BAB IV
HASIL DAN PEMBAHASAN

4.1 Akusisi Data

Dalam proses pembuatan website "*Citrus disease detector* " jenis data set yang digunakan yaitu kita melakukan pengumpulan data yang diperoleh dari Kaggle, sebuah platform sumber terbuka yang menyediakan berbagai *Dataset* untuk keperluan analisis dan proyek-proyek computer vision Data ini merupakan aset berharga dalam mengembangkan model *Deep learning* untuk mendeteksi penyakit pada tanaman. Pada proyek CDD ini kita menggunakan sebanyak 2400 data. Proses pengambilan data melibatkan pengunduhan *Dataset* dari sumbernya, kemudian data ini diproses, dieksplorasi, dan diolah sebelum digunakan dalam pembuatan model. Data yang berkualitas dan berjumlah besar sangat penting dalam memastikan akurasi dan kinerja model deteksi penyakit tersebut.

Proses augmentasi data ini menggunakan beberapa *function* yaitu sebagai berikut :

1. *clockwise_rotation(img):*

Fungsi ini melakukan rotasi gambar searah jarum jam. Menggunakan `cv2.cvtColor()` untuk mengubah warna gambar, meskipun parameter `0` biasanya digunakan untuk mengonversi gambar ke *grayscale*. Pada fungsi ini Gambar diubah ukurannya menjadi 224x224 piksel kemudian Gambar kemudian diputar berlawanan arah jarum jam (negatif sudut) menggunakan fungsi `rotate()`.

2. *flip_up_down(img):*

Fungsi ini membalik gambar secara vertikal (atas ke bawah) kemudaian gambar diubah ukurannya dan dikonversi warna dan Menggunakan `np.flipud()` untuk membalik gambar.

3. *add_brightness(img):*

Pada fungsi ini menambahkan efek kecerahan pada gambar, Mengonversi dan mengubah ukuran gambar sama seperti sebelumnya, Menggunakan `adjust_gamma()` untuk meningkatkan kecerahan gambar dengan gamma yang lebih kecil dari 1 untuk membuat gambar lebih terang.

4. *blur_image(img):*

Fungsi ini menambahkan efek blur (kabur) pada gambar. Setelah mengonversi dan mengubah ukuran gambar, efek *Gaussian blur* diterapkan menggunakan `cv2.GaussianBlur()` dengan kernel berukuran 9x9.

5. *sheared(img):*

Fungsi ini memberikan efek shear (geser). Pada fungsi ini Mengonversi dan mengubah ukuran gambar menjadi tahap awal, kemudian Menggunakan `AffineTransform()` untuk membuat transformasi shear dan menerapkannya dengan `warp()`.

6. *warp_shift(img):*

Fungsi ini melakukan pergeseran gambar. Setelah konversi dan resizing, fungsi ini menggunakan `AffineTransform()` untuk menerapkan translasi, menggeser gambar ke bawah sebesar 40 piksel, dan menggunakan `warp()` untuk menerapkan transformasi.

Secara keseluruhan, kode yang digunakan pada augementasi data ini mendefinisikan beberapa fungsi untuk melakukan transformasi dan efek pada gambar menggunakan OpenCV dan beberapa pustaka lainnya, sehingga dapat digunakan untuk memproses gambar dalam aplikasi tertentu.

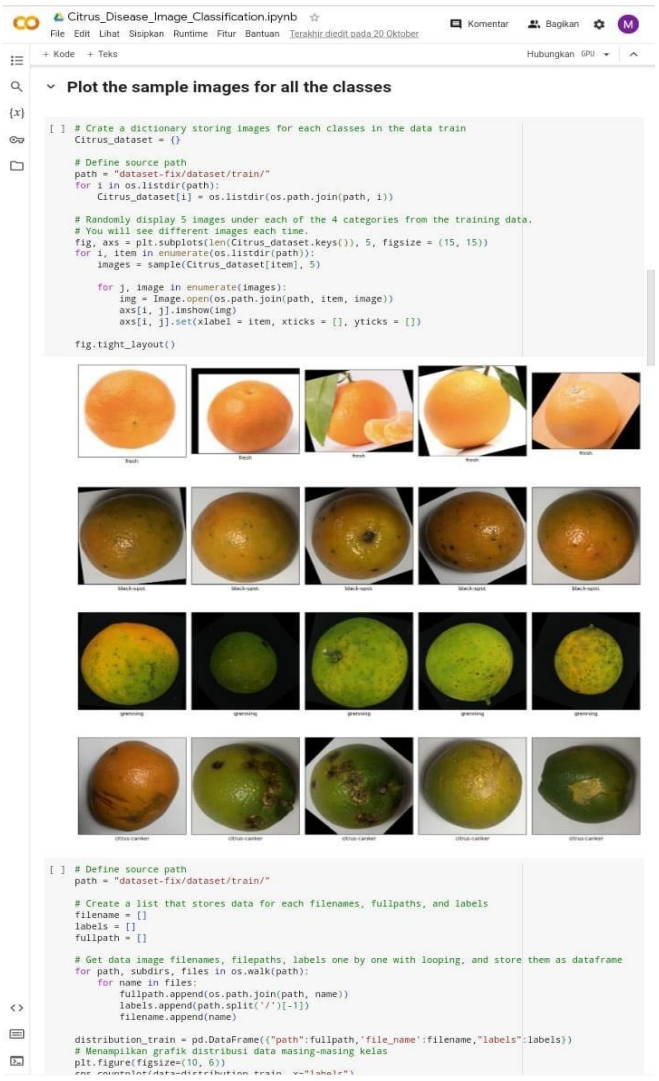
Pertama, terdapat inisialisasi dictionary transformations yang menyimpan nama- nama transformasi sebagai kunci dan fungsi terkait sebagai nilai. Selanjutnya, ditentukan lokasi gambar asli dengan images_path dan lokasi untuk menyimpan gambar yang telah diaugmentasi melalui augmented_path. List images digunakan untuk menyimpan nama-nama gambar yang akan diproses. Kemudian, kode menggunakan os.listdir untuk membaca semua nama file di direktori images_path dan menyimpannya dalam list images. Jumlah gambar yang ingin dihasilkan ditentukan sebanyak 600.

