

TUGAS AKHIR

PERBANDINGAN MODEL ARSITEKTUR CNN DALAM
KLASIFIKASI PENYAKIT PADA DAUN TANAMAN TOMAT



DISUSUN OLEH:

ARDIAN SYAHPUTRA

NPM: 193510526

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS ISLAM RIAU
PEKANBARU
2025

KATA PENGANTAR



Puji syukur kita panjatkan kehadiran Allah SWT yang telah melimpahkan rahmat dan karunianya sehingga penulis dapat menyelesaikan penelitian yang berjudul **“PERBANDINGAN MODEL ARSITEKTUR CNN DALAM KLASIFIKASI PENYAKIT PADA DAUN TANAMAN TOMAT”**.

Terimakasih yang kepada orang tua yang saya sayangi yang telah memberikan kepercayaan dan segenap kasih sayangnya serta perhatian moril maupun material. Semoga Allah SWT selalu melimpahkan Rahmat, Kesehatan, Karunia, dan keberkahan di dunia maupun di akhirat atas budi baik yang telah di berikan kepada saya.

Penulis dapat menyelesaikan skripsi ini tidak lepas dari bantuan dan bimbingan dari bapak/ibu dosen. Oleh karena itu, pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada yang terhormat :

1. Orang tua dan keluarga saya yang telah memberi saya semangat dan motivasi dalam mengerjakan skripsi ini.
2. Seluruh Dosen Program Studi Teknik Informatika yang mendidik serta memberi arahan hingga proposal skripsi ini dapat diselesaikan.
3. Rekan-rekan kelas D angkatan 2019 Teknik Informatika UIR, yang telah memberikan semangat dan motivasi selama penyusunan proposal skripsi ini.
4. Dan terakhir, untuk semua pihak yang telah membantu dalam menyelesaikan proposal skripsi ini.

Terlepas dari semua itu, Penulis menyadari sepenuhnya bahwa masih ada kekurangan baik dari segi susunan kalimat maupun tata bahasanya. Oleh karena itu dengan tangan terbuka Penulis menerima segala saran dan kritik dari pembimbing agar Penulis dapat memperbaiki Proposal Penelitian Skripsi ini.

Akhir kata, dengan segala kerendahan hati dan dengan segala harapan semoga penelitian ini bermanfaat bagi semua pihak.

Pekanbaru, 24 November 2024

Penulis,

Ardian Syahputra

NPM. 193510526

ABSTRAK

Pertanian merupakan salah satu sektor vital dalam perekonomian, termasuk di Indonesia. Salah satu komoditas penting adalah tomat, yang permintaannya terus meningkat. Namun, tanaman tomat rentan terhadap berbagai jenis penyakit yang dapat menurunkan kualitas dan produktivitas. Untuk mengatasi masalah ini, dilakukan klasifikasi penyakit pada daun tomat menggunakan metode *Convolutional Neural Network (CNN)*. Penelitian ini membandingkan tiga arsitektur *CNN*, yaitu *VGG16*, *VGG19*, dan *Xception*. Model dilatih menggunakan dataset dari *website Kaggle* yang terdiri dari 4000 citra daun tomat, terbagi ke dalam empat kelas: *bacterial spot*, *late blight*, *yellow leaf curl virus*, dan daun tomat sehat. Pelatihan model dilakukan dengan kombinasi beberapa *hyperparameter*, seperti jumlah *epoch*, *learning rate*, dan *optimizer*. Untuk mempermudah proses pengujian, model diimplementasikan ke dalam aplikasi *web* berbasis Flask. Hasil pengujian menunjukkan bahwa model *VGG16* memberikan performa terbaik dengan akurasi 99,52%, disusul oleh *VGG19* dengan akurasi 98,08%, dan *Xception* dengan akurasi 97,12%. Penelitian ini menunjukkan bahwa arsitektur *CNN*, khususnya *VGG16*, efektif dalam mengklasifikasikan penyakit daun tomat secara akurat.

Kata Kunci: Penyakit Daun Tomat, *CNN*, *VGG16*, *VGG19*, *Xception*, Klasifikasi Citra, *Deep Learning*

ABSTRACT

Agriculture is one of the vital sectors in the economy of many countries, including Indonesia. Among key agricultural products, tomatoes are in high demand; however, tomato plants are vulnerable to various diseases that can reduce crop quality and yield. To address this issue, disease classification on tomato leaves is carried out, as leaf appearance is an essential indicator of plant health. This study aims to compare several Convolutional Neural Network (CNN) architectures that have shown effectiveness in image-based classification tasks. The research evaluates three CNN models VGG16, VGG19, and Xception using a combination of hyperparameters such as the number of epochs, learning rate, and optimizer. The dataset used in this study was obtained from Kaggle, consisting of 4,000 images divided into four classes: bacterial spot, late blight, yellow leaf curl virus, and healthy leaves. To support the testing process, the models were deployed through a web-based application built using the Flask framework. The experimental results demonstrate that the VGG16 model achieved the best performance with an accuracy of 99.52%, followed by VGG19 at 98.08%, and Xception at 97.12%. These results indicate that VGG16 provides consistent and accurate classification and holds promise for real-world application in plant disease detection systems.

Keywords: *Tomato Leaf Disease, CNN, VGG16, VGG19, Xception, Image Classification, Deep Learning*

DAFTAR ISI

KATA PENGANTAR.....	i
ABSTRAK	iii
DAFTAR ISI.....	v
DAFTAR GAMBAR	viii
DAFTAR TABEL	x
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Identifikasi Masalah	2
1.3 Rumusan Masalah	2
1.4 Batasan Penelitian	3
1.5 Tujuan Penelitian.....	4
1.6 Manfaat Penelitian	4
BAB II LANDASAN TEORI	6
2.1 Tinjauan Pustaka	6
2.2 Dasar Teori	10
2.2.1 Sistem.....	10
2.2.2 Klasifikasi Data	11
2.2.3 <i>Convolutional Neural Network</i>	11
2.2.4 Augmentasi Data	17
2.2.5 <i>Python</i>	20
2.2.6 <i>Tensorflow</i>	20
2.2.7 <i>Keras</i>	21
2.2.8 <i>Matplotlib</i>	21

2.2.9 <i>VGG16</i>	21
2.2.10 <i>VGG19</i>	22
2.2.11 <i>Xception</i>	23
2.2.12 <i>Confusion Matrix</i>	24
BAB III METODE PENELITIAN	27
3.1 Jenis Penelitian.....	27
3.2 Analisa Sistem.....	27
3.3 Tahapan Penelitian	29
3.3.1 Pengumpulan Data	29
3.3.2 Pembagian Dataset	30
3.3.3 <i>Preprocessing Data</i>	31
3.3.4 <i>Training Dataset</i>	32
3.3.5 Rencana Pengujian <i>Hyperparameter</i>	38
3.3.6 Evaluasi Model.....	39
3.4 Rancangan Sistem	39
3.4.1 <i>Flowchart</i> Sistem	39
3.4.2 Rancangan Antarmuka Sistem	40
BAB IV HASIL DAN PEMBAHASAN.....	42
4.1 Hasil pengolahan data penelitian	42
4.2 Pembagian Dataset dan <i>Preprocessing</i>	43
4.3 Rancangan Model <i>CNN</i>	46
4.4 Pengujian Model	47
4.4.1 Pengujian Model <i>VGG16</i>	47
4.4.2 Pengujian Model <i>VGG19</i>	50
4.4.3 Pengujian Model <i>Xception</i>	53

4.5 Evaluasi Model.....	57
4.5.1 Evaluasi Model <i>VGG16</i>	57
4.5.2 Evaluasi Model <i>VGG19</i>	60
4.5.3 Evaluasi Model <i>Xception</i>	63
4.5.4 Hasil Evaluasi.....	67
4.6 Pengaruh <i>HyperParameter</i> pada Model	68
4.6.1 Jumlah <i>Training Epoch</i>	68
4.6.2 <i>Learning Rate</i>	69
4.6.3 <i>Optimizer</i>	70
4.7 Pembuatan REST API Klasifikasi Penyakit Daun Tomat.....	72
4.8 Hasil Klasifikasi Citra dan <i>Testing</i> Menggunakan <i>Website</i>	74
BAB V KESIMPULAN DAN SARAN	80
5.1 Kesimpulan	80
5.2 Saran.....	81
DAFTAR PUSTAKA.....	82

DAFTAR GAMBAR

Gambar 2. 1 Arsitektur <i>Convolutional Neural Network</i>	11
Gambar 2. 2 Proses Konvolusi	12
Gambar 2. 3 <i>Stride</i> Konvolusi	13
Gambar 2. 4 Proses penambahan <i>padding</i>	14
Gambar 2. 5 Plot ReLU	15
Gambar 2. 6 Plot <i>Softmax</i>	16
Gambar 2. 7 Sebelum dan sesudah dilakukan <i>shift</i>	18
Gambar 2. 8 Proses <i>Shear</i>	18
Gambar 2. 9 Contoh proses <i>Zoom</i>	19
Gambar 2. 10 Contoh <i>Horizontal Flip</i>	19
Gambar 2. 11 Model Arsitektur <i>VGG16</i>	22
Gambar 2. 12 Model Arsitektur <i>VGG19</i>	23
Gambar 2. 13 Model Arsitektur <i>Xception</i>	24
Gambar 3. 1 Alur Penelitian	29
Gambar 3. 2 Hasil Augmentasi Citra.....	32
Gambar 3. 3 Arsitektur <i>VGG16</i>	33
Gambar 3. 4 Arsitektur <i>VGG19</i>	33
Gambar 3. 5 Arsitektur <i>Xception</i>	34
Gambar 3. 6 Proses Ekstraksi fitur	35
Gambar 3. 7 Hasil Ektrak fitur	35
Gambar 3. 8 Proses <i>Pooling</i>	36
Gambar 3. 9 Hasil <i>Pooling</i>	37
Gambar 3. 10 Cara Kerja <i>Softmax Activation</i>	38
Gambar 3. 11 <i>Flowchart</i> Sistem	39
Gambar 3. 12 <i>Interface</i> Sistem	40
Gambar 4. 1 Pembuatan Label	42
Gambar 4. 2 Pembagian <i>Dataset</i>	43
Gambar 4. 3 Hasil <i>Resize</i>	44
Gambar 4. 4 Teknik Augmentasi	44

Gambar 4. 5 Hasil Augmentasi.....	45
Gambar 4. 6 Pembuatan Model	46
Gambar 4. 7 <i>Hyperparameter</i>	47
Gambar 4. 8 Hasil <i>Training</i> Dan <i>Validation VGG16</i>	48
Gambar 4. 9 <i>Loss VGG16</i>	48
Gambar 4. 10 <i>Accuracy VGG16</i>	49
Gambar 4. 11 Hasil <i>Training</i> dan <i>Validation VGG19</i>	50
Gambar 4. 12 <i>Loss VGG19</i>	51
Gambar 4. 13 <i>Accuracy VGG19</i>	52
Gambar 4. 14 Hasil <i>Training</i> dan <i>Validation Xception</i>	54
Gambar 4. 15 <i>Loss Xception</i>	54
Gambar 4. 16 <i>Accuracy Xception</i>	56
Gambar 4. 17 Hasil <i>Confusion Matrix VGG16</i>	58
Gambar 4. 18 <i>Classification Report VGG16</i>	59
Gambar 4. 19 Hasil <i>Confusion Matrix VGG19</i>	61
Gambar 4. 20 <i>Classification Report VGG19</i>	62
Gambar 4. 21 Hasil <i>Confusion Matrix Xception</i>	64
Gambar 4. 22 <i>Classification Report Xception</i>	65
Gambar 4. 23 <i>Load Model</i> dan Mendefinisikan Label Tiap Kelas.....	73
Gambar 4. 24 <i>Preprocessing Input Image</i>	73
Gambar 4. 25 <i>Upload Image</i>	73
Gambar 4. 26 <i>Predict Image</i>	74
Gambar 4. 27 Halaman Utama	75
Gambar 4. 28 Halaman Hasil Klasifikasi	75

DAFTAR TABEL

Tabel 2. 1 Penelitian Sebelumnya yang relevan	6
Tabel 2. 2 <i>Confusion Matrix</i>	25
Tabel 3. 1 Pembagian Data	30
Tabel 3. 2 Teknik Augmentasi Data.....	31
Tabel 3. 3 Proses <i>Pooling</i>	37
Tabel 3. 4 Proses <i>Flattening</i>	38
Tabel 3. 5 Parameter Pada Pengujian <i>hyperparameter</i>	39
Tabel 4. 1 Teknik Augmentasi	45
Tabel 4. 2 <i>Confusion Matrix VGG16</i>	59
Tabel 4. 3 <i>Confusion Matrix VGG19</i>	62
Tabel 4. 4 <i>Confusion Matrix Xception</i>	65
Tabel 4. 5 Hasil Evaluasi	67
Tabel 4. 6 Pengujian <i>Epoch</i>	69
Tabel 4. 7 <i>Pengujian Learning Rate</i>	70
Tabel 4. 8 Pengujian <i>Optimizer</i>	71
Tabel 4. 9 Rancangan <i>Request</i> dan Respon	72
Tabel 4. 10 <i>Testing Model VGG16 Via Website</i>	76
Tabel 4. 11 <i>Classification Report Testing VGG16</i>	76
Tabel 4. 12 <i>Testing Model VGG19 Via Website</i>	77
Tabel 4. 13 <i>Classification Report Testing VGG19</i>	77
Tabel 4. 14 <i>Testing Model Xception Via Website</i>	78
Tabel 4. 15 <i>Classification Report Testing Xception</i>	78

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pertanian merupakan salah satu sektor vital dalam perekonomian di banyak negara, termasuk Indonesia. Salah satu tanaman yang menjadi komoditas utama dalam bidang ini yaitu tanaman tomat yang memiliki daya jual dan permintaan pasar yang tinggi. Dengan meningkatnya permintaan pasar terhadap tomat petani berusaha agar dapat menghasilkan tomat dengan kualitas yang bagus agar dapat bersaing dalam persaingan pasar. Namun timbul masalah dalam pembudidayaan tomat yaitu tanaman ini sering kali terancam berbagai jenis penyakit yang menyerang pada bagian daun, seperti bercak bakteri (*bacterial spot*), busuk daun akhir (*late blight*), dan virus kuning-keriting pada daun (*yellow leaf curl virus*) sehingga dapat mengurangi kualitas maupun kuantitas dari tomat.

Beruntung seiring dengan perkembangan teknologi, khususnya dalam bidang kecerdasan buatan (*Artificial Intelligence*), telah banyak diterapkan metode pengolahan citra (*image processing*) dan pembelajaran mesin (*machine learning*) dalam identifikasi penyakit tanaman. Salah satu metode yang cukup populer adalah *Convolutional Neural Network (CNN)*, yang terbukti efektif dalam klasifikasi citra secara otomatis. Dalam implementasinya, terdapat berbagai macam arsitektur *CNN* yang memiliki karakteristik dan performa yang berbeda-beda, misalnya *VGG16*, *VGG19*, dan *Xception*.

Penelitian terbaru menunjukkan bahwa model-model *Convolutional Neural Network (CNN)* dapat memberikan hasil yang sangat akurat dalam deteksi penyakit tanaman. Misalnya, penelitian oleh Nurdin et al (2024) menunjukkan bahwa model *CNN* dapat mendeteksi penyakit daun tanaman tomat dengan akurasi yang tinggi yaitu mencapai 93.8% dengan bantuan *optimizer Adam*. Selain itu, penelitian yang dilakukan oleh Rohma, A. M. N. (2024) mengungkapkan bahwa model *CNN* dapat dioptimalkan untuk mendeteksi penyakit dengan lebih cepat dan efisien dibandingkan metode tradisional. Pencegahan dapat dilakukan dengan

mengidentifikasi dan mengklasifikasi kan jenis penyakit yang ada pada tanaman tomat dengan menggunakan metode *CNN*.

Penelitian ini dilakukan untuk membandingkan kinerja dari arsitektur *CNN* yaitu *VGG16* , *VGG19* dan *Xception* dalam membangun model klasifikasi yang mampu membedakan daun tanaman tomat yang sehat dan yang terserang penyakit. Dengan memanfaatkan dataset citra daun tanaman tomat yang diperoleh dari *platform Kaggle*. Penelitian ini bertujuan untuk mengevaluasi model mana yang paling optimal dari segi akurasi, presisi, *recall*, dan *f1-score*. Selain itu, studi kasus ini juga dilakukan dengan mengambil data pendukung dari lahan pertanian Fakultas Pertanian Universitas Islam Riau sebagai upaya penerapan teknologi di lapangan.

1.2 Identifikasi Masalah

Berdasarkan latar belakang diatas, dapat diidentifikasi beberapa masalah seperti berikut :

1. Tanaman tomat rentan terserang berbagai jenis penyakit daun, seperti *bacterial spot*, *late blight* dan *yellow leaf curl virus*, yang dapat menurunkan kualitas dan kuantitas hasil panen.
2. Keterbatasan dalam identifikasi dan klasifikasi dini pada penyakit daun tanaman tomat. Metode konvensional dalam deteksi dan klasifikasi penyakit sering kali kurang efisien karena memerlukan tenaga kerja yang intensif.
3. Meskipun metode *Convolutional Neural Network (CNN)* telah terbukti efektif dalam klasifikasi citra penyakit tanaman, belum diketahui secara pasti arsitektur model *CNN* mana yang memberikan performa terbaik untuk klasifikasi penyakit daun tanaman tomat.

1.3 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, maka dapat dirumuskan masalah berikut :

1. Bagaimana kinerja masing-masing arsitektur *CNN*, yaitu *VGG16*, *VGG19*, dan *Xception*, dalam mengklasifikasikan penyakit pada daun tanaman tomat?
2. Seberapa efektif penerapan model *CNN* terhadap *dataset* citra daun tanaman tomat yang diperoleh dari platform *Kaggle*.
3. Apakah model-model yang digunakan dapat mengklasifikasi penyakit yang berbeda dari *dataset* yang tersedia?
4. Arsitektur *CNN* mana yang memberikan hasil paling optimal dalam hal akurasi, presisi, *recall* dan *f1-score* pada klasifikasi citra daun tanaman tomat?

1.4 Batasan Penelitian

Dalam penelitian ini, obyek penelitian dibatasi dengan ruang lingkup sebagai berikut:

1. Penelitian ini hanya membandingkan tiga arsitektur model *CNN*, yaitu *VGG16*, *VGG19*, dan *Xception* dalam klasifikasi penyakit pada daun tanaman tomat
2. Citra yang dapat di bandingkan adalah berupa citra yang memiliki ekstensi seperti *png*, *jpg*, *jpeg*,
3. Jenis penyakit yang diklasifikasikan terbatas pada penyakit daun tomat yang telah terlabel secara jelas pada *dataset*, seperti *bacterial spot*, *yellow leaf curl virus*, *late blight* dan kategori daun sehat.
4. Evaluasi performa model dilakukan berdasarkan metrik akurasi, presisi, *recall* dan *f1-score*.
5. Model *CNN* yang telah dilatih akan diuji menggunakan *framework Flask* sebagai antarmuka web sederhana untuk menguji *input* citra secara langsung dan menampilkan hasil klasifikasi.

1.5 Tujuan Penelitian

Adapun tujuan penelitian ini adalah sebagai berikut:

1. Penelitian ini bertujuan untuk membandingkan dan mengevaluasi kinerja tiga model arsitektur *CNN*, yaitu *VGG16*, *VGG19* dan *Xception*, dalam konteks klasifikasi penyakit pada daun tanaman tomat. Aspek yang perlu dipertimbangkan meliputi metrik akurasi, presisi, *recall* dan *f1-score* dari masing-masing model.
2. Untuk menguji hasil pelatihan model *CNN* melalui implementasi sederhana menggunakan *Framework Flask*, yang memungkinkan pengguna mengunggah citra daun tomat dan mendapatkan hasil klasifikasi secara langsung.
3. Menyediakan Informasi untuk Pengembangan Sistem Deteksi Penyakit Tanaman untuk memberikan informasi dan rekomendasi yang berguna bagi pengembangan sistem deteksi yang lebih lanjut.
4. Pembuatan model data yang nantinya dapat digunakan oleh pihak lain dalam membuat sistem deteksi yang jauh lebih praktis dan efisien

1.6 Manfaat Penelitian

1. Menyediakan hasil perbandingan arsitektur *CNN*
 Penelitian ini memberikan analisis perbandingan performa tiga arsitektur *CNN* yaitu (*VGG16*, *VGG19*, dan *Xception*) dalam klasifikasi citra daun tanaman tomat, dengan mengukur metrik seperti akurasi, presisi dan *recall*.
2. Menyediakan model data
 Peneliti berharap model data yang telah dibuat akan dapat digunakan untuk kepentingan klasifikasi penyakit pada daun tanaman tomat untuk kedepannya baik oleh orang-orang yang membutuhkan model untuk penelitian selanjutnya maupun orang-orang yang berada di bidang argoteknologi.
3. Kontribusi terhadap Pengetahuan Akademik dan Praktis

Penelitian ini berkontribusi pada literatur akademik dalam bidang *computer vision* dan *machine learning* dengan menambahkan wawasan baru tentang perbandingan model CNN untuk aplikasi klasifikasi penyakit tanaman. Hal ini juga memberikan panduan praktis bagi peneliti dan praktisi dalam memilih dan menerapkan model CNN yang tepat.

BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian-penelitian sebelumnya dijadikan sebagai landasan dalam pelaksanaan studi ini, guna memperkuat dan memperluas pemahaman teoritis yang digunakan dalam proses analisis. Beberapa hasil studi terdahulu diangkat sebagai referensi untuk menunjang dan memperkaya kajian yang dilakukan. Adapun uraian penelitian terdahulu yang relevan disajikan dalam bentuk beberapa jurnal yang berkaitan dengan topik penelitian ini :

Tabel 2. 1 Penelitian Sebelumnya yang relevan

No	Nama Penulis	Judul	Metode	Hasil Penelitian
1	Muhammad Ilham Rasyid dan Lulu Mawaddah Wisudawati. (2024)	Klasifikasi Hama Ulat Pada Citra Daun Sawi Berbasis Convolutional Neural Network Dengan Model Xception.	Convolutional Neural Network dengan menggunakan arsitektur Xception	Hasil dari klasifikasi penyakit pada tanaman daun sawi dengan model arsitektur Xception dengan uji coba pada pelatihan data dengan 600 data latih, 200 data uji dan 200 data validasi menghasilkan nilai akurasi paling tinggi sebesar 96%, nilai sensitifitas 96% dan nilai spesifisitas 97%.

