

BAB 4

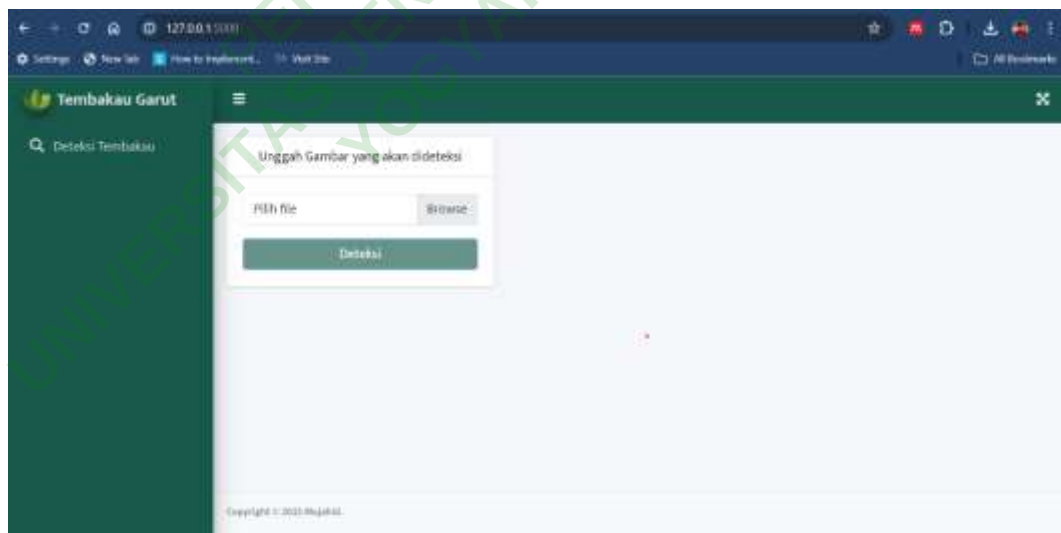
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Hasil penelitian mencakup penjelasan terperinci mengenai hasil yang didapatkan dari penelitian implementasi CNN untuk mendeteksi penyakit tanaman tembakau. Penelitian ini membutuhkan dataset gambar penyakit tembakau dan data gejala serta cara penanganannya, selanjutnya dilakukan pra-pemrosesan data untuk melakukan *splitting*, *resizing* dan augmentasi data. Data tersebut akan digunakan untuk melatih model yang berfungsi mendeteksi penyakit pada tanaman tembakau. Selanjutnya, sistem sederhana yang memuat model dan basis data akan dibuat untuk mempermudah proses deteksi penyakit tembakau.

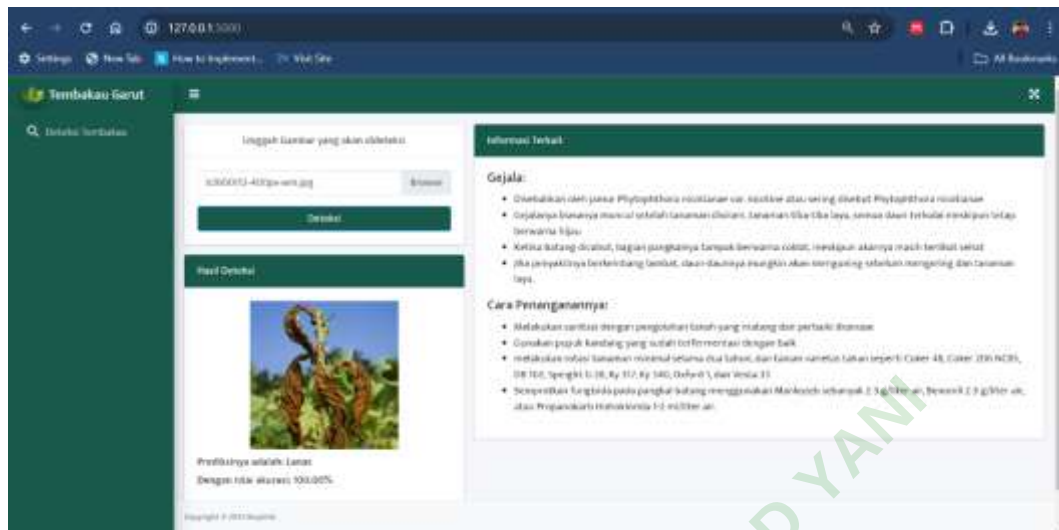
4.2 IMPLEMENTASI DESAIN ANTARMUKA

Tampilan user interface halaman utama sistem deteksi penyakit tembakau ditunjukkan pada Gambar 4.1.



Gambar 4.1 Halaman beranda

Tampilan *user interface* halaman deteksi sistem deteksi penyakit tembakau ditunjukkan pada Gambar 4.2.



Gambar 4.2 Halaman deteksi

4.3 BASIS DATA

Basis data yang digunakan adalah Structured Query Language (SQL). MySQL ini digunakan untuk menyimpan data penyakit terkait dengan gejala dan cara penanganannya seperti yang terlihat pada Gambar 4.3. Data tersebut akan ditampilkan berbarengan dengan output yang dihasilkan ketika deteksi penyakit.

Tabel	Tindakan	Baris
☐ penyakit	<input type="star"/> Jelajahi <input type="book"/> Struktur <input type="magnifying-glass"/> Cari <input type="plus"/> Tambahkan <input type="printer"/> Kosongkan <input type="trash"/> Hapus	5
1 tabel	Jumlah	5

Gambar 4.3 Database penyakit

4.4 FITUR-FITUR SISTEM

Sistem ini memiliki dua *route*, yaitu halaman beranda dan halaman /deteksi. Halaman beranda atau tampilan awal sistem diilustrasikan pada Gambar 4.1 di atas. Sedangkan halaman /deteksi ditunjukkan pada Gambar 4.2 di atas. Untuk mendapatkan hasil deteksi, pengguna perlu memasukkan *input* berupa gambar tembakau.

