BAB III
ANALISIS DAN PERANCANGAN

1.1 Analisis

1.1.1 Identifikasi Masalah

Tanaman buah jeruk memiliki peran yang sangat krusial dalam kehidupan masyarakat, tidak hanya sebagai sumber makanan, tetapi juga menawarkan banyak manfaat bagi kesehatan. Di antara berbagai jenis buah, jeruk dikenal luas sebagai salah satu buah yang kaya akan vitamin dan nutrisi, sehingga sangat bermanfaat bagi kesehatan tubuh. Menurut data dari Badan Pusat Statistik, varietas jeruk siam mencatatkan diri sebagai komoditas buah yang paling banyak diproduksi, mengungguli varietas jeruk lainnya dalam hal volume produksi.

Namun, meskipun jeruk siam memiliki potensi yang besar, sering kali produksi buah ini mengalami penurunan yang signifikan. Salah satu penyebab utama dari penurunan hasil produksi jeruk adalah berbagai penyakit yang menyerang tanaman. Penyakit-penyakit ini tidak hanya dapat merusak tanaman secara fisik, tetapi juga dapat mempengaruhi kualitas buah yang dihasilkan, yang berdampak pada pendapatan para petani dan ketersediaan pasokan di pasar. Oleh karena itu, sangat penting untuk melakukan penelitian dan pengembangan yang bertujuan mendeteksi dan mengatasi penyakit-penyakit yang menyerang tanaman jeruk, agar produktivitas dan keberlangsungan budidaya jeruk dapat terjaga.

Penyakit tersebut biasanya dideteksi secara manual dengan mengenal gejala-gejala mulai dari daun gugur hingga buah masih kecil sudah jatuh. Namun hal tersebut biasanya memakan waktu lama atau menunggu gejala timbul secara

jelas sampai dapat ditentukan penyakit apa yang menyerang tanaman varietas jeruk siam tersebut. Permasalahan yang diajukan dalam penelitian ini dapat diidentifikasi melalui beberapa aspek yang perlu diperhatikan. Berikut adalah daftar permasalahan yang menjadi fokus utama dalam penelitian ini:

1. Identifikasi penyakit tanaman pada varietas jeruk siam masih dilakukan secara manual atau tanpa bantuan alat apapun, sehingga deteksi penyakit yang dilakukan kurang efektif.
2. Petani terkadang masih bingung dan tidak mengerti jenis penyakit apa yang terserang, sehingga asal memberikan obat atau pupuk yang mungkin terkesan sia-sia karena kurang jelasnya penyakit yang diserang pada tanamannya.

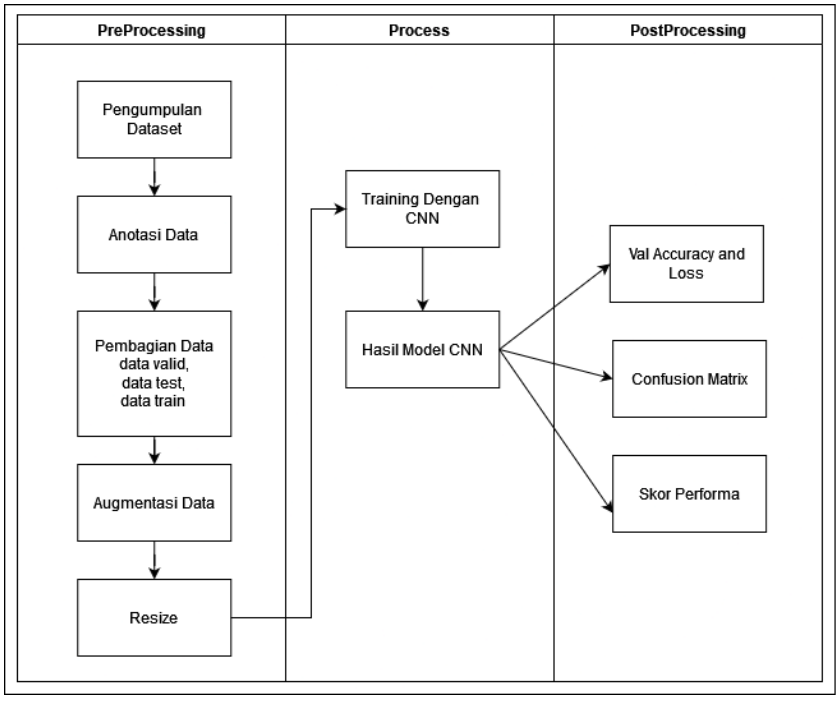
1.1.2 Pemecahan Masalah

Berdasarkan permasalahan yang ada, maka peneliti melakukan sebuah penelitian sebuah algoritma metode CNN yang dinilai mampu membantu mengklasifikasikan penyakit pada varietas tanaman jeruk secara efektif guna memberikan bantuan kepada para petani maupun masyarakat dalam mengidentifikasi penyakit pada varietas tanaman jeruk, dimana identifikasi penyakit berdasarkan foto saja.

1.2 Perancangan

1.2.1 Perancangan Sistem CNN

Pada penelitian penerapan metode CNN untuk mengklasifikasikan penyakit tanaman jeruk siam terdapat 3 proses yang dilakukan yaitu pre-processing, process dan post-processing yang tergambarkan beberapa tahapan yaitu sebagai berikut:



Gambar 3.1 Blok Diagram Sistem CNN

3.2.1.1 Pengumpulan Dataset

Pengumpulan dataset merupakan langkah penting dalam proses penelitian ini, di mana data yang akan digunakan untuk mengidentifikasi berbagai jenis penyakit pada varietas jeruk siam dikumpulkan(Rofiqoh, 2023). Dalam penelitian ini, dataset yang digunakan diperoleh melalui pengambilan data secara langsung di kebun jeruk siam, yang memastikan keaslian dan relevansi terhadap konteks penelitian. Fokus utama dari penelitian ini adalah untuk mengklasifikasikan citra

ke dalam 5 kategori penyakit yang berbeda, serta 3 kategori yang menunjukkan kondisi sehat. Dengan pendekatan ini, diharapkan dapat diperoleh informasi yang komprehensif mengenai kondisi kesehatan tanaman jeruk, sehingga dapat membantu dalam pengembangan solusi untuk mengatasi masalah penyakit yang sering menyerang tanaman tersebut. Pengumpulan data yang cermat dan sistematis ini sangat penting untuk memastikan akurasi dan efektivitas dari model yang akan dikembangkan dalam penelitian ini.

