

**KLASIFIKASI PENYAKIT PADA DAUN MANGGA BERBASIS
CITRA MENGGUNAKAN *CONVOLUTIONAL NEURAL
NETWORK***

TUGAS AKHIR



UBAYA
UNIVERSITAS SURABAYA

Disusun Oleh :

SUBRATA

NRP : 160420002

**PROGRAM DATA SCIENCE & ARTIFICIAL INTELLIGENCE
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK UNIVERSITAS SURABAYA
JANUARI 2024**

KATA PENGANTAR

Puji Syukur kepada Tuhan Yang Maha Esa atas segala Rahmat-Nya sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul “Klasifikasi Penyakit Pada Daun Mangga Berbasis Citra Menggunakan *Convolutional Neural Network*”. Penggerjaan tugas akhir ini dapat selesai dengan bantuan bimbingan dan dukungan dari berbagai pihak. Maka dari itu, penulis ingin mengucapkan banyak terima kasih kepada :

1. Bapak Dr. Joko Siswantoro, S.Si., M.Si., selaku Ketua Jurusan Teknik Informatika Universitas Surabaya dan dosen pembimbing I yang telah membimbing penulis dalam penggerjaan tugas akhir ini.
2. Bapak Mohammad Farid Naufal, M.Kom., selaku dosen pembimbing II yang sudah memberikan waktu, bimbingan, serta saran selama penggerjaan tugas akhir ini.
3. Bapak Drs. Heru Arwoko, M.T., selaku dosen wali yang telah memberikan arahan, bimbingan, serta masukan yang berharga selama masa perkuliahan.
4. Bapak Mochamad Anshori Irfan, selaku Penyuluh Pertanian Lapang Kecamatan Kenjeran Kota Surabaya yang telah bersedia diwawancara untuk memvalidasi penelitian yang dilakukan penulis.
5. Keluarga penulis, terutama kedua orang tua dan saudara penulis yang selalu memberikan dukungan dan doa kepada penulis.
6. Teman-teman penulis yang telah memberikan semangat dan doa kepada penulis.
7. Pihak lain yang tidak dapat disebutkan satu persatu yang turut membantu penulis menyelesaikan tugas akhir ini.

Penulis dengan jujur mengakui bahwa tugas akhir yang disusun masih memiliki beberapa kekurangan dan perlu perbaikan. Kesadaran akan ketidak sempurnaan tersebut mendorong penulis untuk menghargai segala kritik dan saran yang dapat membantu menyempurnakan serta mengembangkan tugas

akhir ini. Dalam upaya untuk mencapai tingkat kesempurnaan yang lebih tinggi, penulis bersedia menerima masukan konstruktif demi peningkatan kualitas karya ini. Dengan demikian, penulis berharap agar tugas akhir ini tidak hanya mencapai standar yang diinginkan, tetapi juga dapat memberikan manfaat yang signifikan bagi pembaca, menggugah pemikiran, dan memperkaya pemahaman terhadap topik yang dibahas.

Surabaya, 5 Januari 2024

Penulis

KLASIFIKASI PENYAKIT PADA DAUN MANGGA BERBASIS CITRA MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK*

Nama : Subrata

Program Studi : Teknik Informatika

Pembimbing I : Dr. Joko Siswantoro, S.Si., M.Si.

Pembimbing II : Mohammad Farid Naufal, M.Kom.

ABSTRAK

Penyakit pada daun mangga merupakan gangguan pada tanaman mangga yang menyebabkan tanaman mati secara perlahan. Melihat akibat dari penyakit tersebut, penting sekali untuk mengklasifikasikan penyakit pada daun mangga. Terdapat 8 kategori yang diklasifikasi, yaitu 7 jenis penyakit dan 1 *healthy*. Aplikasi yang dibuat mendeteksi daun mangga terlebih dahulu menggunakan model YOLOv8 dengan mAP50 terbaik yaitu 99.5% yang *di-training* menggunakan *dataset* dari internet. Setelah daun mangga terdeteksi, aplikasi melakukan klasifikasi dengan model *pre-trained* dari arsitektur terbaik. Arsitektur disusun dengan parameter terbaik hasil dari *hyperband Keras*. Diperoleh akurasi model yaitu ResNet50 dengan akurasi 98.5%, VGG16 dengan akurasi 97.87%, MobileNet dengan akurasi 94.63% dan Inception-v3 dengan akurasi 71.38%. Arsitektur terbaik adalah ResNet50, maka aplikasi menggunakan model ResNet50 untuk melakukan klasifikasi. Evaluasi dilakukan menggunakan *confusion matrix* dengan nilai *precision* sebesar 98.5%, *recall* sebesar 98.5% dan *F1-score* sebesar 98.5%. Hasil dari aplikasi berupa sebuah label dari 8 kategori tersebut dan tingkat keyakinan model dalam melakukan klasifikasi. Aplikasi dibuat berbasis *mobile* dengan sistem operasi Android menggunakan *platform* Flutter dengan bahasa pemrograman Dart untuk *front end* dan bahasa pemrograman Python untuk *back end*. *Front end* dan *back end* dihubungkan dengan *web service* yaitu Flask. Uji coba aplikasi dilakukan di Dinas Ketahanan Pangan dan Pertanian Kota Surabaya, hasil yang diperoleh yaitu aplikasi dapat membantu masyarakat dalam mengklasifikasikan penyakit pada tanaman mangga melalui citra daunnya dengan metode *Convolutional Neural Network* (CNN).

Kata kunci: Penyakit daun mangga, YOLOv8, Klasifikasi, *Convolutional Neural Network* (CNN)

IMAGE-BASED CLASSIFICATION OF DISEASES ON MANGO LEAVES USING CONVOLUTIONAL NEURAL NETWORK

Name: Subrata

Study Program: Informatics Engineering

Contributor I : Dr. Joko Siswantoro, S.Si., M.Si.

Contributor II : Mohammad Farid Naufal, M.Kom.

ABSTRACT

Mango leaf disease is a disorder of the mango plant that causes the plant to die slowly. Seeing the consequences of this disease, it is very important to classify diseases on mango leaves. There are 8 categories classified, namely 7 types of disease and 1 healthy. The application created to detect mango leaves first uses the YOLOv8 model with the best mAP50, namely 99.5%, which was trained using datasets from the internet. After the mango leaves are detected, the application performs classification with a pre-trained model from the best architecture. The architecture is structured with the best parameters resulting from the Keras hyperband. The model accuracy obtained was ResNet50 with an accuracy of 98.5%, VGG16 with an accuracy of 97.87%, MobileNet with an accuracy of 94.63% and Inception-v3 with an accuracy of 71.38%. The best architecture is ResNet50, so the application uses the ResNet50 model to perform classification. Evaluation was carried out using a confusion matrix by comparing precision values of 98.5%, recall of 98.5% and F1-score of 98.5%. The results of the application are a label for the 8 categories and the model's level of confidence in carrying out the classification. The application was created on a mobile basis with the Android operating system using the Flutter platform with the Dart programming language for the front end and the Python programming language for the back end. The front end and back end are connected to a web service, namely Flask. The application trial was carried out at the Surabaya City Food Security and Agriculture Service, the results obtained were that the application could help the public in classifying diseases in mango plants through the image of the leaves using the Convolutional Neural Network (CNN) method.

Keywords: **Mango leaf disease, YOLOv8, Classification, Convolutional Neural Network (CNN)**

DAFTAR ISI

HALAMAN SAMPUL.....	i
KATA PENGANTAR.....	ii
ABSTRAK.....	iv
DAFTAR ISI.....	vi
DAFTAR GAMBAR.....	ix
DAFTAR TABEL.....	xi
DAFTAR LISTING.....	xii
BAB 1 PENDAHULUAN.....	1-1
1.1. Latar Belakang.....	1-1
1.2. Rumusan Masalah.....	1-4
1.3. Tujuan.....	1-4
1.4. Manfaat.....	1-4
1.5. Ruang Lingkup.....	1-5
1.6. Metodologi Penelitian.....	1-6
1.7. Sistematika Penulisan.....	1-8
BAB 2 LANDASAN TEORI.....	2-1
2.1. Penyakit Pada Daun Mangga.....	2-1
2.2. Convolutional Neural Network.....	2-5
2.2.1. Convolution Layer.....	2-6
2.2.2. Activation Layer.....	2-8
2.2.3. Pooling Layer.....	2-11
2.2.4. Fully Connected Layer.....	2-12
2.2.5. Dropout Regulation.....	2-13
2.2.6. Softmax Classifier.....	2-14
2.2.7. Cross Entropy Loss Function.....	2-15

2.3. Arsitektur Convolutional Neural Network.....	2-16
2.3.1. Residual Network 50.....	2-17
2.3.2. Visual Geometry Group 16.....	2-18
2.3.3. MobileNet.....	2-20
2.3.4. GoogleNet (Inception-v3).....	2-21
2.4. Hyperparameter Tuning.....	2-22
2.5. You Only Look Once.....	2-24
2.5.1. Intersection over Union.....	2-26
2.5.2. Mean Average Precision.....	2-27
2.6. Library.....	2-28
2.7. Evaluasi.....	2-30
BAB 3 ANALISIS SISTEM.....	3-1
3.1. Analisis Situasi Saat Ini.....	3-1
3.2. Analisis Penelitian Serupa.....	3-6
3.3. Identifikasi Masalah.....	3-9
3.4. Analisis Kebutuhan Sistem.....	3-10
BAB 4 DESAIN SISTEM.....	4-1
4.1. Desain Proses.....	4-1
4.1.1. Desain Proses Pembagian Dataset.....	4-1

4.1.2. Desain Proses Pelatihan dan Evaluasi Model.....	4-3
4.1.3. Desain Proses Klasifikasi Citra.....	4-6
4.1.4. Desain Proses Penggunaan Aplikasi.....	4-7
4.2. Desain User Interface.....	4-8
BAB 5 IMPLEMENTASI SISTEM.....	5-1
5.1. Implementasi Proses.....	5-1
5.1.1. Implementasi Proses Pelatihan dan Evaluasi Model YOLOv8....	5-1
5.1.2. Implementasi Proses Pembagian Dataset Klasifikasi.....	5-2
5.1.3. Implementasi Proses Pelatihan dan Evaluasi Model Klasifikasi...	5-3
5.1.4. Implementasi Proses Mendeteksi Daun Mangga.....	5-9
5.1.5. Implementasi Proses Klasifikasi Citra.....	5-10
5.1.6. Implementasi Proses Penggunaan Aplikasi.....	5-11
5.2. Implementasi User Interface.....	5-20
BAB 6 UJI COBA DAN EVALUASI.....	6-1
6.1. Verifikasi.....	6-1
6.1.1. Pengujian Aplikasi.....	6-1
6.1.2. Pengujian Model.....	6-6
6.2. Validasi.....	6-17
BAB 7 KESIMPULAN DAN SARAN.....	7-1
7.1. Kesimpulan.....	7-1
7.2. Saran.....	7-2
DAFTAR PUSTAKA.....	A-1

DAFTAR GAMBAR

Gambar 2.1. Arsitektur CNN.....	2-6
Gambar 2.2. Ilustrasi Proses Operasi Konvolusi.....	2-7
Gambar 2.3. Fungsi Aktivasi ReLU.....	2-9
Gambar 2.4. Fungsi Aktivasi Sigmoid.....	2-10
Gambar 2.5. Fungsi Aktivasi Tanh.....	2-10
Gambar 2.6. Ilustrasi Proses Operasi Pooling.....	2-11
Gambar 2.7. Ilustrasi Fully Connected Layer.....	2-13
Gambar 2.8. Ilustrasi Dropout.....	2-14
Gambar 2.9. Ilustrasi Arsitektur ResNet50.....	2-18
Gambar 2.10. Arsitektur VGG16.....	2-20
Gambar 2.11. Ilustrasi Depthwise Separable Convolution.....	2-21
Gambar 2.12. Arsitektur GoogleNet (Inception-v3).....	2-22
Gambar 2.13. Grafik Perbandingan Performa YOLO.....	2-25
Gambar 2.14. Perbandingan Performa Varian YOLOv8.....	2-25
Gambar 2.15. Ilustrasi Bounding Box Pada IoU.....	2-25
Gambar 3.1. Kriteria Responden.....	3-1
Gambar 3.2. Penyakit Anthracnose Pada Daun Mangga.....	3-2
Gambar 3.3. Diagram Jawaban Responden Pada Penyakit Anthracnose.....	3-3
Gambar 3.4. Penyakit Bacterial Canker Pada Daun Mangga.....	3-4
Gambar 3.5. Diagram Jawaban Responden Pada Penyakit Bacterial Canker....	3-4
Gambar 3.6. Diagram Kesulitan Responden.....	3-5
Gambar 4.1. Contoh Citra Daun Mangga Dataset YOLO.....	4-2
Gambar 4.2. Contoh Citra Daun Mangga Dataset Klasifikasi.....	4-3
Gambar 4.3. Diagram Alur Proses Training Model YOLOv8.....	4-4
Gambar 4.4. Diagram Alur Proses Training Model Klasifikasi CNN.....	4-6

Gambar 4.5. Diagram Alur Proses Klasifikasi Citra.....	4-7
Gambar 4.6. Diagram Alur Penggunaan Aplikasi.....	4-8
Gambar 4.7. Desain User Interface.....	4-9
Gambar 5.1. Implementasi User Interface.....	5-21
Gambar 5.2. Tutorial Penggunaan Aplikasi.....	5-22
Gambar 6.1. Tampilan Halaman Aplikasi.....	6-2
Gambar 6.2. Tampilan Tutorial Penggunaan Aplikasi.....	6-3
Gambar 6.3. Uji Coba Input Citra Melalui Galeri.....	6-4
Gambar 6.4. Uji Coba Input Citra Melalui Kamera Ponsel.....	6-5
Gambar 6.5. Uji Coba Menekan Process Image Tanpa Memilih Citra.....	6-6
Gambar 6.6. Hasil Confusion Matrix Model Terbaik.....	6-10

DAFTAR TABEL

Tabel 2.1. Data Jenis Penyakit Pada Daun Mangga.....	2-2
Tabel 2.2. Detail Pembagian Dataset.....	2-4
Tabel 3.1. Perbandingan Penelitian Serupa.....	3-7
Tabel 4.1. Detail Nilai Hyperparameter Tuning.....	4-5
Tabel 6.1. Hasil Hyperparameter Tuning Terbaik.....	6-7
Tabel 6.2. Hasil Evaluasi Model Klasifikasi.....	6-7
Tabel 6.3. Hasil Classification Report Model Terbaik.....	6-8
Tabel 6.4. Perhitungan Akurasi Model Klasifikasi.....	6-11

DAFTAR LISTING

Listing 5.1. Program Pelatihan Model YOLOv8.....	5-2
Listing 5.2. Program Import Library dan Pembagian Dataset Klasifikasi.....	5-2
Listing 5.3. Program Import Library.....	5-3
Listing 5.4. Program Load Dataset Klasifikasi.....	5-4
Listing 5.5. Program Pembentukan Model Klasifikasi.....	5-6
Listing 5.6. Program Mengatur Hyperparameter Tuning.....	5-7
Listing 5.7. Program Mengatur Early Stopping.....	5-8
Listing 5.8. Program Mencari Hyperparameter Tuning Terbaik.....	5-8
Listing 5.9. Program Pelatihan Model Klasifikasi CNN.....	5-9
Listing 5.10. Program Import Library dan Load Model YOLO.....	5-9
Listing 5.11. Program Mendeteksi Citra Daun Mangga.....	5-10
Listing 5.12. Program Load Model Klasifikasi.....	5-10
Listing 5.13. Program Klasifikasi Citra Input.....	5-11
Listing 5.14. Program Pengiriman Citra dan Respon Server.....	5-13
Listing 5.15. Program Back End.....	5-16
Listing 5.16. Program Tampilan Aplikasi.....	5-19

BAB 1

PENDAHULUAN

Bab ini meliputi latar belakang, rumusan masalah, tujuan, manfaat, ruang lingkup, metodologi penelitian, dan sistematika penulisan yang digunakan pada penelitian ini. Penjelasan rinci untuk memberikan pemahaman yang mendalam terkait konteks penelitian. Berikut penjelasan secara rinci untuk setiap konteks pada penelitian.

1.1. Latar Belakang

Mangga yang memiliki nama ilmiah *Mangifera indica L.* adalah tanaman yang berasal dari wilayah perbatasan India dan Burma (Ide, 2013). Namun, persebarannya sampai ke Asia Tenggara salah satunya Indonesia. Mangga banyak diminati masyarakat karena buahnya yang memiliki cita rasa manis dan buah yang tebal. Buah mangga juga kaya antioksidan dan nutrisi yang bermanfaat bagi kesehatan. Oleh karena itu, banyak individu dan komunitas yang melakukan penanaman dan usaha budidaya mangga.

Namun demikian, tanpa perawatan mangga tak lepas dari berbagai macam hama dan penyakit. Hama merupakan organisme perusak akar, batang, daun, maupun bagian lain yang mengganggu pertumbuhan pada tanaman sedangkan penyakit merupakan gangguan pada tanaman yang menyebabkan tanaman mati secara perlahan (Hariyanto & Sa'diyah, 2018). Dengan begitu masyarakat harus peduli dengan tanaman mangga yang dimiliki agar tanaman bisa tumbuh sehat

dan terhindar dari berbagai penyakit. Beberapa contoh penyakit pada daun mangga yaitu *anthracnose*, *bacterial canker*, dan *die back*. Penyakit dan hama sangat merugikan, sehingga keberadaannya sangat tidak diinginkan oleh masyarakat yang menanam mangga. Pengendalian hama dan penyakit memerlukan penanganan yang berbeda-beda. Berdasarkan data survei yang dilakukan terhadap 44 orang masyarakat Indonesia melalui kuesioner, 100% responden mengalami kesulitan dalam mengklasifikasikan penyakit pada daun mangga. Hasil survei menunjukkan sebagian besar dari mereka kesulitan karena kurangnya pengetahuan tentang penyakit pada daun mangga.

Saat ini metode *deep learning* yang memiliki hasil paling signifikan dalam mengenali data citra adalah algoritma *Convolutional Neural Network* (CNN). Hal ini dikarenakan CNN berusaha meniru *visual cortex* manusia dalam mengolah informasi citra. Namun, kelemahan dari CNN yaitu membutuhkan proses *training* model yang lama (Nugroho et al., 2020).

Beberapa penelitian terdahulu yakni, sebuah sistem yang dapat mendiagnosis penyakit pada tumbuhan mangga menggunakan metode *naive bayes* dengan input berupa *checkbox* dan diberi label gejala yang dialami oleh daun mangga. Penelitian ini memperoleh akurasi sebesar 87,5% dalam mendiagnosis penyakit mangga (Effendi et al., 2019). Penelitian serupa yang pernah dilakukan, diagnosis daun mangga menggunakan model CNN dengan input berupa citra daun mangga. Hasil pengujian memperoleh tingkat akurasi sebesar 0,96 atau 96%, nilai ini sudah cukup akurat dalam mendiagnosis daun mangga yang terserang penyakit dan daun mangga yang sehat (Ayu et al., 2021). Selanjutnya penelitian klasifikasi

jenis citra daun mangga dengan metode CNN, pada penelitian ini dilakukan pelatihan sebanyak 60 iterasi (*epochs*) dan mendapatkan akurasi 97,72% dalam mengidentifikasi citra daun mangga (Fitrianingsih & Rodiah, 2020). Penelitian berikutnya, klasifikasi citra buah pir menggunakan CNN dengan tujuan mengatasi permasalahan klasifikasi buah secara manual. Akurasi yang didapatkan pada penelitian tersebut adalah 98% dari 100 sampel data baru. Pada penelitian ini dikatakan semakin besar nilai *epoch* maka waktu *training* semakin lama dan tidak terlalu mempengaruhi akurasi (Juliansyah & Laksito, 2021). Adapun penelitian lain yang membandingkan beberapa algoritma untuk mengklasifikasi citra cuaca yaitu algoritma SVM, KNN, dan CNN, hasil yang dari penelitian tersebut, algoritma yang memiliki performa terbaik dalam mengklasifikasikan cuaca adalah CNN dengan akurasi 0,942 atau 94,2% (Naufal, 2021).

Perbedaan tugas akhir ini dengan penelitian terdahulu yaitu pada penelitian terdahulu yang menggunakan input citra, hasilnya tidak spesifik dalam mengklasifikasikan penyakit daun mangga karena *output* yang dihasilkan yaitu sehat dan terserang penyakit. Kemudian, penelitian lain yang menghasilkan *output* spesifik berupa label penyakit daun mangga, masih menggunakan input gejala berupa *checkbox* dan akurasi kurang dari 90% dengan metode *naive bayes*. Pada tugas akhir ini menggunakan input berupa citra dan *output*-nya label penyakit pada daun mangga secara spesifik.

Dari beberapa penelitian sebelumnya yang menggunakan algoritma CNN menghasilkan akurasi lebih baik daripada algoritma lain dalam pengenalan citra. Dengan demikian, tugas akhir ini menerapkan algoritma CNN untuk

mengklasifikasikan penyakit pada daun mangga. Harapannya dengan mengimplementasikan algoritma CNN bisa mendapatkan akurasi lebih dari 90% dengan performa yang sangat baik.

1.2. Rumusan Masalah

Berdasarkan latar belakang penelitian ini, maka rumusan masalahnya yaitu bagaimana mengklasifikasikan penyakit pada daun mangga menggunakan citra digital?

1.3. Tujuan

Tugas akhir ini bertujuan membuat sebuah aplikasi untuk membantu masyarakat dalam mengklasifikasikan penyakit pada tanaman mangga melalui citra daunnya dengan metode *Convolutional Neural Network* (CNN).

