mengambil langkah-langkah pencegahan yang efektif guna mengontrol dan mengurangi efek negatif yang ditimbulkan oleh penyakit tersebut. Selain itu, temuan dan analisis dalam penelitian ini juga dapat menjadi dasar untuk mengembangkan sistem pendukung keputusan yang efisien dan berkelanjutan dalam manajemen penyakit tanaman jagung.

4.1 Persiapan



Gambar 4.1 Folder Dataset

Berdasarkan gambar 4.1 Folder Dataset, terdapat 3 kelompok: test, train, dan valid. Dataset train adalah dataset yang dipakai untuk melatih model, di mana citra-citra dalam dataset ini digunakan untuk menyempurnakan parameter model melalui algoritma pembelajaran mesin. Dataset validasi (valid) digunakan untuk menilai kinerja model selama proses pelatihan dan membantu dalam memilih model yang paling optimal.



Gambar 4.2 *Dataset*

Berdasarkan Gambar 4.3 folder valid, folder tersebut berisi kumpulan citra daun jagung yang juga berfungsi sebagai kelas citra saat pembuatan model, sesuai dengan yang didefinisikan dalam tabel jenis-jenis penyakit. Dalam pengelompokan citra, terdapat modifikasi dataset di mana citra dipisahkan sesuai dengan kelompok penyakit dan ada penambahan dataset. Selain itu, terdapat folder test dan valid yang memiliki kelas yang sama, dengan keterangan jumlah citra yang tercantum dalam tabel 4.1 dataset berikut:

Tabel 4.1 *Dataset*

Kelas	test	train	valid
Corn__Common_rust	10	1192	325
Corn__Gray_leaf_spot	10	513	300
Corn__Healthy	10	1162	363
Corn__Northern_leaf_blight	10	985	340

Berdasarkan tabel 2.1 *dataset* merupakan banyak jumlah citra yang terkumpul untuk proses pelatihan, validasi dan data uji yang digunakan untuk proses pelatihan model untuk mendapatkan data yang valid.

4.2 Pemodelan CNN

Dalam pengembangan sistem identifikasi jenis penyakit tanaman, penting untuk merancang model arsitektur yang efektif. Salah satu opsi yang layak dipertimbangkan adalah penggunaan model ResNet-152V2. Model ini memiliki 152 lapisan, yang memungkinkan analisis mendalam terhadap citra. Selain itu, ResNet-152V2 telah dilatih menggunakan dataset yang besar dan beragam, sehingga memiliki kemampuan luar biasa dalam mengenali dan membedakan fitur-fitur kompleks dalam citra dengan tingkat akurasi yang sangat tinggi. Keandalan dan ketepatan model ini menjadikannya pilihan yang ideal untuk tugas-tugas klasifikasi yang menuntut detail dan presisi tinggi.

4.2.1 *Importing Library*

```
from flask import Flask, render_template, request, redirect, url_for
import tensorflow as tf
import numpy as np
import os
from tensorflow.keras.preprocessing import image
import cv2
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg')

# Load the trained model
model = tf.keras.models.load_model('corn_disease_cnn_model.h5')

# Define class labels
class_labels = ['Common Rust', 'Gray Leaf Spot', 'Healthy', 'Northern Leaf Blight']

# Create Flask app
app = Flask(__name__)

# Set upload folder
UPLOAD_FOLDER = 'static/uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Load and preprocess image
def load_and_preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0
    return img_array
```

Gambar 4.3 *Import Library*

Berdasarkan gambar 4.3 *import library*, terdapat pengimporan beberapa *library* yang digunakan dalam pemrograman dengan *TensorFlow* untuk tugas pembelajaran mesin. *Library-library* tersebut antara lain *Flask* yang digunakan untuk membangun aplikasi web, *TensorFlow* untuk membangun dan melatih model pembelajaran mesin, *NumPy* untuk operasi numerik dan manipulasi *array*, os untuk operasi sistem seperti mengatur jalur direktori, *OpenCV* (cv2) untuk pemrosesan gambar dan video, serta *matplotlib* untuk visualisasi data. Selain itu, juga diimpor fungsi dari *tensorflow*.keras.*preprocessing* untuk pra-pemrosesan gambar. Kode tersebut mencakup komponen-komponen berikut yang diperlukan dalam pengembangan model dengan *TensorFlow*: memuat model yang telah dilatih menggunakan tf.keras.models.*load*_model, mendefinisikan label kelas yang digunakan untuk klasifikasi penyakit daun jagung, membuat aplikasi *Flask* untuk menangani antarmuka web, mengatur folder upload untuk lokasi di mana citra yang diunggah akan disimpan, dan memuat serta

memproses gambar dengan fungsi *load_and_preprocess_image* yang digunakan untuk memuat dan mempersiapkan gambar agar sesuai untuk diprediksi oleh model. Dengan demikian, kode ini menyediakan semua komponen yang diperlukan untuk pemrosesan data, konstruksi model, pelatihan, evaluasi, dan pengolahan citra dalam pengembangan model pembelajaran mesin menggunakan *TensorFlow*.

4.2.2 Definisi ukuran citra

```
IMG_WIDTH, IMG_HEIGHT = 224, 224
EPOCHS = 50  # Reduced epochs for demonstration
BATCH_SIZE = 32  # Adjusted batch size
IMAGE_SIZE = 256
CHANNELS = 3
```

Gambar 4.4 Definisi ukuran citra

Berdasarkan gambar 4.4 definisi ukuran citra, terdapat beberapa konstanta penting. Konstanta pertama, `*IMG_WIDTH*` dan `*IMG_HEIGHT*`, digunakan untuk menentukan lebar dan tinggi gambar yang akan diproses masing-masing bernilai 224. Konstanta kedua, `*EPOCHS*`, menentukan jumlah iterasi pelatihan model yang dalam hal ini ditetapkan sebanyak 50 untuk keperluan demonstrasi. Konstanta ketiga, `*BATCH_SIZE*` menentukan jumlah sampel gambar yang akan diproses dalam satu iterasi (*batch*) selama proses pelatihan model. Nilai 32 menunjukkan bahwa 32 gambar akan diproses secara bersamaan dalam setiap iterasi. Konstanta keempat, `*IMAGE_SIZE*` digunakan untuk menentukan dimensi gambar yang akan diproses, dengan nilai 256 yang menunjukkan ukuran sisi gambar yang diubah menjadi persegi. Konstanta terakhir, `*CHANNELS*` menunjukkan jumlah saluran warna dalam gambar. Nilai 3 merepresentasikan penggunaan representasi warna RGB.

4.2.3 *Preprocessing*

```
# Load and preprocess image
def load_and_preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0
    return img_array

# Predict disease
def predict_disease(img_path):
    img_array = load_and_preprocess_image(img_path)
    predictions = model.predict(img_array)
    predicted_class = np.argmax(predictions)
    confidence = predictions[0][predicted_class]
    return class_labels[predicted_class], confidence
```

Gambar 4.5 *Preprocessing*

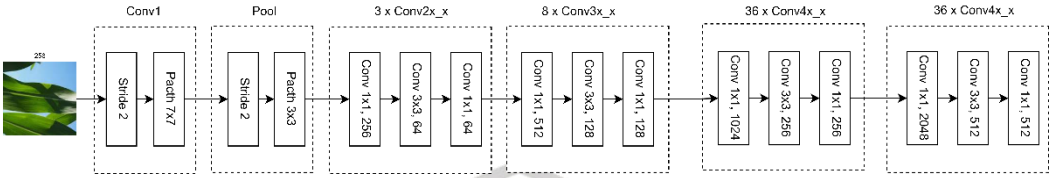
Berdasarkan gambar 4.5, proses dimulai dengan fungsi `*load_and_preprocess_image*(img_path)` yang memuat gambar dari path yang diberikan dengan ukuran target 224x224 piksel menggunakan `*image.load_img*`. Gambar tersebut kemudian diubah menjadi *array NumPy* menggunakan `*image.img_to_array*`, dan ditambahkan dimensi baru dengan `*np.expand_dims*` agar sesuai dengan bentuk input yang diharapkan oleh model. Nilai pixel kemudian dinormalisasi ke rentang 0 hingga 1 dengan membagi *array* gambar dengan 255.0.

Fungsi ini mengembalikan *array* gambar yang telah diproses. Selanjutnya, fungsi `*predict_disease*(img_path)` memanggil fungsi `*load_and_preprocess_image*` untuk memuat dan memproses gambar, kemudian menggunakan `*model.predict*` untuk membuat prediksi berdasarkan gambar tersebut. Indeks kelas dengan probabilitas tertinggi diperoleh menggunakan `np.argmax`, dan nilai probabilitas tertinggi diambil sebagai tingkat kepercayaan dari prediksi tersebut. Fungsi ini mengembalikan label kelas yang diprediksi dan tingkat kepercayaan.

Dengan demikian, kode ini mengimplementasikan langkah-langkah *preprocessing* gambar hingga prediksi penyakit berdasarkan model yang

telah dilatih, serta mengembalikan hasil prediksi dalam bentuk label kelas dan tingkat kepercayaan.

4.2.4 Arsitektur ResNet152v2



Gambar 4.6 Layer Arsitektur ResNet152V2

Berdasarkan gambar 4.6 Layer Arsitektur ResNet-152V2, terdapat beberapa lapisan penting dalam arsitektur ResNet-152v2:

1) Lapisan Konvolusi : ResNet-152v2 menggunakan lapisan konvolusi dengan filter 3x3 dan stride 1, yang berfungsi untuk mengekstraksi fitur dari gambar *input* pada berbagai tingkat resolusi.
2) Blok Residual : Komponen utama dalam ResNet adalah blok residual, yang terdiri dari dua atau lebih lapisan konvolusi yang terhubung. Setiap blok residual memiliki jalur pintas (*shortcut*) yang menghubungkan input langsung ke output blok tersebut, membantu mengatasi masalah gradien yang melemah selama pelatihan jaringan yang dalam.
3) Normalisasi Kelompok (*Batch Normalization*): Setelah setiap operasi konvolusi, ResNet-152v2 menerapkan normalisasi kelompok untuk menormalkan dan menstabilkan output lapisan sebelum diteruskan ke fungsi aktivasi. Normalisasi kelompok ini meningkatkan kecepatan dan stabilitas pelatihan model.
4) Fungsi Aktivasi : ResNet-152v2 menggunakan fungsi aktivasi ReLU setelah setiap lapisan konvolusi dan normalisasi kelompok. Fungsi aktivasi ReLU memperkenalkan non-linearitas pada jaringan, yang penting untuk mempelajari representasi yang lebih kompleks.
5) Pooling rata-rata global : Setelah beberapa blok residual, ResNet-152v2 menerapkan operasi pooling rata-rata global untuk mengurangi dimensi spasial dari fitur yang dihasilkan, menghasilkan

vektor fitur dengan dimensi yang lebih rendah yang mewakili informasi penting dari gambar input.