4.5 PEMBAHASAN

4.5.1 Pengumpulan Data

Sesuai pada penelitian sebelumnya bahwa pengumpulan data dilakukan secara primer atau langsung dari kebun tembakau di daerah kabupaten Garut dengan data gambar yang dikumpulkan sebanyak 600 gambar. Berikut merupakan contoh data gambar yang dikumpulkan seperti yang terlihat pada Gambar 4.4.



Gambar 4.4 Dataset tembakau

Sedangkan pengumpulan data mengenai gejala dan penanganannya dilakukan melalui pendekatan skunder dan merujuk pada penelitian terdahulu. Berikut merupakan datanya yang diuraikan pada Tabel 4.1.

Tabel 4.1 Data gejala penyakit tembakau dan cara penanganannya

No	Nama Penyakit	Gejala	Cara Penanganan
1	Patik	- Disebabkan oleh jamur <i>Cercospora nicotianae</i> Ell. & Ev. - Tanda pertama muncul pada daun berupa sedikit warna coklat muda yang	Lakukan sanitasi dengan buang daun bagian bawah yang terinfeksi dan musnahkan sisa tanaman atau bibit yang tidak lagi dipakai

		kemudian berubah menjadi putih dan kering, sering disertai bintik ikan di tengahnya. • Terjadi di daerah dengan suhu 27-30 derajat dan kelembapan tinggi, sehingga menyebabkan bercak membesar dan nekrosis daun yang akhirnya menjadi berlubang.	untuk mencegah penyebaran penyakit. • Pastikan menggunakan bibit yang bebas penyakit dan sehat. • Jika infeksi sudah parah, segera petik daun yang terinfeksi lebih awal. • Gunakan fungisida sistemik seperti karbendazim (Derosal), tofanat metil (Topsin), dan benomyl (Benlate), serta selingi dengan fungisida kontak seperti mankozeb (Dithane M 45) atau propineb (Antracol) untuk hasil yang lebih efektif.
2	Layu Bakteri	• Disebabkan oleh bakteri <i>Ralstonia solanacearum</i> E.F. Smith. • Tanda khas penyakit ini adalah layu di satu sisi tanaman tembakau. • Awal gejala adalah satu atau dua daun muda yang layu kembali segar pada malam atau pagi hari. • Gejala layu dapat berkembang dengan cepat pada kondisi lingkungan yang lembab dan panas. • Dalam kasus kondisi kering dan panas, gejala layu akan berkembang lambat dan daun akan	• Melakukan rotasi tanaman seperti yang biasanya dilakukan di Jawa, seperti menggantikan dengan menanam padi sawah selama dua musim berturut-turut, dapat membantu menghilangkan patogen yang ada di tanah. • Sanitasi lahan perlu dilakukan dengan membersihkan area dari gulma, mencabut tanaman yang terinfeksi, mengeluarkannya dari area tanam, dan membakarnya setelah kering. • Sterilisasi media

		berubah menjadi pucat hijau sebelum akhirnya menjadi kekuningan dengan daun biasanya tetap melekat pada batang tanaman dan tidak tumbuh dengan baik	bibitan bisa dilakukan dengan cara menjemur di bawah sinar matahari atau menggunakan uap air panas. • Pengendalian hayati dapat dilakukan dengan mengisolasi bakteri antagonis dari akar tanaman mimosa yang kemudian digunakan sebagai biopestisida serta penyiraman bibit yang akan ditanam dengan suspensi bakteri ini dapat membantu menekan serangan penyakit
3	Lanas	• Disebabkan oleh jamur <i>Phytophthora nicotianae</i> var. <i>nicotine</i> atau sering disebut <i>Phytophthora nicotianae</i>. • Gejalanya biasanya muncul setelah tanaman disiram, tanaman tiba-tiba layu, semua daun terkulai meskipun tetap berwarna hijau. • Ketika batang dicabut, bagian pangkalnya tampak berwarna coklat, meskipun akarnya masih terlihat sehat. • Jika penyakitnya berkembang lambat, daun-daunnya mungkin akan menguning sebelum mengering dan	• Melakukan sanitasi dengan mengolah tanah secara optimal dan memperbaiki sistem drainase. • Menggunakan pupuk kandang yang telah difermentasi dengan baik. • Melaksanakan rotasi tanaman setidaknya selama dua tahun, serta menanam varietas tahan seperti Coker 48, Coker 206 NC85, DB 102, Speight G-28, Ky 317, Ky 340, Oxford 1, dan Vesta 33. • Menyemprotkan fungisida pada pangkal batang bisa dilakukan dengan menggunakan Mankozeb sebanyak 2-3 gram per liter air,

		tanaman layu.	Benomil 2-3 gram per liter air, atau Propanokarb Hidroklorida 1-2 mililiter per liter air.
4	Busuk Batang	• Disebabkan oleh bakteri <i>Erwinia carotovora</i> var. <i>carotovora</i>. • Biasanya, setelah proses pangkasan atau penghilangan tunas samping pada tanaman tembakau gejala penyakit ini muncul. • Gejala kelayuan pertama kali terlihat dari bagian atas tanaman, dengan batang yang mengalami pembusukan dan empulur yang berlubang. • Tanaman ini dimasuki bakteri melalui luka-luka bekas pangkasan atau penghilangan tunas samping, kemudian menyerang empulur dan merusaknya sehingga terbentuk lubang-lubang	• Pengendalian penyakit busuk batang dapat dilakukan dengan menggunakan <i>Bacillus</i> sp., yang dicampur dengan pupuk kompos dari kotoran kambing sebanyak 150 gram per polybag. • Oleskan streptomisin sulfat dengan konsentrasi 200-500 ppm untuk luka pada tanaman yang rentan infeksi dalam kondisi lembab.
5	Tobacco Mosaic Virus	• Pengendalian terhadap TMV sering kali tidak memberikan hasil, salah satu penyebabnya adalah penyebarannya yang cepat, didorong oleh vektor dan jangkauan inangnya yang luas. • Gejala yang	• Melakuakn sanitasi seperti membersihkan sisa tanaman dan mencabut tanaman yang sakit. • Sebelum mulai bekerja di kebun, bersihkan tangan dengan mencuci menggunakan deterjen seperti Rinso

		diakibatkan oleh TMV bervariasi tergantung pada tanaman inangnya dan dapat mencakup mosaik, bintik-bintik, nekrosis, kerdil, daun melengkung, dan menguningnya jaringan tanaman	dengan konsentrasi 0,6%, yang efektif untuk menghancurkan protein pelindung virus. - Selain itu, disarankan untuk menanam varietas tembakau yang tahan. - Varietas yang direkomendasikan termasuk tembakau burley seperti Burley 21, Burley 49, Burley 64, Ky 34, Ky 35, Ky 48, Ky 56, dan Ky 57, serta tembakau Besuki varietas NO H 877 dan H 894.
--	--	--	--