Tabel 3.1 Jumlah dataset

Nama Penyakit/*Class*	Jumlah Data
Batang Blendok	270
Batang Sehat	240
Buah Antraknosa	320
Buah Sehat	240
Daun Brownspot	240
Daun Ulat Peliang	320
Daun Greasyspot	270
Daun Sehat	240
Total	**2140**

Data set diatas digunakan untuk proses training dan testing. Berikut adalah Contoh dari beberapa dataset yang akan digunakan:

Gambar 3.2 Contoh Gambar Dataset

3.2.1.2 Anotasi Data

Anotasi gambar memainkan peran kunci dalam pelabelan posisi dan kelas bintik-bintik objek dalam penyakit dan gambar yang sehat untuk deteksi objek multikelas(Rofiqoh, 2023). Dalam computer vision. Proses anotasi data melibatkan manusia yang secara manual melakukan tugas ini atau menggunakan algoritma komputer untuk melakukan anotasi secara otomatis. Anotasi data dapat berupa berbagai jenis informasi, tergantung pada jenis data yang sedang diolah. Contohnya, dalam pengenalan objek pada gambar, anotasi data dapat mencakup lokasi dan klasifikasi objek dalam gambar.

```
Found 1498 validated image filenames belonging to 8 classes.
Found 322 validated image filenames belonging to 8 classes.
Found 320 validated image filenames belonging to 8 classes.
0. batang_blendok
1. batang_sehat
2. buah_antraknosa
3. buah_sehat
4. daun_brownspot
5. daun_clminer
6. daun_greasyspot
7. daun_sehat
```

Gambar 3.3 Contoh Anotasi

3.2.1.3 Pembagian Data

Tahap ini merupakan tahap persiapan untuk pengolahan data dan klasifikasi data. Pembagian setiap data pada jenis penyakit dibagi menjadi data training dan data testing dan juga data valid(Baiq Nurul Azmi et al., 2023). Data dibagi dengan perbandingan tertentu untuk mengetahui model terbaik dalam pelatihan. Data training digunakan untuk menjalankan CNN dalam pencarian model yang terbaik, data testing digunakan untuk menguji model yang telah

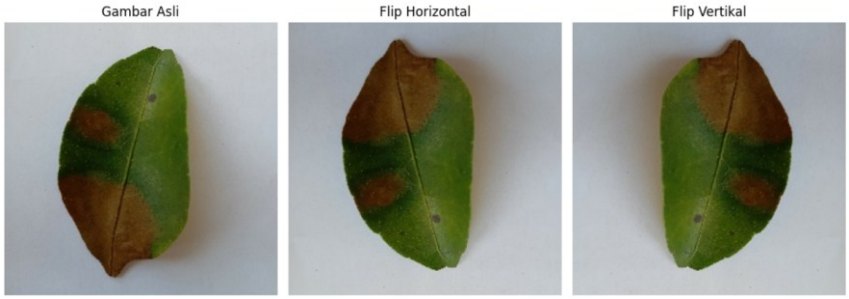
dihasilkan pada proses training, sedangkan data validasi digunakan untuk mengevaluasi kinerja model selama pelatihan. Data ini tidak digunakan untuk melatih model secara langsung, melainkan untuk memonitor dan menyesuaikan hyperparameter model, seperti tingkat pembelajaran atau jumlah lapisan.

Tabel 3.2 Contoh Pembagian Data

Jenis Data	Data *Training* (80%)	Data *Testing* (10%)	Data Valid (10%)
Batang Blendok	100	25	25
Batang Sehat	100	25	25
Buah Antraknosa	100	25	25
Buah Sehat	100	25	25
Daun Brownspot	100	25	25
Daun Ulat Peliang	100	25	25
Daun Greasyspot	100	25	25
Daun Sehat	100	25	25
Total	**800**	**200**	**200**

3.2.1.4 Augmentasi Data

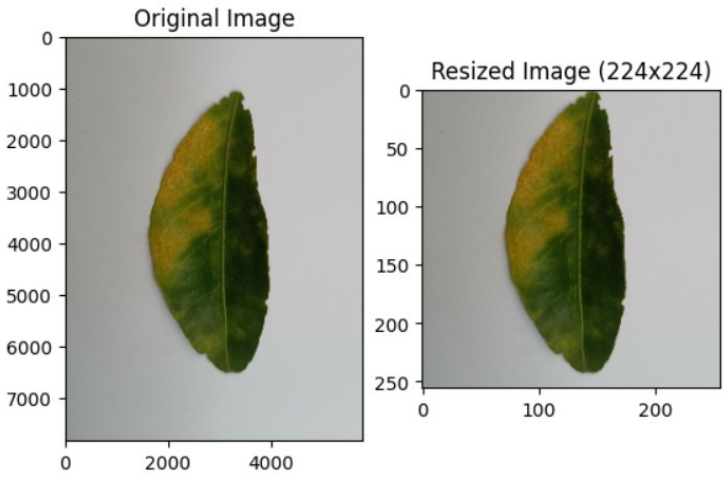
Pada perancangan Augmentasi data adalah proses memperluas himpunan data untuk meningkatkan kinerja model dengan menghasilkan berbagai bentuk gambar. Ini juga berguna dalam mengurangi masalah overfitting dalam model selama tahap pelatihan. Masalah overfitting terjadi ketika ada kebisingan atau kesalahan acak, daripada hubungan yang mendasarinya. Dengan bantuan augmentasi data, lebih banyak gambar dihasilkan dari setiap gambar untuk melatih model karena beberapa pola yang tidak relevan dapat terjadi selama proses pelatihan modelnya(Putri Ayuni et al., 2023). Untuk operasi augmentasi data digunakan beberapa teknik yaitu rescale dan flip.



Gambar 3.4 Contoh Gambar Augmentasi

3.2.1.5 Resize Data

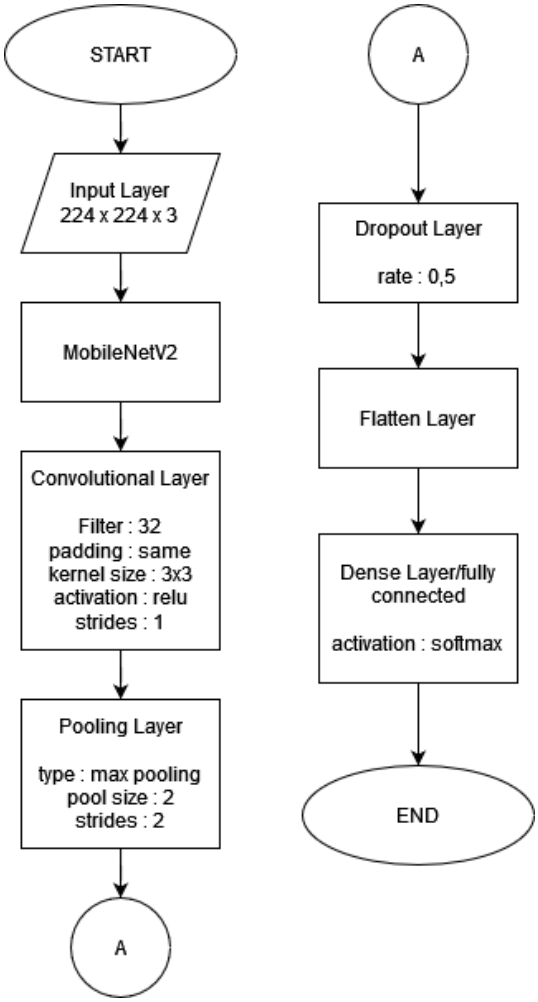
Resize data merupakan proses yang dilakukan untuk mengubah ukuran gambar dari ukuran aslinya menjadi ukuran yang diinginkan. Dalam penelitian ini, proses resize data dilakukan dengan mengubah ukuran gambar menjadi lebar 224 piksel dan tinggi 224 piksel(Rofiqoh, 2023). Pengubahan ukuran ini sangat penting, terutama dalam konteks pemrosesan citra untuk model pembelajaran mesin, karena model umumnya memerlukan citra dengan dimensi yang konsisten agar dapat memproses dan menganalisis data dengan efektif. Dengan menetapkan ukuran gambar yang sama, yaitu 224x224 piksel, dapat memastikan bahwa semua input yang diberikan ke model memiliki spesifikasi yang seragam, sehingga memudahkan proses pelatihan dan meningkatkan akurasi prediksi. Selain itu, pengubahan ukuran ini juga membantu mengurangi kompleksitas komputasi dan mempercepat waktu pemrosesan, sehingga memungkinkan model untuk lebih efisien dalam menangani berbagai gambar yang akan dianalisis(Putri Ayuni et al., 2023).