Proses augmentasi dilakukan dalam sebuah loop yang berjalan hingga mencapai jumlah gambar yang diinginkan. Dalam setiap iterasi, gambar dipilih secara acak dari list images, dibaca, dan transformasi diterapkan secara acak berdasarkan *dictionary transformations.* Setelah semua transformasi diterapkan, gambar yang telah diaugmentasi disimpan dengan nama berurutan di path yang ditentukan. Jika terjadi kesalahan saat membaca gambar, kesalahan tersebut

ditangkap dan ditampilkan tanpa menghentikan eksekusi program. Dengan cara ini, program mampu menghasilkan gambar augmentasi secara efisien sambil menangani berbagai kemungkinan kesalahan.

4.2 Eksplorasi Data

Dalam tahap eksplorasi data proyek " *Citrus disease detector*" sejumlah langkah penting telah diambil untuk memahami karakteristik *Dataset* dan mempersiapkannya untuk pengolahan lebih lanjut. Tahapan eksplorasi data melibatkan pemrosesan data hingga siap digunakan, statistik deskriptif, visualisasi data, dan pemahaman tentang nilai yang hilang. Gambar 4.1 Merupakan proses eksplorasi data pada penelitian ini.



Gambar 4.1 Proses Eksplorasi Data

Gambar 4.1 menggambarkan tahap eksplorasi data, di mana dataset diorganisir sesuai kelasnya. Gambar-gambar dalam dataset dikelompokkan dalam folder berdasarkan empat kategori utama: *black spot*, *canker*, *greening*, dan *fresh*. Langkah ini penting untuk memastikan data siap digunakan dalam pelatihan model. Setelah pengorganisasian selesai, proses dilanjutkan dengan pembagian data (*splitting data*), yaitu membagi dataset menjadi dua bagian: data latih dan data uji. Proporsi yang digunakan adalah 80% untuk data pelatihan dan 20% untuk data pengujian. Pembagian ini bertujuan untuk memberikan cukup data bagi model untuk belajar dan menguji performanya secara terpisah. Dengan struktur dan pembagian data yang baik, model dapat dilatih secara optimal untuk mendeteksi penyakit buah jeruk dengan akurasi yang lebih tinggi. Strategi ini menjadi dasar penting untuk keberhasilan penelitian.

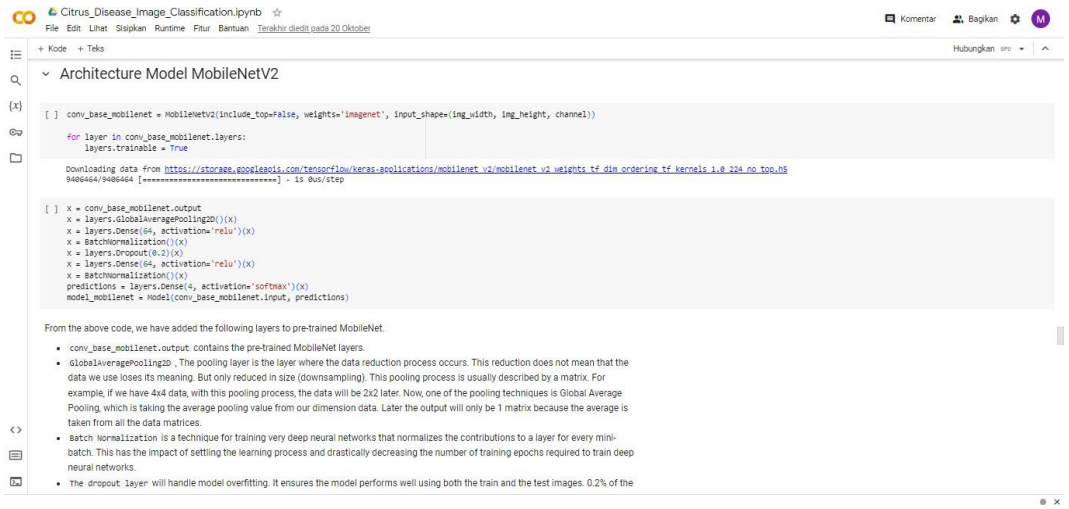
```
      # Variables used in this data separation where
variable x
      = data path and y = data labels
X_Datasetcitrus = df_Datasetcitrus['path']


      # Split the initial Dataset into training and
testing data # Use proportion 80% for training data and
20% for testing data
      X_train, X_test, y_train, y_test = train_test_split(
          X_Datasetc           y_datsetc           test_siz
    itrus,
```

Kode ini membagi dataset citrus menjadi data pelatihan dan pengujian dengan proporsi 80:20, di mana 80% data digunakan untuk melatih model (*X_train* dan *y_train*) dan 20% untuk menguji performa model (*X_test* dan *y_test*). Parameter *random_state=42* memastikan hasil pembagian data konsisten setiap kali kode dijalankan. Pembagian ini memungkinkan model memperoleh data pelatihan yang cukup dan data pengujian yang representatif untuk evaluasi performa. Proses ini penting untuk menjaga validitas hasil prediksi dan meminimalkan potensi *overfitting* selama pelatihan model.

4.3 Modeling

Dalam pembuatan proyek " *Citrus disease detector* " model yang kita gunakan yaitu model *mobileNetV2* . Arsitektur *MobileNetV2* didasarkan pada struktur sisa terbalik dimana masukan dan keluaran dari blok sisa merupakan lapisan hambatan tipis yang berlawanan dengan model sisa tradisional yang menggunakan representasi pada masukan. *MobileNetV2* menggunakan konvolusi mendalam yang ringan untuk memfilter fitur di lapisan ekspansi menengah. Model *mobileNetV2* ini memiliki nilai akurasi yang tinggi. Hal ini dapat dilakukan dengan cara menginput model *mobilenet*v2.predict. Gambar 4.2 merupakan aristektur *MobileNetV2* yang digunakan pada penelitian ini.



