BAB I

PENDAHULUAN

A. Latar Belakang

Tanaman cabai dan tomat merupakan tanaman hortikultura yang sangat penting dalam industri pertanian dan pangan. Namun, penyakit tanaman seperti penyakit layu, karat, atau infeksi jamur sering kali menjadi masalah serius yang dapat menghancurkan hasil panen dan mengurangi produktivitas petani. Untuk mengatasi masalah ini, deteksi dini penyakit tanaman menjadi kunci dalam pengendalian dan pengelolaan penyakit.

Dalam beberapa tahun terakhir, perkembangan teknologi *Artificial Intelligence* (AI) dan khususnya *computer vision* telah memberikan peluang baru untuk mendeteksi penyakit pada tanaman secara cepat dan akurat. Metode *Convolutional Neural Network* (CNN) adalah salah satu teknik yang paling sukses dalam bidang *computer vision*. CNN mampu mempelajari pola dan fitur penting dalam citra secara otomatis, sehingga memungkinkan untuk mengidentifikasi dan mengklasifikasikan penyakit pada tanaman dengan tingkat akurasi yang tinggi.

Meskipun CNN telah banyak digunakan dalam deteksi objek dan pengenalan citra, implementasinya dalam mendeteksi penyakit pada tanaman cabai dan tomat masih terbatas. Penelitian-penelitian terdahulu telah menunjukkan potensi besar metode ini, namun masih banyak aspek yang perlu diteliti lebih lanjut untuk meningkatkan akurasi, keandalan, dan aplikabilitasnya dalam konteks yang lebih luas. Dalam konteks ini, penelitian bertujuan untuk mengimplementasikan

computer vision dengan menggunakan metode CNN dalam mendeteksi penyakit pada tanaman cabai dan tomat.Penelitian ini akan berfokus pada penggunaan model CNN yang mampu mengklasifikasikan citra tanaman yang sehat dan terinfeksi penyakit dengan akurasi yang tinggi. Dengan demikian, hasil penelitian ini diharapkan dapat memberikan kontribusi penting dalam pengendalian dan pengelolaan penyakit tanaman.

Melalui implementasi *computer vision* dan metode CNN dalam mendeteksi penyakit pada tanaman cabai dan tomat, diharapkan solusi yang cepat, akurat, dan efisien dalam mendeteksi penyakit tanaman. Dengan demikian, petani maupun praktisi pertanian atau masyarakat awam akan mendapatkan manfaat kemudahan dalam menghadapi tantangan penyakit pada tanaman, termasuk pengambilan langkah pencegahan dan tindakan yang tepat secara lebih dini.

B. Rumusan Masalah

Berdasarkan latar belakang diatas maka rumusan masalah yang dapat dirumuskan adalah bagaimana cara membangun sebuah sistem yang dapat memanfaatkan teknologi *computer vision* untuk melakukan deteksi penyakit pada tanaman khususnya cabai dan tomat, dan seberapa akurat hasil prediksi dari sistem yang dibuat nantinya dalam deteksi penyakit pada tanaman?

C. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk membuat sistem yang mampu mendeteksi dan mengklasifikasikan penyakit dengan memanfaatkan Arsitektur *Convolutional Neural Network.*

D. Batasan Masalah

Untuk dapat menghasilkan penelitian yang sejalan dengan permasalahan dan tujuan penelitian maka diperlukannya batasan penelitian sebagai berikut:

1. Penelitian ini akan difokuskan pada implementasi teknologi *Computer Vision* dalam mendeteksi penyakit pada tanaman cabai dan tomat.
2. Penelitian ini akan menggunakan Arsitektur *Convolutional Neural Network* (CNN) sebagai algoritma utama untuk pengenalan dan klasifikasi penyakit.
3. Penelitian ini akan membatasi penyakit yang akan dideteksi pada tanaman cabai dan tomat yang dapat terlihat secara kasatmata seperti penyakit *Early Blight*, *late blight*, infeksi bakteri dan sebagainya.

E. Manfaat Penelitian

Adapun manfaat yang penulis harapkan dari ditulis dan dibuatnya penelitian sistem ini yaitu:

1. **Manfaat bagi penulis**

   Penulis dapat mengimplementasikan ilmu yang telah diperoleh selama duduk di bangku perkuliahan khususnya mengimplementasikan kemampuan dalam Bahasa pemrograman, serta dapat memenuhi persyaratan kelulusan bagi penulis menyelesaikan jenjang pendidikan S1.

2. **Manfaat bagi pembaca**

   Penelitian ini diharapkan dapat menjadi referensi bagi para pembaca yang juga terinspirasi melakukan penelitian sejenis dan memberikan pengetahuan lebih bagi para pembaca tentang sistem yang dibuat.

BAB II

TINJAUAN PUSTAKA

A. Penelitian Terdahulu

1. Rasywir, E., Sinaga, R., & Pratama, Y. (2020) “Analisis dan Implementasi Diagnosis Penyakit Sawit dengan Metode *Convolutional Neural Network* (CNN)” Tujuan penelitian adalah untuk memberikan solusi dalam mempercepat proses identifikasi jenis hama dan penyakit pada kelapa sawit tanpa memerlukan tenaga ahli secara langsung. Data citra penyakit kelapa sawit dari Dinas Perkebunan Provinsi Jambi digunakan untuk melatih model CNN. Setelah proses pelatihan, model digunakan untuk menguji diagnosis penyakit kelapa sawit dengan hasil akurasi tertinggi 0,89 dan terendah 0,83, serta rata-rata akurasi 0,87. Hasil ini menunjukkan bahwa klasifikasi citra kelapa sawit dengan CNN sudah cukup baik, dan dapat menjadi dasar untuk pengembangan sistem klasifikasi penyakit kelapa sawit yang otomatis dan *mobile* untuk membantu petani.
2. Sya'bani, D. R., Hamzah, A., & Susanti, E. (2022). “Klasifikasi Buah Segar Dan Busuk Menggunakan Algoritma *Convolutional Neural Network* Dengan Tflite Sebagai Media Penerapan Model *Machine Learning*” penelitian bertujuan untuk membuat model menggunakan Algoritma CNN (*Convolutional Neural Network*) yang dapat mengklasifikasi buah segar dan busuk (apel merah, pisang, dan jeruk mandarin) serta menerapkan model tersebut menggunakan TFLite pada Smartphone Android.

3. Lesmana, A. M., Fadhillah, P. R., & Rozikin, C. (2022) “Identifikasi Penyakit pada Citra Daun Kentang Menggunakan *Convolutional Neural Network* (CNN)”.Fokus penelitian ini adalah menggunakan algoritma deep learning *Convolutional Neural Network* (CNN) untuk mengidentifikasi penyakit pada daun kentang, seperti *Early Blight* dan *late blight*. Tujuan penelitian adalah untuk mempercepat proses identifikasi penyakit pada daun kentang yang dapat menyebabkan kerugian produksi. Dengan memanfaatkan 5400 citra yang terbagi menjadi 3 kelas (sehat, *Early Blight*, *late blight*), hasil pengujian menunjukkan akurasi tertinggi pada data validation mencapai 99%, menunjukkan bahwa CNN efektif dalam mengidentifikasi penyakit pada citra daun kentang.

B. Penyakit Tanaman Cabai dan Tomat

Tanaman cabai, dan tomat memiliki nilai ekonomis yang sangat penting di Indonesia, terutama cabai yang merupakan bagian tak terpisahkan dari masakan pedas yang populer di masyarakat (Wati, et al., 2021).

Tanaman ini telah tersebar luas dalam budidaya di Indonesia. Dalam pertanian, tantangan utama yang dihadapi adalah serangan organisme pengganggu seperti penyakit. Penyakit-penyakit yang umum menyerang tanaman *Solanaceae* seperti cabai dan tomat disebabkan oleh berbagai patogen seperti bakteri, cendawan, virus, dan nematode (Wati, et al., 2021). Oleh karena itu, penting untuk memahami dan mengatasi penyakit-penyakit ini dalam upaya menjaga produktivitas tanaman.

1. Penyakit Pada Tanaman Cabai

Penyakit *antraknosa* pada cabai disebabkan oleh infeksi cendawan *Colletotrichum sp.* dan bakteri patogen *Erwinia carotovora ssp. Carotovora.* Gejala antraknosa pada buah cabai umumnya meliputi bercak yang terlihat agak mengkilap, sedikit berair, dengan adanya *aservulus* (struktur reproduksi jamur) dan mikro *sklerotia* (struktur dorman jamur) berwarna hitam, oren, atau coklat. Gejala ini dapat bervariasi tergantung pada jenis *Colletotrichum* yang menyebabkan infeksi serta kondisi kelembaban lingkungan. Perkembangan penyakit antraknosa akan ditandai dengan semakin meluasnya bercak pada buah cabai. Secara bertahap, buah yang terinfeksi akan berubah warna menjadi coklat kehitaman dan mengalami pembusukan. Serangan yang parah dapat menyebabkan buah yang terinfeksi mengering dan keriput, sehingga mengurangi nilai jual dan kualitasnya.

Gambar 2. 1 Penyakit Antraknosa Pada Cabai.

Sumber : Infarm, 2024 *(www.infarm.co.id)*

Penyakit kuning pada tanaman cabai disebabkan oleh virus *gemini*, khususnya *Pepper yellow leaf curl Indonesian virus* (PepYLCIV). Gejala

khasnya meliputi daun yang berwarna kuning cerah di bagian atas, mengerut, melengkung ke atas, dan mengecil, sementara bagian bawah tetap hijau. Selain itu, tulang daun akan menebal dan daun akan menggulung ke atas, menyebabkan tanaman menjadi kerdil dan tidak berbuah pada tahap infeksi dini. Untuk mencegah infeksi virus pada tanaman cabai, penting untuk meningkatkan ketahanan tanaman terhadap serangan virus. Salah satu cara yang efektif adalah dengan memastikan tanaman cabai mendapatkan pemupukan yang cukup untuk menjaga kesehatannya. Dengan memberikan nutrisi yang optimal, tanaman cabai akan menjadi lebih kuat dan mampu melawan serangan patogen, termasuk virus.

Gambar 2. 2 Pengamatan OPT Tanaman Cabai.
Sumber : Dinas Pertanian, 2023 (distan.bulelengkab.go.id)

2. Penyakit Pada Tanaman Tomat

Penyakit busuk daun pada tanaman tomat, yang dikenal sebagai "*Late Blight*" disebabkan oleh infeksi *Phytophthora infestans*. Gejala penyakit ini tidak hanya terlihat pada daun, tetapi juga dapat memengaruhi bagian lain dari tanaman seperti batang dan buah. Ciri khas dari penyakit ini terlihat

pada daun, di mana daun menunjukkan nekrosis berwarna kehitaman yang sering dimulai dari ujung daun. *Nekrosis* ini kemudian akan menyebar ke batang dan seluruh bagian tanaman. Pada buah yang masih muda berwarna hijau, gejala terlihat saat berubah menjadi coklat tanpa membuat tekstur buah menjadi lunak. Selain itu, gejala pada daun juga termasuk *klorosis* dan *nekrosis*, kadang-kadang dengan miselia cendawan terlihat di sekitar bagian nekrotik terutama pada bagian bawah daun.*Oospora* dari *Phytophthora* infestans pada daun, buah, dan batang akan tersebar melalui angin dan percikan air hujan. Penyebaran juga dapat terjadi melalui benih tanaman dan kulit biji. Kelembaban tinggi bersama dengan banyaknya percikan air hujan akan memperburuk kondisi penyakit ini. Pengendalian penyakit ini melibatkan penggunaan benih yang bebas patogen dan disarankan untuk melakukan perlakuan terhadap benih sebelum penanaman. Selain itu, rotasi tanaman dan pengaturan jarak tanam juga merupakan langkah pengendalian yang efektif.

Gambar 2. 3 *Late Blight* pada Daun Tomat.

Sumber : Plantix (plantix.net)

Penyakit *Early Blight* pada tanaman tomat disebabkan oleh infeksi cendawan *Alternaria solani* yang biasanya menyerang daun tua tanaman. Gejala awalnya berupa lesi berwarna coklat yang kemudian berkembang menjadi cincin konsentris saat penyakit semakin parah. Selain pada daun, gejala juga dapat terlihat pada batang tanaman dan buah dengan adanya lesi berbentuk cincin konsentris yang besar, bahkan hampir menutupi seluruh permukaan buah.

Gambar 2. 4 *Early Blight* pada tomat.
Sumber : Cornell Cals (blogs.cornell.edu)

Untuk mengendalikan penyakit ini, disarankan untuk menggunakan bibit tomat yang memiliki ketahanan terhadap infeksi patogen *Alternaria solani*. Selain itu, praktik rotasi tanaman dengan tanaman lain, pemberian pupuk kalium secara bijak, dan pemanfaatan plastik mulsa untuk mengontrol tingkat kelembaban tanah juga merupakan langkah-langkah yang efektif dalam pengendalian penyakit *Early Blight*. Dengan menerapkan strategi pengendalian yang komprehensif, diharapkan dapat mengurangi dampak

penyakit ini pada tanaman tomat dan meningkatkan produktivitas pertanian secara keseluruhan.

Penyakit-penyakit yang menyerang tanaman cabai dan tomat tersebut adalah contoh nyata dari masalah yang sering muncul dalam pertanian. Oleh karena itu, pemahaman mendalam tentang penyakit-penyakit ini sangat penting untuk membantu dalam penanganan yang lebih efektif dan strategis.

C. Computer Vision

"*Computer vision* adalah bidang yang dapat membuat mesin 'melihat'. Teknologi ini menggunakan kamera dan komputer sebagai pengganti mata manusia untuk mengidentifikasi, melacak, dan mengukur target untuk pengolahan gambar lebih lanjut." (Tian, Wang, Liu, Qiao, & Li, 2020)

Salah satu aplikasi utama dari teknologi *computer vision* dalam pertanian adalah pemantauan pertumbuhan tanaman. Dengan menggunakan citra visual yang diambil dari drone atau kamera tanah, petani dapat memantau kondisi tanaman secara real-time, mendeteksi masalah pertumbuhan seperti kekurangan nutrisi atau serangan hama, dan mengambil tindakan yang diperlukan untuk meningkatkan hasil panen. Selain itu, teknologi ini juga dapat digunakan untuk mendeteksi penyakit tanaman secara dini, sehingga memungkinkan pengendalian yang lebih efektif.

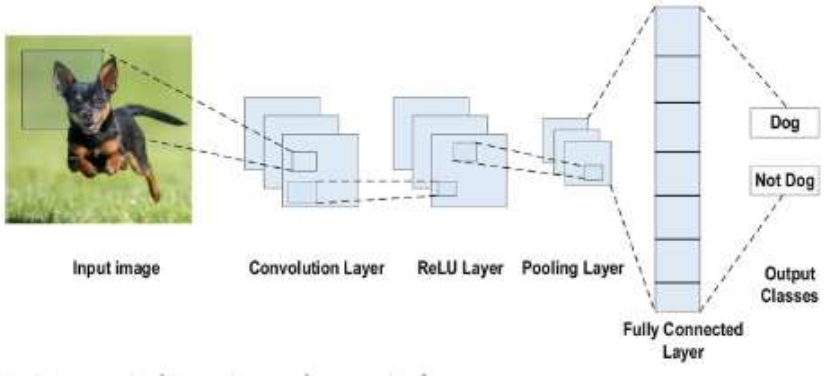
Selain itu, teknologi *computer vision* juga dapat diterapkan dalam proses panen otomatis. Dengan menggunakan sistem penglihatan komputer yang terintegrasi dengan peralatan pertanian seperti traktor atau robot, petani dapat

mengotomatisasi proses panen tanpa perlu campur tangan manusia secara langsung. Hal ini tidak hanya meningkatkan efisiensi dalam proses panen, tetapi juga mengurangi biaya dan waktu yang diperlukan untuk menyelesaikan tugas tersebut. Selain itu, teknologi *computer vision* juga dapat digunakan untuk mengelola kualitas produk pertanian. Dengan analisis gambar dan pengolahan data visual,

petani dapat melakukan inspeksi kualitas secara otomatis terhadap hasil panen mereka, memastikan bahwa hanya produk berkualitas tinggi yang dipasarkan ke konsumen. Hal ini tidak hanya meningkatkan reputasi petani dalam pasar, tetapi juga meningkatkan kepercayaan konsumen terhadap produk pertanian yang dihasilkan.

D. Arsitektur *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan jenis arsitektur jaringan saraf tiruan atau turunan dari ilmu *neural network*. CNN telah merevolusi bidang ML dengan kemampuannya untuk secara otomatis mengekstraksi fitur-fitur yang relevan dari citra data. komponen utama CNN, termasuk lapisan-lapisan yang ada di dalamnya. (Alzubaidi, et al., 2021)



Gambar 2. 5 Salah satu contoh Arsitektur CNN. [Alzubaidi, 2021]

CNN dapat terdiri dari beberapa lapisan yang saling berhubungan. Setiap lapisan memiliki peran dan fungsi tertentu dalam proses pengenalan pola. Lapisan-lapisan utama dalam CNN:

1. Lapisan Konvolusi (*Convolutional Layer*): *Convolutional layer* digunakan untuk mengekstraksi fitur dari gambar dengan menggunakan operasi konvolusi, yang menggantikan operasi perkalian matriks yang umumnya digunakan dalam jaringan saraf tiruan tradisional. Operasi konvolusi memungkinkan jaringan untuk mempelajari pemetaan antara lapisan input dan *output* dengan lebih efisien, sehingga memungkinkan jaringan untuk secara efektif mengekstraksi fitur-fitur penting dari gambar (Sanjaya & Ayub, 2020). Lapisan ini menggunakan operasi konvolusi untuk memindai citra input dengan sejumlah filter konvolusi. Setiap filter, juga dikenal sebagai *kernel*, berukuran kecil dan digeser secara bertahap di seluruh citra. Proses ini menghasilkan peta fitur atau "*feature map*" yang unggul keberadaan fitur-fitur tertentu seperti tepi, garis, tekstur, atau pola yang lebih kompleks. Setiap filter dihubungkan ke seluruh area citra melalui parameter yang dapat dipelajari yang disebut *weight* (bobot). Bobot ini diperbarui selama pelatihan untuk mengoptimalkan pengenalan pola (Geron, 2019).

   Persamaan *Output neuron* dari *convolutional layer* dapat ditulis sebagai berikut :

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_n-1} x_{i',j',k'} . w_{u,v,k',k} \quad \text{..... (1)}$$

dengan$= \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases}$

2. Lapisan Aktivasi (*Activation Layer*): Setelah lapisan konvolusi, lapisan aktivasi diterapkan menggunakan fungsi aktivasi non-linear seperti ReLU (*Rectified Linear Unit*). Fungsi aktivasi ini memperkenalkan non-linearitas ke jaringan dan membantu dalam memodelkan hubungan yang kompleks antara fitur-fitur input. ReLU mengubah semua nilai negatif menjadi nol, sementara nilai positif tetap tidak berubah. Hal ini membantu dalam memperkuat fitur-fitur yang signifikan dan membantu jaringan untuk belajar dengan lebih cepat. Rectified Linear Unit (ReLU) adalah fungsi aktivasi yang paling umum digunakan dalam konteks *Convolutional Neural Networks* (CNN). ReLU mengubah seluruh nilai input menjadi bilangan positif. Keuntungan utama ReLU dibandingkan dengan fungsi aktivasi lainnya adalah beban komputasi yang lebih rendah. (Alzubaidi, et al., 2021) Representasi matematis dari ReLU adalah sebagai berikut:

   $\boldsymbol{f(x) = \max(0, x)}$ ……(2)

3. Lapisan *Pooling*: Lapisan *pooling* digunakan untuk mengurangi dimensi spasial dari fitur-fitur yang ditemukan oleh lapisan konvolusi. Metode *pooling* yang umum adalah *max pooling*. *Pooling* membantu mengurangi jumlah parameter dalam jaringan dan menghasilkan fitur representasi yang lebih tahan terhadap pergeseran dan deformasi kecil pada citra. Ini juga membantu dalam menghindari *overfitting* dan membuat model lebih umum.
4. Lapisan Lengkap (*Fully Connected Layer*): Setelah lapisan *pooling*, fitur-fitur yang diekstraksi diberikan ke lapisan-lapisan yang sepenuhnya

terhubung. Lapisan ini bertindak sebagai “otak” jaringan, menerima fitur-fitur tersebut dan mempelajari hubungan antara fitur-fitur tersebut dengan kelas label yang sesuai. Lapisan ini mirip dengan lapisan-lapisan dalam jaringan syaraf biasa dan menggunakan bobot dan bias yang dapat dibentuk untuk mempelajari representasi fitur yang lebih kompleks dan melakukan klasifikasi yang akurat. Pada akhirnya, lapisan ini menghasilkan probabilitas untuk setiap kelas berdasarkan masukan fitur-fitur.

Selama pelatihan, bobot dan parameter jaringan diatur secara otomatis melalui proses optimisasi seperti *Stochastic Gradient Descent* (SGD) atau algoritma yang lebih canggih seperti Adam atau RMSprop. Proses ini melibatkan perhitungan kesalahan *gradien* menggunakan algoritma *backpropagation* dan memperbarui bobot jaringan untuk mengurangi prediksi kesalahan.

CNN telah mencatat kemajuan besar dalam pengolahan citra. Keunggulan utama CNN adalah kemampuannya untuk secara otomatis mengekstraksi fitur-fitur yang relevan dari citra data tanpa adanya penanganan manual. Dengan arsitektur dan lapisan-lapisannya, CNN telah mampu mencapai tingkat kinerja yang tinggi dalam berbagai tugas pengolahan dan pengklasifikasian gambar, seperti pengenalan objek, deteksi wajah, segmentasi citra, dan klasifikasi citra.

E. Confusion Matrics

Confusion Matrix yakni metode visualisasi untuk hasil algoritma klasifikasi. Dalam kata lain ini merupakan tabel yang memecah jumlah instansi kebenaran dasar dari kelas tertentu terhadap jumlah instansi kelas yang diprediksi (Murel

Ph.D. & Kavlakoglu, 2024). *Confusion Matrix* adalah salah satu dari beberapa metrik evaluasi yang mengukur kinerja model klasifikasi. Mereka dapat digunakan untuk menghitung sejumlah metrik kinerja model lainnya, seperti presisi dan *recall*, antara lain (Geron, 2019).

Confusion Matrix dapat digunakan dengan algoritma klasifikasi apa pun, seperti Naïve Bayes, model regresi logistik, pohon keputusan, dan sebagainya. Karena aplikabilitasnya yang luas dalam ilmu data dan model pembelajaran mesin, banyak paket dan pustaka dilengkapi dengan fungsi untuk membuat matriks kebingungan, seperti modul sklearn.metrics dari *scikit-learn* untuk *Python*.