4.5.2 Binding Data

Binding data bertujuan untuk menghubungkan sumber data yang tersimpan di *cloud storage* dengan *Integrated Development Environment* (IDE) yang digunakan, dalam hal ini Google Colaboratory dan Google Drive, sehingga data dapat diakses langsung tanpa perlu diunggah terlebih dahulu. Proses tersebut dapat dilihat pada kode berikut:

```
from google.colab import drive
drive.mount('/content/drive')
```

4.5.3 Inisialisasi Data

Inisialisasi data bertujuan untuk menetapkan nilai awal pada variabel agar data tersebut dapat digunakan dalam proses komputasi selanjutnya. Dengan melakukan inisialisasi data, kita menyimpan nilai-nilai yang akan diproses dalam variabel sehingga memungkinkan untuk melakukan langkah-langkah komputasi berikutnya. Proses tersebut dapat dilihat pada kode berikut:

```
# Direktori dataset
dataset_dir = "MyDataset"
```

```
# Direktori untuk data training, validasi, dan testing
train_dir = "my_dataset/training_data"
valid_dir = "my_dataset/validation_data"
test_dir = "my_dataset/testing_data"
```

4.5.4 Pra-Pemrosesan Data

Setelah pengumpulan data dan pengelompokan data sesuai jenisnya kemudian dilakukan pra-pemrosesan data untuk mempersiapkan dataset sebelum dimasukan ke dalam model CNN. Langkah-langkahnya adalah sebagai berikut:

1. Pembagian dataset untuk data *training*, *validation* dan *testing*. Berikut merupakan kode untuk pembagian datasetnya.

```
# Memecah data menjadi train_set dan test_set
train_set, test_set = train_test_split(images, test_size=0.3,
random_state=42)

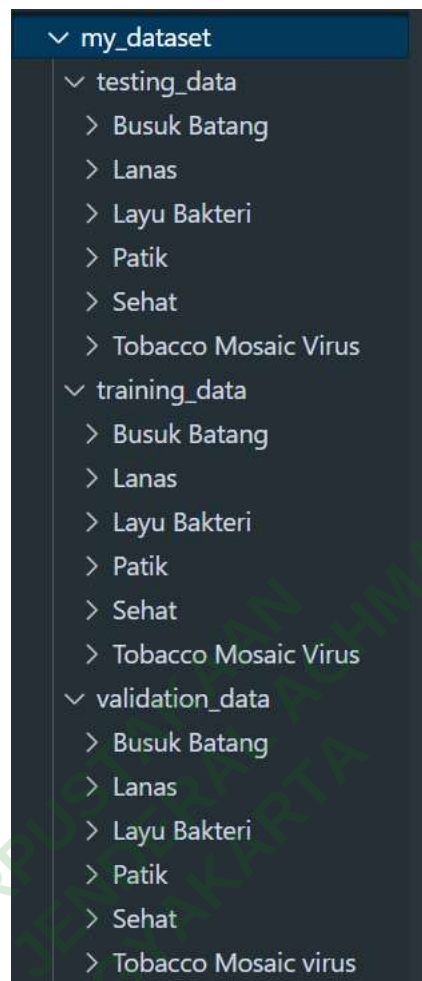
# Memecah test_set menjadi validation_set dan test_set
validation_set, test_set = train_test_split(test_set,
test_size=0.5, random_state=42)

# Menyalin gambar ke dalam direktori masing-masing set
for img_name in train_set:
    src_path = os.path.join(class_dir, img_name)
    dest_path = os.path.join(train_dir, class_name, img_name)
    shutil.copyfile(src_path, dest_path)

for img_name in validation_set:
    src_path = os.path.join(class_dir, img_name)
    dest_path = os.path.join(valid_dir, class_name, img_name)
    shutil.copyfile(src_path, dest_path)

for img_name in test_set:
    src_path = os.path.join(class_dir, img_name)
    dest_path = os.path.join(test_dir, class_name, img_name)
    shutil.copyfile(src_path, dest_path)
```

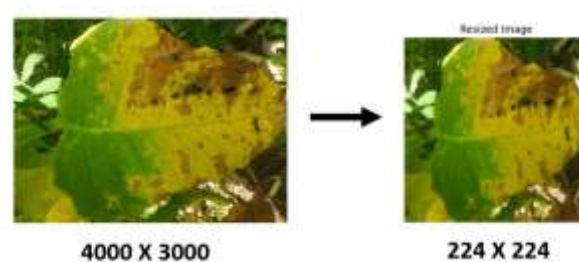
Gambar 4.5 menampilkan hasil dari pembagian dataset tembakau seperti berikut:



Gambar 4.5 Direktori dataset tembakau

2. Setelah pembagian dataset langkah selanjutnya yaitu mengubah ukuran gambar menjadi 224 x 224 piksel seperti yang ditunjukkan Gambar 4.6. Proses tersebut dapat dilihat pada perubahan berikut:

```
# Resizing gambar menjadi 224x224 piksel
target_size = (224, 224)
```