6) Lapisan *Fully Connected*: Setelah operasi *pooling* rata-rata global, ResNet-152v2 menggunakan lapisan *fully connected* untuk menghubungkan fitur yang dihasilkan dengan kelas *output*. Lapisan ini menghasilkan probabilitas berbagai kelas berdasarkan fitur input.

Arsitektur ResNet-152v2 adalah arsitektur yang dalam dan kompleks. Dengan menggunakan jalur pintas dan normalisasi kelompok, arsitektur ini berhasil mengatasi masalah gradien yang melemah dan membantu melatih model dengan lebih baik. Model ini terbukti efektif dalam tugas pengenalan gambar yang kompleks dan memiliki performa yang sangat baik. Berikut adalah cara implementasinya dalam kode pada gambar 4.7 di bawah ini :

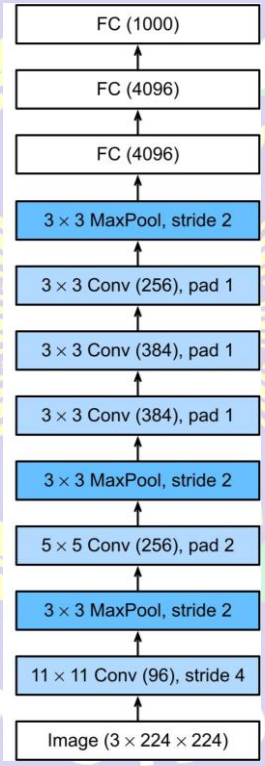
```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(IMG_WIDTH, IMG_HEIGHT, 3))

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(units=512, activation='relu')(x)
predictions = Dense(units=4, activation='softmax')(x)  # Adjusted to 4 classes

model = Model(inputs=base_model.input, outputs=predictions)

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])


history = model.fit(
    train_generator,
    epochs=EPOCHS,
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // BATCH_SIZE,
    verbose=1,
    use_multiprocessing=False  # matikan multiprocessing
)

# Simpan model
model.save('corn_disease_resnet_model.h5')
```

Gambar 4.7 Definisi Arsitektur ResNet152v2

Berdasarkan gambar 4.7 definisi arsitektur ResNet152V2, model ResNet152V2 dimuat dengan bobot yang sudah dilatih sebelumnya pada dataset ImageNet, tidak menyertakan lapisan atasnya (*fully connected layers*), dan menetapkan ukuran input gambar. Seluruh lapisan model dasar dibekukan agar bobotnya tidak berubah selama pelatihan. Kemudian, lapisan kustom ditambahkan: *GlobalAveragePooling*2D untuk mengurangi dimensi spasial, lapisan *Dense* dengan 512 unit dan aktivasi ReLU, serta lapisan *Dense* akhir dengan 4 unit dan aktivasi *softmax* untuk

klasifikasi 4 kelas. Model lengkap dibuat dengan input dari model dasar dan output dari lapisan kustom, lalu dikompilasi menggunakan *optimizer* Adam dan *loss function categorical crossentropy*. Model ini dilatih menggunakan data pelatihan dan validasi yang disediakan oleh *generator*, dengan langkah-langkah yang disesuaikan untuk *batch size* dan *Epoch* tertentu. Pelatihan dilakukan dengan 4 proses paralel dan *multiprocessing* untuk efisiensi. Akhirnya, model yang sudah dilatih disimpan ke file 'covid_disease_resnet_model.h5'.

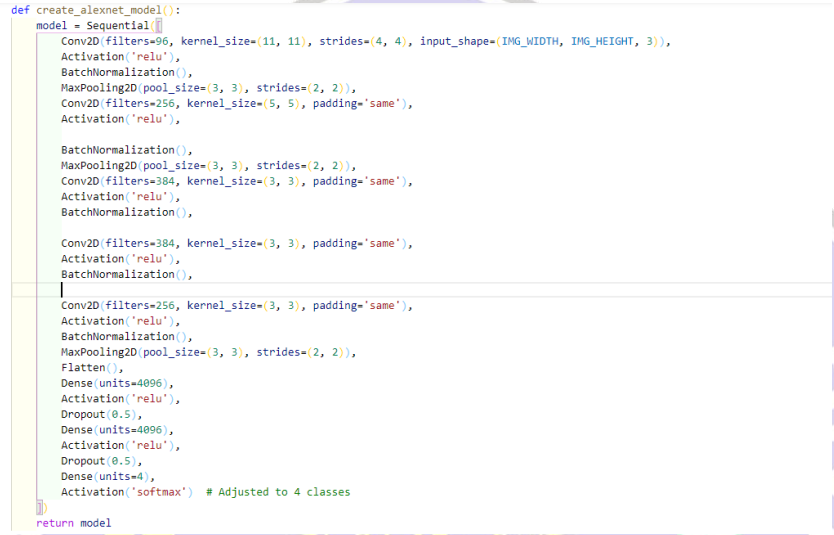
4.2.5 Arsitektur Alexnet



Gambar 4.8 Arsitektur AlexNet

Berdasarkan gambar 4.8 Arsitektur AlexNet, terdapat beberapa lapisan penting dengan fungsi spesifik. Lapisan konvolusi bertujuan untuk mengekstraksi fitur-fitur visual dari input gambar. Setelah itu, lapisan max pooling digunakan untuk mereduksi dimensi spasial dan membangun invariansi terhadap translasi. Lapisan aktivasi ReLU digunakan untuk memperkenalkan non-linearitas ke dalam model, yang penting untuk

mempelajari representasi data yang kompleks. Selanjutnya, lapisan dropout digunakan untuk mengurangi *overfitting* dengan menonaktifkan sebagian unit dalam jaringan secara acak selama pelatihan. Lapisan-lapisan konvolusi dan dropout tersebut diikuti oleh beberapa lapisan dense yang bertujuan untuk melakukan identifikasi objek dalam gambar. Selain itu, AlexNet juga menerapkan teknik regulasi L2 guna mencegah overfitting dan meningkatkan generalisasi model.



```
def create_alexnet_model():
    model = Sequential([
        Conv2D(filters=96, kernel_size=(11, 11), strides=(4, 4), input_shape=(IMG_WIDTH, IMG_HEIGHT, 3)),
        Activation('relu'),
        BatchNormalization(),
        MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),
        Conv2D(filters=256, kernel_size=(5, 5), padding='same'),
        Activation('relu'),

        BatchNormalization(),
        MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),
        Conv2D(filters=384, kernel_size=(3, 3), padding='same'),
        Activation('relu'),
        BatchNormalization(),

        Conv2D(filters=384, kernel_size=(3, 3), padding='same'),
        Activation('relu'),
        BatchNormalization(),

        Conv2D(filters=256, kernel_size=(3, 3), padding='same'),
        Activation('relu'),
        BatchNormalization(),
        MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),
        Flatten(),
        Dense(units=4096),
        Activation('relu'),
        Dropout(0.5),
        Dense(units=4096),
        Activation('relu'),
        Dropout(0.5),
        Dense(units=4),
        Activation('softmax')  # Adjusted to 4 classes
    ])
    return model
```

Gambar 4.9 Definisi Arsitektur AlexNet

Berdasarkan gambar 4.9 definisi arsitektur AlexNet, kode di atas mendefinisikan model neural network menggunakan Keras dengan struktur yang menyerupai arsitektur AlexNet, yang diadaptasi untuk klasifikasi menjadi 4 kelas. Model ini menggunakan urutan lapisan-lapisan yang disusun dalam objek `Sequential` dari Keras. Pertama, model dimulai dengan lapisan konvolusi dengan 96 filter, kernel 11x11, dan langkah 4x4, diikuti oleh fungsi aktivasi ReLU, normalisasi batch, dan lapisan pooling dengan ukuran pool 3x3 dan langkah 2x2. Lapisan kedua menggunakan 256 filter, kernel 5x5, dan padding 'same', diikuti oleh fungsi aktivasi ReLU, normalisasi batch, dan pooling. Lapisan ketiga dan keempat masing-masing terdiri dari konvolusi dengan 384 filter dan kernel 3x3, diikuti oleh aktivasi ReLU dan normalisasi batch. Lapisan kelima menggunakan 256 filter, kernel 3x3, padding 'same', diikuti oleh aktivasi ReLU, normalisasi batch, dan pooling.

Selanjutnya, model memasuki lapisan fully connected yang dimulai dengan meratakan output dari lapisan sebelumnya menjadi vektor satu dimensi, diikuti oleh lapisan dense dengan 4096 unit dan aktivasi ReLU, serta dropout untuk mencegah overfitting. Dua lapisan fully connected dengan konfigurasi yang sama diikuti oleh lapisan *fully connected* terakhir dengan 4 unit dan aktivasi softmax untuk klasifikasi 4 kelas. Model ini diakhiri dengan perintah untuk mengembalikan model yang sudah didefinisikan. Kombinasi dari lapisan-lapisan ini memungkinkan model untuk melakukan ekstraksi fitur yang kompleks dan klasifikasi yang akurat berdasarkan arsitektur AlexNet yang sudah diadaptasi.

4.3 Layer CNN

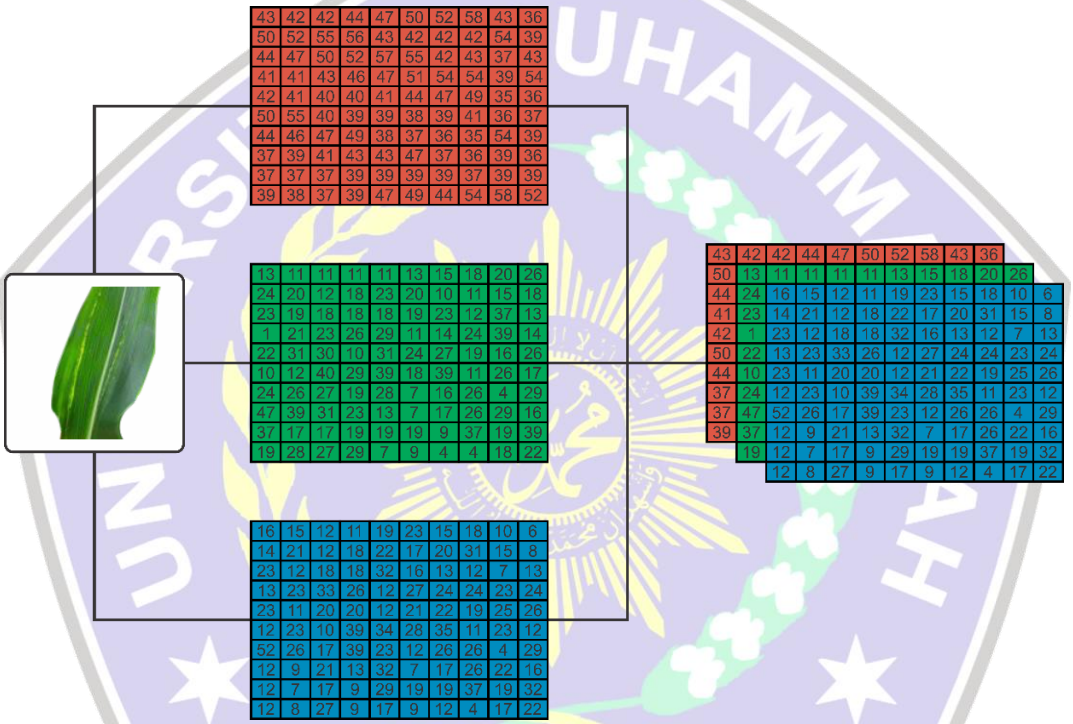
Proses pelatihan pada Metode *Convolutional Neural Network* (CNN) sangat penting untuk menciptakan model yang mampu mengidentifikasi dan memahami informasi dalam citra gambar. Melalui pelatihan ini, CNN mempelajari cara mengekstraksi fitur-fitur penting dari citra, mengenali pola-pola yang ada, dan memahami konteks visualnya. Dalam proses ini, citra-citra berlabel digunakan sebagai data pelatihan sehingga CNN dapat memahami hubungan antara atribut visual dan kelas atau kategori yang relevan.