Confusion Matrix memiliki *layout* standar yang umum digunakan, kolom mewakili nilai-nilai yang diprediksi dari suatu kelas tertentu sementara baris mewakili nilai-nilai aktual (yaitu kebenaran dasar) dari kelas tersebut, atau sebaliknya. Struktur grid ini adalah alat untuk memvisualisasikan akurasi klasifikasi model dengan menampilkan jumlah prediksi yang benar dan prediksi yang salah untuk semua kelas secara bersamaan.

Bentuk standar yang umum digunakan pada *Confusion Matrix* untuk klasifikasi biner dapat digambarkan sebagai berikut :

Gambar 2. 6 *Confusion Matrix.*

Sumber : IBM, 2024. (www.ibm.com)

Kotak di bagian kiri atas menyediakan jumlah *true positives* (TP), yaitu jumlah prediksi yang benar untuk kelas positif. Kotak di bawahnya adalah *false positives* (FP), yaitu instansi kelas negatif yang salah diidentifikasi sebagai kasus positif. Ini juga disebut sebagai kesalahan tipe I dalam statistik. Kotak di bagian kanan atas adalah jumlah *false negatives* (FN), yaitu instansi positif sebenarnya yang salah diprediksi sebagai negatif. Terakhir, kotak di bagian kanan bawah menampilkan jumlah *true negatives* (TN), yang merupakan instansi kelas negatif yang sebenarnya diprediksi negatif dengan benar. Menjumlahkan nilai-nilai ini akan memberikan total prediksi model.

Confusion Matrix tentu juga dapat digunakan untuk memvisualisasikan hasil untuk masalah klasifikasi multikelas. Sebagai contoh:

Gambar 2. 7 Contoh *Confusion Matrix Multiclass.*
Sumber : IBM, 2024.

Pada gambar tabel *Confusion Matrix* diatas merupakan salah satu contoh klasifikasi multikelas dari hasil prediksi jenis ikan. Untuk kotak diagonal menandakan jumlah prediksi benar (TP) dan selain itu maka berisi nilai *false positif* (FP), *true negative* (TN) dan *false negative* (FN).

F. Evaluasi Model

Evaluasi model pada CNN (*Convolutional Neural Network*) adalah proses untuk menilai kinerja dan akurasi model dalam mengklasifikasi objek atau matriks. Dilakukan melalui metode *evaluation matrix* yang menggunakan tabel *Confusion Matrix*. *Confusion Matrix* yang telah dibahas sering digunakan untuk mengukur kinerja dari model klasifikasi di *machine learning*, yang merupakan metode evaluasi yang dapat digunakan untuk menghitung kinerja atau tingkat kebenaran dari proses klasifikasi. "*Confusion Matrix* adalah salah satu teknik yang paling umum digunakan dalam ML, dan mencakup informasi tentang kelas aktual dan diprediksi yang diperoleh oleh sistem klasifikasi." (Taner, Oztekin, & Duran, 2021)

Adapun evaluasi matriks yang dapat digunakan untuk menghitung nilai dari *Accuracy, Precision, Recall* dan *F-1 Score* dari performa model *Machine Learning* yang telah dilatih (Rahman, 2024).

1. *Accuracy*

   Akurasi dari pengklasifikasi diukur dengan metrik ini. Jumlah data yang diklasifikasikan dengan benar dibagi oleh total jumlah data untuk menghitung akurasi. Rumus untuk menghitung akurasi adalah sebagai berikut:

   $$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{.....(3)}$$

   Keterangan :

   - TP adalah *True Positives* (prediksi benar untuk kelas positif),
   - TN adalah *True Negatives* (prediksi benar untuk kelas negatif),
   - FP adalah *False Positives* (prediksi salah untuk kelas positif), dan
   - FN adalah *False Negatives* (prediksi salah untuk kelas negatif).

   Dengan menggunakan rumus di atas, kita dapat menghitung akurasi dari pengklasifikasi dengan membagi jumlah prediksi yang benar (TP dan TN) dengan total jumlah data yang dievaluasi.

2. *Precision*

   Presisi menunjukkan seberapa banyak data yang diprediksi sebagai positif yang diprediksi dengan benar. Dengan kata lain, presisi yang tinggi berarti lebih sedikit *false positives*. Rumus untuk menghitung presisi adalah sebagai berikut :

   $$Precision = \frac{TP}{TP+FP} \text{.....(4)}$$

3. *Recall*

   Recall adalah metrik untuk menentukan kelengkapan dari pengklasifikasi. *Recall* yang lebih tinggi menunjukkan *false negatives* yang lebih rendah, sementara *recall* yang lebih rendah menunjukkan *false negatives* yang lebih tinggi. Presisi sering kali menurun dengan peningkatan *recall*. Rumus untuk menghitung *recall* adalah sebagai berikut:

$$Recall = \frac{TP}{TP+FN} \quad \text{..... (5)}$$

4. *F1-Score*

   Untuk mendapatkan *F1-Score*, hasil kali dari *recall* dan *precision* dibagi oleh jumlah dari *recall* dan *precision*. Rumus untuk menghitung *F1-Score* adalah sebagai berikut:

$$F1 - Score = 2\ x\ \frac{Recall\ x\ Precision}{Recall+Precision} \quad \text{.....(6)}$$

G. *Python*

Menurut (Raschka, 2020) "*Python* tetap menjadi bahasa yang paling disukai untuk komputasi ilmiah, ilmu data, dan *machine learning*, meningkatkan kinerja dan produktivitas dengan memungkinkan penggunaan *library* tingkat rendah dan antarmuka pemrograman aplikasi tingkat tinggi yang bersih".

Python memiliki peran penting dalam pembangunan model *Machine Learning* karena memiliki beberapa keunggulan yang membuatnya menjadi pilihan yang cocok untuk membangun *machine learning*, antara lain:

1. Kemudahan Penggunaan: *Python* dikenal karena kemudahannya dalam dipelajari dan digunakan. Bahasa pemrograman ini memiliki sintaksis yang sederhana dan mudah dipahami, sehingga memungkinkan para pengembang untuk dengan cepat membangun dan menguji model *Machine Learning*.
2. Ekosistem yang Kaya: *Python* memiliki ekosistem yang sangat kaya dengan berbagai *library* dan *framework* yang mendukung pengembangan *Machine Learning*, seperti *NumPy*, Pandas, *Scikit-learn*, *TensorFlow*, dan Keras. *Library-library* ini menyediakan berbagai alat yang diperlukan untuk memproses data, membangun model, dan menganalisis hasil dengan efisien.
3. Performa dan Produktivitas: *Python* memungkinkan penggunaan *library* tingkat rendah dan API tingkat tinggi yang bersih, sehingga dapat meningkatkan performa dan produktivitas dalam pengembangan model *Machine Learning*. Selain itu, *Python* juga mendukung penggunaan GPU untuk komputasi paralel, yang dapat meningkatkan kecepatan pelatihan model
4. Populer di Komunitas *Data Science*: *Python* telah menjadi bahasa yang sangat populer di komunitas *Data Science* dan *Machine Learning*. Banyak professional di bidang ini menggunakan *Python* karena kemudahan penggunaannya dan ekosistem yang mendukung. Hal ini membuat *Python* menjadi pilihan utama dalam pengembangan model *Machine Learning*

Dengan kombinasi dari kemudahan penggunaan, ekosistem yang kaya, performa yang baik, dan popularitasnya di komunitas *Computational Science*, *Python* menjadi pilihan yang sangat cocok untuk membangun model *Machine Learning*.

H. *Tensorflow*

TensorFlow merupakan *library* yang kuat untuk komputasi numerik, terutama sangat cocok dan disesuaikan untuk Pembelajaran Mesin skala besar. *Library* ini dikembangkan oleh tim Google *Brain* dan digunakan dalam banyak layanan skala besar Google, seperti Google *Cloud Speech*, Google *Photos*, dan Google *Search*. *TensorFlow* dirilis sebagai sumber terbuka pada November 2015, dan kini menjadi *library Deep Learning* paling populer (dilihat dari jumlah kutipan dalam makalah, adopsi di perusahaan, bintang di GitHub, dll.). Banyak proyek menggunakan *TensorFlow* untuk berbagai tugas Pembelajaran Mesin, seperti klasifikasi gambar, pemrosesan bahasa alami, sistem rekomendasi, *time series forecasting*. (Geron, 2019).

I. *Sckit-Learn*

Penggunaan *Scikit-learn* atau sklearn sebagai sebuah modul dalam bahasa pemrograman *Python* yang dibangun di atas *library* seperti *NumPy*, SciPy, dan *Matplotlib*. Fungsinya adalah untuk melakukan pelatihan data dalam konteks *machine learning* dan *data science* (Geron, 2019).

Sklearn menyediakan berbagai fitur yang sangat berguna, termasuk model-model klasifikasi, clustering, regresi berbasis model *machine learning*, dan proses-proses yang mendukung tahap feature engineering seperti reduksi dimensi menggunakan PCA (*Principal Component Analysis*) (Suh, Shin, Baang, & Mun, 2020).

Scikit-learn sangat populer di kalangan praktisi data science karena kemudahan penggunaannya dan kaya akan fitur yang disediakan. Dengan menggunakan *scikit-learn*, pengguna dapat dengan mudah mengimplementasikan berbagai algoritma *machine learning* dan melakukan berbagai tugas analisis data dengan efisien. Modul ini memungkinkan pengguna untuk memanggil berbagai fungsi dan prosedur yang diperlukan dalam proses pengembangan model *machine learning*, evaluasi model, dan pemrosesan data.

Dengan demikian, *scikit-learn* merupakan salah satu modul yang sangat penting dalam ekosistem *Python* untuk keperluan *machine learning* dan data science, dan menjadi pilihan utama bagi banyak praktisi di bidang tersebut.

J. *Numpy*

Penggunaan *Numpy* sebagai *library python* dapat membantu dalam pemrograman matematika, ilmiah, teknik, dan *data science*. Ini adalah *library* yang sangat berguna untuk melakukan operasi matematika dan statistika di *Python*. *NumPy* bekerja dengan sangat baik untuk array multi-dimensi dan perkalian matriks, serta mudah diintegrasikan dengan C/C++ dan Fortran. *NumPy* menyediakan berbagai fitur yang mengurangi kompleksitas tugas-tugas analis data,

ilmuwan data, dan peneliti. *NumPy* memberikan fleksibilitas *Python* dan kecepatan kode C yang dioptimalkan dengan baik. Sintaksisnya yang mudah digunakan membuatnya sangat mudah diakses dan produktif bagi para programer. *NumPy* dirancang untuk bekerja dengan array N-dimensi, aljabar linear, angka acak, transformasi *Fourier*, dan lain sebagainya (Gupta & Bagchi, 2024).

Setelah memahami dasar lingkungan *Python*, sangat bermanfaat untuk menjelajahi *library NumPy*. Meskipun *NumPy* sendiri tidak menyediakan fungsionalitas pemodelan atau ilmiah, memahami array *NumPy* dan komputasi berorientasi array akan membantu dalam menggunakan alat-alat dengan semantik berorientasi array, seperti pandas, dengan lebih efektif. Sementara *NumPy* menyediakan dasar komputasi untuk pemrosesan data numerik umum, banyak pembaca akan ingin menggunakan pandas sebagai dasar untuk sebagian besar jenis statistik atau analitika, terutama pada data tabular. Pandas juga menyediakan beberapa fungsionalitas yang lebih spesifik domain seperti manipulasi deret waktu, yang tidak ada dalam *NumPy*.

K. Preprocessing dataset

Proses *preprocessing* sangat penting dalam meningkatkan akurasi pengenalan gambar. Untuk melakukan hal ini, beberapa algoritma yang disediakan oleh OpenCV akan digunakan. Perlu diingat bahwa model juga dapat diuji menggunakan gambar dari *dataset* itu sendiri tanpa perlu melakukan proses *preprocessing* gambar (Dubey, Lazarus, & Mangal, 2020).

1. *Rotation:* Untuk penggunaan di dunia nyata, proyek kita juga harus mampu membaca *digit* tulisan tangan dari berbagai sudut dan orientasi yang berbeda. Oleh karena itu, rotasi digit yang diekstraksi dari gambar menjadi sangat penting. Rotasi adalah mengubah orientasi digit dalam langkah pra-pemrosesan untuk memudahkan proses pengenalan. Model akan dilatih untuk mampu merotasi digit sendiri agar pengenalan digit tulisan tangan menjadi lebih efisien.
2. *Resizing* : Proses *resizing* adalah penyesuaian dimensi gambar tanpa merusak kualitasnya. *resizing* diperlukan untuk memastikan semua gambar berada dalam format yang benar untuk diproses (Dubey, Lazarus, & Mangal, 2020). Ketika me-*resize* gambar, penting untuk mempertimbangkan dampaknya pada komponen terhubung (*connected components*) dalam gambar. *Connected components* adalah kelompok piksel dengan nilai yang sama yang saling terhubung melalui koneksi piksel. Dalam konteks digit tulisan tangan, *connected components* ini kemungkinan mewakili digit individu dalam gambar. Setelah proses *resizing*, *connected components* yang di-*filter* dapat diubah ukurannya menjadi ukuran standar.

L. Data Augmentation

Data Augmentation cukup efektif untuk digunakan sebagai metode untuk mengurangi terjadinya *Overfitting* dari model CNN yang dilatih, yang mungkin disebabkan oleh kurangnya *sample* yang digunakan selama pelatihan *Machine*

Learnin, Dengan meningkatkan jumlah, kualitas, dan keragaman data, model *machine learning* dapat belajar dari variasi yang lebih besar dalam data dan mengoptimalkan kinerjanya. Dengan kata lain, semakin banyak variasi yang diberikan pada model melalui *data augmentation*, semakin baik model tersebut dapat memahami dan menggeneralisasi pola-pola yang ada dalam data, sehingga meningkatkan efektivitas dan akurasi model (Zheng, Yang, Tian, Jiang, & Wang, 2020).

M. *Matplotlib*

Matplotlib adalah salah satu *library* populer dalam bahasa pemrograman *Python* yang digunakan untuk visualisasi data. *Matplotlib* menyediakan berbagai fungsi untuk membuat grafik, plot, diagram, dan visualisasi data lainnya dalam berbagai format seperti grafik garis, *scatter plot*, *histogram*, dan lain sebagainya (Rahman, 2024). *Library* ini memungkinkan pengguna untuk dengan mudah membuat visualisasi yang informatif dan menarik dari data.

Dalam membangun *Convolutional Neural Network* (CNN) untuk tugas pengenalan gambar atau visual, *Matplotlib* dapat digunakan untuk berbagai tujuan, antara lain:

1. Visualisasi *Dataset*: *Matplotlib* dapat digunakan untuk menampilkan contoh gambar dari *dataset*, memvisualisasikan distribusi kelas *dataset*.
2. Visualisasi performa Grafik model: *Matplotlib* dapat digunakan untuk menampilkan grafik *loss* dan akurasi model pada setiap

epoch. Hal ini membantu dalam melakukan evaluasi kinerja model selama pelatihan dan mengevaluasi apakah model sedang belajar dengan baik atau tidak (Rahman, 2024).

Dengan menggunakan *Matplotlib* dalam membangun CNN, pengguna dapat dengan mudah memvisualisasikan data, dan grafik hasil dari performa model, sehingga memudahkan dalam menganalisis dan memahami proses pembangunan model serta kinerjanya.

N. *Javascript*

JavaScript adalah bahasa pemrograman tingkat tinggi yang bersifat dinamis dan banyak digunakan dalam pengembangan web (Supriadi, 2020). Bahasa ini pertama kali dikembangkan oleh Brendan Eich pada tahun 1995 untuk Netscape Navigator dan sejak itu menjadi standar dalam teknologi web bersama HTML dan CSS. *JavaScript* memiliki berbagai karakteristik, seperti bersifat interpreted (dieksekusi langsung oleh browser tanpa perlu dikompilasi), dynamically typed (tidak memerlukan deklarasi tipe data secara eksplisit), serta mendukung paradigma pemrograman fungsional, berorientasi objek, dan imperatif. Selain itu, *JavaScript* juga dikenal sebagai bahasa yang event-driven dan asynchronous, sehingga sangat efisien dalam menangani interaksi pengguna dan komunikasi dengan server.

JavaScript memiliki banyak kegunaan, mulai dari manipulasi DOM (Document Object Model) untuk mengubah elemen HTML dan CSS secara dinamis, validasi formulir sebelum dikirim ke server, hingga pembuatan animasi dan efek visual menggunakan berbagai pustaka seperti GSAP dan Three.js. Selain itu, *JavaScript* memungkinkan interaksi asinkron dengan server menggunakan

teknologi AJAX, yang membuat halaman web lebih responsif tanpa harus dimuat ulang. Saat ini, *JavaScript* tidak hanya digunakan untuk pengembangan sisi klien (client-side), tetapi juga dapat digunakan untuk pengembangan sisi server (server-side) melalui Node.js, yang memungkinkan pengembang membangun aplikasi web full-stack hanya dengan *JavaScript*.

O. *Postmann*

Postman sebagai platform kolaborasi untuk pengembangan API yang dapat digunakan untuk menguji model *Convolutional Neural Network* (CNN). Dengan Postman, pengguna dapat mengirimkan permintaan POST ke endpoint API model CNN, menyertakan gambar sebagai data input. *Respons* dari API akan berisi hasil klasifikasi dari model CNN untuk gambar yang dikirim. Pengguna dapat menganalisis hasil klasifikasi untuk memastikan model memberikan hasil yang sesuai. Dengan melakukan pengujian berulang menggunakan Postman, pengembang dapat memvalidasi kinerja model CNN dan memastikan keakuratannya dalam menerima dan memproses data gambar untuk klasifikasi (Manuaba, Sutedja, & Bahana, 2020).

P. *Python Flask*

Flask dapat digunakan untuk membuat permintaan HTTP (*Hyper Text Transfer Protocol*) serta menerima permintaan tersebut . Metode ini digunakan untuk menghubungkan model *machine learning* ke bagian belakang (backend) (Amanzadi & Karim, 2022).

Flask dianggap 'ringan' karena hanya memerlukan satu file *Python* untuk seluruh *framework* web. Namun, hal ini tidak berarti *Flask* mengorbankan fungsionalitas dalam proses tersebut. Permintaan HTTP ke *Flask* terstruktur dalam bentuk fungsi-fungsi dalam pemrograman *Python*. Setiap fungsi ini dapat dihubungkan dengan URL spesifik (*Uniform Resource Locator*). Setiap kali URL tersebut diakses, maka fungsi yang sesuai akan dijalankan.

Q. HTML

HTML singkatan dari *HyperText Markup Language*, digunakan sebagai dasar dalam membangun struktur dan konten pada halaman web. Dengan menggunakan tag yang mengelilingi elemen-elemen, HTML memberikan instruksi kepada *browser* untuk menampilkan konten sesuai keinginan. Elemen-elemen ini termasuk teks, gambar, video, dan tautan yang bekerja sama untuk membentuk struktur dan menyampaikan informasi pada halaman web (Denishtsany, 2023).

HTML memiliki peran penting dalam menyusun konten dan informasi pada halaman web. Setiap elemen HTML memiliki tujuan dan fungsi spesifik. Misalnya, tag <h1> hingga <h6> digunakan untuk judul dengan tingkat prioritas yang berbeda, sementara tag <p> untuk paragraf teks. Selain itu, HTML memungkinkan integrasi elemen multimedia seperti gambar, video, dan audio ke dalam halaman web.

R. CSS

Cascading Style Sheets berfungsi sebagai bahasa stylesheet yang bertanggung jawab dalam mengatur tampilan dan tata letak halaman web. Dengan bantuan CSS,

pengembang web dapat mengubah berbagai aspek visual dari sebuah halaman web, seperti warna teks, posisi elemen, ukuran gambar, dan berbagai efek visual lainnya (Denishtsany, 2023).

Secara fungsional, CSS berperan penting dalam mengontrol cara elemen-elemen HTML ditampilkan di sebuah halaman web. Tanpa adanya CSS, halaman web akan tampak sederhana dan kurang menarik, dengan tampilan yang polos dan tanpa gaya. Melalui CSS, pengembang web dapat menciptakan desain yang konsisten, estetis, dan menarik secara visual, yang pada akhirnya akan meningkatkan pengalaman pengguna saat menjelajahi halaman web tersebut. Dengan kata lain, CSS memainkan peran kunci dalam menciptakan tampilan yang estetis dan fungsional dari sebuah situs web

S. Visual Studio Code

Visual Studio merupakan *Integrated Development Environment* (IDE) yang dikembangkan oleh Microsoft untuk membuat aplikasi Desktop, antarmuka pengguna grafis (GUI), aplikasi konsol, aplikasi web, aplikasi mobile, cloud, dan layanan web, dll. Dengan bantuan IDE ini, Anda dapat membuat kode yang dikelola serta kode asli. IDE ini menggunakan berbagai platform perangkat lunak pengembangan Microsoft seperti Windows *Store*, Microsoft Silverlight, dan Windows API, dll. Visual Studio bukanlah IDE yang spesifik untuk satu bahasa tertentu karena Anda dapat menggunakannya untuk menulis kode dalam berbagai Bahasa pemrogrman seperti *Python*, PHP dan *Javascript*. Visual Studio Code dapat

digunakan berbagai Sistem Oprasi seperti Windows, MacOs, dan Linux (Introduction Visual Studio Code, 2019).

T. *Jupyter Notebook*

Jupyter Notebook adalah sebuah lingkungan pengembangan interaktif yang sering digunakan oleh para praktisi *machine learning* (ML) untuk melakukan eksperimen dengan cepat dalam mengembangkan solusi ML. Dalam penggunaan *Jupyter Notebook*, menjelaskan aktivitas yang dilakukan dalam setiap sel kode sangat penting untuk meningkatkan keterbacaan dan pemahaman dari Notebook tersebut. Namun, proses anotasi manual pada setiap sel kode dapat menjadi sangat memakan waktu dan rentan terhadap kesalahan. Oleh karena itu, diperlukan pendekatan yang lebih efisien dan akurat untuk melakukan anotasi pada sel kode dalam *Jupyter Notebook* guna meningkatkan produktivitas dan mengurangi potensi kesalahan yang mungkin terjadi (Bredesen & Rehmsmeier, 2022).