Gambar 4.6 Resized gambar

3. Augmentasi dan normalisasi data bertujuan untuk memperkaya variasi dataset dan mencegah *overfitting*, di mana model mungkin menunjukkan kinerja yang bagus saat dilatih namun tidak mampu mengatasi data baru. Dengan mengkombinasikan keduanya, kita dapat meningkatkan performa model dengan memperluas variasi data melalui transformasi gambar serta menyesuaikan nilai piksel untuk menjaga kestabilan saat proses pelatihan dan memastikan adaptasi model yang baik terhadap berbagai situasi dalam dataset. Proses tersebut dapat dilihat pada kode berikut:

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

valid_test_datagen = ImageDataGenerator(rescale=1./255)

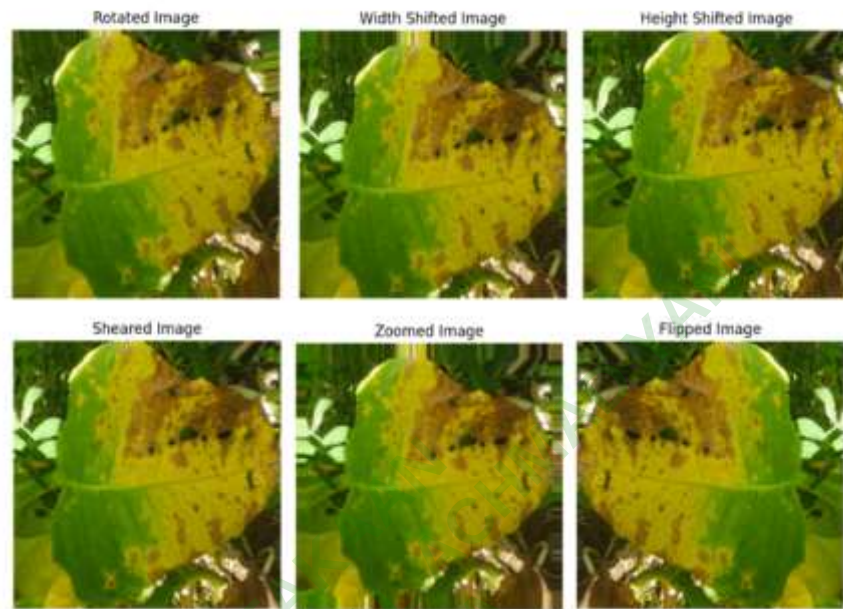
train_generator = train_datagen.flow_from_directory(
    os.path.join(dataset_dir, 'training_data'),
    target_size=target_size,
    batch_size=32,
    class_mode='categorical',
    shuffle=True)

valid_generator = valid_test_datagen.flow_from_directory(
    os.path.join(dataset_dir, 'validation_data'),
    target_size=target_size,
    batch_size=32,
    class_mode='categorical',
    shuffle=False)

test_generator = valid_test_datagen.flow_from_directory(
    os.path.join(dataset_dir, 'testing_data'),
    target_size=target_size,
    batch_size=32,
    class_mode='categorical',
    shuffle=False)
```

Pada algoritma di atas dapat dilihat bahwa proses augmentasi data melakukan rotasi 20 derajat, *shear_range* 20%, *zoom_range* 20% dan normalisasi data. Model akan memproses 32 gambar sekaligus dalam setiap training data.

sesuai dengan besar *batch_size*. Gambar 4.7 merupakan gambar hasil augmentasi dan normalisasi.



Gambar 4.7 Hasil augmentasi & normalisasi data

4.5.5 Model Development

Setelah selesai melakukan preprosesing pada dataset, langkah selanjutnya adalah membangun model menggunakan arsitektur VGG16 dengan bobot yang telah dilatih dari ImageNet, serta model Convolutional Neural Network (CNN).

1. Perancangan model

Memuat model VGG16 yang sudah dilatih sebelumnya, namun tidak menyertakan lapisan atasnya (*fully connected layers*). Proses tersebut dapat dilihat pada kode berikut:

```
VGG16_model = tf.keras.applications.VGG16(weights='imagenet',
include_top=False, input_shape=(224, 224, 3))
VGG16_model.trainable = False

vgg16_model.summary()
```

Summary dari model vgg16 ditunjukkan pada Gambar 4.8.

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
=====		
Total params: 14714688 (56.13 MB)		
Trainable params: 0 (0.00 Byte)		
Non-trainable params: 14714688 (56.13 MB)		

Gambar 4.8 Summary model VGG16

Setelah membuat basis model dengan menggunakan VGG16 yang telah dipra-latih, kemudian membangun dan mengompilasi model CNN tambahan. Model ini menerima input gambar berukuran 224 x 224 x 3, kemudian menambahkan lapisan konvolusional dengan 32 *filter* dan lapisan *pooling* maksimum. Setelah itu, *output* data diratakan dengan lapisan *flatten*, diikuti oleh lapisan *fully connected* dengan 256 unit dan aktivasi ReLU, serta *dropout* sebesar 0.5 untuk mencegah *overfitting*. kemudian lapisan *output* dengan 6 unit dan aktivasi *softmax* digunakan untuk klasifikasi menjadi 6 kelas. Model ini

dikompilasi dengan *optimizer Adam*, *loss categorical_crossentropy*, dan metrik akurasi. Proses tersebut dapat dilihat pada kode berikut:

```
model = models.Sequential([
    layers.Input(shape=(224, 224, 3))
    base_model
    layers.Conv2D(filters=32, padding='same',
kernel_size=3,activation='relu', strides=1),
    layers.MaxPool2D(pool_size=2, strides=2),
    layers.Flatten(),
    layers.Dense(256, activation='relu'), n
    layers.Dropout(0.5),
    layers.Dense(6, activation='softmax')
])

model.compile(optimizer=Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

Berikut merupakan *summary* dari model yang ditunjukkan pada Gambar

4.9.