4.3.1 *Input Layer*

Langkah awal dalam proses pelatihan model adalah memasukkan data citra bunga ke dalam lapisan input model. Pada tahap ini, data citra diubah menjadi matriks tiga dimensi dengan ukuran panjang x lebar x 3 kanal yang mewakili informasi warna RGB (*Red, Green, Blue*).

Selanjutnya, untuk meningkatkan kualitas dataset latihan yang terbatas, diterapkan teknik augmentasi data pada dataset citra. Teknik augmentasi yang digunakan mencakup rotasi, skala, pembalikan, peningkatan kontras, dan perubahan warna, yang menciptakan variasi dalam orientasi, ukuran, dan pencahayaan dalam dataset. Variasi ini membantu model menjadi lebih adaptif terhadap berbagai situasi di dunia nyata, meningkatkan kemampuannya dalam mengenali dan mengidentifikasi berbagai jenis bunga dengan lebih akurat.

Selain meningkatkan adaptasi model terhadap situasi yang beragam, teknik augmentasi juga penting dalam mencegah *overfitting* pada *dataset* pelatihan yang terbatas. *Overfitting* terjadi ketika model terlalu "menghafal" data pelatihan dan tidak mampu melakukan generalisasi dengan baik pada data uji atau data baru. Dengan adanya variasi dari augmentasi data, model dikenalkan pada beragam dengan berbagai variasi data yang mirip, namun tetap berbeda, sehingga model dapat belajar pola yang lebih umum dan general :



Gambar 4.10 Pemecahan Citra menjadi Array

Berdasarkan gambar 4.10 menggambarkan proses pemecahan citra menjadi array untuk input layer 256x256x3:

1. *Input Layer*: Pada tahap ini, citra yang memiliki ukuran 256x256x3 dimasukkan ke dalam model. Ukuran 256x256 menunjukkan resolusi citra dalam piksel, sementara angka 3 merepresentasikan tiga kanal warna (RGB), yang memberikan informasi tentang warna setiap piksel.
2. *Batch Size*: Pada tahap ini, kita menentukan ukuran batch yang akan digunakan selama proses pelatihan. *Batch size* adalah jumlah citra

yang diproses dalam satu sesi pelatihan. Misalnya, jika kita menggunakan batch size sebesar 32, maka 32 citra akan diproses secara bersamaan dalam satu *batch*.

3. Pembagian Menjadi *Batch*: Data input yang berukuran 256x256x3 dipisahkan menjadi beberapa batch berdasarkan ukuran batch size yang telah ditentukan. Sebagai contoh, jika batch size-nya adalah 32, maka data input akan diubah menjadi bentuk array dengan ukuran 32x256x256x3. Angka 32 pada dimensi pertama menunjukkan jumlah citra yang diproses dalam satu *batch*.
4. Iterasi dan Pelatihan: Selama pelatihan model, iterasi dilakukan dengan mengambil satu batch citra dari data pelatihan dan memprosesnya melalui jaringan saraf tiruan. Model akan terus mengulang proses ini untuk setiap batch data dalam data pelatihan selama beberapa *epoch* (siklus pelatihan).
5. Agregasi Hasil: Setelah seluruh proses pelatihan selesai, hasil dari setiap batch digabungkan untuk memperoleh hasil akhir dari pelatihan model.

Dengan penjelasan ini, kita dapat lebih memahami bagaimana citra dengan ukuran 256x256x3 dipecah dan digunakan dalam proses pelatihan model jaringan saraf tiruan. Convolutional Layer

4.3.2 *Convolutional Layer*

1) Konvolusi

Pada tahap ini, filter (atau kernel) berukuran kecil digunakan untuk melakukan operasi konvolusi pada data input. Operasi konvolusi ini dilakukan dengan menggeser filter di seluruh area input untuk menghasilkan peta fitur (*feature map*). Data dari kanal RGB di setiap piksel kemudian diolah melalui lapisan konvolusi, yang bertujuan untuk mengekstraksi fitur-fitur yang terdapat dalam citra menggunakan filter. Sebagai contoh, pada Gambar 4.11, terdapat citra input pada saluran warna merah. Angka-angka tersebut mewakili

intensitas warna merah pada gambar, dengan rentang nilai antara 0 hingga 255.

43	42	42	44	47	50	52	58	43	36
50	52	58	56	43	42	42	42	54	39
44	47	50	52	57	55	43	42	37	43
41	41	43	46	49	51	54	54	39	54
42	41	40	40	41	44	47	49	36	36
50	52	40	39	39	38	39	41	36	37
44	46	47	49	38	37	36	36	54	39
37	39	41	43	43	47	37	36	39	36
37	37	37	39	39	39	39	37	39	39
39	38	37	39	47	49	44	54	58	52

Gambar 4.11 *Input Channel Red*

2) Fitur atau kernel

Filter atau kernel adalah matriks bobot yang digunakan dalam proses konvolusi pada data *input*. Matriks ini memiliki ukuran yang lebih kecil daripada data *input* dan berfungsi untuk mengekstraksi fitur-fitur penting dari data tersebut. Filter diaplikasikan dengan menggesernya secara beraturan melintasi area input menggunakan langkah (*stride*) yang telah ditentukan. Setiap elemen dalam filter merepresentasikan bobot yang akan diterapkan pada bagian yang sesuai dari data *input* saat melakukan konvolusi.

Pada awalnya, nilai-nilai dalam filter diinisialisasi secara acak, namun selama proses pelatihan model, nilai-nilai ini diperbarui melalui proses optimisasi seperti pembelajaran berbasis gradien. Gambar 4.15 memberikan visualisasi dari struktur dan nilai-nilai dalam filter, menunjukkan bagaimana filter tersebut bekerja untuk menangkap pola dan fitur dari data input.

2	2	2	2	2
1	1	1	1	1
0	0	0	0	0
-1	-1	-1	-1	-1
-2	-2	-2	-2	-2

Gambar 4.12 Filter

3) Perhitungan Konvolusi

Proses konvolusi melibatkan operasi perkalian elemen matriks input data dan kernel pada setiap posisi, diikuti oleh penjumlahan dari hasil perkalian tersebut. Hasilnya adalah nilai aktivasi atau respons filter pada posisi tersebut. Perhitungan konvolusi sebagai berikut:

$$y_{m,n} = g\ (\sum\sum x\ [j,k]W[m-n,n-k] + b)$$

Berikut penjelasan rumus diatas :

- $\sum x\ [j,k]$ adalah nilai piksel pada posisi (j,k) dalam citra input c.
- $W[m-n,n-k]$ Adalah nilai bobot (kernel) pada posisi $(m-n,n-k)$ dalam filter.
- b adalah bias (*offset*) dari lapisan konvolusi
- g adalah fungsi aktivitas, yang dalam kasus ini tidak diberikan detail dan diasumsikan sebagai fungsi identitas.

Pencarian kernel matrik sebagai berikut :

$(W * \text{x})_{m,n} = 43 \text{ x } 2 + 42 \text{ x } 2 + 42 \text{ x } 4 + 47 \text{ x } 2 + 50 \text{ x } 1 + 52 \text{ x } 1 + 58 \text{ x } 1 + 56 \text{ x } 1 + 43 \text{ x } 1 + 44 \text{ x } 0 + 47 \times 0 + 50 \times 0 + 52 \times 0 + 57 \times 0 + 41 \times 0 + (-1) \times 43 + (-1) \times 46 + (-1) \times 41 + (-2) \times 41 \times 2 + (-2) = 67$

Fungsi konvolusi mengacu pada proses matematis di mana nilai-nilai piksel dalam citra x diproses dengan memasukkan filter W pada setiap posisi . Setiap posisi ini mewakili bagaimana citra akan "dilintasi" oleh filter untuk menghasilkan output baru. Proses ini juga melibatkan penambahan nilai bias $s1$, yang memungkinkan model untuk menyesuaikan prediksi dengan lebih baik terhadap data yang ada. Dengan demikian, konvolusi tidak hanya menggambarkan integrasi nilai piksel dengan filter, tetapi juga bagaimana filter tersebut dapat mempengaruhi dan mengubah representasi citra secara sistematis.

$$y_{m,n} = g\left(\sum W_j x_j + b\right)$$

$$y_{0,1} = Max(67 + 1{,}0)$$

Pada posisi (0, 1), evaluasi $y_{0,1}$ dilakukan dengan menggantikan nilai-nilai piksel x dan bobot W yang telah ditentukan. Setelah proses konvolusi dilakukan, dihasilkan nilai $y_{0,1}$ yang adalah 68. Langkah selanjutnya adalah menerapkan fungsi aktivasi maksimum (ReLU) pada nilai konvolusi ini. Fungsi aktivasi ReLU mengubah nilai menjadi nol jika nilai konvolusi kurang dari nol, atau mempertahankan nilai tersebut jika lebih besar dari nol. Dalam kasus ini, karena $y_{0,1}$ = 68 (yang jelas lebih besar dari nol), nilai keluaran yang dihasilkan dari posisi (0, 1) setelah ReLU adalah 68.

