BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi Sistem

Implementasi sistem merupakan proses penerapan seluruh hasil perancangan dan analisis yang telah disusun pada bab sebelumnya ke dalam wujud nyata berupa aplikasi Android. Pada tahap ini, komponen-komponen utama sistem yang mencakup antarmuka pengguna, alur proses klasifikasi, serta model *Convolutional Neural Network* (CNN) berbasis MobileNetV2 diintegrasikan secara menyeluruh agar dapat dioperasikan oleh pengguna untuk mengklasifikasikan penyakit pada daun tomat. Proses implementasi dilakukan dengan mengacu pada kebutuhan fungsional yang telah ditetapkan, meliputi pengambilan gambar melalui kamera, pemilihan gambar dari galeri, deteksi secara *realtime*, tampilan hasil klasifikasi, penyimpanan riwayat, serta penyajian informasi seputar aplikasi.

Model yang telah dilatih menggunakan Python dan TensorFlow selanjutnya dikonversi ke dalam format TensorFlow Lite agar dapat berjalan pada perangkat Android. Dengan demikian, proses klasifikasi dapat berlangsung secara langsung di perangkat pengguna tanpa memerlukan koneksi ke *server* eksternal, sehingga memberikan kemudahan akses dan kecepatan respons yang lebih optimal.

4.1.1 Implementasi UI Aplikasi

Implementasi antarmuka aplikasi dilakukan berdasarkan rancangan tampilan pada Bab 3. Setiap halaman dibuat untuk mendukung proses penggunaan aplikasi secara sederhana, mulai dari halaman pembuka, menu utama, proses klasifikasi, hingga tampilan informasi dan riwayat hasil deteksi. Berikut merupakan uraian implementasi antarmuka yang diterapkan pada aplikasi klasifikasi penyakit daun tomat.

4.1.1.1 Implementasi Antarmuka *Splash screen*

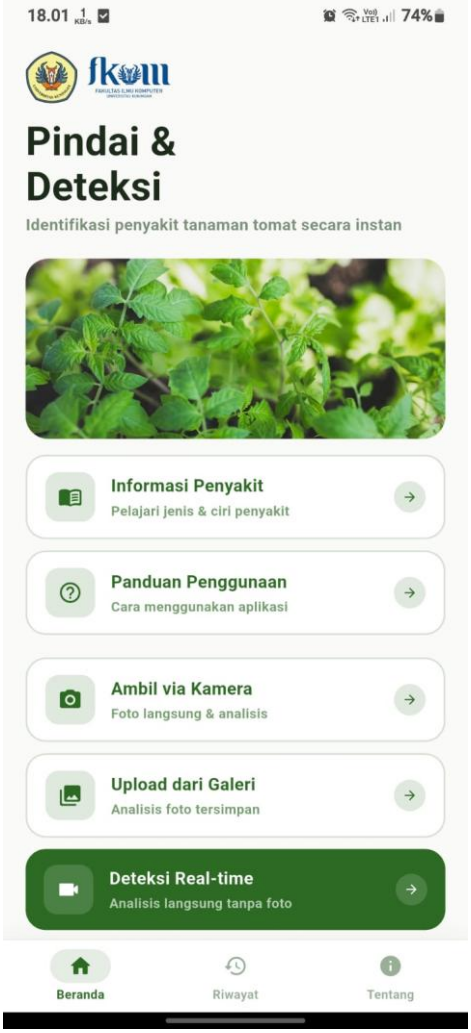
Splash screen merupakan tampilan awal yang muncul ketika aplikasi dijalankan. Halaman ini berfungsi untuk menampilkan identitas

aplikasi sebelum pengguna diarahkan ke menu utama. Pada implementasinya, halaman ini menampilkan logo atau ikon aplikasi serta nama aplikasi klasifikasi penyakit daun tomat.

Gambar 4. 1 Implementasi Antarmuka *Splash screen*

4.1.1.2 Implementasi Antarmuka Menu Utama

Menu utama menjadi pusat navigasi aplikasi. Pada halaman ini pengguna dapat memilih beberapa fitur utama, seperti mengambil gambar melalui kamera, memilih gambar dari galeri, membuka deteksi *realtime*, melihat bantuan, membuka riwayat klasifikasi, serta melihat informasi penyakit dan tentang aplikasi. Antarmuka ini dibuat sederhana agar pengguna dapat memahami fungsi setiap tombol dengan cepat.



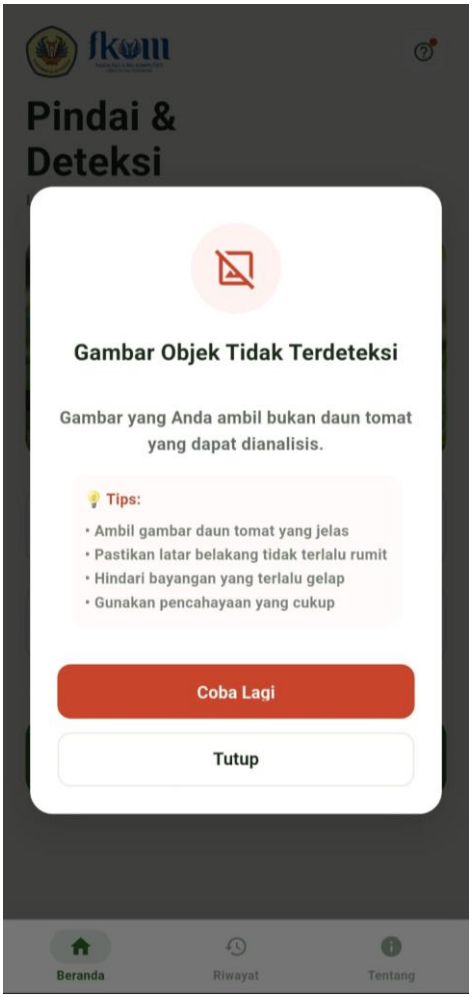
Gambar 4. 2 Implementasi Antarmuka Menu Utama

4.1.1.3 Implementasi Antarmuka Menu Hasil Analisa

Menu hasil analisa digunakan untuk menampilkan keluaran dari proses klasifikasi. Setelah citra daun tomat diproses oleh model, sistem menampilkan nama kelas hasil prediksi, nilai tingkat kepercayaan, dan deskripsi singkat terkait kondisi daun. Kelas yang dapat ditampilkan meliputi Bercak Daun (*Leaf Spot*), Busuk Daun (*Leaf Blight*), Jamur Daun (*Powdery Mildew*), dan daun dalam kondisi sehat.