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14714688
conv2d (Conv2D)	(None, 7, 7, 32)	147488
max_pooling2d (MaxPooling2D)	(None, 3, 3, 32)	0
flatten (Flatten)	(None, 288)	0
dense (Dense)	(None, 256)	73984
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 6)	1542

=====
Total params: 14937702 (56.98 MB)
Trainable params: 223014 (871.15 KB)
Non-trainable params: 14714688 (56.13 MB)
=====

Gambar 4.9 *Summary* model tambahan

2. Training dan Validasi Model

Proses tersebut dapat dilihat pada kode berikut:

```

epochs = 100

history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data=valid_generator,
)

```

Berdasarkan Algoritma di atas, program melatih model CNN menggunakan data yang dimasukkan, sehingga menghasilkan empat nilai berikut di konsol Colaboratory, seperti yang ditunjukkan pada Tabel 4.2.

Tabel 4.2 Hasil training model CNN

No	Loss	Accuracy	Val_Loss	Val_Accuracy
1	1.9169	0.1881	1.6389	0.3111
2	1.5992	0.3190	1.3813	0.4222
3	1.4111	0.3714	1.1251	0.5667
4	1.3995	0.4024	1.698	0.4889
...				
96	0.0762	0.9762	0.6847	0.8889
97	0.0732	0.9690	0.8000	0.8778
98	0.0886	0.9714	0.6103	0.8778
99	0.0696	0.9738	0.8036	0.8556
100	0.0889	0.9738	0.9249	0.8556

Tabel 4.2 dapat disimpulkan bahwa dari 100 *epoch* yang dilakukan proses *training* dan validasi terlihat pada *epoch* 96 mendapatkan akurasi tertinggi yaitu sebesar 97,62% dengan nilai *loss* yang rendah yaitu 0,0762. Memiliki nilai *val_accuracy* sebesar 0.8889 yang menunjukkan bahwa model mampu menghasilkan prediksi yang akurat dan dapat menggeneralisasi dengan baik pada data validasi, dengan *val_loss* mencapai 0.6847 sebagai indikasi keefektifan optimisasi model. Model yang baik akan memiliki *loss* rendah dan akurasi yang tinggi (Allaam & Wibowo, 2021).

4.5.6 Evaluasi Model

Tahap evaluasi model dilakukan dengan menjalankan kode untuk menghasilkan *confusion matrix* dan menghitung tingkat akurasi. Di samping itu, metrik evaluasi lain seperti presisi, *recall*, dan skor F1 juga dianalisis untuk memberikan pemahaman yang menyeluruh tentang kinerja model. Langkah-langkah ini sangat penting Untuk menilai sejauh mana model dapat melakukan klasifikasi data dengan tepat serta untuk mengidentifikasi area model yang mungkin memerlukan peningkatan lebih lanjut. Proses tersebut dapat dilihat pada kode berikut:

```
import numpy as np
from sklearn.metrics import confusion_matrix,
classification_report
import matplotlib.pyplot as plt
import seaborn as sns

def evaluate_model_on_test_set(model, test_generator,
class_names):
    total_samples = test_generator.samples
    test_generator.reset()
    y_pred_prob = model.predict(test_generator,
steps=np.ceil(total_samples / test_generator.batch_size),
verbose=1)

    y_true = test_generator.classes
    y_pred = np.argmax(y_pred_prob, axis=1)

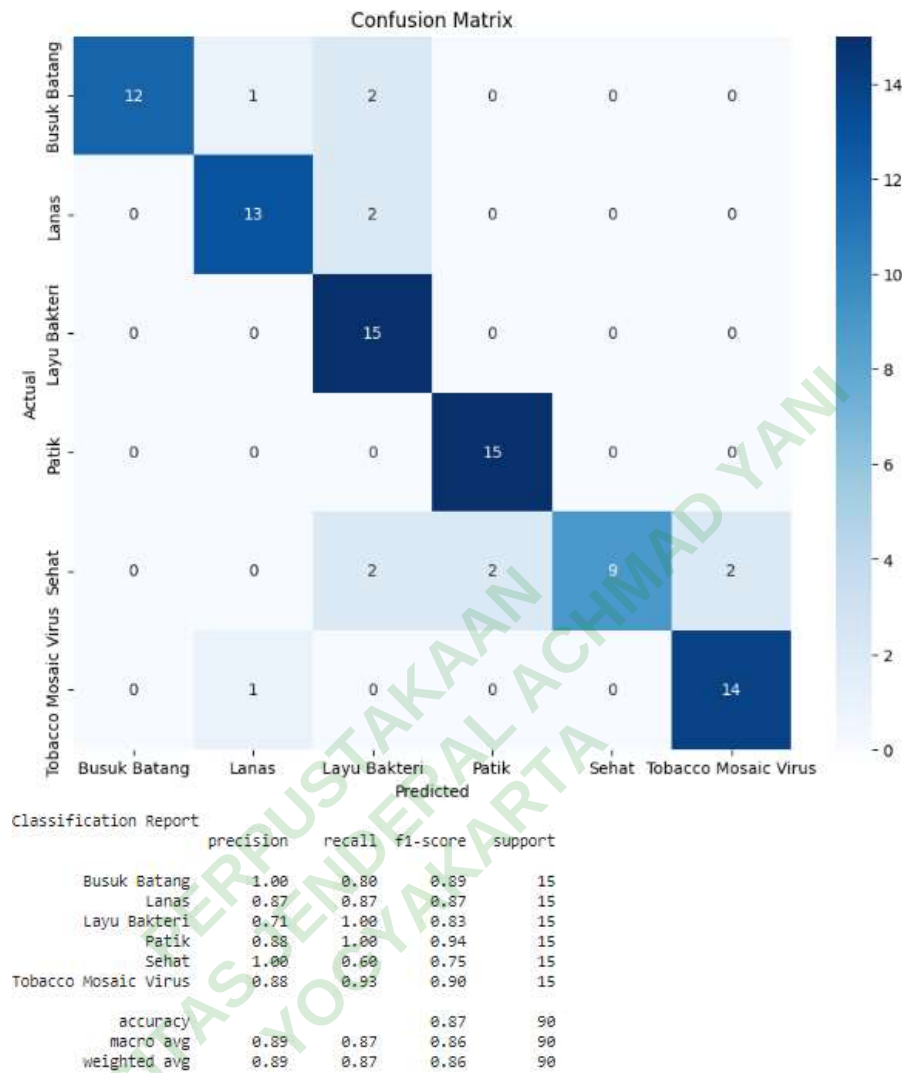
    cm = confusion_matrix(y_true, y_pred)

    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.show()

    cr = classification_report(y_true, y_pred,
target_names=class_names)
    print('Classification Report')
    print(cr)

class_names = list(test_generator.class_indices.keys())
evaluate_model_on_test_set(model, test_generator, class_names)
```

Evaluasi model ini dilakukan dengan data uji sebanyak 90 gambar, untuk setiap kelasnya 15 gambar. Hasilnya ditunjukkan pada Gambar 4.10.