Gambar 3.5 Contoh Gambar Resize

3.2.1.6 Proses Training

Proses training data dilakukan menggunakan metode convolution neural network. Pada penelitian ini terdiri dari beberapa layer yaitu input layer, MobilenetV2, convolution layer, pooling layer, dropout layer, flatten layer, dan fully connected layer dengan output layer seperti flowchart alur berikut:



Gambar 3.6 Alur Pemodelan CNN

Dalam memaksimalkan hasil uji coba dilakukan beberapa uji coba untuk menenmukan hasil yang terbaik, kemudian juga dengan menentukan *hyperparameter* seperti dibawah ini.

Tabel 3.3 Hyperparameter

NO	Hyperparameter	
1	Split Data	Data Train 70%, Val 15%, Test 15%
2	Batch Size	64
3	Layer	6
4	Epoch	50
5	Optimizer	Adam
6	Learning Rate	0,001

Dalam tabel di atas, dataset dibagi menjadi tiga bagian: 70% untuk data pelatihan (train), 15% untuk data validasi (val), dan 15% untuk data pengujian (test). Selama proses pelatihan, digunakan ukuran batch, yaitu jumlah sampel data yang dikirim ke neural network, sebanyak 64. Artinya, jika memiliki total 1000 dataset, algoritma CNN akan mengambil 64 sampel data pertama dari 1000 data yang ada untuk dilatih. Proses ini akan terus berlanjut, di mana neural network akan melatih 64 sampel tersebut hingga selesai, lalu mengambil 64 sampel berikutnya, dan seterusnya, sampai semua data telah digunakan.

Selanjutnya, terdapat informasi mengenai arsitektur model yang terdiri dari 6 layer. Ini mencakup layer base model MobileNetV2, layer konvolusi, layer max pooling, layer dropout, layer flatten, dan layer fully connected. Penggunaan layer-layer ini bertujuan untuk memaksimalkan ekstraksi fitur dan mengurangi risiko overfitting.

Dalam proses pelatihan, digunakan optimizer Adam, yang berfungsi untuk memperbarui bobot model secara iteratif berdasarkan data pelatihan yang tersedia. Optimizer ini dikenal efektif dalam mempercepat konvergensi model. Selain itu, learning rate yang digunakan adalah 0.001. Pengaturan learning rate yang lebih kecil ini memungkinkan model untuk mempelajari data pelatihan dengan lebih detail dan cermat, sehingga diharapkan dapat meningkatkan akurasi dan performa model dalam mengidentifikasi penyakit pada tanaman jeruk. Dengan kombinasi ini, model diharapkan dapat belajar dari data dengan lebih efektif dan memberikan hasil yang optimal.

Adapun tahapan tahapan pada gambar 3.6 akan dijelaskan secara ringkas sebagai berikut:

3.2.1.6.1 Input Layer

Citra dengan dimensi 224 x 224 x 3. Kemudian data itu diproses kedalam input layer dengan nilai pixel dan channel RGB.

3.2.1.6.2 Model MobileNetV2

MobileNetV2 adalah salah satu models CNN yang tersedia di keras dengan akurasi terbaik saat ini untuk mobile. Model ini juga sering digunakan sebagai arsitektur CNN karena memiliki kelebihan yaitu dalam proses training memerlukan beban komputasi yang tidak terlalu tinggi, serta model yang dihasilkan lebih kecil namun tetap dengan performa terbaik sehingga dapat mempermudah dalam deployment si aplikasi android ini.

Model MobileNetV2 yang telah dilatih sebelumnya dengan pendekatan transfer learning kini akan diterapkan pada dataset yang berfokus pada penyakit tanaman jeruk. Pada tahap ini, input yang diberikan berupa citra dengan ukuran 224x224 piksel dan memiliki 3 channel warna (merah, hijau, dan biru). Output yang dihasilkan dari tahapan konvolusi pertama akan digunakan sebagai input untuk tahapan konvolusi kedua, dan proses ini akan berlanjut untuk tahapan konvolusi berikutnya. Dengan menggunakan arsitektur MobileNetV2 yang efisien, model dapat melakukan ekstraksi fitur secara mendalam dari setiap lapisan konvolusi, sehingga meningkatkan kemampuan dalam mengenali

pola-pola yang berkaitan dengan penyakit pada tanaman jeruk. Proses berkelanjutan ini memungkinkan pengolahan informasi yang lebih kompleks dan detail, yang pada akhirnya berkontribusi pada peningkatan akurasi dalam klasifikasi hasil yang diinginkan.

3.2.1.6.3 Convolutional Layer

Setelah penambahan model MobileNetV2, memasuki tahap selanjutnya, yaitu proses konvolusi. Pada tahapan ini, akan memanfaatkan input yang telah diperoleh sebelumnya untuk melakukan operasi konvolusi. Konvolusi dalam tahap ini akan menggunakan 32 filter dengan ukuran kernel 3x3, serta menerapkan padding 'same' (0) untuk memastikan ukuran output tetap sama dengan ukuran input. Selain itu, fungsi aktivasi yang digunakan adalah ReLU (Rectified Linear Unit), dan pergeseran (stride) yang diterapkan adalah 1. Dengan pengaturan ini, konvolusi akan mengoptimalkan proses ekstraksi fitur dari input, memungkinkan model untuk menangkap pola-pola penting yang ada dalam citra secara efisien. Proses ini merupakan langkah kunci dalam mendeteksi dan mengklasifikasikan informasi yang relevan dari citra untuk mencapai akurasi yang lebih baik dalam aplikasi yang kembangkan.