U. *White Box Test*ing

White Box Testing, atau pengujian struktural, adalah metode pengujian yang memeriksa struktur internal dan logika kode sumber dari sistem yang diuji. Dalam metode ini, *test*er memeriksa kode sumber untuk memastikan bahwa semua jalur kode dieksekusi dan mengidentifikasi area potensial untuk kesalahan. *White Box Testing* memerlukan pengetahuan teknis yang mendalam tentang bahasa pemrograman dan teknologi yang digunakan (Wairooy, 2023).

V. *Black Box Test*ing

Black Box Testing merupakan uji fungsional yang berfokus pada fungsiona-

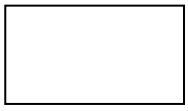
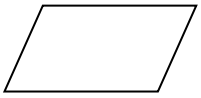
litas dan perilaku eksternal atau dalam kata lain tampilan dari aplikasi tanpa mempertimbangkan struktur *internal* atau kode sumber. Untuk melakukan *test*ing spesifikasi kebutuhan dan desain sistem untuk menciptakan skenario pengujian yang mencakup *input* dan *output* dari sistem (Wairooy, 2023).

W. Diagram Alir (*Flowchart*)

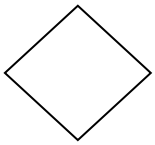
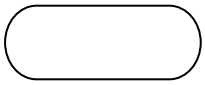
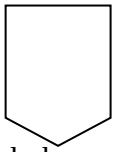
Flowchart adalah diagram yang digunakan untuk menggambarkan proses, sistem, atau algoritma dengan menggunakan simbol-simbol geometris seperti persegi panjang, oval, dan panah untuk menggambarkan langkah-langkah dan aliran informasi. *Flowchart* digunakan untuk mendokumentasikan, menganalisis, merencanakan, dan memahami proses-proses kompleks dengan cara yang mudah dipahami (Lucidchart, 2023).

Berikut adalah simbol-simbol yang umum atau sering dijumpai dalam penggunaan proses pembuatan *flowchart*:

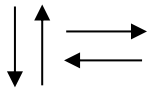
Tabel 2. 1 Simbol *Flowchart* umum digunakan

No.	Simbol	Keterangan
1	 Proses	Menunjukkan suatu proses/pengolahan. Digunakan untuk melambangkan perhitungan dan perubahan nilai peubah.
2	 Operasi *input/output*	Menunjukkan operasi input/*output*. Digunakan untuk melambangkan input/masukan dan *output*/keluaran.

Lanjutan Tabel 2.1

No.	Simbol	Keterangan
3	 Persiapan (*preparation*)	Menunjukkan suatu persiapan. Digunakan untuk memberikan nilai awal pada suatu peubah (*variable*) dan permulaan dari suatu perulangan.
4	 Keputusan (*Decision*)	Menunjukkan proses pembuatan keputusan. Digunakan untuk suatu pilihan/percabangan (ya/tidak).
5	 Terminal (*Terminator*)	Digunakan untuk menunjukkan awal dan akhir suatu program/*flowchart.*
6	 Penghubung (*Connector*)	Digunakan sebagai penghubung antar simbol yang terpisah (dalam satu halaman).
7	 Penghubung antar halaman (*offpage Connector*)	Digunakan sebagai penghubung antar simbol yang terpisah (antar halaman).
8	 Modul	Menunjukkan suau proses/subproses yang telah ditentukan. Dapat berupa suatu prosedur atau fungsi.

Lanjutan Tabel 2.1

No.	Simbol	Keterangan
9	 Panah	Menunjukkan arah dari suatu proses.

X. UML (*Unified Modelling Language*)

UML merupakan salah satu alat yang sering digunakan untuk merancang pengembangan *software* berbasis *object-oriented*. UML menekankan standar penulisan sebuah sistem *blueprint*, yang meliputi konsep dari proses bisnis, penulisan kelas-kelas dalam Bahasa program yang spesifik, skema *database*, dan komponen yang diperlukan dalam sistem *software* (Sonata & Sari, 2019).

UML mendefinisikan notasi dan *syntax*/semantik. Notasi dari UML umumnya merupakan sekumpulan bentuk atau *symbol* khusus untuk menjelaskan menggambarkan dari sistem perangkat lunak. Setiap bentuk pada UML memiliki arti khusus, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan.

Tabel 2. 2 *Symbol Use Case* Diagram

No	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menypesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *Use Case*.

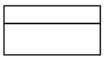
Lanjutan tabel 2.2

No.	Simbol	Nama	Keterangan
2		*Dependency*	Hubungan di mana perubahan yang terjadi pada suatu elemen mandiri (*Independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
3		*Generalization*	Hubungan di mana objek anak *(Descendent*) berbagai perilaku dan struktur data dari objek yang ada di atasnya objek induk (*Ancestor*).
4		*Include*	Menypesifikasikan bahwa *Use Case* sumber secara Eksplisit.
5		*Extend*	Menypesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menypesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor*

Lanjutan Tabel 2.2

No.	Simbol	Nama	Keterangan
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 2. 3 *Symbol Class* Diagram

No.	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan di mana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor*
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.

Lanjutan Tabel 2.3

No.	GAMBAR	NAMA	KETERANGAN
6		*Dependency*	Hubungan di mana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

Tabel 2. 4 *Symbol Sequence Diagram*

No	GAMBAR	NAMA	KETERANGAN
1		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2		*Message*	Spesifikasi dari komunikasi antar objek yang memuat in*form*asi-in*form*asi tentang aktivitas yang terjadi
3		*Message*	Spesifikasi dari komunikasi antar objek yang memuat in*form*asi-in*form*asi tentang aktivitas yang terjadi

Tabel 2. 5 *Symbol State Chart Diagram*

No	GAMBAR	NAMA	KETERANGAN
1		*State*	Nilai atribut dan nilai Link pada suatu waktu tertentu, yang dimiliki oleh suatu objek.
2		*Initial Pseudo State*	Bagaimana objek dibentuk atau diawali
3		*Final State*	Bagaimana objek dibentuk dan dihancurkan
4		*Transition*	Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya
5		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
6		*Node*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 2. 6 *Symbol Activity* Diagram

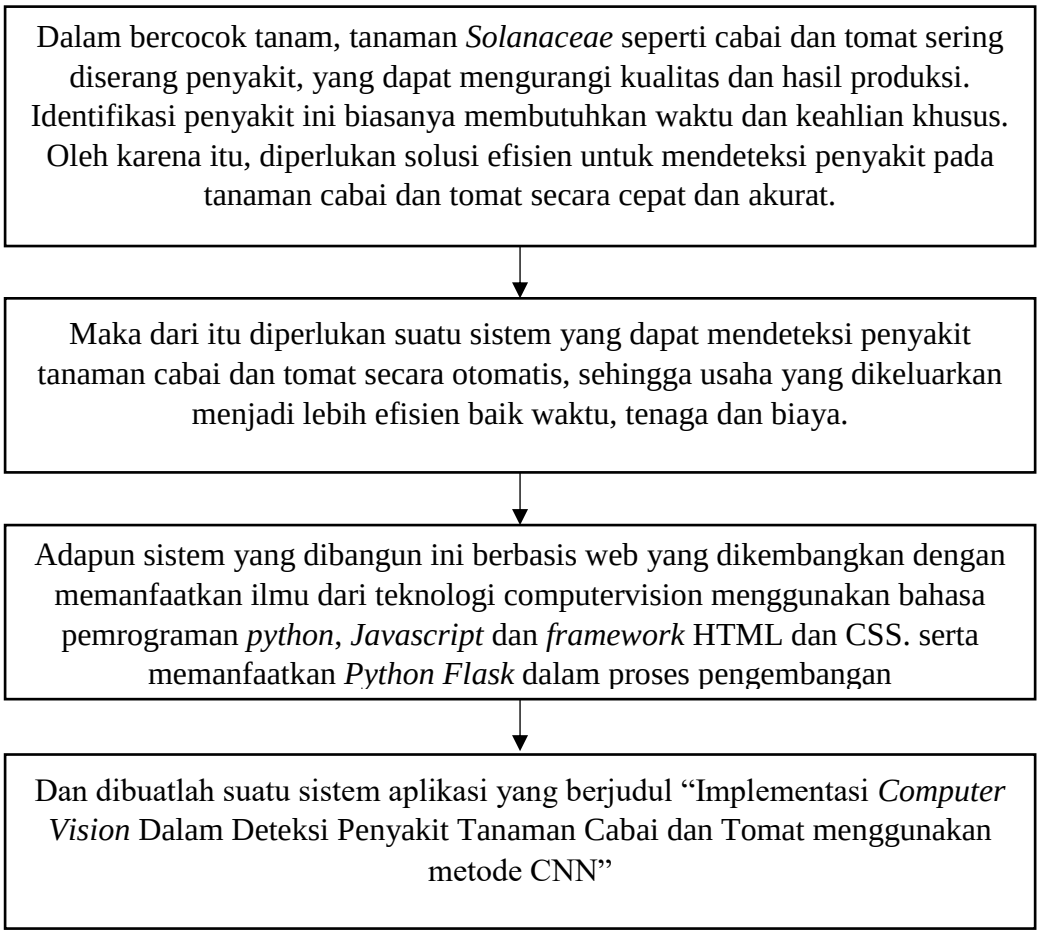
No.	GAMBAR	NAMA	KETERANGAN
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain

Lanjutan Tabel 2.6

No.	GAMBAR	NAMA	KETERANGAN
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

Y. Kerangka Berpikir

Agar dapat lebih memahami alur penelitian yang dilakukan, maka penelitian dapat diuraikan kedalam kerangka berpikir yang disajikan dalam bentuk diagram berikut :



BAB III

METODE PENELITIAN

A. Jenis Penelitian

Penelitian yang diangkat merupakan jenis penelitian eksperimental dengan menggunakan metode kuantitatif yang menggunakan *dataset* penyakit pada tanaman cabai dan tomat. Penelitian ini melibatkan analisa sejumlah data dan grafik dari angka-angka yang dihasilkan selama proses pembangunan model pelatihan *Machine Learning.*

B. Alokasi Waktu

waktu penelitian yang diambil 3 (tiga) bulan dengan rincian alokasi waktu sebagai berikut :

Tabel 3. 1 Jadwal Kegiatan Penelitian

JADWAL KEGIATAN PENELITIAN											
NO	KEGIATAN	Oktober 2024		Desember 2024				Januari 2025			
		3	4	1	2	3	4	1	2	3	4
1	Studi Literatur										
2	Pengumpulan *Dataset*										
3	Persiapan Peralatan/Environment										
4	Pelatihan Model										
5	Evaluasi Model										
6	Pembuatan Aplikasi										
7	Deployment										
8	Pengujian Unjuk Kinerja										
9	Analisa Hasil Pengujian										
10	Penulisan Hasil dan Pembahasan										

C. Alat dan Bahan

Adapun alat dan bahan yang digunakan selama proses penelitian ini adalah sebagai berikut :

1. Perangkat yang digunakan selama proses penelitian:

   Laptop Asus X441U dengan spesifikasi :

Tabel 3. 2 Spesifikasi Laptop

No.	Nama	Spesifikasi
1	Processor	Intel Core I3-6006U, 2.0GHz
2	RAM	12 GB DDR4
3	HDD	512 GB
4	SSD	256 GB
4	Layar	14 inci

2. Aplikasi yang digunakan:

Tabel 3. 3 Aplikasi yang digunakan

No.	Jenis	Software
1	Sistem Oprasi	Windows 10 Home 64 Bit
2	Code Editor	Visual Studio Code, *Jupyter Notebook*
4	Browser	Firefox/Chrome

3. Bahasa Pemrograman

Tabel 3. 4 Bahasa Pemrograman

No.	Bahasa Pemrograman	Penggunaan
1	Python	Digunakan sebagai Bahasa Pemrograman utama untuk membangun model CNN menggunakan TensorFlow/Keras, melakukan preprocessing data, dan membuat API dengan Flask.
2	Javascript	Digunakan untuk meningkatkan interaktivitas aplikasi, termasuk menangani pop-up hasil prediksi, komunikasi AJAX, dan manipulasi DOM.

4. Penggunaan *tools* pengembangan

Tabel 3. 5 Penggunaan *tools* Pengembangan

No.	*Tools*	Penggunaan
1	HTML	Digunakan untuk menyusun struktur halaman website yang digunakan sebagai antarmuka aplikasi.
2	CSS	Digunakan untuk mendesain tampilan antarmuka agar lebih estetis dan responsif dengan konsep *glassmorphism.*
3	*Python Flask*	Digunakan sebagai backend untuk menangani *request* dan *response* dalam aplikasi berbasis *web.*
4	*TensorFlow & Keras*	*Framework deep learning* yang digunakan untuk membangun, melatih, dan mengevaluasi model CNN.
5	*Scikit-learn (sklearn)*	Digunakan untuk evaluasi model dengan metrik seperti *Confusion Matrix, Precision, Recall,* dan *F1-score.*
6	*Matplotlib & Seaborn*	Digunakan untuk visualisasi data dan analisis performa model.
7	*NumPy & Pandas*	Digunakan untuk manipulasi data dan pengolahan dataset sebelum dimasukkan ke model.
8	*Werkzeug*	Digunakan untuk menangani *file upload* pada aplikasi berbasis *Flask.*
9	*MathJax.js*	Menampilkan dan merender persamaan matematika dalam format *LaTeX* di halaman *web*.
10	*jQuery 3.6.0*	Digunakan untuk menangani interaksi pengguna di sisi klien, seperti *AJAX* dan manipulasi *DOM.*

D. Teknik Pengumpulan Data

ada beberapa teknik dalam melakukan pengumpulan data yang akan dilakukan untuk mendukung proses penelitian seperti:

1. Studi Literatur : studi literartur dilakukan untuk menambah informasi mengenai pengembangan model *machine learning* dengan *convolutional neural network* (CNN).

2. Observasi : observasi dilakukan untuk melihat dan mengumpulkan data secara langsung pada tanaman yang sehat maupun pada tanaman terjangkit penyakit.
3. *Dataset* publik : *dataset* publik dapat tersedia di beberapa *platform* seperti kaggle, pengambilan *dataset* publik ini dapat membantu untuk menambah jumlah variasi *dataset*.

E. Teknik Analisa Data

1. Evaluasi *Matrix*

   Metode evaluasi menggunakan matriks, atau yang sering disebut *Confusion Matrix*, adalah alat yang digunakan untuk mengevaluasi kinerja model klasifikasi. *Confusion Matrix* menyediakan gambaran visual tentang performa model dengan membandingkan hasil prediksi model dengan nilai sebenarnya dari *dataset*.

2. *Black Box*

   Uji coba *Black Box (Black Box Testing)* adalah salah satu metode pengujian perangkat lunak yang dilakukan tanpa memerhatikan struktur internal atau kode sumber dari sistem yang diuji. Pendekatan ini berfokus pada fungsionalitas dan perilaku eksternal dari aplikasi atau sistem untuk memverifikasi apakah fungsi-fungsi yang diharapkan berfungsi dengan benar sesuai dengan spesifikasi yang telah ditentukan.

3. *White Box*

   Uji coba *White Box (White Box Testing)* adalah metode pengujian perangkat lunak yang dilakukan dengan memeriksa dan memahami struktur

internal, desain, dan kode sumber dari sistem yang diuji. Pendekatan ini memerlukan pengetahuan mendalam tentang bagaimana sistem dibangun dan beroperasi untuk mengidentifikasi kesalahan, cacat, atau kekurangan yang mungkin ada dalam kode.

F. Tahapan Penelitian

1. Studi Literatur

   Dalam tahap ini, dilakukan studi literatur untuk memahami dasar teoritis dari *computer vision*, metode Convolutional Neural Network (CNN), dan penyakit yang umumnya menyerang tanaman cabai dan tomat. Kajian literatur ini bertujuan untuk mendapatkan wawasan mendalam mengenai penelitian-penelitian terkait yang telah dilakukan sebelumnya, serta untuk memahami pendekatan dan teknik yang telah diterapkan dalam deteksi penyakit tanaman.

2. Pengumpulan Data

   Pada tahap pengumpulan data, sumber gambar tanaman cabai dan tomat yang terinfeksi dan sehat dipilih untuk digunakan dalam penelitian. Proses akuisisi data ini penting untuk memastikan ketersediaan *dataset* yang representatif dan diversifikasi gambar yang diperlukan untuk pelatihan model CNN.

3. Pengolahan dan *Preprocessing* Data

   Data yang telah dikumpulkan kemudian diolah dan di-*preprocessing* untuk persiapan analisis lebih lanjut. Proses ini mencakup normalisasi data dan augmentasi gambar untuk meningkatkan kualitas dan keanekaragaman *dataset*.

Selain itu, data dibagi menjadi set pelatihan, validasi, dan pengujian untuk pelatihan dan evaluasi model.

4. Pengembangan Model CNN

   Arsitektur model CNN yang sesuai dengan kebutuhan penelitian dikembangkan dalam tahap ini. Proses pelatihan model dilakukan menggunakan set data pelatihan untuk mengoptimalkan performa model. Selama proses ini, hyperparameter model dioptimalkan untuk meningkatkan akurasi dan generalisasi model.

5. Validasi dan Evaluasi Model

   Model yang telah dikembangkan kemudian divalidasi dan dievaluasi menggunakan set data validasi dan pengujian. Metrik evaluasi, seperti akurasi, presisi, *recall*, dan *F1-score*, dianalisis untuk mengukur performa dan keandalan model dalam deteksi penyakit tanaman cabai dan tomat.

6. Implementasi dan *Deployment*

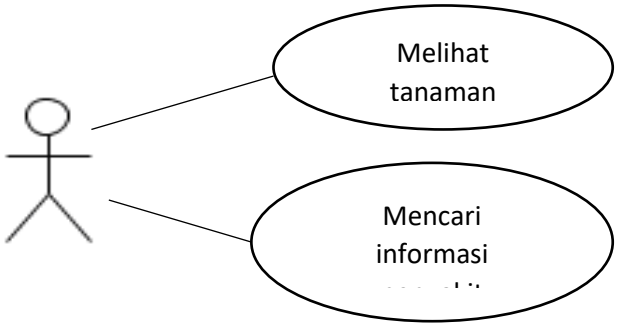
   Setelah melalui proses validasi, model CNN diintegrasikan ke dalam aplikasi atau platform yang dapat diakses untuk pengujian lebih lanjut. Uji coba aplikasi secara keseluruhan dilakukan untuk memastikan fungsionalitas dan kinerja model dalam lingkungan produksi.

7. Evaluasi dan Analisis Hasil

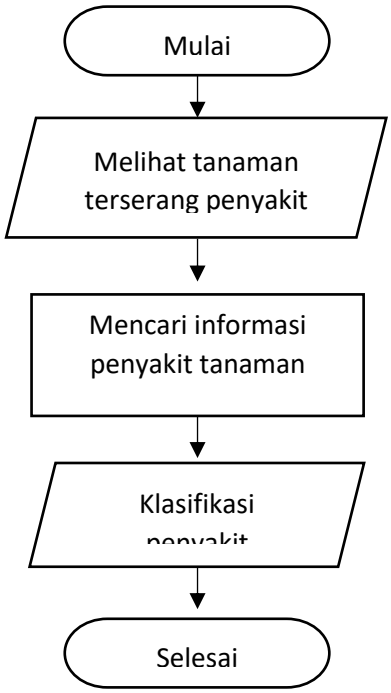
   Hasil dari implementasi model CNN dievaluasi dan dianalisis untuk menentukan keefektifan dan keandalan deteksi penyakit pada tanaman cabai dan tomat. Diskusi dilakukan mengenai interpretasi hasil, keterbatasan model, serta tantangan yang dihadapi selama proses penelitian.