Gambar 4.2 Arsitektur MobilenetV2

Gambar 4.2 merupakan arsitektur MobileNetv2 yang digunakan pada penelitian ini. *MobileNetV2* adalah arsitektur jaringan saraf konvolusional yang dirancang untuk efisiensi dan kecepatan dalam pengolahan gambar. Arsitektur ini menggunakan *depthwise separable convolutions*, yang terdiri dari konvolusi kedalaman dan *konvolusi pointwise*, untuk mengurangi jumlah parameter dan komputasi. *mobilenetV2* juga mengintegrasikan blok residual terbalik yang menggabungkan konvolusi biasa dan *pointwise*, menjaga informasi sambil meningkatkan efisiensi. Selain itu, lapisan *bottleneck* dengan aktivasi linier di lapisan terakhir setiap blok membantu mempertahankan representasi data.

Arsitektur ini menggunakan *ReLU6* sebagai fungsi aktivasi, yang membatasi output untuk menjaga stabilitas dan efisiensi. Dengan penerapan *layer*

normalization, MobilenetV2 meningkatkan konvergensi saat pelatihan. Arsitektur ini sangat efisien dan cocok untuk aplikasi di perangkat dengan sumber daya terbatas, seperti perangkat *mobile* dan *IoT*. Gambar 4.3 merupakan hasil parameter *MobileNetV2*.

```
=================================================================================================
Total params: 2,344,900
Trainable params: 2,310,532
Non-trainable params: 34,368
_________________________________________________________________________________________________
```

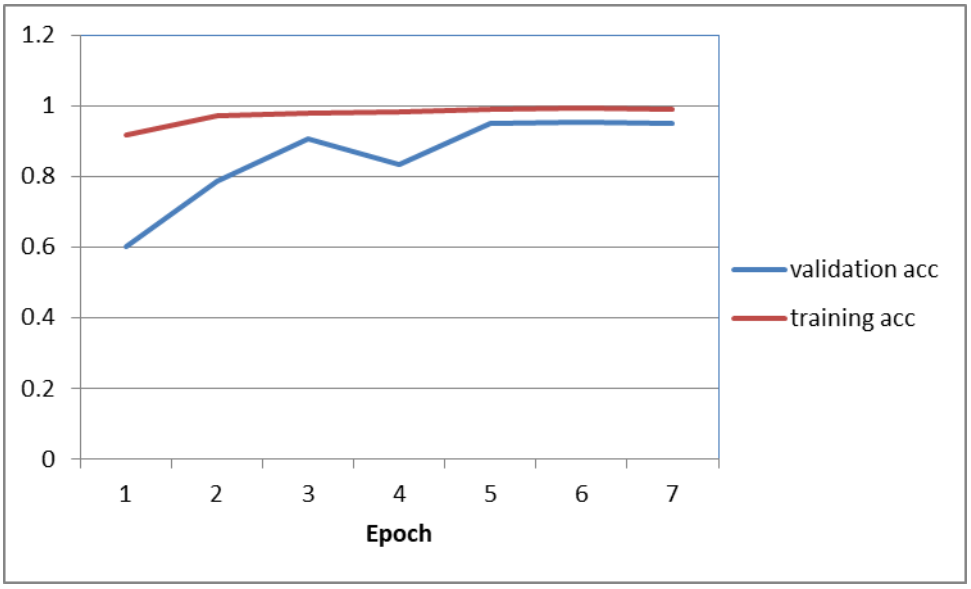
Gambar 4.3 Hasil Parameter MobileNetV2

Gambar 4.3 merupakan hasil parameter *MobileNet* V2. Hasil tersebut menjeaskan dengan menerapkan arsitektur *MobileNet* V2, jumlah total parameter yang dihasilkan adalah 2,344,900, di mana 2,310,532 di antaranya adalah parameter yang dapat dilatih, dan 34,368 adalah parameter yang tidak dapat dilatih.

```
CPU times: user 3 µs, sys: 1 µs, total: 4 µs
Wall time: 6.91 µs
Epoch 1/10
200/200 [==============================] - 63s 207ms/step - loss: 0.2611 - accuracy: 0.9186 - val_loss: 1.0819 - val_accuracy: 0.6005
Epoch 2/10
200/200 [==============================] - 32s 159ms/step - loss: 0.0972 - accuracy: 0.9706 - val_loss: 0.5587 - val_accuracy: 0.7852
Epoch 3/10
200/200 [==============================] - 30s 148ms/step - loss: 0.0766 - accuracy: 0.9775 - val_loss: 0.2256 - val_accuracy: 0.9058
Epoch 4/10
200/200 [==============================] - 31s 156ms/step - loss: 0.0599 - accuracy: 0.9818 - val_loss: 0.5075 - val_accuracy: 0.8329
Epoch 5/10
200/200 [==============================] - 31s 155ms/step - loss: 0.0473 - accuracy: 0.9887 - val_loss: 0.1286 - val_accuracy: 0.9510
Epoch 6/10
200/200 [==============================] - 32s 160ms/step - loss: 0.0294 - accuracy: 0.9919 - val_loss: 0.1463 - val_accuracy: 0.9548
Epoch 7/10
200/200 [==============================] - 33s 165ms/step - loss: 0.0288 - accuracy: 0.9903 - val_loss: 0.1503 - val_accuracy: 0.9485
```

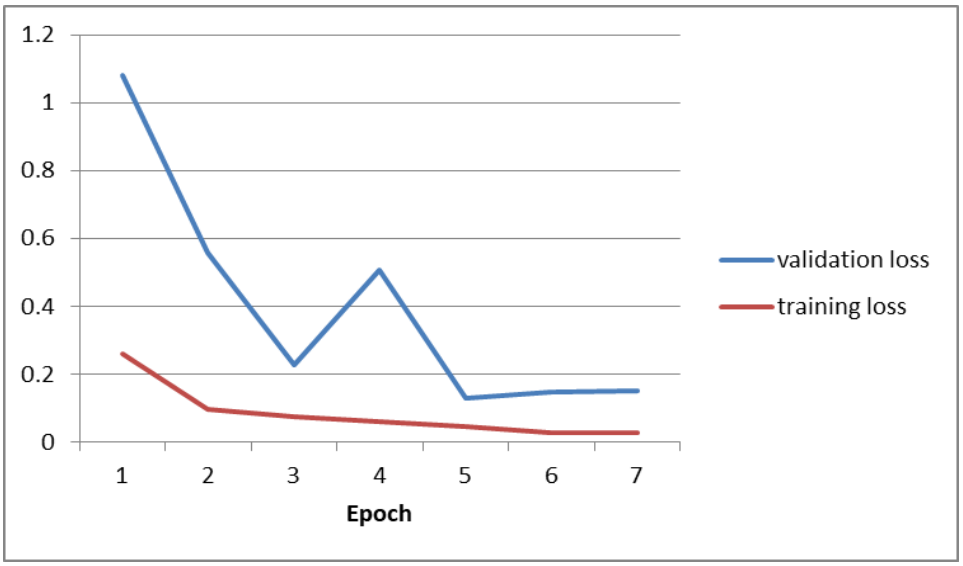
Gambar 4. 4 Hasil *Training Model* MobileNetV2