2	Bella Dwi Mardiana, Wahyu Budi Utomo, Ulfah Nur Oktaviana, Galih Wasis Wicaksono, dan Agus Eko Minarno. (2023).	Herbal Leaves Classification Based on Leaf Image Using CNN Architecture Model VGG16.	Convolutional Neural Network dengan menggunakan arsitektur VGG16	Tujuan penelitian ini adalah untuk mengklasifikasikan 10 jenis daun tanaman herbal menggunakan arsitektur VGG16 dan didapati hasil akurasi meningkat dari penelitian sebelumnya yang hanya 82% menjadi 97%
3	Nining Putri Ningsih, Emi Suryadi, Lalu Darmawan Bakti, dan Bahtiar Imran. (2022).	Klasifikasi Penyakit Early Blight Dan Late Blight Pada Tanaman Tomat Berdasarkan Citra Daun Menggunakan Metode Cnn Berbasis Website.	Convolutional Neural Network	Dalam penelitian ini algoritma CNN menghasilkan akurasi yang tinggi, proses training data menggunakan learning rate 0,0001 dan batch size 20. Epoch 1 menghasilkan loss 98%, akurasi 53%, Recall 46%. Epoch 10 menghasilkan 20, loss 34%, akurasi 85%, recall 81%. Epoch 20 menghasilkan loss 22%, akurasi 94%,

				recall 95%. Epoch 100 menghasilkan loss 5%, akurasi 99%, dan recall 85%
4	Desi Intan Permatasari. (2024)	Implementasi Metode Convolutional Neural Network(CNN) Untuk Klasifikasi Tanaman Herbal Berdasarkan Citra Daun	Convolutional Neural Network	Penelitian ini menggunakan 1000 dataset daun yang terdiri dari 10 jenis daun, hasil yang didapatkan menggunakan metode CNN yaitu akurasi validasi yang mencapai 92,5% dan akurasi keseluruhan 93%.
5	Arief Saputro, Syahri Mu'min, Moch. Lutfi, dan Helmanita Putri. (2022)	Deep Transfer Learning Dengan Model Arsitektur VGG19 Untuk Klasifikasi Jenis Varietas Tanaman Lengkeng Berdasarkan Citra Daun	Transfer Learning Menggunakan model VGG16 yang sudah dilatih	Penelitian ini mengklasifikasi empat jenis varietas tanaman lengkeng dengan menggunakan arsitektur VGG16 yang sudah dilatih, dan mendapatkan hasil akurasi 79% dan hasil validasi yang mencapai 82%

6	Errin Dwi Ratnasari, Dhira Ananta Rudira dan Anom Surya Buana. (2024)	Klasifikasi Penyakit Daun Sawi Hijau Dengan Metode CNN	Convolutional Neural Network	Penelitian ini menggunakan 60 sampel data untuk proses validasi dan 38 diantara nya menunjukkan hasil yang cukup akurat ,sehingga dapat disimpulkan bahwa akurasi dari penelitian ini adalah 63%.
7	Kelvianto Husado, Charisini Lubis dan Zyad Rusdi. (2023)	Klasifikasi Tanaman Anggrek Menggunakan Convolutional Neural Network dengan Arsitektur VGG19	Convolutional Neural Network dengan arsitektur VGG19	Klasifikasi tanaman anggrek pada saat pelatihan mendapatkan hasil akurasi 99,48% namun pada saat data diuji mendapatkan akurasi sebesar 82% dan 81% pada akurasi validasi
8	Rangga Utama Putra, Habib Muhammad Ridwan, Ilham Abiansyah dan Tinuk Agustin. (2024)	Klasifikasi Daun Tomat Sehat dan Terserang Penyakit Menggunakan Metode Convolutional	Convolutional Neural Network dengan model deep, wide dan gabung deep dan wide	Dengan Menggunakan 400 citra daun tomat dan mengklasifikasi kedalam 4 kelas dengan menggunakan model wide mendapatkan

		Neural Network (CNN)		hasil akurasi uji sebesar 98,75% , menggunakan model deep menghasilkan 100% akurasi uji , dan gabungan kedua model tersebut memberikan akurasi sebesar 96,67%.
9	Ardian Syahputra	Perbandingan Model Arsitektur CNN Dalam Klasifikasi Penyakit Pada Daun Tanaman Tomat	Convolutional Neural Network dengan model VGG16, VGG19, dan Xception	Hasil akurasi uji menggunakan VGG16 sebesar 99.52%, menggunakan VGG19 sebesar 98,08% dan hasil dari Xception memberikan akurasi sebesar 97,12%

2.2 Dasar Teori

2.2.1 Sistem

Secara umum, sistem dapat dipahami sebagai sekumpulan elemen yang saling berkaitan dan bekerja sama secara terpadu untuk mencapai tujuan tertentu. Sistem terdiri dari komponen-komponen yang terorganisir dan saling berinteraksi, bergantung satu sama lain, serta membentuk satu kesatuan yang utuh dan terintegrasi (Marcelo D'Agostino et al. 2021).

Sedangkan menurut Xu Jianfeng et al(2023), sistem adalah kumpulan komponen yang saling berinteraksi secara dinamis dan terorganisir untuk mencapai tujuan tertentu dalam konteks teknologi informasi. Sistem tidak berdiri sendiri,

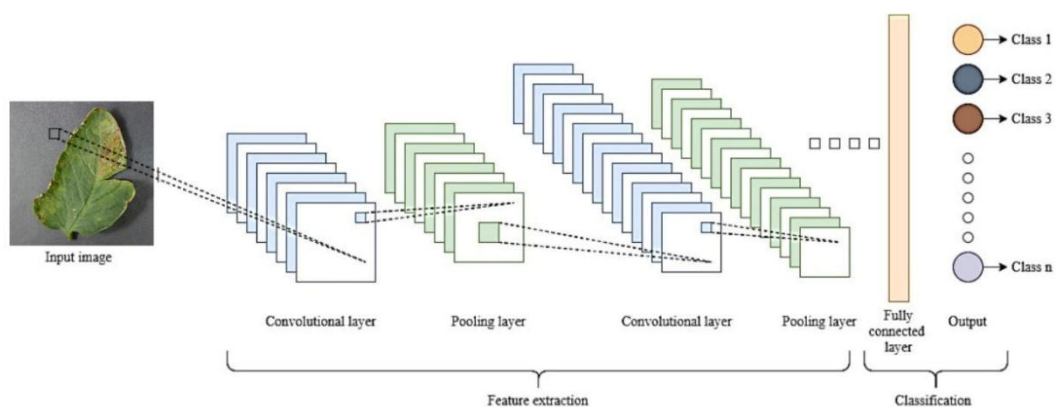
melainkan beroperasi dalam suatu lingkungan yang mempengaruhi dan dipengaruhi oleh sistem tersebut.

2.2.2 Klasifikasi Data

Klasifikasi menurut Naeem Seliya et al (2021), adalah proses membangun model dari data berlabel untuk mengidentifikasi atau memprediksi kelas dari data baru. Dalam konteks *big data*, metode seperti *one-class classification* sangat berguna untuk mengenali anomali atau data yang menyimpang dari pola umum. Klasifikasi biasanya digunakan dalam *machine learning* dan *data mining* untuk memprediksi kelas data baru yang tidak terlihat berdasarkan pola yang diidentifikasi dalam *training data*.

2.2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan pengembangan dari metode *Multi Layer Perceptron (MLP)* yang dirancang secara khusus untuk menangani data berdimensi dua, seperti gambar. Karena kemampuannya dalam membentuk arsitektur jaringan yang mendalam dan banyak digunakan dalam pemrosesan citra, *CNN* termasuk dalam kategori metode *Deep Neural Network*. Proses pembelajaran pada *CNN* melibatkan dua mekanisme utama, yaitu proses klasifikasi menggunakan *feedforward* dan penyesuaian bobot dengan algoritma *backpropagation* (Putra, 2023).

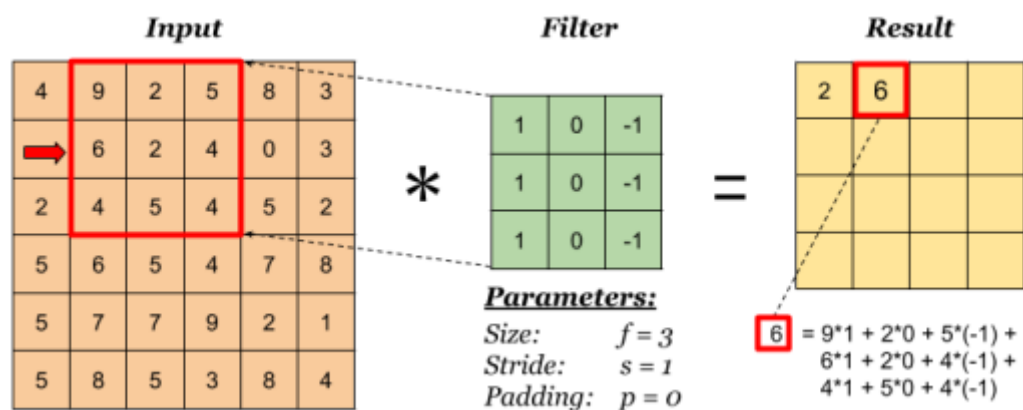


Gambar 2. 1 Arsitektur *Convolutional Neural Network*

a. Convolution Layer

Di dalam *Convolutional Layer*, sebuah matriks bernama *kernel* dipindahkan ke *input* matriks untuk membuat sebuah *feature map* untuk lapisan selanjutnya. Kita mengeksekusi operasi matematika yang di sebut konvolusi dengan menggeser *kernel* ke atas input matriks. Pada setiap lokasi sebuah elemen matriks akan dikalikan dengan *kernel* dan hasilnya akan dijumlahkan sehingga menjadi sebuah *feature map*.

Convolution adalah sebuah operasi linier yang secara khusus digunakan untuk keperluan pada bidang *image processing*, statistik dan fisika. *Convolution Layer* berguna untuk mengekstrak *feature* dari sebuah citra dan *output* dari *convolution layer* ini bisa disebut sebagai *feature map*.



Gambar 2. 2 Proses Konvolusi

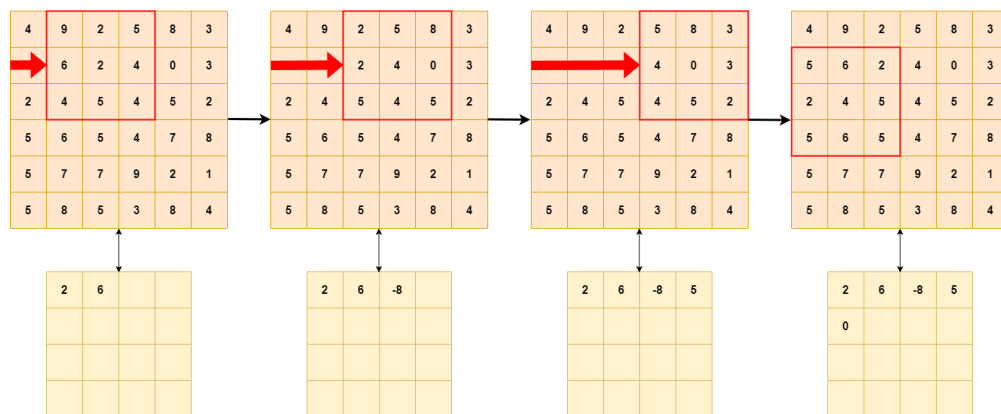
(Sumber : [researchgate.net](https://www.researchgate.net), 2021)

Dapat dilihat pada kotak berwarna oren yang merupakan input citra akan difilter dengan *kernel* berukuran 3x3 dari kiri atas ke kanan bawah sehingga menghasilkan *feature map* berukuran 4x4.

1. Stride

Berdasarkan ilustrasi di atas, *feature map* diperoleh melalui proses konvolusi antara citra input dengan beberapa filter berukuran 3x3. Proses ini dilakukan dengan mengalikan elemen-elemen pada matriks citra dengan elemen-elemen pada filter yang lebih kecil. Perhitungan dimulai dari sudut kiri atas matriks gambar. Setiap hasil perkalian elemen tersebut dijumlahkan untuk membentuk satu nilai

baru yang menjadi bagian dari *feature map*. Selanjutnya, filter akan digeser ke arah kanan sesuai dengan nilai *stride* yang ditentukan. *Stride* (S) adalah parameter yang menunjukkan seberapa jauh filter bergeser dalam satu langkah. Jika *stride* yang digunakan bernilai 1, maka pergeseran filter dilakukan satu piksel secara horizontal hingga mencapai batas, lalu dilanjutkan pergeseran vertikal ke bawah untuk proses berikutnya.



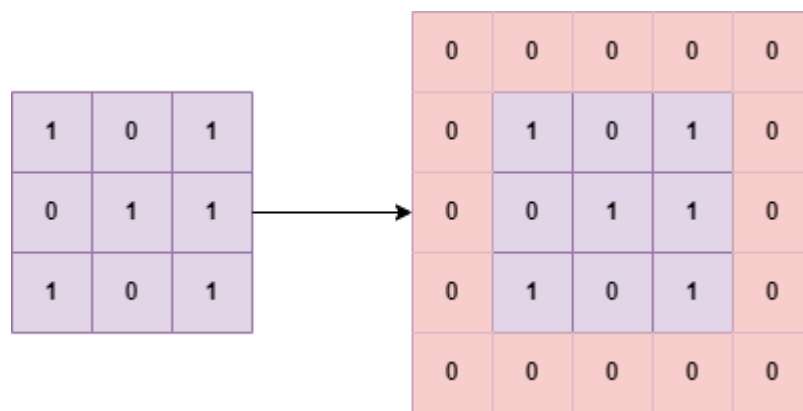
Gambar 2.3 *Stride* Konvolusi

Pada ilustrasi di atas, area pada matriks gambar yang sedang mengalami proses konvolusi ditunjukkan oleh kotak merah. Pada konvolusi pertama, sebagian matriks gambar akan diproses bersama kernel sesuai dengan ukuran volume kernel tersebut. Setelah nilai hasil konvolusi diperoleh, kernel kemudian digeser ke arah kanan sejauh nilai *stride* yang digunakan, dalam hal ini $stride = 1$. Penentuan nilai *stride* tidak dapat dilakukan secara sembarangan, karena harus disesuaikan dengan dimensi matriks gambar. Apabila nilai *stride* yang dipilih tidak sesuai dengan ukuran matriks, maka bisa dilakukan penyesuaian dengan menambahkan *padding* di sisi-sisi matriks, atau dengan mengurangi nilai *stride* agar hasil konvolusi tetap valid.

2. *Padding*

Padding merupakan parameter yang menentukan berapa banyak piksel bernilai nol yang ditambahkan di setiap sisi matriks *input*. Fungsi utama dari *padding* adalah untuk mengatur ukuran output dari *feature map* yang dihasilkan. Parameter *padding* (P) memiliki peran penting dalam proses konvolusi karena,

secara umum, hasil konvolusi memiliki dimensi yang lebih kecil dibandingkan dengan dimensi input awal. Hal ini dapat menyebabkan hilangnya sebagian informasi dari gambar, terutama jika ukuran filter yang digunakan cukup besar. Dengan menerapkan *padding*, ukuran output dapat disesuaikan agar tetap sama atau mendekati ukuran input awal. Hal ini memungkinkan penggunaan *convolution layer* yang lebih dalam tanpa kehilangan terlalu banyak informasi, sehingga proses ekstraksi fitur menjadi lebih optimal.



Gambar 2. 4 Proses penambahan *padding*

b. Fungsi Non-Linear (ReLU)

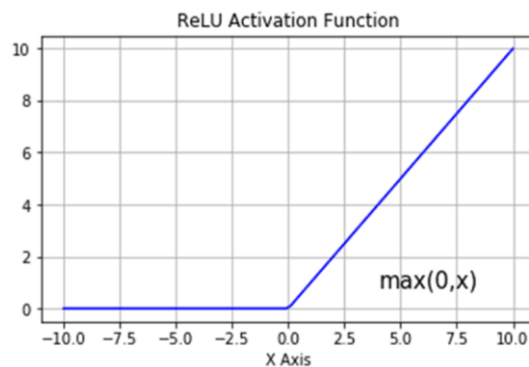
Fungsi aktivasi yang cukup terkenal adalah ReLU. ReLU diperkenalkan oleh Kuniyiko Fukushima pada tahun 1969 dalam konteks ekstraksi fitur *visual neural network* bertingkat. Dewasa ini ReLU adalah fungsi aktivasi yang paling banyak digunakan untuk *deep Convolutional Neural Network* (CNN) (Varshney & Singh, 2021). Berdasarkan temuan sebelumnya, penggunaan fungsi aktivasi *ReLU* terbukti dapat meningkatkan kinerja pelatihan pada jaringan saraf dalam (*deep neural network*). Fungsi ini bekerja dengan memetakan nilai input x secara langsung apabila $x > 0$, sedangkan untuk nilai $x \leq 0$, akan dikonversi menjadi nol. Adapun bentuk persamaan matematis dari fungsi *ReLU* adalah sebagai berikut:

$$\begin{aligned}
 f(x) &= x^+ \\
 &= \max(0, x) = \frac{x + |x|}{2} \\
 &= \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{otherwise.} \end{cases} \dots \dots \dots (1)
 \end{aligned}$$

Fungsi derivatif :

$$\begin{aligned}
 f'(x) &= \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases} \dots \dots \dots (2)
 \end{aligned}$$

Plot ReLU:



Gambar 2. 5 Plot ReLU

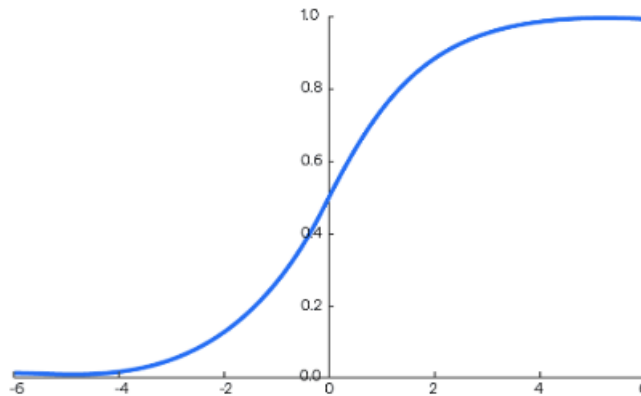
(Sumber : researchgate.net, 2020)

a. Softmax Activation

Softmax Activation atau aktivasi *softmax* adalah sebuah jenis aktivasi yang digunakan untuk mengklasifikasikan dua atau lebih kelas. Aktivasi *softmax* merubah hasil dari *output layer* terakhir menjadi probabilitas dengan rentang nilai 0 hingga 1 yang mana nanti nya akan dikategorikan menjadi kelas yang diinginkan.

Adapun berikut merupakan rumus dan plot aktivasi relu :

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$



Gambar 2. 6 Plot *Softmax*

(Sumber : [researchgate.net](https://www.researchgate.net), 2022)

b. Pooling Layer

Pooling layer berperan dalam mereduksi ukuran spasial dari hasil ekstraksi fitur oleh *convolution layer*. Proses ini dikenal sebagai *downsampling* dan bertujuan untuk menurunkan kompleksitas komputasi dengan cara mengurangi dimensi dari *feature map*, sehingga jumlah parameter yang perlu diperbarui menjadi lebih sedikit dan waktu komputasi pun lebih efisien. Selain itu, *pooling* membantu dalam mempertahankan fitur-fitur penting atau dominan, yang berguna untuk meningkatkan efektivitas proses pelatihan model. Terdapat dua jenis utama *pooling*, yaitu *max pooling* dan *average pooling*. *Max pooling* memilih nilai tertinggi dari area yang dicakup oleh *kernel*, sedangkan *average pooling* menghitung nilai rata-rata dari area tersebut.

c. Fully Connected Layer

Fully connected layer adalah lapisan yang digunakan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. Untuk mendapatkan hasil keluaran dari *layer* ini tidak dibutuhkan operasi konvolusi, tetapi menggunakan komputasi perkalian matriks yang diikuti dengan bias *offset*. Dengan penggunaan operasi tersebut, setiap *neuron*

memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya, sehingga layer ini disebut sebagai *fully connected layer* (Oluwafisayomi, 2022).

2.2.4 Augmentasi Data

Dalam pengolahan citra digital, khususnya pada tugas klasifikasi berbasis *deep learning* seperti *Convolutional Neural Network (CNN)*, kualitas dan kuantitas data sangat berpengaruh terhadap performa model. Namun, pengumpulan data dalam jumlah besar sering kali menjadi tantangan, baik dari segi waktu, tenaga, maupun sumber daya. Untuk mengatasi hal ini, digunakanlah teknik augmentasi data (*data augmentation*), yaitu metode manipulasi data latih guna menghasilkan variasi citra baru dari data yang sudah ada tanpa perlu melakukan labeling ulang.

Menurut penelitian oleh Yuda Prasetyo (2024) dalam jurnal *Overcoming Overfitting in CNN Models for Potato Disease Classification Using Data Augmentation*, augmentasi data terbukti mampu mengurangi *overfitting* dan meningkatkan kemampuan generalisasi model terhadap data baru. Hal ini penting karena model *deep learning* memiliki kecenderungan untuk "hapal" terhadap data pelatihan dan gagal memprediksi dengan baik pada data yang tidak pernah dilihat sebelumnya. Berikut adalah beberapa contoh teknik augmentasi data yang sering digunakan.

1. Rescale

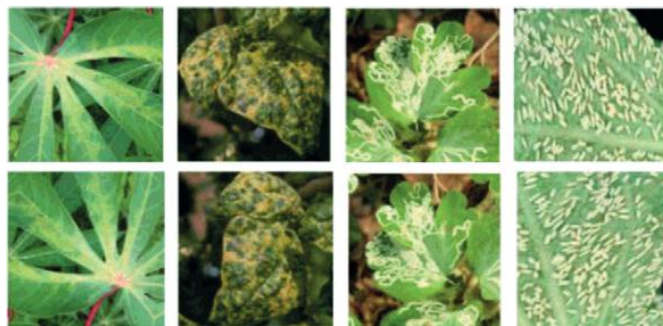
Rescale atau juga biasa disebut *min-max normalization*, proses ini membagi setiap piksel yang memiliki rentang nilai 0 hingga 255 dengan nilai maksimumnya menjadi 0 sampai 1, hal tersebut bertujuan supaya model dapat melakukan proses *training* lebih cepat.

2. Width Shift

Width shift adalah teknik augmentasi yang digunakan untuk memperkenalkan variasi posisi horizontal pada citra dengan mengubah citra secara horizontal. Teknik ini memungkinkan model CNN untuk belajar mengenali objek dari posisi yang berbeda secara horizontal, meningkatkan kemampuan generalisasi, dan mengurangi kemungkinan *overfitting*.

3. *Height Shift*

Proses ini mirip dengan proses *Width Shift* dimana dilakukan pergeseran citra secara vertikal. Dalam kasus ini citra akan dipindahkan ke atas atau ke bawah dengan jarak sebesar secara acak. Berikut adalah contoh gambar dari proses *width* dan *height shift*.



Gambar 2. 7 Sebelum dan sesudah dilakukan *shift*

(Sumber: Surinta., 2022)

4. *Shear*

Shear adalah teknik augmentasi yang berguna untuk mengubah orientasi objek dalam citra dengan cara distorsi geometris. Teknik ini membantu meningkatkan kemampuan model untuk mengenali objek meskipun dalam kondisi miring atau terdistorsi.



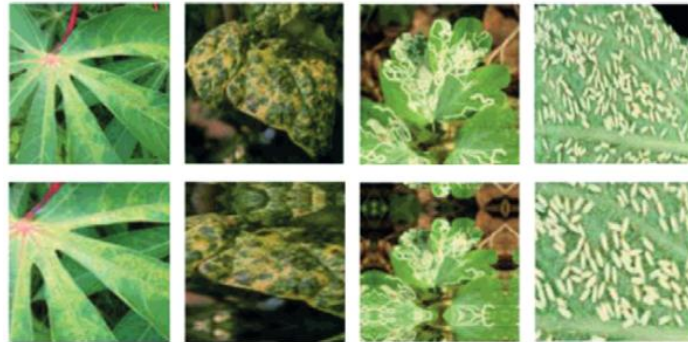
Gambar 2. 8 Proses *Shear*

(Sumber: researchgate.net., 2024)

5. *Zoom*

Zoom dalam *data augmentation* adalah teknik yang digunakan untuk mengubah skala objek dalam gambarr, biasanya dengan memperbesar (*zoom-in*) atau memperkecil (*zoom-out*) citra tersebut. Tujuan dari

penggunaan *zoom* adalah untuk membantu model menggeneralisasi dengan lebih baik terhadap objek yang tampak lebih besar atau lebih kecil dalam citra dan untuk menambah variasi pada saat *training dataset*.