4) Output Konvolusi

68	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

Gambar 4.13 Output Konvolusi

Pada gambar 4.13, lapisan konvolusional terjadi proses pengambilan fitur yang bertujuan untuk menghasilkan peta fitur dari data citra. Tujuan utamanya adalah untuk mengenali pola-pola yang ada pada setiap citra. Proses ekstraksi fitur ini dilakukan dengan menerapkan filter pada setiap citra, di mana filter berfungsi untuk mendeteksi berbagai karakteristik seperti tepi, tekstur, atau bentuk tertentu dalam citra. Filter ini akan melintasi citra dan melakukan operasi konvolusi, sehingga menghasilkan peta fitur yang mencerminkan adanya pola atau fitur spesifik dalam citra tersebut. Hasil dari ekstraksi fitur ini adalah representasi citra dalam bentuk peta fitur yang lebih mudah diolah oleh model untuk tujuan analisis lebih lanjut, seperti klasifikasi atau deteksi objek.

4.3.3 *Activation Function*

Untuk mengenali objek dalam citra, diperlukan proses pemisahan antara objek dan latar belakang. Dalam penelitian ini, fungsi aktivasi ReLU digunakan untuk menentukan apakah neuron dalam jaringan saraf aktif atau tidak aktif. Fungsi aktivasi ini memastikan bahwa hanya neuron yang relevan dengan citra yang dipilih. Selain itu, fungsi aktivasi

bertujuan untuk menambahkan sifat non-linear pada jaringan saraf. Berikut adalah tahapan fungsi aktivasi:

1) *Input Collection*

   Lapisan aktivasi menerima output dari lapisan sebelumnya, yang bisa berupa matriks atau vektor, tergantung pada struktur jaringan.

2) *Weighted Summation*

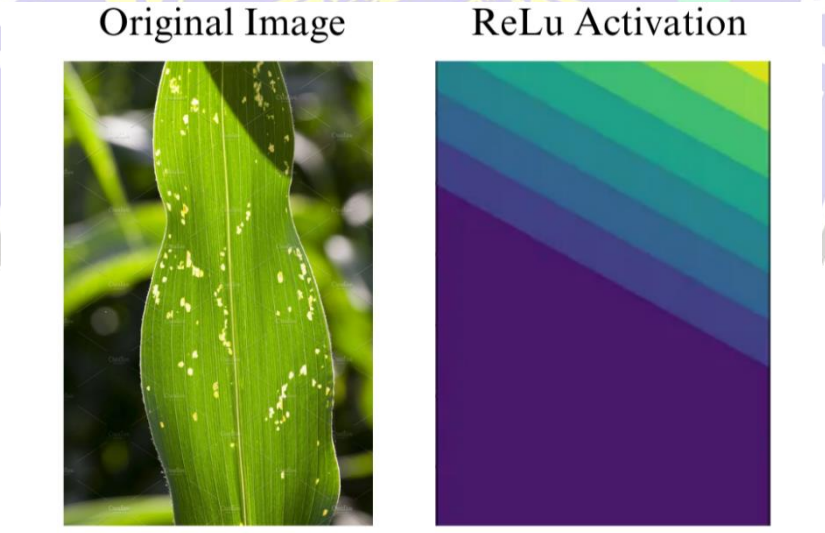
   Setiap elemen dalam output dijumlahkan dengan bobot dan bias (jika ada) sesuai dengan fungsi aktivasi yang digunakan.

3) *Activation*

   Output yang telah dijumlahkan dengan bobot dan bias kemudian dievaluasi menggunakan fungsi aktivasi yang bersangkutan. Fungsi aktivasi ini diperlukan untuk memperkenalkan non-linearitas dalam jaringan saraf. Beberapa fungsi aktivasi yang umum digunakan adalah Sigmoid, ReLU, Tanh, dan Leaky ReLU.

4) *Output Formation*

   Hasil dari fungsi aktivasi menjadi output dari lapisan aktivasi dan akan menjadi input untuk lapisan berikutnya dalam jaringan saraf.



Gambar 4.14 *Output* Fungsi Aktivasi

4.3.4 *Pooling Layer*

Pada tahap ini, *pooling layer* menerima *input* dari lapisan sebelumnya. Biasanya, input ini adalah hasil dari konvolusi atau operasi lainnya yang dilakukan oleh lapisan sebelumnya dalam arsitektur jaringan saraf. Fungsi utama lapisan pooling adalah untuk mengurangi dimensi spasial dari citra serta mengurangi jumlah parameter dan operasi yang dilakukan dalam jaringan saraf. Lapisan pooling bekerja secara independen pada setiap fitur. Input yang diterima oleh lapisan pooling adalah peta fitur yang dihasilkan oleh lapisan konvolusi.

1) *Input Collection*

   Penelitian ini menggunakan *max pooling* dengan ukuran 2x2 dan jarak (*stride*) 2, yang mengurangi ukuran citra hingga 50% dari ukuran aslinya, misalnya dari 150x150 menjadi 75x75.

2) *Operasi Pooling*

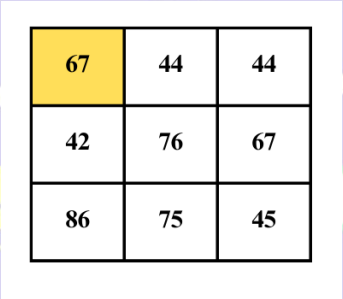
   Operasi pooling diterapkan pada setiap window. Sebagai contoh, seperti yang terlihat pada Gambar 4.18, dilakukan operasi max pooling pada input, di mana outputnya adalah nilai terbesar dari semua nilai input dalam setiap window.

67	42	42	44	38	40
42	41	41	38	44	43
41	42	43	41	67	38
23	12	76	21	32	12
23	23	42	75	23	45
86	23	42	12	43	2

Gambar 4.15 *Input* pada *Pooling Layer*

3) *Output Formation*

Hasil dari operasi *pooling* menjadi *output* dari lapisan *pooling* dan akan digunakan sebagai *input* untuk lapisan berikutnya dalam jaringan saraf. Dalam contoh ini, nilai terbesar adalah 67, sehingga nilai tersebut dipilih dan menghasilkan citra seperti yang ditunjukkan pada Gambar 4.16. Pada citra daun jagung, proses pooling diterapkan setelah proses konvolusi dengan menggunakan max pooling.



Gambar 4.16 *Output* pada *Pooling Layer*

4.3.5 *Fully Connected Layer*

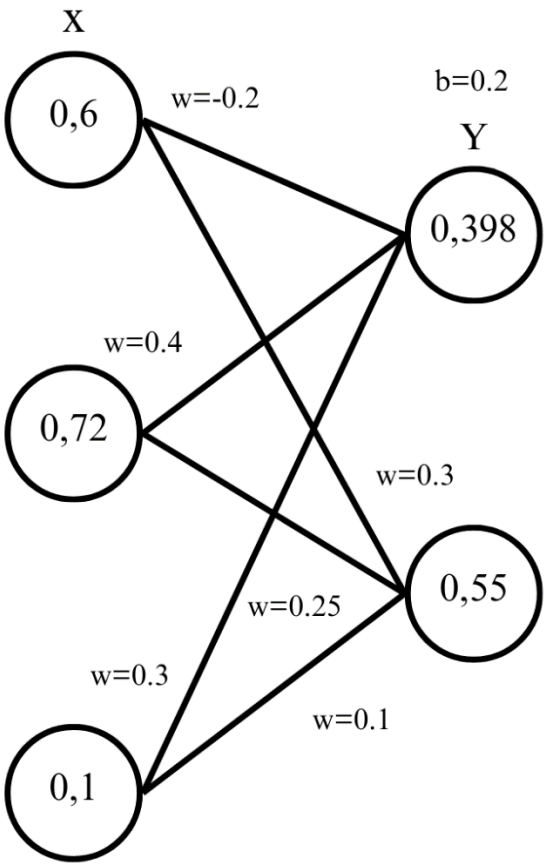
Tahap berikutnya dalam *Convolutional Neural Network* (CNN) adalah lapisan *fully connected*. Pada awal lapisan *fully connected*, data matriks tiga dimensi yang dihasilkan dari tahap konvolusi diubah menjadi vektor satu dimensi melalui operasi "*flatten*". Tahap ini bertujuan untuk mengubah representasi spasial dari data menjadi representasi linear dalam bentuk vektor.

Setelah proses "*flatten*", nilai setiap *neuron* (x) dihitung menggunakan bobot (w) dan ditambahkan dengan bias (b). Perhitungan ini mengikuti prinsip umum dari lapisan *fully connected*, di mana setiap *neuron* dihubungkan dengan setiap neuron di lapisan sebelumnya dan lapisan berikutnya. Proses ini memungkinkan jaringan saraf untuk mempelajari hubungan yang lebih kompleks dalam data.

Hasil dari perhitungan ini menentukan *output* dari setiap *neuron* pada lapisan *fully connected*, yang kemudian menjadi input untuk lapisan berikutnya. Proses ini terus berlanjut hingga mencapai lapisan *output*, di

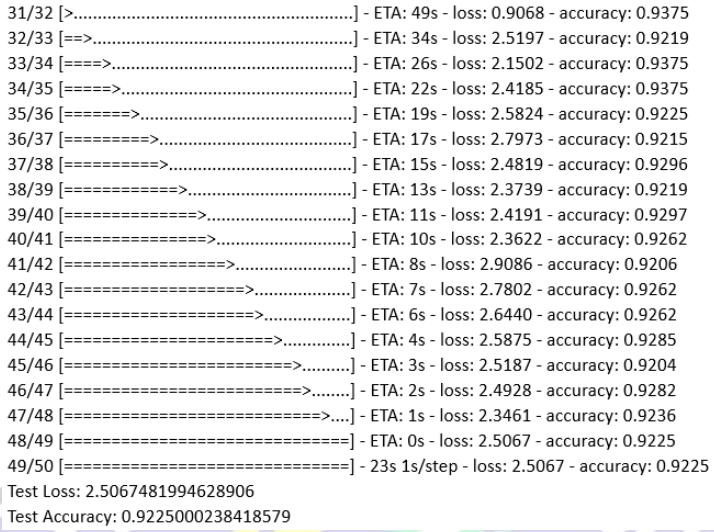
mana output akhir dari jaringan digunakan untuk melakukan identifikasi atau tugas lain sesuai dengan tujuan model.

Gambaran keseluruhan dari jaringan saraf pada model ResNet, yang mencakup lapisan *fully connected* ini, menunjukkan bagaimana arsitektur CNN yang terkenal ini berhasil dalam berbagai tugas penglihatan komputer. Tahap *fully connected* dalam ResNet biasanya diikuti oleh lapisan-lapisan akhir seperti *softmax*, yang menghasilkan distribusi probabilitas dari kelas-kelas yang mungkin pada tugas identifikasi.