Gambar 4.4 diatas menjelaskan hasil evaluasi menunjukkan bahwa pada *epoch* pertama hingga *epoch* ketujuh perolehan akurasi mengalami peningkatan, baik itu *accuracy* maupun di val_*accuracy*. Dimana pada epoch ketujuh nilai perolehan *accuracy* sebesar 0.9903 dan nilai val_*accuracy* sebesar 0.9485. Sedangkan untuk nilai loss dari epoch pertama hingga epoch ketujuh perolehan nilai mengalami penurunan, baik itu di loss maupun di val_loss. Dimana pada epoch ketujuh nilai perolehan loss sebesar 0.0288 dan nilai val_loss sebesar 0.1503. Oleh sebab itu, model yang dirancang sudah dapat dikatakan model yang memiliki performa baik dalam melakukan kalsifikasi gambar dan mengidentikasi gambar sehingga sudah dapat digunakan. Pada Gambar 4.5 merupakan grafik hasil *training accuracy* dan *validation accuracy.*



Gambar 4.5 Grafik Hasil *Training Accuracy dan Validation Accuracy*

Gambar 4.5 Sumbu X (*horizontal*) menunjukkan jumlah *epoch* (siklus pelatihan penuh), Sumbu Y (*vertical*) menunjukkan nilai akurasi, garis untuk *training accuracy* menunjukkan perubahan akurasi pada data pelatihan selama pelatihan dan garis untuk *validation accuracy* menunjukkan perubahan akurasi data validasi selama pelatihan. Pada Gambar 4.6 merupakan grafik hasil dari *training loss* dan *validation loss.*

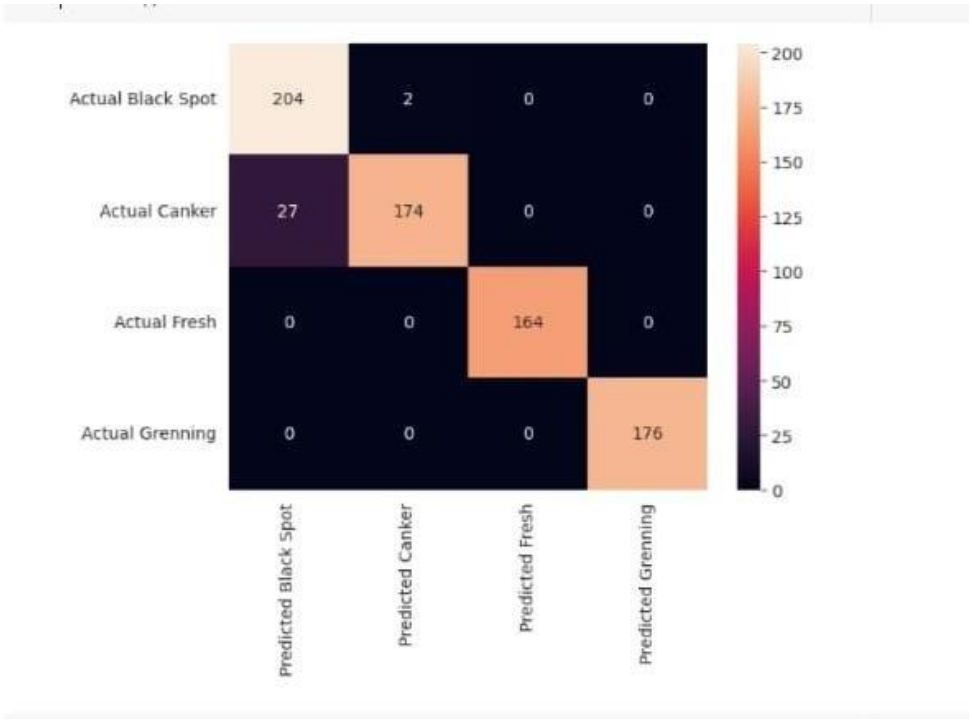


Gambar 4.6 Grafik Hasil *Training Loss* dan *Validation Loss* MobileNetV2

Gambar 4.6, sumbu X (horizontal) mewakili jumlah *epoch* atau siklus pelatihan penuh, sedangkan sumbu Y (vertikal) menunjukkan nilai akurasi. Garis akurasi pelatihan menggambarkan perubahan akurasi pada data pelatihan selama proses pelatihan, sementara garis akurasi validasi menunjukkan perubahan akurasi pada data validasi. Grafik ini digunakan untuk memvisualisasikan kinerja model dan membantu mengidentifikasi pola seperti *overfitting* atau *underfitting*. Jika akurasi validasi stagnan atau menurun meskipun akurasi pelatihan terus meningkat, hal itu mengindikasikan potensi *overfitting* pada model.

4.4 Evaluasi

Dalam evaluasi algoritma dan model yang digunakan untuk mendeteksi penyakit pada aplikasi website " *Citrus disease detector* " beberapa metrik penting digunakan untuk mengukur kinerja masing-masing algoritma. Beberapa metrik yang digunakan Pada perancangan aplikasi berbasis web deteksi penyakit buah jeruk yaitu *metrics precision, recall*, F1 *score,* dan *accuracy* serta menggunakan *confusion matrix* pada model tersebut. Evaluasi pada algoritma yang digunakan pada proyek memperoleh akurasi sebesar 0.9612. Akurasi merupakan metrik yang mengukur sejauh mana model mampu mengklasifikasikan dengan benar, yaitu jumlah prediksi yang benar dibagi dengan total prediksi. Akurasi yang tinggi menunjukkan bahwa algoritma ini memiliki kemampuan yang sangat baik . Dengan diperolehnya akurasi sebesar 0.9612 pada penelitian ini menandakan bahwa algoritma ini mampu memberikan prediksi yang akurat dalam mendeteksi penyakit pada tanaman jeruk. Pada Gambar 4.7 merupakan *confussion matrix* dari penelitian ini.