Untuk memudahkan penjelasan dan penulisan mengenai proses perhitungan pada convolution layer, penulis akan memberikan contoh perhitungan yang menggunakan citra berukuran 4x4 dengan format RGB. Dalam contoh ini, proses konvolusi akan dilakukan dengan menerapkan nilai filter pada matriks. Citra yang terdiri dari tiga channel merah (red),

hijau (green), dan biru (blue) akan diambil data pixelnya masing-masing. Selain itu, pada setiap pixel akan ditambahkan padding nol (0) untuk memastikan bahwa ukuran output citra tetap sama dengan ukuran input, yang dikenal sebagai padding 'same'. Dengan melakukan penambahan padding ini, dapat menjaga informasi di tepi citra dan memastikan bahwa seluruh bagian citra dapat terproses dengan baik. Melalui proses ini, akan mendapatkan hasil pixel yang lebih lengkap dan detail untuk setiap channel, yang akan digunakan dalam tahapan berikutnya dalam jaringan syaraf konvolusi:

Red

0	0	0	0	0	0
0	70	35	28	42	0
0	45	25	30	48	0
0	31	28	32	49	0
0	30	30	35	50	0
0	0	0	0	0	0

Green

0	0	0	0	0	0
0	107	59	45	58	0
0	76	42	45	65	0
0	51	41	46	65	0
0	42	41	48	64	0
0	0	0	0	0	0

Blue

0	0	0	0	0	0
0	112	69	51	59	0
0	79	48	47	63	0
0	56	43	46	60	0
0	47	42	46	59	0
0	0	0	0	0	0

Gambar 3.7 Nilai Pixel RGB

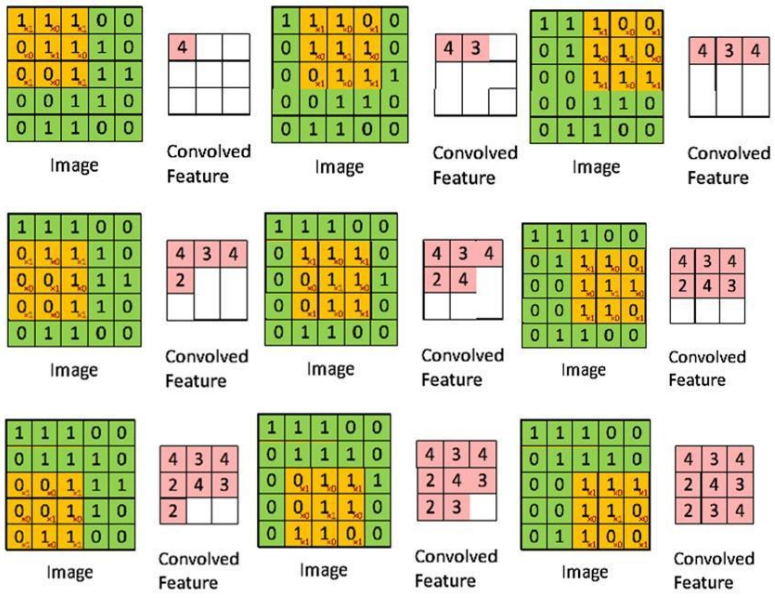
Setelah nilai pixel yang dipecah selanjutnya untuk kernel 3x3 dengan nilai seperti dibawah:

Kernel

1	0	-1
1	0	-1
1	0	-1

Gambar 3.8 Nilai Kernel

Langkah selanjutnya dalam proses ini adalah melakukan perhitungan di setiap channel dengan mengalikan citra dengan kernel berukuran 3x3, seperti yang ditunjukkan pada gambar 3.9. Proses ini dilakukan secara berulang, di mana kernel akan bergerak sebanyak 1 stride pada setiap channel, sehingga memungkinkan untuk memperoleh perhitungan yang dilakukan di setiap channel secara terpisah seperti alur contoh pada gambar 3.9 dibawah ini:



Gambar 3.9 Contoh Perhitungan

Setelah dihitung didapatkanlah nilai setiap channel untuk nilai red, green, dan blue seperti gambar dibawah:

Hasil Red

-60	57	-30	58
--88	56	-51	90
-83	9	-64	97
-58	-6	-41	67

Hasil Green

-101	93	-22	90
-142	98	-46	136
-124	30	-70	139
-58	-6	-41	67

Hasil Blue

-101	93	-5	98
-160	103	-22	144
-133	43	-49	139
-85	11	-34	92

Gambar 3.10 Hasil Perhitungan di Setiap RGB

Setelah dihitung dan didapatkanlah nilai setiap channel maka selanjutnya akan dijumlahkan sehingga menghasilkan output hasil total konvolusi pada gambar 3.11 dengan rumus total=R+G+B

Hasil Total RGB

-278	243	-57	246
-390	257	-119	370
-340	82	-183	375
-225	4	-122	253f

Gambar 3.11 Hasil Perhitungan di Setiap RGB

Kemudian setelah mendapatkan total nilai konvolusi dilakukanlah ReLu yaitu setiap nilai negative dijadikan 0, sehingga didaptkanlah nilai seperti dibawah ini:

-278	243	-57	246
-390	257	-119	370
-340	82	-183	375
-225	4	-122	253

$f(x) = \max(0, x)$

0	243	0	246
0	257	0	370
0	82	0	375
0	4	0	253

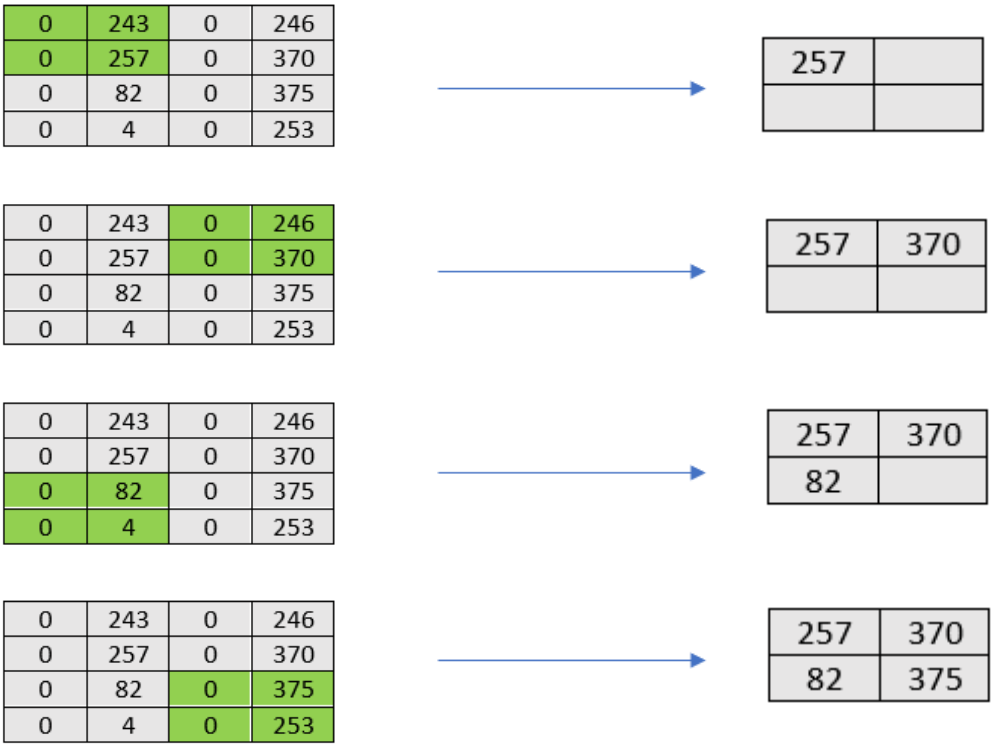
Gambar 3.12 Hasil ReLu

Proses ini dilakukan berulang kali untuk semua citra menggunakan 32 jenis filter yang berbeda. Sebagai hasil dari tahapan konvolusi ini, akan dihasilkan banyak feature map, dan gambar yang ditampilkan di atas adalah salah satu dari feature map tersebut.