Gambar 4.17 *Fully Connected Layer*

4.4 Hasil Pelatihan



Gambar 4.18 Hasil Pelatihan

Berdasarkan gambar 4.18 yaitu hasil pelatihan di atas, pada bagian akhir terdapat informasi mengenai pelatihan model dengan 50 *Epoch* (siklus iterasi) yang ditampilkan. Baris terakhir menunjukkan bahwa telah diproses sebanyak 50 *batch* dengan waktu sekitar 23 detik (1 detik per step) untuk setiap *batch*. Hasil evaluasi model dengan beberapa metrik juga ditampilkan. *"Loss"* menunjukkan nilai fungsi *loss* pada *Epoch* terakhir, yaitu 2.5067. Semakin kecil nilai *loss*, semakin baik model dapat mempelajari pola dari data. "*Accuracy*" menunjukkan tingkat akurasi model pada *Epoch* terakhir, yaitu 0.9225. Akurasi ini merupakan persentase dari jumlah prediksi yang benar terhadap total data yang diproses.

Selanjutnya, evaluasi model pada data uji juga ditampilkan. "*Test Loss*" menunjukkan nilai *loss* pada data uji, yaitu 2.5067481994628906, yang mengindikasikan seberapa baik model tersebut mampu memprediksi data uji. "*Test Accuracy*" menunjukkan tingkat akurasi pada data uji, yaitu 0.9225000238418579, yang mencerminkan persentase prediksi yang benar terhadap total data uji. Dengan demikian, kita dapat menyimpulkan bahwa model memiliki tingkat akurasi yang tinggi baik pada data pelatihan maupun data uji, meskipun terdapat sedikit perbedaan pada nilai loss dan akurasi antara

kedua data tersebut. Nilai-nilai ini menunjukkan bahwa model cukup efektif dalam mengenali dan mempelajari pola dari data yang diberikan.

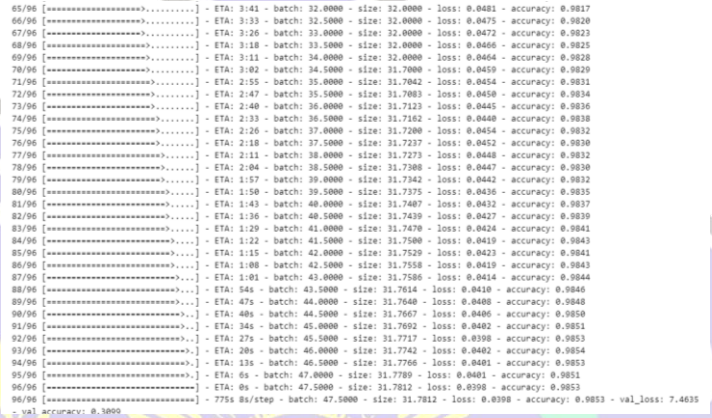
4.5 Pengujian Model

Pengujian model *Convolutional Neural Network* (CNN) adalah proses evaluasi kinerja model setelah dilatih untuk menentukan seberapa baik model tersebut mengenali dan mengklasifikasikan data yang belum pernah dilihat sebelumnya. Proses ini melibatkan pembagian *dataset* menjadi *training set*, *validation set*, dan *test set*, di mana *test set* digunakan untuk evaluasi akhir setelah pelatihan selesai. Model diuji menggunakan *test set* yang tidak pernah dilihat sebelumnya, dan kinerjanya dievaluasi dengan metrik seperti akurasi, presisi, *recall*, *F1-score*, dan *confusion matrix*. Hasil pengujian sering kali divisualisasikan melalui *confusion matrix*, *ROC curve*, atau *precision-recall curve* untuk memudahkan interpretasi. Selain itu, analisis kesalahan dilakukan untuk mengidentifikasi jenis kesalahan yang paling sering dilakukan oleh model, yang membantu dalam perbaikan dan peningkatan model di masa mendatang. Pengujian ini penting untuk memastikan bahwa model tidak hanya bekerja baik pada data pelatihan tetapi juga dapat menggeneralisasi dengan baik pada data baru[20].

4.5.1 Pengujian Model ResNet152V2

Pada pengujian model ResNet152V2, terdapat dua grafik utama yang biasanya digunakan untuk memvisualisasikan performa model selama proses pelatihan, yaitu grafik *loss* dan grafik *accuracy*. Grafik *loss* menunjukkan bagaimana nilai kerugian (*loss*) model berubah seiring waktu selama pelatihan, yang membantu dalam memahami seberapa baik model sedang menyesuaikan diri dengan data pelatihan. Grafik ini biasanya menunjukkan penurunan nilai kerugian, yang mengindikasikan bahwa model semakin baik dalam memprediksi data pelatihan. Sementara itu, grafik *accuracy* menunjukkan bagaimana akurasi model berubah selama proses pelatihan, memberikan gambaran

tentang seberapa sering model membuat prediksi yang benar pada data pelatihan dan data validasi. Peningkatan grafik *accuracy* mengindikasikan bahwa model semakin baik dalam mengenali pola dalam data dan membuat prediksi yang benar. Kedua grafik ini bersama-sama memberikan wawasan penting tentang kinerja model dan membantu dalam proses tuning untuk meningkatkan performa model lebih lanjut.

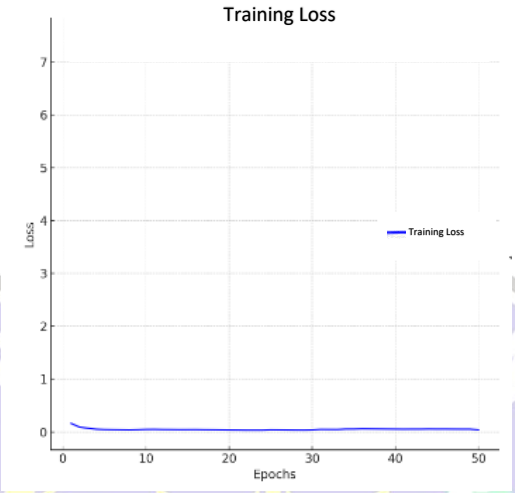


Gambar 4.19 Hasil Pelatihan dengan Arsitektur ResNet152V2

Berdasarkan gambar 4.19, terlihat bahwa proses pelatihan model berjalan dalam 96 langkah (*steps*). Setiap langkah menampilkan informasi mengenai *Estimated Time of Arrival* (ETA), ukuran *batch*, ukuran total data, nilai *loss*, dan akurasi pada setiap *batch*. Nilai akurasi bertahap meningkat selama proses pelatihan, menunjukkan bahwa model belajar dengan baik dari data yang disediakan. Pada langkah-langkah akhir, nilai akurasi hampir mencapai nilai maksimal, berkisar antara 0.9800 hingga 0.9853, dengan nilai loss yang terus menurun, menunjukkan perbaikan kinerja model.

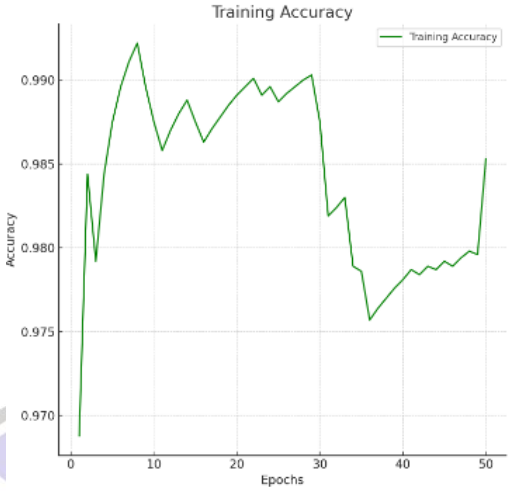
Pada akhir pelatihan, terdapat ringkasan yang menunjukkan bahwa proses pelatihan selesai dalam 7758 detik per langkah, dengan ukuran *batch* 47.5000 dan total data sebesar 31.7812. Nilai akhir dari *loss* adalah 0.0098 dan akurasi sebesar 0.9853, sedangkan nilai *loss* pada data validasi (*val_loss*) tercatat sebesar 7.4635 dengan akurasi validasi (*val_accuracy*) sebesar 0.9809. Nilai-nilai ini menunjukkan bahwa

model memiliki kinerja yang baik baik pada data pelatihan maupun pada data validasi, meskipun terdapat sedikit *overfitting* yang ditunjukkan oleh perbedaan antara loss pelatihan dan loss validasi.



Gambar 4.20 Loss ResNet

Berdasarkan gambar 4.20, grafik "*Training Loss*" menunjukkan penurunan nilai *loss* selama proses pelatihan model hingga mencapai 50 *epoch*. Pada awal pelatihan, nilai *loss* berada di sekitar 0.9, dan dengan cepat menurun pada beberapa *epoch* pertama. Setelah itu, nilai *loss* cenderung stabil di sekitar nilai yang sangat rendah, mendekati 0. Grafik ini mengindikasikan bahwa model berhasil meminimalkan *loss* pada data pelatihan secara efektif. Stabilitas nilai *loss* yang rendah setelah beberapa *epoch* menunjukkan bahwa model telah mencapai konvergensi, di mana penambahan *epoch* tidak lagi memberikan penurunan signifikan pada nilai *loss*. Hasil tersebut menandakan bahwa model telah belajar pola dari data pelatihan dengan baik dan tidak mengalami *overfitting* yang signifikan.

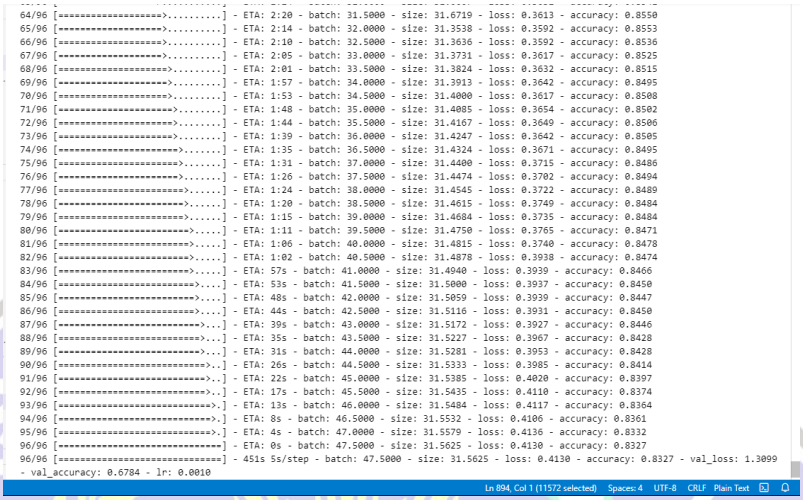


Gambar 4.21 *Accuracy* ResNet

Berdasarkan hasil pada gambar 4.21 yang menunjukkan grafik "*Training Accuracy*," kita dapat melihat bagaimana akurasi model berubah selama 50 *epoch* pelatihan. Pada awal pelatihan, akurasi meningkat dengan cepat, menunjukkan bahwa model dengan cepat mempelajari pola dari data pelatihan. Akurasi mencapai puncaknya di sekitar *epoch* ke-10 dengan nilai hampir 0.990. Namun, setelah mencapai puncak ini, akurasi mengalami fluktuasi. Ada beberapa penurunan yang signifikan setelah sekitar *epoch* ke-30, di mana akurasi turun di bawah 0.980, sebelum kemudian meningkat lagi menjelang akhir pelatihan. Akurasi pada akhir pelatihan menunjukkan tren peningkatan, meskipun belum mencapai nilai puncak awal. Fluktuasi ini bisa menunjukkan beberapa hal, seperti adanya noise dalam data pelatihan atau perubahan dalam pembelajaran model yang mungkin disebabkan oleh pengaturan hyperparameter seperti laju pembelajaran. Namun, secara keseluruhan, akurasi yang tinggi pada sebagian besar epoch menunjukkan bahwa model ini memiliki kemampuan generalisasi yang baik terhadap data pelatihan.