Gambar 4. 3 Implementasi Antarmuka Menu Hasil Analisa

4.1.1.4 Implementasi Antarmuka Notifikasi Gagal Deteksi



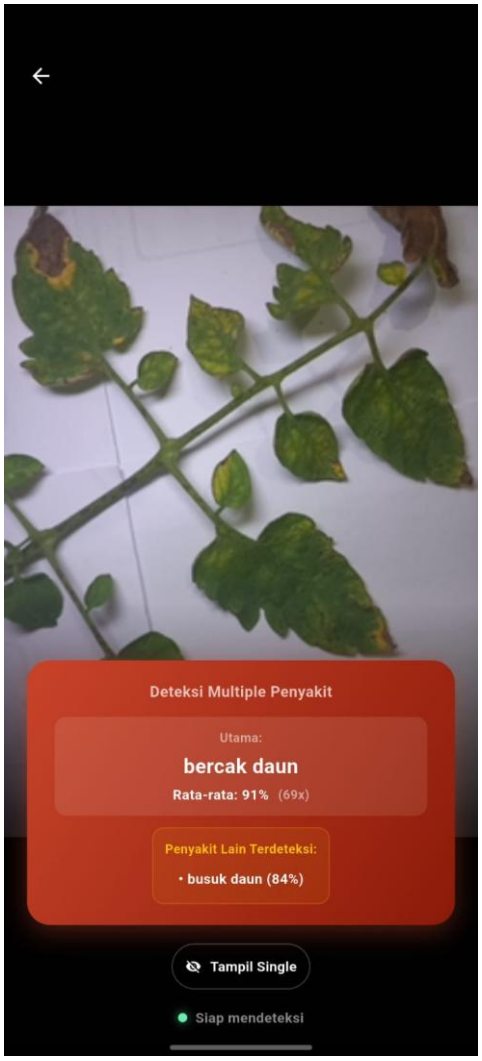
Gambar 4. 4 Implementasi Antarmuka Notifikasi Gagal Deteksi

4.1.1.5 Implementasi Antarmuka Menu Deteksi *Realtime*

Menu deteksi *realtime* digunakan untuk melakukan klasifikasi secara langsung melalui kamera perangkat. Pengguna cukup mengarahkan kamera ke objek daun tomat, kemudian aplikasi menampilkan hasil prediksi berdasarkan *frame* yang diproses. Fitur ini dirancang agar proses identifikasi dapat dilakukan secara cepat tanpa harus menyimpan gambar terlebih dahulu. Dalam proses deteksi, aplikasi mendukung dua kondisi pengambilan gambar, yaitu daun tunggal dan daun majemuk. Pada kondisi daun tunggal, kamera diarahkan secara langsung ke satu helai daun tomat sehingga model dapat memfokuskan analisis pada satu objek citra yang dominan dan menghasilkan prediksi

dengan tingkat keyakinan yang lebih tinggi. Sementara itu, pada kondisi daun majemuk, kamera menangkap beberapa helai daun sekaligus dalam satu *frame*; dalam kondisi ini model tetap melakukan prediksi berdasarkan pola visual keseluruhan citra yang masuk, meskipun akurasi hasil deteksi dapat dipengaruhi oleh keberadaan beberapa daun yang saling tumpang tindih atau memiliki kondisi yang berbeda-beda. Oleh karena itu, pengguna disarankan mengarahkan kamera pada satu daun yang paling jelas terlihat gejala penyakitnya untuk memperoleh hasil deteksi yang lebih optimal.

Gambar 4. 5 Implementasi Antarmuka Menu Deteksi *Realtime* Daun Tunggal



Gambar 4. 6 Implementasi Antarmuka Menu Deteksi *Realtime* Daun Majemuk

4.1.1.6 Implementasi Antarmuka Panduan

Antarmuka bantuan berisi panduan penggunaan aplikasi, seperti cara mengambil gambar melalui kamera, memilih gambar dari galeri, menjalankan deteksi *realtime*, serta membaca hasil klasifikasi. Halaman ini dibuat untuk membantu pengguna baru memahami alur penggunaan aplikasi.