3.2.1.6.4 Pooling Layer

Setelah itu, hasil dari fungsi aktivasi ReLU pada output dari tahapan konvolusi akan digunakan sebagai input untuk pooling layer. Pada tahap ini, metode yang digunakan adalah max pooling dengan ukuran kernel 2x2 dan stride 2. Tahapan max pooling berfungsi untuk

mengurangi dimensi data dengan mengambil nilai tertinggi dari area yang ditentukan oleh kernel. Proses ini tidak hanya membantu mengurangi kompleksitas komputasi, tetapi juga mempertahankan fitur yang paling penting dari data. Ilustrasi dari proses max pooling ini akan ditunjukkan di bawah ini, yang menggambarkan bagaimana nilai maksimum dari setiap jendela kernel diambil untuk menghasilkan output yang lebih ringkas namun informatif. Ilustrasi proses max pooling ditunjukkan seperti dibawah ini:



Gambar 3.13 Hasil Max Pooling

3.2.1.6.5 Dropout Layer

Lapisan Dropout adalah teknik yang digunakan dalam jaringan saraf untuk mencegah overfitting. Dengan cara selama pelatihan, lapisan Dropout secara acak menghapus sejumlah neuron dengan probabilitas tertentu (dropout rate).

Misalnya, dengan dropout rate 0.5, setiap neuron memiliki kemungkinan 50% untuk dihapus pada setiap iterasi pelatihan.

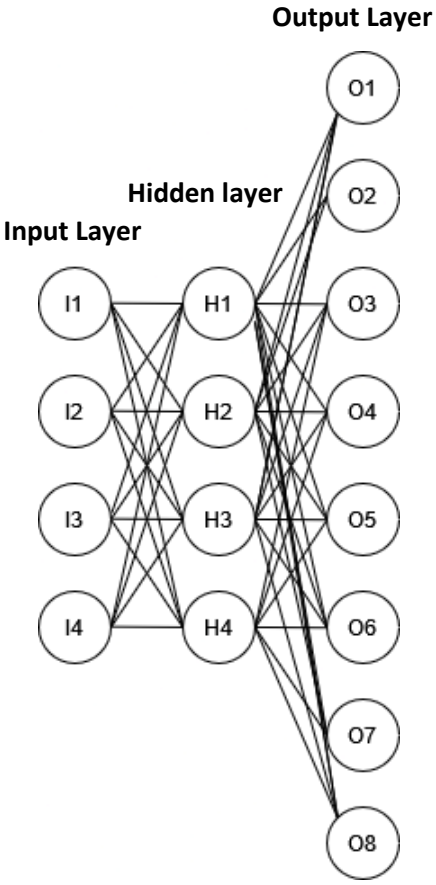
3.2.1.6.6 Flatten Layer

Gambar 3.14 Hasil Flatten

Flatten merupakan tahapan yang mengubah matriks multi-dimensi menjadi vektor atau matriks 1 dimensi. Dalam konteks jaringan saraf tiruan flatten digunakan untuk meratakan (flattening) output dari layer konvolusional atau pooling agar bisa dihubungkan ke fully connected layer (dense layer).

3.2.1.6.7 Fully Connected

Selanjutnya output dari Flatten akan menjadi inputan bagi fully connected layer untuk ilustrasi tentang fully connected dan softmax seperti dibawah ini:



Gambar 3.15 Fully Connected

Selanjutnya tahapan akhir dari model CNN yaitu dengan menghitung input layer dikalikan bobot dan ditambah bias agar menghasilkan output layer yang akan digunakan untuk aktivasi softmax sebagai prediksi dengan perhitungan berikut:

$$output = x \cdot W + b$$

Gambar 3.16 Rumus Output

Keterangan:

X = vektor input

W = bobot

B = bias

Bobot dan bias dalam CNN dihasilkan secara acak pada awal pelatihan dan diperbarui selama proses pembelajaran tetapi untuk memudahkan pembelajaran berikut ini cara perhitungannya:

Neuron 1:

$257\times0.001+370\times0.001+82\times0.001+375\times0.001=0.257+0.37+0.082+0.375$

$=1.084$

Hasil + bias:

$1.084+0.5=1.584$

Neuron 2:

$257\times0.002+370\times0.001+82\times0.002+375\times0.001=0.514+0.37+0.164+0.375$

$=1.423$

Hasil + bias:

$1.423+1.0=2.423$

Neuron 3:

$257\times0.003+370\times0.001+82\times0.001+375\times0.002=0.771+0.37+0.082+0.75$

$=1.973$

Hasil + bias:

$1.973+0.0=1.973$

Neuron 4:

$257\times0.004+370\times0.002+82\times0.002+375\times0.001=1.028+0.74+0.164+0.375$

$=2.307$

Hasil + bias:

$2.307+0.3=2.607$

Neuron 5:

$257\times0.01+370\times0.002+82\times0.005+375\times0.003=2.57+0.74+0.41+1.125$

$=4.845$

Hasil + bias:

$4.845+0.5=5.345$

Neuron 6:

$257\times0.002+370\times0.003+82\times0.003+375\times0.002=0.514+1.11+0.246+0.75$

$=2.620$

Hasil + bias:

$2.620+0.5=3.120$

Neuron 7:

$257\times0.001+370\times0.001+82\times0.001+375\times0.001=0.257+0.37+0.082+0.375$

$=1.084$

Hasil + bias:

$1.084+0.2=1.284$

Neuron 8:

$257\times0.001+370\times0.001+82\times0.001+375\times0.001=0.257+0.37+0.082+0.375$

$=1.084$

Hasil + bias:

$1.084+0.5=1.584$

Setelah semua perhitungan selesai, maka didapatkan lah hasil akhir output:

$$output = \begin{bmatrix} 1.584 \\ 2.423 \\ 1.973 \\ 2.607 \\ 5.345 \\ 3.120 \\ 1.284 \\ 1.584 \end{bmatrix}$$

Gambar 3.17 Hasil Output

Setelah semua perhitungan selesai maka masuk ke dalam tahap aktivasi softmax untuk mengetahui hasil nilai *predict*.