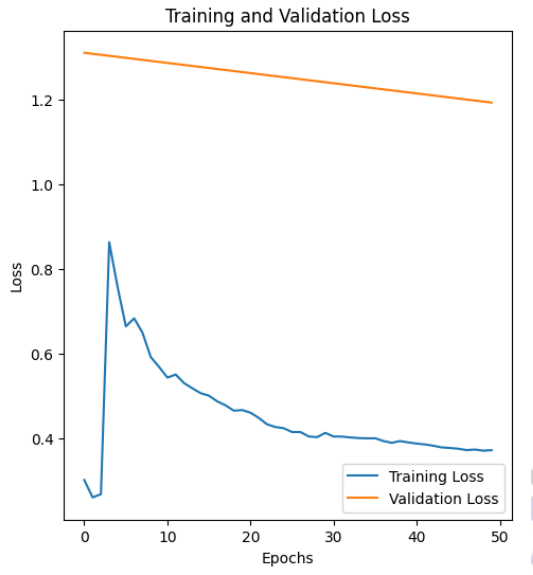
4.5.2 Pengujian Model AlexNet

Pada pengujian model AlexNet, terdapat dua grafik utama yang biasanya digunakan untuk memvisualisasikan performa model selama proses pelatihan, yaitu grafik *loss* dan grafik *accuracy* sebagai berikut :



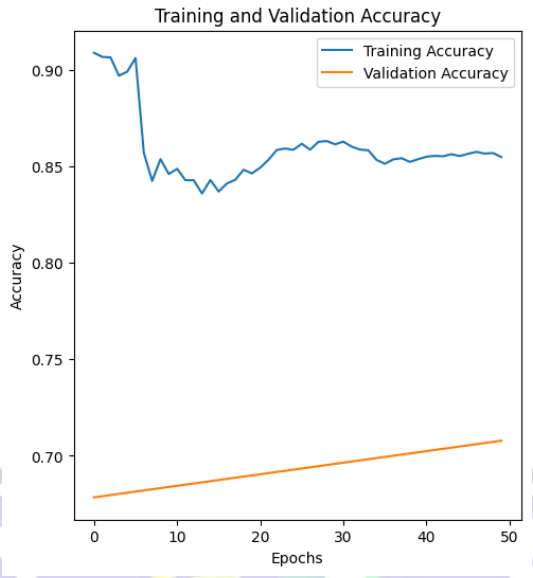
Gambar 4.22 Hasil Pelatihan dengan Arsitektur AlexNet

Gambar 4.22 menunjukkan proses pelatihan model jaringan saraf dalam 96 iterasi, dengan setiap iterasi memproses *batch* yang bervariasi antara 31.4 hingga 47.5 ukuran *batch*. Waktu yang diperkirakan untuk menyelesaikan setiap iterasi ditampilkan dalam format ETA. Nilai *loss* berkisar dari sekitar 0.3613 hingga 0.4136, menunjukkan seberapa baik model menyesuaikan data pelatihan. Akurasi bervariasi dari sekitar 0.8356 hingga 0.8556, menunjukkan tingkat keberhasilan model dalam memprediksi *output* yang benar pada data pelatihan. Pada akhir pelatihan, ditampilkan nilai *val_accuracy* sebesar 0.6784 dan *val*_loss sebesar 1.8099, yang menunjukkan performa model pada data validasi. *Learning rate* yang digunakan selama pelatihan adalah 0.0010, yang merupakan faktor penting dalam menentukan seberapa besar penyesuaian yang dilakukan pada bobot model selama pelatihan.



Gambar 4.23 *Loss* AlexNet

Berdasarkan gambar 4.23 menunjukkan Hasil Grafik *Training Loss* dan *Validation Loss* selama 50 *epoch* memiliki karakteristik berikut : *Training Loss* (garis biru) menurun signifikan selama sekitar 10 *epoch* pertama, menunjukkan bahwa model belajar dan memperbaiki kinerjanya pada data *training*. *Training Loss* mulai dari sekitar 0.9 dan turun hingga sekitar 0.3 pada akhir *epoch*. Setelah 10 *epoch* pertama, laju penurunan *loss* melambat dan stabil sekitar 40 *epoch*. Sebaliknya, *Validation Loss* (garis oranye) dimulai pada nilai lebih tinggi dari *Training Loss* sekitar 1.2 dan tidak menunjukkan penurunan signifikan sepanjang 50 *epoch*, turun sedikit menjadi sekitar 1.1. Hal ini menunjukkan bahwa model mengalami *overfitting*, di mana performa model pada data training meningkat tetapi tidak ada peningkatan signifikan pada data *validation*. Bahkan, *Validation Loss* cenderung sedikit meningkat seiring waktu, yang merupakan indikasi *overfitting*. Salah satu cara mengatasi *overfitting* adalah dengan menggunakan *regularization, dropout*, atau pengumpulan lebih banyak data *training*. Penggunaan teknik *early stopping* juga dapat membantu menghentikan pelatihan lebih awal guna menghindari *overfitting*.



Gambar 4.24 *Accuracy* AlexNet

Berdasarkan gambar 4.24 menunjukkan Hasil Grafik *Training Accuracy* dan *Validation Accuracy* selama 50 *epoch*. *Training Accuracy* (garis biru) memulai dengan nilai sekitar 0.90, kemudian mengalami penurunan tajam selama sekitar 10 *epoch* pertama, sebelum stabil di sekitar 0.85. Hasil tersebut menunjukkan bahwa model belajar dari data *training* dengan baik pada awalnya, tetapi setelah itu, peningkatan kinerja menjadi lebih lambat dan akhirnya stabil. Sebaliknya, *Validation Accuracy* (garis oranye) dimulai dengan nilai sekitar 0.70 dan meningkat secara perlahan sepanjang 50 *epoch*, berakhir sedikit di bawah 0.75. Hal ini menunjukkan bahwa meskipun model mampu meningkatkan kinerja pada data *training*, peningkatan pada data *validation* jauh lebih lambat. Perbedaan yang signifikan antara *Training Accuracy* dan *Validation Accuracy* ini bisa mengindikasikan adanya *overfitting*, di mana model bekerja dengan baik pada data *training* tetapi tidak dapat menggeneralisasi dengan baik pada data *validation*. Untuk mengatasi masalah ini, dapat diterapkan teknik seperti *regularization*, *dropout*, atau meningkatkan ukuran data *training*.

4.6 *Interface* Sistem

Sistem deteksi penyakit pada tanaman jagung menggunakan Metode CNN memiliki dua *interface* utama yaitu *input images* dan *output images*. Pada *interface input*, *user* mengunggah atau mengambil gambar tanaman jagung yang akan diperiksa. Setelah gambar diproses oleh model CNN, hasil analisis ditampilkan di hasil deteksi. Hasil ini mencakup gambar yang diberi anotasi, jenis penyakit yang terdeteksi, tingkat kepercayaan model, dan saran tindakan yang dapat diambil. Sistem ini dirancang agar sederhana dan *user-friendly*, sehingga user dapat dengan mudah mengunggah gambar dan segera mendapatkan hasil deteksi.

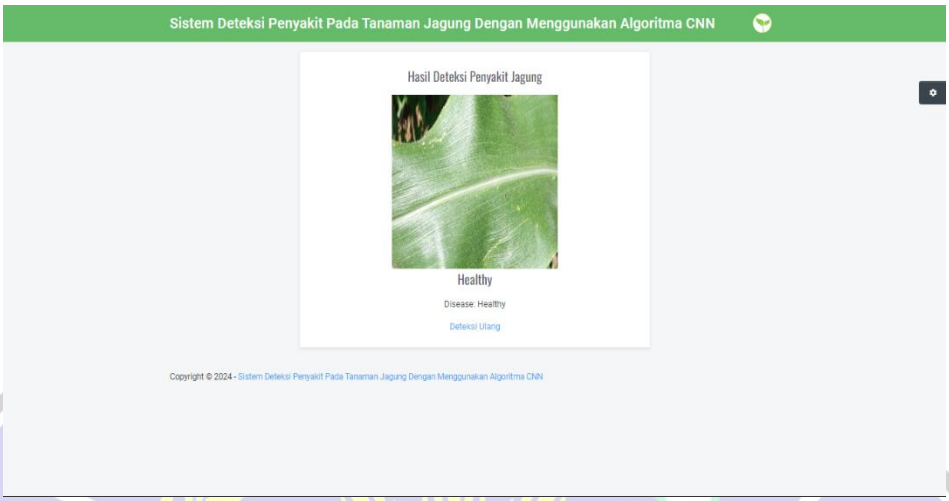
1) *Interface Input Images*

Gambar 4.25 *Interface Input Images*

Gambar 4.25 menampilkan halaman *interface* untuk Sistem Deteksi Penyakit Pada Tanaman Jagung Menggunakan Metode *Convolutional Neural Network* (CNN). *Interface* tersebut berfungsi untuk *user* mengunggah gambar daun jagung yang ingin diperiksa untuk mendeteksi adanya penyakit. *User* dapat memilih file gambar dari perangkat mereka melalui tombol "*Choose File*" dan kemudian mengunggahnya dengan menekan tombol "*Upload*". Setelah gambar diunggah, sistem akan memprosesnya menggunakan Metode CNN untuk mengidentifikasi dan mendeteksi penyakit pada tanaman jagung. *Interface* ini dirancang untuk

memudahkan user dalam melakukan deteksi penyakit tanaman secara cepat dan akurat.

2) *Interface* Hasil Deteksi



Gambar 4.26 *Interface* Hasil Deteksi

Gambar 4.26 menampilkan *interface* hasil deteksi penyakit pada tanaman jagung menggunakan Metode CNN. *Interface* ini menunjukkan gambar daun jagung yang telah dianalisis, dengan hasil deteksi yang menunjukkan bahwa tanaman tersebut sehat ("*Healthy*"). Selain itu, terdapat informasi tentang penyakit yang terdeteksi (dalam hal ini "*Healthy*") dan opsi untuk melakukan deteksi ulang. *Interface* ini dirancang agar mudah digunakan, memberikan hasil deteksi secara langsung kepada pengguna.