Gambar 4. 7 Implementasi Antarmuka Bantuan

4.1.1.7 Implementasi Antarmuka Riwayat

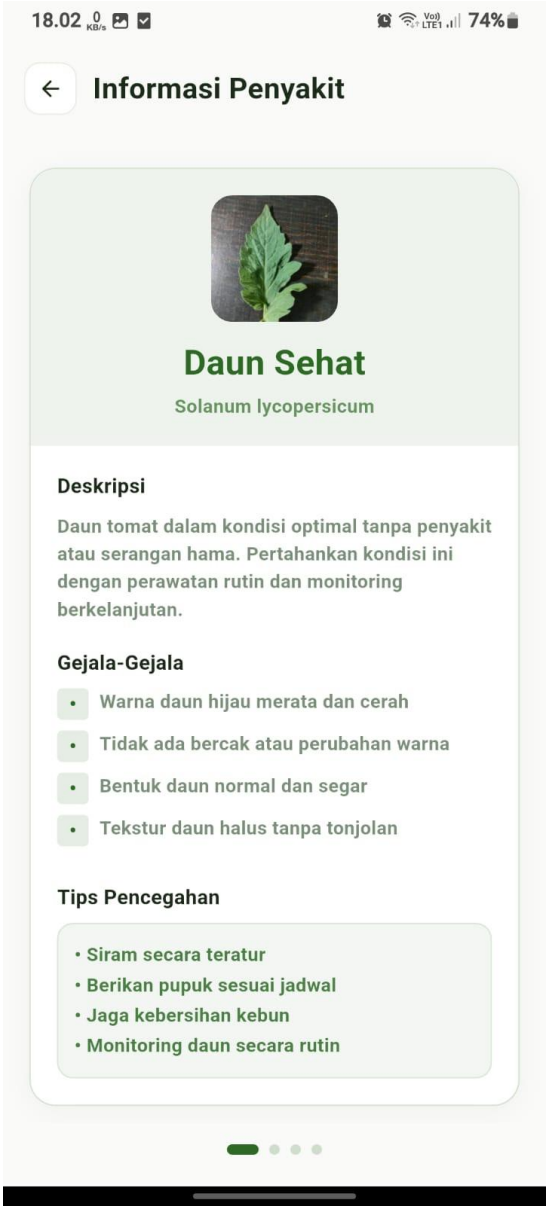
Menu riwayat berfungsi untuk menampilkan daftar hasil klasifikasi yang pernah dilakukan oleh pengguna. Informasi yang ditampilkan mencakup gambar daun, nama hasil klasifikasi, dan waktu proses deteksi. Fitur ini membantu pengguna meninjau kembali hasil klasifikasi sebelumnya.

Gambar 4. 8 Implementasi Antarmuka Riwayat

4.1.1.8 Implementasi Antarmuka Informasi Tentang Penyakit

Menu informasi tentang penyakit menyediakan referensi deskriptif mengenai empat kategori kondisi daun tomat yang dikenali oleh aplikasi, yaitu Bercak Daun (*Leaf Spot*), Busuk Daun (*Leaf Blight*), Jamur Daun (*Leaf Mold*), dan Daun Sehat. Setiap entri informasi memuat penjelasan mengenai gejala visual yang tampak pada daun, penyebab penyakit, serta rekomendasi penanganan yang dapat dilakukan oleh petani. Informasi ini disajikan dalam bahasa yang mudah dipahami agar dapat dimanfaatkan langsung oleh pengguna di lapangan tanpa memerlukan keahlian khusus di bidang pertanian. Antarmuka menu ini menampilkan daftar jenis penyakit beserta gambar ilustrasi pendukung,

sehingga pengguna dapat membandingkan kondisi daun yang ditemukan di lapangan dengan referensi visual yang tersedia di dalam aplikasi.



Gambar 4. 9 Implementasi Antarmuka Riwayat

4.1.1.9 Implementasi Antarmuka Menu Tentang

Menu tentang berisi informasi umum aplikasi, identitas pengembang, teknologi yang digunakan, dan penjelasan singkat mengenai model klasifikasi. Halaman ini juga memberikan informasi bahwa aplikasi menggunakan CNN dengan arsitektur MobileNetV2 yang

telah dikonversi ke TensorFlow Lite untuk dijalankan pada perangkat Android.

Gambar 4. 10 Implementasi Antarmuka Menu Tentang

4.1.2 Metode Penyelesaian Masalah

Metode penyelesaian masalah pada penelitian ini dilakukan melalui tahapan pengumpulan *dataset*, *preprocessing* data, penerapan arsitektur CNN

berbasis MobileNetV2, pelatihan model, evaluasi model, dan integrasi model ke dalam aplikasi Android. Tahapan tersebut bertujuan menghasilkan model yang mampu mengenali pola visual pada citra daun tomat dan mengelompokkannya ke dalam kelas penyakit yang sesuai.

4.1.2.1 Pengumpulan *dataset*

Dataset yang digunakan dalam penelitian ini berjumlah 4.000 citra daun tomat dengan empat kelas, yaitu Bercak Daun, Busuk Daun, Jamur Daun dan daun sehat. Setiap kelas memiliki 1.000 citra. *Dataset* kemudian dibagi menjadi data latih, data validasi, dan data uji dengan rasio 80:10:10. Pembagian data tersebut bertujuan agar model dapat dilatih, divalidasi, dan diuji menggunakan data yang berbeda sehingga hasil evaluasi lebih objektif.

Tabel 4. 1 Pembagian *Dataset*

Kelas	**Jumlah Data**	***Training* 80%**	**Validation 10%**	***Testing* 10%**
Bercak Daun	1.000	800	100	100
Busuk Daun	1.000	800	100	100
Jamur Daun	1.000	800	100	100
Sehat	1.000	800	100	100
Total	4.000	3.200	400	400

4.1.2.2 *Preprocessing* data

Setelah *dataset* dibagi ke dalam kelompok *train*, valid, dan *test*, tahap selanjutnya adalah melakukan augmentasi citra guna menambah variasi data. Teknik augmentasi yang diterapkan meliputi *rescale,* rotasi*, flip* horizontal*, zoom, dan shear*. Seluruh citra kemudian diubah ukurannya menjadi 224 x 224 piksel sesuai kebutuhan model

MobileNetV2, dan nilai piksel dinormalisasi ke rentang 0–1. Tujuan dari proses ini adalah untuk meningkatkan keberagaman data sehingga dapat meminimalisasi kesalahan klasifikasi dan meningkatkan akurasi model secara keseluruhan.