Gambar 4.7 *Confussion Matrix Model* Mobilenet V2

Gambar 4.7, ditampilkan hasil *confusion matrix* MobileNetV2 pada aplikasi website deteksi penyakit buah jeruk. *Confusion matrix* ini menunjukkan jumlah klasifikasi yang benar dan salah pada masing-masing kelas penyakit jeruk. Dengan menggunakan *confusion matrix*, dapat dianalisis berapa banyak kasus yang diklasifikasikan dengan tepat dan berapa banyak yang salah, serta identifikasi kelas mana yang memiliki tingkat kesalahan tertinggi. *Matrix* ini memberikan gambaran yang lebih jelas mengenai kinerja model dalam mendeteksi penyakit buah jeruk berdasarkan data yang diberikan, yang membantu dalam evaluasi dan peningkatan akurasi model deteksi penyakit.

1. Berikut *Black spot*

Tabel 4.1 merupakan hasil *confusion matrix* pada kelas *black spot.*

Tabel 4.1 *Confussion Matrix Black Spot*

	Positif	Negatif
Positif	204	27
Negatif	2	514

Tabel 4.1 menampilkan hasil *confusion matrix* untuk klasifikasi penyakit *black spot* pada buah jeruk. Dalam *matriks* tersebut, terdapat 204 *True Positif* (TP), yang berarti 204 gambar penyakit black spot terdeteksi dengan benar. *True Negatif* (TN) sebanyak 514 menunjukkan jumlah gambar yang tidak terdeteksi sebagai *black spot* dan memang bukan *black spot*. Sementara itu, terdapat 27 *False Positif* (FP), di mana gambar yang tidak mengandung penyakit malah diklasifikasikan sebagai *black spot*. Terakhir, 2 *False Negatif* (FN) berarti ada 2 gambar yang sebenarnya mengandung penyakit namun tidak terdeteksi.

2. *Citrus canker*

Berikut Tabel 4.2 merupakan *confussion matrix* pada kelas *citrus canker*.Tabel 4.2 Confussion Matrix Citrus Canker

	Positif	Negatif
Positif	174	2
Negatif	27	514

Tabel 4.2, ditampilkan hasil *confusion matrix* untuk klasifikasi penyakit citrus canker pada buah jeruk. Dalam matriks tersebut, terdapat 174 *True Positif* (TP), yang berarti 174 gambar penyakit citrus canker terdeteksi dengan benar. *True Negatif* (TN) sebanyak 514 menunjukkan jumlah gambar yang tidak terdeteksi sebagai citrus canker dan memang bukan citrus canker. Sementara itu, terdapat 2 *False Positif* (FP), di mana gambar yang tidak mengandung penyakit malah diklasifikasikan sebagai *citrus canker*. Terakhir, 27 *False Negatif* (FN) berarti ada 27 gambar yang sebenarnya mengandung penyakit namun tidak terdeteksi.

3. *Fresh*

Berikut Tabel 4.3 merupakan *confussion matrix* pada kelas *fresh.*

Tabel 4.3 *Confussion Matrix Fresh*

	Positif	Negatif
Positif	164	0
Negatif	0	583

Tabel 4.3, ditampilkan hasil confusion matrix untuk klasifikasi penyakit "fresh" pada buah jeruk. Dalam matriks tersebut, terdapat 164 True Positif (TP),

yang berarti 164 gambar yang benar-benar menunjukkan buah jeruk yang segar terdeteksi dengan benar. True Negatif (TN) sebanyak 583 menunjukkan gambar

yang tidak mengandung penyakit dan memang terklasifikasi sebagai buah jeruk segar. Tidak ditemukan False Positif (FP) maupun False Negatif (FN), yang menunjukkan bahwa model berhasil mendeteksi semua gambar segar dengan akurat tanpa kesalahan klasifikasi.

4. *Greening*

Berikut Tabel 4.4 merupakan merupakan *confussion matrix* pada kelas *greening.*

Tabel 4.4 *Confussion Matrix Greening*

	Positif	Negatif
Positif	176	0
Negatif	0	571

Tabel 4.4, disajikan hasil confusion matrix untuk klasifikasi penyakit "greening" pada buah jeruk. Matriks tersebut menunjukkan 176 True Positif (TP), yang berarti 176 gambar buah jeruk yang terinfeksi penyakit greening terdeteksi dengan benar. True Negatif (TN) sebanyak 571 menunjukkan gambar yang tidak mengandung penyakit greening dan terklasifikasi dengan tepat. Tidak ada False Positif (FP) atau False Negatif (FN), yang menandakan bahwa model berhasil mendeteksi semua gambar dengan akurasi tinggi tanpa kesalahan dalam klasifikasi.

Berdasarkan hasil yang diperoleh, beberapa jenis penyakit pada buah jeruk belum terklasifikasi dengan baik. Hal ini disebabkan oleh perbedaan karakteristik visual pada setiap penyakit yang menyebabkan kesulitan dalam membedakan antara satu penyakit dengan yang lainnya. Setiap jenis penyakit memiliki gejala atau tanda khas yang berbeda, yang membuat model kesulitan untuk membedakan dan mengklasifikasikan dengan akurasi tinggi. Misalnya, perbedaan warna, tekstur, dan pola pada kulit buah jeruk dapat mempengaruhi kemampuan model dalam mendeteksi penyakit secara tepat.

Hasil dari confusion matrix menunjukkan adanya kesalahan dalam klasifikasi, yang terdeteksi pada nilai False Positive (FP) atau False Negative (FN). Penyebab kesalahan ini sering kali berkaitan dengan kekurangan data yang

representatif atau variasi dalam kondisi buah jeruk yang diuji. Meskipun model menunjukkan kinerja yang baik secara keseluruhan, adanya kesalahan klasifikasi ini menunjukkan perlunya peningkatan dalam jumlah data pelatihan dan teknik augmentasi data untuk meningkatkan akurasi deteksi.