1.4. Manfaat

Berikut manfaat dari penelitian ini yaitu:

1. Bagi Masyarakat:
 - a. Memudahkan masyarakat dalam mengklasifikasikan penyakit pada daun mangga.
 - b. Untuk mengetahui cara mengaplikasikan algoritma *Convolutional Neural Network* (CNN) menggunakan input data berupa citra.
 - c. Menambah pengetahuan tentang jenis penyakit pada daun mangga.
2. Bagi Pusat Penelitian dan Pengembangan Perkebunan:

- a. Sebagai sarana untuk mengedukasi jenis penyakit pada daun mangga.
- b. Untuk mengetahui kinerja metode CNN pada kasus klasifikasi penyakit pada daun mangga.
- c. Dapat dijadikan referensi penelitian berikutnya.

1.5. Ruang Lingkup

Ruang lingkup pada penelitian ini meliputi batasan-batasan sebagai berikut:

1. Aplikasi berbasis *mobile* dengan sistem operasi Android.
2. Input atau data yang diterima adalah citra daun mangga dengan *background* putih dan tanpa gangguan objek lain.
3. Aplikasi dapat menerima input melalui kamera ponsel dan galeri.
4. *Output* yang dihasilkan berupa keyakinan model dan label dari klasifikasi penyakit pada daun mangga.
5. Daun mangga diklasifikasikan menjadi salah satu dari 8 kategori. Kategori daun mangga yang diklasifikasikan terdiri dari 7 jenis penyakit dan 1 *healthy*. Adapun jenis penyakitnya yaitu *anthracnose*, *bacterial canker*, *die back*, *cutting weevil*, *gall midge*, *powdery mildew*, *sooty mould*.
6. Metode yang digunakan untuk mengklasifikasikan penyakit pada daun mangga yaitu *Convolutional Neural Network* (CNN).

1.6. Metodologi Penelitian

Dalam penelitian ini, rencana kegiatan yang dilakukan sebagai berikut:

1. Persiapan

Beberapa persiapan yang dilakukan seperti:

- a. Melakukan survei untuk menentukan permasalahan yang akan diselesaikan.
- b. Mencari *dataset* berupa citra daun mangga yang terserang penyakit.
- c. Membaca jurnal penelitian yang telah dilakukan sebelumnya untuk dijadikan referensi.

2. Analisis

Tahap ini menganalisis algoritma yang cocok digunakan pada kasus klasifikasi. Hasil analisis yaitu menggunakan algoritma *Convolutional Neural Network* (CNN) karena algoritma ini memiliki kinerja yang baik dalam mengklasifikasikan data citra. Kemudian menganalisis implementasi aplikasi yang sesuai dengan kasus ini.

3. Desain

Desain yang dilakukan berupa diagram untuk mengetahui alur dari input *user* hingga mendapatkan hasil *output*. Kemudian desain *dataset*, data citra diberi label yang sesuai dan dikelompokkan berdasarkan labelnya. Lalu desain model klasifikasi menggunakan metode CNN. Desain selanjutnya *split dataset* dengan proporsi 80% *training* dan 20% *testing*. Selanjutnya, desain performa menggunakan akurasi, *f1-score*, *precision*, dan *recall* untuk mengukur kinerja dari model. Kemudian

desain tahap *preprocessing* seperti melakukan *resize* untuk mengubah resolusi sesuai kebutuhan, *cropping*, *rotating*, *remove background*, dan lain-lain.

4. Implementasi

Implementasi pada tugas akhir ini dilakukan berdasarkan analisis sebelumnya yaitu aplikasi dibuat berbasis *mobile* Android. Karena lebih mudah digunakan dan lebih praktis serta tidak memerlukan server. Berdasarkan data CNN Indonesia, pada tahun 2022 pengguna Android di Indonesia mencapai 89,42%. Oleh karena itu, sistem operasi yang dipilih adalah Android. Input citra dapat melalui kamera ponsel dan galeri.

5. Uji Coba dan Evaluasi

Pengujian dilakukan terhadap beberapa citra daun mangga yang terserang penyakit, namun belum memiliki label jenis penyakitnya. Citra diproses menggunakan model yang sudah di-*training* sebelumnya. Aplikasi diharapkan dapat mengklasifikasikan dengan benar penyakit pada daun mangga. Selanjutnya dilakukan evaluasi apakah *output* yang dihasilkan sesuai dengan harapan dan memiliki akurasi yang baik. Jika *output* tidak sesuai dengan harapan maka model dapat disesuaikan kembali untuk menghasilkan akurasi yang lebih baik.

6. Penyusunan Laporan

Pada tahap ini, penyusunan laporan dilakukan secara paralel dengan pembuatan aplikasi. Laporan ditulis berdasarkan hasil penelitian yang dilakukan.

1.7. Sistematika Penulisan

Tahap ini berisi gambaran umum dari penelitian ini, sistematika penulisannya adalah sebagai berikut:

BAB 1: PENDAHULUAN

Bab ini meliputi latar belakang, rumusan masalah, tujuan, manfaat, ruang lingkup, metodologi penelitian, dan sistematika penulisan yang digunakan dalam penelitian ini.

BAB 2: LANDASAN TEORI

Bab ini meliputi teori-teori dan konsep yang mendukung penelitian ini. Bab ini juga berisi algoritma dan arsitektur yang diimplementasikan pada penelitian tugas akhir.

BAB 3: ANALISIS SISTEM

Bab ini meliputi analisis terhadap situasi dan kondisi yang terjadi saat ini. Analisis dilakukan menggunakan data yang dikumpulkan dari penelitian.

BAB 4: DESAIN SISTEM

Bab ini meliputi desain *user interface*, arsitektur dan model yang digunakan pada sistem.

BAB 5: IMPLEMENTASI SISTEM

Bab ini meliputi penerapan sistem dan model pada aplikasi Android untuk mengklasifikasikan penyakit pada daun mangga berbasis citra menggunakan CNN.

BAB 6: UJI COBA DAN EVALUASI

Bab ini meliputi pengujian dan evaluasi terhadap sistem pada penelitian ini. Dari hasil pengujian, dilakukan evaluasi terhadap sistem untuk mengambil kesimpulan.

BAB 7: KESIMPULAN DAN SARAN

Bab ini meliputi rangkuman penelitian dan rekomendasi yang dapat digunakan sebagai referensi pada penelitian berikutnya.

BAB 2

LANDASAN TEORI

Bab ini meliputi teori-teori dan konsep yang menunjang penelitian ini. Pada bab ini juga memuat algoritma dan arsitektur yang diimplementasikan pada penelitian tugas akhir.

2.1. Penyakit Pada Daun Mangga

Mangga yang memiliki nama ilmiah *Mangifera indica L.* adalah tanaman yang berasal dari wilayah perbatasan India dan Burma (Ide, 2013). Namun, persebarannya sampai ke Asia Tenggara salah satunya Indonesia. Mangga memiliki lebih dari 1000 spesies di dunia dan mangga dapat tumbuh di daerah tropis. Mangga banyak diminati masyarakat karena buahnya yang memiliki cita rasa manis dan buah yang tebal. Selain memiliki rasa yang lezat dan menyegarkan, buah mangga juga mengandung vitamin C, antioksidan dan nutrisi yang memberikan manfaat bagi kesehatan tubuh.

Daun mangga memiliki banyak variasi bentuk, warna dan ukuran yang menunjukkan beragamnya spesies mangga. Struktur daun mangga memiliki 2 komponen yaitu tangkai daun dan lamina daun. Lamina daun memiliki tulang dan urat yang ditutupi oleh daging daun. Susunan tulang daun menyerupai sirip ikan. Hal ini merupakan salah satu ciri untuk membedakan daun mangga dari berbagai jenis tumbuhan.

Namun demikian, mangga tak lepas dari berbagai jenis penyakit. Penyakit pada daun mangga memiliki ciri yang berbeda-beda. Berikut ini adalah tabel data 7 jenis penyakit pada daun mangga beserta contoh gambarnya (Ahmed et al., 2023).

Tabel 2.1. Data Jenis Penyakit Pada Daun Mangga

No.	Jenis penyakit	Ciri-ciri penyakit daun mangga	Gambar daun mangga
1.	<i>Anthracnose</i>	<ol style="list-style-type: none"> 1. Munculnya bercak hitam pada sisi daun mangga. 2. Daun yang terkena dampak parah, biasanya mulai melengkung. 	
2.	<i>Bacterial Canker</i>	<ol style="list-style-type: none"> 1. Terdapat bintik-bintik basah pada tangkai dan badan daun. 2. Ketika bintik-bintik sudah banyak, berubah menjadi bercak kanker yang tidak teratur. 	
3.	<i>Cutting Weevil</i>	<ol style="list-style-type: none"> 1. Daun terlihat seperti dipotong dengan gunting. 2. Biasa berdampak pada 	

		daun muda.	
4.	<i>Die Back</i>	<p>1. Daun berubah menjadi coklat dan menggulung.</p> <p>2. Diikuti dengan ranting atau dahan menjadi mati dan layu sehingga daun rontok.</p>	
5.	<i>Gall Midge</i>	<p>1. Muncul bintik-bintik seperti jerawat pada daun.</p> <p>2. Dalam keadaan berat terjadi defoliasi dan penurunan buah.</p>	
6.	<i>Powdery Mildew</i>	<p>1. Terdapat jamur pada permukaan daun.</p> <p>2. Jamur menyerupai tepung ini juga dapat muncul pada tangkai bunga dan buah.</p>	
7.	<i>Sooty Mould</i>	<p>1. Terdapat cairan manis dan lengket yang dibuat oleh serangga untuk menarik serangga lain.</p> <p>2. Tumbuh jamur pada</p>	

		<p>cairan tersebut dan perlahan menyebar ke seluruh permukaan daun.</p> <p>3. Warna daun perlahan menjadi hitam.</p>	
--	--	--	--

Dataset yang digunakan adalah citra daun mangga yang didapatkan dari *MangoLeafBD* pada repositori *Mendeley* (Ahmed et al., 2023). Dalam deskripsi *dataset* dikatakan bahwa data telah divalidasi oleh pakar pertanian dan diberi label secara manual. Pada tahapan ini, *dataset* dibagi dengan rasio 8:2, 80% digunakan untuk data pelatihan (*training*) dan 20% sebagai data pengujian (*testing*). Total 4000 citra pada *dataset*, dimana memiliki 8 kategori sehingga setiap kategori memiliki 500 citra.

Tabel 2.2. Detail Pembagian Dataset

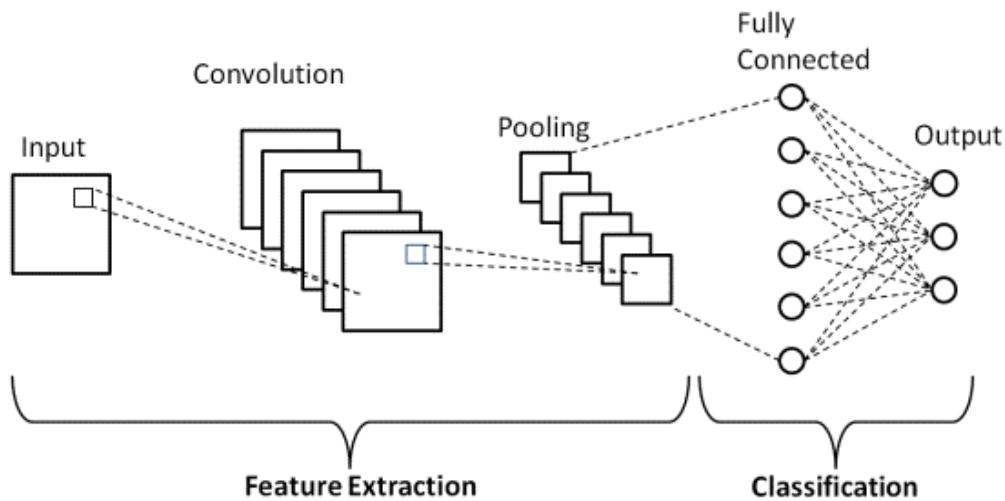
Kategori	Training (80%)	Testing (20%)	Jumlah
<i>Anthracnose</i>	400	100	500
<i>Bacterial Canker</i>	400	100	500
<i>Cutting Weevil</i>	400	100	500
<i>Die Back</i>	400	100	500
<i>Gall Midge</i>	400	100	500

<i>Healthy</i>	400	100	500
<i>Powdery Mildew</i>	400	100	500
<i>Sooty Mould</i>	400	100	500
Total Citra	3200	800	4000

2.2. *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan algoritma yang berasal dari pengembangan *MultiLayer Perceptron* (MLP) yang digunakan untuk memproses data berupa citra dua dimensi (Shinta, 2023). CNN masuk ke dalam jenis *Deep Neural Network* karena banyak diimplementasikan pada data citra dan memiliki struktur jaringan yang banyak lapis atau kedalaman jaringan yang tinggi. Kerja CNN menyerupai jaringan saraf, perbedaannya setiap lapisan dari CNN dilakukan konvolusi menggunakan masukan dari lapisan tersebut (Cahya et al., 2021).

Terdapat tiga lapisan utama pada CNN yaitu lapisan input, lapisan *hidden*, dan lapisan *output*. Pada lapisan *hidden* berisi sejumlah lapisan di dalamnya seperti lapisan konvolusi, lapisan aktivasi, lapisan *pooling* dan lapisan *fully connected*. Berikut ilustrasi dari arsitektur CNN.

**Gambar 2.1. Arsitektur CNN**

Sumber : (Phung & Rhee, 2019)

2.2.1. *Convolution Layer*

Convolution layer adalah komponen dari *hidden layer* pada arsitektur CNN. Dalam tahap *convolution layer*, dilakukan operasi konvolusi terhadap hasil dari lapisan sebelumnya. Operasi ini diaplikasikan secara berulang dan digeser ke seluruh permukaan citra. Hasil dari operasi konvolusi berupa sebuah matriks *feature map*. Rumus menghitung ukuran *feature map* sebagai berikut.

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

Keterangan:

n_{out} : Dimensi matriks *feature map*

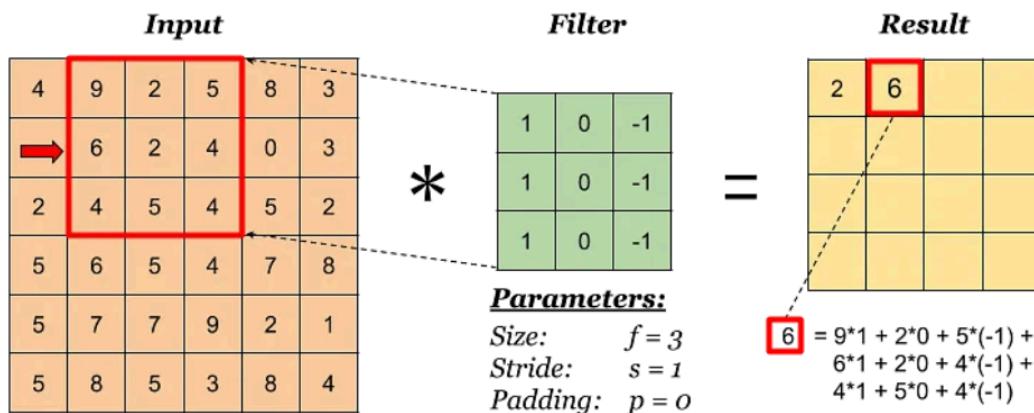
n_{in} : Dimensi matriks masukan

k : Dimensi matriks filter

p : Nilai *Padding*

s : Ukuran pergeseran (*Stride*)

Sebagai contoh, pada *input layer* gambar yang digunakan memiliki dimensi 6 x 6, filter atau kernel berukuran 3 x 3, nilai *padding* 0 dan pergeseran (*Stride*) yang diterapkan sebanyak 1. *Stride* adalah parameter yang digunakan untuk menentukan pergeseran filter saat dilakukan konvolusi, sedangkan *padding* merupakan nilai yang digunakan untuk memperoleh angka di sekitar *border* citra. Maka hasil *feature map* dapat dihitung menggunakan persamaan (2.1). Hasil perhitungan sebagai berikut, $n_{out} = (6-3+(2*0)/1)+1 = 4$. Ukuran *feature map* yang didapatkan adalah 4x4. Setelah itu lakukan operasi konvolusi menggunakan persamaan (2.2). Perhatikan ilustrasi berikut:



Gambar 2.2. Ilustrasi Proses Operasi Konvolusi

Sumber : (Kumar, 2020)

Adapun rumus untuk melakukan operasi konvolusi sebagai berikut.

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

Keterangan :

FM[i] : Matriks *feature map* ke-i

N : Matriks citra input

F : Matriks filter

bF : Nilai bias filter

j,k : Baris dan kolom pada matriks citra masukan

m,n : Baris dan kolom pada matriks filter konvolusi

2.2.2. *Activation Layer*

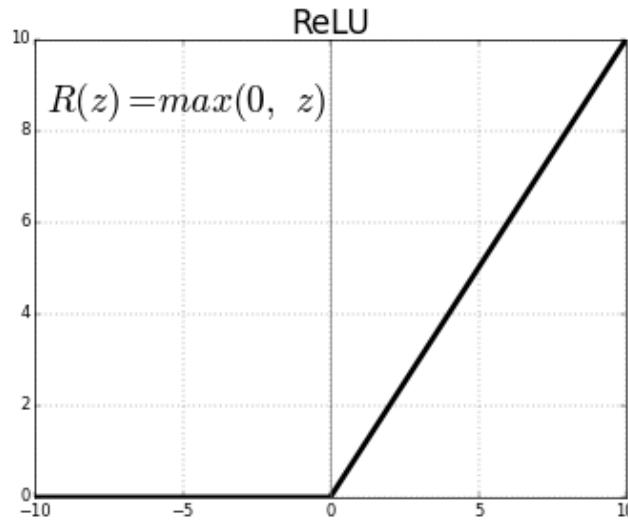
Activation layer merupakan *layer* yang digunakan untuk menghitung nilai linier atau non linier dari hasil konvolusi menggunakan fungsi aktivasi. Secara umum, fungsi aktivasi terbagi menjadi 2 yaitu fungsi aktivasi linier dan non linier.

Beberapa contoh fungsi aktivasi antara lain:

1. *ReLU* merupakan singkatan dari *Rectified Linear Unit*. Fungsi aktivasi ini merubah nilai dari piksel dengan *threshold* 0 hingga *infinity*. Piksel yang bernilai negatif atau kurang dari 0 akan diubah menjadi 0. Adapun persamaan yang dapat digunakan sebagai berikut.

$$R(z) = \max(0, z) \quad (2.3)$$

Berikut gambar dari grafik fungsi aktivasi *ReLU*.

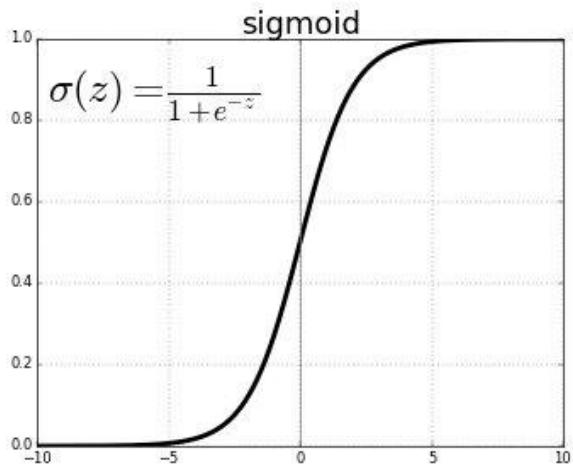


Gambar 2.3. Fungsi Aktivasi *ReLU*

2. *Sigmoid* merupakan fungsi aktivasi yang merubah nilai piksel dalam rentang 0 hingga 1. Persamaan yang dapat digunakan untuk menghitung nilai *Sigmoid* sebagai berikut.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2.4)$$

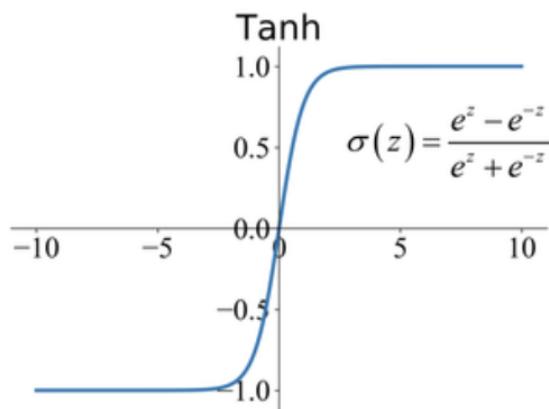
Apabila input dari piksel bernilai negatif, maka *output* yang dihasilkan 0, sebaliknya input bernilai positif maka *output* yang dihasilkan 1. Grafik fungsi aktivasi ini membentuk kurva seperti huruf S. Berikut gambar dari fungsi aktivasi *Sigmoid*.

**Gambar 2.4. Fungsi Aktivasi *Sigmoid***

3. *Tanh* merupakan fungsi aktivasi yang merubah nilai piksel dengan rentang *output* -1 hingga 1. Persamaan yang dapat digunakan untuk menghitung nilai *Tanh* sebagai berikut.