Gambar 2. 9 Contoh proses *Zoom*

(Sumber: Surinta., 2022)

6. *Horizontal Flip*

Horizontal Flip dalam *data augmentation* adalah teknik yang digunakan untuk membalik citra secara horizontal, sehingga objek di dalam citra akan terlihat seperti dipantulkan dalam cermin. Teknik ini sangat berguna untuk membantu model belajar mengenali objek dari orientasi yang berbeda.



Gambar 2. 10 Contoh *Horizontal Flip*

(Sumber: [researchgate.net](https://www.researchgate.net)., 2024)

7. *Fill Mode*

Fill Mode adalah metode yang digunakan untuk mengisi ruang kosong pada citra setelah transformasi (seperti rotasi, *zoom*, atau pergeseran) dengan warna dari piksel. Ini membantu memastikan bahwa citra yang dihasilkan tetap penuh, tidak ada ruang kosong, dan terlihat alami.

2.2.5 *Python*

Python merupakan salah satu bahasa pemrograman yang sangat populer dan banyak digunakan saat ini. Bahasa ini pertama kali dikembangkan oleh *Guido van Rossum* pada tahun 1991 di *Stichting Mathematisch Centrum* (CWI), Amsterdam (Raschka et al., 2020). *Python* sendiri terinspirasi dari bahasa pemrograman *ABC* yang lebih dahulu muncul. Keunggulan *Python* dibandingkan dengan bahasa pemrograman lainnya terletak pada sifatnya yang *open source*, sehingga proses pengembangannya melibatkan kontribusi dari jutaan pengguna, mulai dari kalangan programmer, peneliti, hingga masyarakat umum dari berbagai disiplin ilmu, tidak terbatas pada bidang teknologi informasi.

Python merupakan bahasa pemrograman yang dijalankan menggunakan *interpreter*, yang memungkinkan kode dieksekusi secara langsung tanpa perlu dikompilasi terlebih dahulu. Hal ini memberikan kemudahan bagi pengguna dalam menulis dan menjalankan program. Selain itu, *Python* bersifat lintas platform, sehingga dapat dijalankan di berbagai sistem operasi seperti *Windows*, *Linux*, dan lainnya. Bahasa ini menggabungkan berbagai paradigma pemrograman dari bahasa lain, seperti pemrograman prosedural dari *C*, paradigma *object-oriented* dari *Java*, serta pendekatan fungsional dari *Lisp*.

2.2.6 *Tensorflow*

TensorFlow merupakan sebuah platform *open source* yang dirancang secara menyeluruh untuk mendukung pengembangan *machine learning*. Platform ini dikembangkan oleh tim *Google Brain* dan menyediakan berbagai alat, pustaka, serta dukungan komunitas yang luas untuk memfasilitasi proses penelitian dan implementasi dalam bidang pembelajaran mesin. Dengan arsitektur yang fleksibel, *TensorFlow* memungkinkan proses komputasi dijalankan secara efisien pada berbagai perangkat keras, seperti *CPU*, *GPU*, maupun *TPU*, serta kompatibel dengan berbagai lingkungan, mulai dari komputer desktop, layanan *cloud*, hingga perangkat mobile.

2.2.7 Keras

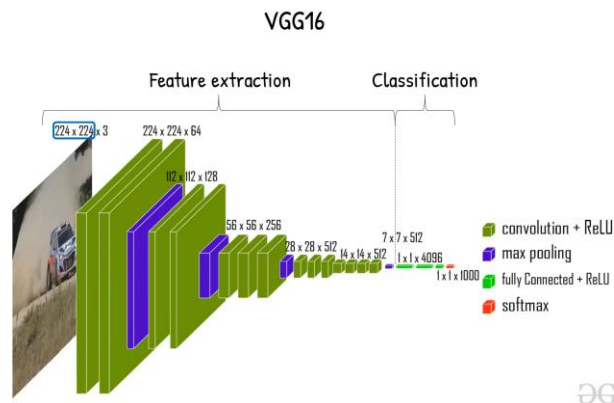
Keras merupakan sebuah *Application Programming Interface (API)* yang memudahkan pengembangan jaringan saraf tiruan (*artificial neural network*). Melalui *Keras*, pengguna dapat dengan mudah membangun berbagai arsitektur jaringan seperti *multi-layer perceptron* dan *convolutional neural network*. Implementasi *Keras* cukup luas, antara lain dalam tugas klasifikasi citra, pemrosesan bahasa alami (*natural language processing*), pengenalan suara, serta prediksi deret waktu (*time series prediction*).

2.2.8 Matplotlib

Matplotlib adalah salah satu visualisasi data yang paling populer digunakan dalam *library python*. *Library* ini dibuat oleh John hunter bersama rekan-rekannya yang telah meluangkan sangat banyak waktu agar *library* ini dapat digunakan oleh peneliti dari seluruh dunia. *Matplotlib* merupakan sebuah *library* grafis untuk memvisualisasikan data dengan python yang mencakup aspek integral dan juga *matplotlib* ini mudah untuk digunakan bersamaan dengan *Numpy*, *Pandas* dan *library python* yang lainnya (Ali Hassan Sial, 2021).

2.2.9 VGG16

VGG16, yang dikembangkan oleh Simonyan dan Zisserman, dikenal dengan arsitekturnya yang menggunakan lapisan konvolusi yang dalam untuk ekstraksi fitur (Simonyan, 2022). *VGG16* merupakan salah satu arsitektur *deep convolutional neural network* yang terdiri dari 16 lapisan dan termasuk ke dalam model awal yang banyak digunakan dalam teknik *transfer learning*. Struktur arsitektur *VGG16* terdiri atas 13 lapisan *convolutional*, dua lapisan *fully connected*, dan satu lapisan akhir sebagai *classifier*. Ilustrasi dari arsitektur *VGG16* dapat dilihat pada gambar berikut.



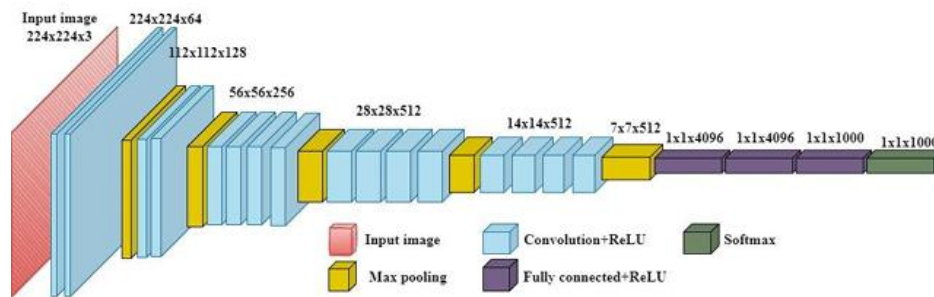
Gambar 2. 11 Model Arsitektur *VGG16*

(Sumber : geeksforgeeks.org, 2024)

Setiap rangkaian *convolutional layer* pada arsitektur di atas menggunakan ukuran *kernel* sebesar 3×3 . Perbedaan utama dari tiap susunan konvolusi terletak pada jumlah *filter* yang digunakan. Sebanyak 64 *filter* digunakan pada dua rangkaian awal, kemudian meningkat menjadi 128 pada susunan ketiga dan keempat. Selanjutnya, pada susunan kelima hingga ketujuh digunakan 256 *filter*, dan 512 *filter* digunakan mulai dari susunan kedelapan hingga ketigabelas. Setelah proses konvolusi tertentu, dilakukan operasi *max pooling* berukuran 2×2 pada susunan ke-2, ke-4, ke-7, ke-10, dan ke-13. Hasil akhir dari *pooling layer* tersebut akan diteruskan ke *fully connected layer*, kemudian menuju *classifier* yang berfungsi untuk menentukan kelas atau label dari data yang diklasifikasikan (Arief Saputro, 2022).

2.2.10 *VGG19*

Arsitektur *VGG19* merupakan salah satu model *deep learning* yang meliputi 19 *layer*. Model arsitektur *VGG19* dapat dilihat pada gambar dibawah.



Gambar 2. 12 Model Arsitektur VGG19

(Sumber : Nguyen et al., 2022)

Dalam setiap lapisan konvolusi memiliki kernel berdimensi 3x3. Perbedaan yang terlihat antara *VGG16* dan *VGG19* terletak pada jumlah lapisan konvolusi. Jumlah filter 64 ada pada 2 susunan konvolusi awal, sedangkan jumlah filter 128 susunan 3 serta 4. Begitu pada susunan konvolusi lain yang jumlah filternya berbeda ialah 256 (susunan 5 6 7 8) serta 512 (susunan 9 10 11 12 13 14 15 16). 2x2 max pooling digunakan sesudah susunan konvolusi 2 4 8 12 serta 16.

2.2.11 Xception

Arsitektur *Xception* dikembangkan dengan mengadaptasi konsep dari *Inception*, namun melakukan modifikasi pada modulnya dengan mengganti komponen *Inception* menggunakan *depthwise separable convolution*. Istilah ini, yang juga dikenal sebagai *separable convolution* dalam berbagai *framework deep learning* seperti *TensorFlow* dan *Keras*, terdiri atas dua tahap utama: pertama, *depthwise convolution*, yaitu proses konvolusi spasial yang dilakukan secara terpisah pada setiap saluran (*channel*) input; dan kedua, *pointwise convolution*, yaitu konvolusi berukuran 1x1 yang berfungsi untuk memproyeksikan hasil dari *depthwise convolution* ke ruang saluran baru. Secara umum, struktur dari *Xception* dapat digambarkan sebagai susunan linear dari beberapa *depthwise separable convolution layer* yang dilengkapi dengan koneksi *residual* (Akram, 2023).

Secara umum, terdapat dua perbedaan utama antara arsitektur *Inception* dan *Xception* yang menggunakan *depthwise separable convolution*. Pertama, perbedaan terletak pada urutan operasinya. Pada *depthwise separable convolution*, proses diawali dengan konvolusi spasial per saluran (*channel-wise convolution*), kemudian

terdapat 4 (empat) istilah pengelompokan. Nilai *True Negative* (TN) adalah jumlah informasi negatif yang dapat dibedakan secara akurat, sedangkan *False Positive* (FP) adalah informasi negatif tetapi diidentifikasi sebagai informasi yang pasti. Sementara itu, *True Positive* (TP) adalah informasi positif yang dibedakan secara akurat. *False Negative* (FN) adalah sesuatu yang bertentangan dengan *True Positive*, sehingga informasi yang positif, namun diakui sebagai informasi yang negatif. Seperti pada penelitian yang dilakukan Tantri (2024) jenis klasifikasi binary yang hanya memiliki 2 keluaran kelas, *confusion matrix* dapat disajikan seperti pada berikut ini :

Tabel 2. 2 *Confusion Matrix*

		Actual Values	Actual Values
		Positive	Negative
Predicted Values	Positive	True Positive	False Positive
Predicted Values	Negative	True Negative	False Negative

Dengan mempertimbangkan nilai *True Negative*, *False Positive*, *False Negative*, dan *True Positive*, maka metrik evaluasi seperti akurasi, presisi, dan *recall* dapat dihitung. Presisi menunjukkan tingkat ketepatan model dalam mengklasifikasikan data secara benar, khususnya dalam mengidentifikasi data positif yang relevan dibandingkan dengan seluruh prediksi positif yang dibuat oleh model.

a. *Presicion*

Precision score adalah metrik evaluasi yang digunakan dalam klasifikasi untuk mengukur akurasi dari prediksi positif yang dibuat oleh model. *Precision* menghitung proporsi dari prediksi positif yang benar-benar positif. Dalam konteks klasifikasi biner, *precision* dapat dihitung dengan menggunakan rumus:

$$Precision = \left(\frac{TP}{TP+FP} \right) \times 100\%$$

b. *Recall*

Recall merupakan ukuran sejauh mana sistem mampu menemukan seluruh data relevan yang ada di dalam kumpulan data berdasarkan *query* yang diberikan. Dalam konteks klasifikasi, *recall* juga dikenal dengan istilah *sensitivity*, yang merepresentasikan kemampuan model dalam mendeteksi seluruh data yang benar-benar termasuk dalam kelas positif. Nilai *recall* menunjukkan seberapa besar peluang sistem dalam mengidentifikasi data relevan secara tepat. Adapun rumus *recall* dapat dituliskan sebagai berikut:

$$Recall = \left(\frac{TP}{TP+FN} \right) \times 100\%$$

c. *Accuracy*

Accuracy adalah hasil prediksi benar dari semua *dataset*. akurasi juga dapat di persentasekan dari total e-mail yang benar dengan mengidentifikasinya kedalam rasio prediksi positif atau negatif. Rumus *Accuracy* adalah :

$$Accuracy = \left(\frac{TP+TN}{TP+FP+FN+FP} \right) \times 100\%$$

d. *F-1 Score*

F-1 score adalah metrik evaluasi yang digunakan dalam klasifikasi untuk mengukur keseimbangan antara *precision* dan *recall*. Ini adalah rata-rata nilai dari *precision* dan *recall*, memberikan nilai tunggal yang mempertimbangkan keduanya. *F-1 score* sangat berguna ketika Anda ingin menyeimbangkan *trade-off* antara *precision* dan *recall*, terutama dalam kasus di mana *dataset* tidak seimbang atau ketika *false positives* dan *false negatives* sama-sama penting. Berikut rumus *F-1 score*:

$$F-1 = \left(\frac{2 \times Precision \times Recall}{Precision + Recall} \right) \times 100\%$$

Keterangan:

TP : *True Positive*

TN : *True Negative*

FP : *False Positive*

FN : *False Negative*

BAB III

METODE PENELITIAN

3.1 Jenis Penelitian

Penelitian ini menggunakan pendekatan kualitatif dan termasuk ke dalam jenis penelitian pengembangan atau *Research and Development (R&D)*. Pendekatan kualitatif digunakan karena sesuai untuk mengkaji permasalahan yang membutuhkan analisis terhadap data deskriptif non-numerik, sehingga mampu memberikan pemahaman mendalam mengenai objek yang diteliti (Taherdoost, 2023). Adapun pendekatan *R&D* dipilih karena penelitian ini bertujuan untuk menghasilkan atau menyempurnakan suatu produk. Produk yang dikembangkan tidak terbatas pada bentuk fisik atau perangkat keras, melainkan juga dapat berupa perangkat lunak seperti aplikasi atau program komputer.

Penelitian ini bertujuan untuk membandingkan dan mengevaluasi kinerja tiga model arsitektur *CNN*, yaitu *VGG16*, *VGG19* dan *Xception*, dalam konteks klasifikasi penyakit pada daun tanaman tomat. Aspek yang perlu dipertimbangkan meliputi metrik akurasi, presisi dan *recall* dari masing-masing model.

3.2 Analisa Sistem

Sebelum merancang sebuah sistem ada baiknya mempersiapkan komponen-komponen atau instrumen yang diperlukan dalam proses pembuatan sebuah sistem. Beberapa hal yang dibutuhkan yaitu terkait kebutuhan perangkat keras (*hardware*) dan kebutuhan perangkat lunak (*software*). Adapun alat dan bahan yang digunakan adalah sebagai berikut:

A. Perangkat Keras

Perangkat keras yang digunakan berupa sebuah laptop HP OMEN 15 dengan spesifikasi sebagai berikut:

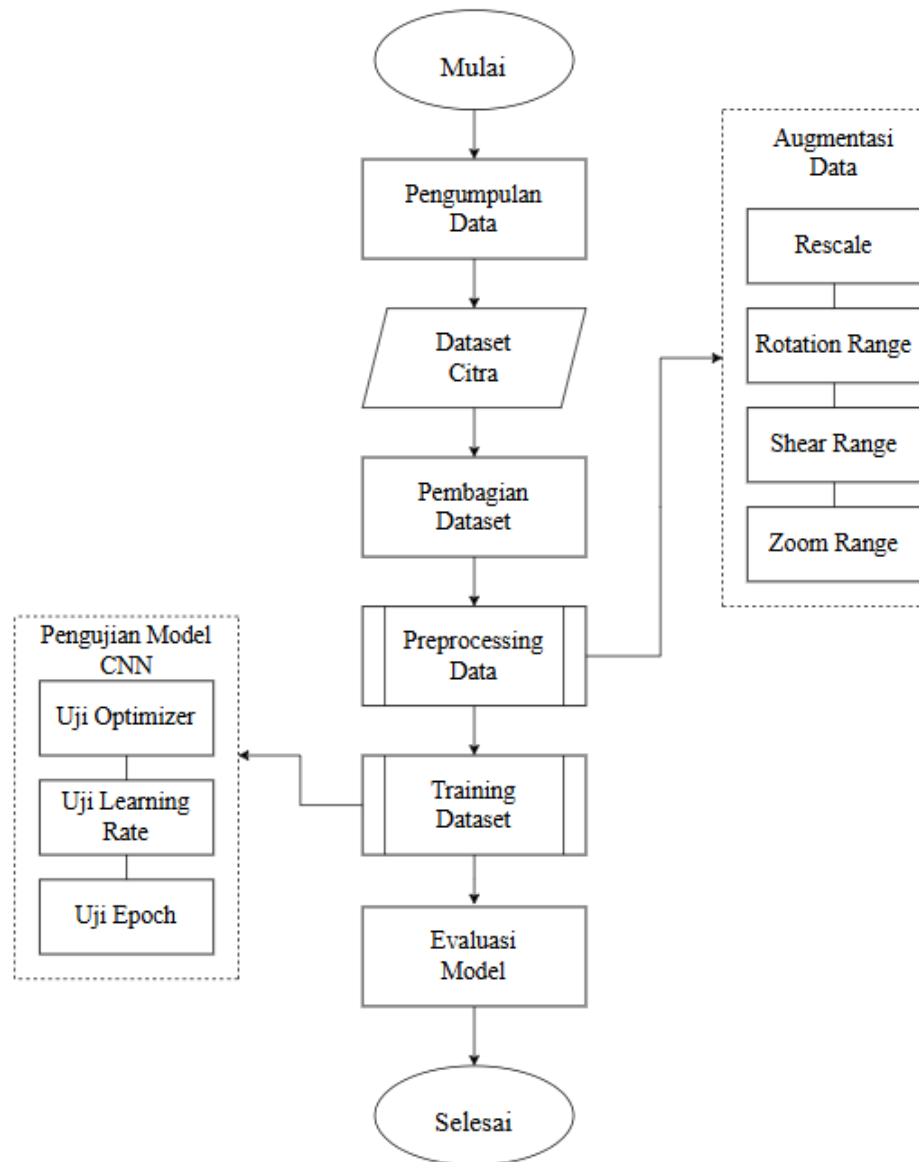
1. CPU : Ryzen 7 5800H
2. GPU : RTX 3060 Mobile
3. RAM : 16GB DDR4
4. SSD : 1 TB SSD NVme.

B. Perangkat Lunak

1. Sistem Operasi Windows 11 Home
2. *Pyhton* 3.10 sebagai Bahasa pemrograman
3. *Jupyter Notebook* sebagai teks editor
4. *Visual Studio Code* sebagai teks editor
5. *Flask* sebagai alat *deployment*
6. *Tensorflow* dan *Keras* sebagai *library Machine Learning*

3.3 Tahapan Penelitian

Berikut merupakan tahapan-tahapan pada penelitian yang saya lakukan:



Gambar 3. 1 Alur Penelitian

3.3.1 Pengumpulan Data

Proses pengumpulan data pada penelitian ini dilakukan menggunakan data sekunder yang diperoleh dari *website Kaggle* dengan nama *Tomato Leaf Disease* yang nantinya akan digunakan sebagai *dataset* dalam proses pelatihan model.

Dalam proses ini diambil sampel data dan nantinya akan diklasifikasikan menjadi empat kelas .

Data yang digunakan berupa citra daun tomat yang sehat dan yang terinfeksi penyakit dengan total citra sebanyak 4000 dan dibagi menjadi, 1000 citra daun tomat yang sehat (*Healthy*) , 1000 citra daun tomat busuk daun(*Late Blight*), 1000 citra daun tomat bercak bakteri(*Bacterial Spot*) dan 1000 citra daun tomat yang terkena virus kuning keriting(*Yellow Leaf Curl Virus*). Citra yang diambil dari dataset ini berukuran 512x512 pixel sehingga perlu dilakukan *preprocessing* pada citra untuk mendapatkan hasil yang maksimal.

Dalam proses pengujian nantinya akan digunakan data citra daun tomat yang telah dikumpulkan dari *website Kaggle*. Dan untuk data uji sekunder digunakan citra yang dikumpulkan pada saat melakukan observasi di lahan pertanian Fakultas Pertanian Universitas Islam Riau sebagai referensi citra yang berada diluar *training dataset*.

3.3.2 Pembagian Dataset

Sebelum masuk ke tahap *Preprocessing data* dilakukan pembagian data menjadi data *training*, data *testing*, dan data *validation*. Pada penelitian ini data dibagi dengan perbandingan 80% : 10% : 10% dengan keterangan seperti tabel berikut:

Tabel 3. 1 Pembagian Data

No	Nama	Jenis	Jumlah
1	Citra Tomat Healthy	Training	800
2	Citra Tomat Bacterial Spot	Training	800
3	Citra Tomat Late Blight	Training	800
4	Citra Tomat Yellow Leaf Curl Virus	Training	800
5	Citra Tomat Healthy	Testing	100
6	Citra Tomat Bacterial Spot	Testing	100
7	Citra Tomat Late Blight	Testing	100
8	Citra Tomat Yellow Leaf Curl Virus	Testing	100

9	Citra Tomat Healthy	Validation	100
10	Citra Tomat Bacterial Spot	Validation	100
11	Citra Tomat Late Blight	Validation	100
12	Citra Tomat Yellow Leaf Curl Virus	Validation	100

3.3.3 Preprocessing Data

Setelah dilakukan pembagian data selanjutnya masuk ketahap *preprocessing* dengan cara melakukan augmentasi data, tahapan ini bertujuan untuk mencegah terjadinya *overfitting* (data mendapatkan hasil yang buruk jika diuji dengan data diluar *data training*). Dalam penelitian ini, proses augmentasi dilakukan menggunakan fungsi “*ImagDataGenerator*” dari *library keras*. Keunggulan menggunakan *ImageDataGenerator* ini yaitu memungkinkan proses augmentasi citra secara *real-time* selama proses *training* sedang berlangsung. Artinya, citra yang telah dilakukan augmentasi tidak perlu disimpan secara permanen di penyimpanan lokal.

Adapun beberapa Teknik augmentasi data yang digunakan adalah sebagai berikut:

Tabel 3. 2 Teknik Augmentasi Data

No	Nama Parameter	Nilai	Keterangan
1	Rescale	1/255	Berfungsi untuk mengubah ukuran data piksel RGB gambar (0-255) menjadi rentang angka (0-1) untuk memudahkan proses training data
2	Rotation	15°	Berfungsi untuk mengubah rotasi gambar kekiri dan kenanan sebesar 15°
3	Width Shift	10	Berfungsi untuk mengatur posisi gambar pada lebar gambar sebesar 10%

4	Height Shift	10	Berfungsi untuk mengatur posisi gambar pada tinggi gambar sebesar 10%
5	Shear	20	Berfungsi meregangkan citra kekanan dan ke kiri sebesar 20%
6	Zoom	20	Berfungsi untuk memperbesar ukuran citra sebesar 20%
7	Horizontal Flip	1(True)	Berfungsi untuk memutar citra secara horizontal
8	Fill_Mode	Nearest	Berfungsi mengisi kosongan piksel setelah dilakukan transformasi citra

Pada tahap ini juga dilakukan proses *resize* untuk memperkecil ukuran citra yang awalnya 256x256 piksel menjadi 224x224 piksel untuk menyesuaikan ukuran input citra dengan model yang digunakan. Berikut adalah hasil dari proses augmentasi pada citra daun tanaman tomat.