Output Layer Sebelum Softmax:

output = [1.584, 2.423, 1.973, 2.607, 5.345, 3.120, 1.284, 1.584]

1. Menghitung Eksponensial dari Setiap Elemen dengan rumus

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Gambar 3.18 Rumus Softmax

e^1.584 = 4.867

e^2.423 = 11.193

e^1.973 = 7.212

e^2.607 = 13.593

e^5.345 = 229.927

e^3.120 = 22.93

e^1.284 = 3.605

e^1.584 = 4.867

e^output = [4.867, 11.193, 7.212, 13.593, 229.927, 22.93, 3.605, 4.867]

2. Kemudian menghitung Jumlah Eksponensial:

Jumlah e^output =

4.867 + 11.193 + 7.212 + 13.593 + 229.927 + 22.93 + 3.605 + 4.867

= 293.593.

3. Kemudian menghitung Softmax untuk Setiap Elemen:

Softmax Neuron 1: softmax(z1) = 4.867 / 293.593 = 0.01658

Softmax Neuron 2: softmax(z2) = 11.193 / 293.593 = 0.0381

Softmax Neuron 3: softmax(z3) = 7.212 / 293.593 = 0.0245

Softmax Neuron 4: softmax(z4) = 13.593 / 293.593 = 0.0463

Softmax Neuron 5: softmax(z5) = 229.927 / 293.593 = 0.7824

Softmax Neuron 6: softmax(z6) = 22.93 / 293.593 = 0.0781

Softmax Neuron 7: softmax(z7) = 3.605 / 293.593 = 0.0123

Softmax Neuron 8: softmax(z8) = 4.867 / 293.593 = 0.01658

4. Hasil akhir maka nilai softmax

softmax(output) =

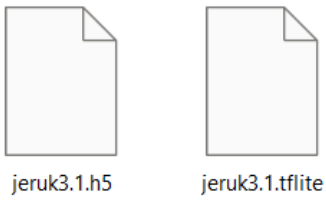
[0.01658, 0.0381, 0.0245, 0.0463, 0.7824, 0.0781, 0.0123, 0.01658]

$$\text{softmax}(output) = \begin{bmatrix} 0.01658 \\ 0.0381 \\ 0.0245 \\ 0.0463 \\ 0.7824 \\ 0.0781 \\ 0.0123 \\ 0.01658 \end{bmatrix}$$

Gambar 3.19 Output Softmax

3.2.1.7 Hasil Model Training

Setelah dilakukan proses training, data dilatih oleh algoritma CNN untuk menghasilkan model yang siap untuk proses testing. Jika model tersebut belum mencapai akurasi yang maksimal, maka sistem akan mentraining ulang model dengan batasan epoch tertentu sampai mendapat akurasi yang maksimal. Setelah model tersebut mencapai akurasi yang diinginkan, model tersebut akan disimpan dalam format h5 untuk memudahkan penyimpanan dan pengelolaan. Selanjutnya, model yang telah disimpan dalam format h5 ini kemudian dikonversi ke dalam format TensorFlow Lite (tflite). Proses konversi ini dilakukan agar model dapat digunakan secara optimal dalam aplikasi mobile dengan tingkat efisiensi yang lebih tinggi serta kompatibilitas yang lebih baik dengan platform Android. Dengan mengonversi model ke format tflite, aplikasi tidak hanya dapat mempercepat proses inferensi, tetapi juga mengurangi konsumsi sumber daya yang diperlukan, seperti memori dan daya baterai. Hal ini sangat penting dalam pengembangan aplikasi mobile, di mana perangkat sering kali memiliki keterbatasan dalam hal daya pemrosesan dan energi. Dengan demikian, pengguna dapat menjalankan aplikasi dengan lebih lancar dan responsif, sambil tetap mempertahankan akurasi dan efektivitas dalam mendeteksi penyakit pada tanaman jeruk.



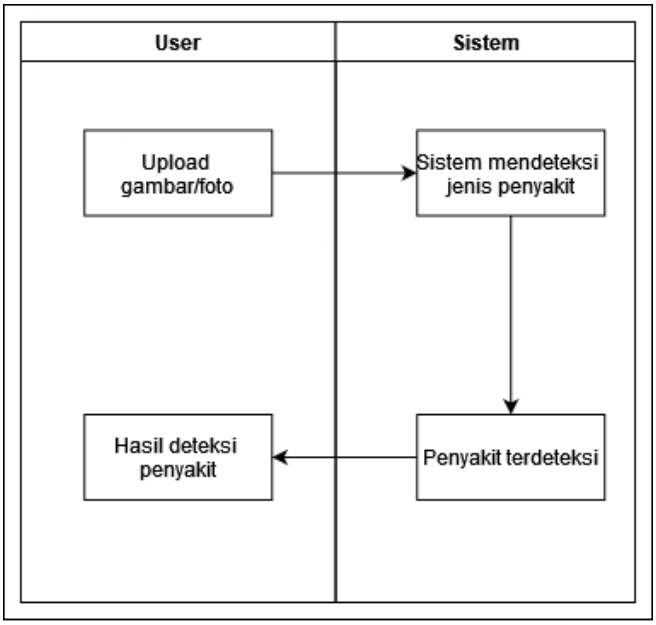
Gambar 3.20 Model H5 dan Tflite

3.2.1.8 Deteksi Penyakit

Proses ini juga dikenal sebagai pengujian model. Tahapan terakhir dalam sistem ini adalah kemampuan untuk mendeteksi dan menganalisis data pengujian menggunakan model yang telah dilatih pada tahap-tahap sebelumnya. Model yang beroperasi pada tahap pengujian ini dirancang untuk mengidentifikasi data yang akan diuji. Pada tahap ini, digunakan data pengujian yang telah dibagi sebelumnya dalam proses pembagian dataset. Penting untuk diingat bahwa proses pengujian ini hanya bertujuan untuk mengevaluasi kinerja model, terutama dalam menilai bobot yang dihasilkan melalui pengujian menggunakan confusion matrix. Dengan menganalisis hasil dari confusion matrix, bisa mendapatkan informasi berharga mengenai akurasi, presisi, recall, dan F1 score model, yang memungkinkan untuk menilai seberapa baik model dalam mengklasifikasikan data yang belum pernah dihadapi sebelumnya. Proses ini sangat penting untuk memastikan bahwa model tidak hanya berfungsi dengan baik pada data pelatihan tetapi juga dapat diandalkan ketika dihadapkan pada data baru.