Sebagai langkah selanjutnya, penyesuaian pada model, seperti tuning parameter dan eksperimen dengan arsitektur yang lebih kompleks atau data yang lebih bervariasi, mungkin diperlukan untuk mencapai klasifikasi yang lebih akurat.

$$Accuracy = \frac{total\ TP}{total\ semua\ data}$$

$$Accuracy = \frac{204+174+164+176}{747}$$

$$Accuracy = 0{,}9612$$

Perhitungan ini menghasilkan nilai akurasi sebesar 0,9612. Selain adanya perhitungan akurasi, perhitungan- perhitungan lainnya dilakukan pada *confussion matrix* seperti *precision, recall dan F1-score*. Berikut merupakan perhitungan-perhitungan matrik pada deteksi penyakit buah jeruk sesuai kelasnya :

1. Perhitungan *black spot*

a) $Precision = \frac{TP}{TP+FP}$

$$precision = \frac{204}{204+27}$$

$$precision = 0{,}8831$$

b) $Recall = \frac{TP}{TP+FN}$

$$Recall = \frac{204}{204+2}$$

$$Recall = 0{,}9903$$

c) $F1\text{-}score = 2X\frac{Precision\,.\,Recall}{Precission+Recall}$

$$F1\text{-}score = 2X\frac{0{,}8831\,.\,0{,}9903}{0{,}8831+0{,}9903}$$

$$F1\text{-}score = 0{,}9336$$

Berdasarkan perhitungan diatas pada kelas *black spot* didapatkan nilai *precision* sebesar 0,8831, *recall* sebesar 0,9903 dan *F1-score* sebesar 0,9336.

2. Perhitungan *citrus canker*

a) $Precision = \frac{TP}{TP+FP}$

$precision = \frac{174}{174+2}$

$precision = 0{,}9986$

b) $Recall = \frac{TP}{TP+FN}$

$Recall = \frac{174}{174+27}$

$Recall = 0{,}8657$

c) $F1\text{-}score = 2X\frac{Precision\,.\,Recall}{Precission+Recall}$

$F1\text{-}score = 2X\frac{0{,}9986\,.\,0{,}8657}{0{,}9986+0{,}8657}$

$F1\text{-}score = 0{,}9231$

Berdasarkan perhitungan diatas pada kelas *citrus canker* didapatkan nilai *precision* sebesar 0,9986, *recall* sebesar 0,8657 dan *F1-score* sebesar 0,9231.

3. Perhitungan *fresh*

a) $Precision = \frac{TP}{TP+FP}$

$precision = \frac{164}{164+0}$

$precision = 1$

b) $Recall = \frac{TP}{TP+FN}$

$Recall = \frac{164}{164+0}$

$Recall = 1$

c) $F1\text{-}score = 2X\frac{Precision\,.\,Recall}{Precission+Recall}$

$F1\text{-}score = 2X\frac{1\,.\,1}{1+1}$

$F1\text{-}score = 1$

Berdasarkan perhitungan diatas pada kelas *fresh* didapatkan nilai *precision* sebesar 1, *recall* sebesar 1 dan *F1-score* sebesar 1.

4. Perhitungan *greening*

a) $Precision = \frac{TP}{TP+ FP}$

$precision = \frac{176}{176+0}$

$precision = 1$

b) $Recall = \frac{TP}{TP+ FN}$

$Recall = \frac{176}{176+0}$

$Recall = 1$

c) $F1\text{-}score = 2X\frac{Precision\ .\ Recall}{Precission+Recall}$

$F1\text{-}score = 2X\frac{1\ .\ 1}{1+1}$

$F1\text{-}score = 1$

Berdasarkan perhitungan diatas pada kelas *greening* didapatkan nilai *precision* sebesar 1, *recall* sebesar 1 dan *F1-score* sebesar 1. Pada Gambar 4.8 merupakan hasil pemodelan *confussion matrix* CNN.

	precision	recall	f1-score	support
Black Spot	0.8831	0.9903	0.9336	206
Canker	0.9886	0.8657	0.9231	201
Fresh	1.0000	1.0000	1.0000	164
Grenning	1.0000	1.0000	1.0000	176
accuracy			0.9612	747
macro avg	0.9679	0.9640	0.9642	747
weighted avg	0.9647	0.9612	0.9610	747

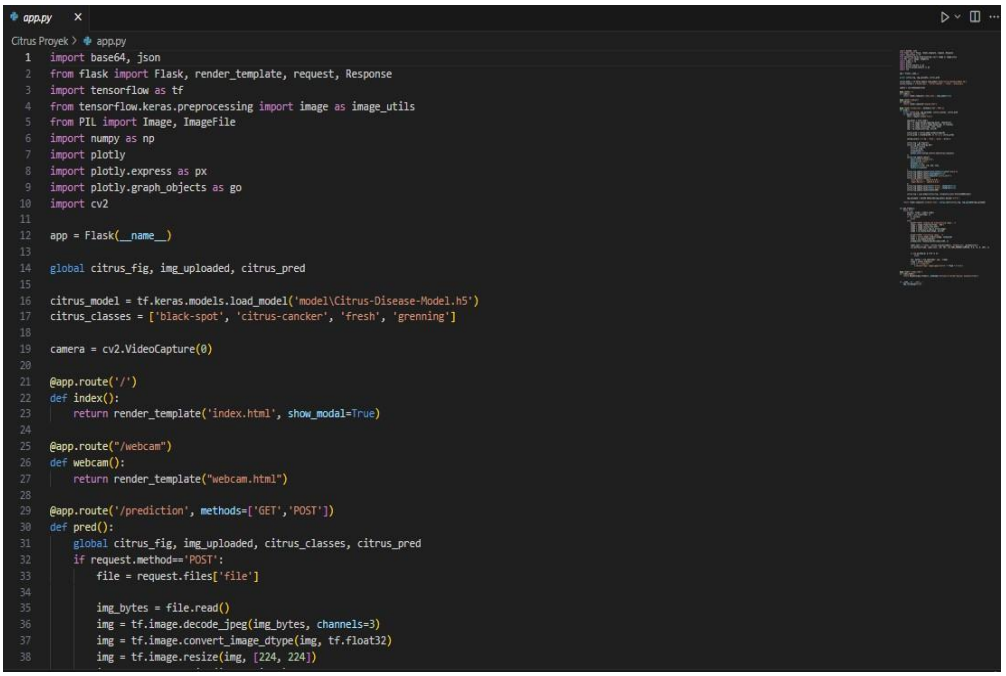
Gambar 4.8 Hasil *Model Confusion Matrix* CNN

Gambar 4.8 merupakan hasil dari model confussion matrix CNN yang dibuat oleh program dimana hasil tersebut sesuai dengan hasil perhitungan manual dengan menggunakan rumus.