Proses *preprocessing* dimulai dengan menentukan parameter dasar, yaitu ukuran *input* gambar 224x224 piksel, ukuran batch sebanyak 32, dan jumlah *epoch* sebanyak 100. Untuk meningkatkan kemampuan generalisasi model, diterapkan teknik augmentasi data menggunakan objek ImageDataGenerator. Selama pelatihan, citra mengalami berbagai transformasi secara acak yang mencakup normalisasi piksel (*rescale*=1./255), pembalikan horizontal, rotasi hingga 30 derajat, *zoom* sebesar 20%, pergeseran horizontal dan vertikal masing-masing 10%, serta transformasi geser (*shear*) hingga 20%.

```
# — ImageDataGenerator ————————————————————————————————
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    zoom_range=0.2,
    shear_range=0.2,
    horizontal_flip=True
)
```

```
train_gen = train_datagen.flow_from_dataframe(
    train_df,
    x_col='filepath',
    y_col='label',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True,
    seed=SEED
)

valid_gen = val_test_datagen.flow_from_dataframe(
    valid_df,
    x_col='filepath',
    y_col='label',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False,    # diacak untuk menghindari overfitting evaluasi
    seed=SEED
)

test_gen = val_test_datagen.flow_from_dataframe(
    test_df,
    x_col='filepath',
    y_col='label',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False,    # diacak untuk menghindari overfitting evaluasi
    seed=SEED
```

Gambar 4. 11 Tahapan *Preprocessing* Data

4.1.2.3 Arsitektur *Convolutional Neural Network* (CNN)

Model klasifikasi dalam penelitian ini memanfaatkan arsitektur *Convolutional Neural Network* (CNN) dengan basis MobileNetV2. MobileNetV2 dikenal sebagai arsitektur yang ringan dan efisien, sangat cocok untuk dijalankan pada perangkat dengan keterbatasan sumber daya. Dengan memanfaatkan mekanisme *depthwise separable convolution* dan *inverted residual blocks*, arsitektur ini berhasil menekan jumlah parameter dan mempercepat proses inferensi tanpa kehilangan akurasi secara signifikan.

Arsitektur model dibangun dengan memanfaatkan MobileNetV2 yang telah dilatih sebelumnya menggunakan *dataset* ImageNet (*transfer learning*), dengan lapisan atas (*top*) sehingga *output* berupa *feature map* yang kemudian diteruskan ke lapisan klasifikasi kustom. Adapun

penjelasan dari masing-masing komponen arsitektur tersebut adalah sebagai berikut:

```
base_model = MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=(224,224,3)
)

for layer in base_model.layers:
    layer.trainable = False

x = base_model.output

# Pooling
x = GlobalAveragePooling2D()(x)

# Dense utama
x = Dense(256, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)

# Dense tambahan (opsional)
x = Dense(128, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.3)(x)

# Output
num_classes = len(train_gen.class_indices)

output = Dense(num_classes, activation='softmax')(x)
```

Gambar 4. 12 Arsitektur CNN Menggunakan MobileNetV2

1. *Feature Extraction*

   Pada tahap awal, model memanfaatkan arsitektur MobileNetV2 yang sudah dilatih sebelumnya menggunakan *dataset* ImageNet sebagai *feature extractor*. MobileNetV2 bertugas mengekstrak fitur-fitur penting dari citra *input* berukuran 224x224 piksel dengan 3 kanal warna (*RGB*). Fitur-fitur yang diekstraksi mencakup pola bercak, warna daun, tekstur permukaan, dan bentuk gejala penyakit yang menjadi pembeda antarkelas.

```
base_model = MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=(224,224,3)
)

for layer in base_model.layers:
    layer.trainable = False
```

Gambar 4. 13 Kode *Feature Extraction*

2. *Convolutional Layers &* ReLu

   Convolutional layers digunakan untuk mengekstraksi fitur lokal dari citra melalui operasi konvolusi dengan berbagai *filter*. Sementara itu, fungsi aktivasi ReLU (*Rectified Linear Unit*) diterapkan setelah setiap lapisan konvolusi untuk memberikan sifat non-linear pada model, sehingga model mampu membedakan variasi gejala penyakit yang memiliki kemiripan visual antarkelas.

3. *Pooling Layer (Global Average Pooling)*

   Tahap *pooling* menggunakan GlobalAverage*Pooling*2D yang secara otomatis menggantikan proses *flattening* konvensional. Lapisan ini mengubah *output* tiga dimensi dari MobileNetV2 menjadi vektor satu dimensi dengan cara merata-ratakan seluruh nilai pada setiap *feature map*. Pendekatan ini lebih efisien karena dapat mengurangi jumlah parameter secara signifikan dibandingkan *flattening* biasa, sekaligus mempertahankan informasi spasial yang relevan untuk proses klasifikasi.

```
x = base_model.output

# Pooling
x = GlobalAveragePooling2D()(x)
```

Gambar 4. 14 Kode *Pooling Layer*

4. *Flatten layer*

   Flatten layer berfungsi mengubah *feature map* yang masih berbentuk matriks menjadi vektor satu dimensi. Vektor tersebut kemudian digunakan sebagai masukan pada lapisan klasifikasi.

5. *Fully connected layer*

   Fully connected layer menghubungkan fitur yang telah diekstraksi dengan proses pengambilan keputusan. Pada tahap ini, fitur yang diperoleh dari citra daun tomat dipelajari untuk menentukan kemungkinan kelas yang paling sesuai.

```
# Dense utama
x = Dense(256, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)

# Dense tambahan (opsional)
x = Dense(128, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.3)(x)
```

Gambar 4. 15 Kode *Fully connected layer*

6. *Output layer*

   Output layer menggunakan fungsi aktivasi *Softmax* untuk menghasilkan probabilitas pada setiap kelas. Kelas dengan nilai probabilitas tertinggi akan ditampilkan sebagai hasil klasifikasi aplikasi.

```
# Output
num_classes = len(train_gen.class_indices)

output = Dense(num_classes, activation='softmax')(x)
```