1.2.2 Perancangan Sistem Android

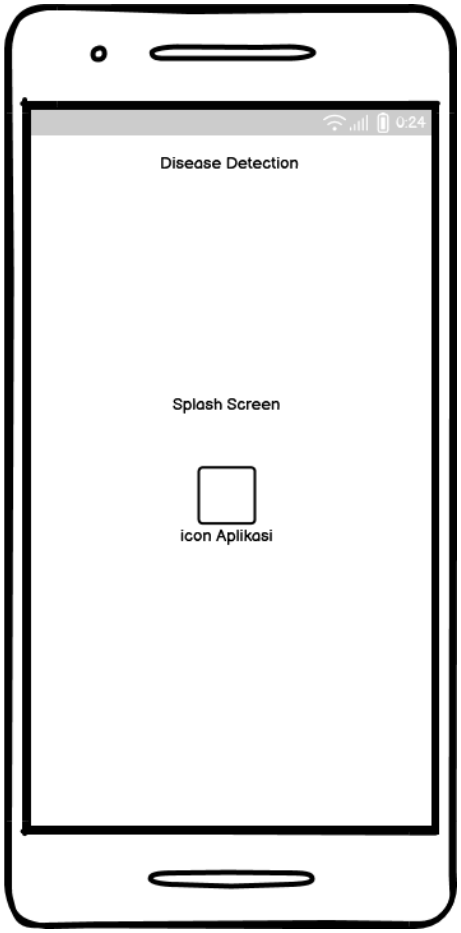
3.2.2.1 Perancangan Use Case



Gambar 3.21 Blok Diagram Android

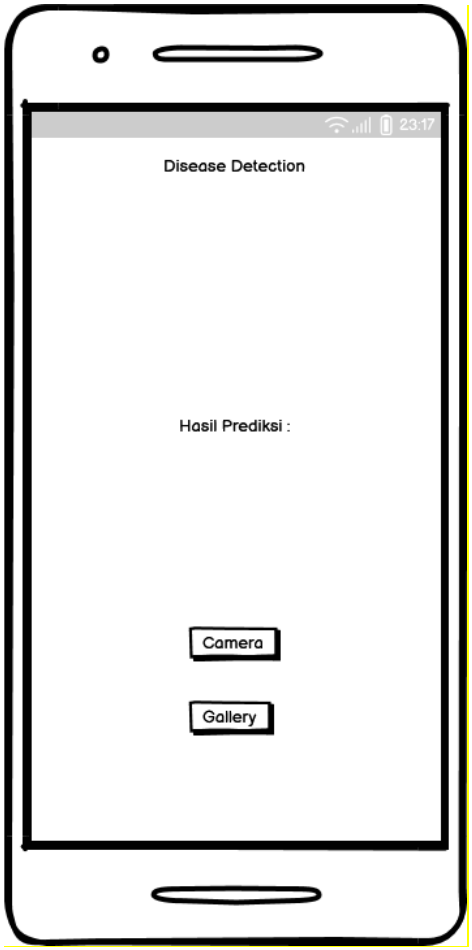
Pada sistem yang dibuat, pengguna hanya dapat mengupload foto atau gambar penyakit. Lalu proses deteksi dilakukan oleh sistem dan hasil akan muncul setelah terdeteksi. Hasil yang ditampilkan setelah proses deteksi adalah jenis penyakit dan keterangan dari penyakit tersebut.

1.2.3 Perancangan User Interface / *Mock-up* aplikasi



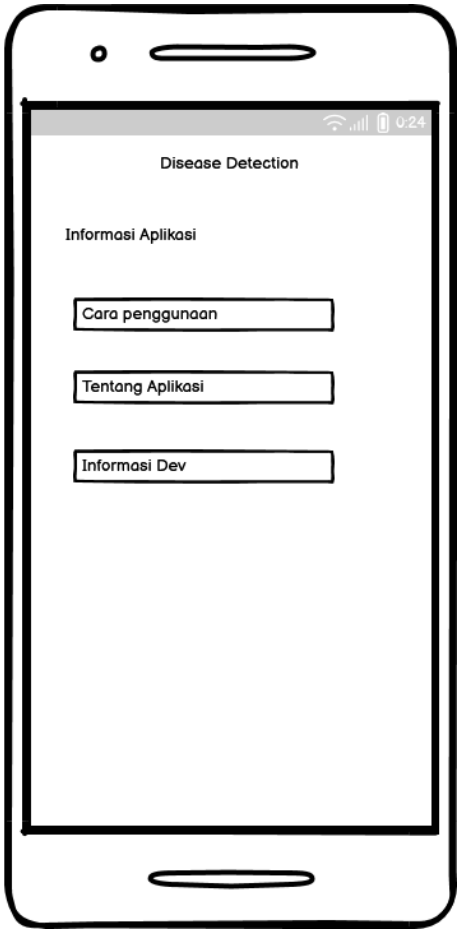
Gambar 3.22 Halaman Splash Screen

Merupakan halaman awal aplikasi ketika pertama kali dibuka. Muncul icon aplikasi yang berfungsi sebagai penanda aplikasi telah dibuka.



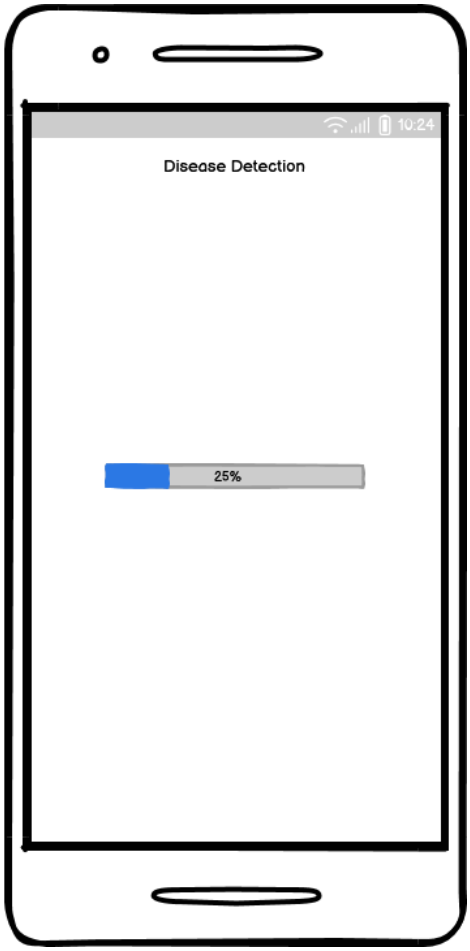
Gambar 3.23 Halaman Awal

Merupakan halaman awal yang digunakan untuk deteksi penyakit. Pada halaman ini menampilkan camera dan gallery untuk memilih foto atau gambar yang akan dideteksi.



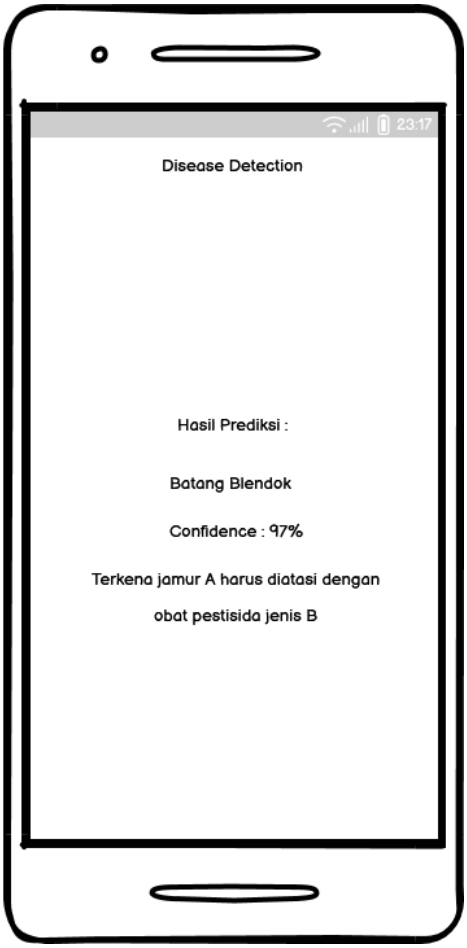
Gambar 3.24 Halaman Informasi

Merupakan halaman informasi yang digunakan untuk mengetahui informasi aplikasi termasuk cara penggunaannya. Pada halaman ini menampilkan informasi yang dibutuhkan user ketika awal penggunaan.