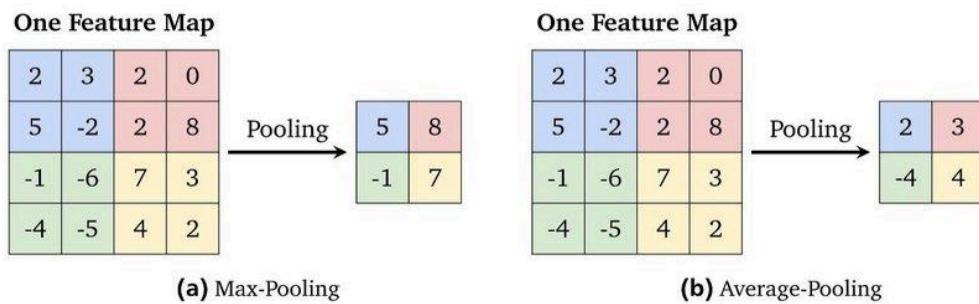
$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2.5)$$

Berikut gambar dari grafik fungsi aktivasi *Tanh*.

**Gambar 2.5. Fungsi Aktivasi *Tanh***

2.2.3. Pooling Layer

Pooling layer atau lapisan *pooling* merupakan lapisan yang digunakan untuk mengurangi ukuran matriks *feature map*. Umumnya, lapisan ini berada setelah *convolutional layer*. Terdapat 2 jenis *pooling* yang sering digunakan yaitu *max pooling* dan *average pooling*. Cara kerja lapisan *pooling* mirip dengan *convolutional layer* yaitu dengan cara bergeser secara bergantian sesuai dengan nilai *Stride*. Pergeseran dilakukan hingga seluruh area *feature map* dilewati. *Max pooling* membandingkan nilai setiap piksel pada filter dan mengambil nilai maksimal, sedangkan *average pooling* mengambil nilai rata-rata pada filter. Sebagai contoh, hasil *feature map* berdimensi 4×4 , *padding* bernilai 0 dan *stride* bernilai 2, maka dengan persamaan (2.1) didapatkan hasil $n_{out} = (4-2+(2*0)/2)+1 = 2$. Ukuran *feature map* yang didapatkan adalah 2×2 . Berikut ilustrasi operasi *max pooling* dan *average pooling*.



Gambar 2.6. Ilustrasi Proses Operasi *Pooling*

Sumber : (Guisous, 2019)

Adapun persamaan yang dapat digunakan untuk operasi *max pooling* adalah sebagai berikut.

$$\text{Max Pooling}(x) = \max(x_{[i:i+k, j:j+k]}) \quad (2.6)$$

Keterangan:

i, j : Baris dan kolom yang akan dihitung

k : Lebar blok yang akan dihitung - 1

$x_{[i : i+k, j : j+k]}$: Input x pada baris i hingga $i+k$, kolom j hingga $j+k$

$\max_{[i : i+k, j : j+k]}$: Mencari nilai maksimum dari $x_{[i : i+k, j : j+k]}$

Sementara itu, persamaan yang dapat digunakan untuk operasi *average pooling* adalah sebagai berikut.

$$\text{Avg Pooling}(x) = \text{avg}(x_{[i:i+k, j:j+k]}) \quad (2.7)$$

Keterangan:

i, j : Baris dan kolom yang akan dihitung

k : Lebar blok yang akan dihitung - 1

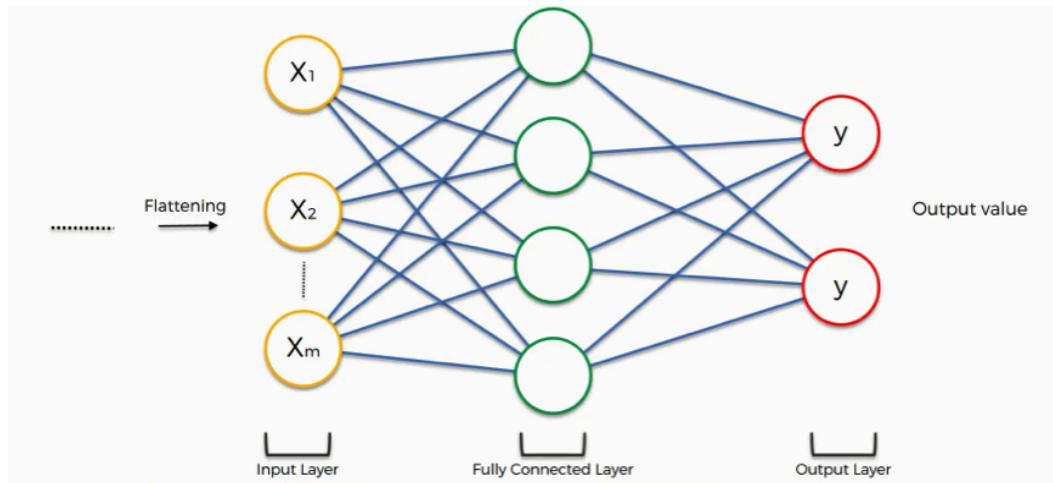
$x_{[i : i+k, j : j+k]}$: Input x pada baris i hingga $i+k$, kolom j hingga $j+k$

$\text{avg}_{[i : i+k, j : j+k]}$: Mencari nilai rata-rata dari $x_{[i : i+k, j : j+k]}$

2.2.4. *Fully Connected Layer*

Fully connected layer merupakan lapisan yang digunakan untuk mengubah dimensi data agar dapat diklasifikasikan secara linear. Dalam *fully connected layer* terbagi menjadi 3 komponen yaitu lapisan input, lapisan *fully connected* dan lapisan *output*. Lapisan input didapatkan dari lapisan sebelumnya dan dilakukan

flattening agar matriks menjadi vektor 1 dimensi. Selanjutnya, *input layer* dimasukkan ke dalam *fully connected layer* sebagai fitur untuk melakukan klasifikasi. Kemudian *output layer* menghasilkan *output* dari proses klasifikasi berupa label. Berikut ini adalah ilustrasi dari *fully connected layer*.



Gambar 2.7. Ilustrasi Fully Connected Layer

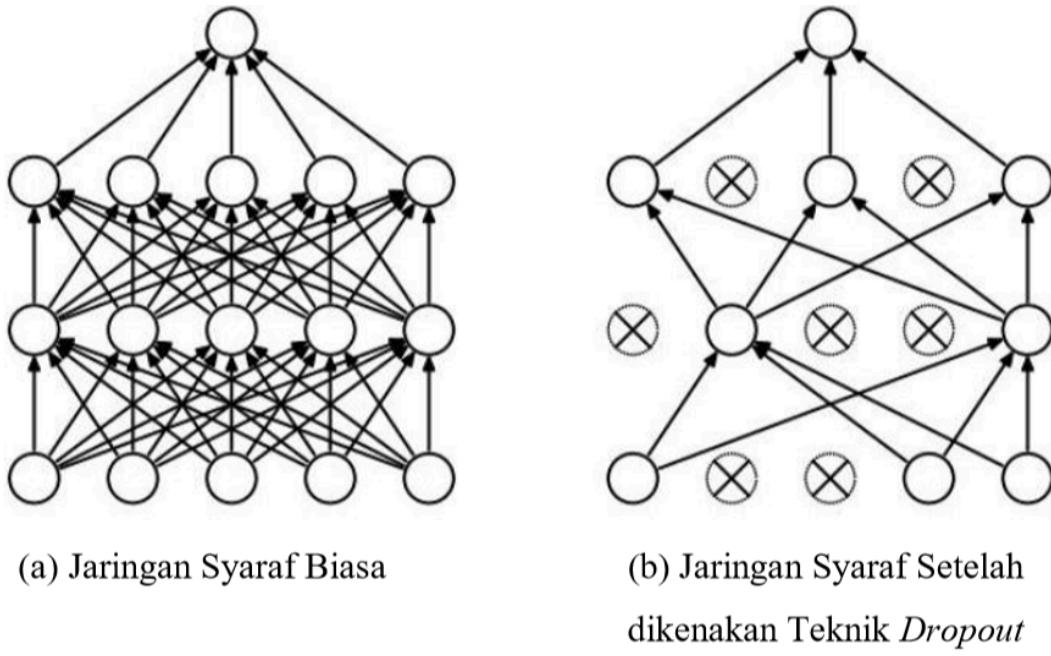
Sumber : (Team, 2018)

Gambar 2.7 merupakan ilustrasi dari *fully connected layer*. Dapat dilihat proses ini dijalankan setelah *flattening*. Dimana nilai dari *input layer* didapat dari hasil *flattening* pada proses sebelumnya. Selanjutnya, dilakukan *fully connected* dan diklasifikasikan pada *output layer*. Pada *output layer* banyak label disesuaikan dengan banyak kelas yang dimiliki.

2.2.5. Dropout Regulation

Dropout merupakan sebuah teknik untuk mencegah terjadinya *overfitting* dengan cara memilih secara acak neuron-neuron yang tidak digunakan saat proses

pelatihan. *Overfitting* adalah kondisi dimana proses pelatihan memiliki persentase yang baik, tetapi tidak sesuai dengan proses prediksi. Teknik *overfitting* ini akan berdampak pada performa model dan waktu pelatihan menjadi lebih cepat. Berikut ilustrasi sebelum dan setelah dilakukan *dropout* pada neuron.



Gambar 2.8. Ilustrasi *Dropout*

Gambar 2.8(a) merupakan ilustrasi jaringan syaraf sebelum dilakukan teknik *dropout*. Setelah dilakukan teknik *dropout* beberapa neuron dinonaktifkan seperti pada gambar 2.8(b). Jaringan menjadi lebih sederhana sehingga komputasi juga bisa semakin cepat.

2.2.6. *Softmax Classifier*

Softmax Classifier adalah sebuah fungsi logistik yang digunakan untuk mengklasifikasi lebih dari dua kelas atau kategori. *Output* yang dihasilkan oleh

softmax bernilai antara 0 sampai dengan 1. Berikut persamaan yang dapat digunakan pada *softmax classifier*.

$$f(z_r) = \frac{e^{z_r}}{\sum_{m=1}^M e^{z_m}} \quad (2.8)$$

Keterangan:

$f(z_r)$: Probabilitas setiap kategori

z_r : Nilai neuron ke-r pada

M : Jumlah kategori

Fungsi *softmax* penting karena menghasilkan output dalam bentuk probabilitas, memungkinkan model memberikan estimasi seberapa yakin terhadap setiap kategori. Ini sangat relevan dalam tugas klasifikasi multi kelas, di mana satu sampel dapat masuk ke dalam satu dari banyak kategori yang mungkin. Proses normalisasi *softmax* memastikan bahwa probabilitas yang dihasilkan sesuai dengan distribusi probabilitas, sehingga model dapat memberikan prediksi yang lebih dapat diinterpretasikan.

2.2.7. Cross Entropy Loss Function

Loss function adalah sebuah fungsi kerugian yang menunjukkan selisih prediksi dengan nilai sebenarnya. Tujuan utama *loss function* adalah untuk mengukur kesalahan dari prediksi yang dihasilkan model. Persamaan dari *loss function* sebagai berikut.

$$L = -\frac{1}{N} \sum_{n=1}^N \sum_{r=1}^M y_{r,n} \log(f(z_r)n) \quad (2.9)$$

Keterangan:

L : Nilai *loss function* (semakin mendekati 0, semakin baik)

N : Jumlah data citra

M : Jumlah kategori

$f(z_r)$: Probabilitas setiap kategori

$y_{r,n}$: Indikator biner

Fungsi ini umum digunakannya bersama dengan fungsi aktivasi *softmax* pada *output layer*. Penggunaan fungsi aktivasi softmax memastikan bahwa nilai-nilai prediksi ($f(z_r)n$) diinterpretasikan sebagai probabilitas, dan *loss* dihitung berdasarkan perbandingan antara probabilitas prediksi dan nilai sebenarnya $y_{r,n}$. Tujuan akhirnya adalah mengoptimalkan parameter model sehingga *loss* dapat diminimalkan, dan model dapat memberikan prediksi yang lebih akurat pada data yang belum pernah dilihat sebelumnya.

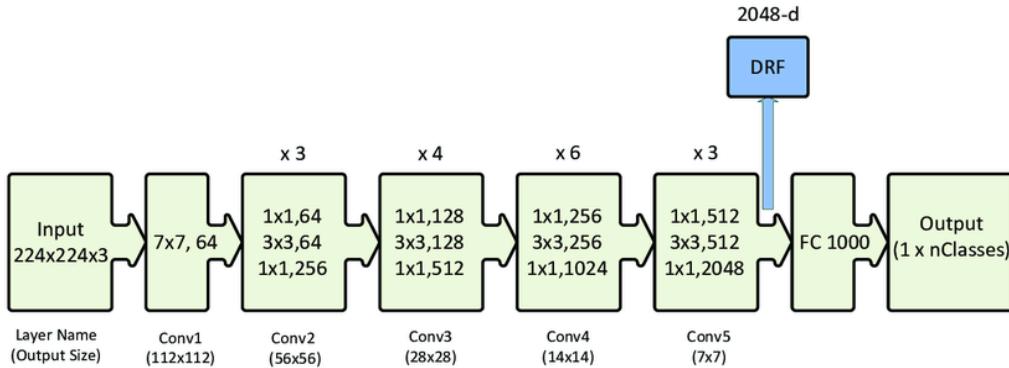
2.3. Arsitektur *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan jenis arsitektur neural network yang dirancang khusus untuk memproses data grid, seperti gambar atau citra. CNN memanfaatkan operasi konvolusi, lapisan *pooling*, dan lapisan *fully connected* untuk mengekstraksi dan memahami pola dalam data spasial. Dalam operasi konvolusi, filter kecil digunakan untuk mendeteksi fitur-fitur spesifik dalam gambar, menghasilkan peta fitur. Lapisan *pooling* digunakan untuk mereduksi dimensi spasial dari peta fitur, dengan metode seperti *max pooling* atau *average pooling*.

Convolutional Neural Network (CNN) memiliki beberapa arsitektur seperti ResNet50, VGG16, MobileNet, dan Inception-v3. Pada penelitian ini dipilih keempat arsitektur tersebut karena keunggulan dari masing-masing arsitektur. Keunggulan dari arsitektur tersebut dinilai cocok untuk tugas mengklasifikasikan citra.

2.3.1. Residual Network 50

Residual Network 50 atau disingkat ResNet50 merupakan arsitektur model CNN yang memperkenalkan konsep *shortcut connections*. Konsep *shortcut connections* yaitu melewati *layer* untuk menghindari masalah hilangnya *gradient*. Pada tahun 2015, arsitektur ResNet50 pernah memenangkan kompetisi *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) dengan nilai *error rate* 3.57 dan jumlah parameter sebanyak 25 juta (Krishna & Kalluri, 2019). Arsitektur ResNet50 dikenal dapat mengatasi masalah *vanishing gradient* dengan menggunakan blok residu yang membuatnya juga lebih mudah untuk dilatih dengan jaringan yang sangat dalam. Arsitektur ResNet50 memiliki 50 *layer* yang terdiri dari 48 *layer* konvolusi, 1 *max pool* dan 1 *average pool layer*. Berikut ilustrasi arsitektur ResNet50 dapat dilihat pada gambar 2.7.



Gambar 2.9. Ilustrasi Arsitektur ResNet50

Sumber : (Mahmood et al., 2020)

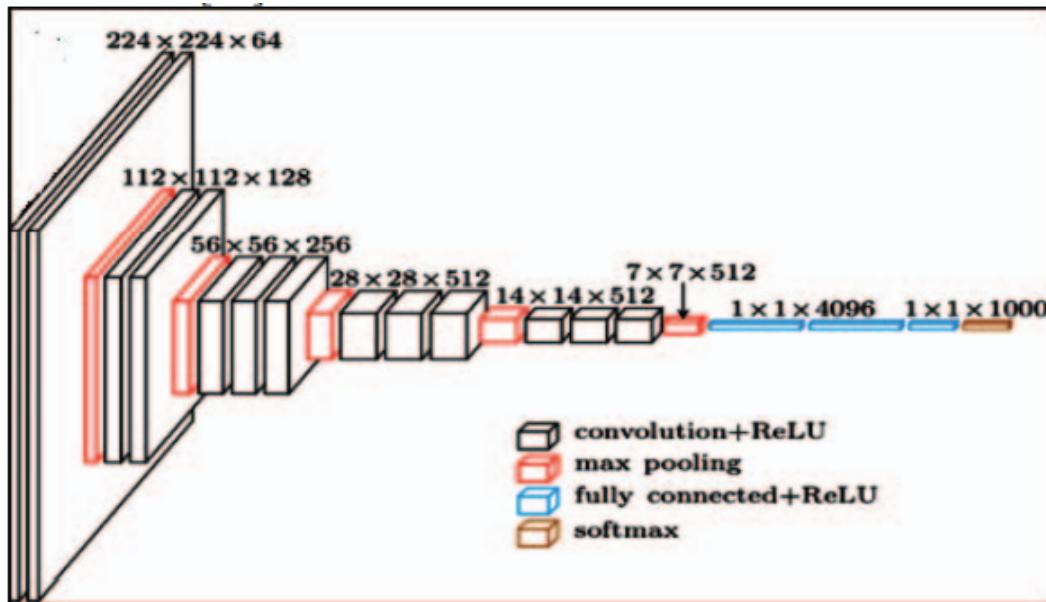
Lapisan konvolusi kelima memiliki dimensi 7 x 7 x 2048 dan digunakan sebagai input pada lapisan *fully connected*. Sebelum itu, *array* dikonversi menjadi vektor 2048 menggunakan lapisan *max pool* dan vektor diekstraksi untuk *Deep Residual Feature* (DRF). *Output* disesuaikan dengan jumlah kelas yang dimiliki 1 x nClasses.

2.3.2. Visual Geometry Group 16

Visual Geometry Group 16 disingkat VGG16 merupakan sebuah arsitektur model CNN dengan kedalaman 16 *layer* pada konfigurasinya (Miftahuddin & Adani, 2022). Arsitektur ini memiliki susunan layer yang sangat sederhana dengan lapisan konvolusi bertumpuk dan *max pooling*. Pada tahun 2014, arsitektur ini menjuarai kontes *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) sebagai *1st Runner-up Image Classification* dan pemenang

Image Localization. VGG16 memiliki 13 lapisan konvolusi dan 3 lapisan *fully connected*.

Pada lapisan konvolusi pertama menerima input citra dengan dimensi 224 x 224, lapisan ini memiliki dimensi filter 3 x 3 dan *max pooling* berukuran 2 x 2. Lapisan konvolusi 1 & 2 memiliki filter sebanyak 64, lapisan 3 & 4 mempunyai 128 filter, lapisan 5,6 & 7 memiliki 256 filter, lapisan 8 hingga 13 memiliki 512 filter. Fungsi aktivasi yang digunakan pada lapisan konvolusi adalah fungsi aktivasi ReLU. Kemudian dilakukan proses *max pooling* setelah lapisan 2,4,7,10 dan 13. Lapisan *max pooling* yang terakhir berhubungan dengan lapisan *fully connected* pertama. Terdapat 3 lapisan *fully connected* yang memiliki ukuran berturut-turut yaitu 4096, 4096 dan 1000. Setiap neuron menerima input dari neuron *layer* sebelumnya. *Fully connected layer* terakhir terhubung dengan *classifier* yang menggunakan fungsi *softmax*. Arsitektur VGG16 menggunakan dimensi filter berukuran 3 x 3. Berikut gambaran dari arsitektur VGG16.



Gambar 2.10. Arsitektur VGG16

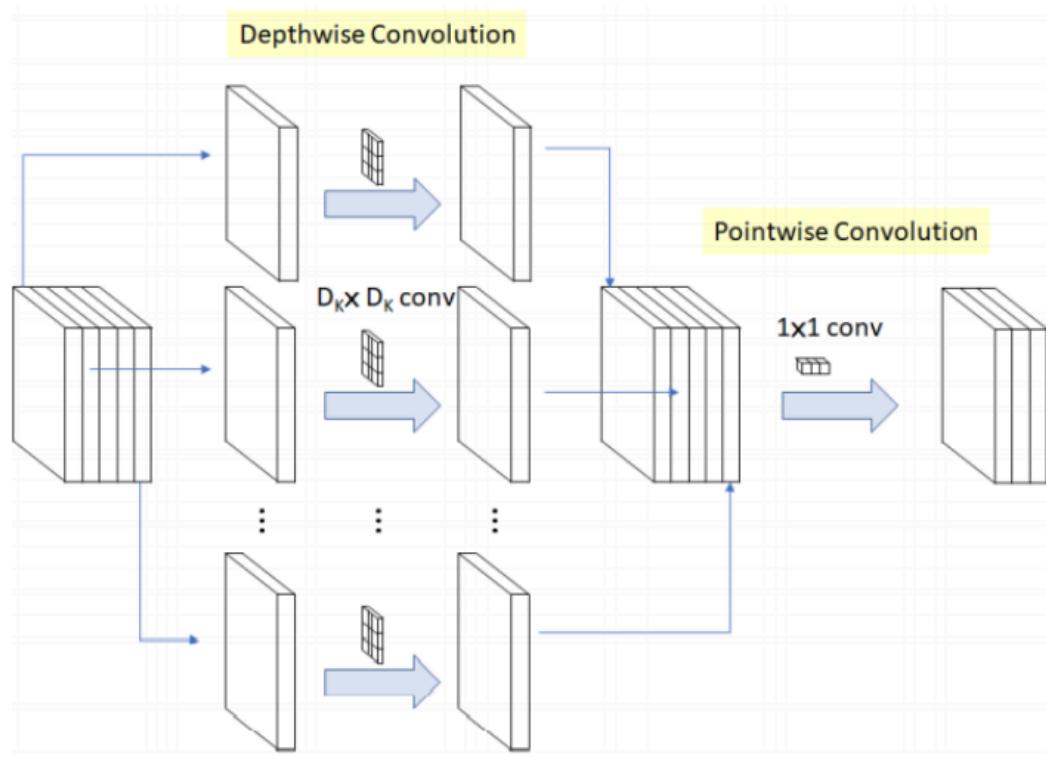
Sumber : (Kaur & Gandhi, 2019)

2.3.3. MobileNet

MobileNet merupakan arsitektur CNN yang dapat digunakan untuk mengatasi sumber komputasi yang berlebihan (Jauhari, 2022). Arsitektur MobileNet dirancang khusus untuk aplikasi *mobile* dan perangkat dengan daya komputasi yang terbatas, arsitektur ini juga menggunakan nilai kedalaman filter yang sesuai dengan kedalaman citra input. Sehingga model relatif lebih ringan dan cepat yang menjadikannya pilihan yang baik untuk perangkat *mobile*. Konvolusi dalam arsitektur MobileNet ada 2 yaitu *depthwise convolution* dan *pointwise convolution*. Pada arsitektur ini model dibentuk berdasarkan konvolusi yang dipisah-pisah secara mendalam (*depthwise separable convolution*) menjadi

konvolusi yang menguraikan *standar convolution* menjadi *depthwise convolution* dan konvolusi 1×1 atau disebut *pointwise convolution* (Howard et al., 2017).