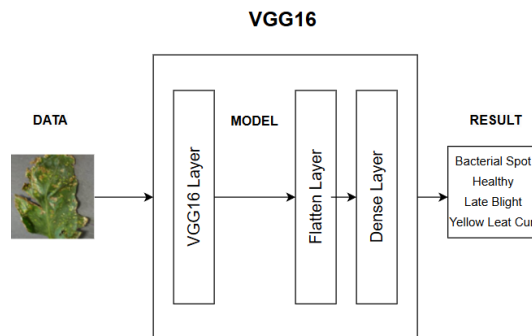
Gambar 3. 2 Hasil Augmentasi Citra

3.3.4 Training Dataset

Proses ini digunakan untuk menemukan pola tertentu yang ada pada data. Tahapan ini dilakukan setelah tahapan *preprocessing* pada citra dan akan dibangun menggunakan beberapa model arsitektur yaitu *VGG16*, *VGG19* (Husodo, 2023) dan *Xception* (Chollet, 2022). Secara keseluruhan proses pada model ini terbagi menjadi 2 kategori yaitu proses *feature learning* dan *classification feature learning*.

adalah proses terjadi nya konvolusi dalam pengenalan data dan fitur yang ada didalam citra sedangkan *classification* merupakan bagian dimana hasil presdiksi data keluar. Perbedaan setiap model terletak pada bagian *feature learning*.

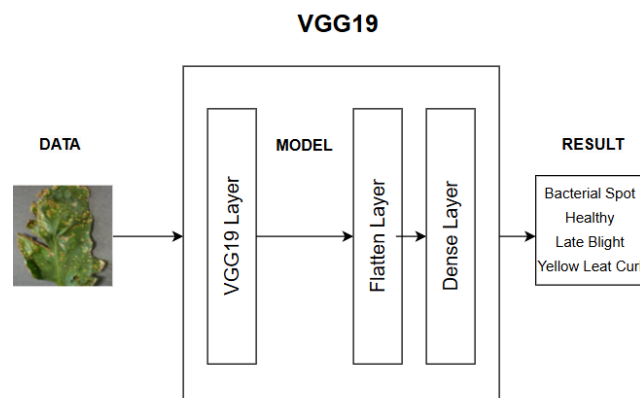
1. *VGG16*



Gambar 3. 3 Arsitektur *VGG16*

Arsitektur ini melakukan proses feature learning menggunakan *VGG16* layer. Sebuah layer yang berisi model *VGG16*. Layer yang terdapat di dalam model *VGG16* dapat dilihat pada gambar 2.7. Dalam proses menggunakan *VGG16* data akan menggunakan proses *feature learning* dari model *VGG16* dan nanti nya akan dilakukan proses *classification* yang menghasilkan empat hasil keluaran.

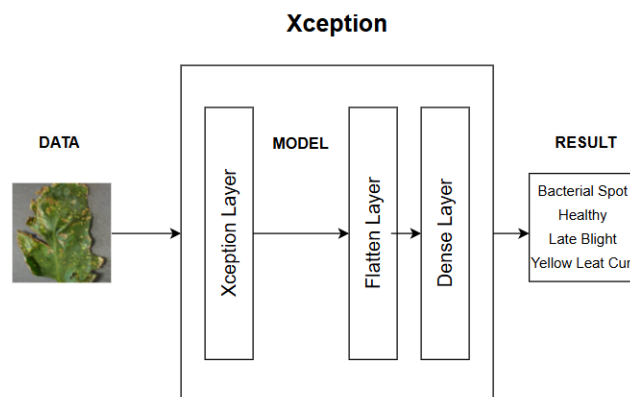
2. *VGG19*



Gambar 3. 4 Arsitektur *VGG19*

Arsitektur ini memiliki cara kerja yang mirip dengan arsitektur *VGG16* namun memiliki jumlah *layer* yg berbeda yaitu berjumlah 16 *layer* konvolusi dan 3 *layer fully connected*. Adapun jumlah keluaran yang dihasilkan dengan menggunakan model ini akan sama dengan jumlah *dense layer* terakhir yang digunakan.

3. *Xception*



Gambar 3. 5 Arsitektur *Xception*

Arsitektur yang digunakan dalam penelitian ini memiliki kemiripan dengan arsitektur *InceptionResNetV2*, namun proses *feature learning* dilakukan melalui penggunaan *Xception layer*. *Xception layer* merupakan sebuah lapisan yang memuat keseluruhan model *Xception*. Susunan lapisan yang ada di dalam model *Xception* ditampilkan pada Gambar 2.9. Mekanisme kerja model *Xception* serupa dengan *InceptionResNetV2*, di mana proses *feature learning* dilakukan melalui lapisan konvolusi yang terpisah (*separable convolution*). Hasil dari proses ini kemudian dimanfaatkan dalam tahap klasifikasi untuk menentukan label *output*.

A. *Input Image*

Tahapan pertama dalam melakukan pelatihan terhadap model adalah memasukkan data citra daun ke *input layer*, pada *layer* ini citra dikonversi kedalam matriks tiga dimensi, dengan ukuran panjang x lebar x 3 *channels* RGB (*Red, Green, Blue*). Pada penelitian ini nilai RGB pada setiap piksel dinormalisasi menjadi 0-1 untuk mempermudah proses komputasi.

B. Convolution

Proses konvolusi berfungsi untuk mengekstraksi fitur-fitur (*feature map*) yang ada pada citra dengan menggunakan filter. Penelitian ini menggunakan *filter size* berukuran 3x3 piksel, dengan stride 1. Pada *CNN* sendiri *filter size* juga disebut sebagai *kernel* atau *weight* yang nilainya diinisialisasi secara acak. Contoh prosesnya dapat dilihat pada Gambar 3.6 berikut.

4	7	1	9	8	5
3	8	5	1	3	9
2	2	8	2	5	7
3	3	4	6	7	9
2	5	6	1	4	8
7	3	6	3	8	5

6x6

*

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

3x3

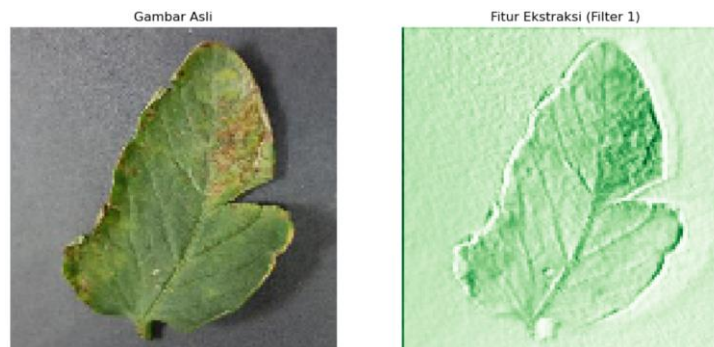
=

15	2	0	0
0	19	0	1
0	0	10	4
7	4	0	0

4x4

Gambar 3. 6 Proses Ekstraksi fitur

Pada ilustrasi gambar 3.6 terdapat *input* citra berukuran 6x6 yang akan di konvolusi menggunakan *filter size* berukuran 3x3 yang akan menghasilkan *feature map* berukuran 4x4. Operasi konvolusi dilakukan dengan menggeser kernel konvolusi piksel per piksel. Hasil konvolusi disimpan di dalam matriks yang baru. Proses perhitungannya adalah sebagai berikut $(0 \times 4) + (-1 \times 7) + (0 \times 1) + (-1 \times 3) + (4 \times 8) + (-1 \times 5) + (0 \times 2) + (-1 \times 2) + (0 \times 8) = 15$. Dari proses tersebut menghasilkan nilai 15 yang akan menempati 1 buah sel di citra yang baru. Setelah itu filter digeser lagi kekanan dan kebawah hingga terbentuk 1 citra baru.



Gambar 3. 7 Hasil Ektrak fitur

C. *Activation Function*

Untuk memperoleh objek utama dalam sebuah citra, diperlukan proses pemisahan antara objek dan latar belakang. Dalam penelitian ini, digunakan *activation function* ReLU (*Rectified Linear Unit*) untuk menentukan aktivitas suatu neuron dalam *neural network*. Dengan menerapkan fungsi aktivasi tersebut, hanya neuron-neuron yang memberikan kontribusi terhadap representasi objek dalam citra yang akan diaktifkan, sehingga proses ekstraksi fitur menjadi lebih efektif. Adapun cara kerja *activation* ReLU telah dijelaskan pada gambar 2.5, dimana *activation* ini berfungsi untuk merubah nilai *negative* dari hasil ekstraksi menjadi nilai 0.

D. *Pooling Layer*

Pooling layer digunakan dalam penelitian ini untuk mereduksi ukuran spasial citra sekaligus mengurangi jumlah parameter dan beban komputasi pada *neural network*. Lapisan ini menerima masukan dari hasil *feature map* yang diperoleh melalui proses konvolusi pada *convolution layer*. Penelitian ini menerapkan metode *max pooling* dengan ukuran kernel 2x2, sehingga menghasilkan citra yang lebih kecil secara dimensi. Sebagai ilustrasi, Gambar 3.8 menunjukkan proses kerja *max pooling*, di mana nilai output yang diambil adalah nilai tertinggi dari setiap jendela 2x2. Dalam contoh tersebut, nilai 19 merupakan nilai maksimum sehingga dipilih sebagai hasil keluaran.

15	2	0	0
0	19	0	1
0	0	10	4
7	4	0	0

=

19	1
7	10

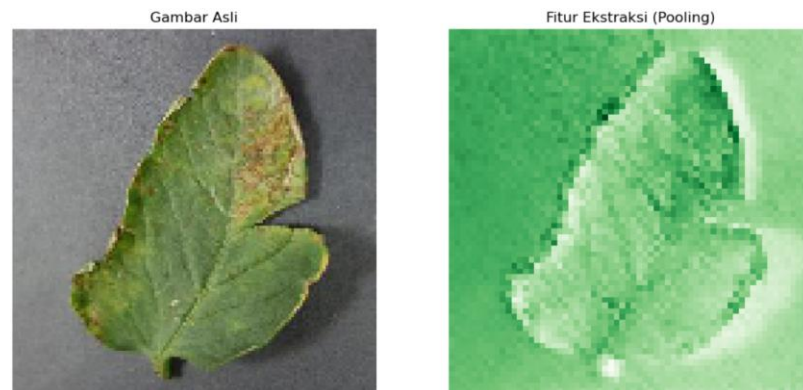
Gambar 3. 8 Proses *Pooling*

Berdasarkan Gambar 3.8 dengan elemen berwarna hijau nilai terbesarnya adalah 19 maka nilai yang digunakan untuk input citra baru adalah 19 begitupun untuk elemen berwarna kuning dengan nilai terbesar adalah 1 maka *input* citra barunya adalah 1, begitupun seterusnya. .Proses komputasi yang terjadi pada

pooling layer dijelaskan seperti pada Tabel 3.3, dan hasil dari *proses pooling* dapat dilihat pada Gambar 3.9 dimana jumlah piksel yang ada menjadi semakin sedikit dan hanya menampilkan nilai terbesarnya saja

Tabel 3. 3 Proses *Pooling*

No	Sel	Nilai Pada Area Matriks	Hasil
1	$f(1,1)$	15,2,0,19	19
2	$f(1,2)$	0,0,0,1	1
3	$f(1,3)$	0,0,7,4	7
4	$f(1,4)$	10,4,0,0	10



Gambar 3. 9 Hasil *Pooling*

E. *Fully Connected Layer*

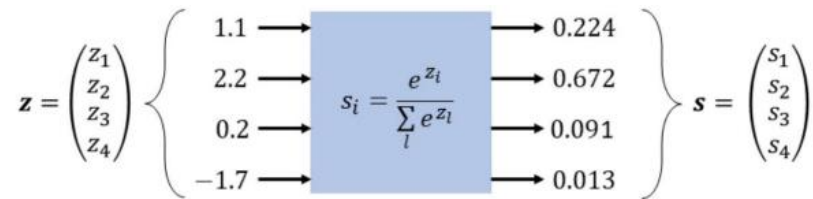
Fully Connected Layer merupakan tahap terakhir sebelum dilakukan klasifikasi pada *output data*, pada tahap ini dilakukan proses untuk mengubah nilai dari tahap konvolusi sebelumnya dari data matriks 3 dimensi menjadi data matriks 1 dimensi (*Flatten*). Proses komputasi *flatten* dapat dilihat pada tabel berikut :

Tabel 3. 4 Proses *Flattening*

Data Matriks				Data Vektor (x)								
6	4	9		6	4	9	7	3	5	1	2	7
7	3	5										
1	2	7										

F. Output Layer

Output layer dalam penelitian ini berfungsi untuk menghasilkan hasil akhir dari proses klasifikasi berupa prediksi kelas. Lapisan ini bekerja dengan memproyeksikan hasil pembelajaran fitur ke dalam bentuk nilai probabilitas untuk setiap kelas. Kelas dengan nilai probabilitas tertinggi akan dipilih sebagai hasil prediksi. Penelitian ini menggunakan fungsi aktivasi *Softmax* pada lapisan ini, yang disesuaikan untuk mengklasifikasikan data ke dalam empat kelas. Proses kerja dari lapisan ini dapat dilihat pada Gambar 3.10.

**Gambar 3. 10** Cara Kerja *Softmax Activation*

Keterangan :

Z = Layer Output

S = Probabilitas

3.3.5 Rencana Pengujian Hyperparameter

Pada tahap ini dilakukan beberapa *scenario* yang nantinya digunakan untuk perbandingan model sehingga didapatkan model dengan hasil terbaik . Berikut adalah beberapa *parameter* dan *scenario* yang akan dilakukan pengujian diantaranya , *Epoch*, *Optimizer* , Dan *Learning Rate*.

Tabel 3. 5 Parameter Pada Pengujian *hyperparameter*

No	Hyper Parameter	Nilai		
1	Optimizer	RMSprop	Adam	
2	Epoch	10	15	20
3	Learning Rate	0.1	0.01	0.001

3.3.6 Evaluasi Model

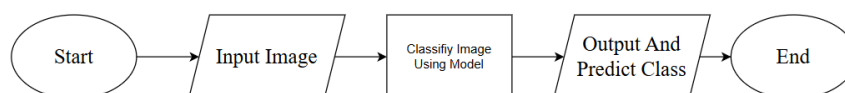
Setelah dilakukan berbagai skenario dengan menggunakan beberapa *hyperparameter* yang telah disediakan Langkah selanjutnya adalah dengan mengevaluasi model dengan melihat metrik nilai akurasi , presisi dan *recall* sehingga dapat menentukan mana model terbaik dalam hal klasifikasi penyakit pada daun tanaman tomat.

3.4 Rancangan Sistem

Pada tahapan ini dilakukan perancangan sistem dengan *flowchart* dan perancangan antarmuka sistem berupa web sederhana menggunakan *framework flask*. Dengan antarmuka sederhana ini diharapkan dapat mempermudah dalam melakukan klasifikasi terhadap penyakit pada daun tanaman tomat .

3.4.1 Flowchart Sistem

Berikut merupakan alur sistem yang akan digunakan dalam mengklasifikasi citra yang akan digunakan.

**Gambar 3. 11** Flowchart Sistem

Pada tahap awal setelah menjalankan program pengguna dapat memasukkan citra daun tanaman tomat kedalam sistem menggunakan antarmuka yang sudah disediakan. Setelah citra dimasukkan program akan langsung melakukan

klasifikasi menggunakan model *CNN* yang telah dilatih sebelumnya, dimana pada tahap ini model akan menganalisa citra dan menentukan class dari citra yang telah di *input*-kan.

Hasil dari klasifikasi tersebut kemudian akan ditampilkan dimana sistem akan memberikan label prediksi class beserta nilai kepercayaan (*confidence score*) dari prediksi tersebut. Nilai ini akan menunjukkan seberapa yakin model terhadap hasil klasifikasi yang telah diperoleh.

3.4.2 Rancangan Antarmuka Sistem

Berdasarkan hasil analisis kebutuhan sistem, diartikan rancangan antarmuka aplikasi yang ditampilkan pada gambar 3.12 berikut :

LOGO		
Image Classification	<p>Input Image</p> <p><input type="button" value="Browse File"/></p> <p><input type="button" value="Submit"/></p>	<p>Classification Result</p> <div data-bbox="1050 1043 1236 1227" style="border: 1px solid black; width: 100px; height: 80px; margin: 10px auto; background-color: #e6f2ff;"> <p style="text-align: center;">IMAGE</p> </div> <p>Result <input type="text"/></p>

Gambar 3. 12 *Interface Sistem*

Pada rancangan antarmuka ini dapat dilihat ada tombol “*Browse File*” yang digunakan untuk mengunggah citra daun tanaman tomat dari perangkat local. Citra yang diunggah akan dilakukan *resize* otomatis sebelum dilakukan klasifikasi. Setelah citra dari perangkat *local* diunggah pengguna dapat menekan tombol “*Submit*” untuk memulai proses klasifikasi menggunakan model yang sudah dilatih sebelumnya.

Pada sebelah kanan dari rancangan antarmuka tersebut nantinya akan menampilkan hasil klasifikasi berupa citra yang telah diunggah sebelumnya untuk mengetahui citra mana yang tadinya diunggah. Dibawah citra tersebut nantinya juga akan menampilkan hasil prediksi sesuai dengan label kelas penyakit yang

terdeteksi pada daun tersebut. Selain itu sistem juga akan menampilkan *confidence score* dari masing-masing model dalam bentuk persentase yang akan mengindikasikan tingkat keyakinan model terhadap hasil prediksi yang diberikan.

BAB IV

HASIL DAN PEMBAHASAN

Pada bab ini, penulis akan menguraikan tentang tahapan-tahapan dalam memperoleh hasil akhir penelitian berdasarkan metode yang telah dijabarkan pada bab sebelumnya.

4.1 Hasil pengolahan data penelitian

Penelitian ini diawali dengan melakukan proses pengumpulan data yang nantinya akan digunakan sebagai sebuah *dataset*. Data yang digunakan dalam penelitian ini adalah data citra dari daun tanaman tomat. Sumber dari *Dataset* yang digunakan dalam penelitian ini berasal dari *website Kaggle* tepatnya didapatkan dari akun Kaustubh B. Proses pengumpulan data dilakukan dengan memilah *dataset* daun tanaman tomat di *website Kaggle* lalu mengunduhnya ke perangkat lokal.

Untuk mempermudah proses pengolahan data menggunakan model *Convolutional Neural Network (CNN)*, dataset yang telah diunduh perlu diubah menjadi format yang lebih sistematis. Oleh karena itu, dilakukan proses iterasi terhadap seluruh *folder* dan *file* gambar di dalam direktori *dataset* untuk menghasilkan sebuah *dataframe* yang memuat dua informasi penting: jalur *file* citra (*filepath*) dan label kelas (*label*).

```
sdir="C:/Users/ianba/Documents/4 class/train/"

filepath=[]
labels=[]
classlist=os.listdir(sdir)
for class in classlist:
    classpath=os.path.join(sdir,class)
    if os.path.isdir(classpath):
        flist=os.listdir(classpath)
        for f in flist:
            fpath=os.path.join(classpath,f)
            filepath.append(fpath)
            labels.append(class)
Fseries= pd.Series(filepath, name='filepath')
Lseries=pd.Series(labels, name='labels')
df=pd.concat([Fseries, Lseries], axis=1)
print (df.head())
print (df['labels'].value_counts())
```

	filepath	labels
0	C:/Users/ianba/Documents/4 class/train/bacteri...	bacterial_spot
1	C:/Users/ianba/Documents/4 class/train/bacteri...	bacterial_spot
2	C:/Users/ianba/Documents/4 class/train/bacteri...	bacterial_spot
3	C:/Users/ianba/Documents/4 class/train/bacteri...	bacterial_spot
4	C:/Users/ianba/Documents/4 class/train/bacteri...	bacterial_spot

labels	
bacterial_spot	1000
healthy	1000
late_blight	1000
yellow_leaf_curl_virus	1000

Name: count, dtype: int64

Gambar 4. 1 Pembuatan Label

Proses ini bertujuan untuk mempermudah tahap *preprocessing dan training data* nantinya, khususnya pada saat menggunakan fungsi-fungsi seperti *flow_from_dataframe()* dari Pustaka *keras*, yang membutuhkan data dalam bentuk struktur tabel agar dapat mengakses dan melabeli *dataset* citra dengan benar (Liao et al., 2021). Dari gambar diatas dapat dilihat bahwa pada penelitian ini saya memiliki empat folder dengan masing-masing folder memiliki 1000 citra daun tanaman tomat dari setiap *class*.

4.2 Pembagian Dataset dan *Preprocessing*

Sebelum dilakukan *Preprocessing* pada data disini dilakukan pembagian *dataset* terlebih dahulu agar mempermudah pada saat tahap *preprocessing* sehingga tidak adanya kebocoran data uji kedalam data latih (Bouke et al., 2024). Untuk pembagian data disini dilakukan pembagian dengan *ratio* 80:10:10.

```
train_df, temp_df = train_test_split(df, test_size=0.2, stratify=df['labels'], random_state=42, shuffle=True)
valid_df, test_df = train_test_split(temp_df, test_size=0.5, stratify=temp_df['labels'], random_state=42, shuffle=True)

print("Jumlah total data:", len(df))
print("Jumlah train:", len(train_df))
print("Jumlah validasi:", len(valid_df))
print("Jumlah test:", len(test_df))
```

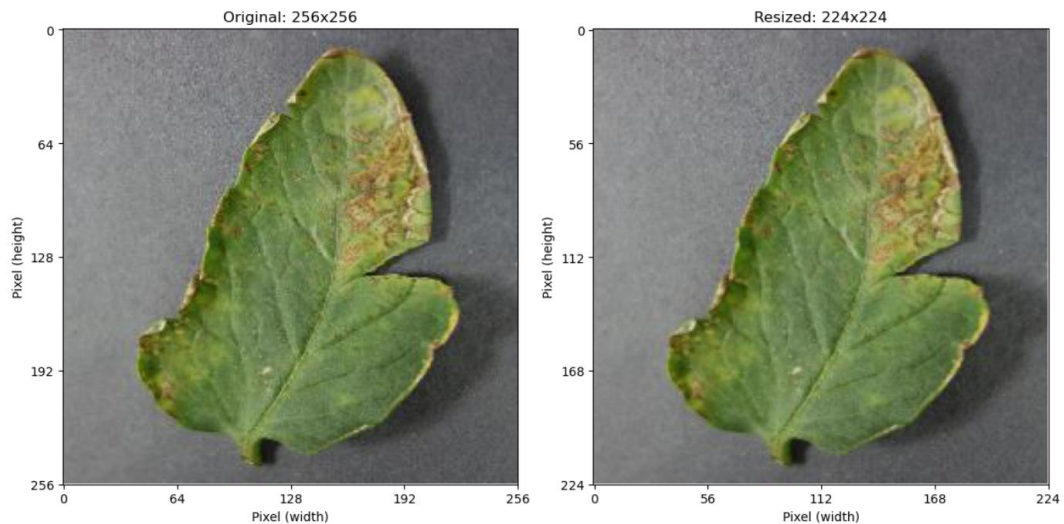
Jumlah total data: 4000
 Jumlah train: 3200
 Jumlah validasi: 400
 Jumlah test: 400

Gambar 4. 2 Pembagian *Dataset*

Dapat dilihat pada gambar diatas bahwa *dataset* dibagi menjadi tiga bagian yaitu data latih, data uji, dan data validasi. Dimana data latih berjumlah sebanyak 3200 citra dengan *ratio* 80% dari total *dataset*, data uji sebanyak 400 citra dan data validasi sebanyak 400 citra.