Gambar 4.10 Confusion matrix & clasification report

Hasil dari *confusion matrix* dan *clasification report* pada gambar 4.10 dijelaskan sebagai berikut:

1. Busuk Batang memiliki *precision* yang sempurna (1.00) tetapi *recall* lebih rendah (0.80), menunjukkan bahwa model terkadang tidak mendeteksi semua contoh busuk batang dengan benar.
2. Lanas menunjukkan kinerja yang seimbang dengan *precision* dan *recall* sebesar 0.87, berarti model cukup akurat dalam mengidentifikasi penyakit ini.

3. Layu Bakteri memiliki *recall* sempurna (1.00) tetapi *precision* lebih rendah (0.71), berarti meskipun model mendeteksi semua kasus layu bakteri, ada banyak prediksi yang salah.
4. Patik menunjukkan kinerja sangat baik dengan *precision* dan *recall* mendekati sempurna.
5. Sehat memiliki *precision* yang tinggi (1.00) tetapi *recall* yang lebih rendah (0.60), menunjukkan model sering gagal mengidentifikasi kondisi sehat dengan benar.
6. Tobacco Mosaic Virus memiliki kinerja baik dengan *precision* dan *recall* tinggi.
7. Secara keseluruhan, model ini cukup akurat dengan akurasi 87%, tetapi ada ruang untuk perbaikan, terutama dalam meningkatkan *recall* untuk kelas "Sehat" dan *precision* untuk kelas "Layu Bakteri". Seperti menurut Barrihadiano (2023) bahwa model dengan *precision* dan *recall* yang seimbang lebih diinginkan karena menunjukkan kinerja yang baik dalam kedua aspek, meskipun seringkali ada *trade-off* di antara keduanya.

4.5.7 Implementasi Model

Di tahap ini, model yang telah dilatih sebelumnya dan tersimpan dalam format .h5 menggunakan TensorFlow akan diintegrasikan ke dalam aplikasi web menggunakan *framework* Flask. Proses ini memastikan bahwa model yang telah terlatih dapat dijalankan dan digunakan secara langsung dalam konteks aplikasi web yang interaktif. Langkah pertama dalam penggunaan model yang disimpan adalah dengan memuatnya kembali menggunakan fungsi “*load_model*” dari TensorFlow. Fungsi ini memungkinkan aplikasi Flask untuk mengakses dan menggunakan model CNN yang telah tersimpan, sehingga dapat digunakan untuk melakukan prediksi berdasarkan gambar yang diunggah oleh pengguna melalui antarmuka web. Proses *load* model dapat dilihat pada kode berikut:

```
model_path = "models/modeltembakau.h5"
model = load_model(model_path))
```

Setelah menyimpan *path file* dari gambar yang akan diklasifikasikan, proses klasifikasi akan dilakukan melalui fungsi '*predict*'. Fungsi tersebut mengambil *file* gambar sebagai *input* yang kemudian melakukan proses klasifikasi dengan menjalankan kode berikut:

```
@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'})

    file = request.files['file']

    if file.filename == '':
        return jsonify({'error': 'No selected file'})

    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'],
filename)
        file.save(filepath)

        img = preprocess_image(filepath)
        img_array = np.expand_dims(img, axis=0)

        prediction = model.predict(img_array)
        predicted_class_index = np.argmax(prediction)
        predicted_class_probability =
float(prediction[0][predicted_class_index])
        minimum_accuracy_threshold = 0.50

        if predicted_class_probability <
minimum_accuracy_threshold:
            return jsonify({'error': 'Prediksi kurang akurat,
silakan unggah gambar yang lebih sesuai.'})

        class_names = ['Busuk Batang', 'Lanas', 'Layu Bakteri',
'Patik', 'Sehat', 'Tobacco Mosaic Virus']
        predicted_class_name = class_names[predicted_class_index]

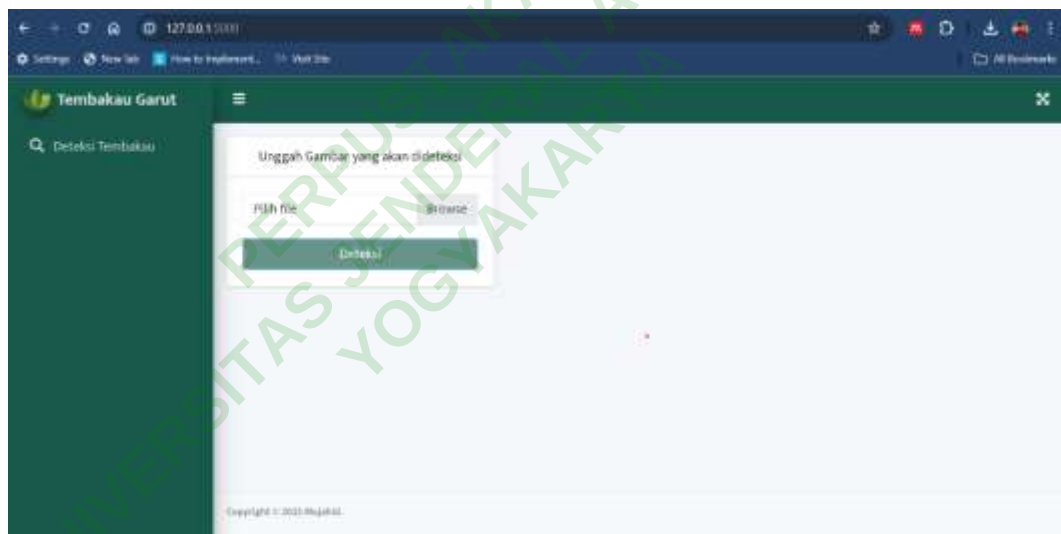
        disease_info = get_disease_info(predicted_class_name)
        if predicted_class_name == 'Sehat':
            result = {
                'class': predicted_class_name,
                'probability': predicted_class_probability,
                'Gejala': None,
                'Solusi': None,
            }
        else:
            disease_info = get_disease_info(predicted_class_name)
            if disease_info:
                Gejala, Solusi = disease_info
```

```

        result = {
            'class': predicted_class_name,
            'probability': predicted_class_probability,
            'Gejala': Gejala,
            'Solusi': Solusi,
        }
    else:
        result = {'error': 'Informasi penyakit tidak
ditemukan dalam database.'}
        return jsonify(result)
    else:
        return jsonify({'error': 'Format File tidak
dizinkan
tolong ubah ke format jpg, jpeg atau png'})