Cara kerja arsitektur MobileNet yaitu *depthwise convolution* menerapkan satu filter pada setiap *channel* input, lalu *pointwise convolution* menggabungkan hasil dari *depthwise convolution* menggunakan konvolusi 1×1 . Selanjutnya, *standard convolution* melakukan filter dan menggabungkan input menjadi sebuah *output* yang baru. *Depthwise separable convolution* terbagi menjadi 2 bagian dengan fungsi berbeda yaitu *layer* filter dan *layer* untuk menggabungkan. Komputasi dan ukuran model dapat dikurangi secara drastis dengan pembagian *layer* ini. Ilustrasi dari *depthwise separable convolution* sebagai berikut.



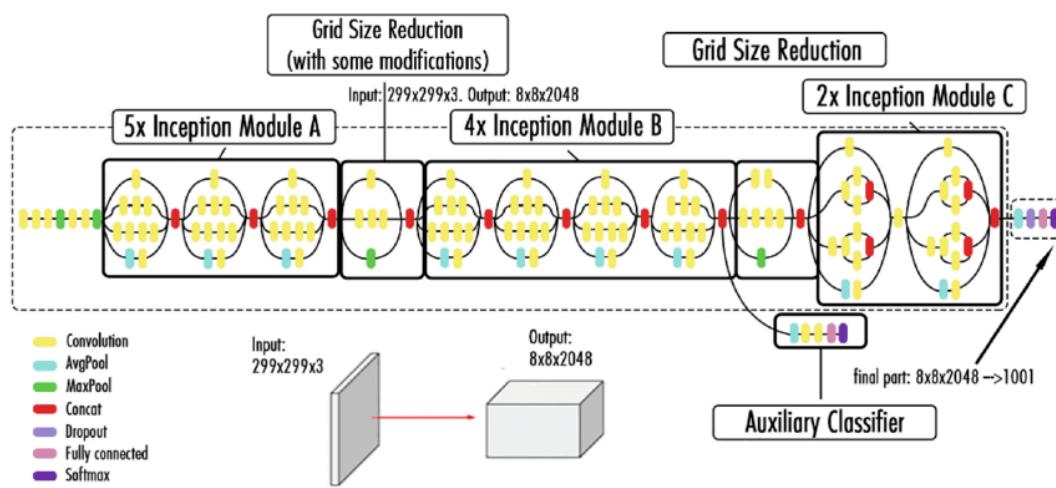
Gambar 2.11. Ilustrasi Depthwise Separable Convolution

Sumber : (Jauhari, 2022)

Seluruh lapisan pada arsitektur MobileNet menggunakan fungsi aktivasi ReLU, kecuali lapisan terakhir yang terhubung dan tidak memiliki non linier tetapi termasuk lapisan *softmax*.

2.3.4. GoogleNet (Inception-v3)

GoogleNet merupakan arsitektur CNN yang dikembangkan oleh Google dan telah dilatih dengan lebih dari satu juta citra. Inception-v3 merupakan pengembangan dari GoogleNet dengan menambahkan faktorisasi pada proses konvolusi. Arsitektur ini dikenal dengan modul *inception* yang kompleks dengan menggunakan berbagai ukuran kernel. Sehingga dapat menangani objek dengan ukuran dan tingkat kompleksitas yang berbeda serta efektif untuk tugas klasifikasi dan deteksi objek. Pada tahun 2015 Inception-v3 menjadi 1st *Runner-up Image Classification* pada kontes *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) dalam mengenali 1000 objek kelas ImageNet. Berikut gambaran dari arsitektur GoogleNet (Inception-v3).



Gambar 2.12. Arsitektur GoogleNet (Inception-v3)

Sumber : (Zorgui et al., 2020)

Inception-v3 memiliki kedalaman 42 *layer*, pada lapisan pertama (Module A) menggunakan filter dengan dimensi 3 x 3 untuk mengurangi komputasi dan parameter serta meningkatkan kinerja (Zorgui et al., 2020). Module B membagi setiap lapisan konvolusi n x n menjadi 1 x n dan n x 1. Module C memperluas filter untuk mengurangi informasi yang hilang.

2.4. Hyperparameter Tuning

Hyperparameter tuning merupakan teknik untuk mengoptimalkan model CNN dengan pemilihan parameter yang tepat. *Hyperparameter* adalah konfigurasi yang didefinisikan di luar model dan tidak dapat dipelajari oleh model sendiri selama proses pelatihan. Berikut beberapa contoh *hyperparameter* dalam CNN yang sering disesuaikan selama proses tuning:

- Batch Size

Batch size adalah jumlah citra yang dilatih dalam setiap iterasi. Semakin besar ukuran *batch*, maka membutuhkan memori yang semakin besar.

- Epoch

Epoch digunakan untuk menentukan berapa kali citra dilewati oleh model. Jumlah *epoch* yang terlalu kecil mungkin dapat mengurangi performa belajar, sementara jumlah *epoch* yang terlalu besar juga dapat menyebabkan *overfitting*.

- Learning Rate

Learning rate berfungsi untuk mengontrol seberapa besar langkah pembelajaran yang diambil oleh algoritma optimasi dalam proses pelatihan.

Apabila nilai *learning rate* terlalu kecil dapat menyebabkan proses pelatihan menjadi lambat, sedangkan nilai *learning rate* terlalu besar menyebabkan model sulit mencapai solusi optimal (konvergensi).

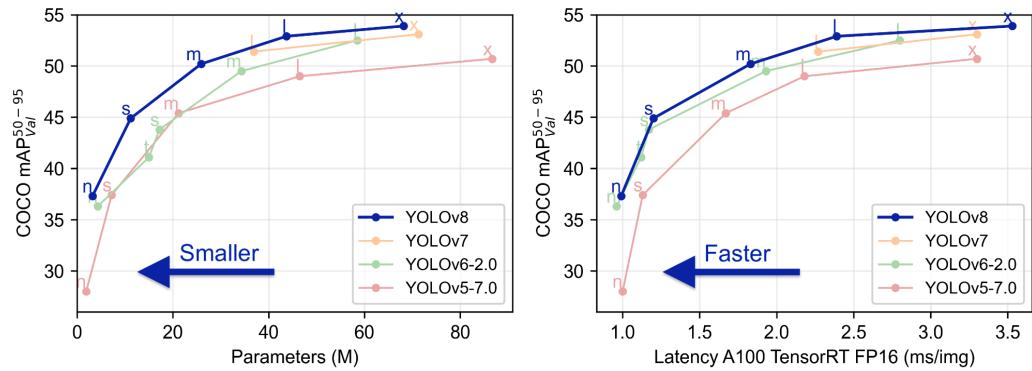
- Optimizer

Optimizer merupakan *hyperparameter* dengan algoritma optimasi yang digunakan untuk menemukan model yang optimal. Beberapa contoh *optimizer* seperti *Stochastic Gradient Descent* (SGD), *RMSprop*, dan *Adam*.

Hyperparameter tuning melibatkan eksperimen dengan berbagai kombinasi nilai untuk hyperparameter ini guna menemukan konfigurasi yang menghasilkan kinerja model terbaik pada data validasi atau data uji. Teknik seperti grid search, random search, atau optimasi bayesian sering digunakan untuk mencari kombinasi terbaik dari hyperparameter.

2.5. You Only Look Once

You Only Look Once (YOLO) versi ke-8 merupakan model tercanggih yang dibangun berdasarkan penyempurnaan YOLO sebelumnya untuk meningkatkan kinerja dan fleksibilitas. YOLOv8 dirancang agar cepat dan akurat dalam mendekripsi, melacak objek dan klasifikasi gambar. Pada penelitian ini YOLOv8 digunakan untuk mengenali objek daun mangga sebelum masuk ke tahap klasifikasi penyakit pada daun mangga. Berikut grafik perbandingan performa YOLO dari versi ke-5 hingga 8.

**Gambar 2.13. Grafik Perbandingan Performa YOLO**

Sumber : (Ultralytics, 2023)

Grafik perbandingan performa YOLO pada Gambar 2.11 menunjukkan YOLOv8 memiliki *latency* tertinggi dengan parameter yang lebih sedikit. Terdapat beberapa varian pada YOLOv8 yaitu YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l dan YOLOv8x. Berikut gambar perbandingan performa varian YOLOv8.

Model	size (pixels)	mAP _{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Gambar 2.14. Perbandingan Performa Varian YOLOv8

Sumber : (Ultralytics, 2023)

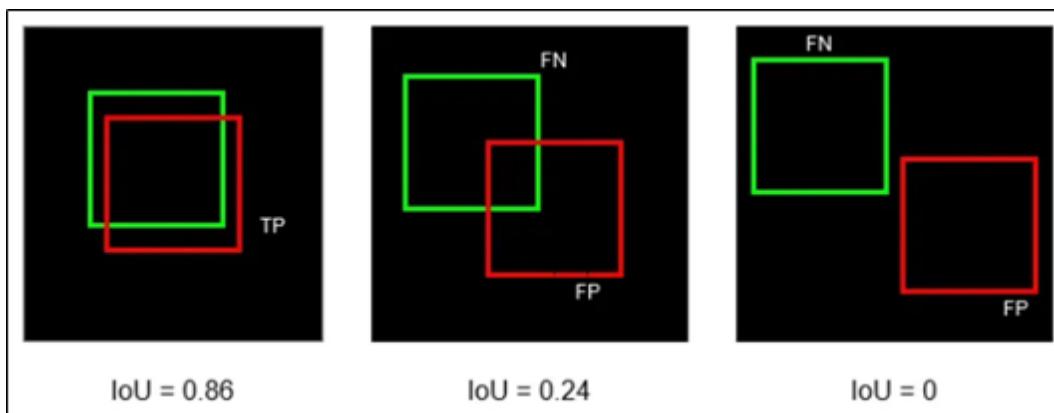
Dari perbandingan performa varian YOLOv8 pada Gambar 2.12 menunjukkan varian *nano* (n) dan *small* (s) memiliki nilai *mean average precision* (mAP) yang paling rendah dibandingkan dengan varian lain dengan tingkat kecepatan yang relatif tinggi. Selain itu, varian *medium* (m) memiliki nilai *mean average precision* (mAP) dan kecepatan yang optimal. Sedangkan varian *large* (l) dan *extra large* (xl) memiliki nilai mAP yang paling tinggi dengan kecepatan yang lebih lama dibandingkan varian lainnya. Evaluasi YOLO dapat menggunakan beberapa metode uji antara lain *intersection over union* (IoU), *recall*, *precision*, *accuracy* dan *mean average precision* (mAP).

2.5.1. *Intersection over Union*

Intersection over Union (IoU) merupakan rasio antara luas daerah yang bertumpuk (*overlap*) dari dua kotak pembatas (*bounding box*) dengan luas daerah gabungan (*union*) dari kedua kotak pembatas tersebut. Berikut persamaan yang dapat digunakan untuk menghitung IoU.

$$IoU = \frac{\text{Area of overlap}}{\text{Area of union}} \quad (2.10)$$

IoU mengambil rasio area bertumpuk antara *bounding box* dan kotak prediksi terhadap gabungan dari luas daerah mereka. Nilai IoU berkisar antara 0 hingga 1, nilai 0 menunjukkan tidak terdapat perpotongan antara kedua kotak, dan nilai 1 menunjukkan kedua kotak bertumpuk sempurna. Berikut ilustrasi dari *bounding box* pada IoU.



Gambar 2.15. Ilustrasi *Bounding Box* Pada IoU

2.5.2. *Mean Average Precision*

Mean Average Precision (mAP) merupakan metrik untuk mengevaluasi hasil deteksi model YOLO (Maleh et al., 2023). *Mean Average Precision* menggabungkan hasil *Average Precision* (AP) dari setiap kelas untuk memberikan nilai rata-rata secara keseluruhan. AP sendiri mengukur sejauh mana model mampu memberikan hasil deteksi yang presisi dan *recall* yang tinggi untuk suatu kelas. Persamaan yang digunakan untuk menghitung mAP sebagai berikut.

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \quad (2.11)$$

Keterangan:

n = banyak kelas

AP_k = rata-rata *precision* kelas k

AP (*average precision*) didapatkan dari perhitungan akurasi klasifikasi objek dari *recall* dan *precision*. Persamaan yang digunakan untuk menghitung AP sebagai berikut.

$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1) * Precisions(k)] \quad (2.12)$$

Average Precision (AP) merupakan metrik evaluasi yang lebih terperinci dan sering digunakan dalam tugas deteksi objek. AP dihitung dari *precision-recall curve* yang menunjukkan hubungan antara presisi (*precision*) dan *recall* pada berbagai nilai ambang batas (*threshold*). Pada setiap kelas, AP dihitung sebagai area di bawah kurva *precision-recall*. Semakin besar nilai AP, semakin baik performa model dalam mendekripsi objek pada kelas tersebut. AP kemudian diambil rata-rata untuk semua kelas, dan hasilnya digunakan untuk menghitung mAP.

2.6. Library

Dalam penelitian ini *library* digunakan untuk membantu tugas-tugas tertentu. *Library* menyediakan fungsi-fungsi atau modul-modul yang dapat dipanggil untuk memudahkan pengembangan perangkat lunak. Dimana penggunaannya cukup mudah tanpa harus menulis ulang kode dari awal, dan dapat meningkatkan efisiensi dan produktivitas pengembangan.

Library seperti OpenCV, Tensorflow, dan Scikit-learn menyediakan alat dan fungsi yang kuat untuk memecahkan masalah tertentu dalam berbagai bidang pengembangan perangkat lunak. Dengan memanfaatkan library-library ini, pengembang dapat menghemat waktu dan usaha, serta membangun solusi yang lebih andal dan efisien. Berikut beberapa *library* yang digunakan.

1. OpenCV

Open Source Computer Vision Library (OpenCV) merupakan *library python* yang berfungsi untuk memproses dan menganalisis gambar (Darusalam, 2021). *Library* ini bersifat gratis (*open source*) yang dirancang oleh intel pada tahun 1999. OpenCV dapat digunakan oleh beberapa bahasa pemrograman antara lain bahasa C, C++, Python, dan Java. Dalam penelitian ini OpenCV digunakan menggunakan bahasa Python.

2. Tensorflow

Tensorflow merupakan *open source framework* yang digunakan untuk mengembangkan, melatih dan mendeteksi objek dalam *deep learning* (Sergeev & Del Balso, 2018). *Tensorflow* menggunakan teknik optimasi kompilasi untuk menggabungkan aljabar komputasi yang meliputi banyak perhitungan matematika. Berikut beberapa fitur utama yang terdapat dalam tensorflow (Dewi, 2018):

- Mendefinisikan, mengoptimalkan, dan menghitung secara matematis dengan melibatkan *array multidimension* (tensors).
- Pemrograman teknik machine learning dan pendukung jaringan syaraf.

- Pemakaian GPU (Graphics Processing Unit) yang efisien, mengotomasi manajemen dan optimalisasi memori yang sama terhadap data yang digunakan. Tensorflow dapat menulis kode yang sama dan menjalankannya di CPU atau GPU. Khususnya tensorflow mengetahui bagian yang harus dipindahkan ke GPU.
- Memiliki Skalabilitas komputasi yang tinggi terhadap kumpulan data yang besar pada mesin.

3. Scikit-learn

Scikit-learn (SkLearn) merupakan *library python* yang dapat melakukan evaluasi pada *machine learning* menggunakan *confusion matrix* (Hao & Ho, 2019). SkLearn digunakan untuk mengevaluasi hasil prediksi dengan label sebenarnya (*true positif*). Evaluasi ini meliputi akurasi, presisi, *recall* dan *f1-score*.

2.7. Evaluasi

Evaluasi merupakan tahap untuk mengukur performa dari suatu model. Saat mengukur performa dari model, kurang tepat jika melihat dari sisi akurasi saja. Karena akurasi bisa tidak sesuai apabila terjadi ketidakseimbangan data pada kelas, dimana distribusi data pada kelas tidak merata. Pengukuran performa lebih akurat jika mempertimbangkan akurasi dan nilai *loss*. Nilai *loss* adalah pengukuran seberapa baik model dalam melakukan prediksi terhadap data pelatihan dibandingkan nilai sebenarnya.

Confusion matrix adalah matriks $N \times N$ yang digunakan untuk mengevaluasi model klasifikasi, dimana N adalah jumlah kategori. *Confusion matrix* digunakan untuk membandingkan nilai prediksi dengan nilai sebenarnya. Berikut penjelasan dari simbol yang digunakan pada persamaan 2.13 hingga 2.16, *True Positive* (TP) adalah jumlah sampel positif yang diklasifikasi benar sebagai positif, *True Negative* (TN) adalah jumlah sampel negatif yang diklasifikasi benar sebagai negatif, *False Positive* (FP) adalah jumlah sampel negatif yang salah diklasifikasi sebagai positif, dan *False Negative* (FN) adalah jumlah sampel positif yang salah diklasifikasi sebagai negatif. Berikut nilai-nilai yang digunakan untuk mengevaluasi model.

1. *Accuracy* digunakan untuk menghitung probabilitas model melakukan klasifikasi dengan benar. Persamaan yang digunakan untuk menghitung *accuracy* sebagai berikut.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.13)$$

2. *Precision* digunakan untuk menghitung presisi dari keseluruhan hasil prediksi positif. Berikut persamaan untuk menghitung *precision*.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.14)$$

3. *Recall* digunakan untuk menghitung jumlah data yang diklasifikasikan dengan benar. Persamaan yang dapat digunakan untuk menghitung *recall* sebagai berikut.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.15)$$

4. *F1-Score* berfungsi untuk membandingkan nilai rata-rata dari *precision* dan *recall* yang dihitung bobotnya. Rentang nilai dari perhitungan *f1-score* adalah 0 sampai dengan 1. Nilai terbaik dari perhitungan *f1-score* adalah 1 dan terburuk adalah 0. Berikut persamaan yang dapat digunakan untuk menghitung *f1-score*.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (2.16)$$

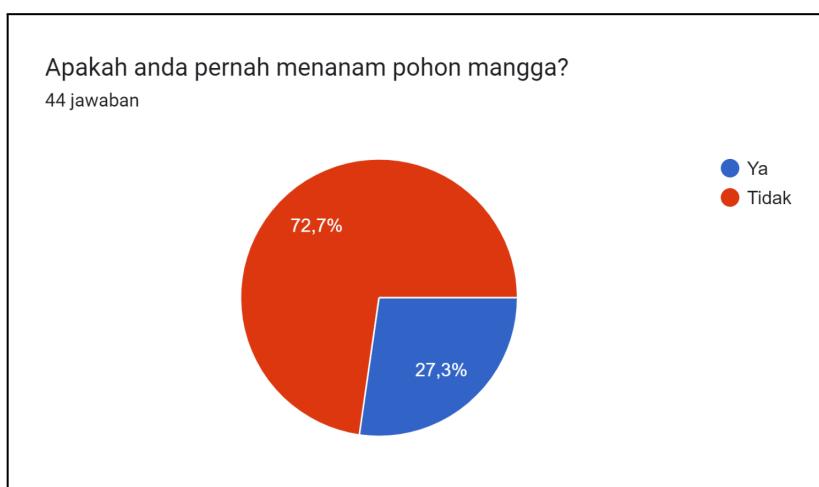
BAB 3

ANALISIS SISTEM

Bab ini meliputi analisis terhadap situasi saat ini, identifikasi masalah, analisis sistem serupa dan analisis kebutuhan sistem berdasarkan pengumpulan data yang diperoleh dari penelitian.

3.1. Analisis Situasi Saat Ini

Tahap ini berisi analisis terhadap situasi dan kondisi saat ini. Analisis dilakukan secara tertulis melalui kuesioner yang disebarluaskan kepada masyarakat umum. Kuesioner dalam penelitian ini mendapatkan 44 responden. Kriteria responden terbagi menjadi 2 yaitu responden yang pernah menanam pohon mangga dan responden yang tidak pernah menanam pohon mangga. Berikut ini hasil dari kuesioner yang telah disebarluaskan.



Gambar 3.1. Kriteria Responden

Berdasarkan hasil data yang didapatkan dari 44 responden pada gambar 3.1. Terdapat 72,7% atau 32 orang yang tidak pernah menanam mangga, sementara 27,3% atau 12 orang lainnya pernah menanam mangga. Pertanyaan ini untuk mengetahui kriteria dari target penelitian ini. Dapat disimpulkan masyarakat yang tidak pernah menanam mangga yang lebih dominan.