Setelah dilakukan pembagian dataset tahapan selanjutnya dilakukan *preprocessing* terhadap citra yg digunakan. *Preprocessing* yang dilakukan yaitu *rescale* dan *resize*. Adapun tujuan dari penggunaan *rescale* adalah untuk menormalisasikan nilai piksel dari skala asli [0, 255] menjadi skala baru [0. 1]. Penskalaan ini menawarkan beberapa manfaat seperti optimisasi perhitungan, stabilitas numerik dan menyediakan normaliasi data. Selain itu juga digunakan *resize* untuk mengubah ukuran citra dengan tujuan untuk mempercepat proses

komputasi dan menyesuaikan ukuran input citra terhadap model yang digunakan. Dalam penelitian ini penulis menggunakan *resize* untuk mengubah citra menjadi 224x224 piksel sesuai dengan standar *input* yang digunakan pada model *CNN*.



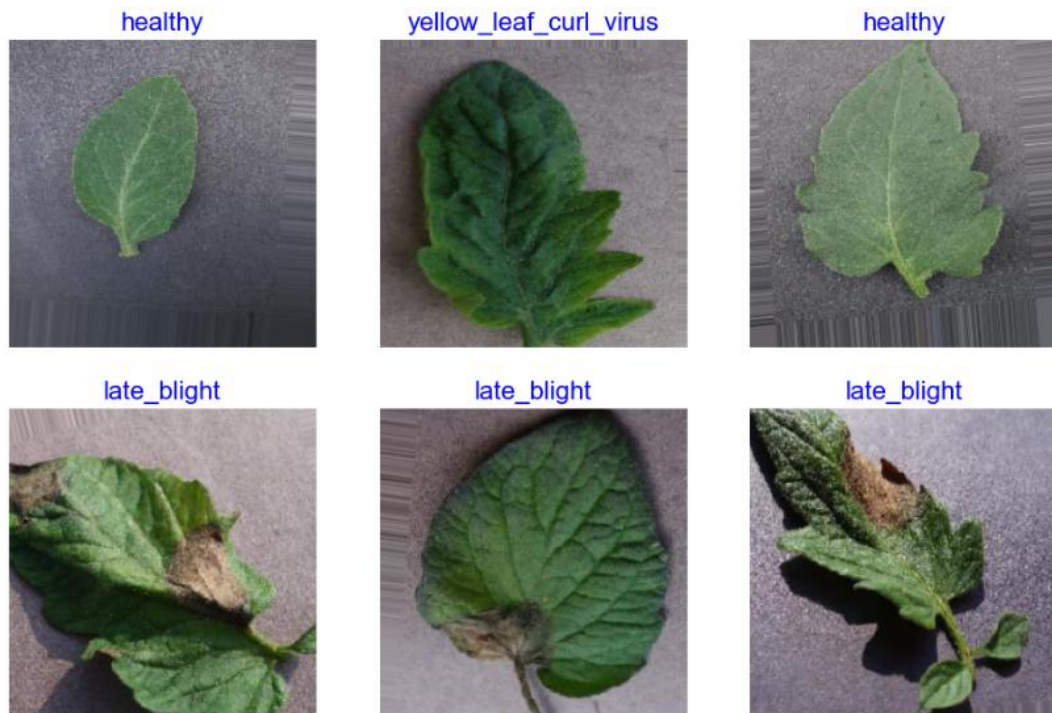
Gambar 4. 3 Hasil *Resize*

Dataset yang telah melalui proses pembagian dan normalisasi selanjutnya dilakukan tahap augmentasi guna menambah variasi data latih. Proses augmentasi ini bertujuan untuk memperbanyak jumlah data citra dengan cara melakukan transformasi terhadap citra yang ada, sehingga model dapat mengenali berbagai variasi objek yang serupa. Pada penelitian ini, augmentasi diterapkan menggunakan fungsi *ImageDataGenerator*. Fungsi ini digunakan untuk menerapkan beberapa teknik augmentasi yang meliputi *rotation*, *width shift*, *height shift*, *shear*, *zoom*, *horizontal flip*, dan *fill mode*. Berikut merupakan deskripsi dari teknik-teknik yang digunakan:

```
gen=ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Gambar 4. 4 Teknik Augmentasi

Berikut adalah gambar dari proses augmentasi:



Gambar 4. 5 Hasil Augmentasi

Berikut adalah deskripsi dari teknik augmentasi yang digunakan:

Tabel 4. 1 Teknik Augmentasi

No	Teknik Augmentasi	Keterangan
1	Rotation	Berfungsi untuk mengubah rotasi gambar kekiri dan kenanan sebesar 15°
2	Width Shift	Berfungsi untuk mengatur posisi gambar pada lebar gambar sebesar 10%
3	Height Shift	Berfungsi untuk mengatur posisi gambar pada tinggi gambar sebesar 10%
4	Shear	Berfungsi meregangkan citra kekanan dan ke kiri sebesar 20%
5	Zoom	Berfungsi untuk memperbesar ukuran citra sebesar 20%

6	Horizontal Flip	Berfungsi untuk memutar citra secara horizontal
7	Fill_Mode	Berfungsi mengisi kokosongan piksel setelah dilakukan transformasi citra

4.3 Rancangan Model CNN

Setelah melalui beberapa tahapan, maka selanjutnya data yang telah diproses akan di-training. penulis menggunakan pendekatan *transfer learning* dengan memanfaatkan arsitektur *pretrained model* yang telah dilatih sebelumnya pada *dataset* berskala besar, yaitu ImageNet. *Base Model* yang digunakan dalam penelitian ini adalah *VGG16*, *VGG19* dan *Xception*. Ketiga model tersebut dimuat tanpa menyertakan bagian *fully connected (FC) layer* terakhir atau *top layer*, sehingga dapat dimodifikasi sesuai kebutuhan klasifikasi pada dataset citra daun tomat.

```

create the model

base_model=tf.keras.applications.Xception(include_top=False, weights="imagenet", input_tensor=Input(shape=(224,224,3)))

base_model.trainable = False

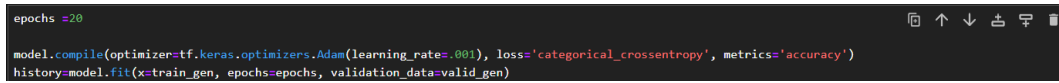
model_name='Xception'
print("Building model with", base_model)
model = tf.keras.Sequential([
    # This is the first convolution
    base_model,

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(4, activation='softmax')
])

```

Gambar 4. 6 Pembuatan Model

Seluruh Penelitian dilakukan dengan mempertahankan arsitektur inti dari *pretrained model*, serta hanya melakukan pelatihan ulang (*fine-tuning*) pada bagian *top layer* yang telah dimodifikasi. Strategi ini dipilih untuk menjaga efisiensi komputasi dan memanfaatkan fitur-fitur yang telah dipelajari dari *pretrained model* sebelumnya. Untuk memperoleh performa model terbaik, penulis juga melakukan beberapa tahap *tuning hyperparameter*. Penyesuaian ini dilakukan untuk mengetahui kombinasi parameter yang menghasilkan akurasi dan stabilitas pelatihan yang optimal. Adapun beberapa *hyperparameter* yang dilakukan *tuning* yaitu *epoch*, *optimizer* dan *learning rate*.



```
epochs = 20
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=.001), loss='categorical_crossentropy', metrics='accuracy')
history=model.fit(x=train_gen, epochs=epochs, validation_data=valid_gen)
```

Gambar 4. 7 Hyperparameter

4.4 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 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.

4.4.1 Pengujian Model *VGG16*

Pada pengujian model *VGG16*, terdapat dua grafik utama yang biasanya digunakan untuk memvisualisasikan performa model selama proses pelatihan, yaitu grafik *loss* dan grafik *accuracy* sebagai berikut :

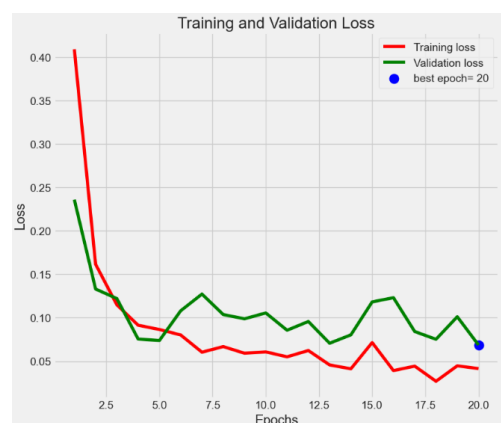
```

Epoch 5/20
100/100 [=====] - 27s 266ms/step - loss: 0.1803 - accuracy: 0.9456 - val_loss: 0.1924 - val_accuracy: 0.9450
Epoch 6/20
100/100 [=====] - 26s 261ms/step - loss: 0.1684 - accuracy: 0.9447 - val_loss: 0.0543 - val_accuracy: 0.9850
Epoch 7/20
100/100 [=====] - 26s 260ms/step - loss: 0.1617 - accuracy: 0.9569 - val_loss: 0.1153 - val_accuracy: 0.9625
Epoch 8/20
100/100 [=====] - 27s 265ms/step - loss: 0.1458 - accuracy: 0.9563 - val_loss: 0.1278 - val_accuracy: 0.9625
Epoch 9/20
100/100 [=====] - 27s 266ms/step - loss: 0.1361 - accuracy: 0.9603 - val_loss: 0.0915 - val_accuracy: 0.9750
Epoch 10/20
100/100 [=====] - 27s 265ms/step - loss: 0.1130 - accuracy: 0.9644 - val_loss: 0.1059 - val_accuracy: 0.9625
Epoch 11/20
100/100 [=====] - 26s 264ms/step - loss: 0.1049 - accuracy: 0.9666 - val_loss: 0.0614 - val_accuracy: 0.9875
Epoch 12/20
100/100 [=====] - 27s 266ms/step - loss: 0.1030 - accuracy: 0.9672 - val_loss: 0.0668 - val_accuracy: 0.9825
Epoch 13/20
100/100 [=====] - 29s 288ms/step - loss: 0.0899 - accuracy: 0.9716 - val_loss: 0.1538 - val_accuracy: 0.9650
Epoch 14/20
100/100 [=====] - 26s 264ms/step - loss: 0.1016 - accuracy: 0.9697 - val_loss: 0.0545 - val_accuracy: 0.9850
Epoch 15/20
100/100 [=====] - 27s 266ms/step - loss: 0.0890 - accuracy: 0.9716 - val_loss: 0.0928 - val_accuracy: 0.9750
Epoch 16/20
100/100 [=====] - 26s 259ms/step - loss: 0.0992 - accuracy: 0.9700 - val_loss: 0.0477 - val_accuracy: 0.9925
Epoch 17/20
100/100 [=====] - 27s 269ms/step - loss: 0.0543 - accuracy: 0.9803 - val_loss: 0.0471 - val_accuracy: 0.9875
Epoch 18/20
100/100 [=====] - 26s 259ms/step - loss: 0.0761 - accuracy: 0.9766 - val_loss: 0.0516 - val_accuracy: 0.9850
Epoch 19/20
100/100 [=====] - 28s 283ms/step - loss: 0.0908 - accuracy: 0.9737 - val_loss: 0.0623 - val_accuracy: 0.9725
Epoch 20/20
100/100 [=====] - 26s 261ms/step - loss: 0.0739 - accuracy: 0.9772 - val_loss: 0.0889 - val_accuracy: 0.9800

```

Gambar 4. 8 Hasil *Training* Dan *Validation VGG16*

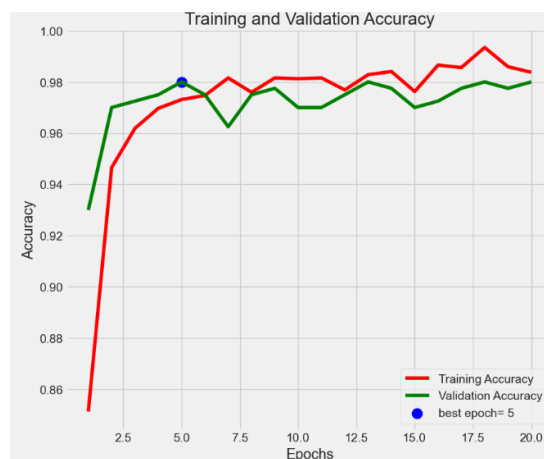
Gambar 4.8 menunjukkan proses *training* dengan menggunakan 20 *epoch* Dimana dilakukan dalam 100 iterasi, dengan setiap iterasi memproses *batch* dengan ukuran *batch* sebesar 32. Waktu yang dibutuhkan untuk menyelesaikan setiap iterasi berkisar antara 26 sampai 28 detik. Nilai *loss* pada *training* berkisar antara 0.0270 hingga 0.4090, menunjukkan seberapa baik model menyesuaikan data pelatihan. Akurasi pada saat *training* bervariasi antara 0.8512 hingga 0.9866, menunjukkan tingkat keberhasilan model dalam memprediksi *output* yang benar pada data pelatihan. Adapun nilai *validation loss* berkisar antara 0.0686 hingga 0.2360 dengan nilai *validation accuracy* berkisar dari 0.9300 hingga 0.9800.



Gambar 4. 9 Loss *VGG16*

Dapat dilihat pada gambar 4.9 dimana menunjukkan hasil grafik *Training Loss* dan *Validation Loss* selama 20 *epoch* memiliki karakteristik berikut: *Training loss* (garis merah) menurun signifikan selama sekitar 3 *epoch* pertama, menunjukkan bahwa model belajar dan memperbaiki kinerjanya pada data *training*. *Training loss* mulai turun perlahan dari sekitar *epoch* 4 dan seterusnya hingga mendapatkan hasil *loss* terendah pada *epoch* 18 dengan nilai *loss* sebesar 0.0270 .

Validation loss (garis hijau) mulai dengan nilai lebih rendah daripada *training loss* yaitu sekitar 0.236 dan mulai turun perlahan hingga *epoch* ke-5. Namun, mulai dari *epoch* ke-6 hingga sekitar *epoch* ke-12, *validation loss* mengalami sedikit fluktuasi, naik menjadi 0.1081 di *epoch* ke-6 dan bahkan mencapai 0.1273 di *epoch* ke-7. Meskipun demikian, fluktuasi ini tidak menunjukkan indikasi *overfitting* yang serius karena nilai *validation loss* tidak terus meningkat dan tetap berada dalam rentang yang relatif rendah. Selain itu, *training loss* tetap menurun secara bertahap dan stabil di bawah angka 0.07, menunjukkan bahwa model masih melakukan pembelajaran yang efisien terhadap data pelatihan.



Gambar 4. 10 Accuracy VGG16

Pada gambar 4.10, tepatnya *epoch* pertama, model mencatat *training accuracy* sebesar 85,12% dan *validation accuracy* sebesar 93,00%. Hal ini menandakan bahwa model cukup cepat dalam mengenali pola dasar dari data, bahkan sejak *epoch* awal. Seiring bertambahnya *epoch*, akurasi pelatihan meningkat secara signifikan. Pada *epoch* ke-5, *training accuracy* mencapai 97,31%,

sementara *validation accuracy* juga terus meningkat hingga 98,00%. Setelah itu, nilai akurasi pada kedua metrik ini cenderung stabil dengan fluktuasi yang sangat kecil.

Hingga *epoch* ke-20, *training accuracy* mencapai 98,37%, sedangkan *validation accuracy* bertahan di angka tinggi sebesar 98,00%. Kestabilan nilai akurasi ini menunjukkan bahwa model mampu belajar dengan baik dari data pelatihan sekaligus melakukan generalisasi secara efektif terhadap data validasi. Tidak ditemukan adanya gap besar antara akurasi pelatihan dan validasi, yang berarti tidak terjadi *overfitting* selama proses pelatihan berlangsung. Dengan performa akurasi yang konsisten tinggi di kedua sisi (*training* dan *validation*), dapat disimpulkan bahwa model berhasil mencapai tingkat prediksi yang sangat baik, serta memiliki potensi yang kuat untuk digunakan pada data baru yang belum pernah dilihat sebelumnya.

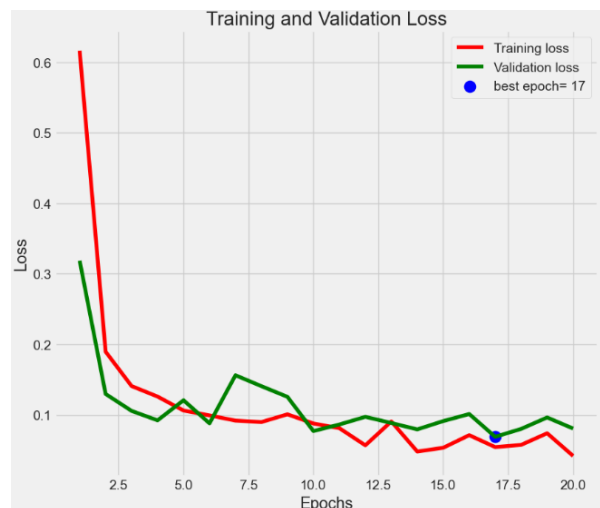
4.4.2 Pengujian Model VGG19

Pada pengujian model *VGG19*, 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.

```
Epoch 5/20
100/100 [=====] - 28s 276ms/step - loss: 0.1063 - accuracy: 0.9684 - val_loss: 0.1209 - val_accuracy: 0.9600
Epoch 6/20
100/100 [=====] - 27s 272ms/step - loss: 0.0994 - accuracy: 0.9659 - val_loss: 0.0882 - val_accuracy: 0.9775
Epoch 7/20
100/100 [=====] - 30s 295ms/step - loss: 0.0920 - accuracy: 0.9675 - val_loss: 0.1563 - val_accuracy: 0.9525
Epoch 8/20
100/100 [=====] - 27s 272ms/step - loss: 0.0900 - accuracy: 0.9681 - val_loss: 0.1408 - val_accuracy: 0.9475
Epoch 9/20
100/100 [=====] - 31s 310ms/step - loss: 0.1010 - accuracy: 0.9634 - val_loss: 0.1257 - val_accuracy: 0.9550
Epoch 10/20
100/100 [=====] - 28s 277ms/step - loss: 0.0879 - accuracy: 0.9678 - val_loss: 0.0772 - val_accuracy: 0.9675
Epoch 11/20
100/100 [=====] - 33s 331ms/step - loss: 0.0813 - accuracy: 0.9731 - val_loss: 0.0865 - val_accuracy: 0.9825
Epoch 12/20
100/100 [=====] - 30s 302ms/step - loss: 0.0570 - accuracy: 0.9778 - val_loss: 0.0974 - val_accuracy: 0.9725
Epoch 13/20
100/100 [=====] - 31s 304ms/step - loss: 0.0908 - accuracy: 0.9703 - val_loss: 0.0887 - val_accuracy: 0.9725
Epoch 14/20
100/100 [=====] - 28s 278ms/step - loss: 0.0482 - accuracy: 0.9834 - val_loss: 0.0796 - val_accuracy: 0.9775
Epoch 15/20
100/100 [=====] - 33s 327ms/step - loss: 0.0536 - accuracy: 0.9819 - val_loss: 0.0914 - val_accuracy: 0.9775
Epoch 16/20
100/100 [=====] - 28s 282ms/step - loss: 0.0714 - accuracy: 0.9753 - val_loss: 0.1014 - val_accuracy: 0.9800
Epoch 17/20
100/100 [=====] - 32s 322ms/step - loss: 0.0545 - accuracy: 0.9809 - val_loss: 0.0691 - val_accuracy: 0.9850
Epoch 18/20
100/100 [=====] - 29s 287ms/step - loss: 0.0576 - accuracy: 0.9797 - val_loss: 0.0803 - val_accuracy: 0.9800
Epoch 19/20
100/100 [=====] - 29s 293ms/step - loss: 0.0741 - accuracy: 0.9741 - val_loss: 0.0965 - val_accuracy: 0.9825
Epoch 20/20
100/100 [=====] - 28s 279ms/step - loss: 0.0420 - accuracy: 0.9869 - val_loss: 0.0808 - val_accuracy: 0.9750
```

Gambar 4. 11 Hasil *Training* dan *Validation VGG19*

Gambar 4.11 Menunjukkan proses pelatihan model dengan 100 iterasi, dengan setiap iterasi memproses batch yang berukuran 32. Waktu yang dibutuhkan untuk menyelesaikan setiap iterasi berkisar antara 27 sampai 38 detik. Nilai *loss* pada *training* berkisar antara 0.0420 hingga 0.6163, menunjukkan seberapa baik model menyesuaikan data pelatihan. Akurasi pada saat *training* bervariasi antara 0.7966 hingga 0.9869, menunjukkan tingkat keberhasilan model dalam memprediksi *output* yang benar pada data pelatihan. Adapun nilai *validation loss* berkisar antara 0.0691 hingga 0.3187 dengan nilai *validation accuracy* berkisar dari 0.9025 hingga 0.9850.

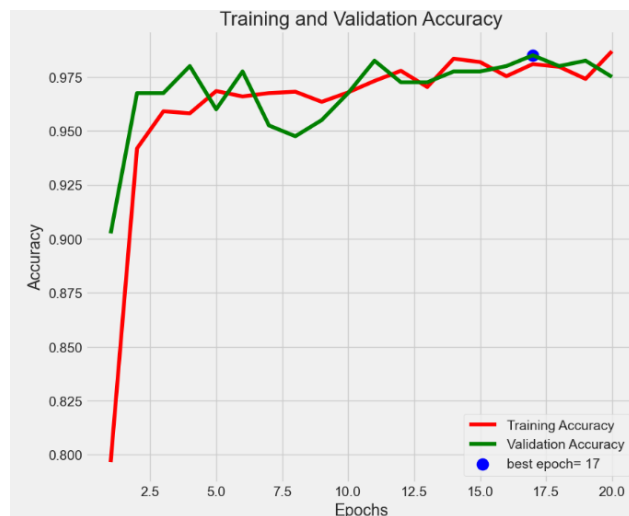


Gambar 4. 12 *Loss VGG19*

Gambar di atas menunjukkan grafik perbandingan antara nilai *training loss* dan *validation loss* selama proses pelatihan model selama 20 epoch. Berdasarkan grafik tersebut, dapat diamati bahwa kedua kurva *loss* mengalami penurunan secara signifikan pada awal pelatihan, yang menandakan bahwa model berhasil mempelajari pola-pola dasar dari data input secara cepat.

Pada *epoch* pertama, nilai *training loss* masih cukup tinggi, yaitu di atas 0.6, sedangkan *validation loss* berada di kisaran 0.3. Namun, pada beberapa *epoch* berikutnya, kedua nilai tersebut menunjukkan penurunan yang cukup tajam. Hal ini mengindikasikan bahwa model mengalami peningkatan dalam proses pembelajaran.

Setelah memasuki *epoch* pertengahan (sekitar *epoch* ke-5 hingga ke-15), grafik menunjukkan adanya fluktuasi ringan pada *validation loss*, sedangkan *training loss* tetap berada dalam tren menurun yang relatif stabil. Meskipun terdapat fluktuasi, *validation loss* tidak mengalami lonjakan yang signifikan dan tetap berada dalam kisaran rendah (sekitar 0.07 hingga 0.15), yang menunjukkan bahwa tidak terdapat indikasi overfitting yang serius.



Gambar 4. 13 Accuracy VGG19

Pada awal pelatihan, yakni *epoch* ke-1, nilai *training accuracy* tercatat sebesar 79,66%, sementara *validation accuracy* sudah mencapai 90,25%. Hal ini mengindikasikan bahwa meskipun model masih dalam tahap awal pembelajaran, kemampuan awal dalam mengenali pola pada data validasi sudah cukup baik. Hal ini dapat terjadi karena bobot awal model (hasil *pre-trained*) sudah membawa pengetahuan umum dari domain lain.

Seiring bertambahnya *epoch*, *training accuracy* meningkat secara signifikan. Pada *epoch* ke-4, nilai *training accuracy* mencapai 95,81%, dan *validation accuracy* mencapai 98,00%, menunjukkan bahwa model telah berhasil mempelajari pola secara lebih akurat dan mampu mengenali data validasi dengan baik. Meskipun pada *epoch* ke-5 hingga *epoch* ke-9 terdapat sedikit fluktuasi pada *validation accuracy* (misalnya turun ke 94,75% pada *epoch* ke-8), nilai tersebut masih tetap tinggi dan stabil. Sementara itu, *training accuracy* juga tetap berada di

atas 96%, menunjukkan bahwa model tetap mampu melakukan klasifikasi dengan baik terhadap data pelatihan tanpa menunjukkan tanda-tanda *overfitting* yang parah.

Pada fase akhir pelatihan, nilai *training accuracy* dan *validation accuracy* mencapai angka yang sangat tinggi. Misalnya, pada *epoch* ke-17, *training accuracy* mencapai 98,09%, dan *validation accuracy* meningkat hingga 98,50%, yang merupakan nilai tertinggi selama proses pelatihan. Hingga *epoch* ke-20, nilai *training accuracy* meningkat menjadi 98,69%, sedangkan *validation accuracy* sedikit menurun ke 97,50%, namun masih tergolong sangat baik dan menunjukkan stabilitas performa model.

4.4.3 Pengujian Model *Xception*

Pada pengujian model *Xception*, digunakan dua grafik utama untuk memvisualisasikan performa model pada saat proses pelatihan, yaitu grafik loss dan grafik accuracy. Grafik loss bertujuan untuk melihat bagaimana nilai kerugian (loss) model berubah seiring waktu pelatihan, yang membantu memahami seberapa baik model 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.

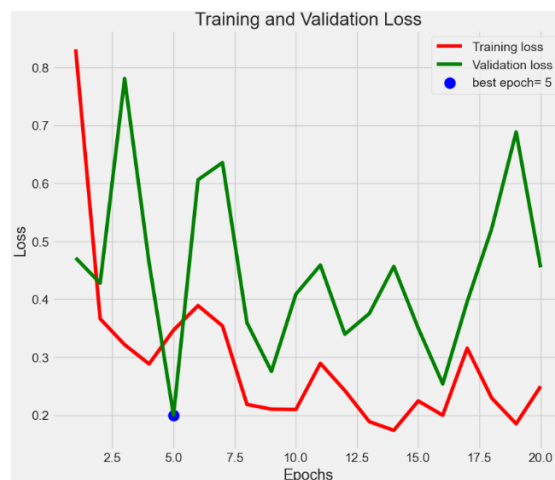