```

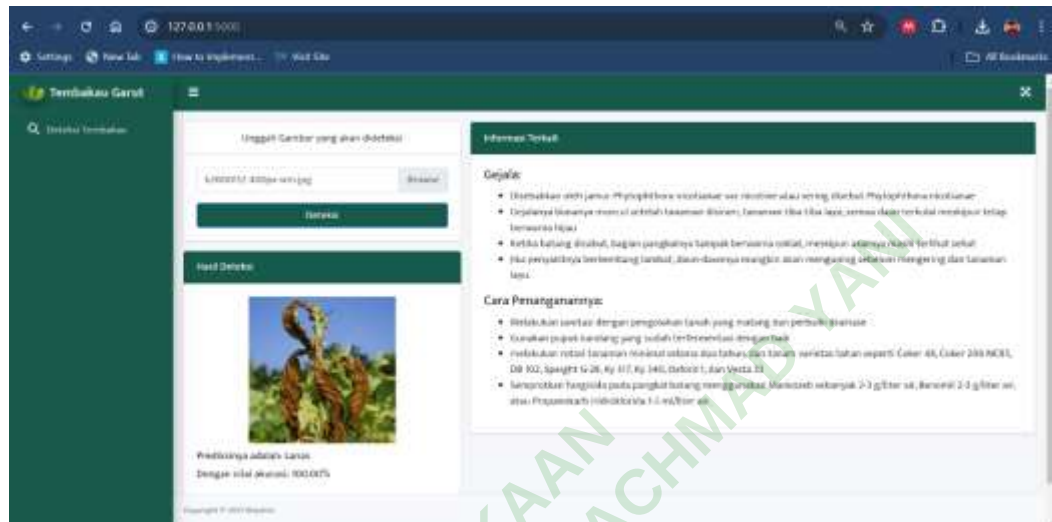
Sehingga ketika aplikasi dijalankan akan menampilkan seperti pada Gambar 4.11 dan ketika proses klasifikasi dilakukan, hasil dari deteksi akan ditampilkan kepada pengguna sesuai dengan kondisi daun tembakau yang diunggah seperti yang terlihat pada Gambar 4.12 dan Gambar 4.13.



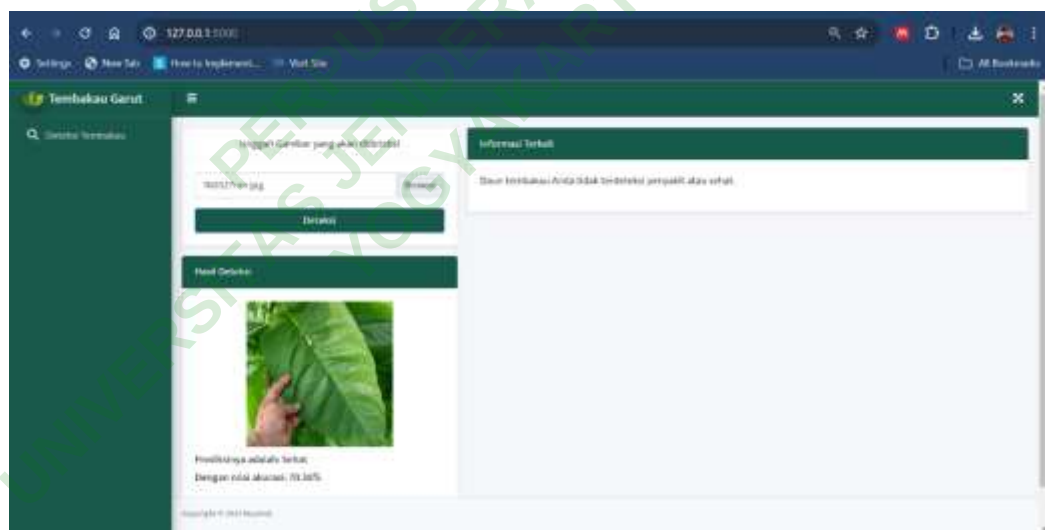
Gambar 4.11 Halaman awal untuk input gambar

Gambar 4.11 merupakan halaman awal untuk input gambar yang akan dideteksi. Pada gambar tersebut terdapat menu untuk mengunggah file, kemudian setelah gambar berhasil diinputkan pengguna harus menekan tombol deteksi agar gambar tersebut diproses dan menampilkan hasil seperti pada Gambar 4.12 dan 4.13. Pada Gambar 4.12 hasil deteksi menunjukkan tembakau berpenyakit sehingga output yang dihasilkan tidak hanya menampilkan hasil prediksi dan akurasinya saja melainkan informasi terkait gejala dan solusi untuk penyakit tersebut.

Sedangkan Gambar 4.13 menampilkan hasil prediksi tembakau dengan kondisi sehat sehingga output yang ditampilkan hanya berisi hasil prediksi dan akurasinya saja.