Gambar 4. 16 Kode *Softmax*

4.1.2.4 *Training* Model

Training model dilakukan menggunakan data latih yang telah melalui proses *preprocessing*. Model MobileNetV2 dilatih untuk mengenali empat kelas citra daun tomat, yaitu Bercak Daun(*Leaf Spot)*, Busuk Daun(*Leaf Blight)*, Jamur Daun(*Powdery Mildew)*, dan daun sehat. Selama proses pelatihan, data validasi digunakan untuk memantau performa model agar tidak hanya baik pada data latih, tetapi juga mampu mengenali data yang belum pernah dilihat sebelumnya. Model terbaik kemudian disimpan dan dikonversi ke format TensorFlow Lite agar dapat diintegrasikan dengan aplikasi Android.

```
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(train_gen.classes),
    y=train_gen.classes
)

class_weights = dict(enumerate(class_weights))
print("Class Weight:", class_weights)

early_stop = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True,
    verbose=1
)

lr_reduce = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=3,
    verbose=1
)

checkpoint = ModelCheckpoint(
    filepath='/content/best_model_mobilenetv2.h5',
    monitor='val_accuracy',
    save_best_only=True,
    verbose=1
)
```

```
EPOCHS = 100

history = model.fit(
    train_gen,
    validation_data=valid_gen,
    epochs=EPOCHS,
    callbacks=callbacks,
    class_weight=class_weights
)
```

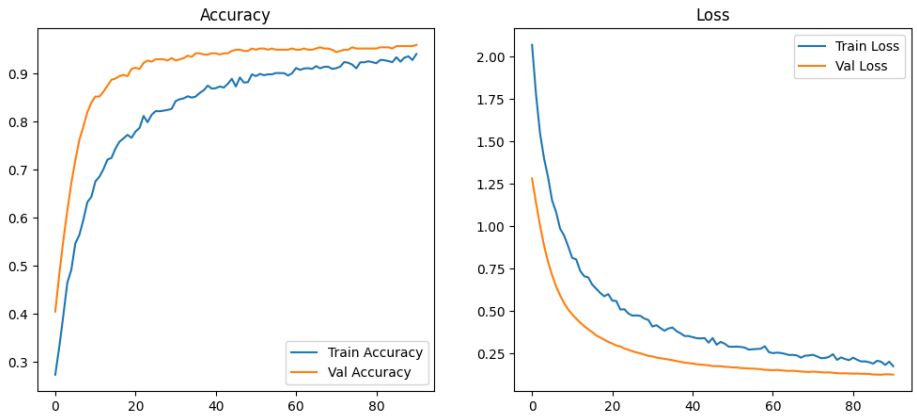
Gambar 4. 17 Konfigurasi *Training* Model

4.1.2.5 Evaluasi Model

Setelah proses pelatihan selesai, langkah selanjutnya adalah mengevaluasi model dengan menggunakan grafik yang menampilkan nilai *loss, accuracy, val_loss, dan val_accuracy*. Nilai *loss* menggambarkan seberapa besar kesalahan model saat mengenali gambar, sedangkan *accuracy* menunjukkan persentase keberhasilan model dalam melakukan klasifikasi dengan benar. Visualisasi dalam bentuk grafik ini memudahkan analisis dan perbandingan antara nilai *loss* dan *accuracy*, sehingga performa model dapat terlihat dengan lebih jelas.

Grafik akurasi menunjukkan bahwa akurasi pada data pelatihan terus meningkat dan hampir mendekati angka sempurna, yang

mengindikasikan bahwa model berhasil mempelajari pola dari data latih dengan sangat baik. Akurasi pada data validasi juga mengalami peningkatan yang cukup signifikan di awal pelatihan dan kemudian stabil, menunjukkan kemampuan model dalam menangani data baru dengan cukup baik. Sementara itu, grafik *loss* menunjukkan penurunan drastis di awal pelatihan dan terus menurun secara perlahan seiring bertambahnya *epoch*, yang mengindikasikan bahwa kesalahan prediksi model semakin mengecil dari waktu ke waktu. Secara keseluruhan, grafik ini memperlihatkan bahwa proses pelatihan berjalan dengan baik dan model memiliki kemampuan generalisasi yang solid terhadap data yang belum pernah dikenalinya.



Gambar 4. 18 Hasil Evaluasi Model

4.2 Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan bahwa aplikasi yang dibangun dapat berjalan sesuai kebutuhan pengguna dan rancangan sistem. Pengujian mencakup pengujian *white box*, *black box*, akurasi model menggunakan *Confusion matrix*, serta *User acceptance test* (UAT). Setiap pengujian digunakan untuk menilai aspek yang berbeda, mulai dari alur logika program, kesesuaian fungsi aplikasi, kemampuan model dalam melakukan klasifikasi, hingga penerimaan pengguna akhir terhadap aplikasi.

4.2.1 Pengujian *Blackbox*

Pengujian *blackbox* dilakukan untuk memeriksa kesesuaian fungsi aplikasi berdasarkan *input* dan *output* tanpa melihat struktur kode program. Pengujian ini difokuskan pada fitur-fitur utama aplikasi yang digunakan oleh pengguna.

Tabel 4. 2 Pengujian *Blackbox* Aplikasi

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Membuka aplikasi	Splash screen tampil dan berpindah ke menu utama	Aplikasi terbuka dan menampilkan halaman utama.	Berhasil
2	Menekan tombol kamera	Aplikasi membuka kamera perangkat	Kamera berjalan dengan baik.	Berhasil
3	Menekan tombol galeri	Aplikasi membuka galeri perangkat	Kamera berjalan dengan baik.	Berhasil
4	Memilih citra daun tomat	Aplikasi menampilkan citra dan memproses klasifikasi	Gambar dan analisis berhasil ditampilkan.	Berhasil
5	Menjalankan deteksi *realtime*	Aplikasi menampilkan kamera dan hasil prediksi *realtime*	*Realtime* berfungsi.	Berhasil
6	Membuka menu panduan	Aplikasi menampilkan	Menampilkan halaman	Berhasil

		panduan penggunaan	menu panduan.	
7	Membuka menu riwayat	Aplikasi menampilkan data riwayat klasifikasi	Berhasil menampilkan menu riwayat.	Berhasil
8	Membuka menu tentang	Aplikasi menampilkan informasi aplikasi	Berhasil menampilkan menu tentang.	Berhasil

Hasil pengujian *blackbox* menunjukkan bahwa setiap fitur utama aplikasi dapat merespons *input* pengguna sesuai dengan keluaran yang diharapkan. Dengan demikian, aplikasi telah memenuhi kebutuhan fungsional yang telah dirancang pada tahap perancangan sistem.