G. Desain Sistem

1. Desain sistem yang berjalan



Gambar 3. 1 Diagram UML berjalan



Gambar 3. 2 *Flowchart* sistem berjalan

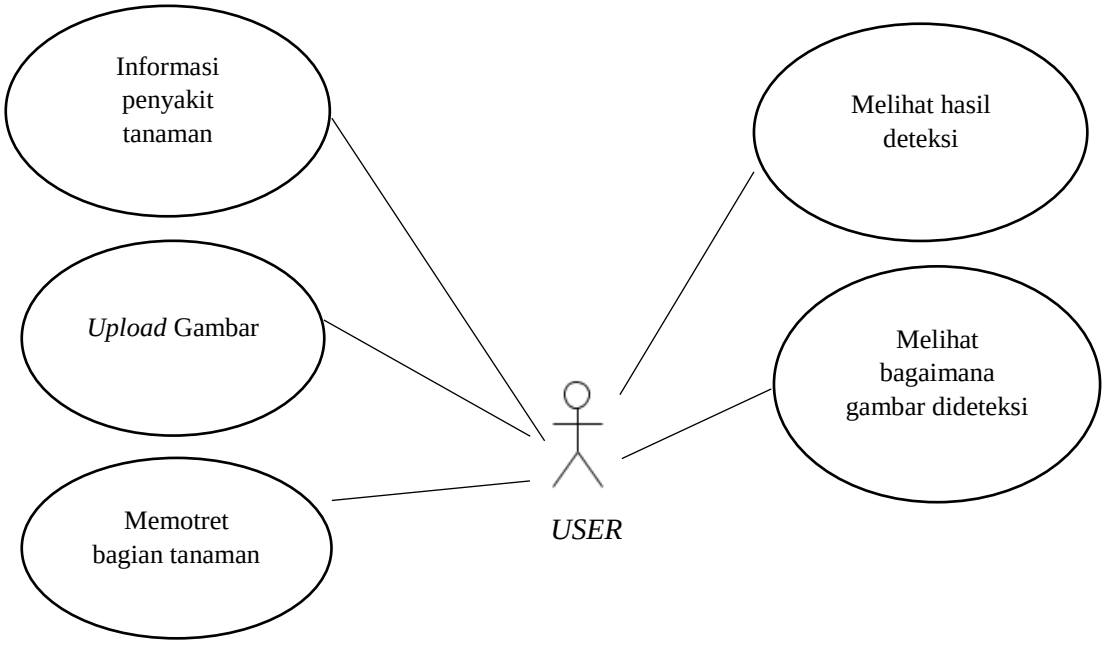
Prosedur deteksi penyakit tanaman cabai dan tomat yang sedang berjalan sebagai berikut :

a. Pengguna melihat bahwa tanaman tomat atau cabainya terserang penyakit tanaman

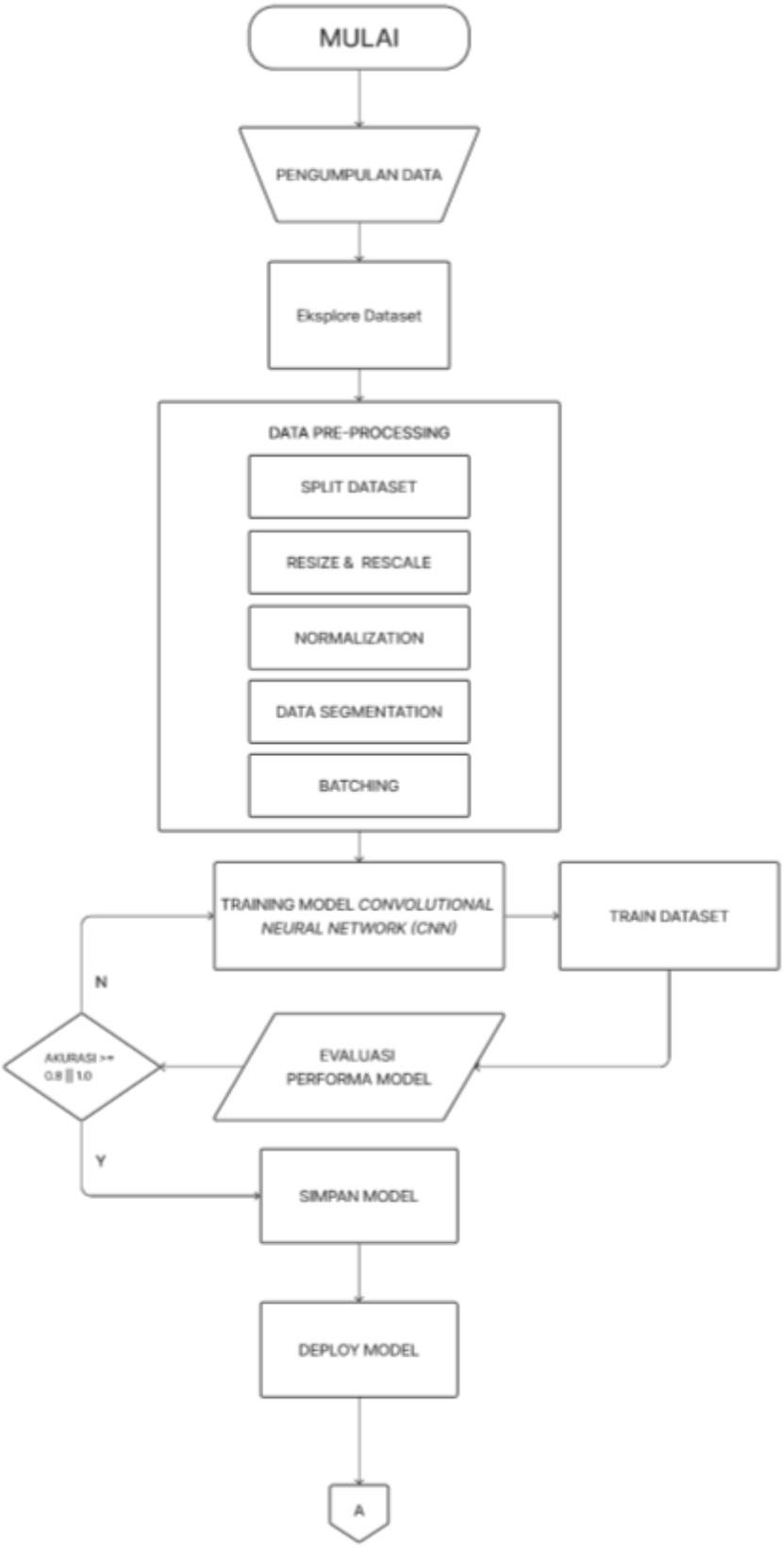
b. Pengguna mencari informasi mengenai gejala penyakit pada tanaman cabai atau tanaman tomat guna mendeteksi penyakit apa yang menyerang tanaman cabai atau tomatnya.

c. Setelah mendapatkan informasi yang cukup barulah pengguna dapat mulai mendeteksi penyakit apa yang sedang menyerang tanaman.

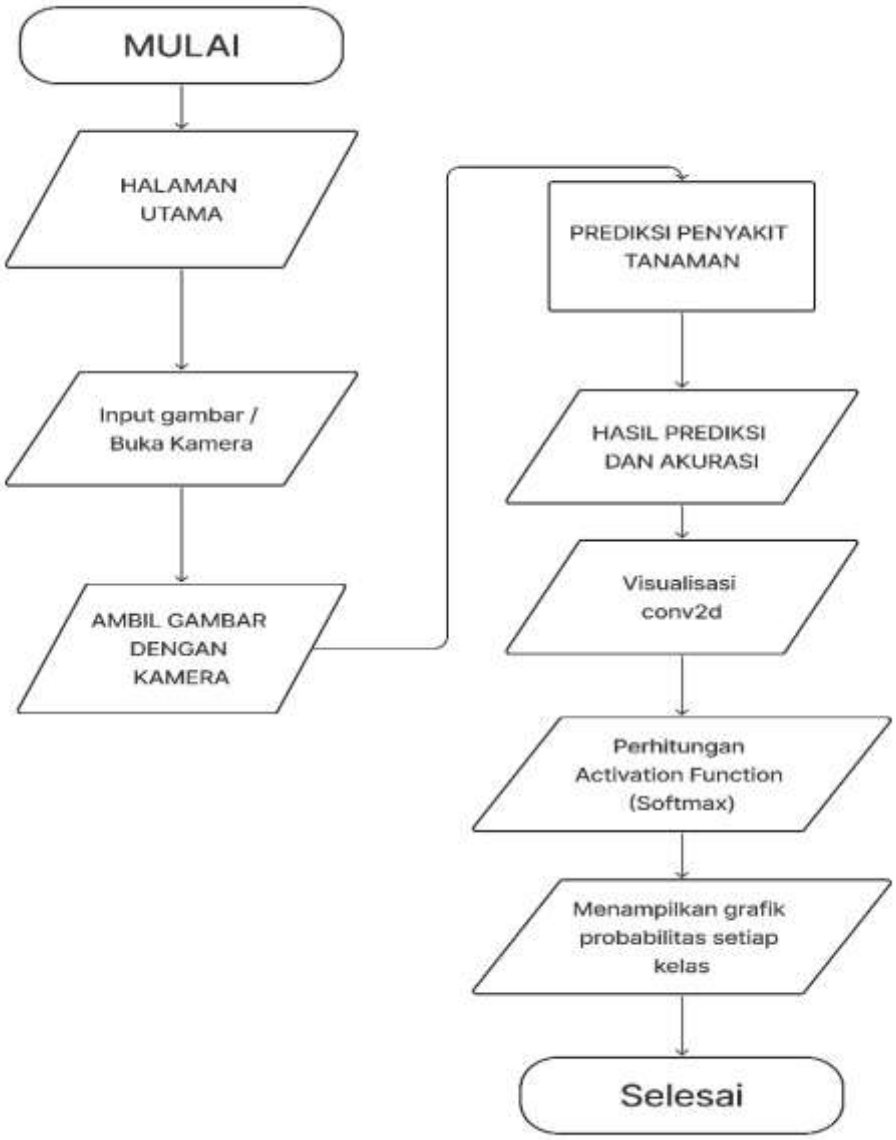
2. Desain sistem yang diusulkan



Gambar 3. 3 *Use Case* sistem yang diusulkan



Gambar 3. 4 *Flowchart Training* model



Gambar 3. 5 *Flowchart* Sistem Yang Diusulkan

Prosedur deteksi penyakit pada tanaman cabai dan tomat yang diusulkan :

a. User

Pada sistem yang diusulkan *User* tidak perlu melakukan registrasi akun maupun proses *login*, *user* langsung dibawa ke halaman Utama. Pada

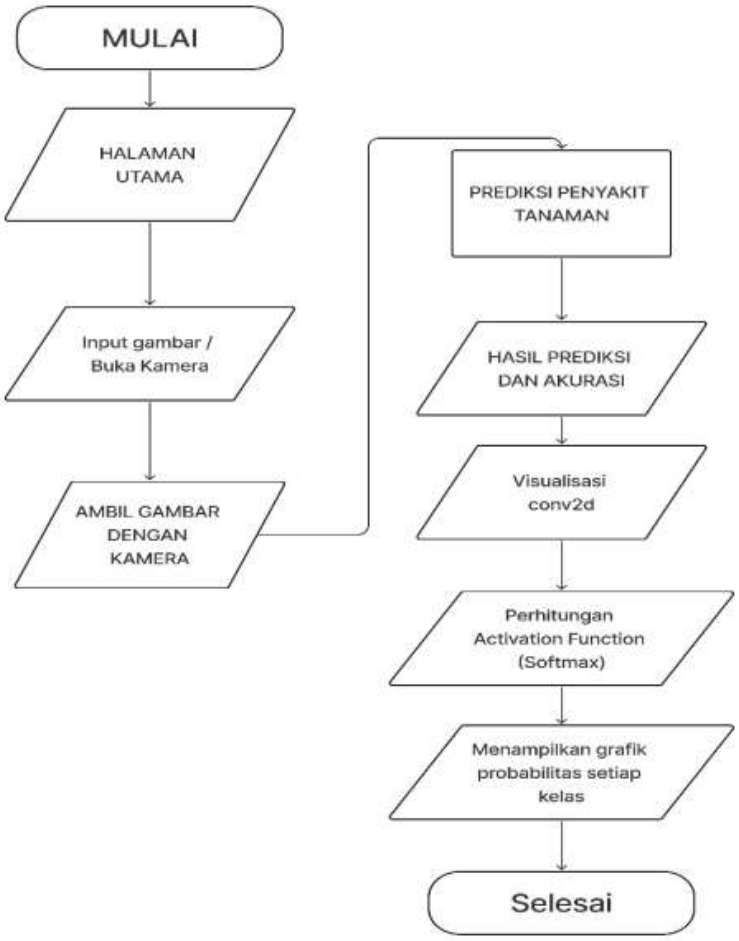
halaman utama, *User* dapat mengakses informasi penyakit apa saja yang bisa dideteksi. *User* dapat langsung melakukan pengambilan gambar penyakit pada tanaman cabai atau tomat secara langsung melalui kamera maupun menggunakan menu *input* gambar apabila sudah memiliki gambar dari penyakit yang ingin di deteksi.

BAB IV

HASIL DAN PEMBAHASAN

A. Diagram Alir

1. *Flowchart* pengguna



Gambar 4. 1 *Flowchart* pengguna

Flowchart interaksi pengguna ini menggambarkan tahapan yang dilalui dalam aplikasi deteksi penyakit tanaman, dimulai dari halaman utama hingga

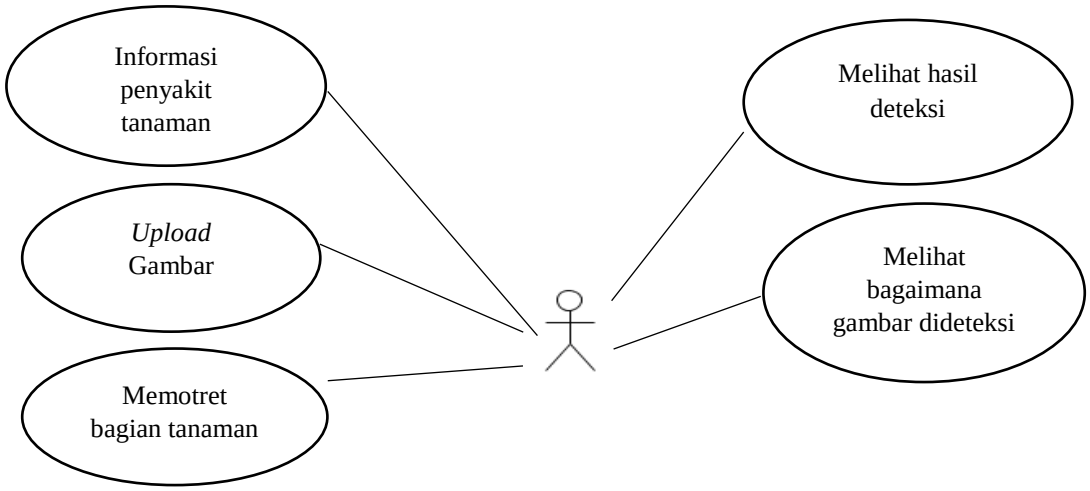
menampilkan hasil analisis. Proses diawali dengan pengguna membuka aplikasi dan masuk ke halaman utama. Di sini, mereka diberikan pilihan untuk mengunggah gambar dari galeri atau mengambil gambar langsung menggunakan kamera. Setelah gambar diperoleh, sistem akan memprosesnya untuk melakukan prediksi penyakit tanaman menggunakan model *Convolutional Neural Network* (CNN).

Setelah prediksi selesai, aplikasi menampilkan hasil deteksi penyakit beserta tingkat akurasinya. Untuk memberikan pemahaman lebih lanjut, sistem juga menyediakan visualisasi dari lapisan *convolutional* (Conv2D), yang memperlihatkan bagaimana gambar diolah oleh jaringan saraf tiruan. Selain itu, aplikasi menghitung nilai probabilitas dari setiap kelas penyakit menggunakan fungsi aktivasi *Softmax*, yang kemudian divisualisasikan dalam bentuk grafik. Grafik ini membantu pengguna memahami seberapa besar kemungkinan gambar tersebut diklasifikasikan ke dalam kategori tertentu.

B. Analisis Sistem Yang Diusulkan

1. *Use Case* Diagram

   Use case diagram digunakan untuk melihat hubungan antar actor dan sistem. Mendeskripsikan interaksi antara keduanya.



Gambar 4. 2 *Use case* sistem yang diusulkan

Tabel 4. 1 Penjelasan *use case* diagram pengguna

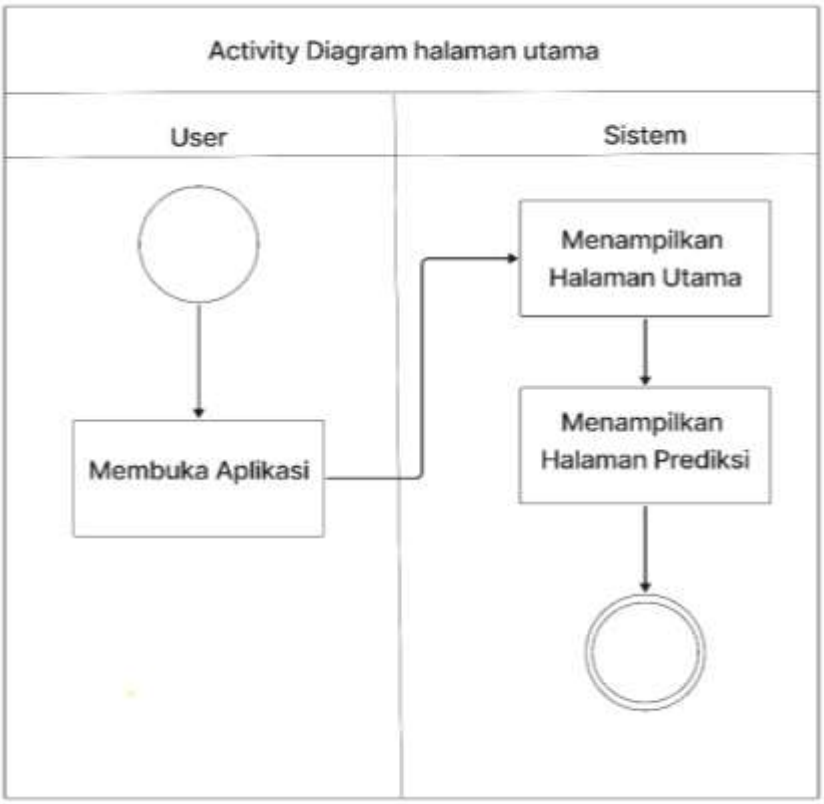
Nama *Use case*	Deskripsi *Use Case*
Melihat Hasil Deteksi	Setelah gambar diproses oleh sistem, pengguna dapat melihat hasil deteksi apakah tanaman terkena penyakit atau tidak
Informasi penyakit	Pengguna mendapatkan informasi penyakit yang terdeteksi, seperti penyebab dan gejala pada tanaman yang terindikasi terserang penyakit.
Upload gambar	Pengguna dapat melakukan upload gambar yang sudah tersedia untuk dideteksi
Memotret bagian tanaman	Pengguna dapat langsung mengambil foto pada bagian tanaman yang ingin diperiksa
Melihat bagaimana gambar dideteksi	Pengguna dapat melihat proses analisis gambar, termasuk visualisasi Conv2D, perhitungan *output layer* yang menghasilkan probabilitas, dan grafik probabilitas yang menunjukkan tingkat kepastian deteksi. Fitur ini membantu pengguna memahami cara kerja sistem dalam menghasilkan deteksi.

2. *Activity* diagram

Activity diagram berguna untuk memberikan gambaran proses simulasi dari alur kerja atau aliran control dari pengguna. Diagram aktifitas ini menampilkan urutan pelaksanaan dari sistem.

a. Activity diagram pengguna

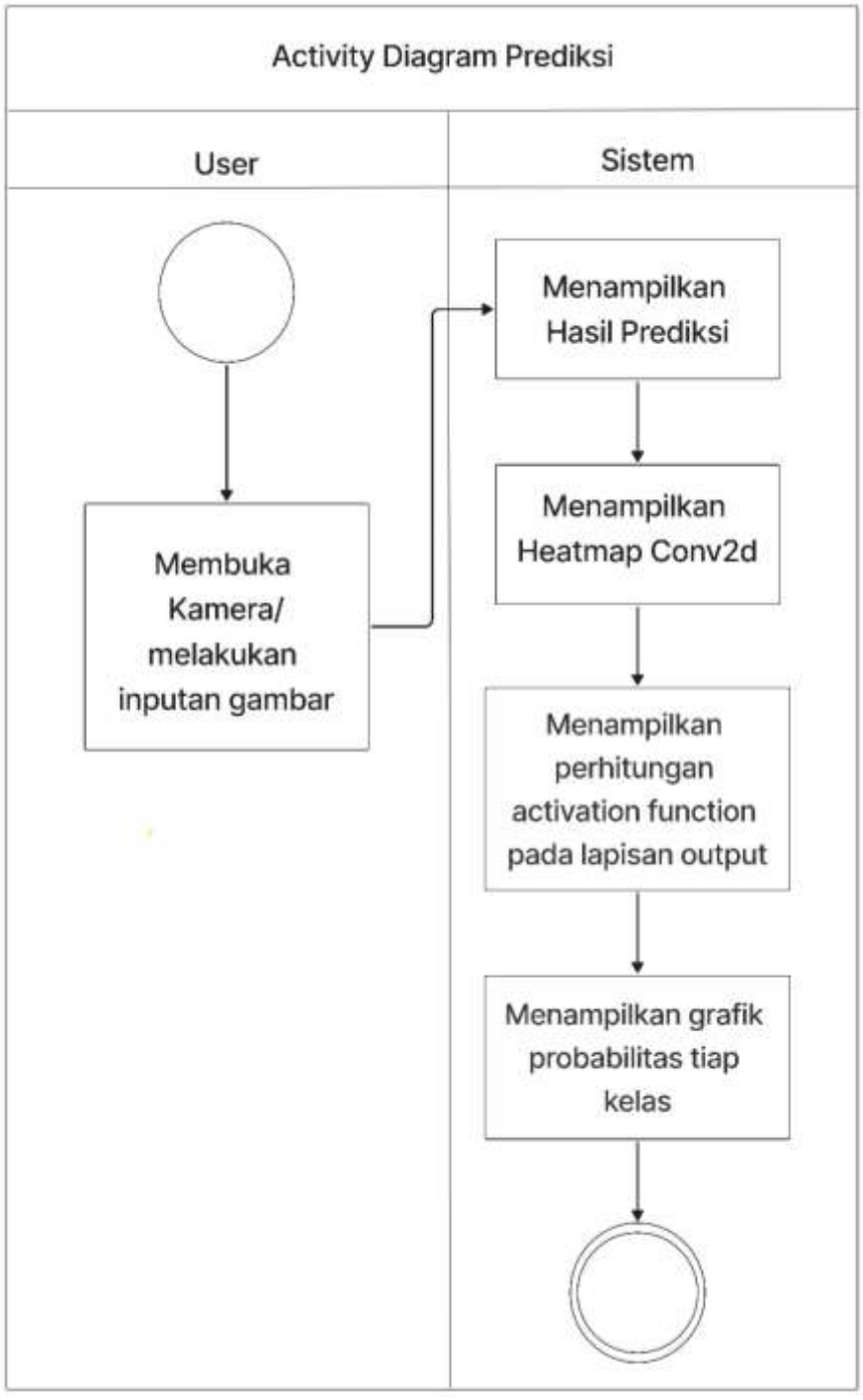
1.) *Activity diagram* halaman utama



Gambar 4. 3 *Activity diagram* halaman utama

Pada gambar 4.5 menggambarkan cara pengguna dapat mengakses halaman utama dan halaman prediksi, pengguna masuk dengan membuka halaman *website* melalui *browser*, sistem kemudian akan menampilkan halaman *website one page.*

3. *Activity diagram* prediksi



Gambar 4. 4 *Activity diagram* prediksi

Gambar *activity* diagram yang disajikan menggambarkan alur proses prediksi penyakit tanaman, dimulai dari interaksi

pengguna hingga keluaran yang dihasilkan oleh sistem. Diagram ini terdiri dari dua swimlane yang memisahkan tanggung jawab antara entitas pengguna dan sistem. Proses dimulai ketika pengguna membuka kamera atau memilih gambar untuk diinputkan ke dalam sistem, yang kemudian memicu aktivitas pada sisi sistem untuk melakukan prediksi.

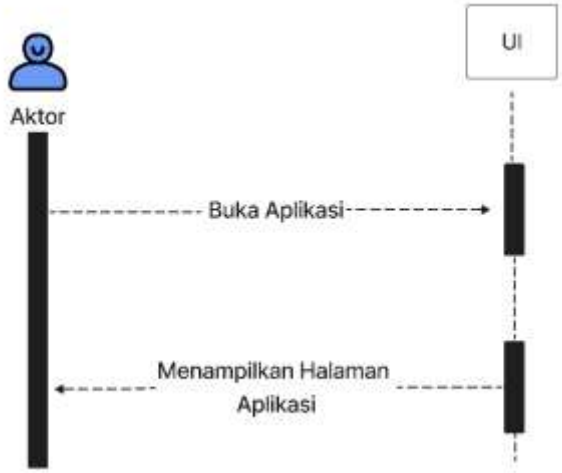
Setelah menerima input gambar, sistem akan memproses data dan menampilkan hasil prediksi berupa informasi mengenai kelas penyakit atau kondisi tanaman yang terdeteksi. Langkah berikutnya, sistem memberikan visualisasi dalam bentuk *feature map* dari lapisan Convolutional 2D (Conv2D), yang menunjukkan area pada gambar yang menjadi fokus utama model dalam membuat prediksi. Tahap ini bertujuan untuk memberikan pemahaman lebih mendalam terkait cara kerja model dalam menganalisis input gambar.