4.5 *Deployment*

Tujuan dari *Deployment* pada aplikasi website " *Citrus disease detector* " adalah memungkinkan pengguna, dalam hal ini, pengusaha argibisnis jeruk, untuk

mengakses dan menggunakan alat deteksi pennyakit dengan mudah dan efisien. Proses *Deployment* aplikasi web ini dilakukan pada *local host*, sehingga untuk menjalankan aplikasi perlu menggunakan bantuan *command prompt*. Semua file yang diperlukan dimasukkan dalam satu folder. Pada *command prompt*, lokasi folder dibuka menggunakan command "cd". Lalu, program *flask* app.py dibuka dengan *command "python*". *Local host* akan berjalan dan menghasilkan alamat url untuk dibuka pada web browser. Web browser dan program akan berjalan, lalu data input dapat dimasukkan. Algoritma akan berjalan dan menghasilkan prediksi berdasarkan data input. Pada Gambar 4.9 merupakan proses *deployment* pada penelitian ini.



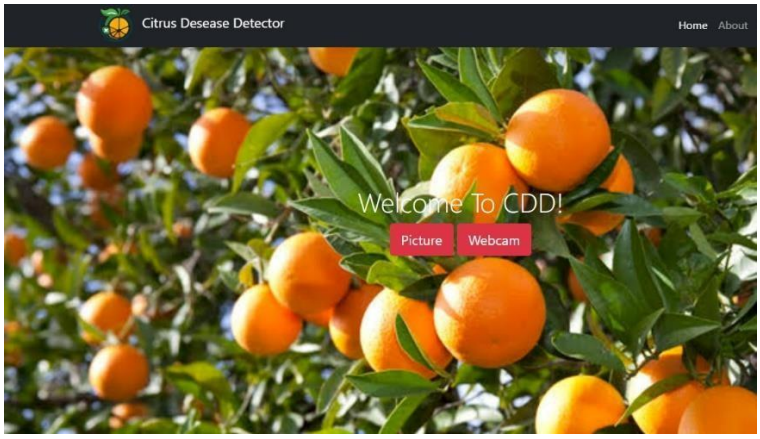
Gambar 4.9 Proses *Deployment*

Gambar 4.9 merupakan poses *deployment* pada aplikasi web deteksi penyakit buah jeruk mengugunakan metode CNN dengan arsitektur *MobileNetV2*. Proses *deployment* merupakan serangkaian langkah untuk merilis aplikasi perangkat lunak ke lingkungan produksi. Proses ini dimulai dengan persiapan kode yang telah selesai dan diuji, dilanjutkan dengan membangun kode menjadi versi yang dapat dijalankan. Setelah itu, dilakukan pengujian tambahan di lingkungan staging untuk memastikan semua fitur berfungsi dengan baik. Kemudian, aplikasi diunggah ke server produksi, dan verifikasi dilakukan untuk memastikan semuanya berjalan

lancar. Setelah *deployment*, pemantauan dilakukan untuk mendeteksi masalah yang mungkin muncul, dan jika diperlukan, langkah *rollback* dapat diambil untuk mengembalikan aplikasi ke versi sebelumnya.

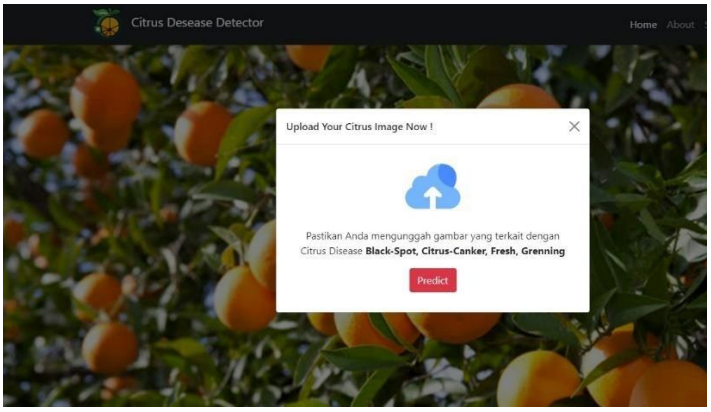
4.6 Tampilan

Antarmuka pengguna adalah tampilan grafis yang berinteraksi langsung dengan pengguna. Gambar 4.10 menunjukkan tampilan dashboard dari aplikasi web *Citrus Disease Detector*. Pada *dashboard* ini, pengguna dapat mengakses berbagai fitur untuk mendeteksi penyakit pada buah jeruk, baik melalui unggahan gambar maupun menggunakan kamera secara langsung. Antarmuka ini dirancang untuk memudahkan pengguna dalam melakukan deteksi penyakit secara efektif dan cepat, memberikan pengalaman pengguna yang interaktif dan mudah dioperasikan untuk memantau kondisi buah jeruk.