4.2.2 Pengujian *Whitebox*

Pengujian *whitebox* dilakukan dengan meninjau alur logika program pada proses klasifikasi citra. Pengujian ini berfokus pada proses internal aplikasi, mulai dari *input* citra, konversi citra menjadi bitmap, *preprocessing* citra, pemanggilan model TensorFlow Lite, hingga penampilan hasil klasifikasi. Tujuan dari pengujian ini adalah memastikan setiap jalur proses dapat berjalan sesuai alur yang telah dirancang. Berikut adalah kode program yang digunakan sebagai dasar *white box testing* pada aplikasi klasifikasi penyakit daun tomat:

Tabel 4. 3 Pengujian *Whitebox* Proses Klasifikasi

No	Kode Program
1	Static const String _modelPath = assets/models/model_daun_tomat.tflite';

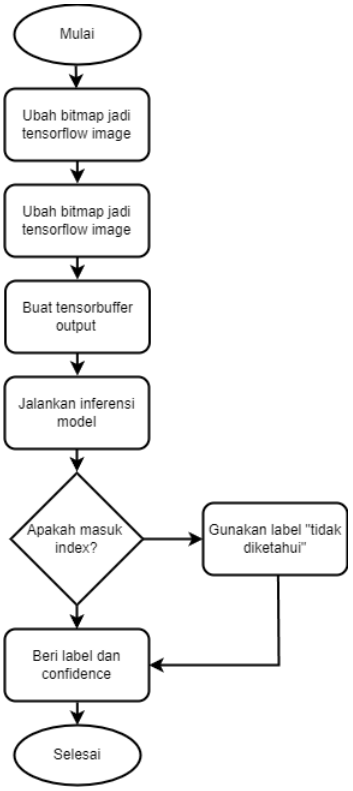
2	static const String _labelsPath = 'assets/labels/labels.txt';
3	static const int _inputSize = 224;
4	static const Set _excludedLabels 'kotak', 'background', 'latar belakang', 'unknown', 'tidak diketahui', 'other', 'lainnya',
5	bool get isHealthy { final normalized = *label*.toLowerCase(); return normalized } == 'healthy' || normalized == 'sehat';

6	bool get isValidTomatoLeaf {
	return
	_validTomatoLeafLabels.contains(*label*.toLowerCase().trim()) &&
	| | | confidence >= 0.75;
	}

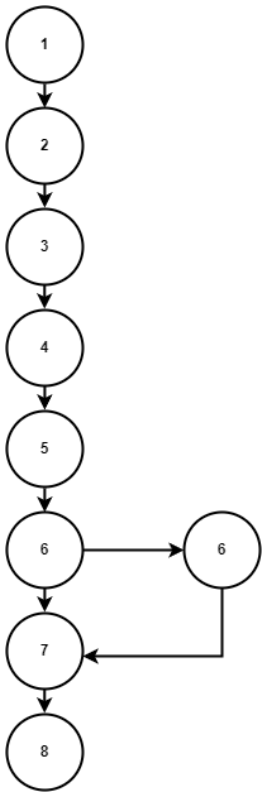
7	String get confidencePercent => '${(confidence * 100).toStringAsFixed(1)}%';

8	String get confidenceLabel {
	if (confidence >= 0.90) return 'Sangat Yakin';
	if (confidence >= 0.75) return 'Yakin';
	if (confidence >= 0.60) return 'Cukup Yakin'; return 'Tidak Yakin';
	}

Berdasarkan tabel pengujian 4.2 diatas, maka dapat dibentuk *flowchart* dan *flowgraph* seperti pada gambar dibawah



Gambar 4. 19 *Flowchart* Pengujian *Whitebox*



Gambar 4. 20 *Flowgraph* Pengujian *Whitebox*

Cyclomatic Complexity adalah metrik yang digunakan untuk mengukur jumlah jalur logika independen dalam suatu fungsi atau program. Metrik ini penting karena membantu menentukan jumlah minimal test case yang dibutuhkan untuk menguji seluruh kemungkinan alur eksekusi dalam kode.

Diketahui: E = (Jumlah *edge* pada *flowgraph*) = 9, N = (Jumlah *node* pada *flowgraph*) = 9. Maka rumusnya adalah V(G) = E – N + 2 = 9 – 9 + 2 = 2. Terdapat 2 jalur yang dihasilkan, yaitu:

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

Path ini terjadi ketika model berhasil melakukan klasifikasi terhadap citra daun tomat dan menghasilkan indeks kelas (*label*) yang valid, yaitu berada dalam rentang daftar LABELS. Pada jalur ini, sistem mengembalikan nama kelas penyakit daun tomat beserta nilai akurasinya sebagai keluaran kepada pengguna.