Sistem kemudian melanjutkan dengan menampilkan perhitungan activation function pada lapisan *output*, di mana nilai probabilitas untuk setiap kelas dihitung. Proses ini memberikan transparansi terhadap keyakinan model dalam menentukan hasil prediksi. Tahap akhir melibatkan penyajian grafik probabilitas untuk setiap kelas, yang memberikan representasi visual mengenai keyakinan model terhadap masing-masing kategori.

4. *Sequence Diagram*

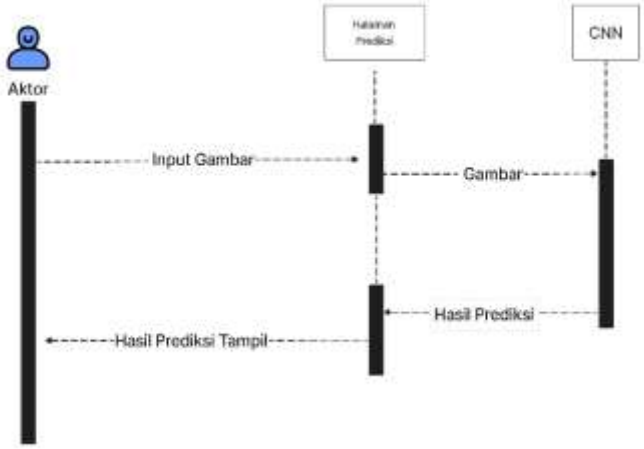
Sequence diagram menggambarkan interaksi antara objek dalam suatu sistem berdasarkan urutan waktu, membantu memahami alur komunikasi dalam sistem.

a. *Sequence diagram* membuka aplikasi



Gambar 4. 5 *Sequence diagram* membuka aplikasi

b. *Sequence diagram* deteksi



Gambar 4. 6 *Sequence diagram* deteksi

C. Implementasi *Tools* Pengembangan

Tools pengembangan yang digunakan dalam penelitian ini berperan dalam mendukung berbagai tahapan, mulai dari perancangan aplikasi, pengolahan data, hingga evaluasi hasil. Berikut ini dijelaskan implementasi *tools* yang digunakan serta kontribusi untuk mencapai tujuan penelitian.

1. *Matplotlib* & *Seaborn*

   Digunakan untuk melakukan visualisasi data dan analisis performa model selama proses pelatihan. Berikut adalah sampel penggunaan *Matplotlib* dan *Seaborn*

```
# syntax untuk menampilkan 16 sampel data
plt.figure(figsize=(15, 15))
for images, labels in dataset.take(1):
    for i in range(16):
        ax = plt.subplot(4, 4, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(f'{class_names[labels[i]]}')
        plt.axis("off")
plt.show()

# syntax untuk visualisasi grafik training dan validation

# Konversi history ke DataFrame
history_df = pd.DataFrame({
    'Epoch': range(1, len(history.history['loss']) + 1),
    'Training Accuracy': history.history['accuracy'],
    'Validation Accuracy': history.history['val_accuracy'],
    'Training Loss': history.history['loss'],
    'Validation Loss': history.history['val_loss']
})

# Pastikan Epoch dalam bentuk integer
history_df['Epoch'] = history_df['Epoch'].astype(int)

# Plot menggunakan DataFrame
plt.figure(figsize=(12, 6))

# Plot Accuracy
plt.subplot(1, 2, 1)
sns.lineplot(
    data=history_df.melt(id_vars='Epoch',
                         value_vars=['Training Accuracy', 'Validation Accuracy']),
    x='Epoch',
    y='value',
    hue='variable',
    style='variable',
```

```
    dashes=[(2,2), ()],
    palette="tab10"
)
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xticks(history_df['Epoch'])

# Plot Loss
plt.subplot(1, 2, 2)
sns.lineplot(
    data=history_df.melt(id_vars='Epoch',
                value_vars=['Training Loss', 'Validation Loss']),
    x='Epoch',
    y='value',
    hue='variable',
    style='variable',
    dashes=[(), (2,2)],
    palette="tab10"
)
plt.title('Training and Validation Loss')
plt.ylabel('Loss')
plt.xticks(history_df['Epoch'])

plt.tight_layout()
plt.show()
```

Pada sampel syntax diatas matplotlib dan seaborn digunakan untuk menampilkan 16 sampel data dan melakukan visualisasi grafik training dan validation dari Epoch.

2. *Scikit-Learn* (SkLearn)

   Digunakan untuk melakukan evaluasi model dengan metrik seperti Confusion Matrix, Precision, Recall, dan F1-score setelah proses pelatihan selesai. Berikut penggunaan *syntaxnya.*

```
# menampilkan confusion matrix
cm = confusion_matrix(true_classes, predicted_classes)
cm_display = ConfusionMatrixDisplay(cm, display_labels=class_names)

plt.figure(figsize=(8, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
        xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

#mengambil nilai Precision. Recall, dan F1-Score
precision = precision_score(true_classes, predicted_classes, average=None)
recall = recall_score(true_classes, predicted_classes, average=None)
f1 = f1_score(true_classes, predicted_classes, average=None)

metrics_df = pd.DataFrame({
```

```
    'Class': class_names,
    'Precision': precision,
    'Recall': recall,
    'F1-Score': f1
})
print(metrics_df)
```

Syntax tersebut digunakan untuk melakukan evaluasi *confusion matrix* dan mendapatkan nilai dari *recall, precision* dan *f1-score*

3. *MathJax*.

   Digunakan untuk menampilkan dan merender persamaan (Formula) matematika dalam format *LaTex* pada halaman aplikasi website.

```
// file client_side.js
 // menampilkan perhitungan Softmax dari inputan
    html += "<h4>Perhitungan Softmax:</h4>";
    const expLogits = logits.map((l) => Math.exp(l));
    const sumExp = expLogits.reduce((a, b) => a + b, 0);

    expLogits.forEach((exp, i) => {
      html += `
        <div class="softmax-step">
          <div>$$P(Kelas_{${i}}) = \\frac{e^{z_{${i}}}}{\\sum e^{z}}$$</div>
          <div>$$= \\frac{e^{${logits[i].toFixed(4)}}}{${expLogits
        .map((e) => `e^{${logits[expLogits.indexOf(e)].toFixed(4)}}`)
        .join(" + ")}}$$</div>
          <div>$$= \\frac{${exp.toFixed(4)}}{${sumExp.toFixed(
        4
      )}} = ${probabilities[i].toFixed(4)}$$</div>
        </div>`;
    });
    html += "</div></div>";

    $("#math-calculation").html(html);
    MathJax.Hub.Queue(["Typeset", MathJax.Hub]);
  }
```

Syntax diatas menampilkan formula dari perhitungan Softmax untuk mendapatkan nilai probabilitas deteksi tanaman

4. *jQuery* 3.6.0

   digunakan untuk menangani interaksi pengguna di sisi klien. Seperti AJAX dan manipulasi DOM.

```
#memastikan fungsi di dalamnya dijalankan setelah dokumen HTML selesai dimuat. Mengatur animasi dan interaksi navbar.
$(document).ready(function () {
  // -[Animasi Scroll]---------------------------
  $(".navbar a, footer a[href='#halamanku']").on("click", function (event) {
    if (this.hash !== "") {
```

```
    event.preventDefault();
    var hash = this.hash;
    $("html, body").animate(
      {
        scrollTop: $(hash).offset().top,
      },
      900,
      function () {
        window.location.hash = hash;
      }
    );
  }
});

$(window).scroll(function () {
  $(".slideanim").each(function () {
    var pos = $(this).offset().top;
    var winTop = $(window).scrollTop();
    if (pos < winTop + 600) {
      $(this).addClass("slide");
    }
  });
});
```

Sample JQuery diatas digunakan untuk memastikan halaman html telah dimuat dan memberikan animasi interaktif pada bagian navbar serta animasi scrool ketika pengguna menekan tombol menu.

5. *Python Flask*

Digunakan sebagai *backend* untuk menangani API dan antarmuka *web* pada aplikasi deteksi penyakit tanaman. Aplikasi ini mengatur unggahan gambar, memproses prediksi dengan model CNN, serta menampilkan hasilnya melalui rute utama (/) dan API (/api/deteksi).

```
# Konfigurasi Flask untuk Upload File
app = Flask(__name__, static_url_path='/static')
app.config['MAX_CONTENT_LENGTH'] = 6 * 1024 * 1024
app.config['UPLOAD_EXTENSIONS'] = ['.jpg', '.JPG', '.jpeg', '.png']
app.config['UPLOAD_PATH'] = './static/images/uploads/'

@app.route('/process', methods=['POST'])
def process_image():
    uploaded_file = request.files['file']
    if uploaded_file.filename != '':
        image = Image.open(uploaded_file.stream).convert('RGB')

        image = image.resize((256, 256))
        image_array = tf.keras.preprocessing.image.img_to_array(image)
        image_array = np.expand_dims(image_array, axis=0) / 255.0


        conv_layer_outputs = [layer.output for layer in model.layers if 'conv2d' in layer.name]
        activation_model = tf.keras.models.Model(inputs=model.input, outputs=conv_layer_outputs)
        activations = activation_model.predict(image_array)
```

```
        layer_names = [layer.name for layer in model.layers if 'conv2d' in layer.name]
        num_layers = len(layer_names)

        fig, axes = plt.subplots(1, num_layers, figsize=(15 * num_layers, 15))
        if num_layers == 1:
            axes = [axes]

        for i, (activation, layer_name) in enumerate(zip(activations, layer_names)):
            ax = axes[i]
            mean_activation = np.mean(activation[0], axis=-1)
            ax.imshow(mean_activation, cmap='viridis')
            ax.set_title(layer_name, fontsize=10)
            ax.axis('off')


        buf = io.BytesIO()
        plt.tight_layout()
        plt.savefig(buf, format='png')
        buf.seek(0)
        encoded_image = base64.b64encode(buf.read()).decode('utf-8')
        buf.close()
        plt.close()


        return jsonify({"visualization": encoded_image})

    return jsonify({"error": "No file uploaded"})
```

Pada sampel *flask,* penggunaan *flask* untuk mengatur unggahan gambar dan rute *process* gambar serta menghasilkan visualisasi *layers*.

6. *Werkzeug*

   Digunakan untuk menangani file upload pada aplikasi berbasis Flask.

```
from werkzeug.utils import secure_filename

filename = secure_filename(uploaded_file.filename)
```

   mengamankan nama file yang diunggah oleh pengguna agar tidak menyebabkan konflik saat disimpan di server.

7. *Tensorflow Keras*

   TensorFlow Keras digunakan sebagai framework utama untuk membangun model *deep learning*. *TensorFlow* menyediakan berbagai fungsi yang mempermudah proses pelatihan model, sedangkan *Keras* sebagai API tingkat tinggi memungkinkan pembuatan arsitektur *Convolutional Neural Network (CNN)* dengan lebih sederhana.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Activation

def make_model(n_classes):
    model = Sequential()

    # Conv Block 1
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    # Conv Block 2
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    # Conv Block 3
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    # Conv Block 4
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    # Fully Connected
    model.add(Flatten())
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu'))

    # Output dengan nama 'logits'
    model.add(Dense(n_classes, name='logits'))
    model.add(Activation('softmax'))

    return model
```

Sintaks di atas adalah arsitektur *Convolutional Neural Network (CNN)* yang dibangun menggunakan *TensorFlow Keras* untuk tugas klasifikasi gambar.

D. Teknik Pengumpulan Data

Teknik pengumpulan data dalam penelitian ini dilakukan melalui dua metode utama, yaitu pemanfaatan *dataset* publik dan pengambilan gambar secara mandiri. *Dataset* publik digunakan sebagai sumber utama karena telah memiliki label yang jelas dan mencakup berbagai kondisi pencahayaan serta sudut pengambilan gambar, sehingga membantu meningkatkan kemampuan model dalam mengenali variasi gambar.

1. *Dataset* Publik (*PlantVillage Dataset* - Kaggle)

PlantVillage Dataset dari Kaggle digunakan sebagai sumber utama dalam pelati-

han model CNN. *Dataset* ini berisi berbagai gambar tanaman dengan kondisi sehat dan terinfeksi penyakit, yang telah diberi label dengan jelas. Keunggulan *dataset* ini adalah kualitas gambar yang konsisten serta keberagaman jenis penyakit yang diklasifikasikan, sehingga sangat mendukung proses pelatihan model dalam mengenali pola visual penyakit tanaman cabai dan tomat. Sebelum digunakan, *dataset* ini melalui tahap pra-pemrosesan, termasuk pengecekan kualitas gambar, penyesuaian ukuran gambar, serta pengelompokan ulang ke dalam enam kelas utama, yaitu Cabai Antraknosa, Cabai Normal, Cabai *Leaf Curl*, Tomat Normal, Tomat *Early Blight*, dan Tomat *Late Blight*.

2. *Dataset* Mandiri

Selain menggunakan *dataset* publik, penelitian ini juga mengumpulkan gambar secara mandiri untuk menambah keberagaman data dan menguji performa model dalam kondisi dunia nyata. Gambar tanaman cabai dan tomat diambil menggunakan kamera beresolusi tinggi dalam berbagai kondisi pencahayaan, sudut pengambilan, dan lingkungan yang berbeda. Tujuan dari penggunaan *dataset* mandiri adalah untuk mengukur seberapa baik model dapat mendeteksi penyakit pada gambar yang tidak berasal dari *dataset* pelatihan, serta mengidentifikasi potensi tantangan seperti perbedaan pencahayaan, kualitas gambar, dan variasi latar belakang.

E. *Explore Dataset*

Dataset yang digunakan dalam penelitian ini berjumlah 3.734 data terbagi menjadi 6 (enam) kelas dengan data yang terdiri dari citra daun tomat dan buah

cabai dengan kondisi normal dan kondisi terserang penyakit, dengan distribusi data sebagai berikut :

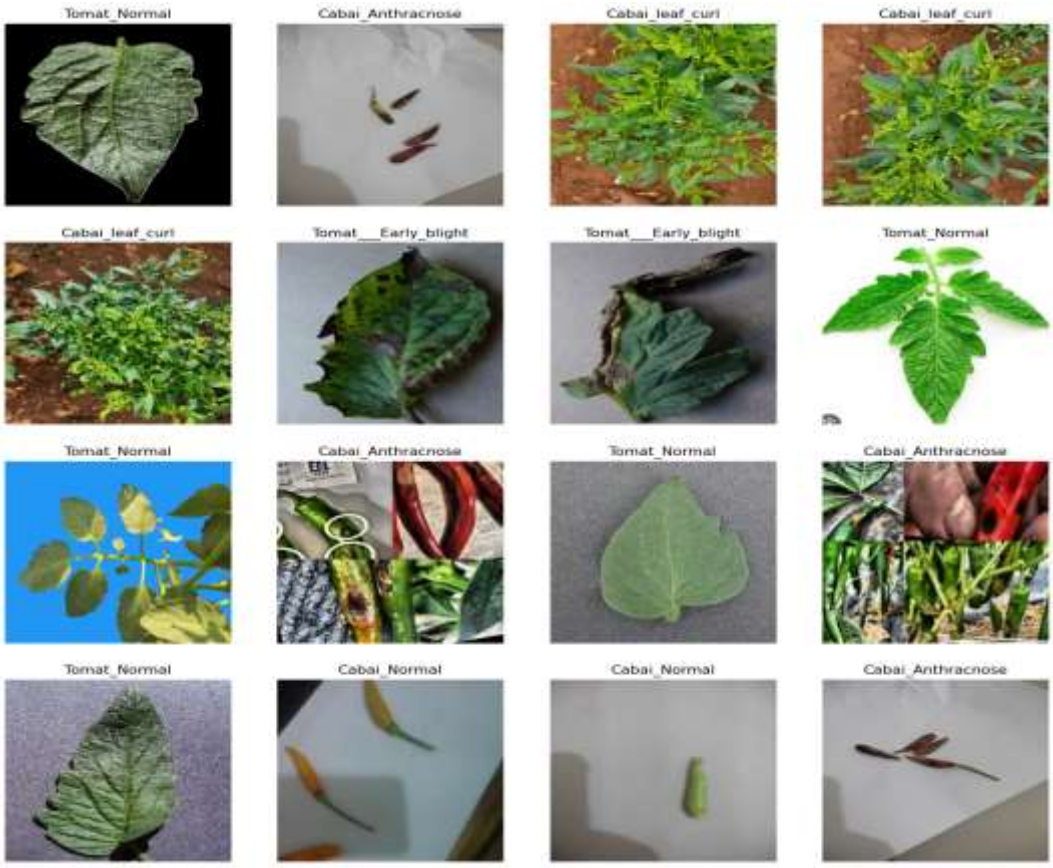
Tabel 4. 2 Distribusi *dataset*

Kelas	Jumlah Data
Cabai Normal	600
Cabai Antraknosa / Patek buah	620
Cabai *leaf curl*	621
Tomat Normal	670
Tomat *Early Blight*	612
Tomat *Late Blight*	611

Visualisasi *sample* data dari tiap kelas disajikan pada Gambar 4.1. visualisasi *sample* data ditujukan untuk memberikan gambaran tentang karakteristik dari masing-masing kelas penyakit dan kondisi normal tanaman.

Dari visualisasi Terlihat bahwa :

a. Daun tomat yang terinfeksi *early blight* dan *late blight* memiliki *spot* berwarna coklat kehitaman dangan pola penyebaran yang khas.
b. Cabai antraknosa yang terlihat memiliki bercak hitam dengan buah yang mengkerut atau terlihat bagian pembusukan pada buah.
c. Daun cabai keriting yang menampilkan karakteristik pada daun dengan ciri daun berbentuk keriting dengan warna sedikit kekuningan.



Gambar 4. 7 Visualisasi *sample* data

Dataset tersebut dibagi menjadi 3 (tiga) bagian yakni *Train Dataset*, *Validation Dataset*, dan *Test Dataset* dengan rasio 70:15:15 dan juga dilakukan augmentasi data.

F. Arsitektur Model dan Parameter

Arsitektur *Convolutional Neural Network (CNN)* pada sistem pendeteksian penyakit pada tanaman cabai dan tomat menggunakan arsitektur *Custom* atau *Plain Model* yang dibangun secara *Sequential*. Berbeda dengan *pre-trained models* seperti VGG-Net, ResNet, YOLO, dan lainnya, arsitektur *Custom* memberikan kebebasan lebih pada proses eksperimen dalam mengatur lapisan-lapisan yang

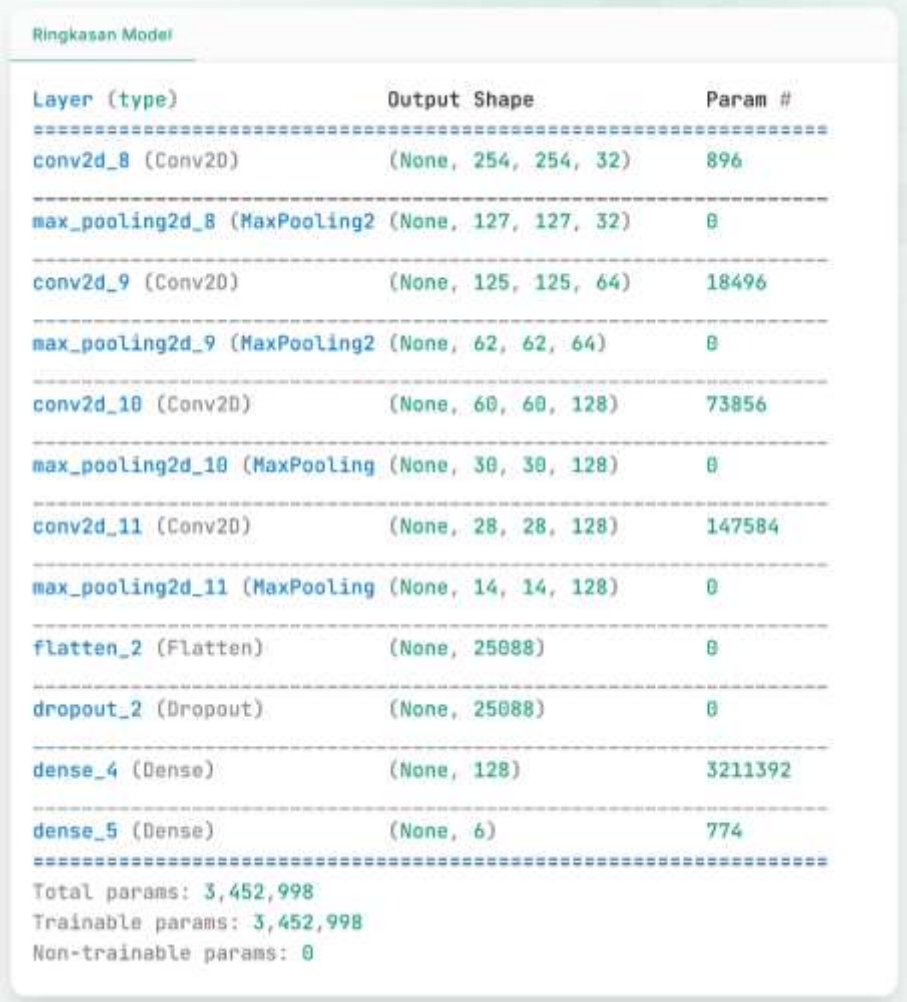
digunakan, seperti jumlah lapisan, *filter*, *kernel*, teknik regulasi maupun *activation function* yang digunakan.

Arsitektur Model