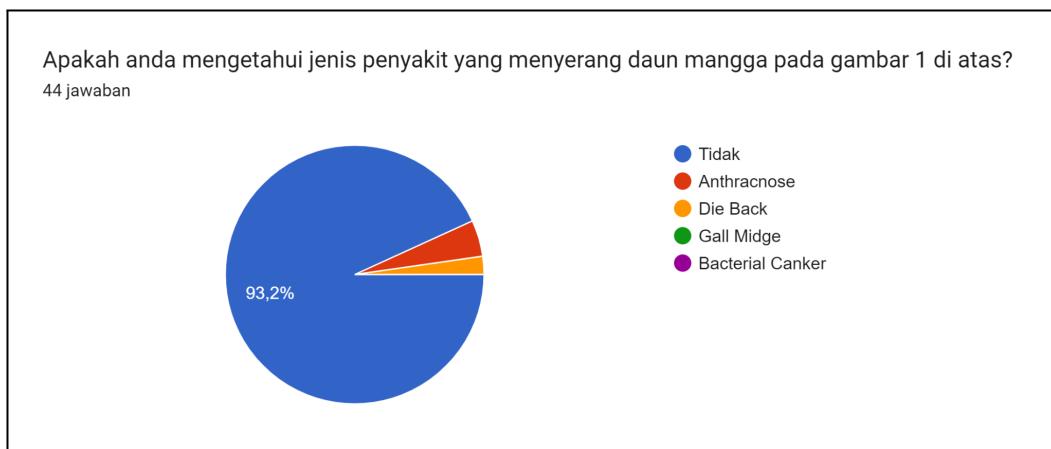
Pertanyaan selanjutnya diberikan 2 citra daun mangga yang terserang penyakit. Citra ini merupakan daun mangga yang terserang penyakit *Anthracnose*. Berikut tampilan dari citra tersebut.

Gambar 1



Gambar 3.2. Penyakit *Anthracnose* Pada Daun Mangga

Responden diberikan opsi dan diminta untuk memilih jawaban dari penyakit yang dialami oleh daun mangga pada gambar 3.2. Berikut diagram dari jawaban responden.



Gambar 3.3. Diagram Jawaban Responden Pada Penyakit *Anthracnose*

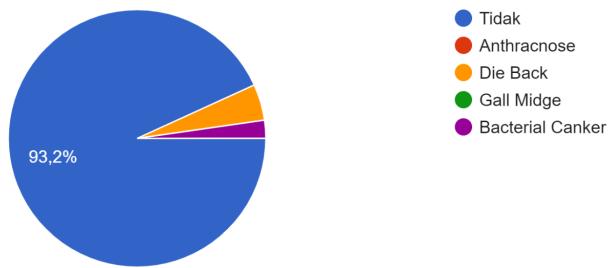
Berdasarkan diagram pada gambar 3.3 hanya 4,5% atau 2 responden saja yang menjawab dengan benar daun mangga tersebut terkena penyakit *anthracnose*, sementara 2,3% atau 1 responden jawabannya salah yaitu *die back*. 93,2% atau 41 responden lainnya tidak mengetahui jenis penyakit pada daun mangga tersebut. Kemudian pertanyaan yang mirip dengan sebelumnya, yang membedakan adalah citra penyakit pada daun mangga. Berikut tampilan citra penyakit *bacterial canker* pada daun mangga yang ditampilkan kepada responden.

Gambar 2

**Gambar 3.4. Penyakit *Bacterial Canker* Pada Daun Mangga**

Responden diberi opsi seperti sebelumnya yaitu memilih penyakit yang dialami daun mangga pada gambar 3.4. Berikut diagram dari jawaban responden.

Apakah anda mengetahui jenis penyakit yang menyerang daun mangga pada gambar 2 di atas?
44 jawaban

**Gambar 3.5. Diagram Jawaban Responden Pada Penyakit *Bacterial Canker***

Berdasarkan diagram pada gambar 3.5, data menunjukkan 2,3% atau 1 responden berhasil menjawab dengan benar. Sementara 4,5% atau 2 responden jawabannya salah yaitu *die back*. 93,2% atau 41 responden lainnya tidak mengetahui jenis penyakit yang dialami daun mangga pada gambar 3.4.

Pertanyaan berikutnya mencari tahu, apakah responden mengalami kesulitan saat mengklasifikasikan penyakit pada daun mangga. Berikut tampilan pertanyaan beserta diagram jawaban.



Gambar 3.6. Diagram Kesulitan Responden

Berdasarkan gambar 3.6 data menunjukkan 100% atau 44 responden mengalami kesulitan saat mengklasifikasikan penyakit pada daun mangga. Pertanyaan berikutnya untuk mencari tahu kesulitan responden dalam mengklasifikasikan penyakit pada daun mangga. Dari hasil jawaban responden mayoritas mengalami kesulitan karena kurangnya pengetahuan terkait penyakit pada daun mangga dan juga ciri-ciri dari penyakit tersebut.

3.2. Analisis Penelitian Serupa

Tahap ini berisi analisis terhadap penelitian serupa yang ada sebelumnya. Analisis ini digunakan untuk menemukan permasalahan yang ada dari penelitian terdahulu. Berikut ini ringkasan pembahasan dari beberapa penelitian tersebut.

1. Pendiagnosa Daun Mangga Dengan Model *Convolutional Neural Network* (Ayu et al., 2021)

Penelitian ini mendekripsi penyakit/bakteri pada daun mangga menggunakan algoritma CNN. Dataset yang digunakan didapatkan dari Kaggle dengan total 435 citra daun mangga yang terbagi atas data *training* dan data *testing*. Citra dataset terdiri dari 2 kategori yaitu *diseased* dan *healthy*. Pada tahap *training epoch* yang digunakan sebanyak 100 dan penelitian tersebut memperoleh tingkat akurasi sebesar 0,96 atau 96%, nilai ini sudah cukup akurat dalam mendiagnosis daun mangga yang terserang penyakit dan daun mangga yang sehat.

2. Sistem Diagnosis Penyakit Tumbuhan Mangga Menggunakan Metode *Naive Bayes* (Effendi et al., 2019)

Dalam penelitian ini, sistem menggunakan input berupa *checkbox* dan diberi label gejala yang dialami oleh daun mangga. Penelitian ini melakukan uji coba terhadap 30 data dan mendapat hasil yang sesuai sebanyak 28. Sehingga memperoleh akurasi sebesar 93,3% dalam mendiagnosis penyakit mangga.

3. Sistem Diagnosa Penyakit Tanaman Mangga Menggunakan Metode *Bayesian Network* (Herdianto et al., 2019)

Penelitian ini menggunakan metode *bayesian network* untuk melakukan diagnosa penyakit pada tanaman mangga. Input yang diperlukan yaitu gejala yang dialami oleh tanaman mangga. Gejala diinputkan melalui *checkbox* yang diberikan label. Saat pengujian akurasi, sistem diuji dengan 32 data pengujian. Hasil pengujian menunjukkan akurasi sebesar 87,5%, dimana terdapat 28 data yang sesuai dan 4 data yang tidak sesuai. Berikut adalah tabel perbandingan dari beberapa penelitian sebelumnya yang serupa.

Tabel 3.1. Perbandingan Penelitian Serupa

	Input	Metode	Jumlah Kategori	Akurasi
Penelitian 3.2.1. (Ayu et al., 2021)	Citra	<i>Convolutional Neural Network</i>	2 yaitu diseased, healthy.	96%
Penelitian 3.2.2. (Effendi et al., 2019)	Gejala dalam bentuk <i>checkbox</i>	<i>Naive Bayes</i>	12 yaitu ulat penggorok buah, lalat buah, penggerek buah, bintil daun mangga, ulat perusak daun, wereng mangga,	93,3%

			penggerek pucuk, penggerek cabang/batang, antraknosa, penyakit kulit, embun jelaga, kudis buah.	
Penelitian 3.2.3. (Herdianto et al., 2019)	Gejala dalam bentuk <i>checkbox</i>	<i>Bayesian Network</i>	12 yaitu ulat penggorok buah, lalat buah, penggerek buah, bintil daun mangga, ulat perusak daun, wereng mangga, penggerek pucuk, penggerek cabang atau batang, antraknosa, penyakit kulit, embun jelaga, kudis buah.	87,5%

3.3. Identifikasi Masalah

Berdasarkan analisis penelitian serupa ditemukan beberapa kekurangan pada penelitian sebelumnya yang dilakukan oleh Ayu et al. (2021) penelitian tersebut menggunakan input citra, hasilnya tidak spesifik dalam mengklasifikasikan penyakit daun mangga karena *output* yang dihasilkan yaitu sehat dan terserang penyakit. Kemudian, penelitian lain yang dilakukan oleh Herdiyanto et al. (2019) menghasilkan *output* spesifik berupa label penyakit buah mangga, tetapi masih menggunakan input gejala berupa *checkbox* dan akurasi kurang dari 90% dengan metode *bayesian network*.

Pada tugas akhir ini menggunakan input berupa citra dan *output*-nya label penyakit pada daun mangga secara spesifik. Kemudian masalah yang muncul dari analisis kondisi saat ini adalah kurangnya pengetahuan terkait penyakit pada daun mangga dan juga ciri-ciri dari penyakit tersebut. Harapannya dengan mengimplementasikan metode CNN bisa mendapatkan akurasi lebih dari 90% dengan performa yang sangat baik dalam mengklasifikasikan penyakit pada daun mangga. Sistem juga diharapkan dapat memberikan edukasi penyakit pada daun mangga dan juga ciri-ciri dari penyakit tersebut kepada pengguna.

Dari penjelasan sebelumnya, dapat disimpulkan bahwa responden sedang mengalami kesulitan sebagai berikut.

1. Responden kesulitan dalam mengklasifikasikan penyakit pada daun mangga karena kurangnya pengetahuan terkait penyakit pada daun mangga.

2. Sistem sebelumnya tidak menghasilkan *output* yang spesifik dan input masih berupa gejala dalam bentuk *checkbox*.

3.4. Analisis Kebutuhan Sistem

Bagian ini mencakup analisis terhadap kebutuhan sistem berdasarkan permasalahan yang ditemukan saat ini. Berikut kebutuhan sistem untuk klasifikasi penyakit pada daun mangga.

1. Menyediakan aplikasi yang dapat mengklasifikasikan penyakit pada daun mangga dan mengedukasi pengguna terkait penyakit pada daun mangga.
2. Sistem mampu menghasilkan *output* penyakit pada daun mangga secara spesifik dan input yang digunakan berupa citra melalui kamera atau galeri.

BAB 4

DESAIN SISTEM

Bab ini meliputi desain proses yang digunakan untuk sistem dan aplikasi. Desain proses ini berupa desain pembagian *dataset*, desain proses pelatihan dan evaluasi model, desain proses klasifikasi citra, desain proses penggunaan aplikasi dan desain tampilan pada aplikasi atau *user interface*.

4.1. Desain Proses

Desain proses berisi alur jalannya aplikasi, dimulai dari pengguna menginputkan citra hingga menampilkan hasil berupa label klasifikasi.

4.1.1. Desain Proses Pembagian *Dataset*

Dataset yang digunakan ada dua macam yaitu untuk melatih model YOLO dan melatih model klasifikasi CNN. *Dataset* yang digunakan untuk melatih model YOLO didapatkan dari *dataset open source* milik Roboflow (Mangga, 2023). *Dataset* ini memiliki total 1000 citra yang dibagi menjadi 3 yaitu *train*, *test*, *valid* dengan perbandingan kurang lebih 70:20:10. Sehingga *train* memiliki 697 citra, *test* memiliki 199 citra dan *valid* memiliki 104 citra. Berikut contoh citra daun mangga pada *dataset* YOLO.



Gambar 4.1. Contoh Citra Daun Mangga Dataset YOLO

Sementara itu, *dataset* yang digunakan untuk melatih model CNN adalah citra daun mangga yang didapatkan dari *MangoLeafBD* pada repositori *Mendeley* (Ahmed et al., 2023). Citra daun mangga dikumpulkan dari empat kebun mangga. Total *dataset* memiliki 4000 citra, diambil dari sekitar 1800 daun mangga yang mencakup tujuh jenis penyakit. Citra daun mangga tersebut diambil menggunakan kamera ponsel. Beberapa citra dengan kualitas kurang baik dihilangkan. Selain itu, untuk variasi citra telah dilakukan beberapa operasi terhadap citra daun mangga yaitu *resize*, rotasi dan *zoom*. Sehingga menghasilkan 4000 citra dan masing-masing kategori memiliki 500 citra. Dalam deskripsi *dataset* dikatakan bahwa data telah divalidasi oleh pakar pertanian dan diberi label secara manual.

Pada tahapan ini, *dataset* dibagi menjadi 80% digunakan untuk data pelatihan (*training*) dan 20% atau 800 citra sebagai data pengujian (*testing*), lalu dari 80% data pelatihan diambil 20% sebagai data validasi. Sehingga dari keseluruhan *dataset*, terdapat 16% atau 640 citra sebagai data validasi dan 64% atau 2560 citra sebagai data pelatihan. Pembagian *dataset* ini menggunakan *library* Python. Maka terbentuk 3 folder yaitu *train*, *val*, dan *test*. Setiap folder ini memuat 8 folder yaitu 7 kategori penyakit daun mangga dan 1 kategori *healthy*. Berikut contoh citra daun mangga pada *dataset*.

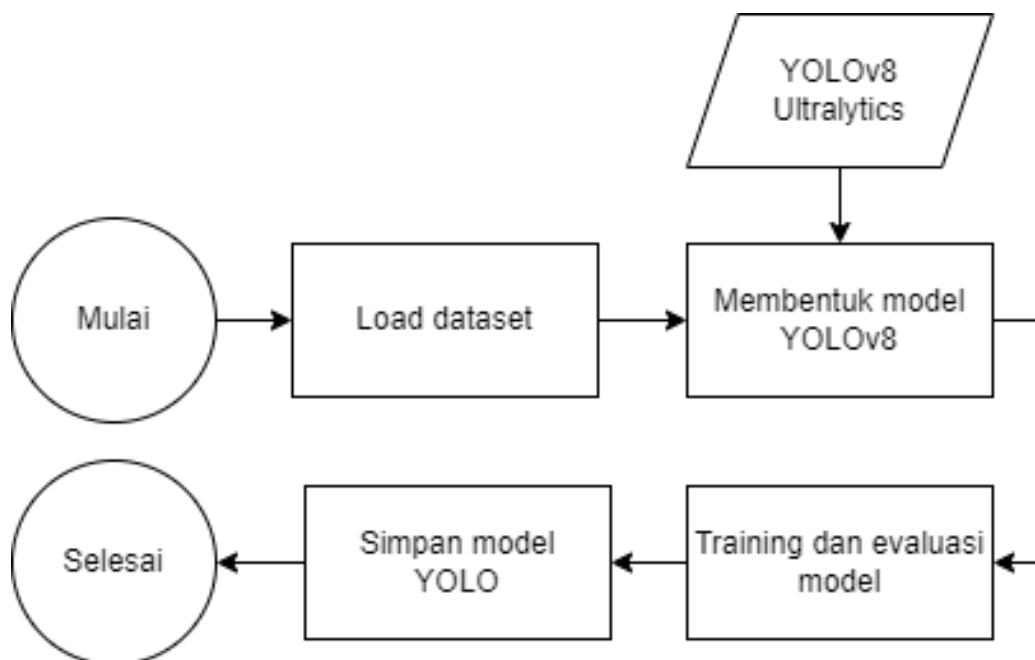


Gambar 4.2. Contoh Citra Daun Mangga *Dataset Klasifikasi*

4.1.2. Desain Proses Pelatihan dan Evaluasi Model

Tahap desain proses pelatihan (*training*) dilakukan setelah pembagian *dataset*. Pertama, *dataset* dibagi menjadi data *train*, *valid*, dan *test*, data ini juga diperiksa jumlahnya, apakah sudah sesuai dengan total *dataset* yang dimiliki.

Setelah itu membentuk model YOLO untuk mendeteksi daun mangga. Model YOLO dilatih dengan *dataset* YOLO yang didapat dari Roboflow. Berikut diagram alur pada proses *training* model YOLOv8.



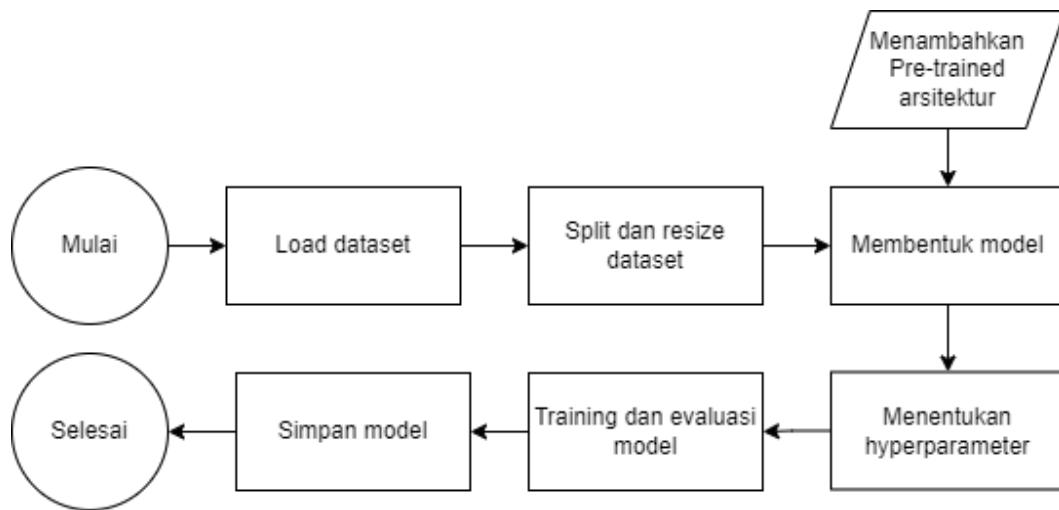
Gambar 4.3. Diagram Alur Proses Training Model YOLOv8

Selanjutnya, membentuk model klasifikasi menggunakan *pre-trained* dengan 4 jenis arsitektur yaitu ResNet50, VGG16, MobileNet, dan Inception-v3. Model setiap arsitektur juga ditambahkan *dense layer* dan *dropout layer* yang parameter dan jumlah *layer* didapatkan dari *hyperparameter tuning*. Jumlah *dense layer* adalah 0 hingga 3 *layer* dan jumlah *dropout layer* adalah 0 hingga 3 *layer*. Berikut tabel detail setiap nilai yang digunakan pada *hyperparameter tuning*.

Tabel 4.1. Detail Nilai *Hyperparameter Tuning*

<i>Hyperparameter</i>	Nilai
<i>Dense layer</i>	<i>Units</i> : 32 hingga 1024 dengan <i>step</i> 32
<i>Dropout layer</i>	<i>Rate</i> : 0.3 hingga 0.7 dengan <i>step</i> 0.1
<i>Learning rate</i>	Nilai terbaik dari 0.1, 0.01, dan 0.001
Jumlah <i>dense layer</i>	0 hingga 3 <i>layer</i>
Jumlah <i>dropout layer</i>	0 hingga 3 <i>layer</i>

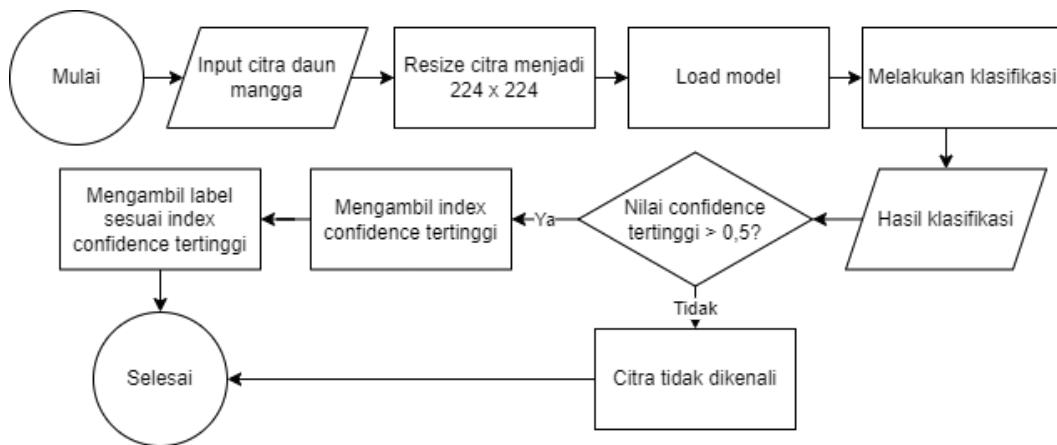
Setelah model terbentuk dengan nilai dari *hyperparameter tuning*, model dilakukan *training* dengan jumlah *epochs* sebanyak 10. Seluruh model yang telah di-*training* disimpan dengan format H5 (HDF5). Kemudian model pada setiap arsitektur dievaluasi menggunakan data *testing*, model yang menghasilkan akurasi tertinggi yang akan digunakan pada aplikasi. Berikut diagram alur pada proses *training* model klasifikasi CNN.



Gambar 4.4. Diagram Alur Proses Training Model Klasifikasi CNN

4.1.3. Desain Proses Klasifikasi Citra

Desain proses klasifikasi citra menjelaskan alur proses model dalam melakukan klasifikasi terhadap citra yang diinputkan. Citra yang diinputkan adalah daun mangga baik yang berpenyakit maupun daun mangga yang sehat. Selanjutnya, dilakukan *resize* pada citra input dengan ukuran 224 x 224 agar sesuai dengan input arsitektur. Setelah itu, citra diklasifikasikan menggunakan model yang telah *di-training*. Hasil dari klasifikasi berupa sebuah *array* yang memuat tingkat *confidence* dari setiap kategori. Kategori yang memiliki tingkat *confidence* tertinggi yang akan digunakan sebagai *output* label klasifikasi. Sebelum label klasifikasi ditampilkan, nilai *confidence* diperiksa terlebih dahulu apakah lebih tinggi daripada nilai *threshold* yang telah ditentukan yaitu 0,5. Jika terpenuhi, maka label hasil klasifikasi ditampilkan. Jika tidak terpenuhi, maka menampilkan pesan bahwa citra tidak dikenali. Berikut merupakan diagram alur proses klasifikasi.