3) Pengkodean Sistem

```python
from flask import Flask, render_template, request, redirect, url_for
import tensorflow as tf
import numpy as np
import os
from tensorflow.keras.preprocessing import image
import cv2
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg')

# Load the trained model
model = tf.keras.models.load_model('corn_disease_cnn_model.h5')

# Define class labels
class_labels = ['Common Rust', 'Gray Leaf Spot', 'Healthy', 'Northern Leaf Blight']

# Create Flask app
app = Flask(__name__)

# Set upload folder
UPLOAD_FOLDER = 'static/uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Load and preprocess image
def load_and_preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0
    return img_array
```

```
def predict_disease(img_path):
    (variable) predicted_class: Any
    predicted_class = np.argmax(predictions)
    confidence = predictions[0][predicted_class]
    return class_labels[predicted_class], confidence

# Home route
@app.route('/')
def index():
    return render_template('index.html')

# Upload and predict route
@app.route('/predict', methods=['POST'])
def upload_file():
    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)
        file = request.files['file']
        if file.filename == '':
            return redirect(request.url)
        if file:
            filename = file.filename
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(file_path)
            disease, confidence = predict_disease(file_path)
            return render_template('result.html', disease=disease, confidence=confidence, image_path=file_path)

if __name__ == '__main__':
    app.run(debug=True)
```

Gambar 4.27 Pengkodean Sistem

Berdasarkan gambar 4.27, pengkodean sistem yang menggunakan *Flask* dan *TensorFlow* untuk mendeteksi penyakit pada tanaman jagung melibatkan beberapa langkah utama sebagai berikut. Pertama, modul yang dibutuhkan seperti *Flask*, *TensorFlow*, dan *NumPy* diimpor. Model CNN yang telah dilatih sebelumnya dimuat, dan label kelas penyakit didefinisikan. Aplikasi *Flask* dibuat dan diatur folder untuk mengunggah gambar. Gambar yang diunggah oleh pengguna dimuat dan diproses untuk menyesuaikan dengan format input model CNN. Fungsi prediksi kemudian digunakan untuk mendeteksi penyakit pada gambar yang diunggah, mengembalikan hasil prediksi dalam bentuk label penyakit dan tingkat kepercayaan. Rute utama merender halaman utama (`*index.html*`), sementara rute untuk unggahan gambar menangani pengunggahan file, melakukan prediksi, dan merender hasilnya pada halaman hasil (`*result.html*`). Aplikasi *Flask* kemudian dijalankan pada mode *debug*.

4.7 Pengujian Identifikasi

Pengujian sistem identifikasi jenis penyakit pada daun jagung dilakukan melalui sebuah *website* yang dikembangkan menggunakan *Flask*. Dalam eksperimen ini, diterapkan dan dievaluasi pengenalan 4 kelas penyakit yang berbeda, yaitu *Common Rust, Gray Leaf Spot, Healthy*, dan *Northern Leaf Bligh*, masing-masing diuji sebanyak 5 kali. Eksperimen menggunakan set data gambar yang sama untuk setiap kelas. Tujuan eksperimen ini adalah untuk membandingkan kinerja dan akurasi dalam mengidentifikasi penyakit pada daun jagung. Metode eksperimen yang sistematis dan objektif diterapkan untuk memperoleh pemahaman yang lebih mendalam tentang kelebihan dan kekurangan masing-masing kelas dalam konteks identifikasi penyakit pada tanaman jagung, sebagai berikut:

1) Pengujian Identifikasi *Common Rust*
   a) Pengujian Identifikasi *Commone Rust* Menggunakan ResNet152V2

Gambar 4.28 Hasil Pengujian Identifikasi *Commone Rust* ResNet152V2

Berdasarkan gambar 4.28 yang menunjukkan hasil identifikasi penyakit daun jagung, sistem ini mendeteksi *common rust* sebanyak 5 kali dengan tingkat akurasi 100%. Artinya, model CNN yang digunakan sangat efektif dalam membedakan daun jagung sehat dari yang terinfeksi. Dalam penelitian ini, arsitektur ResNet152V2 digunakan karena kemampuannya mengatasi masalah *vanishing gradient* dan

memungkinkan pelatihan jaringan yang sangat dalam. Penggunaan ResNet152V2 terbukti memberikan hasil akurat dalam identifikasi penyakit daun jagung, termasuk *common rust*, menunjukkan kecocokan arsitektur ini untuk deteksi penyakit tanaman.

b) Pengujian Identifikasi *Commone Rust* Menggunakan AlexNet

Gambar 4.29 Hasil Pengujian Identifikasi *Commone Rust* AlexNet

Berdasarkan gambar 4.29 yang menunjukkan hasil identifikasi penyakit daun jagung, sistem ini mendeteksi adanya penyakit *common rust* sebanyak 5 kali dengan tingkat akurasi sebesar 92,2945210015869%. Deteksi ini dilakukan menggunakan arsitektur AlexNet dalam Metode CNN yang telah dikembangkan untuk menganalisis kondisi daun jagung. Hasil deteksi ini menegaskan kemampuan sistem dalam mengidentifikasi penyakit dengan tingkat akurasi yang tinggi, sehingga dapat digunakan sebagai alat bantu dalam pemantauan kesehatan tanaman jagung.

2) Pengujian Identifikasi *Gray Leaf Spot*

a) Pengujian Identifikasi *Gray Leaf Spot* ResNet152V2

Gambar 4.30 Hasil Pengujian Identifikasi *Gray Leaf Spot* ResNet152V2

Berdasarkan gambar 4.30, yang menunjukkan hasil identifikasi penyakit daun jagung, sistem deteksi menggunakan Metode *Convolutional Neural Network* (CNN) berhasil mengidentifikasi penyakit pada daun jagung sebagai *Gray Leaf Spot* sebanyak 5 kali dengan tingkat akurasi sebesar 99.9991469311523%. Hasil ini menunjukkan bahwa model CNN yang menggunakan arsitektur ResNet152V2 dalam sistem ini sangat efektif dan akurat dalam membedakan daun jagung yang sehat dari yang terinfeksi penyakit.

b) Pengujian Identifikasi *Gray Leaf* Menggunakan AlexNet

Gambar 4.31 Hasil Pengujian Identifikasi *Gray Leaf Spot* AlexNet

Berdasarkan gambar 4.31 menunjukkan hasil sistem deteksi penyakit pada daun jagung menggunakan *Convolutional Neural Network* (CNN) yang mengidentifikasi penyakit *Gray Leaf Spot* sebanyak 5 kali dengan akurasi 44.22%. Akurasi ini menunjukkan seberapa sering model mengklasifikasikan gambar dengan benar; nilai 44.22% menunjukkan bahwa model ini masih memiliki ruang untuk perbaikan agar lebih akurat. Evaluasi akurasi melibatkan perbandingan hasil prediksi model terhadap label sebenarnya dalam dataset uji. Angka akurasi yang rendah bisa disebabkan oleh beberapa faktor seperti dataset yang tidak cukup besar atau beragam, model yang belum optimal, atau fitur yang kurang relevan untuk deteksi penyakit spesifik.

3) Pengujian Identifikasi *Healthy*

a) Pengujian Identifikasi *Healthy* ResNet152V2

Gambar 4.32 Hasil Pengujian Identifikasi *Healthy* ResNet152V2

Berdasarkan gambar 4.32, yang menunjukkan hasil identifikasi penyakit daun jagung sistem deteksi penyakit pada tanaman jagung menggunakan Metode *Convolutional Neural Network* (CNN) telah berhasil mengidentifikasi bahwa daun jagung tersebut dalam kondisi sehat *Healthy* sebanyak 5 kali dengan tingkat akurasi sebesar 100.0%. Hasil ini menunjukkan bahwa model CNN yang digunakan dalam sistem

ini sangat efektif dan akurat dalam membedakan daun jagung yang sehat dari yang terinfeksi penyakit.

b) Pengujian Identifikasi *Healthy* Menggunakan AlexNet

Gambar 4.33 Hasil Pengujian Identifikasi *Healthy* AlexNet

Berdasarkan gambar 4.33, menunjukkan hasil deteksi penyakit pada daun jagung menggunakan Metode *Convolutional Neural Network* (CNN) yang mengidentifikasi daun tersebut sebagai *Healthy* sebanyak 5 kali dengan akurasi 79.99%. Akurasi ini mengindikasikan tingkat keberhasilan model dalam mengklasifikasikan gambar dengan benar, di mana nilai 79.99% menunjukkan performa yang cukup baik namun masih memerlukan peningkatan untuk mencapai tingkat akurasi yang lebih tinggi. Evaluasi akurasi melibatkan perbandingan hasil prediksi model terhadap label sebenarnya dalam dataset uji, dan angka akurasi yang lebih tinggi ini bisa disebabkan oleh dataset yang lebih representatif atau model yang lebih baik dalam mengidentifikasi fitur-fitur yang relevan untuk mendeteksi kondisi kesehatan tanaman.

4) Pengujian Identifikasi *Northern Leaf Blight*

a) Pengujian Identifikasi *Northern Leaf Blight* ResNet152V2

Gambar 4.34 Hasil Pengujian Idetifikasi *Northern Leaf Blight* ResNet152V2

Berdasarkan gambar 4.34, sistem deteksi penyakit pada tanaman jagung menggunakan Metode *Convolutional Neural Network* (CNN), dengan model ResNet152V2, berhasil mengidentifikasi bahwa daun jagung dalam kondisi sehat *Northern Leaf Blight* sebanyak 5 kali dengan tingkat akurasi 100.0%. Model ResNet152V2 terbukti sangat efektif dalam membedakan daun jagung yang sehat dari yang terinfeksi penyakit, menegaskan kemampuannya yang handal dalam analisis gambar dan deteksi penyakit tanaman dengan tingkat keakuratan yang tinggi.

b) Pengujian Identifikasi *Northern Leaf Blight* AlexNet

Gambar 4.35 Hasil Pengujian Idetifikasi *Northern Leaf Blight* AlexNet

Berdasarkan gambar 4.35, menunjukkan hasil deteksi penyakit pada daun jagung menggunakan Metode *Convolutional Neural Network* (CNN) dengan arsitektur AlexNet, yang mengidentifikasi penyakit *Northern Leaf Blight* sebanyak 5 kali dengan akurasi 99.99%. Tingkat akurasi yang sangat tinggi ini menunjukkan bahwa model AlexNet berhasil mengenali fitur-fitur yang khas dari penyakit ini dengan sangat baik. AlexNet, dengan delapan lapisan belajarnya, menggunakan fungsi aktivasi ReLU, dropout untuk mencegah overfitting, dan max-pooling untuk mengurangi dimensi data, memungkinkan ekstraksi fitur yang efisien dan akurat. Proses deteksi melibatkan preprocessing gambar, pelatihan model dengan dataset yang terlabel, evaluasi performa model, dan penggunaan model untuk prediksi pada gambar baru, seperti yang ditunjukkan dalam gambar ini.