Gambar 4.12 Hasil deteksi tembakau berpenyakit



Gambar 4.13 Hasil deteksi tembakau sehat

4.5.8 Pengujian Sistem

Dari hasil implementasi sistem di atas akan dilakukan pengujian yang hasilnya ditunjukkan pada table 4.3.

Tabel 4.3 Hasil pengujian

Fungsi yang diuji	Kasus Uji	Hasil yang diharapkan	Hasil Pengujian	
			Sesuai	Tidak sesuai
Fitur Unggah	Unggah file dengan format jpg, jpeg atau png.	File berhasil terunggah dan siap diproses	✓	
	Unggah file dengan format selain jpg, jpeg dan png.	Menampilkan validasi format file salah	✓	
	Unggah gambar tembakau dengan hipotesis berpenyakit	Menampilkan hasil deteksi dan informasi terkait dengan penyakit yang terdeteksi.	✓	
	Unggah gambar tembakau dengan hipotesis sehat	Menampilkan hasil deteksi dan informasi bahwa tembakau tersebut sehat.	✓	
	Unggah gambar selain tembakau atau hasil deteksi dibawah 50%.	Menampilkan validasi unggah dengan gambar yang sesuai.	✓	

Tabel 4.3 menunjukkan bahwa fitur yang ada pada sistem deteksi ini berhasil berjalan sesuai harapan. Sedangkan hasil pengujian terhadap kinerja model yang telah diimplementasikan kedalam sistem dengan data yang diuji yaitu 12 data yang diambil dari dataset uji dapat dilihat pada Tabel 4.4 dan untuk hasil pengujian terhadap 20 data yang diambil dari internet atau diluar dataset dapat dilihat pada Tabel 4.5.

Tabel 4.4 Hasil pengujian terhadap dataset uji

No	Gambar yang Diuji	Klasifikasi yang Sesuai	Hasil Deteksi	Kesimpulan
1		Busuk Batang	Busuk Batang	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
2		Busuk Batang	Busuk Batang	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
3		Lanas	Lanas	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
4		Lanas	Lanas	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
5		Layu Bakteri	Layu Bakteri	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
6		Layu Bakteri	Layu Bakteri	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
7		Patik	Patik	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.

8		Patik	Patik	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
9		Sehat	Sehat	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
10		Sehat	Patik	Disimpulkan salah karena hasil deteksi dengan klasifikasinya bernilai berbeda.
11		TMV	TMV	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
12		TMV	TMV	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.

Berdasarkan pengujian di atas didapatkan informasi bahwa dari 12 data yang diuji terdapat 1 data yang berkesimpulan salah yaitu pada data dengan klasifikasi sehat tetapi hasil deteksinya penyakit patik. Faktor penyebabnya bisa jadi karena model untuk pengklasifikasian terhadap kelas sehat kurang baik seperti yang telah dijelaskan sebelumnya bahwa perlu perbaikan terhadap nilai *recall* kelas sehat. Berikut merupakan perhitungan nilai akurasi dari hasil pengujian pada Tabel 4.4.

$$Akurasi = \frac{11}{12} \times 100\% = 91,67\%$$

Tabel 4.5 Hasil pengujian data dari internet

No	Gambar yang Diuji	Klasifikasi yang Sesuai	Hasil Deteksi	Kesimpulan
1		Lanas	Lanas	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
2		Lanas	Lanas	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
3		Layu Bakteri	Lanas	Disimpulkan salah karena hasil deteksi dengan klasifikasinya bernilai berbeda.
4		Sehat	Patik	Disimpulkan salah karena hasil deteksi dengan klasifikasinya bernilai berbeda.
5		TMV	TMV	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
6		Lanas	Lanas	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.

7		TMV	TMV	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
8		Busuk Batang	Busuk Batang	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
9		Busuk Batang	Busuk Batang	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
10		Lanas	Lanas	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
11		Patik	Patik	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
12		Patik	Patik	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
13		Patik	Patik	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.

14		Layu Bakteri	Lanas	Disimpulkan salah karena hasil deteksi dengan klasifikasinya bernilai berbeda.
15		Layu Bakteri	Lanas	Disimpulkan salah karena hasil deteksi dengan klasifikasinya bernilai berbeda.
16		Sehat	Layu Bakteri	Disimpulkan salah karena hasil deteksi dengan klasifikasinya bernilai berbeda.
17		Layu Bakteri	Layu Bakteri	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
18		Sehat	Layu bakteri	Disimpulkan salah karena hasil deteksi dengan klasifikasinya bernilai berbeda.
19		Sehat	Sehat	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.

20		Sehat	Sehat	Disimpulkan benar karena hasil deteksi dengan klasifikasinya bernilai sama.
----	---	-------	-------	--

Berdasarkan pengujian di atas didapatkan informasi bahwa dari 20 data yang diuji terdapat 6 data yang berkesimpulan salah yaitu mayoritas pada data dengan klasifikasi sehat dan layu bakteri. Faktor penyebab salahnya hasil deteksi bisa jadi karena model untuk pengklasifikasian terhadap kelas sehat dan layu bakteri kurang baik seperti yang telah dijelaskan sebelumnya bahwa perlu perbaikan terhadap nilai *recall* kelas sehat dan *precision* kelas layu bakteri. Kualitas gambar seperti resolusi rendah, kurangnya pencahayaan dan kejernihan juga bisa menjadi salah satu faktornya. Seperti menurut (Setiyono et al., 2023) bahwa tingkat pencahayaan pada gambar memiliki pengaruh yang signifikan terhadap hasil deteksi aplikasi. Sedangkan untuk perhitungan nilai akurasi dari hasil pengujian pada Tabel 4.5. adalah sebagai berikut:

$$Akurasi = \frac{14}{20} \times 100\% = 70\%$$

Hasil pengujian pada Tabel 4.4 dan Tabel 4.5 menunjukkan bahwa model CNN yang telah dilatih cukup baik dalam mengenali jenis penyakit tanaman tembakau, akan tetapi model ini kurang baik dalam mengklasifikasikan untuk kelas layu bakteri dan sehat khususnya untuk data yang diuji dari internet atau di luar dataset.