**Gambar 4.5. Diagram Alur Proses Klasifikasi Citra**

4.1.4. Desain Proses Penggunaan Aplikasi

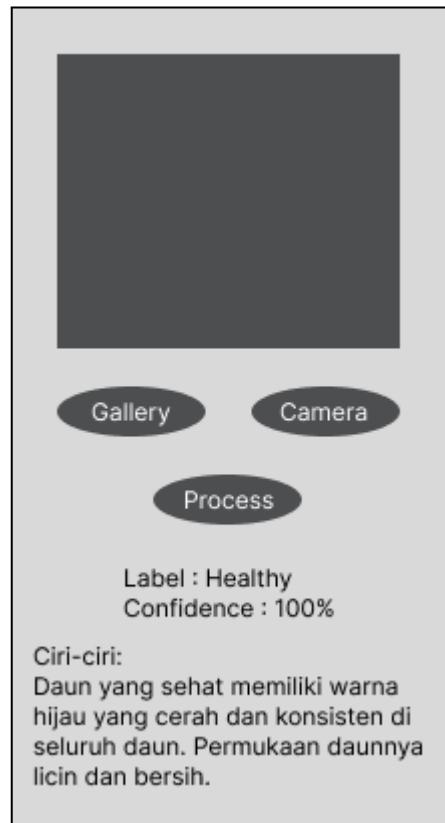
Desain proses penggunaan aplikasi berisi diagram yang menjelaskan alur untuk menggunakan aplikasi. Pertama, pengguna menginputkan citra bisa melalui galeri maupun kamera ponsel. Setelah menginputkan citra, pengguna dapat menekan tombol proses untuk melakukan proses klasifikasi. Jika citra belum diinputkan, maka aplikasi akan menampilkan pesan untuk mengingatkan pengguna memilih citra terlebih dahulu. Apabila citra sudah diinputkan, maka aplikasi akan memproses citra dan pengguna dapat menunggu proses klasifikasi hingga selesai. Setelah proses klasifikasi selesai, pengguna dapat melihat hasil klasifikasi dan tingkat keyakinan model pada layar. Berikut diagram alur penggunaan aplikasi untuk klasifikasi penyakit pada daun mangga.



Gambar 4.6. Diagram Alur Penggunaan Aplikasi

4.2. Desain *User Interface*

Sub bab ini berisi penjelasan desain *user interface* dari aplikasi klasifikasi penyakit pada daun mangga, aplikasi dibuat berbasis Android. Terdapat 1 halaman utama yang memuat citra dan 3 tombol yaitu tombol *gallery* yang digunakan untuk mengambil citra dari galeri, selanjutnya tombol *camera* yang digunakan untuk mengambil citra melalui kamera ponsel, dan tombol *process* untuk mengklasifikasikan citra daun mangga. Setelah citra diproses dan diklasifikasi, hasil ditampilkan pada halaman yang sama. Hasil berupa label dari klasifikasi dan tingkat keyakinan model dalam melakukan klasifikasi, serta ciri-ciri dari penyakit pada daun mangga tersebut. Berikut gambar desain *user interface*.



Gambar 4.7. Desain User Interface

BAB 5

IMPLEMENTASI SISTEM

Bab ini meliputi implementasi dari desain sistem yang telah dilakukan pada bab sebelumnya. Implementasi yang dijelaskan yaitu mulai dari implementasi pembagian dataset, pelatihan dan evaluasi model, klasifikasi citra, penggunaan aplikasi, dan implementasi *user interface*. Dalam mengimplementasikan sistem, program dibuat menggunakan bahasa pemrograman Python versi 3.11.5 yang dilakukan pada Microsoft Visual Studio Code versi 1.85.0 dengan sistem operasi Windows. Sementara itu, aplikasi dibuat menggunakan *framework* Flutter dengan bahasa pemrograman Dart.

5.1. Implementasi Proses

Sub bab ini meliputi penjelasan proses program dengan bahasa pemrograman Python dan Dart mulai dari proses pembagian *dataset*, proses pelatihan dan evaluasi model, proses klasifikasi citra dan proses penggunaan aplikasi.

5.1.1. Implementasi Proses Pelatihan dan Evaluasi Model YOLOv8

Proses pelatihan model YOLOv8 dilakukan pada Google Colab dan *dataset* disimpan ke Google Drive. Untuk mempercepat proses pelatihan menggunakan GPU yang tersedia pada Google Colab. Pertama hubungkan Drive yang menyimpan *dataset*, setelah Drive berhasil terhubung dapat dilanjutkan

dengan proses pelatihan. Program pelatihan model YOLOv8 dapat dilihat pada Listing 5.1.

Listing 5.1. Program Pelatihan Model YOLOv8

```
!yolo detect train  
data=/content/drive/MyDrive/YOLOv8/data.yaml epochs=25
```

Setelah model YOLOv8 selesai dilatih, model dengan *weights* terbaik disimpan. Model ini yang digunakan untuk mendekripsi daun mangga.

5.1.2. Implementasi Proses Pembagian *Dataset* Klasifikasi

Implementasi proses pembagian *dataset* diawali dengan melakukan *import library* *splitfolders*. Dengan bantuan *library* *splitfolders*, *dataset* dibagi menjadi 3 yaitu *training*, *testing* dan *validation*. Pembagian *Dataset* menggunakan rasio 80% *training* dan 20% *testing*. Kemudian dari 80% data *training* diambil 20% untuk data *validation*. Sehingga dalam program rasinya menjadi 0,64 data *training* 0,16 data *validation* dan 0,2 data *testing*. Sehingga jumlah citra untuk setiap kategori pada data *training* berjumlah 320 citra, data *testing* berjumlah 100 citra, data *validation* berjumlah 80 citra. Program untuk *import library* dan pembagian *dataset* dapat dilihat pada Listing 5.2.

Listing 5.2. Program *Import Library* dan Pembagian *Dataset* Klasifikasi

```
import splitfolders  
  
source = f'D:\SEMESTER 7\TA\dataset\MangoLeafBD'  
destination = f'D:\SEMESTER 7\TA\dataset\MangoLeaf'
```

```
splitfolders.ratio(source, destination, seed = 42, ratio =  
(0.64, 0.16, 0.2))
```

5.1.3. Implementasi Proses Pelatihan dan Evaluasi Model Klasifikasi

Proses pelatihan model dilakukan menggunakan Visual Studio Code. Pelatihan model menggunakan *dataset* yang telah dibagi pada tahap pembagian *dataset*. Langkah berikutnya adalah melakukan *import library*. *Library* yang dibutuhkan yaitu numpy, cv2, os, tensorflow, keras, dan sklearn. Sementara arsitektur yang dilakukan *import* ada 4 yaitu ResNet50, VGG16, MobileNet, InceptionV3. Program *import library* dapat dilihat pada Listing 5.3.

Listing 5.3. Program *Import Library*

```
import numpy as np  
  
import cv2  
  
import os  
  
import tensorflow as tf  
  
import keras_tuner as kt  
  
from tensorflow.keras.utils import  
image_dataset_from_directory  
  
from keras.applications import ResNet50  
  
from keras.applications.vgg16 import VGG16  
  
from keras.applications.mobilenet import MobileNet  
  
from keras.applications import InceptionV3  
  
from keras.optimizers import Adam  
  
from keras.callbacks import EarlyStopping
```

```
from keras.layers import Flatten, Dense, Dropout  
from sklearn.metrics import precision_score, recall_score,  
f1_score
```

Selanjutnya melakukan *load dataset* menggunakan *image_dataset_from_directory* milik tensorflow. Parameter *batch_size* yang digunakan sebesar 32, lalu Citra diatur dengan ukuran 224 x 224 dan RGB. Selain itu *train_dataset* dilakukan *shuffle* untuk mencegah model memahami pola label *dataset* yang berurutan. Jumlah citra pada setiap variabel yaitu *train_dataset* berjumlah 2560 citra, *val_dataset* berjumlah 640 citra dan *test_dataset* berjumlah 800 citra, total keseluruhan adalah 4000 citra. Program *load dataset* dapat dilihat pada Listing 5.4.

Listing 5.4. Program *Load Dataset* Klasifikasi

```
train_dataset = image_dataset_from_directory(f'D:\\SEMESTER  
7\\TA\\dataset\\MangoLeaf\\train',  
batch_size=32, image_size=(224,224), color_mode='rgb',  
seed=42, shuffle=True)  
  
val_dataset = image_dataset_from_directory(f'D:\\SEMESTER  
7\\TA\\dataset\\MangoLeaf\\val',  
batch_size=32, image_size=(224,224), color_mode='rgb',  
seed=42, shuffle=False)  
  
test_dataset = image_dataset_from_directory(f'D:\\SEMESTER  
7\\TA\\dataset\\MangoLeaf\\test',  
batch_size=32, image_size=(224,224), color_mode='rgb',
```

```
seed=42, shuffle=False)
```

Kemudian membentuk model untuk 4 arsitektur yang digunakan. Model dibentuk dalam sebuah *function* `build_model`. Langkah pertama melakukan *load* model, sebagai contoh model ResNet50. *Load* model menggunakan *pre-trained weights* dari imagenet. *Layer fully connected* terakhir pada model dihilangkan menggunakan *include_top = False*, karena *dense layer* terakhir akan diganti sesuai dengan kebutuhan. Citra input pada model ini berukuran 224 x 224 piksel dan 3 menunjukkan RGB. Selanjutnya *trainable* diatur menjadi *False* guna untuk menjaga agar bobot dari *layer* tidak diperbarui saat proses pelatihan. Berikutnya mendefinisikan jumlah kategori pada variabel `class_num`. Kemudian membuat sebuah model *sequential* keras yang akan menampung semua *layer*. Setelah itu, menambahkan *layer* ResNet50 ke dalam model, lalu tambahkan juga *Flatten* untuk meratakan *output* dari ResNet50 agar dapat dihubungkan dengan *dense layer* baru.

Selanjutnya menentukan *hyperparameter tuning* pada *dense layer*, *dropout* dan *learning rate*. Banyak *dense layer* dan *dropout* ditentukan oleh *hyperparameter tuning* yang akan ditambahkan melalui *for* dengan batas minimal adalah 0 *layer* dan maksimal adalah 3 *layer*. Jumlah *units* pada *dense layer* juga ditentukan oleh *hyperparameter tuning* dalam rentang antara 32 hingga 1024 dengan *step* 32. Sementara itu, nilai *dropout* dalam rentang antara 0.3 hingga 0.7 dengan *step* 0.1. Kemudian menambahkan *dense layer* terakhir yang parameternya `class_num` nilainya sesuai dengan jumlah *output* yaitu 8 dan

menggunakan *activation function softmax*. Lalu membentuk *optimizer Adam* dengan *learning rate* antara 0.1, 0.01, 0.001. Langkah terakhir *compile* model dengan *optimizer* yang dibentuk sebelumnya, *loss* yang digunakan adalah *sparse_categorical_crossentropy* dan *metrics accuracy*. Program pembentukan model klasifikasi CNN dapat dilihat pada Listing 5.5.

Listing 5.5. Program Pembentukan Model Klasifikasi

```
def build_model(hp):  
    resnet50 = ResNet50(weights='imagenet',  
                         include_top=False, input_shape=(224, 224, 3))  
  
    resnet50.trainable = False  
  
    class_num = 8  
  
    model = tf.keras.models.Sequential()  
  
    model.add(resnet50)  
  
    model.add(Flatten())  
  
    for i in range(0, hp.Int('num_layers', 0, 3)):  
  
        units = hp.Int(f'dense_{i}_units', min_value=32,  
                      max_value=1024, step=32)  
  
        dropout_rate = hp.Float(f'dropout_{i}_rate',  
                               min_value=0.3, max_value=0.7, step=0.1)  
  
        model.add(Dense(units=units, activation='relu'))  
  
        model.add(Dropout(rate=dropout_rate))  
  
    model.add(Dense(class_num, activation='softmax'))  
  
    optimizer = Adam(learning_rate=hp.Choice('learning_rate',  
                                             values=[0.1, 0.01, 0.001]))  
  
    model.compile(optimizer=optimizer,
```

```
loss='sparse_categorical_crossentropy', metrics=['accuracy'])  
return model
```

Langkah berikutnya mengatur *tuning*, jenis *tuning* yang digunakan adalah Hyperband milik Keras Tuner. Hyperband ditampung dalam sebuah variabel yaitu tuner dengan parameter berisi build_model yang telah dibentuk, *objective* yang diutamakan adalah meningkatkan akurasi validasi, nilai *epochs* maksimal adalah 10, *factor* yang digunakan untuk mereduksi sumber daya sebesar 3 untuk setiap iterasi pada *epochs*. Kemudian hasilnya disimpan ke *directory* dan diberi nama resnet50. Program mengatur *hyperparameter tuning* dapat dilihat pada Listing 5.6.

Listing 5.6. Program Mengatur *Hyperparameter Tuning*

```
tuner = kt.Hyperband(build_model, objective='val_accuracy',  
max_epochs=10, factor=3, directory='D:\\SEMESTER  
7\\TA\\tuning', project_name='resnet50')
```

Untuk menghemat sumber daya dan menghindari *overfitting* saat proses pelatihan dapat menggunakan *early stopping* milik Keras Callback. Pelatihan model akan dihentikan ketika memenuhi suatu kondisi tertentu. *Metrics* yang dimonitor adalah val_loss dengan *patience* sebesar 3. Jadi dalam 3 *epochs* berturut-turut tidak mengalami perbaikan val_loss, maka pelatihan model dihentikan dan bobot terbaik model sebelum pelatihan dihentikan yang diambil. Karena bobot terbaik tidak selalu berada pada *epoch* terakhir. Program *early stopping* dapat dilihat pada Listing 5.7.

Listing 5.7. Program Mengatur *Early Stopping*

```
early_stopping = EarlyStopping(monitor='val_loss',
                               patience=3, restore_best_weights=True)
```

Tahap berikutnya proses pencarian *hyperparameter* yang terbaik dengan melakukan percobaan terhadap kombinasi pada *hyperparameter*. Setelah proses pencarian *hyperparameter* selesai. Percobaan dengan akurasi tertinggi yang diambil sebagai *best_hps* dan nilai *learning rate* disimpan pada variabel *best_learning_rate* yang digunakan pada *optimizer* Adam. Kemudian *result_summary* berfungsi untuk menampilkan 10 percobaan dengan kombinasi *hyperparameter* terbaik yang diurutkan berdasarkan akurasi tertinggi. Program pencarian *hyperparameter tuning* terbaik dapat dilihat pada Listing 5.8.

Listing 5.8. Program Mencari *Hyperparameter Tuning* Terbaik

```
tuner.search(train_dataset, validation_data=val_dataset,
              epochs=10, callbacks=[early_stopping])
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
best_learning_rate = best_hps.get('learning_rate')
optimizer = Adam(learning_rate=best_learning_rate)
tuner.results_summary()
```

Selanjutnya proses pelatihan model dengan memanggil *build_model* yang sudah dibentuk sebelumnya beserta konfigurasi *hyperparameter* terbaik dan ditampung dalam sebuah variabel yaitu *model*. Kemudian melakukan pelatihan menggunakan *train_dataset* dan *val_dataset* dengan *epochs* sebanyak 10 iterasi.

Ringkasan model ditampilkan menggunakan summary, hasilnya berupa *layer*, *output shape*, dan jumlah parameter yang dapat dilatih. Lalu simpan model yang sudah dilatih untuk digunakan saat proses klasifikasi citra. Program pelatihan model klasifikasi dapat dilihat pada Listing 5.9.

Listing 5.9. Program Pelatihan Model Klasifikasi CNN

```
model = build_model(best_hps)
history = model.fit(train_dataset,
validation_data=val_dataset, epochs=10)
model.summary()
model.save(f'D:\\SEMESTER 7\\TA\\model\\resnet50.h5')
```

5.1.4. Implementasi Proses Mendeteksi Daun Mangga

Tahap implementasi proses mendeteksi daun mangga dimulai dari melakukan *import library* YOLO, kemudian *load* model YOLOv8 terbaik untuk mendeteksi daun mangga. Program *import library* dan *load* model YOLO dapat dilihat pada Listing 5.10.

Listing 5.10. Program Import Library dan Load Model YOLO

```
from ultralytics import YOLO
model = YOLO('/content/drive/MyDrive/YOLOv8/best.pt')
```

Setelah melakukan *load* model YOLOv8, dilanjutkan dengan melakukan prediksi citra yang diinputkan. *Output* yang dihasilkan berupa label yang menunjukkan

nilai 0 bukan daun mangga dan nilai 1 berarti daun mangga. Program mendeteksi citra daun mangga dapat dilihat pada Listing 5.11.

Listing 5.11. Program Mendeteksi Citra Daun Mangga

```
results = model('/content/drive/MyDrive/images.jpeg')
```

5.1.5. Implementasi Proses Klasifikasi Citra

Tahap implementasi proses klasifikasi citra diawali dengan melakukan *load* model terbaik yang memiliki akurasi tertinggi dari 4 arsitektur yang telah diuji. Sebagai contoh *load* model resnet50, dapat dilihat pada Listing 5.12.

Listing 5.12. Program Load Model Klasifikasi

```
model = load_model("D:\\SEMESTER 7\\TA\\model\\resnet50.h5")
```

Setelah melakukan *load* model, input citra menggunakan cv2 dan disimpan dalam variabel img. Buat sebuah *list* yang menampung label klasifikasi. Lalu, mengatur *threshold* sebesar 0.5 pada sebuah variabel. Kemudian citra diubah ukurannya menjadi 224 x 224 piksel dengan interpolasi area untuk mempertahankan detail citra dengan baik. Selanjutnya lakukan penambahan dimensi dengan numpy expand_dims agar sesuai dengan dimensi input citra model. Lalu citra diprediksi menggunakan model yang sudah di-*load* sebelumnya. Indeks hasil prediksi dengan tingkat keyakinan paling tinggi disimpan dalam top_pred. Selanjutnya indeks tersebut digunakan untuk menentukan label yang sesuai. Nilai tingkat keyakinan disimpan pada variabel top_prob, jika nilainya lebih besar dari *threshold* maka label prediksi dan tingkat keyakinan ditampilkan.

Jika tidak, maka tampil *output* citra tidak dikenali. Program klasifikasi citra input dapat dilihat pada Listing 5.13.

Listing 5.13. Program Klasifikasi Citra Input

```
img = cv2.imread("D:\\SEMESTER  
7\\TA\\IMG_20230701_210708.jpg")  
  
vals = ['Anthracnose', 'Bacterial Canker', 'Cutting Weevil',  
'Die Back', 'Gall Midge', 'Healthy', 'Powdery Mildew', 'Sooty  
Mould']  
  
threshold = 0.5  
  
img = cv2.resize(img, (224, 224),  
interpolation=cv2.INTER_AREA)  
  
img = np.expand_dims(img, axis=0)  
  
pred = model.predict(img)  
  
top_pred = np.argmax(pred)  
  
label = vals[top_pred]  
  
top_prob = pred[0][top_pred]  
  
if top_prob >= threshold:  
  
    print("Label prediksi:", label)  
  
    print("Tingkat keyakinan:", top_prob)  
  
else:  
  
    print("Citra tidak dikenali")
```

5.1.6. Implementasi Proses Penggunaan Aplikasi

Implementasi proses penggunaan aplikasi menggunakan Visual Studio Code. Aplikasi dibuat berbasis Android menggunakan Flutter dengan bahasa

pemrograman Dart. Terdapat *web service* yang berfungsi sebagai *server* untuk menghubungkan *front end* dan *back end*. *Web service* yang digunakan adalah Flask dan *back end* menggunakan bahasa pemrograman Python. Proses penggunaan aplikasi diawali dengan memilih citra daun mangga, lalu citra dikirimkan ke *server* dan diproses. *Preprocessing* dan klasifikasi citra dilakukan di *back end*, lalu hasil klasifikasi dan tingkat keyakinan dikirimkan kembali ke *front end* dalam bentuk json. Kemudian hasil json di-*decode* menggunakan *library* Dart *Convert* agar data dapat dipahami oleh bahasa pemrograman.

Program pengiriman citra dan menerima respon server dibuat dalam sebuah *function* yaitu `_processImage` yang berbentuk *asynchronous*. Pertama citra yang diinputkan diperiksa dahulu, apabila *null* maka aplikasi menampilkan pesan “Please select an image before processing.” sementara jika hasilnya tidak *null* maka muncul indikator *loading* yang menunjukkan citra sedang diproses. Selanjutnya *routing* ke server disimpan dalam variabel url. Url ini digunakan untuk melakukan permintaan dengan metode POST dan mengirimkan sebuah citra. Permintaan dikirim menggunakan metode *send* dan menunggu respon server. Jika respon *status code* 200 artinya permintaan berhasil dan hasilnya dapat diolah. Sementara itu jika respon *status code* bukan 200 maka dianggap kesalahan. Lalu data respon diubah menjadi string dan di-*decode* menjadi sebuah objek agar propertinya dapat diakses. Kemudian properti *label* dan *confidence* ditampung pada variabel yang akan ditampilkan kepada pengguna. Program pengiriman citra dan respon server dapat dilihat pada Listing 5.14.