```
model = Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(IMAGE_SIZE, IMAGE_SIZE, CHANNELS),),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(64, (3, 3), activation='relu',),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(128, (3, 3), activation='relu',),
    layers.MaxPooling2D(2, 2),

    layers.Conv2D(128, (3, 3), activation='relu',),
    layers.MaxPooling2D(2, 2),

    layers.Flatten(),
    layers.Dropout(0.5),

    layers.Dense(128, activation='relu',),
    layers.Dense(n_classes, activation='softmax')
])
```

Gambar 4. 8 Arsitektur Model yang digunakan

Model ini terdiri dari empat lapisan konvolusi dengan *kernel* berukuran 3×3, masing-masing menggunakan fungsi aktivasi ReLU, di-ikuti oleh lapisan *MaxPooling* (2×2) untuk mengurangi dimensi fitur. Setelah melalui proses ekstraksi fitur, data kemudian diratakan menggunakan *Flatten layer*, diikuti oleh *Dropout* (0.5) untuk mengurangi risiko *overfitting*. Model ini juga memiliki *Dense layer* dengan 128 *neuron* dan fungsi aktivasi ReLU sebelum akhirnya masuk ke lapisan *output* yang menggunakan *Softmax activation* untuk mengklasifikasikan gambar ke dalam beberapa kategori penyakit tanaman.



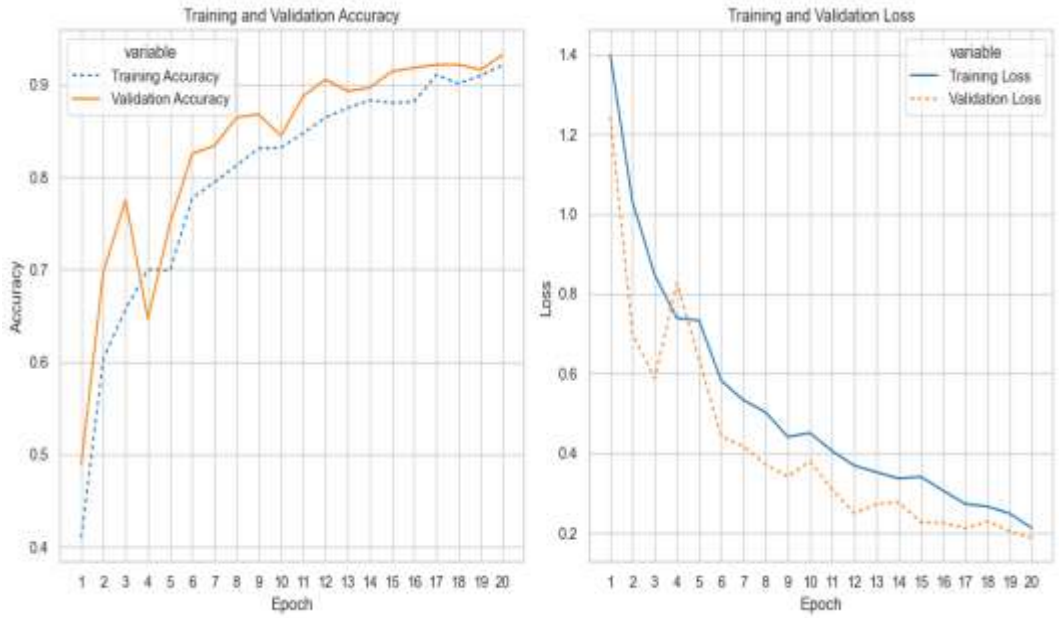
Gambar 4. 9 *Model Summary*

Summary model ini menunjukkan bahwa total parameter yang digunakan adalah 3.452.998, di mana seluruhnya dapat dilatih tanpa adanya parameter *non-trainable.* Lapisan konvolusi pertama memiliki 896 parameter, sedangkan jumlah parameter bertambah pada lapisan konvolusi berikutnya seiring dengan peningkatan jumlah *filter*, mencapai 147.584 pada lapisan konvolusi keempat. Setelah melewati proses ekstraksi fitur, lapisan *Dense* pertama memiliki 3.211.392 parameter, yang merupakan jumlah terbesar dalam model ini, sementara lapisan *output* dengan enam unit hanya memiliki 774 parameter. Struktur ini menunjukkan

bahwa sebagian besar parameter berada pada lapisan *fully connected*, yang berperan dalam klasifikasi akhir berdasarkan fitur yang telah diekstraksi oleh lapisan konvolusi.

G. Evaluasi Performa Model

1. *Plotting Accuracy* dan *Loss*



Gambar 4. 10 Grafik *Accuracy dan Loss*

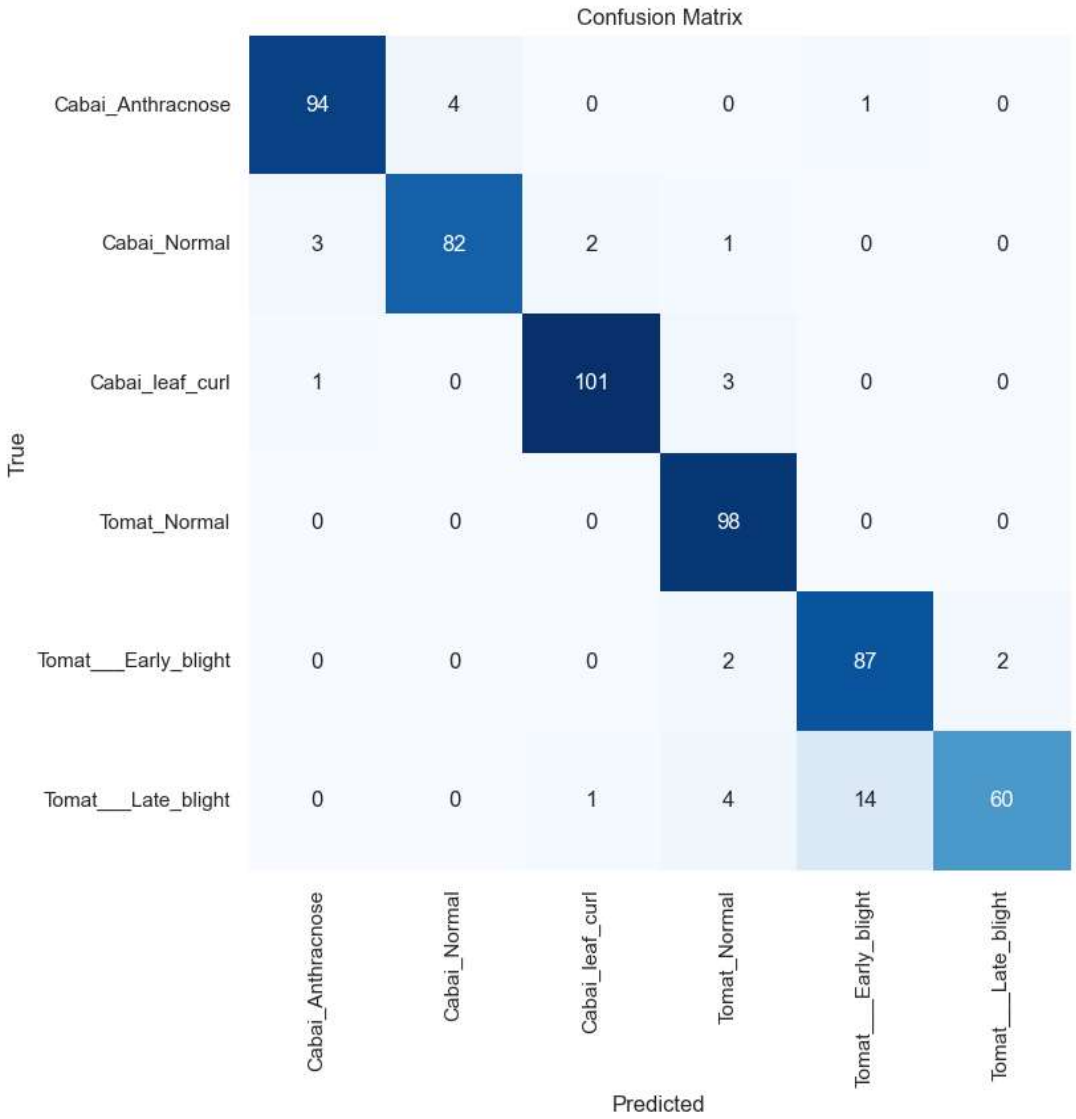
Grafik akurasi menunjukkan tren positif di mana akurasi untuk data *training* dan validasi meningkat seiring bertambahnya *epoch.* Peningkatan ini mencerminkan bahwa model mampu mempelajari pola dalam data secara efektif. Pada tahap akhir *training*, terutama pada *epoch* ke-18 hingga ke-20, akurasi *training* dan validasi berada pada tingkat yang hampir sama. Hal ini mengindikasikan bahwa model telah mencapai titik konvergensi, di mana kemampuan model untuk mempelajari data *training* dan generalisasi terhadap

data validasi berada dalam keseimbangan. Menariknya, pada beberapa epoch awal, akurasi validasi terlihat lebih tinggi dibandingkan akurasi *training*. Fenomena ini menunjukkan bahwa model mampu melakukan generalisasi dengan baik pada data baru.

Grafik *loss* memperlihatkan penurunan nilai *loss* untuk data *training* dan validasi seiring bertambahnya *epoch*. Penurunan ini menunjukkan bahwa model berhasil meminimalkan *error* dengan baik selama proses *training*. Pada sebagian besar *epoch*, *loss* validasi sedikit lebih tinggi dibandingkan *loss training*, yang merupakan hal umum karena model diuji pada data yang belum pernah dilihat sebelumnya. Selain itu, pada *epoch-epoch* akhir, nilai *loss* validasi cenderung stabil dan tidak mengalami peningkatan yang signifikan. Hal ini menunjukkan bahwa model tidak belajar noise pada data, sehingga *overfitting* dapat dihindari.

2. *Confusion Matrix*

Confusion Matrix dilakukan untuk melihat dan melakukan perhitungan performa model klasifikasi berdasarkan hasil prediksi yang dibandingkan dengan label sebenarnya. Matriks ini memungkinkan analisis mendalam terhadap kesalahan prediksi, sehingga dapat mengidentifikasi kelemahan model dalam mengenali kelas tertentu.



Gambar 4. 11 *Confusion Matrix* Model

Nilai pada diagonal utama (dari kiri atas ke kanan bawah) menunjukkan jumlah prediksi yang benar untuk setiap kelas.

Tabel 4. 3 Prediksi benar pada *test dataset*

Kelas Aktual	Prediksi Benar
Cabai Antraknosa	94
Cabai Normal	82
Cabai *Leaf Curl*	101

Lanjutan Tabel 4.3

Kelas Aktual	Prediksi Benar
Tomat Normal	98
Tomat *Early Blight*	87
Tomat *Late Blight*	60

Nilai yang berada diluar diagonal utama menunjukkan jumlah kesalahan prediksi yang salah untuk setiap kelas, dapat dilihat pada tabel berikut :

Tabel 4. 4 Kesalahan deteksi pada *test dataset*

Kelas Aktual	Kesalahan Kelas deteksi	Jumlah Kesalahan	Keterangan
Cabai Antraknosa	Cabai Normal	4	Beberapa sampel salah diklasifikasikan sebagai Cabai_Normal.
Cabai Antraknosa	Tomat *Early Blight*	1	Satu sampel salah diklasifikasikan sebagai Tomat *Early Blight*
Cabai Normal	Cabai Antraknosa	3	Sebagian kecil prediksi salah diarahkan ke Cabai_*Anthracnose*.
Cabai Normal	Cabai *Leaf Curl*	2	Sebagian kecil kesalahan diarahkan ke Cabai *leaf curl*.
Cabai Normal	Tomat Normal	1	Satu kesalahan deteksi diarahkan ke Tomat Normal
Cabai *Leaf Curl*	Cabai Antraknosa	1	Kesalahan kecil kelas diarahkan ke Cabai *Anthracnose*
Cabai *Leaf Curl*	Tomat Normal	3	Beberapa kesalahan diarahkan ke Tomat Normal
Tomat *Early Blight*	Tomat Normal	2	kesalahan ke kelas Tomat Normal yang berbeda.
Tomat *Early Blight*	Tomat *Late Blight*	2	Kesalahan cukup besar ke kelas yang mirip secara visual.
Tomat *Late Blight*	Cabai *Leaf Curl*	1	Kesalahan diarahkan ke kelas Cabai *Leaf Curl*

Lanjutan Tabel 4.4 *Kesalahan deteksi pada test dataset*

Kelas Aktual	Kesalahan Kelas Deteksi	Jumlah Kesalahan	Keterangan
Tomat *Late Blight*	Tomat Normal	4	Sebagian kecil Kesalahan diarahkan ke Tomat Normal
Tomat *Late Blight*	Tomat *Early Blight*	14	Kesalahan terbesar antara dua kelas yang mirip secara visual.

Kesalahan klasifikasi yang terjadi menunjukkan bahwa model mengalami kesulitan dalam membedakan beberapa kelas, terutama yang memiliki kemiripan visual. Pada tanaman cabai, kesalahan klasifikasi paling banyak terjadi pada Cabai Normal, di mana 3 sampel salah diklasifikasikan sebagai Cabai Antraknosa, 2 sampel sebagai Cabai *Leaf Curl*, dan 1 sampel sebagai Tomat Normal. Hal ini menunjukkan bahwa model mengalami kesulitan dalam membedakan kondisi cabai yang sehat dan yang mengalami penyakit tertentu, terutama yang memiliki gejala serupa seperti perubahan warna dan tekstur daun. Selain itu, Cabai Antraknosa juga mengalami beberapa kesalahan, dengan 4 sampel salah diklasifikasikan sebagai Cabai Normal dan 1 sampel salah terdeteksi sebagai Tomat *Early Blight*, yang menandakan adanya fitur visual yang tumpang tindih antara kelas tersebut. Cabai *Leaf Curl* juga mengalami kesalahan klasifikasi, di mana 1 sampel salah diklasifikasikan sebagai Cabai Antraknosa dan 3 sampel salah dikenali sebagai Tomat Normal, yang menunjukkan bahwa model kurang mampu membedakan ciri khas dari kondisi daun yang menggulung dengan kondisi normal tanaman tomat.

Sementara itu, pada tanaman tomat, kesalahan terbesar terjadi antara Tomat *Early Blight* dan Tomat *Late Blight*, di mana masing-masing memiliki kesalahan silang dengan 2 sampel salah diklasifikasikan ke kelas yang berlawanan. Hal ini

menunjukkan bahwa gejala visual kedua penyakit ini cukup mirip, seperti bercak dan perubahan warna daun, yang membuat model kesulitan dalam membedakannya. Selain itu, Tomat *Early Blight* juga memiliki 2 kesalahan klasifikasi ke Tomat Normal, yang menunjukkan bahwa beberapa sampel mungkin memiliki gejala ringan sehingga terdeteksi sebagai kondisi sehat. Pada Tomat *Late Blight*, terdapat 1 sampel yang salah diklasifikasikan sebagai Cabai *Leaf Curl*, yang bisa terjadi akibat pola visual tertentu yang menyerupai gejala penyakit pada cabai. Selain itu, Tomat *Late Blight* juga memiliki kesalahan yang cukup signifikan, dengan 4 sampel salah diklasifikasikan sebagai Tomat Normal dan 14 sampel salah dikenali sebagai Tomat *Early Blight*, yang memperkuat indikasi bahwa model mengalami kesulitan dalam membedakan penyakit yang memiliki karakteristik visual serupa.

Kesalahan klasifikasi ini mengindikasikan bahwa model perlu ditingkatkan agar lebih akurat dalam mendeteksi perbedaan antar kelas yang memiliki kemiripan visual. Beberapa langkah perbaikan yang dapat dilakukan antara lain meningkatkan jumlah dan kualitas data pelatihan, menerapkan teknik augmentasi yang lebih beragam untuk memperjelas perbedaan antar kelas

Data yang diperoleh dari *Confusion Matrix* digunakan untuk mendapatkan nilai dari *Precission, Recall*, dan *F1-Score* dari tiap kelas, sebagai berikut :

a. Kelas Cabai Antraknosa

$$Precision = \frac{TP}{TP+FP} = \frac{94}{94+(3+1+0+0+0)} = \frac{94}{98} = 0.9591$$

$$Recall = \frac{TP}{TP+FN} = \frac{94}{94+(4+0+0+1+0)} = \frac{94}{99} = 0.9494$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = 2 \cdot \frac{0.9591 \cdot 0.9494}{0.9591 + 0.9494} = 0.9543$$

b. Kelas Cabai Normal

$$Precision = \frac{TP}{TP+FP} = \frac{82}{82+(4+0+0+0+0)} = \frac{82}{86} = 0.9534$$

$$Recall = \frac{TP}{TP+FN} = \frac{82}{82+(3+2+1+0)} = \frac{82}{88} = 0.9318$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = 2 \cdot \frac{0.9534 \cdot 0.9318}{0.9534 + 0.9318} = 0.9425$$

c. Kelas Cabai *Leaf Curl*

$$Precision = \frac{TP}{TP+FP} = \frac{101}{101+(0+2+0+0+1)} = \frac{101}{104} = 0.9711$$

$$Recall = \frac{TP}{TP+FN} = \frac{101}{101+(1+0+3+0+0)} = \frac{101}{105} = 0.9619$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = 2 \cdot \frac{0.9711 \cdot 0.9619}{0.9711 + 0.9619} = 0.9665$$

d. Kelas Tomat Normal

$$Precision = \frac{TP}{TP+FP} = \frac{98}{98+(0+1+3+2+4)} = \frac{98}{108} = 0.9074$$

$$Recall = \frac{TP}{TP+FN} = \frac{98}{98+(0+0+0+0+0)} = \frac{98}{98} = 0.95600$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = 2 \cdot \frac{0.9074 \cdot 0.95600}{0.9074 + 0.95600} = 0.9514$$

e. Kelas Tomat *Early Blight*

$$Precision = \frac{TP}{TP+FP} = \frac{87}{87+(1+0+0+0+14)} = \frac{87}{102} = 0.8529$$

$$Recall = \frac{TP}{TP+FN} = \frac{87}{87(0+0+0+2+2)} = \frac{87}{91} = 0.9560$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = 2 \cdot \frac{0.8529 \cdot 0.9560}{0.8529 + 0.9560} = 0.9015$$

f. Kelas *Tomat Late Blight*

$$Precision = \frac{TP}{TP+FP} = \frac{60}{60+(0+0+0+0+2)} = \frac{60}{62} = 0.9677$$

$$Recall = \frac{TP}{TP+FN} = \frac{60}{60+(0+0+1+4+14)} = \frac{60}{79} = 0.7594$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall} = 2 \cdot \frac{0.9677 \cdot 0.7594}{0.9677+0.7594} = 0.8510$$

Tabel 4. 5 *Precision, Recall,* dan F1-Score

Kelas	*Precision*	*Recall*	F1 - Score
Cabai Antraknosa	0.9591	0.9494	0.9543
Cabai Normal	0.9534	0.9318	0.9425
Cabai *Leaf Curl*	0.9711	0.9619	0.9665
Tomat Normal	0.9074	0.95600	0.9514
Tomat *Early Blight*	0.8529	0.9560	0.9015
Tomat *Late Blight*	0.9677	0.7594	0.8510

Berdasarkan evaluasi *Precision, Recall,* dan *F1-Score,* model menunjukkan performa yang cukup baik dalam mendeteksi penyakit pada cabai dan tomat, meskipun terdapat beberapa kelas yang perlu ditingkatkan.