Berdasarkan pengujian identifikasi terhadap setiap *class* sebanyak 5 kali percobaan dengan menggunakan arsitektur ResNet152V2 dan AlexNet, berikut adalah hasil yang dijelaskan pada tabel 4.2 :

Tabel 4.2 Hasil Pengujian Identifikasi

Class dataset	Rata-rata Hasil Pengujian Arsitektur ResNet152V2	Rata-rata Hasil Pengujian Arsitektur AlexNet
Common Rust	100%	92,29%
Gray Leaf Spot	99.99%	44,22%
Healthy	100%	79,99%
Northern Leaf Blight	100%	99,99%
Rata-rata *Accuracy*	99,9975%	79,1225%

Berdasarkan hasil pengujian pada tabel 4.2 dapat disimpulkan bahwa ResNet152V2 secara konsisten menunjukkan akurasi yang sangat tinggi dengan rata-rata mencapai 99.9975%, menjadikannya pilihan yang sangat baik

untuk tugas-tugas klasifikasi yang memerlukan presisi tinggi. Di sisi lain, meskipun AlexNet menunjukkan beberapa hasil yang baik dengan akurasi tertinggi 99.99%, rata-rata akurasinya adalah 79.1225% dengan variasi yang lebih besar, menandakan kinerja yang kurang stabil. Secara keseluruhan, ResNet152V2 lebih andal dibandingkan dengan AlexNet untuk identifikasi kelas pada dataset yang diuji.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian identifikasi yang membandingkan model ResNet152V2 dan AlexNet, penggunaan arsitektur *Convolutional Neural Network* (CNN) menunjukkan performa yang berbeda dalam mengidentifikasi penyakit pada daun jagung. ResNet152V2 secara konsisten menunjukkan akurasi yang sangat tinggi pada semua kelas dataset yang diuji, dengan rata-rata akurasi mencapai 99.9975%. Angka ini menunjukkan bahwa ResNet152V2 sangat efektif dalam mengidentifikasi setiap kelas dengan tingkat kesalahan yang sangat rendah, menjadikannya sebagai pilihan yang sangat baik untuk tugas-tugas klasifikasi yang memerlukan presisi tinggi. Keandalan dan konsistensi akurasi tinggi ini menegaskan keunggulan ResNet152V2 dalam identifikasi penyakit pada daun jagung.

Di sisi lain, performa AlexNet menunjukkan variasi yang lebih besar dalam akurasinya, dengan rata-rata akurasi sebesar 79.1225%. Meskipun AlexNet masih menunjukkan kemampuan yang memadai, akurasi tertingginya tercatat pada kelas *Northern Leaf Blight* dengan 99.99%, sedangkan akurasi terendahnya pada kelas *Gray Leaf Spot* hanya 44.22%. Hal ini mengindikasikan bahwa AlexNet memiliki kinerja yang kurang stabil dan mungkin memerlukan penyesuaian lebih lanjut atau pelatihan yang lebih intensif untuk meningkatkan akurasinya pada beberapa kelas dataset. Secara keseluruhan, hasil ini menunjukkan bahwa ResNet152V2 adalah arsitektur yang lebih andal dibandingkan dengan AlexNet untuk tugas identifikasi penyakit pada daun jagung.

5.2 Saran

Berdasarkan kesimpulan tersebut saran untuk penelitian selanjutnya yaitu :

1. Menambahkan dataset dengan dengan lebih banyak gambar yang bervariasi untuk setiap kelas penyakit penting untuk meningkatkan

representasi dan performa model. Fokus khusus perlu diberikan pada kasus dengan akurasi rendah, seperti deteksi *Gray Leaf Spot* menggunakan AlexNet, untuk memastikan model dapat mengenali fitur-fitur spesifik dengan lebih baik.

2. Menambahkan metode augmentasi data seperti rotasi, *flipping*, dan perubahan kecerahan dapat meningkatkan variasi dataset dan membuat model lebih *robust*. Teknik regularisasi seperti *dropout* dan *weight decay* dapat mengatasi *overfitting*, meningkatkan performa model pada data uji dan dunia nyata.
3. Menambahkan pertimbangan pada pengembangan model hybrid yang menggabungkan kelebihan dari beberapa arsitektur untuk mencapai akurasi lebih tinggi dan meningkatkan keandalan sistem deteksi penyakit pada tanaman.

DAFTAR PUSTAKA

[1] D. Iswantoro and D. Handayani UN, "Klasifikasi Penyakit Tanaman Jagung Menggunakan Metode Convolutional Neural Network (CNN)," *J. Ilm. Univ. Batanghari Jambi*, vol. 22, no. 2, p. 900, 2022, doi: 10.33087/jiubj.v22i2.2065.

[2] Q. N. Azizah, "Klasifikasi Penyakit Daun Jagung Menggunakan Metode Convolutional Neural Network AlexNet," *J. Tek. Inform.*, vol. 2, no. 1, pp. 28–33, 2023, doi: 10.56211/sudo.v2i1.227.

[3] M. Wafa Akhyari, A. Suyoto, and F. Wahyu Wibowo, "Klasifikasi Penyakit Pada Daun Jagung Menggunakan Convolutional Neural Network," *J. Inf. J. Penelit. dan Pengabdi. Masyarakat.*, vol. 7, no. 2, pp. 12–15, 2021, [Online]. Available: https://github.com.

[4] C. N. Wardayanti, "Identifikasi Jenis Penyakit Daun Tomat Menggunakan Algoritma CNN (Convolutional Neural Network)," Universitas Muhammadiyah Ponorogo, 2023.

[5] A. J. Rozaqi, A. Sunyoto, and M. rudyanto Arief, "Deteksi Penyakit Pada Daun Kentang Menggunakan Pengolahan Citra dengan Metode Convolutional Neural Network," *Creat. Inf. Technol. J.*, vol. 8, no. 1, p. 22, 2021, doi: 10.24076/citec.2021v8i1.263.

[6] F. Sulistiyana and S. Anardani, "Aplikasi Deteksi Penyakit Tanaman Jagung Dengan Convolutional Neural Network dan Support Vector Machine," *Semin. Nas. Teknol. Inf. dan Komunikasi-2023*, vol. 6, no. 1, pp. 423–432, 2023.

[7] B. Widianto, E. Utami, and D. Ariatmanto, "Identifikasi Penyakit Tanaman Jagung Berdasarkan Citra Daun Menggunakan Convolutional Neural Network," *Techno.Com*, vol. 22, no. 3, pp. 599–608, 2023, doi: 10.33633/tc.v22i3.8425.

[8] A. B. Prakosa, Hendry, and R. Tanone, "Implementasi Model Deep Learning Convolutional Neural Network (CNN) Pada Citra Penyakit Daun Jagung Untuk Klasifikasi Penyakit Tanaman," *J. Pendidik. Teknol. Inf.*, vol. 6, no.

1, pp. 107–116, 2023.

[9] A. Yoggyanto, A. Maulana, and D. A. Tri Cahyo, “Penerapan Metode Convolutional Neural Network (CNN) Dalam Klasifikasi Penyakit Tanaman Jagung,” *Pros. Semin. Nas. Teknol. Dan Sains*, vol. 3, no. 2022, pp. 251–256, 2024.

[10] M. I. Prasetyo Raharjo, “Klasifikasi Citra Penyakit Daun Jagung Dengan Algoritma Convolutional Neural Network,” UPN Veteran Jawa Timur, 2022.

[11] N. Hidayah, A. N. Istiani, and A. Septiani, “Pemanfaatan jagung (Zea mays) sebagai bahan dasar pembuatan keripik jagung untuk meningkatkan perekonomian masyarakat di desa panca tunggal,” *J. Pengabdi. Masy.*, vol. 1, no. 1, pp. 42–48, 2020, [Online]. Available: http://www.ejournal.radenintan.ac.id/index.php/ajpm/article/view/6181

[12] A.-N. Sharkawy, “Principle of Neural Network and Its Main Types: Review,” *J. Adv. Appl. Comput. Math.*, vol. 7, pp. 8–19, 2020.

[13] N. S. Putra, B. F. Hutabarat, and U. Khaira, “Implementasi Algoritma Convolutional Neural Network Untuk Identifikasi Jenis Kelamin Dan Ras,” *Decod. J. Pendidik. Teknol. Inf.*, vol. 3, no. 1, pp. 82–93, 2023, doi: 10.51454/decode.v3i1.123.

[14] M. Memari, M. Shekaramiz, M. A. S. Masoum, and A. C. Seibi, “Data Fusion and Ensemble Learning for Advanced Anomaly Detection Using Multi-Spectral RGB and Thermal Imaging of Small Wind Turbine Blades,” *Energies*, vol. 17, no. 3, pp. 1–29, 2024, doi: 10.3390/en17030673.

[15] K. H. Hanif, N. R. Muntiari, and P. A. Ramadhani, “Penerapan Metode Certainty Factor untuk Mendiagnosa Penyakit Preekslamsia pada Ibu Hamil dengan Menggunakan Bahasa Pemrograman Python,” *Insect (Informatics Secur. J. Tek. Inform.*, vol. 7, no. 2, pp. 63–71, 2022, doi: 10.33506/insect.v7i2.1818.

[16] Muhammad Romzi and B. Kurniawan, “Pembelajaran Pemrograman Python Dengan Pendekatan Logika Algoritma,” *JTIM J. Tek. Inform. Mahakarya*, vol. 03, no. 2, pp. 37–44, 2020.

[17] S. Panjaitan, C. Sitepu, and J. Sinaga, “Deteksi Jerawat Menggunakan

Arsitektur YOLOV3," *J. Ekon. Sos. Hum.*, vol. 4, no. 6, pp. 1–6, 2023, [Online]. Available: https://jurnalintelektiva.com/index.php/jurnal/article/view/929

[18] R. K. Ngantung and M. A. I. Pakereng, "Model Pengembangan Sistem Informasi Akademik Berbasis User Centered Design Menerapkan Framework Flask Python," *J. Media Inform. Budidarma*, vol. 5, no. 3, p. 1052, 2021, doi: 10.30865/mib.v5i3.3054.

[19] I. Budiman, S. Saori, R. N. Anwar, Fitriani, and M. Y. Pangestu, "Analisis Pengendalian Mutu Di Bidang Industri Makanan (Studi Kasus: UMKM Mochi Kaswari Lampion Kota Sukabumi)," *J. Inov. Penelit.*, vol. 1, no. 10, pp. 2185–2190, 2021.

[20] N. Fadlia and R. Kosasih, "Klasifikasi Jenis Kendaraan Menggunakan Metode Convolutional Neural Network (Cnn)," *J. Ilm. Teknol. dan Rekayasa*, vol. 24, no. 3, pp. 207–215, 2019, doi: 10.35760/tr.2019.v24i3.2397.