```

Epoch 5/20
100/100 [=====] - 26s 257ms/step - loss: 0.3466 - accuracy: 0.9519 - val_loss: 0.1997 - val_accuracy: 0.9650
Epoch 6/20
100/100 [=====] - 25s 250ms/step - loss: 0.3889 - accuracy: 0.9513 - val_loss: 0.6060 - val_accuracy: 0.9450
Epoch 7/20
100/100 [=====] - 26s 256ms/step - loss: 0.3539 - accuracy: 0.9622 - val_loss: 0.6356 - val_accuracy: 0.9525
Epoch 8/20
100/100 [=====] - 25s 251ms/step - loss: 0.2184 - accuracy: 0.9709 - val_loss: 0.3593 - val_accuracy: 0.9675
Epoch 9/20
100/100 [=====] - 26s 257ms/step - loss: 0.2102 - accuracy: 0.9744 - val_loss: 0.2754 - val_accuracy: 0.9725
Epoch 10/20
100/100 [=====] - 27s 271ms/step - loss: 0.2096 - accuracy: 0.9747 - val_loss: 0.4086 - val_accuracy: 0.9600
Epoch 11/20
100/100 [=====] - 26s 259ms/step - loss: 0.2893 - accuracy: 0.9691 - val_loss: 0.4588 - val_accuracy: 0.9475
Epoch 12/20
100/100 [=====] - 27s 269ms/step - loss: 0.2423 - accuracy: 0.9744 - val_loss: 0.3395 - val_accuracy: 0.9775
Epoch 13/20
100/100 [=====] - 27s 270ms/step - loss: 0.1887 - accuracy: 0.9806 - val_loss: 0.3752 - val_accuracy: 0.9625
Epoch 14/20
100/100 [=====] - 26s 255ms/step - loss: 0.1737 - accuracy: 0.9809 - val_loss: 0.4564 - val_accuracy: 0.9700
Epoch 15/20
100/100 [=====] - 27s 270ms/step - loss: 0.2244 - accuracy: 0.9769 - val_loss: 0.3499 - val_accuracy: 0.9725
Epoch 16/20
100/100 [=====] - 25s 253ms/step - loss: 0.1997 - accuracy: 0.9828 - val_loss: 0.2538 - val_accuracy: 0.9725
Epoch 17/20
100/100 [=====] - 26s 254ms/step - loss: 0.3154 - accuracy: 0.9712 - val_loss: 0.3958 - val_accuracy: 0.9550
Epoch 18/20
100/100 [=====] - 25s 252ms/step - loss: 0.2294 - accuracy: 0.9794 - val_loss: 0.5216 - val_accuracy: 0.9675
Epoch 19/20
100/100 [=====] - 25s 252ms/step - loss: 0.1851 - accuracy: 0.9819 - val_loss: 0.6884 - val_accuracy: 0.9600
Epoch 20/20
100/100 [=====] - 26s 257ms/step - loss: 0.2496 - accuracy: 0.9769 - val_loss: 0.4549 - val_accuracy: 0.9650

```

Gambar 4. 14 Hasil *Training* dan *Validation Xception*

Gambar diatas Menunjukkan proses pelatihan model dengan 100 iterasi, dengan setiap iterasi memproses batch yang berukuran 32. Waktu yang dibutuhkan untuk menyelesaikan setiap iterasi berkisar antara 25 sampai 27 detik. Nilai *loss* pada *training* berkisar antara 0.1737 hingga 0.8295, menunjukkan seberapa baik model menyesuaikan data pelatihan. Akurasi pada saat *training* bervariasi antara 0.7966 hingga 0.9828, menunjukkan tingkat keberhasilan model dalam memprediksi *output* yang benar pada data pelatihan. Adapun nilai *validation loss* berkisar antara 0.1997 hingga 0.7852 dengan nilai *validation accuracy* berkisar dari 0.9450 hingga 0.9775.

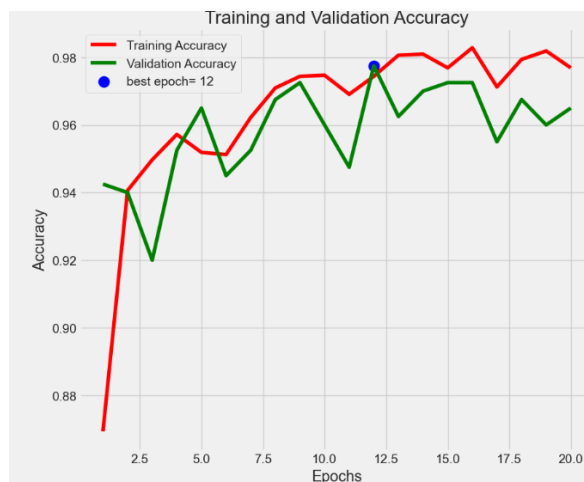


Gambar 4. 15 Loss *Xception*

Berdasarkan hasil pada gambar 4.15 yang menunjukkan grafik dari “*Training and Validation Loss*”, kita dapat melihat bagaimana *loss* dari model berubah selama 20 *epoch* pelatihan. Pada awal pelatihan, *training loss* dimulai dari nilai yang cukup tinggi, yaitu sekitar 0.8295, menandakan bahwa model masih dalam tahap awal memahami pola dari data latih. Seiring bertambahnya *epoch*, nilai *training loss* mengalami penurunan yang cukup konsisten. Penurunan signifikan terjadi hingga mencapai nilai terendah sebesar 0.1737 pada epoch ke-14. Setelah itu, meskipun terdapat sedikit fluktuasi, *training loss* tetap berada dalam kisaran rendah, yang menunjukkan bahwa model mampu mempertahankan performa pembelajaran terhadap data pelatihan secara stabil.

Sementara itu, *validation loss* menunjukkan pola yang sedikit berbeda. Nilai awal *validation loss* relatif tinggi dan mengalami penurunan hingga mencapai titik terendah sebesar 0.1997 pada epoch ke-5. Namun, setelah mencapai titik tersebut, *validation loss* menunjukkan pola yang fluktuatif, dengan beberapa kali peningkatan yang cukup signifikan, di antaranya pada epoch ke-7 (0.6356) dan epoch ke-18 (0.6884). Meskipun pada beberapa epoch nilai *validation loss* kembali menurun, fluktuasi yang terjadi mengindikasikan adanya ketidakstabilan dalam kemampuan generalisasi model terhadap data uji.

Perbedaan antara *training loss* yang menurun stabil dan *validation loss* yang fluktuatif dapat menjadi indikasi awal terjadinya *overfitting*. Hal ini menunjukkan bahwa meskipun model belajar dengan baik terhadap data pelatihan, kemampuannya untuk mempertahankan performa yang konsisten terhadap data yang tidak dilatih masih perlu diperhatikan. Oleh karena itu, evaluasi lebih lanjut dan penyesuaian *hyperparameter* dapat menjadi langkah penting untuk meningkatkan generalisasi model.



Gambar 4. 16 *Accuracy Xception*

Berdasarkan hasil pada Gambar diatas yang menampilkan grafik “*Training and Validation Accuracy*”, dapat diamati bahwa akurasi model mengalami peningkatan yang signifikan pada awal proses pelatihan selama 20 *epoch*. *Training accuracy* meningkat secara konsisten dari nilai awal sekitar 0.8756 dan mencapai nilai puncaknya sebesar 0.9878 pada *epoch* ke-19. Meskipun sempat mengalami sedikit fluktuasi setelah *epoch* ke-10, akurasi pelatihan secara umum tetap tinggi dan stabil, menunjukkan bahwa model mampu mempelajari pola dari data pelatihan dengan baik.

Sementara itu, *validation accuracy* juga menunjukkan peningkatan yang cukup baik, dimulai dari nilai awal sekitar 0.9425 dan mencapai nilai tertinggi sebesar 0.9775 pada *epoch* ke-12. Namun, berbeda dengan akurasi pelatihan, grafik *validation accuracy* menunjukkan adanya fluktuasi yang lebih nyata, terutama setelah *epoch* ke-13. Meskipun demikian, akurasi validasi secara umum tetap berada dalam kisaran tinggi (di atas 0.9550), yang mengindikasikan bahwa model memiliki performa generalisasi yang cukup baik terhadap data uji.

Perbedaan antara *training accuracy* yang stabil dan *validation accuracy* yang sedikit fluktuatif dapat disebabkan oleh beberapa faktor, seperti kompleksitas data validasi atau ketidakseimbangan kelas. Namun secara keseluruhan, performa akurasi pada kedua *dataset* tersebut menunjukkan bahwa model telah mampu belajar dan melakukan klasifikasi dengan tingkat ketepatan yang tinggi

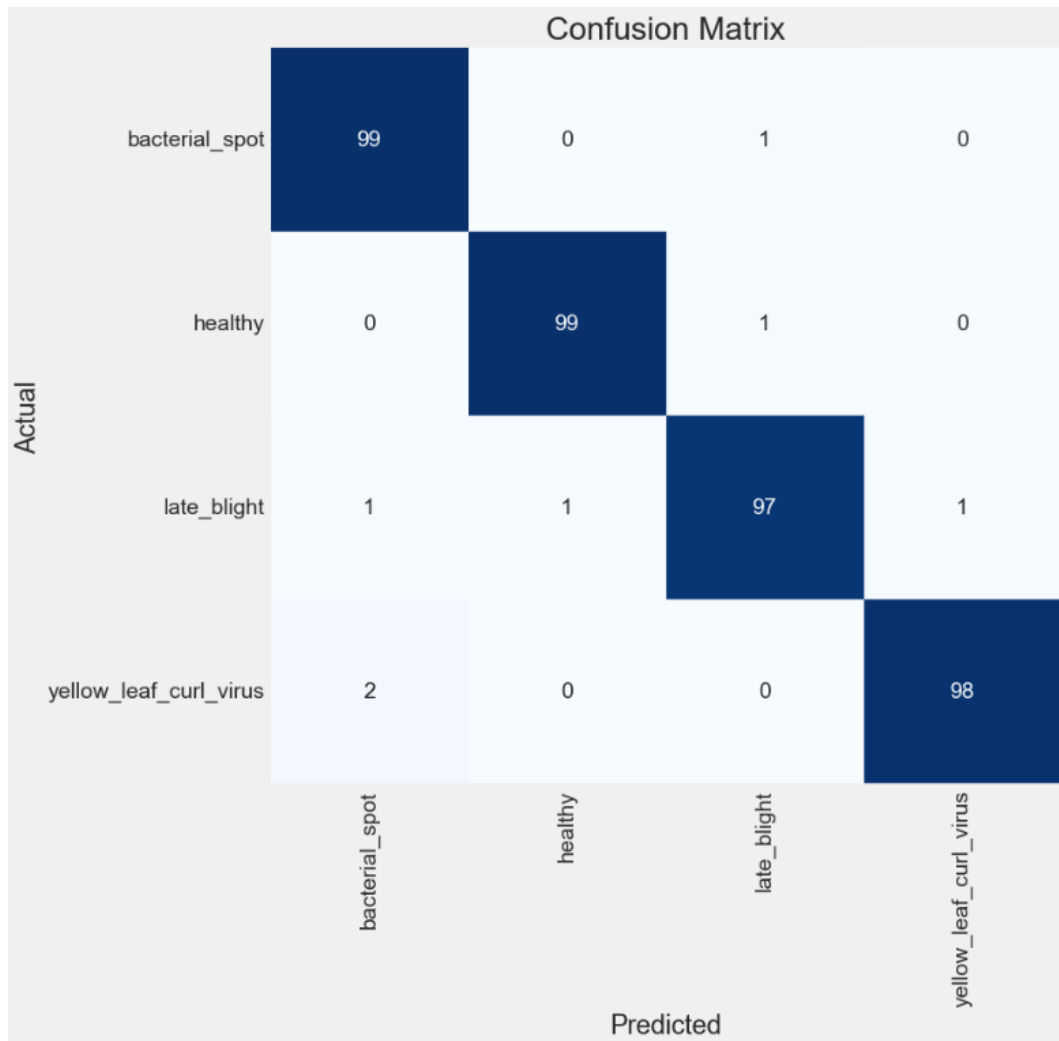
4.5 Evaluasi Model

Tujuan utama dari evaluasi model pada bab ini adalah untuk mengukur kinerja arsitektur *CNN* yang telah dibangun dalam melakukan klasifikasi penyakit pada daun tanaman tomat. Evaluasi dilakukan berdasarkan metrik seperti *accuracy*, *loss*, *precision*, *recall*, dan *F1-score* yang dihitung terhadap data pelatihan dan validasi. Hal ini bertujuan untuk menilai sejauh mana model mampu belajar dari data serta menggeneralisasikan pengetahuannya terhadap data yang belum pernah dilihat sebelumnya.

Selain itu, evaluasi ini bertujuan untuk mengidentifikasi potensi permasalahan seperti *overfitting* atau *underfitting* serta untuk membandingkan performa antar model atau konfigurasi, guna memperoleh arsitektur dan parameter terbaik. Evaluasi semacam ini penting dalam pengembangan sistem klasifikasi berbasis *machine learning* sebagaimana disampaikan oleh Mohammadi et al. (2020), yang menyatakan bahwa evaluasi performa merupakan komponen esensial untuk menjamin efektivitas model dalam aplikasi nyata.

4.5.1 Evaluasi Model *VGG16*

Untuk mengevaluasi performa model *VGG16* dalam klasifikasi penyakit daun tomat, dilakukan pengujian terhadap data uji yang telah dipisahkan sebelumnya. Hasil evaluasi ditampilkan dalam bentuk *confusion matrix* seperti pada Gambar 4.17.



Gambar 4. 17 Hasil *Confusion Matrix VGG16*

Dari gambar *confusion matrix* diatas, model VGG16 menunjukkan performa klasifikasi yang sangat baik dalam mengidentifikasi empat kelas penyakit pada daun tanaman tomat, yaitu *bacterial spot*, *healthy*, *late blight*, dan *yellow leaf curl virus*. Model mampu mengklasifikasikan sebagian besar citra dengan akurasi yang tinggi. Pada kelas *bacterial spot*, sebanyak 99 citra berhasil diklasifikasikan dengan benar, sementara hanya 1 citra yang salah diklasifikasikan sebagai *late blight*. Kelas *healthy* menunjukkan hasil yang hampir sempurna, dengan 99 citra diklasifikasikan dengan benar dan hanya 1 citra yang salah terdeteksi sebagai *late blight*. Untuk kelas *late blight*, model mengklasifikasikan 97 citra dengan benar, sementara masing-masing satu citra salah diklasifikasikan sebagai *healthy* dan

yellow leaf curl virus. Adapun pada kelas *yellow leaf curl virus*, sebanyak 98 citra diklasifikasikan dengan benar, dan 2 citra lainnya keliru dikenali sebagai *bacterial spot*.

Berikut adalah tabel hasil data uji menggunakan model *VGG16*:

Tabel 4. 2 *Confusion Matrix VGG16*

No.	Class	Total Image	TP	TN	FP	FN
1	bacterial_spot	100	99	299	1	1
2	healthy	100	99	299	1	1
3	late_blight	100	97	298	2	3
4	yellow_leaf_curl_virus	100	98	299	1	2

Dari hasil *confusion matrix* tersebut, didapatkan hasil perhitungan *accuracy*, *precision*, *recall*, dan *f1-score* sebagai berikut:

Classification Report:				
	precision	recall	f1-score	support
bacterial_spot	0.97	0.99	0.98	100
healthy	0.99	0.99	0.99	100
late_blight	0.98	0.97	0.97	100
yellow_leaf_curl_virus	0.99	0.98	0.98	100
accuracy			0.98	400
macro avg	0.98	0.98	0.98	400
weighted avg	0.98	0.98	0.98	400

Gambar 4. 18 *Classification Report VGG16*

Berdasarkan Tabel 4.2 dan Gambar 4.18, dapat disimpulkan bahwa model menghasilkan performa prediksi yang sangat baik pada tahap *testing* dengan menggunakan 400 data uji. Model yang digunakan memiliki konfigurasi ukuran citra sebesar 224×224 piksel, jumlah *epoch* sebanyak 20, *batch size* 32, *learning rate* 0,001, dan *optimizer* Adam. Dari konfigurasi tersebut, model mampu mencapai tingkat akurasi sebesar 98%. Nilai ini menunjukkan bahwa model telah mampu mengenali dan mengklasifikasikan citra dengan sangat baik pada data uji.

Sebagai contoh, berikut ini disajikan perhitungan nilai *precision*, *recall*, dan *f1-score* pada kelas “*bacterial_spot*” untuk memberikan gambaran performa model pada masing-masing kelas secara lebih mendalam:

$$Precision = \left(\frac{TP}{TP+FP} \right) = \frac{99}{99+1} = \frac{99}{100} = 0.99$$

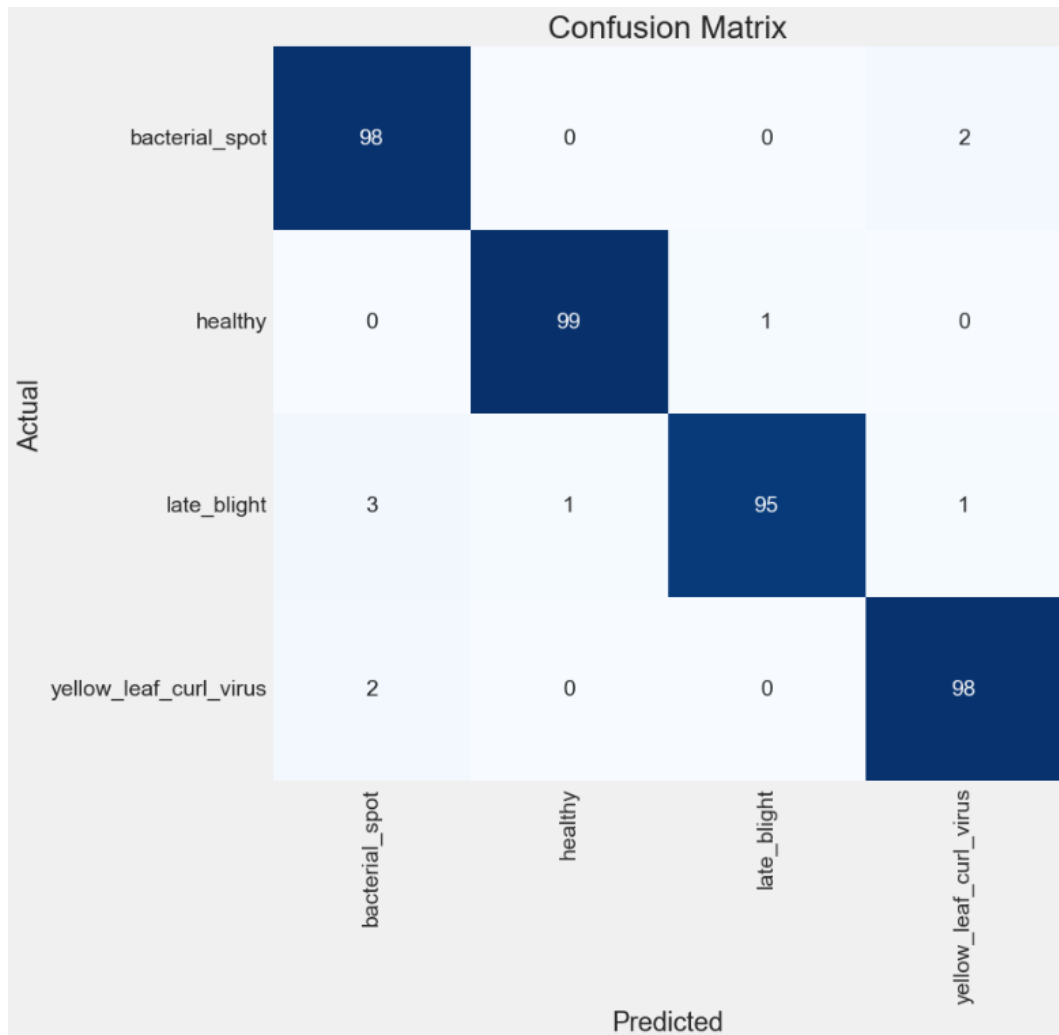
$$Recall = \left(\frac{TP}{TP+FN} \right) = \frac{99}{99+1} = \frac{99}{100} = 0.99$$

$$F-1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) = 2 \times \left(\frac{0.99 \times 0.99}{0.99 + 0.99} \right) = 2 \times \left(\frac{0.9801}{1.98} \right) = 0.99$$

Dari hasil perhitungan yang telah dilakukan, dapat disimpulkan bahwa rasio prediksi benar positif terhadap seluruh hasil prediksi positif (*precision*) pada kelas “*bacterial_spot*” adalah sebesar 0,99. Selanjutnya, rasio prediksi benar positif terhadap seluruh data aktual positif (*recall*) juga menunjukkan nilai sebesar 0,99. Nilai *precision* dan *recall* yang tinggi tersebut menunjukkan bahwa model mampu melakukan klasifikasi dengan tingkat kesalahan yang sangat rendah, khususnya pada kelas tersebut. Selain itu, dapat diketahui bahwa dari 100 data *test* dengan kelas “*bacterial_spot*” terdapat 1 data yang diprediksi “*bacterial_spot*” namun kenyataannya data tersebut tidak termasuk ke dalam kelas “*bacterial_spot*” (*False Positive*) Lalu dari 100 data *test* kelas “*bacterial_spot*” terdapat 1 data dari kelas “*bacterial_spot*” yang masuk atau terklasifikasi pada kelas lain (*False Negative*). Pada *classification report* di Gambar 4.17, seluruh kelas mendapatkan akurasi rata-rata *precision*, *recall*, dan *f1-score* 98%.

4.5.2 Evaluasi Model VGG19

Evaluasi terhadap performa model *VGG19* dilakukan menggunakan data uji, dan hasilnya divisualisasikan dalam bentuk *confusion matrix* seperti ditunjukkan pada Gambar 4.19.



Gambar 4. 19 Hasil *Confusion Matrix VGG19*

Berdasarkan gambar diatas, model berhasil mengklasifikasikan sebagian besar citra dengan tingkat akurasi yang tinggi. Pada kelas *bacterial spot*, sebanyak 98 citra berhasil diklasifikasikan dengan benar, sementara 2 citra salah diklasifikasikan sebagai *yellow leaf curl virus*. Untuk kelas *healthy*, model menunjukkan kinerja yang sangat baik dengan 99 citra diklasifikasikan secara benar, dan hanya 1 citra yang salah diprediksi sebagai *late blight*.

Kelas *late blight* menunjukkan tingkat kesalahan klasifikasi yang sedikit lebih tinggi dibanding kelas lain, dengan 95 citra diklasifikasikan secara benar, sedangkan masing-masing 3 citra salah diprediksi sebagai *bacterial spot*, 1 citra sebagai *healthy*, dan 1 citra lainnya sebagai *yellow leaf curl virus*. Sementara itu,

pada kelas *yellow leaf curl virus*, sebanyak 98 citra dikenali dengan benar dan 2 citra salah diklasifikasikan sebagai *bacterial spot*.

Berikut adalah tabel hasil data uji menggunakan model *VGG19*:

Tabel 4. 3 *Confusion Matrix VGG19*

No	Class	Total Image	TP	TN	FP	FN
1	bacterial_spot	100	98	295	5	2
2	healthy	100	99	299	1	1
3	late_blight	100	95	299	1	5
4	yellow_leaf_curl_virus	100	98	297	3	2

Dari hasil *confusion matrix* tersebut, didapatkan hasil perhitungan *accuracy*, *precision*, *recall*, dan *f1-score* sebagai berikut:

Classification Report:				
	precision	recall	f1-score	support
bacterial_spot	0.95	0.98	0.97	100
healthy	0.99	0.99	0.99	100
late_blight	0.99	0.95	0.97	100
yellow_leaf_curl_virus	0.97	0.98	0.98	100
accuracy			0.97	400
macro avg	0.98	0.97	0.98	400
weighted avg	0.98	0.97	0.98	400

Gambar 4. 20 *Classification Report VGG19*

Berdasarkan Tabel 4.3 dan Gambar 4.18, dapat disimpulkan bahwa model memberikan hasil prediksi yang sangat baik pada proses *testing* dengan jumlah data uji sebanyak 400 citra. Model yang digunakan memiliki konfigurasi ukuran citra sebesar 224×224 piksel, jumlah *epoch* sebanyak 20, *batch size* sebesar 32, *learning rate* 0,001, dan *optimizer* Adam. Dengan konfigurasi tersebut, diperoleh nilai akurasi sebesar 97%.

Sebagai ilustrasi, berikut ini disajikan contoh perhitungan untuk mendapatkan nilai *precision*, *recall*, dan *f1-score* pada kelas "*late_blight*". Nilai *precision* menunjukkan rasio prediksi benar positif terhadap seluruh prediksi positif, sedangkan *recall* menunjukkan rasio prediksi benar positif terhadap seluruh data

aktual yang benar positif. Nilai *f1-score* diperoleh sebagai rata-rata dari nilai *precision* dan *recall*, yang mencerminkan keseimbangan antara keduanya.

$$Precision = \left(\frac{TP}{TP+FP} \right) = \frac{95}{95+1} = \frac{95}{96} \approx 0.989$$

$$Recall = \left(\frac{TP}{TP+FN} \right) = \frac{95}{95+5} = \frac{95}{100} = 0.95$$

$$F-1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) = 2 \times \left(\frac{0.99 \times 0.95}{0.99 + 0.95} \right) = 2 \times \left(\frac{0.9405}{1.94} \right) \approx 0.969$$

Berdasarkan hasil perhitungan pada kelas “*late_blight*”, diperoleh bahwa nilai *precision* mencapai 0,99. Artinya, rasio prediksi benar positif terhadap seluruh hasil prediksi positif sangat tinggi. Sementara itu, nilai *recall* yang diperoleh adalah 0,95, yang menunjukkan bahwa dari seluruh data aktual kelas “*late_blight*”, sebanyak 95% berhasil diprediksi dengan benar oleh model.

Diketahui bahwa dari 100 data uji untuk kelas “*late_blight*”, terdapat 1 data yang diprediksi sebagai “*late_blight*” padahal sebenarnya bukan (False Positive), dan terdapat 5 data “*late_blight*” yang terklasifikasi ke kelas lain (False Negative). Hal ini menunjukkan bahwa meskipun model telah bekerja dengan sangat baik, masih terdapat kesalahan klasifikasi dalam jumlah kecil.

Secara keseluruhan, berdasarkan *classification report* yang ditampilkan pada Gambar 4.18, semua kelas memperoleh rata-rata nilai *precision*, *recall*, dan *f1-score* sebesar 97%. Hasil ini mengindikasikan bahwa model memiliki performa klasifikasi yang sangat baik dan stabil di seluruh kelas target.