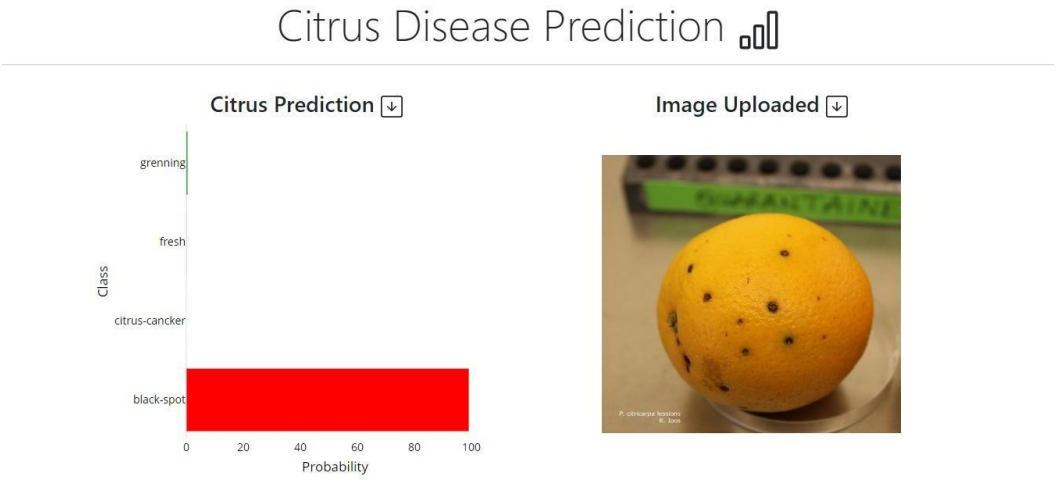
Gambar 4.10 Tampilan *Dashboard*

Gambar 4.13 merupakan tampilan fitur *webcam* yang memungkinkan deteksi penyakit buah jeruk secara real-time. Pada gambar tersebut, hasil deteksi menunjukkan bahwa buah jeruk yang diunggah dalam kondisi segar atau tidak terinfeksi penyakit. Fitur ini memberikan kemudahan untuk melakukan deteksi langsung tanpa perlu mengunggah gambar terlebih dahulu. Selanjutnya, pada Gambar 4.14 merupakan tampilan yang memberikan informasi lebih lanjut mengenai penyakit yang terdeteksi pada buah jeruk melalui aplikasi *Citrus Disease Detector*, memungkinkan pengguna untuk memahami lebih detail kondisi buah yang diidentifikasi.



Gambar 4.11 *Image Detection*

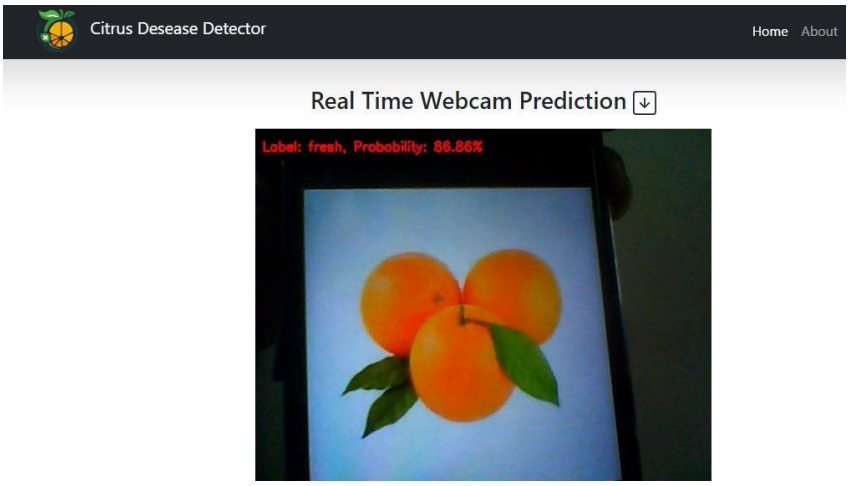
Gambar 4.11 ditampilkan halaman *image detection* pada aplikasi web, di mana pengguna dapat mengunggah gambar penyakit tanaman jeruk yang ingin dideteksi. Halaman ini memudahkan pengguna untuk memasukkan gambar sebagai input deteksi. Gambar 4.12 merupakan hasil deteksi dari gambar yang telah diunggah. Sistem akan memproses gambar dan menampilkan hasil berupa jenis penyakit yang terdeteksi, seperti *black spot*, *canker*, *greening*, atau *fresh*. Tampilan ini mempermudah pengguna untuk memahami hasil analisis secara cepat dan intuitif.



Gambar 4.12 Hasil Deteksi

Gambar 4.12 merupakan tampilan hasil deteksi penyakit pada buah jeruk menggunakan aplikasi *Citrus Disease Detector*. Hasil menunjukkan bahwa buah jeruk tersebut terdeteksi terkena penyakit *black spot*. Proses deteksi dilakukan dengan mengunggah gambar melalui fitur yang tersedia di aplikasi. Gambar 3.13 merupakan tampilan fitur *webcam*, yang memungkinkan pengguna mengambil

gambar langsung dari kamera untuk dideteksi secara real-time. Fitur ini menambah kemudahan dan fleksibilitas dalam proses identifikasi penyakit pada buah jeruk.



Gambar 4.13 Tampilan Fitur *Webcam*

Gambar 4.13 merupakan tampilan fitur *webcam* yang memungkinkan deteksi penyakit buah jeruk secara real-time. Pada gambar tersebut, hasil deteksi menunjukkan bahwa buah jeruk yang diunggah dalam kondisi segar atau tidak terinfeksi penyakit. Fitur ini memberikan kemudahan untuk melakukan deteksi langsung tanpa perlu mengunggah gambar terlebih dahulu. Gambar 4.14 merupakan tampilan yang memberikan informasi lebih lanjut mengenai penyakit yang terdeteksi pada buah jeruk melalui aplikasi *Citrus Disease Detector*, memungkinkan pengguna untuk memahami lebih detail kondisi buah yang diidentifikasi.

Gambar 4.14 Tampilan Informasi Citrus

Gambar 4.14, ditampilkan halaman "*About*" pada aplikasi web *Citrus Disease Detector*. Halaman ini menyediakan informasi tentang berbagai penyakit yang dapat menyerang buah jeruk. Pengguna dapat memperoleh pengetahuan lebih mendalam mengenai setiap jenis penyakit yang terdeteksi oleh aplikasi ini. Dengan informasi yang lengkap, pengguna dapat memahami gejala, penyebab, dan cara pencegahan penyakit-penyakit tersebut, sehingga mereka dapat merawat buah jeruk dengan lebih baik. Fitur ini membantu meningkatkan kesadaran dan memberikan edukasi yang berguna bagi pengguna dalam mengidentifikasi dan menangani penyakit tanaman jeruk.