Gambar 3.25 Halaman Progres

Merupakan halaman progres yang digunakan untuk deteksi penyakit. Pada halaman ini menampilkan *progress bar* sebelum menuju ke halaman yang akan dideteksi.



Gambar 3.26 Halaman Hasil

Merupakan halaman hasil foto yang telah dideteksi. Halaman ini menampilkan hasil presentase deteksi dan keterangan terkait penyakit pada varietas jeruk siam.

1.3 Rancangan Pengujian

3.3.1 Pengujian Pembagian Data dan *Epoch*

Rancangan pengujian pada proses training dilakukan pengujian jumlah epoch dan jumlah pembagian data. Pada proses ini dilakukan pembatasan perbandingan dengan jumlah epoch lalu perbandingan jumlah pembagian data.

Tabel 3.4 Parameter Nilai Training

NO	PARAMETER	NILAI
1	*Epoch*	20,50
2	Pembagian data	Data Train: 80%,70%,60%,50% Data Val: 10%,15%,20%,25% Data Test: 10%,15%,20%,25%

3.3.2 Pengujian *Confusion Matrix*

Confusion Matrix digunakan sebagai pengukur kinerja setelah mengolah data mining dengan model klasifikasi. Pada dasarnya, pengukuran dengan Confusion Matrix digunakan untuk memberikan informasi perbandingan dari hasil klasifikasi yang dilakukan oleh algoritma yang digunakan dengan hasil klasifikasi sebenarnya (Romadloni et al., 2022).

Confusion matrix memiliki empat kombinasi nilai prediksi dan nilai aktual yang berbeda, seperti berikut:

Tabel 3.5 Perhitungan Confusion Matrix

		Aktual	
		Terdeteksi Penyakit	**Tidak Terdeteksi Penyakit**
Prediksi	**Terdeteksi Penyakit**	TP	FP
	Tidak Terdeteksi Penyakit	FN	TN

Keterangan:

Pada tabel di atas, terdapat empat istilah yang merepresentasikan hasil dari proses klasifikasi dalam confusion matrix, yaitu:

1. True Positive (TP), yang mengacu pada data positif yang berhasil diprediksi dengan benar oleh model.
2. True Negative (TN), yaitu data negatif yang juga diprediksi dengan benar.
3. False Positive (FP), yang merupakan kesalahan tipe 1, di mana data negatif salah diprediksi sebagai data positif.
4. False Negative (FN), yang merupakan kesalahan tipe 2, di mana data positif tidak teridentifikasi dengan benar dan diprediksi sebagai data negatif.

Setelah memperoleh hasil dari confusion matrix, dapat menghitung berbagai matrik evaluasi model, termasuk nilai akurasi, presisi, dan recall, yang masing-masing memiliki arti penting dalam menilai kinerja model.

1. Nilai akurasi adalah salah satu metrik yang paling umum digunakan untuk menilai kinerja model klasifikasi, yang menggambarkan sejauh mana model dapat mengklasifikasikan data dengan benar

   Akurasi = (TP+TN) / (TP+FP+FN+TN)
2. Presisi adalah metrik yang memberikan informasi tentang sejauh mana model dapat memberikan prediksi yang akurat untuk data positif. Presisi

menggambarkan rasio antara jumlah prediksi positif yang benar dengan total prediksi positif yang dihasilkan oleh model

Presisi = (TP) / (TP + FP)

3. Recall, yang juga dikenal sebagai sensitivitas, mengukur kemampuan model dalam menemukan kembali semua informasi positif yang ada. Ini adalah rasio antara jumlah prediksi positif yang benar dengan jumlah total data positif yang seharusnya dikenali oleh model

   Recall = TP / (TP + FN)

4. F1 Score adalah metrik yang menggabungkan presisi dan recall menjadi satu nilai tunggal, dengan memberikan bobot yang sama kepada keduanya. F1 Score sangat berguna ketika perlu menyeimbangkan antara presisi dan recall, terutama ketika data tidak seimbang

   F1 Score = 2 * (Recall*Presisi) / (Recall +Presisi).

3.3.3 Pengujian *Black Box*

Aplikasi deteksi penyakit pada jeruk siam berbasis Android dikembangkan dengan memanfaatkan model terlatih yang menggunakan TensorFlow Lite, yang memungkinkan aplikasi berfungsi dengan efisien pada perangkat mobile. Aplikasi ini dirancang khusus untuk mengidentifikasi berbagai jenis penyakit yang dapat menyerang tanaman jeruk siam, dengan memanfaatkan teknologi pengolahan citra. Pengguna dapat mengunggah gambar tanaman jeruk yang dicurigai terinfeksi, dan aplikasi akan menganalisis gambar tersebut untuk mendeteksi tanda-tanda penyakit. Dengan demikian, aplikasi ini tidak hanya memberikan

solusi cepat dan akurat dalam mendeteksi penyakit, tetapi juga mendukung petani dalam menjaga kesehatan tanaman dan meningkatkan hasil produksi jeruk siam.

Gambar yang digunakan untuk deteksi dapat berasal dari galeri perangkat atau diambil langsung melalui kamera perangkat. Untuk memastikan bahwa hasil deteksi yang diperoleh adalah akurat, sangat penting untuk memperhatikan kualitas kamera yang digunakan serta sudut pengambilan gambar. Gambar yang diambil dari galeri perangkat juga harus dipilih dengan hati-hati, memastikan bahwa gambar tersebut jelas dan berasal dari sumber yang dapat dipercaya.

Dalam proses pengujian aplikasi, metode blackbox digunakan untuk mengevaluasi kinerjanya. Pengujian ini melibatkan serangkaian percobaan di mana setiap kelas penyakit diuji dengan lima gambar yang berbeda, baik gambar yang diambil langsung dari kamera maupun yang diambil dari galeri perangkat. Setiap percobaan ini menghasilkan data mengenai apakah deteksi yang dilakukan benar atau salah, serta tingkat kepercayaan (confidence) dari setiap deteksi yang dilakukan oleh aplikasi. Melalui pengujian ini, aplikasi menunjukkan kemampuannya untuk mendeteksi penyakit jeruk siam dengan tingkat akurasi yang didapatkan. Hasil pengujian ini bisa menjadi acuan bahwa aplikasi memiliki potensi besar untuk membantu petani dalam mendeteksi dan menangani penyakit pada tanaman jeruk siam secara lebih efektif dan efisien.