Listing 5.14. Program Pengiriman Citra dan Respon Server

```
Future<void> _processImage() async {
    if (_pickedImage == null) {
        _showSnackBar('Please select an image before
processing.');
        return;
    }
    showDialog(
        context: context,
        builder: (BuildContext context) {
            return Center(child: CircularProgressIndicator());
        },
    );
    var url = 'http://192.168.210.60:5000/classify';
    try {
        var request = http.MultipartRequest('POST',
Uri.parse(url));
        request.files
            .add(await http.MultipartFile.fromPath('image',
_pickedImage!.path));
        var response = await request.send();
        if (response.statusCode == 200) {
            Navigator.pop(context);
            var responseData = await
response.stream.bytesToString();
            var data = jsonDecode(responseData);
    }
}
```

```
var label = data['label'];

var confidence = (data['confidence'] *
100).toStringAsFixed(2);

print('Predicted Label: $label');

print('Confidence: $confidence%');

setState(() {

    _predictedLabel = 'Predicted Label: $label';

    _predictedConfidence = 'Confidence: $confidence%';

});

} else {

    Navigator.pop(context);

    print('Error: ${response.statusCode}');

    _showSnackBar('Error: ${response.statusCode}');

}

} catch (e) {

    Navigator.pop(context);

    print('Error: $e');

    _showSnackBar('Error: $e');

}

}
```

Setelah proses pengiriman citra ke server, citra diterima dan diproses oleh program *back end*. Pertama lakukan *import library* yang diperlukan yaitu Flask, tensorflow, PIL (pillow), numpy dan YOLO. Selanjutnya menginisialisasi aplikasi Flask dan CORS untuk menangani permintaan lintas domain. Lalu *load* model YOLOv8 dan model klasifikasi CNN terbaik yang akan digunakan, sebagai contoh model ResNet50. Setelah itu, simpan label kategori dalam sebuah *list* dan menetapkan nilai *threshold*. Langkah berikutnya membuat sebuah *function* yang digunakan untuk *preprocessing*. Pada tahap *preprocessing* melakukan *load* citra, mengubah ukuran menjadi 224 x 224 piksel, mengubah citra menjadi bentuk *array* dan menambahkan dimensi citra agar sesuai dengan input dari model.

Selanjutnya membuat *endpoint* “/classify” dengan *methods* POST, artinya server hanya akan merespon permintaan POST dengan *endpoint* “/classify” dan menyertakan citra. Kemudian citra dideteksi keberadaan daun mangganya menggunakan model YOLOv8. Jika daun mangga terdeteksi simpan nilai keyakinan model YOLOv8 ke variabel *confidence_level*, tingkat keyakinan ini diperiksa apakah sudah melebihi nilai *threshold*, jika ya maka daun mangga terdeteksi, sebaliknya jika tidak maka daun mangga tidak terdeteksi dan mengembalikan label citra tidak dikenali menggunakan *jsonify*. Apabila daun mangga terdeteksi proses dilanjutkan dengan melakukan *preprocessing* menggunakan *function preprocess_image* terhadap citra. Lalu citra diprediksi menggunakan model CNN yang sudah di-*load* sebelumnya. Indeks hasil prediksi dengan tingkat keyakinan paling tinggi disimpan dalam *predicted_class*. Selanjutnya indeks tersebut digunakan untuk menentukan label dan ciri yang

sesuai dan disimpan pada variabel predicted_label dan predict_detail, sementara itu nilai tingkat keyakinan disimpan pada variabel confidence, lalu nilai confidence diperiksa kembali apakah lebih kecil dari *threshold*, jika ya maka citra dinyatakan tidak dikenali, sebaliknya jika lebih besar maka citra dikenali dan label prediksi dan confidence dikirim kembali menggunakan jsonify. Berikutnya mengatur mode debug menjadi *True* dan host ke 0.0.0.0 agar dapat diakses dari luar localhost. Program *back end* dapat dilihat pada Listing 5.15.

Listing 5.15. Program Back End

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import tensorflow as tf
from PIL import Image
import numpy as np
from ultralytics import YOLO

app = Flask(__name__)
CORS(app)

detection_model = YOLO('best.pt')

classification_model =
tf.keras.models.load_model('resnet50.h5')

class_labels = ['Anthracnose', 'Bacterial Canker', 'Cutting
Weevil', 'Die Back', 'Gall Midge', 'Healthy', 'Powdery
Mildew', 'Sooty Mould']

labels_detail = ['Munculnya bercak hitam pada sisi daun
mangga. Daun yang terkena dampak parah, biasanya mulai
melengkung.', 'Terdapat bintik-bintik basah pada tangkai dan
badan daun. Ketika bintik-bintik sudah banyak, berubah
menjadi bercak kanker yang tidak teratur.', 'Daun terlihat
seperti dipotong dengan gunting. Biasa berdampak pada daun
muda.', 'Daun berubah menjadi coklat dan menggulung. Diikuti
```

dengan ranting atau dahan menjadi mati dan layu sehingga daun rontok.', 'Muncul bintik-bintik seperti jerawat pada daun. Dalam keadaan berat terjadi defoliasi dan penurunan buah.', 'Daun yang sehat memiliki warna hijau yang cerah dan konsisten di seluruh daun. Permukaan daunnya licin dan bersih.', 'Terdapat jamur pada permukaan daun. Jamur menyerupai tepung ini juga dapat muncul pada tangkai bunga dan buah.', 'Terdapat cairan manis dan lengket yang dibuat oleh serangga untuk menarik serangga lain. Tumbuh jamur pada cairan tersebut dan perlahan menyebar ke seluruh permukaan daun. Warna daun perlahan menjadi hitam.]

```
threshold = 0.3

def preprocess_image(image_path):
    img = Image.open(image_path)
    new_size = (224, 224)
    img = img.resize(new_size)
    img = np.array(img)
    img = np.expand_dims(img, axis=0)
    return img

@app.route('/classify', methods=['POST'])
def classify_image():
    if 'image' not in request.files:
        return jsonify({'error': 'No image provided'}), 400

    image = request.files['image']
    img = Image.open(image)

    detection_results = detection_model(img)
    if(detection_results[0].boxes.shape[0] > 0):
        confidence_level =
float(detection_results[0].boxes.conf[0])
        mango_leaf_detected = confidence_level > threshold
    else:
        mango_leaf_detected = False

    if mango_leaf_detected:
```

```
processed_image = preprocess_image(image)
classification_predictions =
classification_model.predict(processed_image)
predicted_class =
np.argmax(classification_predictions)
confidence =
classification_predictions[0][predicted_class]
predicted_label = class_labels[predicted_class]
predicted_detail = labels_detail[predicted_class]

if confidence < threshold:
    return jsonify({'label': 'Citra tidak dikenali',
'confidence': 0.0, 'ciri': '-'}), 200

    return jsonify({'label': predicted_label,
'confidence': float(confidence), 'ciri': predicted_detail}),
200
else:
    return jsonify({'label': 'Citra tidak dikenali',
'confidence': 0.0, 'ciri': '-'}), 200

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

Berikutnya program tampilan aplikasi dengan membuat *widget* pada Flutter berisi sebuah *Scaffold* yang memiliki atribut *appbar*, *title*, *body* dan *child*. Supaya tampilan dapat di-scroll, dalam *body* ditambahkan sebuah *SingleChildScrollView*. Program tampilan aplikasi dapat dilihat pada Listing 5.16.

Listing 5.16. Program Tampilan Aplikasi

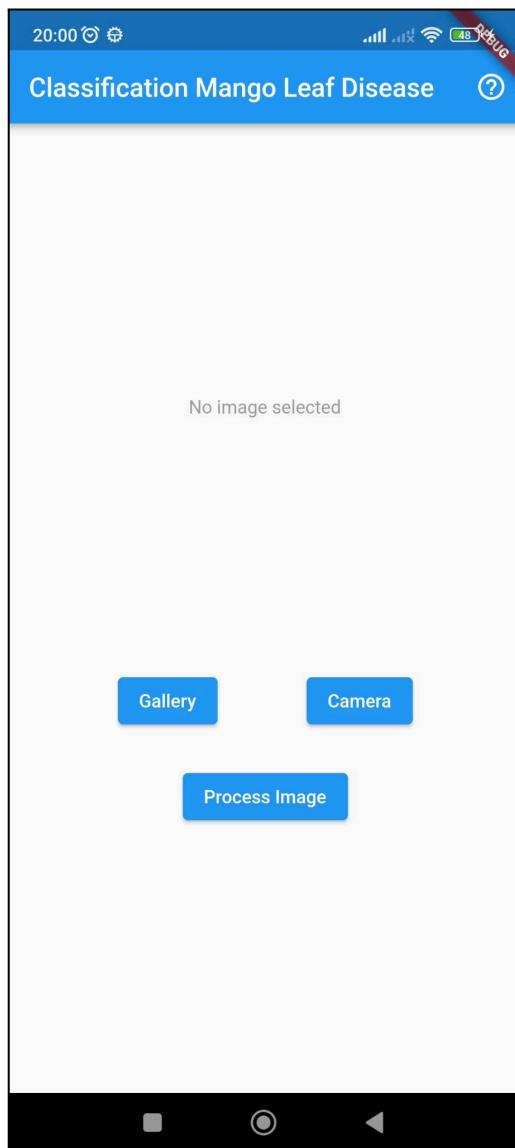
```
@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text('Classification Mango Leaf Disease'),
```

```
actions: [
    _buildIconButton(Icons.help_outline,
_showTutorial),
],
),
body: SingleChildScrollView(
padding: EdgeInsets.all(16),
child: Center(
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: <Widget>[
SizedBox(height: 30),
_buildImageDisplay(),
SizedBox(height: 30),
Row(
mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
children: [
_buildButton('Gallery', () =>
_getImage(ImageSource.gallery)),
_buildButton('Camera', () =>
_getImage(ImageSource.camera)),
],
),
SizedBox(height: 20),
_buildButton('Process Image', _processImage),
SizedBox(height: 30),
Text(_predictedLabel, style:
TextStyle(fontSize: 18)),
Text(_predictedConfidence, style:
TextStyle(fontSize: 18)),
SizedBox(height: 15),
Text(_predictedDetail, style:
TextStyle(fontSize: 18)),
],
),
),
),
),
);
};
```

```
}
```

5.2. Implementasi *User Interface*

Implementasi *user interface* menerapkan desain *user interface* yang telah dibuat sebelumnya. Terdapat 1 halaman utama yang memuat citra dan 3 tombol yaitu tombol *gallery* yang digunakan untuk mengambil citra dari galeri, selanjutnya tombol *camera* yang digunakan untuk mengambil citra melalui kamera ponsel, dan tombol *process image* untuk mengklasifikasikan citra daun mangga. Setelah citra diproses dan diklasifikasi, hasil ditampilkan pada halaman yang sama. Hasil berupa label dari klasifikasi dan tingkat keyakinan model dalam melakukan klasifikasi. Berikut gambar implementasi *user interface*.



Gambar 5.1. Implementasi *User Interface*

Untuk membantu pengguna dalam menggunakan aplikasi, di halaman tersebut juga terdapat sebuah tombol dengan simbol tanda tanya. Tombol ini berisi tutorial dan contoh foto yang diinputkan agar mendapatkan hasil yang lebih akurat. Berikut gambar dari tutorial penggunaan.



Gambar 5.2. Tutorial Penggunaan Aplikasi

BAB 6

UJI COBA DAN EVALUASI

Bab ini berisi uji coba dan evaluasi terhadap aplikasi dalam melakukan klasifikasi citra. Tahap verifikasi dilakukan untuk memastikan program berjalan dengan benar dan bebas dari *error*. Sedangkan tahap validasi untuk memastikan bahwa tujuan dari penelitian ini telah tercapai.

6.1. Verifikasi

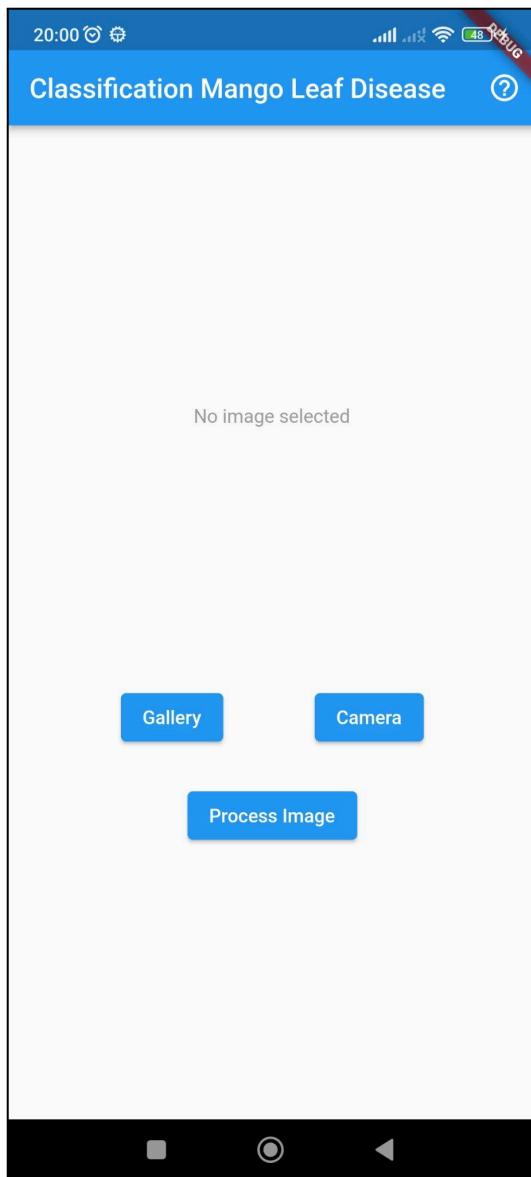
Tahap verifikasi menjelaskan hasil pengujian fitur dan tombol-tombol pada aplikasi. Uji coba ini dilakukan untuk memastikan program berjalan dengan benar dan bebas dari *error*. Pengujian ini juga dilakukan terhadap model klasifikasi, lalu model dievaluasi.

Verifikasi dibagi menjadi 2 yaitu uji coba jalannya aplikasi dan uji coba model dalam melakukan klasifikasi citra. Uji coba pada aplikasi meliputi pengujian terhadap fitur pada aplikasi seperti tutorial dan tombol-tombol yang tersedia. Sedangkan uji coba pada model meliputi *preprocessing* dan klasifikasi citra.

6.1.1. Pengujian Aplikasi

Aplikasi hanya memiliki 1 halaman utama yang memuat citra dan 3 tombol yaitu tombol *gallery* yang digunakan untuk mengambil citra dari galeri, selanjutnya tombol *camera* yang digunakan untuk mengambil citra melalui

kamera ponsel, dan tombol *process image* untuk mengklasifikasikan citra daun mangga. Berikut tampilan halaman aplikasi dapat dilihat pada Gambar 6.1.



Gambar 6.1. Tampilan Halaman Aplikasi

Pada halaman aplikasi juga terdapat sebuah tombol dengan simbol tanda tanya. Tombol ini jika ditekan akan menampilkan *pop up* yang berisi tutorial

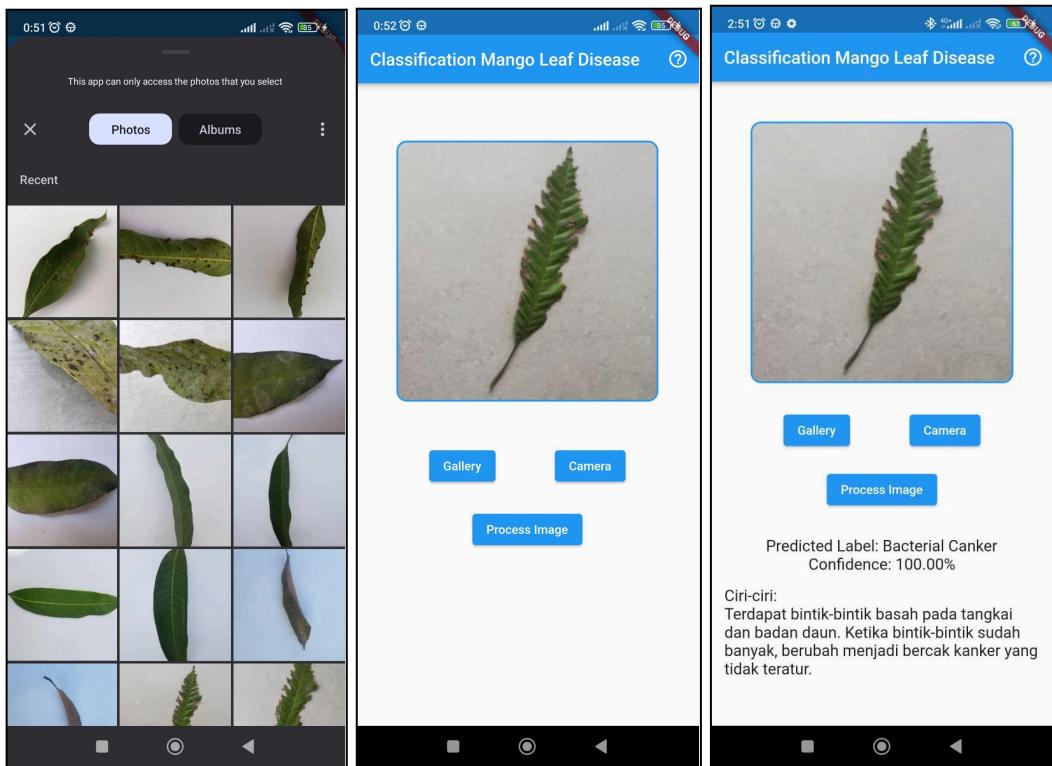
penggunaan aplikasi dan contoh citra yang diinputkan agar mendapatkan hasil yang lebih akurat. Berikut tampilan tutorial penggunaan aplikasi.



Gambar 6.2. Tampilan Tutorial Penggunaan Aplikasi

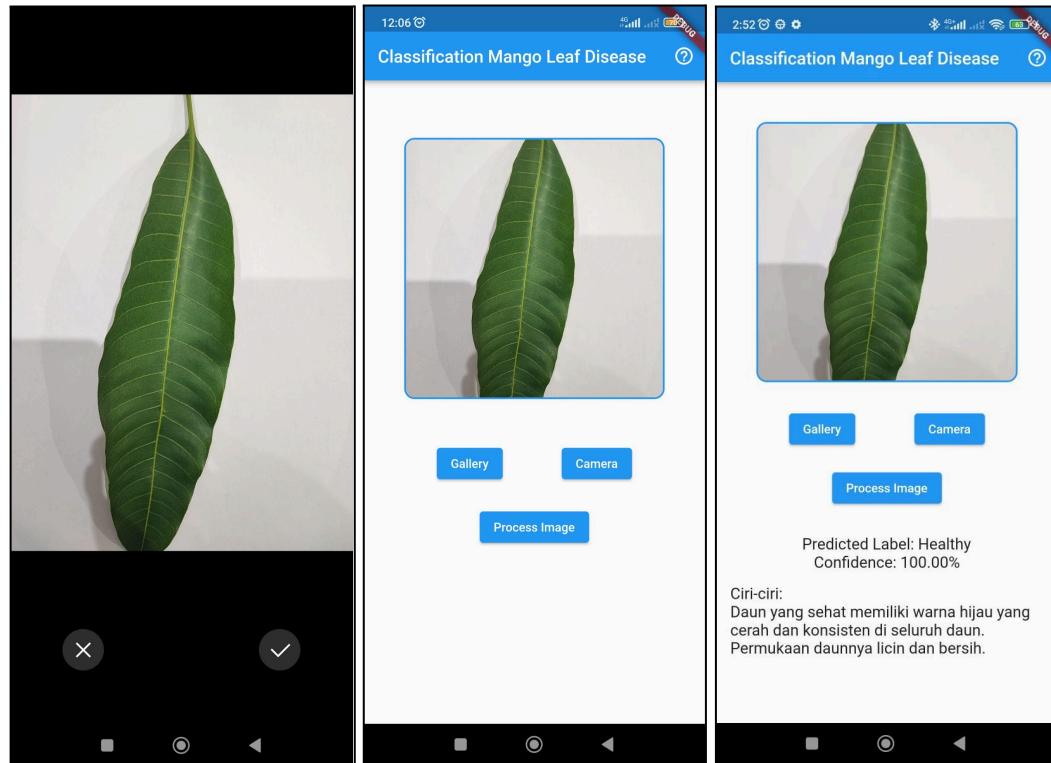
Setelah itu, pengguna dapat mengikuti tutorial dengan langkah pertama yaitu memilih citra. Uji coba dilakukan dengan memilih citra melalui galeri terlebih dahulu, lalu citra yang dipilih ditampilkan dan pengguna melanjutkan

dengan menekan tombol *process image*. Berikut gambar hasil dari uji coba input citra melalui galeri.