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

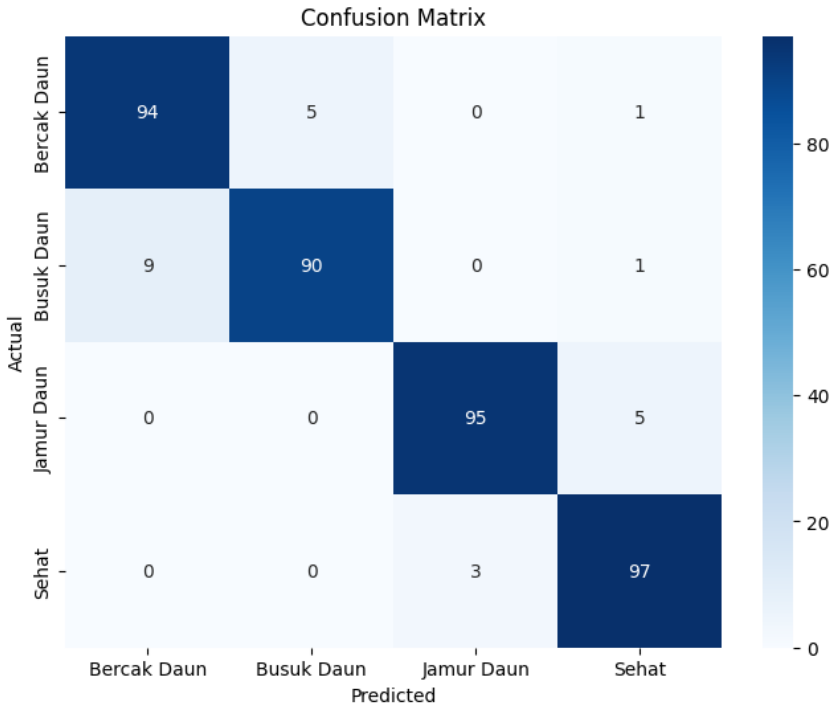
Path ini terjadi ketika model tidak berhasil menghasilkan prediksi yang valid. Hal ini dapat disebabkan oleh beberapa kondisi, antara lain: terjadi error internal selama proses inferensi, *array confidences* yang dihasilkan bernilai kosong atau nol, maupun kondisi *input* citra yang tidak memenuhi kriteria sehingga model tidak mampu mengenali objek sebagai daun tomat. Pada jalur ini, sistem mengembalikan nilai "Tidak Diketahui" sebagai keluaran.

4.2.3 Pengujian Akurasi Model dengan menggunakan *Confusion Matrix*

Pengujian menggunakan *Confusion Matrix* dilakukan dengan tujuan untuk mengevaluasi kinerja model klasifikasi secara menyeluruh, bukan hanya berdasarkan nilai akurasi keseluruhan, melainkan juga untuk mengidentifikasi pola kesalahan prediksi yang terjadi antar kelas. Dalam konteks klasifikasi penyakit daun tomat, gejala visual antar kelas seringkali memiliki kemiripan yang tinggi, sehingga model berpotensi keliru dalam membedakan satu kelas

dengan kelas lainnya. *Confusion Matrix* memungkinkan analisis yang lebih rinci terhadap nilai *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) pada setiap kelas, sehingga dari nilai-nilai tersebut dapat diturunkan metrik evaluasi lanjutan berupa *Accuracy*, *Precision*, *Recall*, dan *F1-Score*. Dengan demikian, penggunaan *Confusion Matrix* pada penelitian ini bertujuan untuk memberikan gambaran yang komprehensif dan objektif mengenai kekuatan serta kelemahan model dalam mengklasifikasikan masing-masing kategori penyakit daun tomat.

Pengujian akurasi model dilakukan menggunakan data pengujian sebanyak 400 citra yang terdiri dari masing-masing 100 citra per kelas. *Confusion matrix* digunakan untuk melihat perbandingan antara kelas aktual dan kelas hasil prediksi secara menyeluruh. Melalui *Confusion matrix* ini, dapat diketahui kelas yang paling banyak diprediksi dengan benar serta kelas yang masih berpotensi mengalami kesalahan prediksi akibat kemiripan gejala visual antar penyakit.



Gambar 4. 21 *Confusion Matrix* Model Klasifikasi Daun Tomat

Berdasarkan hasil *Confusion matrix* yang diperoleh, nilai pada diagonal utama menunjukkan jumlah citra yang berhasil diprediksi secara benar pada setiap kelas. Semakin besar nilai pada diagonal utama, semakin tinggi kemampuan model dalam mengenali kelas tersebut. Nilai di luar diagonal utama merepresentasikan kesalahan prediksi yang terjadi antarkelas, yang dapat disebabkan oleh kemiripan pola visual, variasi kualitas citra, kondisi pencahayaan, serta keragaman bentuk gejala pada daun tomat.

Dari *Confusion matrix* tersebut, dapat dihitung beberapa metrik evaluasi penting, yaitu:

1. Bercak Daun:

$$Precision = \frac{94}{94+9} = \frac{94}{103} = 0.9126$$

$$Recall = \frac{94}{94+6} = \frac{94}{100} = 0.9400$$

$$F1\text{-}Score = \frac{2 \times 0.9126 \times 0.9400}{0.9126 + 0.9400} = 0.9261$$

2. Busuk Daun:

$$Precision = \frac{90}{90+5} = \frac{90}{95} = 0.9474$$

$$Recall = \frac{90}{90+10} = \frac{90}{100} = 0.9000$$

$$F1\text{-}Score = \frac{2 \times 0.9474 \times 0.9000}{0.9474 + 0.9000} = 0.9231$$

3. Jamur Daun:

$$Precision = \frac{95}{95+3} = \frac{95}{98} = 0.9694$$

$$Recall = \frac{95}{95+5} = \frac{95}{100} = 0.9500$$

$$F1\text{-}Score = \frac{2 \times 0.9694 \times 0.9500}{0.9694 + 0.9500} = 0.9596$$

4. Sehat:

$$Precision = \frac{97}{97 + 7} = \frac{97}{104} = 0.9327$$

$$Recall = \frac{97}{97 + 3} = \frac{97}{100} = 0.9700$$

$$F1\text{-}Score = \frac{2 \times 0.9327 \times 0.9700}{0.9327 + 0.9700} = 0.9510$$

Accuracy dihitung berdasarkan keseluruhan prediksi yang tepat:

$$Accuracy = \frac{\text{Jumlah Prediksi Benar}}{\text{Total Data}} = \frac{376}{400} = 0.9400 \ (94.00\%)$$

Akurasi keseluruhan model mencapai 94.00%, yang menunjukkan kemampuan model dalam mengidentifikasi kelas penyakit daun tanaman dengan sangat baik. Rata-rata nilai *precision*, *recall*, dan *F1-Score* untuk keempat kelas mencapai sekitar 94.0%, yang mengindikasikan keseimbangan performa klasifikasi yang baik di antara seluruh kelas.