Pada Cabai Antraknosa, model memiliki keseimbangan performa dengan *Precision* 0.9591 dan *Recall* 0.9494, menghasilkan *F1-Score* 0.9543. Hal ini menunjukkan bahwa model dapat mendeteksi sebagian besar sampel dengan akurasi tinggi dan hanya sedikit kesalahan dalam prediksi. Cabai Normal memiliki *F1-Score* terendah (0.9425) dibandingkan kelas lainnya, dengan *Recall* lebih rendah (0.9318), menunjukkan adanya kesulitan dalam membedakan kelas ini dari yang lain, kemungkinan karena kemiripan fitur atau distribusi data yang kurang merata.

Untuk Cabai *Leaf Curl*, model bekerja sangat baik dengan *Precision* 0.9711 dan *Recall* 0.9619, menghasilkan *F1-Score* 0.9665. Hal ini menunjukkan bahwa model mampu mengenali kelas ini dengan baik dan memiliki sedikit kesalahan prediksi.

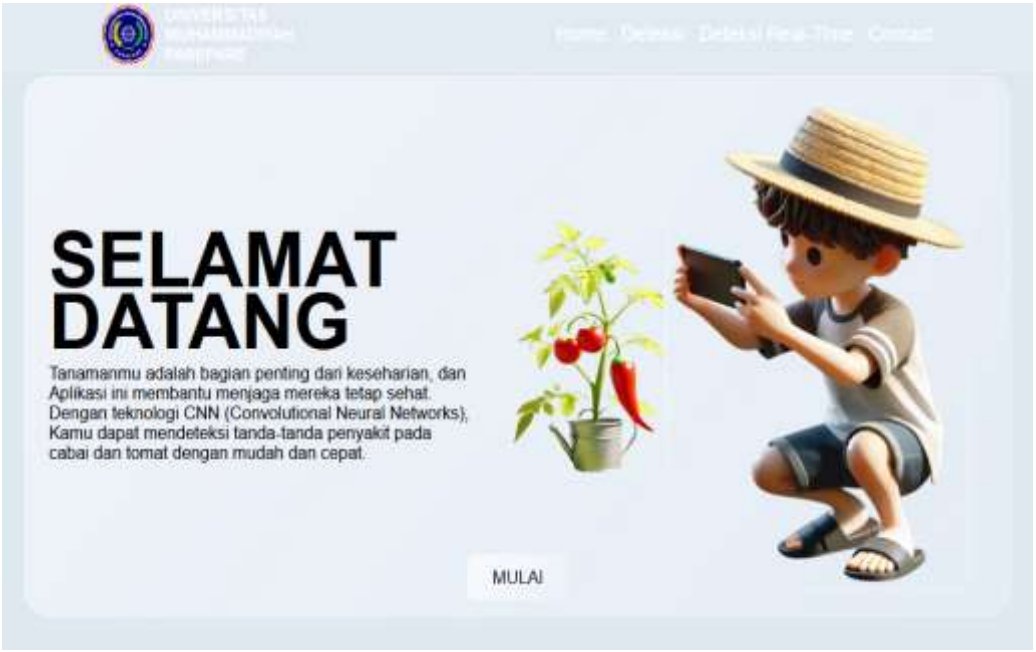
Pada Tomat Normal, model memiliki *Recall* tinggi (0.9560), menunjukkan bahwa hampir semua sampel terdeteksi dengan benar, namun *Precision* yang lebih rendah (0.9074) mengindikasikan adanya *False Positives*. Dengan *F1-Score* 0.9514, performanya cukup baik tetapi masih dapat ditingkatkan.

Kelas Tomat *Early Blight* memiliki *Recall* tinggi (0.9560), menandakan model mampu mengenali sebagian besar sampel, tetapi *Precision* 0.8529 menunjukkan bahwa masih ada kesalahan dalam klasifikasi. *F1-Score* sebesar 0.9015 mencerminkan bahwa model cukup baik, namun masih bisa diperbaiki untuk mengurangi kesalahan prediksi.

Sementara itu, Tomat *Late Blight* memiliki *Precision* tertinggi (0.9677), tetapi *Recall* yang rendah (0.7594) menunjukkan bahwa model masih melewatkan sejumlah besar sampel yang sebenarnya termasuk dalam kelas ini. *F1-Score* sebesar 0.8510 menandakan adanya ketidakseimbangan, di mana model sangat yakin dalam prediksinya tetapi kurang sensitif dalam mendeteksi semua kasus yang ada.

H. Tampilan Aplikasi

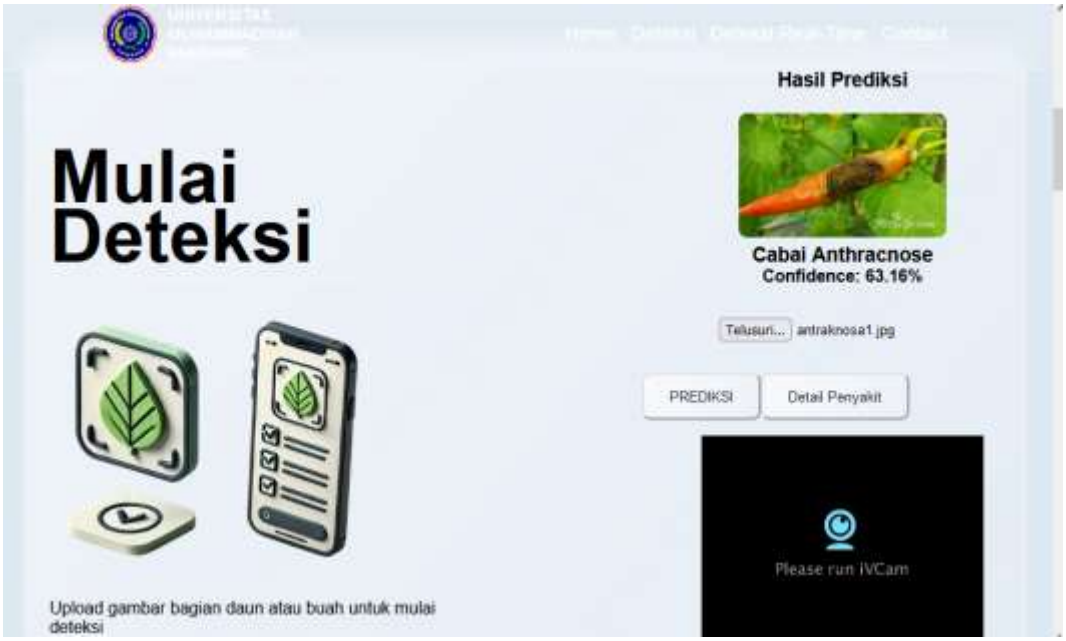
1. Tampilan Utama



Gambar 4. 12 Tampilan Utama

Gambar 4.11 merupakan tampilan awal ketika pengguna membuka website. Bagian utama halaman menyajikan pesan sambutan dengan judul "SELAMAT DATANG" yang ditulis dengan huruf tebal untuk menarik perhatian. Teks di bawahnya menjelaskan tujuan aplikasi, yaitu membantu pengguna mendeteksi tanda-tanda penyakit pada cabai dan tomat menggunakan teknologi *Convolutional Neural Networks* (CNN). Di sebelah kanan, terdapat ilustrasi seorang anak dengan gaya kartun, yang sedang memotret tanaman cabai dan tomat, menggambarkan cara kerja aplikasi secara visual.

2. Tampilan deteksi



Gambar 4. 13 Tampilan Deteksi

Gambar ini menampilkan halaman *Deteksi* dari aplikasi web pendeteksi penyakit tanaman cabai dan tomat. Halaman ini dirancang untuk memungkinkan pengguna mengunggah gambar tanaman yang ingin diperiksa dan mendapatkan hasil prediksi berdasarkan model *Convolutional Neural Networks* (CNN). Di bagian kanan, terdapat Hasil Prediksi, yang menampilkan gambar tanaman yang telah diunggah, nama penyakit yang terdeteksi

ksi (dalam hal ini *Cabai Anthracnose*), serta tingkat kepercayaan (*confidence score*) sebesar 63.16%. Ini menunjukkan seberapa yakin model terhadap hasil prediksi. Pengguna dapat mengunggah gambar melalui tombol "Telusuri..." dan kemudian menjalankan deteksi menggunakan tombol "PREDIKSI". Setelah hasil muncul, tombol "Detail Penyakit" memungkinkan pengguna mendapatkan informasi lebih lanjut mengenai penyakit yang terdeteksi dan cara mengatasinya.Di

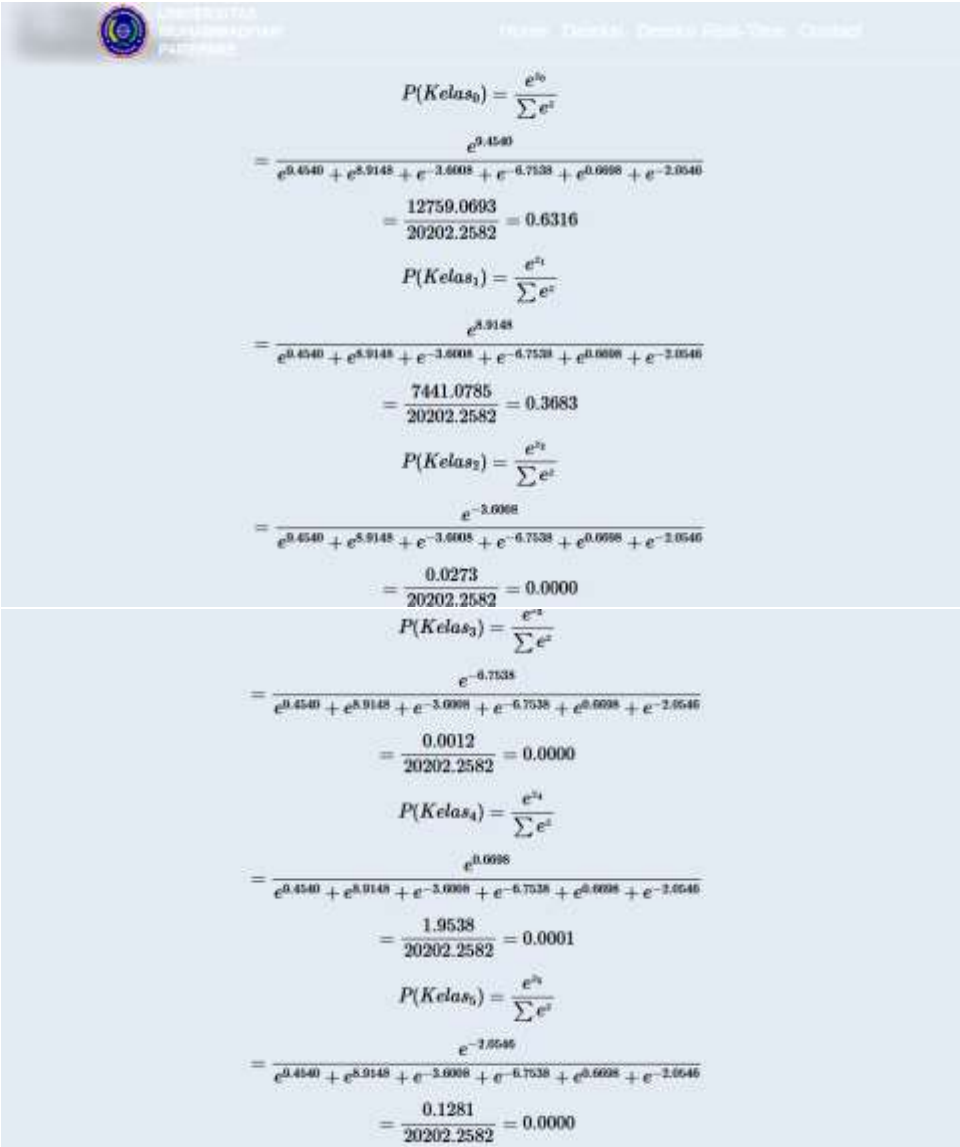
bagian bawah kanan, terdapat tampilan kamera untuk mengambil gambar penyakit secara langsung menggunakan kamera.

3. Tampilan Visualisasi Conv2D

Gambar 4. 14 Visualissasi Conv2d

Gambar ini menampilkan halaman *Visualisasi Model* pada aplikasi web pendeteksi penyakit tanaman cabai dan tomat. Halaman ini berfungsi untuk menampilkan proses analisis yang dilakukan oleh model *Convolutional Neural Networks* (CNN) dengan memvisualisasikan bagaimana model mengekstraksi fitur dari gambar yang diunggah. Bagian utama halaman ini berisi *feature map* hasil aktivasi dari beberapa lapisan *Conv2D. Feature map* ini menunjukkan area-area dalam gambar yang dianggap penting oleh model dalam membuat prediksi. Setiap gambar dalam visualisasi ini merepresentasikan tahap berbeda dari ekstraksi fitur, mulai dari fitur sederhana seperti tepi dan tekstur hingga pola yang lebih kompleks yang membantu model mengenali penyakit pada tanaman. Dengan adanya visualisasi ini, pengguna dapat memahami bagaimana model bekerja dalam mendeteksi penyakit serta mengetahui bagian mana dari gambar yang berkontribusi paling besar dalam pengambilan keputusan model.

4. Tampilan *Output* perhitungan *probabilitas*



$$P(Kelas_0) = \frac{e^{z_0}}{\sum e^z}$$

$$= \frac{e^{0.4540}}{e^{0.4540} + e^{5.9148} + e^{-3.6008} + e^{-6.7538} + e^{0.6098} + e^{-2.0546}}$$

$$= \frac{12759.0693}{20202.2582} = 0.6316$$

$$P(Kelas_1) = \frac{e^{z_1}}{\sum e^z}$$

$$= \frac{e^{5.9148}}{e^{0.4540} + e^{5.9148} + e^{-3.6008} + e^{-6.7538} + e^{0.6098} + e^{-2.0546}}$$

$$= \frac{7441.0785}{20202.2582} = 0.3683$$

$$P(Kelas_2) = \frac{e^{z_2}}{\sum e^z}$$

$$= \frac{e^{-3.6008}}{e^{0.4540} + e^{5.9148} + e^{-3.6008} + e^{-6.7538} + e^{0.6098} + e^{-2.0546}}$$

$$= \frac{0.0273}{20202.2582} = 0.0000$$

$$P(Kelas_3) = \frac{e^{z_3}}{\sum e^z}$$

$$= \frac{e^{-6.7538}}{e^{0.4540} + e^{5.9148} + e^{-3.6008} + e^{-6.7538} + e^{0.6098} + e^{-2.0546}}$$

$$= \frac{0.0012}{20202.2582} = 0.0000$$

$$P(Kelas_4) = \frac{e^{z_4}}{\sum e^z}$$

$$= \frac{e^{0.6098}}{e^{0.4540} + e^{5.9148} + e^{-3.6008} + e^{-6.7538} + e^{0.6098} + e^{-2.0546}}$$

$$= \frac{1.9538}{20202.2582} = 0.0001$$

$$P(Kelas_5) = \frac{e^{z_5}}{\sum e^z}$$

$$= \frac{e^{-2.0546}}{e^{0.4540} + e^{5.9148} + e^{-3.6008} + e^{-6.7538} + e^{0.6098} + e^{-2.0546}}$$

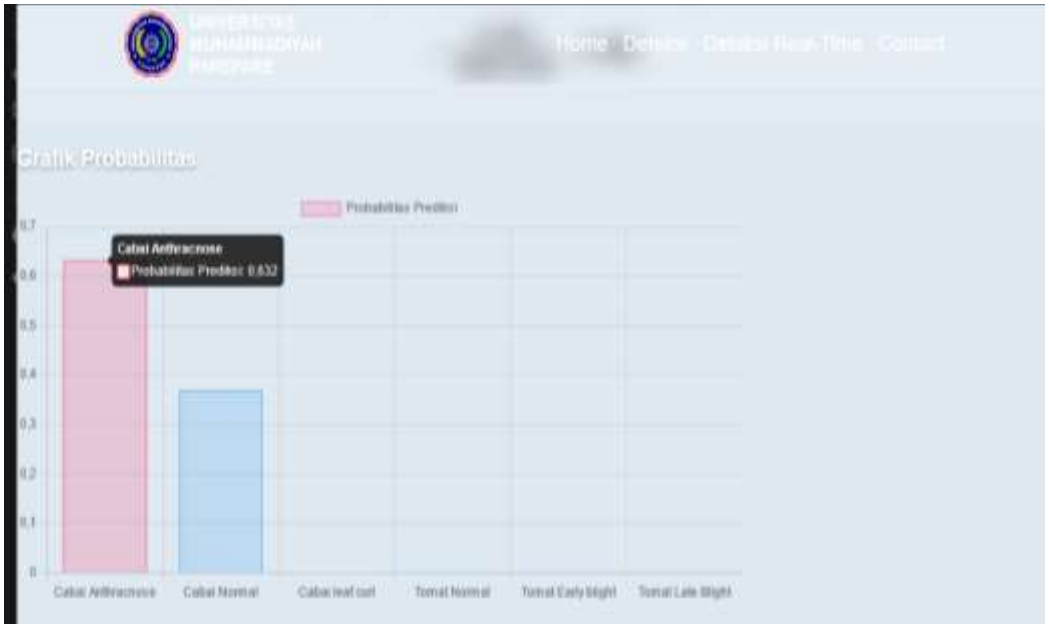
$$= \frac{0.1281}{20202.2582} = 0.0000$$

Gambar 4. 15 Tampilan *output softmax*

Tampilan ini menunjukkan hasil perhitungan probabilitas dari *softmax function*, yang digunakan oleh model untuk menentukan kelas mana yang memiliki probabilitas tertinggi. *Softmax* digunakan dalam klasifikasi multi-kelas untuk mengubah *output* dari model menjadi distribusi probabilitas, di mana jumlah dari

semua probabilitas kelas selalu sama dengan 1. Pada gambar ini, persamaan *softmax* ditampilkan dalam format *LaTeX* untuk memudahkan pemahaman, dengan setiap kelas memiliki probabilitas yang dihitung berdasarkan nilai eksponensial dari *output* logit (*z*) dibandingkan dengan jumlah eksponensial dari semua kelas. Hasil probabilitas ini menunjukkan bahwa kelas dengan *z* tertinggi memiliki probabilitas terbesar, yang berarti model lebih yakin bahwa gambar termasuk dalam kelas tersebut.

5. Tampilan Grafik kelas

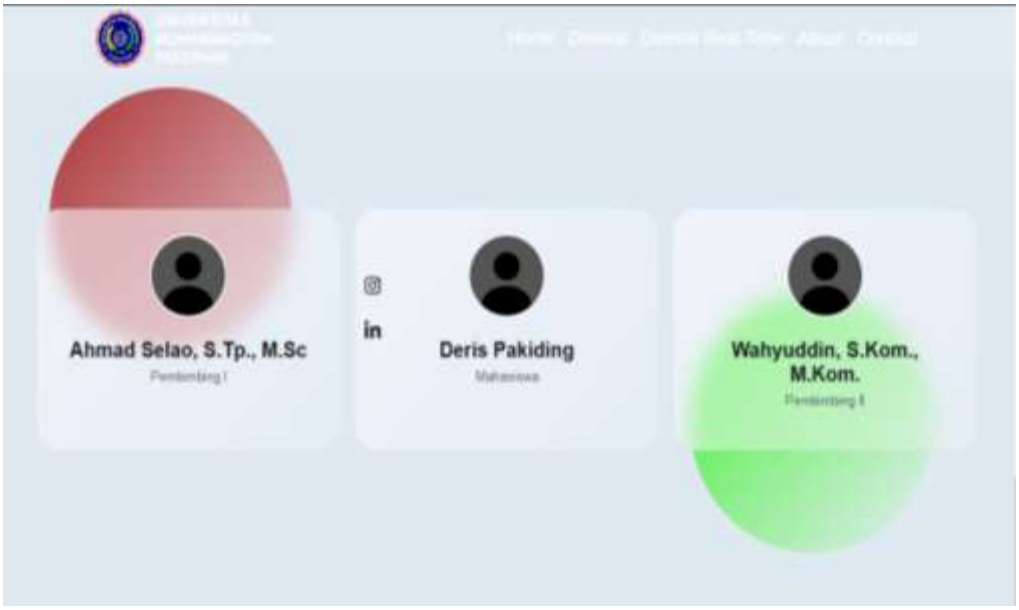


Gambar 4. 16 Grafik *probabilita*s kelas

Gambar 4.15 menampilkan grafik probabilitas prediksi yang dihasilkan oleh model deteksi penyakit tanaman cabai dan tomat. Yang digunakan untuk memvisualisasikan tingkat keyakinan model dalam mengklasifikasikan gambar yang diunggah ke dalam berbagai kategori penyakit atau kondisi normal. Sumbu horizontal (X) merepresentasikan kategori atau kelas yang tersedia dalam model, sementara sumbu vertikal (Y) menunjukkan nilai probabilitas prediksi untuk

masing-masing kelas. Warna dalam grafik menunjukkan tingkat probabilitas, dengan tooltip yang muncul saat pengguna mengarahkan kursor ke salah satu batang untuk melihat nilai spesifiknya. Visualisasi ini membantu dalam memahami bagaimana model memproses data dan seberapa besar keyakinan model terhadap hasil prediksi yang diberikan.

6. Tampilan *Contact*



Gambar 4. 17 Tampilan Informasi Aplikasi

Bagian akhir dari aplikasi ini berfungsi sebagai halaman informasi mengenai tim pengembang, yang terdiri dari mahasiswa dan dosen pembimbing.Pengujian Sistem

I. Pengujian Sistem

1. Pengujian *Black Box*

   a. Halaman Utama

Test Faktor	Hasil	Keterangan
Menu Home	✓	Berhasil, aplikasi menampilkan halaman utama

   b. Halaman prediksi

Test Faktor	Hasil	Keterangan
Deteksi melalui inputan file gambar	✓	Berhasil, aplikasi menampilkan hasil deteksi inputan file gambar

c. Deteksi gambar dari kamera

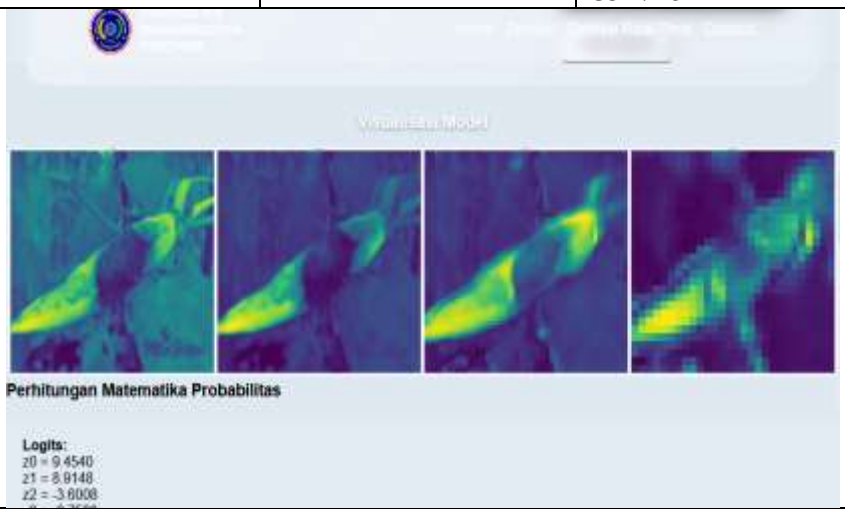
Test Faktor	Hasil	Keterangan
Membuka kamera dan mendeteksi gambar yang diambil dari kamera	✓	Berhasil, aplikasi dapat membuka kamera dan ambil gambar untuk dideteksi

d. Menu detail penyakit

Test Faktor	Hasil	Keterangan
Visualisas i*Feature map layers*	✓	Berhasil, aplikasi menampilkan visualiasi *layer* conv2d

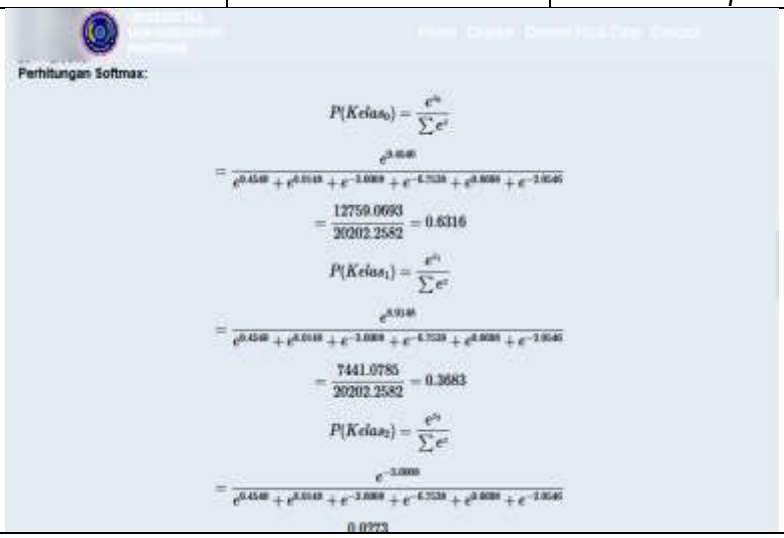
e. Visualisasi *layer*

Test Faktor	Hasil	Keterangan
Visualisas i*Feature map layers*	✔	Berhasil, aplikasi menampilkan visualiasi *layer* conv2d



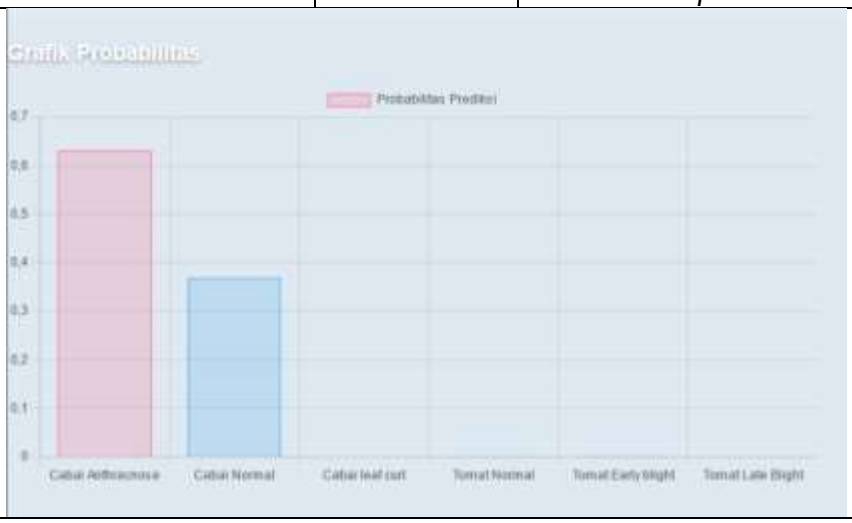
f. Perhitungan nilai *probabilitas* tiap kelas

Test Faktor	Hasil	Keterangan
Perhitungan probabilitas	✔	Berhasil, aplikasi menampilkan perhitungan dari *layer output* dengan *activation softmax*



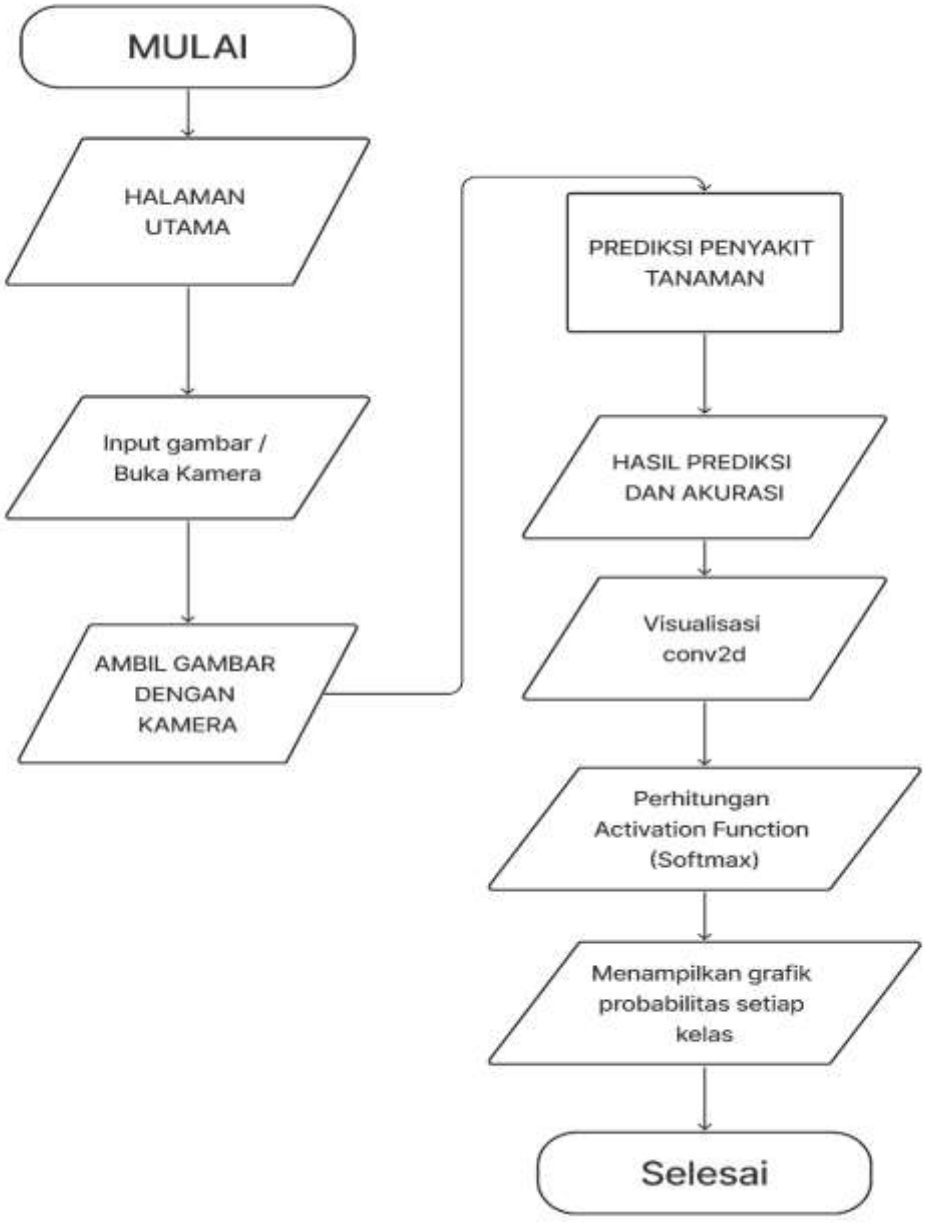
g. Grafik *Probabilitas*

Test Faktor	Hasil	Keterangan
Menampilkan grafik probabilitas hasil deteksi	✓	Berhasil, aplikasi menampilkan perhitungan dari *layer output* dengan *activation softmax*



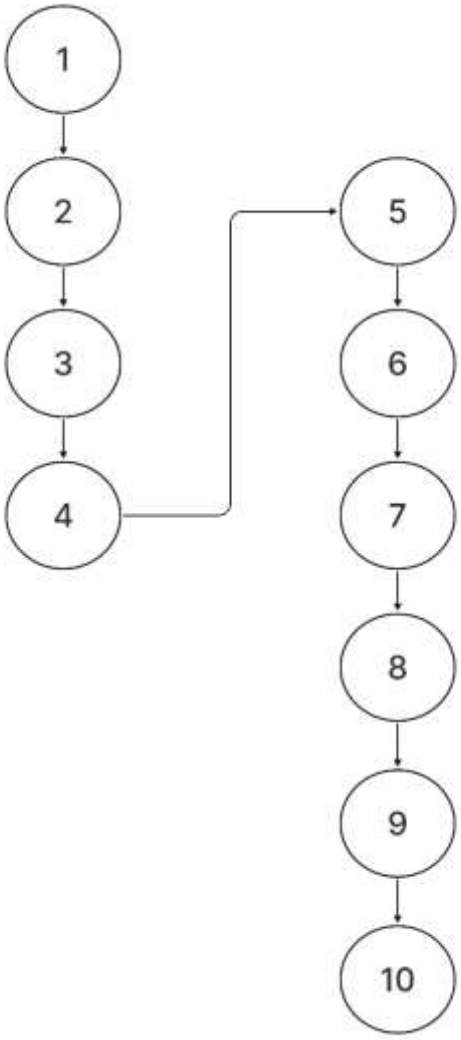
2. Pengujian White *Box*

a. *Flowchart* sistem aplikasi



Gambar 4. 18 *Flowchart* sistem aplikasi

b. *Flowgraph*



Gambar 4. 19 *Flowgraph* sistem aplikasi

Berdasarkan diagram:

Node (N): Terdapat 10 node

Edge (E): 9

Menghitung siklomatik kompleksitas

Rumus :

$V(G) = E - N + 2$

Substitusi nilai E = 9 dan N = 10:
V(G) = 9 - 10 + 2 = 1

Region adalah area tertutup pada *flowchart*. Karena tidak ada loop atau branching, jumlah region tetap:
Region = 1

Independent path

1. Path 1: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10

Tabel 4. 6 Grafik *Matrix*

	1	2	3	4	5	6	7	8	9	10	E -1
1	0	1	0	0	0	0	0	0	0	0	1 – 1 = 0
2	0	0	1	0	0	0	0	0	0	0	1 – 1 = 0
3	0	0	0	1	0	0	0	0	0	0	1 – 1 = 0
4	0	0	0	0	1	0	0	0	0	0	1 – 1 = 0
5	0	0	0	0	0	1	0	0	0	0	1 – 1 = 0
6	0	0	0	0	0	0	1	0	0	0	1 – 1 = 0
7	0	0	0	0	0	0	0	1	0	0	1 – 1 = 0
8	0	0	0	0	0	0	0	0	1	0	1 – 1 = 0
9	0	0	0	0	0	0	0	0	0	1	1 – 1 = 0
10	0	0	0	0	0	0	0	0	0	0	0
	SUM (E + 1)										0 + 1 = 1

3. Pengujian deteksi aplikasi

a. Deteksi kelas penyakit

Tabel 4. 7 Uji cabai *anthracnose*

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Cabai *Antrachnose*	1		Berhasil

Lanjutan Tabel 4.7 Uji cabai *antrahracnose*

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Cabai *Anthracnose*	2		Berhasil
	3		Berhasil
	4		Berhasil
	5		Berhasil
	6		Berhasil
	7		Berhasil

Lanjut Tabel 4.7 Uji cabai *anthracnose*

Kelas AKtual	Uji ke-	Hasil Deteksi	Keterangan
Cabai *Anthracnose*	8		Berhasil
	9		Berhasil
	10		Berhasil

Tabel 4. 8 Uji cabai normal

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Cabai Normal	1		Berhasil
	2		Berhasil
	3		Berhasil

Lanjutan Tabel 4.8 Uji cabai normal

Kelas Aktual	Uji Ke-	Hasil Deteksi	Keterangan
Cabai Normal	4		Berhasil
	5		Gagal
	6		Berhasil
	7		Berhasil
	8		Berhasil

Lanjutan Tabel 4.8 Uji cabai normal

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Cabai Normal	9		Berhasil
	10		Berhasil

Tabel 4. 9 Uji cabai *leaf curl*

Kelas Aktual	Uji Ke-	Hasil Deteksi	Keterangan
Cabai *Leaf Curl*	1		Berhasil
	2		Gagal
	3		Berhasil

Lanjutan Tabel 4.9 Uji cabai *leaf curl*

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Cabai *Leaf Curl*	4		Berhasil
	5		Gagal
	6		Gagal
	7		Berhasil
	8		Berhasil
	9		Gagal
	10		Gagal

Tabel 4. 10 Uji tomat normal

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Tomat Normal	1		Berhasil