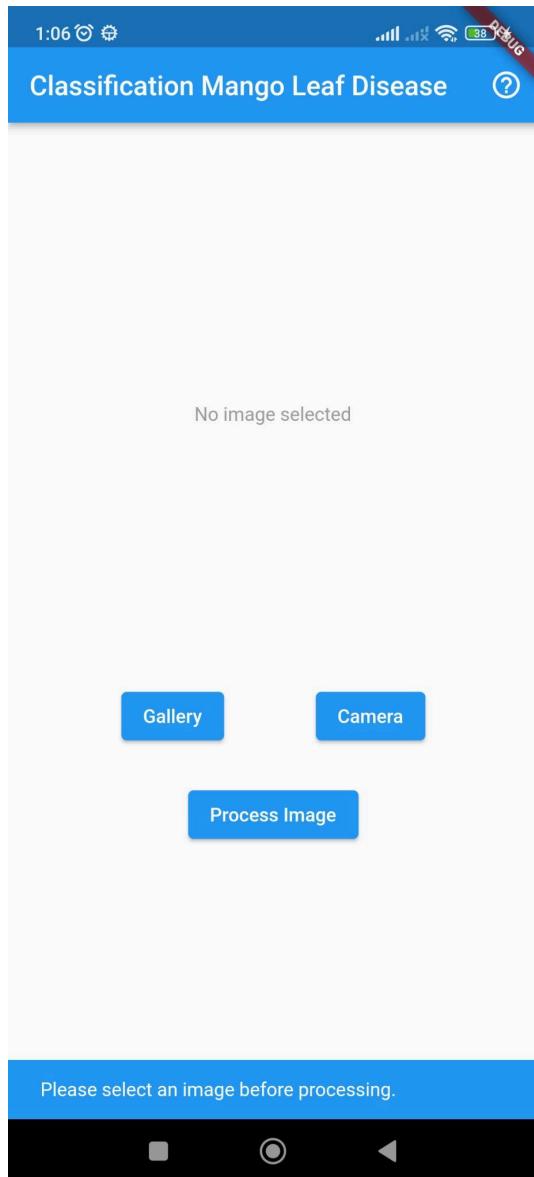
Gambar 6.3. Uji Coba Input Citra Melalui Galeri

Uji coba berikutnya input citra diambil melalui kamera ponsel. Setelah menangkap citra, pengguna menekan tombol *process image*. Proses klasifikasi ini mirip seperti sebelumnya, yang membedakan hanya dari mana input yang diberikan oleh pengguna. Berikut gambar hasil dari uji coba input citra melalui kamera ponsel.



Gambar 6.4. Uji Coba Input Citra Melalui Kamera Ponsel

Berikutnya uji coba apabila pengguna belum memilih citra tetapi langsung menekan tombol *process image*. Hasilnya adalah muncul sebuah pesan yang meminta pengguna untuk memilih citra terlebih dahulu. Berikut gambar uji coba menekan tombol *process image* tanpa memilih citra terlebih dahulu.



Gambar 6.5. Uji Coba Menekan *Process Image* Tanpa Memilih Citra

6.1.2. Pengujian Model

Tahap pengujian model menjelaskan hasil dari uji coba model dalam mengklasifikasikan citra. Model yang diuji yaitu ResNet50, VGG16, MobileNet, dan Inception-v3. Setiap model ini menggunakan konfigurasi *hyperparameter* yang sama, proses pencarian *hyperparameter* terbaik menggunakan *Hyperband*

milik Keras. Pencarian *hyperparameter* dilakukan sebanyak 30 kali percobaan. Berikut tabel hasil percobaan terbaik dari semua model yang diuji.

Tabel 6.1. Hasil *Hyperparameter Tuning* Terbaik

Model	Dense Layer	Units	Dropout	Learning Rate	Validation Accuracy
ResNet50	1	864	0.6	0.001	98.90 %
VGG16	1	768	0.4	0.001	98.75 %
MobileNet	1	928	0.5	0.001	94.21 %
Inception-v3	0	-	-	0.1	79.84 %

Setelah mendapatkan hasil konfigurasi *hyperparameter tuning* terbaik, konfigurasi ini yang digunakan sebagai parameter model. Pengujian model menggunakan data *validation* dan *testing*. Kemudian dilakukan evaluasi terhadap akurasi dari *validation* dan *testing*. Berikut tabel evaluasi model dengan konfigurasi *hyperparameter tuning* terbaik.

Tabel 6.2. Hasil Evaluasi Model Klasifikasi

Model	Validation Loss	Testing Loss	Validation Accuracy	Testing Accuracy
ResNet50	1.05	0.54	97.50 %	98.50 %

VGG16	0.09	1.23	99.06 %	97.87 %
MobileNet	0.21	0.19	92.81 %	94.63 %
Inception-v3	6867.48	7536.25	76.41 %	71.38 %

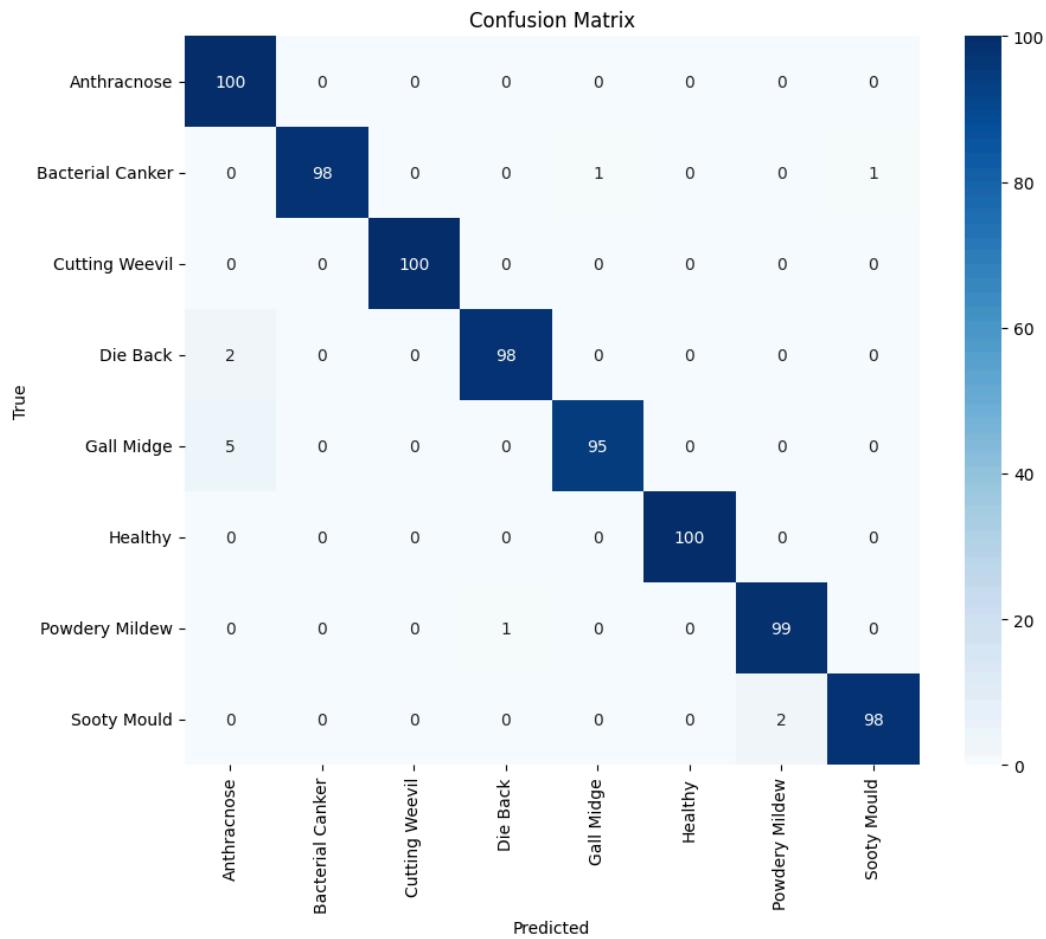
Dari hasil evaluasi model klasifikasi, model yang memiliki performa klasifikasi terbaik adalah *ResNet50*, hal ini ditunjukkan oleh akurasi *testing* tertinggi yaitu 98.50 %. Selanjutnya mengevaluasi *classification report* terhadap 8 kategori klasifikasi. *Classification report* berisi nilai dari *precision*, *recall* dan *F1-score*. Berikut tabel hasil *classification report*.

Tabel 6.3. Hasil Classification Report Model Terbaik

Label	Precision	Recall	F1-score
<i>Anthracnose</i>	0.93	1	0.97
<i>Bacterial Canker</i>	1	0.98	0.99
<i>Cutting Weevil</i>	1	1	1
<i>Die Back</i>	0.99	0.98	0.98
<i>Gall Midge</i>	0.99	0.95	0.97
<i>Healthy</i>	1	1	1
<i>Powdery Mildew</i>	0.98	0.99	0.99

<i>Sooty Mould</i>	0.99	0.98	0.98
Rata-rata	0.985	0.985	0.985
<i>Accuracy</i>	0.98	0.98	0.98
<i>Macro Average</i>	0.99	0.98	0.99
<i>Weighted Average</i>	0.99	0.98	0.99

Dari hasil *classification report* model terbaik, diperoleh 2 kategori yang memiliki bobot terbaik yaitu *cutting weevil* dan *healthy* dengan bobot *f1-score* sebesar 1. Sedangkan kategori yang memiliki bobot *f1-score* terendah adalah *anthracnose* dan *gall midge* dengan bobot senilai 0.97. Kemudian evaluasi dilanjutkan dengan *confusion matrix*. Berikut gambar hasil *confusion matrix* dari model terbaik.



Gambar 6.6. Hasil Confusion Matrix Model Terbaik

Tahap selanjutnya melakukan perhitungan akurasi secara manual dengan memprediksi 24 citra daun mangga dari keseluruhan kelas. Citra yang digunakan diambil dari *dataset test*. Berikut tabel hasil perhitungan akurasi dan prediksi model.

Tabel 6.4. Perhitungan Akurasi Model Klasifikasi

Citra Daun Mangga	Label Sebenarnya	Label Prediksi	Citra Daun Mangga	Label Sebenarnya	Label Prediksi
	Anthracnose	Anthracnose		Gall Midge	Gall Midge
	Anthracnose	Anthracnose		Gall Midge	Gall Midge
	Anthracnose	Anthracnose		Healthy	Healthy
	Anthracnose	Anthracnose		Healthy	Healthy
	Bacterial Canker	Bacterial Canker		Healthy	Healthy
	Bacterial Canker	Bacterial Canker		Powdery Mildew	Powdery Mildew
	Bacterial Canker	Bacterial Canker		Powdery Mildew	Powdery Mildew
	Cutting Weevil	Cutting Weevil		Powdery Mildew	Powdery Mildew

	Cutting Weevil	Cutting Weevil		Sooty Mould	Powdery Mildew
	Cutting Weevil	Cutting Weevil		Sooty Mould	Sooty Mould
	Die Back	Die Back		Die Back	Die Back
	Die Back	Die Back		Gall Midge	Sooty Mould
Rata-rata akurasi		91.67%			

Setelah itu dilakukan uji coba terhadap model YOLOv8 menggunakan 800 citra, dimana setiap kategori diambil 100 citra dari dataset *MangoLeafBD* dan diberi label daun mangga. Proses pelatihan menggunakan parameter *default* yaitu epochs sebanyak 100. Hasil evaluasi yang didapatkan pada model tersebut yaitu mAP50 sebesar 0.995 dan mAP50-95 sebesar 0.886, akurasi ini sudah cukup baik. Namun saat diuji citra yang bukan merupakan daun mangga, hasilnya selalu daun mangga. Sementara itu untuk model yang dilatih menggunakan dataset dari Roboflow, dapat bekerja lebih baik dalam mendekripsi daun mangga dengan mAP50 terbaik yaitu 0.995. Tetapi masih terdapat kekurangan, model hanya dapat mengenali daun mangga yang sehat saja.

6.2. Validasi

Tahap validasi dilakukan dengan cara wawancara/*interview*. Wawancara dilakukan kepada pihak Dinas Pertanian Kota Surabaya yaitu dengan Bapak Mochamad Anshori Irfan selaku Penyuluh Pertanian Lapang Kecamatan Kenjeran Kota Surabaya. Pertama, cara kerja aplikasi dijelaskan dan didemokan ke narasumber. Setelah melakukan demo, beberapa pertanyaan diajukan kepada narasumber. Berikut pertanyaan yang diajukan.

Pertanyaan pertama yaitu bagaimana menurut Anda mengenai tampilan aplikasi. Narasumber menyatakan tampilan aplikasi sudah baik dalam pengenalan objek daun mangga yang sehat dan terkena penyakit.

Pertanyaan kedua yaitu bagaimana penilaian Anda terhadap keakuratan hasil klasifikasi pada aplikasi. Narasumber menyatakan bahwa aplikasi sudah akurat dan sudah baik, karena indikasi penyakit daun pada daun mangga dapat diketahui dengan jelas.

Pertanyaan ketiga yaitu bagaimana pandangan Anda terhadap kecepatan aplikasi dalam melakukan klasifikasi. Narasumber menyatakan aplikasi sudah sangat cepat dalam melakukan klasifikasi.

Pertanyaan keempat yaitu apakah aplikasi dapat memudahkan masyarakat dalam melakukan klasifikasi penyakit pada daun mangga. Narasumber menyatakan aplikasi dapat memudahkan masyarakat untuk menindaklanjuti hasil daun mangga yang terkena penyakit.

Pertanyaan kelima yaitu apa kritik dan saran yang Anda miliki terkait aplikasi ini. Saran dari narasumber sebaiknya penjelasan 7 jenis penyakit tersebut disajikan juga dalam bentuk video agar masyarakat dapat langsung paham.

Berdasarkan hasil wawancara yang dilakukan, aplikasi yang dibuat telah berhasil membantu masyarakat dalam melakukan klasifikasi penyakit pada daun mangga menggunakan citra digital.

BAB 7

KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran dari narasumber mengenai aplikasi yang dibuat. Saran ini dapat digunakan untuk pengembangan aplikasi pada penelitian berikutnya.

7.1. Kesimpulan

Berdasarkan hasil uji coba dan evaluasi yang telah dilakukan sebelumnya, dapat diambil kesimpulan sebagai berikut:

1. Aplikasi telah diuji coba oleh narasumber dari Dinas Pertanian Kota Surabaya dan aplikasi dapat membantu masyarakat dalam mengklasifikasikan penyakit pada daun mangga.
2. Model YOLOv8 sudah baik dalam mengenali objek daun mangga yang sehat, tetapi kurang akurat untuk objek daun mangga yang tidak sehat.
3. Hasil pengujian model yang dilakukan terhadap arsitektur ResNet50, VGG16, MobileNet dan GoogleNet (Inception-v3) didapatkan model dengan performa terbaik yaitu arsitektur ResNet50 dengan nilai akurasi 98.5%, *precision* 98.5%, *recall* 98.5% dan *f1-score* 98.5%.

7.2. Saran

Dari hasil wawancara yang telah dilakukan sebelumnya, didapatkan beberapa saran sebagai berikut:

1. Menambahkan penjelasan 7 jenis penyakit tersebut disajikan juga dalam bentuk video agar masyarakat dapat langsung paham.
2. Menambahkan lebih banyak jenis penyakit pada daun mangga.
3. Mengembangkan aplikasi agar dapat digunakan untuk pengguna IOS dan website.
4. Memperbanyak *dataset* untuk model YOLOv8 agar lebih akurat dalam mengenali objek daun mangga.

DAFTAR PUSTAKA

- Ahmed, S. I., Ibrahim, M., Nadim, M., Rahman, M. M., Shejunti, M. M., Jabid, T., & Ali, M. S. (2023). MangoLeafBD: A comprehensive image dataset to classify diseased and healthy mango leaves. *Data in Brief*, 47, 108941. <https://doi.org/10.1016/j.dib.2023.108941>
- Ayu, T., Dwi, V., & Minarno, A. E. (2021). Pendiagnosa Daun Mangga Dengan Model Convolutional Neural Network. *CESS (Journal of Computer Engineering, System and Science)*, 6(2), 230. <https://doi.org/10.24114/cess.v6i2.22857>
- Cahya, F. N., Hardi, N., Riana, D., & Hadiyanti, S. (2021). Klasifikasi Penyakit Mata Menggunakan Convolutional Neural Network (CNN). *Sistemasi*, 10(3), 618. <https://doi.org/10.32520/stmsi.v10i3.1248>
- Darusalam, U. (2021). Face Recognition Menggunakan Opencv Dengan Bahasa Pemograman Python Oop Untuk Sistem Presensi Rumah Sakit. *Journal of Artificial Intelligence and Innovative Applications*, 2(3), 180–184. <http://openjournal.unpam.ac.id/index.php/JOAIIA/index218>
- Dewi, S. R. (2018). Deep Learning Object Detection Pada Video. *Deep Learning Object Detection Pada Video Menggunakan Tensorflow Dan Convolutional Neural Network*, 1–60. https://dspace.uii.ac.id/bitstream/handle/123456789/7762/14611242_Syariyah_Rosita_Dewi_Statistika.pdf?sequence=1
- Effendi, M. T., Hidayat, N., & Dewi, R. K. (2019). Sistem Diagnosis Penyakit Tumbuhan Mangga Menggunakan Metode Naive Bayes. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 3(4), 3896–3902. <https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/5072>
- Fitrianingsih, & Rodiah. (2020). Klasifikasi Jenis Citra Daun Mangga Menggunakan Convolutional Neural Network. *Jurnal Ilmiah Teknologi*

- Dan Rekayasa, 25(3), 223–238.
<https://doi.org/10.35760/tr.2020.v25i3.3519>
- Guissous, A. E. (2019). Skin lesion classification using deep neural network. *arXiv preprint arXiv:1911.07817*.
- Hao, J., & Ho, T. K. (2019). Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. *Journal of Educational and Behavioral Statistics*, 44(3), 348–361.
<https://doi.org/10.3102/1076998619832248>
- Hariyanto, R., & Sa'diyah, K. (2018). Sistem Pakar Diagnosis Penyakit dan Hama Pada Tanaman Tebu Menggunakan Metode Certainty Factor. *JOINTECS (Journal of Information Technology and Computer Science)*, 3(1), 1–4.
<https://doi.org/10.31328/jointecs.v3i1.500>
- Herdiyanto, A. A., Hidayat, N., & Dewi, R. K. (2019). Sistem Diagnosis Penyakit Tanaman Mangga Menggunakan Metode Bayesian Network. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3(4), 3597-3602.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*.
<http://arxiv.org/abs/1704.04861>
- Ide, P. (2013). *Health Secret of Mango*. Elex Media Komputindo.
- Jauhari, A. F. (2022). *Klasifikasi jenis beras menggunakan metode convolutional neural network pada arsitektur mobilenet*. Universitas Islam Negeri Maulana Malik Ibrahim.
- Juliansyah, S., & Laksito, A. D. (2021). Klasifikasi Citra Buah Pir Menggunakan Convolutional Neural Networks. *Jurnal Telekomunikasi Dan Komputer*, 11(1), 65. <https://doi.org/10.22441/incomtech.v11i1.10185>

- Kaur, T., & Gandhi, T. K. (2019). Automated brain image classification based on VGG-16 and transfer learning. *2019 International Conference on Information Technology (ICIT)*, 94–98.
- Krishna, S. T., & Kalluri, H. K. (2019). Deep learning and transfer learning approaches for image classification. *International Journal of Recent Technology and Engineering*, 7(5), 427–432.
- Kumar, V. (2020). Convolutional Neural Networks. Diakses 26 Juni 2023 dari <https://towardsdatascience.com/convolutional-neural-networks-f62dd896a856>
- Mahmood, A., Ospina, A. G., Bennamoun, M., An, S., Sohel, F., Boussaid, F., Hovey, R., Fisher, R. B., & Kendrick, G. A. (2020). Automatic hierarchical classification of kelps using deep residual features. *Sensors (Switzerland)*, 20(2), 1–20. <https://doi.org/10.3390/s20020447>
- Maleh, I. M. D., Teguh, R., Sahay, A. S., Okta, S., & Pratama, M. P. (2023). Implementasi Algoritma You Only Look Once (YOLO) Untuk Object Detection Sarang Orang Utan Di Taman Nasional Sebangau. *Jurnal Informatika*, 10(1), 19–27. <https://doi.org/10.31294/inf.v10i1.13922>
- Mangga, D. H. P. D. (2023, November). Daun Mangga Dataset. *Roboflow Universe*. Retrieved from <https://universe.roboflow.com/deteksi-hama-penyakit-daun-mangga/daun-mangga-mjf8d>
- Naufal, M. F. (2021). Analisis Perbandingan Algoritma SVM, KNN, dan CNN untuk Klasifikasi Citra Cuaca. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, 8(2), 311. <https://doi.org/10.25126/jtiik.2021824553>
- Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi Deep Learning Menggunakan Convolutional Neural Network (Cnn) Pada Ekspresi Manusia. *Algor*, 2(1), 12–21.

- Phung, V. H., & Rhee, E. J. (2019). A High-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences (Switzerland)*, 9(21). <https://doi.org/10.3390/app9214500>
- Sergeev, A., & Del Balso, M. (2018). *Horovod: fast and easy distributed deep learning in TensorFlow*. September. <http://arxiv.org/abs/1802.05799>
- Shinta, R. (2023). Klasifikasi Citra Penyakit Daun Tanaman Padi Menggunakan CNN dengan Arsitektur VGG-19. *Klasifikasi Citra Penyakit Daun Tanaman Padi Menggunakan CNN dengan Arsitektur VGG-19*, 9(01), 37-45.
- Team, S. (2018). Convolutional Neural Networks (CNN): Step 4-Full Connection. Diakses 26 Juni 2023 dari <https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection>
- Ultralytics. (2023). Ultralytics YOLOv8 Docs. Diakses 04 Januari 2024 dari <https://docs.ultralytics.com/#yolo-licenses-how-is-ultralytics-yolo-licensed>
- Zorgui, S., Chaabene, S., Bouaziz, B., Batatia, H., & Chaari, L. (2020). A Convolutional Neural Network for Lentigo Diagnosis. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*: Vol. 12157 LNCS. Springer International Publishing. https://doi.org/10.1007/978-3-030-51517-1_8