4.5.3 Evaluasi Model *Xception*

Berikut adalah gambar dari hasil *confusion matrix* yang akan digunakan untuk mengevaluasi model *Xception*:

		Confusion Matrix			
Actual	bacterial_spot	91	2	1	6
	healthy	0	98	0	2
	late_blight	1	1	98	0
	yellow_leaf_curl_virus	2	0	1	97
		bacterial_spot	healthy	late_blight	yellow_leaf_curl_virus
		Predicted			

Gambar 4. 21 Hasil *Confusion Matrix Xception*

Gambar 4.21 menunjukkan hasil evaluasi model terhadap empat kelas penyakit daun tomat, yaitu *bacterial spot*, *healthy*, *late blight*, dan *yellow leaf curl virus*. Model menunjukkan kinerja klasifikasi yang cukup baik secara keseluruhan, meskipun terdapat beberapa kesalahan klasifikasi antar kelas.

Pada kelas *bacterial spot*, sebanyak 91 citra berhasil diklasifikasikan dengan benar, sementara sisanya mengalami kesalahan klasifikasi ke dalam kelas *healthy* (2 citra), *late blight* (1 citra), dan *yellow leaf curl virus* (6 citra). Kelas *healthy* diklasifikasikan dengan sangat baik, dengan 98 citra dikenali secara akurat dan hanya 2 citra yang salah diklasifikasikan sebagai *yellow leaf curl virus*. Model juga menunjukkan performa tinggi pada kelas *late blight*, dengan 98 citra

diklasifikasikan dengan benar, dan masing-masing satu citra salah terklasifikasi ke dalam *bacterial spot* dan *healthy*.

Sementara itu, pada kelas *yellow leaf curl virus*, sebanyak 97 citra berhasil dikenali dengan tepat, dan masing-masing satu citra salah diklasifikasikan sebagai *bacterial spot* dan *late blight*. Kesalahan klasifikasi yang paling menonjol terdapat pada kelas *bacterial spot*, terutama yang tertukar ke *yellow leaf curl virus*, yang mungkin disebabkan oleh kemiripan visual antara gejala kedua penyakit tersebut.

Berikut adalah tabel hasil data uji menggunakan model *Xception*:

Tabel 4. 4 *Confusion Matrix Xception*

No	Class	Total Image	TP	TN	FP	FN
1	bacterial_spot	100	91	297	3	9
2	healthy	100	98	297	3	2
3	late_blight	100	98	298	2	2
4	yellow_leaf_curl_virus	100	97	292	8	3

Dari hasil *confusion matrix* tersebut, didapatkan hasil perhitungan *accuracy*, *precision*, *recall*, dan *f1-score* sebagai berikut:

Classification Report:				
	precision	recall	f1-score	support
bacterial_spot	0.97	0.91	0.94	100
healthy	0.97	0.98	0.98	100
late_blight	0.98	0.98	0.98	100
yellow_leaf_curl_virus	0.92	0.97	0.95	100
accuracy			0.96	400
macro avg	0.96	0.96	0.96	400
weighted avg	0.96	0.96	0.96	400

Gambar 4. 22 *Classification Report Xception*

Berdasarkan Tabel 4.4 dan Gambar 4.19, dapat disimpulkan bahwa model memberikan hasil prediksi yang sangat baik pada proses *testing* dengan jumlah data uji sebanyak 400 citra. Model yang digunakan memiliki konfigurasi berupa ukuran citra 224×224 piksel, jumlah *epoch* sebanyak 20, *batch size* 32, *learning rate* sebesar 0,001, dan *optimizer* Adam. Dengan konfigurasi tersebut, diperoleh nilai

akurasi sebesar 96%. Untuk mengukur kinerja model secara lebih spesifik, dilakukan perhitungan terhadap metrik evaluasi lainnya seperti *precision*, *recall*, dan *f1-score*. Berikut merupakan contoh perhitungan pada kelas “*bacterial_spot*”:

$$Precision = \left(\frac{TP}{TP+FP} \right) = \frac{91}{91+3} = \frac{91}{94} \approx 0.968$$

$$Recall = \left(\frac{TP}{TP+FN} \right) = \frac{91}{91+9} = \frac{91}{100} = 0.91$$

$$F-1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) = 2 \times \left(\frac{0.968 \times 0.91}{0.968 + 0.91} \right) = 2 \times \left(\frac{0.88}{1.878} \right) \approx 0.938$$

Dari hasil perhitungan pada kelas “*bacterial_spot*”, dapat diketahui bahwa nilai *precision* sebesar 0,97 menunjukkan bahwa 97% dari data yang diprediksi sebagai “*bacterial_spot*” merupakan prediksi yang benar. Sementara itu, nilai *recall* sebesar 0,91 mengindikasikan bahwa sebanyak 91% dari seluruh data aktual kelas “*bacterial_spot*” berhasil diklasifikasikan dengan tepat oleh model.

Selain itu, dari 100 data uji pada kelas “*bacterial_spot*”, terdapat 3 data yang diprediksi sebagai “*bacterial_spot*”, padahal sebenarnya berasal dari kelas lain (*False Positive*). Sedangkan sebanyak 9 data dari kelas “*bacterial_spot*” justru terklasifikasi ke dalam kelas lain (*False Negative*). Temuan ini mengindikasikan bahwa meskipun model memiliki performa yang sangat baik, masih terdapat kesalahan prediksi, terutama dalam membedakan kelas yang memiliki kemiripan fitur.

Secara keseluruhan, berdasarkan hasil *classification report* yang ditampilkan pada Gambar 4.19, seluruh kelas memperoleh rata-rata nilai *precision*, *recall*, dan *f1-score* sebesar 96%. Hal ini menunjukkan bahwa model memiliki performa klasifikasi yang cukup stabil dan akurat dalam mengenali masing-masing kelas pada data uji.

4.5.4 Hasil Evaluasi

Tabel 4. 5 Hasil Evaluasi

No.	Model	Accuracy (%)	Precision	Recall	F1-Score
1	VGG16	98.25	0.98	0.98	0.98
2	VGG19	97.50	0.97	0.97	0.97
3	Xception	95.99	0.95	0.95	0.95

Tabel 4.5 menyajikan perbandingan hasil evaluasi dari tiga model *CNN*, yaitu *VGG16*, *VGG19*, dan *Xception*, berdasarkan empat metrik utama: *Accuracy*, *Precision*, *Recall*, dan *F1-Score*. Metrik-metrik ini dihitung berdasarkan hasil pengujian pada data uji, dan masing-masing nilai merupakan rata-rata dari keempat kelas yang digunakan dalam klasifikasi citra daun tanaman tomat.

Model *VGG16* menunjukkan performa terbaik dengan akurasi tertinggi, yaitu 98,25%, serta nilai *precision*, *recall*, dan *F1-score* sebesar 0,98. Hal ini menunjukkan bahwa model *VGG16* tidak hanya mampu mengklasifikasikan data dengan sangat akurat, tetapi juga memiliki konsistensi dalam mengenali kelas yang benar dan meminimalkan kesalahan prediksi. Model *VGG19* menempati posisi kedua dengan akurasi 97,50%, serta nilai *precision*, *recall*, dan *F1-score* sebesar 0,97. Meskipun performanya sedikit di bawah *VGG16*, model ini tetap menunjukkan kinerja klasifikasi yang sangat baik dan stabil.

Sementara itu, model *Xception* mencatat akurasi terendah, yaitu 95,99%, dengan *precision*, *recall*, dan *F1-score* masing-masing sebesar 0,95. Penurunan ini kemungkinan disebabkan oleh kompleksitas arsitektur *Xception* yang belum sepenuhnya sesuai dengan karakteristik *dataset* yang peneliti digunakan, atau adanya pengaruh dari parameter pelatihan seperti *learning rate* maupun *overfitting* ringan yang terlihat dari fluktuasi nilai loss selama pelatihan.

Secara keseluruhan, hasil evaluasi ini menunjukkan bahwa *VGG16* merupakan model yang paling optimal dalam melakukan klasifikasi citra daun tanaman tomat pada penelitian ini, baik dari sisi akurasi maupun kualitas prediksi secara keseluruhan.

4.6 Pengaruh *HyperParameter* pada Model

Proses *training* pada jaringan diatur dengan mendefinisikan beberapa variabel yang disebut *hyperparameter*, seperti *learning rate*, jumlah *hidden layer*, jumlah *neuron*, fungsi aktivasi, jumlah *training epoch* dan lainnya (Haq et al., 2022). *Hyperparameter* sangat memberikan dampak pada pelatihan serta kinerja model. Oleh karena itu, menetapkan *hyperparameter* terbaik merupakan langkah yang penting dalam membuat model yang akan menghasilkan akurasi dan prediksi yang baik. Dalam penelitian ini kombinasi *hyperparameter* yang digunakan diantaranya jumlah *epoch*, *learning rate* dan *optimizer*.

4.6.1 Jumlah *Training Epoch*

Berdasarkan proses pelatihan model, *epoch* berperan sebagai siklus pembelajaran di mana seluruh data latih diproses sebanyak satu kali penuh. Penggunaan jumlah *epoch* yang terlalu sedikit dapat menyebabkan model tidak mampu mengenali pola secara optimal, yang mengakibatkan *underfitting*. Sebaliknya, jika jumlah *epoch* terlalu banyak, model berisiko mengalami *overfitting*, yaitu kondisi ketika model justru mempelajari data secara berlebihan termasuk *noise* yang tidak diperlukan (Afaq & Rao, 2020).

Dalam penelitian ini, dilakukan pengujian terhadap beberapa variasi nilai *epoch* untuk memperoleh hasil pelatihan yang optimal. Evaluasi dilakukan dengan memperhatikan nilai *training loss* dan *validation loss* pada setiap *epoch*, yang menjadi indikator penting dalam menentukan titik terbaik untuk menghentikan pelatihan model. Melalui pengamatan tersebut, dapat diketahui jumlah *epoch* yang memberikan akurasi pelatihan tertinggi tanpa mengorbankan kemampuan generalisasi model terhadap data uji. Pengujian ini dilakukan dengan menggunakan model *VGG16* dan menggunakan *hyperparameter* yang lain berupa *learning rate* 0.001 dan *optimizer adam*.

Tabel 4. 6 Pengujian *Epoch*

Epoch	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
10	98.22%	98.00%	0.0555	0.0744
15	98.44%	98.00%	0.0477	0.0687
20	98.66%	98.00%	0.0270	0.0686

Dari percobaan pengujian *epoch* berikut rata-rata *Precision*, *Recall* dan *F1-score* dimulai dari 10 *epoch* dengan rata-rata *Precision* 97%, *Recall* 97% dan *F1-score* 97%, 15 *epoch* dengan rata-rata *Precision* 98%, *Recall* 98% dan *F1-score* 98% dan 20 *epoch* dengan rata-rata *Precision* 98%, *Recall* 98% dan *F1-score* 98%.

Dengan hasil pengujian *epoch* ini dapat diambil kesimpulan bahwa jumlah *epoch* dapat mempengaruhi nilai akurasi dan loss dari model yang dilakukan pelatihan dimana nilai akurasi perlahan naik dan nilai loss perlahan-lahan mengalami penurunan. Hal ini sesuai dengan penelitian sebelumnya yang dilakukan oleh Ningsih (2024) dimana dengan penambahan jumlah *epoch* dapat mempengaruhi nilai akurasi dan loss menjadi lebih baik seiring pelatihan model dengan peningkatan jumlah *epoch*.

4.6.2 *Learning Rate*

Learning rate merupakan salah satu *hyperparameter* penting yang mengatur seberapa cepat model dalam memperbarui bobot selama proses pelatihan. Jika nilai *learning rate* terlalu tinggi, model dapat belajar dengan cepat namun berisiko mengalami ketidakstabilan. Sebaliknya, nilai yang terlalu rendah dapat menyebabkan proses pelatihan menjadi lambat dan konvergensi tidak optimal (Peng, 2024). Lalu, bagaimana kita mengetahui nilai *learning rate* yang tepat? Tidak ada satu jawaban pasti, karena hal ini sangat tergantung pada kompleksitas data dan arsitektur model yang digunakan. Oleh karena itu, diperlukan proses uji coba beberapa nilai *learning rate* agar dapat diketahui nilai yang memberikan performa terbaik. Dalam tabel dibawah ini, pengujian dilakukan menggunakan

model *VGG16*, dengan kombinasi *hyperparameter* lain berupa jumlah *epoch* 20 dan *optimizer Adam*.

Tabel 4. 7 *Pengujian Learning Rate*

Learning Rate	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
0.1	97.91%	97.75%	2.4882	4.3341
0.01	97.56%	98.00%	0.2611	0.2839
0.001	98.66%	98.00%	0.0270	0.0686

Dari tabel diatas dengan pengujian menggunakan *learning rate* 0.1 mendapatkan nilai *loss* terendah pada saat *training* yaitu 2.4882, *learning rate* 0.01 memberikan hasil 0.2611 dan *learning rate* 0.001 dengan hasil *training loss* terendah 0.0270. Dapat diambil kesimpulan bahwa nilai dari *learning rate* disini mempengaruhi hasil pengujian baik dari segi akurasi maupun *loss*. Namun, dapat dilihat bahwa nilai dari *training loss* dan *validation loss* disini mengalami penurunan yang sangat signifikan sedangkan nilai akurasi naik perlahan menandakan bahwa jumlah *learning rate* sangat mempengaruhi nilai *loss* dari model yang dilakukan pengujian.

4.6.3 *Optimizer*

Pada penelitian ini, *optimizer* digunakan untuk mempercepat proses pelatihan dengan menyesuaikan bobot jaringan saraf guna meminimalkan *loss function*. Peran *optimizer* sangat penting karena menentukan seberapa cepat dan stabil model mencapai konvergensi.

Tujuan utama penggunaan *optimizer* adalah untuk membantu model belajar secara efisien dan akurat dalam mengklasifikasikan data. Dengan *optimizer*, proses pembaruan bobot dapat diarahkan menuju nilai optimal, sehingga model tidak hanya cepat belajar, tetapi juga menghindari kesalahan umum seperti *overfitting* atau *underfitting*. Pengujian dilakukan dengan menggunakan model *VGG16* dengan

mengkombinasikan beberapa *hyperparameter* yaitu, *learning rate* 0.001 dan jumlah *epoch* 20.

Tabel 4. 8 Pengujian *Optimizer*

Optimizer	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
RMSprop	97.87%	98.00%	0.0646	0.0895
Adam	98.66%	98.00%	0.0270	0.0686

Dengan mengkombinasikan beberapa *hyperparameter* tadi dengan *optimizer RMSprop* dan *optimizer Adam* didapati bahwa *Adam* memberikan hasil yang lebih baik dari segi akurasi maupun *loss* dalam pelatihan model. Dapat disimpulkan juga disini bahwa *optimizer* yang digunakan mempengaruhi hasil dari pelatihan model dan dapat memberikan nilai yang lebih baik jika digunakan dengan kombinasi *hyperparameter* yang tepat.

Berdasarkan hasil pengujian terhadap kombinasi *hyperparameter* yang meliputi jumlah *epoch*, *learning rate*, dan *optimizer*, diperoleh bahwa pemilihan nilai *learning rate* memberikan pengaruh paling signifikan terhadap proses penurunan nilai *loss* selama pelatihan model. *Learning rate* yang terlalu besar cenderung membuat proses pelatihan tidak stabil karena nilai *loss* dapat naik-turun secara drastis. Sebaliknya, *learning rate* yang terlalu kecil memperlambat proses pelatihan karena penurunan *loss* terjadi sangat lambat dan membutuhkan lebih banyak *epoch* untuk mencapai hasil yang optimal.

Selain itu, kombinasi *hyperparameter* yang tepat terbukti penting untuk mendapatkan performa terbaik dari model. Misalnya, pemilihan jumlah *epoch* yang sesuai dapat mencegah terjadinya *underfitting* maupun *overfitting*, sedangkan jenis *optimizer* yang digunakan dapat memengaruhi kecepatan dan efisiensi proses pembelajaran. Berdasarkan pengujian yang telah dilakukan, model menunjukkan performa terbaik ketika menggunakan *learning rate* sebesar 0.001, jumlah *epoch* sebanyak 20, dan *optimizer Adam*, yang ditunjukkan oleh nilai *accuracy* dan *loss*

yang stabil pada data validasi. Kombinasi parameter tersebut memberikan hasil pelatihan yang efektif dan efisien dalam mengklasifikasikan citra daun tomat.

4.7 Pembuatan REST API Klasifikasi Penyakit Daun Tomat

Pada tahap ini penulis membuat rancangan *request* dan *response* yang akan menjelaskan aturan dalam proses permintaan sumber data dan hasil kembalian dari layanan. *Request* prediksi gambar dilakukan untuk meminta hasil probabilitas beberapa kelas terhadap gambar yang dikirimkan. Berikut merupakan rancangan request dan response dalam prediksi gambar:

Tabel 4. 9 Rancangan *Request* dan Respon

Method	POST
URL Scheme	http
URL Host	http://127.0.0.1:6000
Endpoint	/Predict
Headers	-
Body	Form-data
Keterangan	Pada saat pengujian dilakukan melalui metode <i>request</i> dengan <i>body type</i> berupa <i>form-data</i> , data citra disisipkan menggunakan <i>key</i> bernama <i>file</i> . Nilai dari parameter <i>file</i> tersebut merupakan gambar dengan format ekstensi seperti .jpg, .jpeg, .png, atau .jiff. Gambar yang dikirimkan melalui parameter ini kemudian diproses oleh layanan <i>REST API</i> untuk menghasilkan output berupa hasil deteksi dari citra yang diunggah.
Bentuk Respon	JSON
Respon	Prediction = ['Bacterial Spot', 'Healthy', 'Late Blight', 'Yellow Leaf Curl Virus']

REST API yang digunakan dalam penelitian ini dibangun menggunakan *microframework Python*, yaitu *Flask*. Proses *preprocessing* hingga penyajian *output* berupa probabilitas kelas dari hasil klasifikasi dilakukan dengan bantuan modul *TensorFlow*. Untuk tahapan *preprocessing* citra, digunakan modul *tensorflow.keras.preprocessing.image*, sedangkan dalam proses prediksi terhadap citra yang telah diproses, digunakan fungsi *model.predict()* yang merupakan bagian dari kelas *tf.keras.Model*.