4.2.4 Pengujian User Acceptance *Test*

UAT merupakan tahap evaluasi di mana sejumlah responden diminta untuk mencoba aplikasi secara langsung dan memberikan penilaian terhadap berbagai aspek, meliputi fungsionalitas, kemudahan penggunaan, tampilan antarmuka, serta kinerja sistem secara menyeluruh. Pengujian ini bertujuan untuk mengukur sejauh mana aplikasi mampu memenuhi kebutuhan dan harapan pengguna dalam konteks identifikasi penyakit daun tomat di lapangan.

Pengujian UAT dilaksanakan dengan melibatkan petani sebagai responden yang berpartisipasi secara langsung dalam proses pengujian aplikasi. Kuesioner disebarkan melalui tautan Google Form kepada para responden. Setiap responden diminta untuk mencoba seluruh fitur aplikasi secara mandiri, kemudian mengisi kuesioner yang memuat 10 pernyataan yang mencerminkan aspek usability, akurasi, dan kebermanfaatan aplikasi. Penilaian menggunakan skala lima poin, di mana setiap jawaban memiliki bobot nilai

yang berbeda guna memudahkan proses analisis dan evaluasi secara kuantitatif. Rincian bobot nilai jawaban disajikan pada tabel berikut:

Tabel 4. 4 Bobot Nilai Jawaban Kuesioner UAT

Jawaban	**Keterangan**	**Bobot**
STS	Sangat Tidak Setuju	1
TS	Tidak Setuju	2
N	Netral	3
S	Setuju	4
SS	Sangat Setuju	5

Adapun hasil data dan jawaban pengujian UAT yang diperoleh dari seluruh responden adalah sebagai berikut:

Tabel 4. 5 Data dan Jawaban Pengujian UAT

No	Pernyataan	STS	TS	N	S	SS
1	Aplikasi mudah digunakan tanpa memerlukan pelatihan khusus	0	0	0	6	9
2	Tampilan antarmuka aplikasi mudah dipahami dan tidak membingungkan	0	0	1	6	8
3	Hasil klasifikasi penyakit ditampilkan dengan jelas dan mudah dibaca	0	0	0	6	9
4	Hasil klasifikasi penyakit yang ditampilkan aplikasi sesuai dengan kondisi daun tomat yang saya amati	0	0	0	8	7
5	Aplikasi mampu membedakan jenis penyakit daun tomat dengan baik	0	0	0	5	10
6	Aplikasi memberikan hasil yang konsisten saat digunakan berulang kali	0	0	1	7	7

7	Aplikasi merespons dengan cepat setelah foto daun diunggah atau diambil	0	0	1	6	8
8	Aplikasi berjalan lancar tanpa mengalami gangguan atau error	0	0	0	7	8
9	Aplikasi ini membantu saya mengetahui jenis penyakit daun tomat lebih cepat dibandingkan cara manual	0	0	0	6	9
10	Aplikasi ini membantu saya menentukan penanganan yang lebih tepat terhadap penyakit daun tomat	0	0	0	4	11

Berdasarkan data yang tersaji pada Tabel 4.5, setiap jawaban responden selanjutnya dikalikan dengan bobot yang telah ditetapkan. Proses ini dilakukan untuk memperoleh skor numerik dari masing-masing pernyataan, yang kemudian dijumlahkan untuk mendapatkan total skor keseluruhan. Hasil perhitungan tersebut dirangkum dalam tabel berikut:

Tabel 4. 6 Hasil Pengujian UAT

No	Nilai					Jumlah
	STS×1	TS×2	N×3	S×4	SS×5	
1	0	0	0	24	45	69
2	0	0	3	24	40	67
3	0	0	0	24	45	69
4	0	0	0	32	35	67
5	0	0	0	20	50	70
6	0	0	3	28	35	66
7	0	0	3	24	40	67
8	0	0	0	28	40	68
9	0	0	0	24	45	69
10	0	0	0	16	55	71

Total	0	0	9	244	430	683

Berdasarkan Tabel 4.6, total skor yang diperoleh dari hasil pengujian mencapai 448, sementara skor maksimum yang dapat dicapai adalah 500. Nilai skor maksimum tersebut diperoleh dari hasil perkalian antara jumlah pernyataan (10 pernyataan), jumlah responden (15 responden), dan bobot nilai tertinggi (5), sehingga skor maksimum = 10 × 15 × 5 = 750. Setelah diketahui nilai skor maksimalnya, maka persentase kelayakan keseluruhan dapat dihitung menggunakan rumus berikut:

$$\text{Persentase Kelayakan}(\%) = \frac{\text{Skor Hasil Pengujian}}{\text{Skor yang Diharapkan}} \times 100\%$$

$$= \frac{683}{750} \times 100\%$$

$$= 91{,}07\%$$

Berdasarkan hasil pengujian UAT yang telah dilaksanakan berdasarkan kuesioner tingkat kepuasan pengguna, aplikasi klasifikasi penyakit daun tomat mendapatkan total skor 683 dari skor maksimum 750, sehingga persentase kelayakan yang diperoleh sebesar 91,07%. Hasil ini menunjukkan bahwa aplikasi mendapatkan respons yang sangat positif dari pengguna dan mampu memenuhi harapan pengguna dengan baik. Persentase tersebut mengindikasikan bahwa secara keseluruhan aplikasi ini layak digunakan sebagai alat bantu dalam mengidentifikasi penyakit daun tomat secara cepat, akurat, dan mudah diakses oleh petani maupun pengguna umum melalui perangkat Android tanpa memerlukan keahlian teknis khusus.