	2		Berhasil
	3		Berhasil
	4		Gagal
	5		Berhasil
	6		Gagal

Lanjutan Tabel 4.10 Uji tomat normal

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Tomat *Early Blight*	7		Gagal
	8		Berhasil
	9		Gagal
	10		Berhasil

Tabel 4. 11 Uji tomat *early blight*

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Tomat *Early blight*	1		Gagal

Lanjutan Tabel 4.11 Uji tomat *early blight*

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Tomat *Early Blight*	2		Berhasil
	3		Berhasil
	4		Gagal
	5		Gagal
	6		Berhasil
	7		Berhasil

Lanjutan Tabel 4. 11 Uji tomat *early blight*

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Tomat *Early Blight*	8		Berhasil
	9		Berhasil
	10		Berhasil

Tabel 4. 12 Uji tomat *late blight*

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Tomat *Early Blight*	1		Berhasil
	2		Gagal

Lanjutan Tabel 4.12 Uji tomat *lateblight*

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Tomat *Late Blight*	3		Berhasil
	4		Gagal
	5		Berhasil
	6		Berhasil
	7		Berhasil
	8		Berhasil

Lanjutan Tabel 4.12 Uji tomat *late blight*

Kelas Aktual	Uji ke-	Hasil Deteksi	Keterangan
Tomat *Late Blight*	10		Berhasil
	9		Berhasil

Tabel 4. 13 Hasil uji deteksi aplikasi

Kelas Aktual	Deteksi Benar	Deteksi Salah
Cabai Antraknosa	10	0
Cabai Normal	9	1
Cabai *Leaf Curl*	5	5
Tomat Normal	6	4
Tomat *Early Blight*	7	3
Tomat *Late Blight*	8	2

Hasil pengujian ini diperoleh dari percobaan deteksi menggunakan data nyata yang diambil langsung melalui kamera. Data ini mencakup enam kelas, yaitu Cabai Antraknosa, Cabai Normal, Cabai *Leaf Curl*, Tomat Normal, Tomat *Early Blight*, dan Tomat *Late Blight***.** Setiap kelas memiliki jumlah sampel tertentu yang digunakan untuk mengukur kemampuan model dalam mengklasifikasikan kondisi tanaman dengan benar atau salah.

Menghitung akurasi deteksi :

$$Akurasi = \frac{Jumlah\ deteksi\ benar}{Total\ sampel} \times 100\%$$

$$Akurasi = \frac{(10+9+5+6+7+8)}{(10)+(9+1)+(5+5)+(6+4)+(7+3)+(8+2)} = \frac{42}{60} \times 100\% = 75\%$$

berdasarkan pengujian manual yang dilakukan setelah implementasi model ke dalam aplikasi berbasis web, akurasi deteksi penyakit tanaman cabai dan tomat mencapai 75%**.** Hasil ini menunjukkan adanya perbedaan dengan akurasi yang diperoleh selama pelatihan model, yang sebelumnya mencapai 91%**.** Perbedaan ini bisa disebabkan oleh beberapa faktor, seperti kondisi pencahayaan, kualitas gambar yang diambil melalui kamera, sudut pengambilan gambar, serta kemungkinan adanya perbedaan distribusi data antara *dataset* pelatihan dan data nyata yang digunakan untuk pengujian langsung.Dalam pengujian ini, model mengalami beberapa kesalahan klasifikasi, terutama pada kelas Cabai *Leaf Curl* dan Tomat Normal**,** yang memiliki tingkat kesalahan masing-masing sebesar 50% dan 40%. Selain itu, model juga masih menunjukkan kelemahan dalam mengenali objek yang tidak termasuk dalam kategori pelatihan, yang dapat menyebabkan prediksi keliru.

b. Deteksi gagal

Gambar 4. 20 Pendeteksian tidak berhasil

Gambar 4. 21 Visualisasi *feature map* dari Pendeteksi tidak berhasil

Visualisasi *feature map* ini menunjukkan bagaimana model CNN memberikan perhatian pada area tertentu dalam gambar yang tidak termasuk dalam kategori yang telah dilatih, namun tetap diklasifikasikan sebagai cabai antraknosa. Dari pola aktivasi yang terlihat, model tampaknya mengidentifikasi bagian dengan tekstur atau kontras tertentu yang menyerupai karakteristik penyakit pada cabai, sehingga terjadi salah klasifikasi. Aktivasi yang kuat pada area tertentu menunjukkan bahwa model memiliki pola spesifik yang dijadikan acuan untuk mengenali cabai antraknosa, yang mungkin juga ditemukan dalam gambar input ini.

Kesalahan ini bisa disebabkan oleh keterbatasan *dataset* pelatihan yang tidak memiliki cukup variasi, sehingga model kurang mampu membedakan objek yang tidak relevan. Selain itu, model tidak dilatih untuk mengenali kategori "di luar kelas" (*out of distribution*), sehingga setiap input yang diberikan tetap akan dipetakan ke salah satu kelas yang ada.

BAB V

PENUTUP

A. Kesimpulan

Penelitian ini mengimplementasikan teknologi *computer vision* pada aplikasi berbasis web untuk mendeteksi penyakit pada tanaman cabai dan tomat menggunakan metode *Convolutional Neural Networks* (CNN)**.** Aplikasi ini memungkinkan pengguna untuk mengunggah gambar tanaman melalui antarmuka berbasis HTML, CSS, *Python*, *JavaScript*, dan *Flask*, yang kemudian diproses oleh model CNN untuk mengidentifikasi jenis penyakit tanaman. Model yang dibangun mampu mengklasifikasikan enam kategori**,** yaitu cabai normal, cabai *leaf curl*, cabai antraknosa, tomat normal, tomat *early blight*, dan tomat *late blight.* Hasil evaluasi menggunakan *dataset* uji menunjukkan bahwa model memiliki performa yang cukup baik dengan akurasi 91% pada pelatihan**.** Namun, dalam tahap implementasi pada data nyata melalui kamera secara real-time, model mengalami sedikit penurunan performa dengan akurasi sebesar 75%**.**

Hasil pengujian lebih lanjut menunjukkan bahwa kesalahan prediksi cenderung terjadi pada kelas-kelas yang memiliki karakteristik visual yang mirip**,** seperti tomat *early blight* dan tomat *late blight,* serta cabai normal yang sering diklasifikasikan sebagai cabai *leaf curl.* Selain itu, model mengalami kesulitan dalam menangani gambar yang tidak termasuk dalam kategori pelatihan, sehingga beberapa objek di luar *dataset* dapat salah diklasifikasikan. Analisis menggunakan

Confusion Matrix menunjukkan adanya *misclassifications* yang signifikan pada beberapa kelas tertentu. Visualisasi *feature map* juga memperlihatkan bahwa model lebih banyak fokus pada fitur tertentu dalam gambar, tetapi terkadang area yang diperhatikan kurang relevan dengan karakteristik penyakit yang sebenarnya. Hal ini menunjukkan bahwa meskipun model sudah cukup baik dalam mendeteksi penyakit tanaman, masih diperlukan peningkatan lebih lanjut agar model lebih *robust* terhadap data dunia nyata**.**

B. Saran

Berdasarkan hasil penelitian dan evaluasi yang telah dilakukan, terdapat beberapa saran yang dapat diterapkan untuk pengembangan lebih lanjut. Pertama, perlu dilakukan perluasan *dataset* dengan menambahkan lebih banyak data dari kondisi nyata untuk meningkatkan kemampuan generalisasi model. Selain itu, augmentasi data yang lebih variatif dapat diterapkan untuk membuat model lebih *robust* terhadap berbagai kondisi pencahayaan, sudut pengambilan gambar, dan latar belakang yang beragam.

penelitian ini dapat dikembangkan lebih lanjut dengan mengeksplorasi arsitektur model yang lebih kompleks atau menerapkan teknik *transfer learning* dari model yang telah dilatih pada *dataset* yang lebih besar. Dengan demikian, model yang dikembangkan dapat semakin akurat dan aplikatif dalam mendukung deteksi penyakit tanaman secara otomatis di berbagai kondisi lingkungan.

DAFTAR PUSTAKA

Amanzadi, A., & Karim, M. (2022). *Comparison of Machine Learning Models Used for Swedish Text Classification in Chat Messaging.* Stockholm: DiVa.

Alzubaidi, L., Zhang, J., Humaidi, A., Al-Djuaili, A., Duan, Y., Al-Shama, O., . . . Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*.

Bredesen, B. A., & Rehmsmeier, M. (2022). Gnocis: An integrated system for interactive and reproducible analysis and modelling of cis-regulatory elements in *Python* 3. *Plus One*, 17 (9).

Denishtsany, D. R. (2023). *Apa Perbedaan HTML, CSS, dan JavaScript?* diambil dari toffeedev.com: sumber : https://toffeedev.com/blog/digital-marketing/crm-apa-perbedaan-htm-css-dan-html/ (diakses 18 April 2024)

Dubey, A., Lazarus, A., & Mangal, D. (2020). Handwritten Digit Recognition using Image *Preprocessing* and CNN. *IJSRCSEIT*.

Geron, A. (2019). *Hands-On Machine Learning With Scikit-Learn, Keras & TensorFlow*. Canada: O'Reilly Media.

Gupta, P., & Bagchi, A. (2024). Introduction to *Numpy*. Dalam "*Essentials of Python for Artificial Intelligence and Machine Learning* " (pp. 127-128). USA: Springer, Cham.

Introduction Visual Studio Code. (2019). In B. Jhonson, *Visual Studio Code End to End Editing and Debugging Tools for web Developers* (pp. 1-12). Indianapolis: John Wiley & Sons, Inc.

Lesmana, A. M., Fadhillah , R. P., & Rozikin, C. (2022). Identifikasi Penyakit pada Citra Daun Kentang Menggunakan Convolutional Neural Network (CNN). *Jurnal Sains dan Informatika Vol.8 No.1*.

Lucidchart. (2023). *What is a Flowchart*. Retrieved from Lucidchart: https://www.lucidchart.com/pages/what-is-a-*flowchart*-tutorial (diakses 18 April 2024)

Manuaba, I. K., Sutedja, I., & Bahana, R. (2020). *The Evaluation Of Supervised Classifier Modelsto Develop A Machine Learning Api For Predicting Cardiovascular Disease Risk*. Icic Express Letters, *VOL.14, NO.3*, 219-226.

Murel Ph.D., J., & Kavlakoglu, E. (2024, 01 19). *What Is Confusion a Matrix*. diambil dari IBM: https://www.ibm.com/topics/confusion-*matrix* (diakses 19 April 2024)

Rahman, A. (2024). *Brain Tumor Detection and Classification by Using CNN*. Finland: UEF eRepository.

Raschka, S. (2020). *Machine Learning* in *Python*: Main Developments and Technology Trends In Data Science, *Machine Learning* and Artificial Intelligence. *Information*.

Rasywir, E., & Sinaga, R. (2020). Analisis dan Implementasi Diagnosa Penyakit Sawit Dengan Metode *Convolutional Neural Network*. *Paradigma - Jurnal Informatika dan Komputer Vo.22 No.22*.

Sanjaya, J., & Ayub, M. (2020). Augmentasi Data Pengenalan Citra Mobil Menggunakan Pendekatan Random *Crop, Rotate,dan Mixup*. *Jurnal Teknik Informatika dan Sistem Informasi*.

Sonata, F., & Sari, V. W. (2019). Pemanfaatan UML (Unified Modeling Language) Dalam Perancangan Sistem Informasi E-Commerce Jenis Customer-To-Customer. *Jurnal Komunikasi, Media dan Informatika*, 8 (1).

Suh, Y. S., Shin, S. K., Baang, D., & Mun, S. (2020). A Method to Reduce Data Dimensions in *Machine Learning* Programmi. *Transactions of the Korean Nuclear Society Virtual Autumn Meeting*.

Sya'bani, D. R., Hamzah, A., & Susanti, E. (2022). *Klasifikasi Buah Segar Dan Busuk Menggunakan Algoritma Convolutional Neural Network Dengan Tflite Sebagai Media Penerapan Model Machine Learning*. diambil dari: ejournal.akprind.ac.id: https://ejournal.akprind.ac.id/index.php/snast/article/download/4180/2976/6861? (diakses 19 februari 2024)

Taner, A., Oztekin, Y. B., & Duran, H. (2021). *Performence Analysis of Deep Learning CNN Models for Variety Classification in Hazelnut*. Sustainability.

Tian, H., Wang, T., Liu, Y., Qiao, X., & Li, Y. (2020). *Computer Vision* Technology in Agricultural Automation. *INFORMATION PROCESSING IN AGRICULTURE vol.7*, 1-19.

Wairooy, I. K. (2023). *Teknik Dalam White-box dan Black-box Testing*. Retrieved from Binus: https://socs.binus.ac.id/2020/07/02/teknik-dalam-white-*box*-dan-*black-box-test*ing (diakses 20 april 2024)

Waskom, M. L. (2021). seaborn: *statistical data visualization*. *The Journal of Open Source Software*.

Wati, C., Arsi, Karenina, T., Riyanto, Nirwanto, Y., Nurcahya, I., . . . Nurul, D. (2021). *Hama dan Penyakit Tanaman.* bogor: Yayasan Kita Menulis.

Zheng, Q., Yang, M., Tian, X., Jiang, N., & Wang, D. (2020). A Full Stage Data Augmentation Method in Deep Convolutional Neural Network For Natural Image Classification. *Discrete Dynamics in Nature and Society*.