```
# Load your trained models
modelvgg16 = load_model("VGG16-0.001AdamEpoch20-98.25.h5")
modelvgg19 = load_model("VGG19-0.001AdamEpoch20-97.50.h5")
modelxception = load_model("Xception-0.001AdamEpoch20-95.99.h5")

class_names = ['Bacterial Spot', 'Healthy', 'Late Blight', 'Yellow Leaf Curl Virus']
```

Gambar 4. 23 *Load Model dan Mendefinisikan Label Tiap Kelas*

Pada gambar 4.23 REST API ini akan menggunakan model CNN yang telah dibuat dan berformat h5. Selain itu, didefinisikan variabel '*class_names*' yang akan digunakan untuk pelabelan setiap kelas.

```
# Preprocess input image
img = load_img(predict_image_path, target_size=(224, 224))
x = img_to_array(img)
x = np.expand_dims(x, axis=0)
x = x / 255.0
```

Gambar 4. 24 *Preprocessing Input Image*

Pada gambar 4.24, dibuat sebuah proses sederhana sebelum diklasifikasikan oleh model. Gambar yang telah diunggah akan diubah ukurannya menjadi 224x224 piksel, dikonversi menjadi *array* numerik, ditambahkan dimensi *batch*, lalu dinormalisasi ke skala 0–1 agar sesuai dengan format *input* saat *training model*.

```
UPLOAD_FOLDER = 'static/uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

Gambar 4. 25 *Upload Image*

Pada tahap ini disiapkan sebuah folder yang digunakan untuk menampung citra yang akan dilakukan klasifikasi dengan ketentuan ekstensi yang bisa diinputkan yaitu png, jpg, dan jpeg. Setelah itu maka citra akan siap saat ingin digunakan dalam proses klasifikasi.

```
# Predict using normalized input
pred_vgg16 = modelvgg16.predict(x)
pred_vgg19 = modelvgg19.predict(x)
pred_xception = modelxception.predict(x)

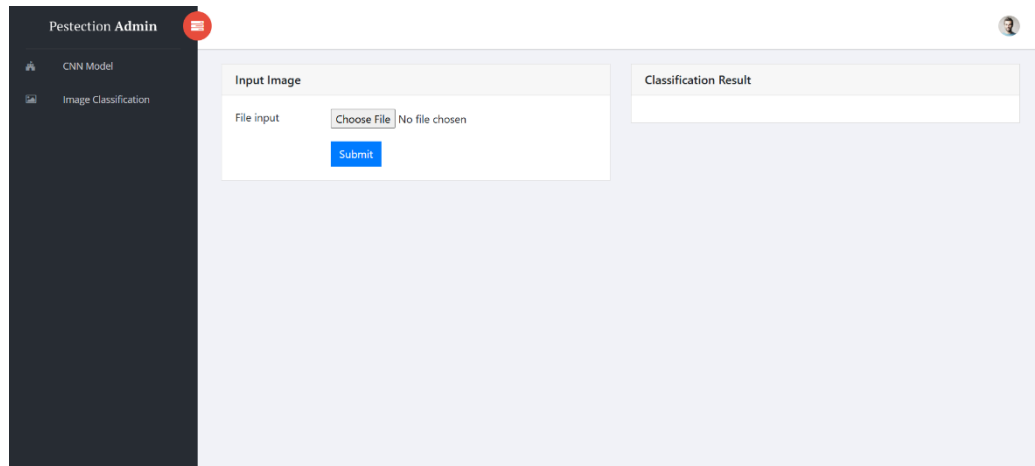
def get_top_prediction(pred_array):
    idx = np.argmax(pred_array)
    return class_names[idx], '{:.2f}%'.format(100 * pred_array[0][idx])
```

Gambar 4. 26 *Predict Image*

Pada gambar 4.26 ini merupakan tahap yang berfungsi untuk memprediksi citra yang telah dipersiapkan sebelumnya. Disini citra akan diprediksi menggunakan model CNN yang telah didefinisikan sebelumnya. Hasil prediksi tersebut nantinya akan memperlihatkan nilai probabilitas tertinggi sekaligus label kelas dari setiap nilai probabilitas.

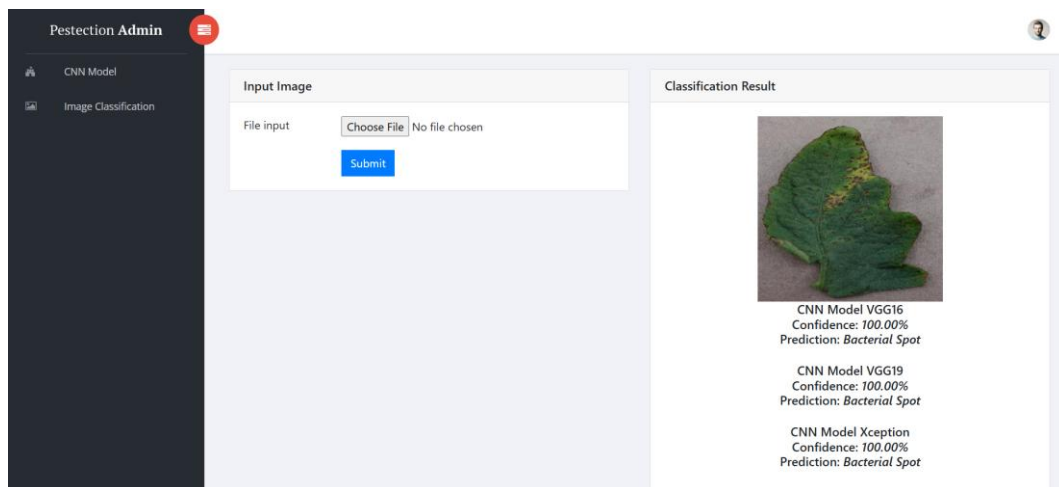
4.8 Hasil Klasifikasi Citra dan *Testing* Menggunakan *Website*

Pada tahap ini dilakukan proses pengujian klasifikasi citra melalui aplikasi berbasis web yang telah diintegrasikan dengan model CNN yang telah dilatih sebelumnya. Aplikasi web ini dibangun menggunakan *microframework* Python, yaitu *Flask*, dan dijalankan secara lokal melalui alamat URL <http://127.0.0.1:5000/>. Tampilan awal dari aplikasi tersebut dapat dilihat pada Gambar berikut, yang menunjukkan antarmuka utama yang digunakan dalam proses klasifikasi citra oleh pengguna.



Gambar 4. 27 Halaman Utama

Pada halaman utama, terdapat tombol untuk melakukan *input* citra dengan cara memilih citra dari penyimpanan lokal yang nantinya akan dilakukan proses klasifikasi jenis penyakit daun tanaman tomat.



Gambar 4. 28 Halaman Hasil Klasifikasi

Setelah pemilihan citra yang akan di-inputkan dan menekan tombol *submit* maka sistem akan mengirimkan *file* tersebut ke model untuk diproses dalam klasifikasi jenis penyakit pada daun tanaman tomat. Setelah citra berhasil diproses oleh Model, maka akan dihasilkan *output*. *Output* yang dihasilkan merupakan kelas dengan probabilitas tertinggi dan nilai kepercayaan dari masing-masing model terhadap hasil klasifikasi tersebut.

Selanjutnya dilakukan proses pengujian menggunakan data baru (diluar dari dataset) dengan scenario data sebanyak 52 citra dengan masing-masing kelas berjumlah 13 citra daun tanaman tomat.

Berikut adalah hasil pengujian model *VGG16*:

Tabel 4. 10 *Testing Model VGG16 Via Website*

No	Class	Total Image	TP	TN	FP	FN
1	bacterial_spot	13	12	39	1	0
2	healthy	13	13	39	0	0
3	late_blight	13	13	39	0	0
4	yellow_leaf_curl_virus	13	13	39	0	0
	TOTAL	52	51	156	1	0

Dari hasil confusion matriks tersebut didapatkan hasil perhitungan akurasi precision, recall dan f1-score setiap kelas sebagai berikut:

Tabel 4. 11 *Classification Report Testing VGG16*

No	Class	Precision	Recall	F1 Score	Support
1	bacterial_spot	0.923	1.00	0.96	13
2	healthy	1.00	1.00	1.00	13
3	late_blight	1.00	1.00	1.00	13
4	yellow_leaf_curl_virus	1.00	1.00	1.00	13
		0.98	1.00	0.99	52

$$Accuracy = \left(\frac{TP+TN}{TP+TN+FP+FN} \right) = \frac{51+156}{51+156+1+0} = \frac{207}{208} = 0.9952$$

Berdasarkan tabel 4.11 didapatkan hasil klasifikasi model yang sangat baik pada saat dilakukan pengujian menggunakan sebanyak 52 data uji. Akurasi yang diperoleh yaitu sebesar 99.52%.

Berikut adalah hasil pengujian model *VGG19*:

Tabel 4. 12 *Testing Model VGG19 Via Website*

No	Class	Total Image	TP	TN	FP	FN
1	bacterial_spot	13	11	39	2	0
2	healthy	13	12	39	1	0
3	late_blight	13	12	39	1	0
4	yellow_leaf_curl_virus	13	13	39	0	0
	TOTAL	52	48	156	4	0

Dari hasil confusion matriks tersebut didapatkan hasil perhitungan akurasi precision, recall dan f1-score setiap kelas sebagai berikut:

Tabel 4. 13 *Classification Report Testing VGG19*

No	Class	Precision	Recall	F1 Score	Support
1	bacterial_spot	0.846	1.00	0.916	13
2	healthy	0.923	1.00	0.96	13
3	late_blight	0.923	1.00	0.96	13
4	yellow_leaf_curl_virus	1.00	1.00	1.00	13
		0.923	1.00	0.96	52

$$Accuracy = \left(\frac{TP+TN}{TP+TN+FP+FN} \right) = \frac{48+156}{48+156+4+0} = \frac{204}{208} = 0.9808$$

Berdasarkan tabel 4.13 didapatkan hasil klasifikasi model yang sangat baik pada saat dilakukan pengujian menggunakan sebanyak 52 data uji. Akurasi yang diperoleh yaitu sebesar 98.08%.

Berikut adalah hasil pengujian model VGG19:

Tabel 4. 14 *Testing Model Xception Via Website*

No	Class	Total Image	TP	TN	FP	FN
1	bacterial_spot	13	12	39	1	0
2	healthy	13	11	39	2	0
3	late_blight	13	10	39	3	0
4	yellow_leaf_curl_virus	13	13	39	0	0
	TOTAL	52	46	156	6	0

Dari hasil confusion matriks tersebut didapatkan hasil perhitungan akurasi precision, recall dan f1-score setiap kelas sebagai berikut:

Tabel 4. 15 *Classification Report Testing Xception*

No	Class	Precision	Recall	F1 Score	Support
1	bacterial_spot	0.923	1.00	0.9600	13
2	healthy	0.846	1.00	0.9167	13
3	late_blight	0.769	1.00	0.8696	13
4	yellow_leaf_curl_virus	1.00	1.00	1.00	13
		0.884	1.00	0.921	52

$$Accuracy = \left(\frac{TP+TN}{TP+TN+FP+FN} \right) = \frac{46+156}{46+156+6+0} = \frac{202}{208} = 0.9712$$

Berdasarkan tabel 4.15 didapatkan hasil klasifikasi model yang sangat baik pada saat dilakukan pengujian menggunakan sebanyak 52 data uji. Akurasi yang diperoleh yaitu sebesar 97.12%.

Berdasarkan hasil pengujian ketiga model arsitektur *CNN* pada aplikasi berbasis web yang dikembangkan menggunakan framework *Flask*, diperoleh bahwa model *VGG16* memberikan performa klasifikasi terbaik dibandingkan

dengan *VGG19* dan *Xception*. Pengujian dilakukan menggunakan data uji yang sama melalui antarmuka web, di mana masing-masing model telah diintegrasikan ke dalam sistem. Model *VGG16* berhasil mencapai akurasi tertinggi sebesar 99.52%, diikuti oleh *VGG19* dengan akurasi 98.08%, dan *Xception* dengan akurasi 97.12%. Hasil ini menunjukkan bahwa *VGG16* lebih stabil dan konsisten dalam mengenali pola pada citra daun tomat dibandingkan dua model lainnya. Perbandingan nilai akurasi tersebut menunjukkan bahwa pemilihan arsitektur model sangat berpengaruh terhadap performa sistem klasifikasi berbasis *web* yang telah dibangun.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan rangkaian eksperimen dan analisis yang telah dilakukan, penelitian ini memberikan sejumlah temuan penting terkait penerapan model *Convolutional Neural Network (CNN)* dalam klasifikasi penyakit pada daun tanaman tomat. Adapun kesimpulan dari penelitian ini adalah sebagai berikut:

1. Pemanfaatan arsitektur *CNN* seperti *VGG16*, *VGG19*, dan *Xception* terbukti mampu melakukan klasifikasi penyakit daun tomat dengan tingkat akurasi yang tinggi. Model-model ini dilatih menggunakan 4.000 citra daun tanaman tomat dari *dataset Kaggle*, yang mencakup empat kelas daun: *bacterial spot*, *late blight*, *yellow leaf curl virus*, dan daun sehat.
2. Kombinasi *hyperparameter* yang digunakan dalam proses pelatihan memiliki pengaruh besar terhadap performa model. Penyesuaian pada jumlah *epoch*, *learning rate*, dan jenis *optimizer* secara langsung memengaruhi hasil pelatihan, khususnya *hyperparameter learning rate* yang memberikan penurunan pada nilai *training loss* dan *validation loss* secara signifikan. Hal ini menunjukkan pentingnya pemilihan *hyperparameter* yang tepat untuk menghindari *underfitting* maupun *overfitting*.
3. Model *VGG16* menunjukkan performa terbaik di antara ketiga arsitektur yang dilatih. Dengan kombinasi *epoch* sebanyak 20, *learning rate* sebesar 0.001, dan optimizer *Adam*, model ini berhasil memperoleh nilai *training loss* sebesar 0.0270 dan *validation loss* sebesar 0.686, yang merupakan hasil terbaik dibandingkan *VGG19* dan *Xception*.
4. Pengujian dilakukan dalam sebuah aplikasi *web* berbasis *Flask* terhadap 52 citra uji (13 citra per kelas) dan menghasilkan tingkat akurasi tertinggi pada model *VGG16* sebesar 99,52%. Sementara itu, model *VGG19* memperoleh akurasi 98,05%, dan *Xception* memperoleh 97,12%.

5. Secara keseluruhan, *VGG16* dinilai sebagai model paling efektif dan konsisten dalam mengklasifikasikan penyakit pada daun tanaman tomat. Hasil ini menunjukkan potensi besar *VGG16* untuk diterapkan lebih lanjut dalam sistem pendeteksi penyakit tanaman berbasis citra digital, khususnya dalam bidang pertanian.

5.2 Saran

Berdasarkan dari Kesimpulan diatas ada beberapa saran untuk peneliti selanjutnya yaitu:

1. Menambahkan *dataset* dengan lebih banyak citra yang bervariasi untuk setiap kelas penyakit untuk meningkatkan representasi dan performa model.
2. Menambahkan kelas penyakit daun tanaman tomat lainnya untuk melatih model agar mengenali lebih banyak jenis penyakit pada daun tanaman tomat.
3. Menggunakan *pretrained model* lainnya yang tersedia dalam *keras* untuk melakukan klasifikasi seperti, *DenseNet*, *NasNetMobile*, *ResNet50*, dan lain sebagainya untuk mengetahui keakuratan dari model-model yang ada dalam menentukan penyakit khususnya pada daun tanaman tomat.
4. Peneliti selanjutnya bisa untuk mengoptimalkan konfigurasi *hyperparameter* yang digunakan untuk dalam pelatihan model guna meningkatkan nilai akurasi.
5. Model yang sudah teruji kedepannya diharapkan dapat diimplementasikan untuk klasifikasi penyakit secara *realtime*.

DAFTAR PUSTAKA

- Akram, M. N., Yaseen, M. U., Waqar, M., Imran, M., & Hussain, A. (2023). A Double-Branch Xception Architecture for Acute Hemorrhage Detection and Subtype Classification. *Computers, Materials & Continua*, 76(3).
- Alayrac, J. B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., ... & Simonyan, K. (2022). Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35, 23716-23736.
- Alwanda, M. R., Ramadhan, R. P. K., & Alamsyah, D. (2020). Implementasi metode convolutional neural network menggunakan arsitektur LeNet-5 untuk pengenalan doodle. *Jurnal Algoritme*, 1(1), 45-56.
- Bastari, A. J., & Cherid, A. (2023). Klasifikasi Penyakit Tanaman Tomat Menggunakan Convolutional Neural Network Dan Implementasi Model H5 Pada Aplikasi Desktop. *Jurnal Sistem Informasi dan Sistem Komputer*, 8(2), 199-207.
- Bouke, M. A., Zaid, S. A., & Abdullah, A. (2024). Implications of data leakage in machine learning preprocessing: a multi-domain investigation.
- Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).
- D'Agostino, M., Marti, M., Otero, P., Doane, D., Brooks, I., Saiso, S. G., ... & de Cosio, G. (2021). Toward a holistic definition for Information Systems for Health in the age of digital interdependence. *Revista Panamericana de Salud Pública*, 45, e143.
- Dahmane, O., Khelifi, M., Beladgham, M., & Kadri, I. (2021). Pneumonia detection based on transfer learning and a combination of VGG19 and a CNN built from scratch. *Indonesian Journal of Electrical Engineering and Computer Science*, 24(3), 1469-1480.
- DARMATASIA, D. (2020). Deteksi Penggunaan Masker Menggunakan Xception Transfer Learning. *Jurnal INSTEK (Informatika Sains dan Teknologi)*, 5(2), 279-288.

- Darmawan, A., Rositasari, A., & Muhimmah, I. (2020, April). The Identification System of Acne Type on Indonesian People's Face Image. In *IOP Conference Series: Materials Science and Engineering* (Vol. 803, No. 1, p. 012028). IOP Publishing.
- Enkvetchakul, P., & Surinta, O. (2022). Effective data augmentation and training techniques for improving deep learning in plant leaf disease recognition. *Applied Science and Engineering Progress*, 15(3), 3810-3810.
- Husodo, K., Lubis, C., & Rusdi, Z. (2023). KLASIFIKASI TANAMAN ANGGREK MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK DENGAN ARSITEKTUR VGG-19. *Simtek: jurnal sistem informasi dan teknik komputer*, 8(2), 253-258.
- Kusumawati, W. I., & Noorizki, A. Z. (2023). Perbandingan Performa Algoritma VGG16 dan VGG19 Melalui Metode CNN untuk Klasifikasi Varietas Beras. *Journal of Computer, Electronic, and Telecommunication*, 4(2).
- L. Li, S. Zhang and B. Wang, "Plant Disease Detection and Classification by Deep Learning—A Review," in *IEEE Access*, vol. 9, pp. 56683-56698, 2021, doi: 10.1109/ACCESS.2021.3069646.
- Liao, Y. H., Kar, A., & Fidler, S. (2021). Towards good practices for efficiently annotating large-scale image classification datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4350-4359).
- Mare, B. S. (2022). Perancangan Sistem Informasi Berbasis Web Pada Koperasi Simpan Pinjam Sejahtera Bersama. *Indonesian Journal of Networking and Security (IJNS)*, 11(2).
- Nguyen, T. H., Nguyen, T. N., & Ngo, B. V. (2022). A VGG-19 model with transfer learning and image segmentation for classification of tomato leaf disease. *AgriEngineering*, 4(4), 871-887.
- Ningsih, N. P., Suryadi, E., Bakti, L. D., & Imran, B. (2022). Klasifikasi Penyakit Early Blight Dan Late Blight Pada Tanaman Tomat Berdasarkan Citra Daun Menggunakan Metode Cnn Berbasis Website. *Jurnal Kecerdasan Buatan dan Teknologi Informasi*, 1(3), 27-35.

- Nurdin, A., Kartika, D. S. Y., & Najaf, A. R. E. (2024). Klasifikasi Penyakit Daun Tomat Dengan Metode Convolutional Neural Network Menggunakan Arsitektur Inception-V3. *JURNAL ILMIAH INFORMATIKA*, 12(02), 114-119.
- Oluwafisayomi, K. F., & Andrew, J. (2022). Fully connected layer vs dense layer in image processing.
- Panjaitan, S., Sitepu, C., & SINAGA, J. (2023). Deteksi jerawat menggunakan arsitektur yolov3. *JURNAL EKONOMI, SOSIAL & HUMANIORA*, 4(06), 1-6.
- Permatasari, D. I. (2024). IMPLEMENTASI METODE CONVOLUTIONAL NEURAL NETWORK (CNN) UNTUK KLASIFIKASI TANAMAN HERBAL BERDASARKAN CITRA DAUN. *Kohesi: Jurnal Sains dan Teknologi*, 3(9), 1-10.
- Prasetyo, S. Y. (2024). Overcoming Overfitting in CNN Models for Potato Disease Classification Using Data Augmentation. *Engineering, MAThematics and Computer Science Journal (EMACS)*, 6(3), 179-184.
- Putra, J. V. P., Ayu, F., & Julianto, B. (2023, January). Implementasi Pendeteksi Penyakit pada Daun Alpukat Menggunakan Metode CNN. In *Seminar Nasional Teknologi & Sains* (Vol. 2, No. 1, pp. 155-162).
- Radikto, R., Mulyana, D. I., Rofik, M. A., & Zakaria, M. O. Z. (2022). Klasifikasi Kendaraan pada Jalan Raya menggunakan Algoritma Convolutional Neural Network (CNN). *Jurnal Pendidikan Tambusai*, 6(1), 1668-1679.
- Rahman, S., Sembiring, A., Siregar, D., Prahmana, I. G., Puspadini, R., & Zen, M. (2023). Python: Dasar dan Pemrograman Berorientasi Objek. *Penerbit Tahta Media*.
- Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, 11(4), 193.
- Rohma, A. M. N. (2024). Diagnosa Penyakit Tanaman Tomat pada Citra Daun Menggunakan Metode Convolutional Neural Network (CNN). *JIMU: Jurnal Ilmiah Multidisipliner*, 2(03), 555-567.

- Saleem, M. H., Potgieter, J., & Arif, K. M. (2020). Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers. *Plants*, 9(10), 1319.
- Saputro, A., Mu'min, S., Lutfi, M., & Putri, H. (2022). DEEP TRANSFER LEARNING DENGAN MODEL ARSITEKTUR VGG16 UNTUK KLASIFIKASI JENIS VARIETAS TANAMAN LENGKENG BERDASARKAN CITRA DAUN. *JATI (Jurnal Mahasiswa Teknik Informatika)*, 6(2), 609-614.
- Sathyanarayanan, S., & Tantri, B. R. (2024). Confusion matrix-based performance evaluation metrics. *African Journal of Biomedical Research*, 4023-4031.
- Seliya, N., Abdollah Zadeh, A., & Khoshgoftaar, T. M. (2021). A literature review on one-class classification and its potential applications in big data. *Journal of Big Data*, 8, 1-31.
- Sial, A. H., Rashdi, S. Y. S., & Khan, A. H. (2021). Comparative analysis of data visualization libraries Matplotlib and Seaborn in Python. *International Journal*, 10(1), 277-281.
- Soufitri, F. (2023). *Konsep sistem informasi*. PT Inovasi Pratama Internasional.
- Taherdoost, H. (2022). What are different research approaches? Comprehensive review of qualitative, quantitative, and mixed method research, their applications, types, and limitations. *Journal of Management Science & Engineering Research*, 5(1), 53-63.
- Xu, J., Liu, Z., Wang, S., Zheng, T., Wang, Y., Wang, Y., & Dang, Y. (2023). Foundations and applications of information systems dynamics. *Engineering*, 27, 254-265.
- Yoraeni, A., Handayani, P., Rakhmah, S. N., Siregar, J., Al Afghani, D. Y., Rianto, H., ... & Nurrohman, A. (2023). *Sistem Informasi Manajemen*. PT. Scifintech Andrew Wijaya.