BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi CNN

Implementasi CNN dibagi menjadi 2 bagian, yaitu implementasi dari proses training model dan implementasi klasifikasi pada perangkat lunak berdasarkan skenario yang telah dirancang pada Bab 3. Dari hasil pengujian yang telah didapatkan selanjutnya diberikan pembahasan dan analisis pengujian untuk mendapatkan hasil penelitian, sehingga mendapatkan kesimpulan yang diberikan pada pembahasan selanjutnya.

4.1.1 Binding Data

Binding data atau pengikatan data adalah proses menghubungkan komponen antarmuka pengguna (UI) dengan sumber data. Dalam hal ini menggunakan google colab sehingga dapat digunakan tanpa proses uploding. Proses tersebut dapat dilihat pada algoritma berikut:

```
[ ]  from google.colab import userdata
     import os
```

Gambar 4. 1 algoritma binding data

4.1.2 Inisilisasi Data

Inisialisasi data adalah proses pengisian nilai awal pada objek data atau variabel dalam pemrograman computer. Proses tersebut dapat dilihat pada algoritma berikut:

```
base_dir = "/content/dataset_new"
train_dir = os.path.join(base_dir, "train")
val_dir = os.path.join(base_dir, "val")
test_dir = os.path.join(base_dir, "test")
```

Gambar 4. 2 Algoritma Inisialisasi Data

4.1.3 Proses Preprocessing dan Augmentasi Data

Salah satu proses utama yang ada pada aplikasi ini adalah prepocessing. Prose ini berujuan untuk mempersiapkan data sebelum diolah lebih lanju dan di augmentasi.

Augmentasi data berfungsi agar kebuuhan program terhadap data yang besar dapat dipenuhi. Proses tersebutdapat dilihat pada gambar berikut:

```
# Augmentasi dan memuat dataset
train_datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

val_test_datagen = ImageDataGenerator(rescale=1.0/255.0)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)

val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
```

Gambar 4. 3 Prepocecing Dan Augmentasi Data

Pada gambar 4 dapat kia lihat dilakukan proses augmentasi data citra dengan melakukan rotasi sebesar 20 derajat, share range 20%, zoom range 20%, dan normalisasi cira.

4.1.4 Inisialisasi Model CNN

Inisialisasi model CNN diperlukan sebelum memasuki proses *training*. Proses pembuatan model CNN ini diawali dengan *preprocessing* yang telah dibahas pada subbab 4.1.3. Dimana terdiri dari rescale, rotation, shear, dan zoom kemudian pada proses ini dilakukan input hyperparameter yang akan diujikan. Proses tersebut dapat kita lihat pada gambar 4.4 berikut

```
# # Membekukan layer VGG19 untuk transfer learning
# base_model.trainable = False

# Menambahkan lapisan klasifikasi
model = Sequential([
    # base_model,
    # Flatten(),
    # Dense(256, activation='relu'),
    # Dropout(0.5),
    # Dense(num_classes, activation='softmax')
    layers.Conv2D(filters = 16, kernel_size = (3,3), activation = 'relu', input_shape = (img_height, img_width, 3)),
    layers.MaxPool2D((2,2)),
    layers.Conv2D(64, (3,3), activation = 'relu'),
    layers.MaxPool2D((2,2)),
    layers.Conv2D(128, (3,3), activation = 'relu'),
    layers.MaxPool2D((2,2)),
    layers.Conv2D(64, (3,3), activation = 'relu'),
    layers.MaxPool2D((2,2)),
    layers.Conv2D(128, (3,3), activation = 'relu'),
    layers.MaxPool2D((2,2)),
    layers.Conv2D(64, (3,3), activation = 'relu'),
    layers.MaxPool2D((2,2)),
    layers.Flatten(),
    layers.Dense(num_classes, activation = 'softmax')
])
```

```
# Kompilasi model
model.compile(
    optimizer=RMSprop(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Gambar 4. 4 Inisialisasi Model CNN

4.1.5 Proses Training Model CNN

Setelah melakukan prose inisialisasi model CNN, model tersebu melalui tahap selanjutnya yaitu proses training. Proses ini dilakukan dengan menginputkn beberapa nilai hyperparamenter lainnya yaitu number of epochs dan number of validation steps. Proses tersebut dapat dilihat pada gambar 3.5 berikut:

```
# Melatih model
history = model.fit(
    train_generator,
    epochs=15,
    validation_data=val_generator,
    verbose=1
)
```

Gambar 4. 5 Proses Training Model

Berdasarkan gambar 4.5 Program melakukan training pada model CNN berdasarkan data yang diinputkan sehingga menghasilkan empat nilai berikut seperti pada tabel 4.1 berikut:

Tabel 4. 1 Hasil training Model CNN

No	Accuracy	Loss	Validaion Accuracy	Validation Loss
1	0.4502	1.0691	0.7081	0.6847
2	0.7081	0.6587	0.7405	0.5584
3	0.7254	0.6118	0.7351	0.7619
....	....	....	....	....
9	0.9007	0.2469	0.9054	0.1921
....	....	....	....	....
13	0.9305	0.1737	0.9027	0.1825
14	0.9400	0.1449	0.9324	0.1549
15	0.9409	0.1450	0.8865	0.2527

Berdasarkan tabel 4.1 Dari 15 epochs yang dilakukan proses training terlihat bahwa epoch 15 mendapatkan akurasi teringgi sebesar 0.9409 dengan loss 0.1450.

4.1.6 Evaluasi Model CNN

Tahapan evaluasi model dilakukan dengan mengeksekusi kode untuk menghasilkan confusion matrix, tingkat akurasi. Hal ini menjadi point penting pada proses evaluasi model CNN. Proses pemanggilan tersebut disaikan pada gambar 4.6 Berikut:

```
# Evaluasi model
loss, accuracy = model.evaluate(test_generator)
print(f"Test Loss: {loss}")
print(f"Test Accuracy: {accuracy}")
```

Gambar 4. 6 Evaluasi Akurasi

Sehingga didapatkan hasil output seperti pada gambar 4.7 Berikut:

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 65ms/step - accuracy: 0.9166 - loss: 0.2184
Test Loss: 0.2158350944519043
Test Accuracy: 0.9164420366287231
```

Gambar 4. 7 Output Evaluasi Model CNN

Dari gambar 4.6 Menghasilkan output seperti pada gambar 4.7, sehingga diketahui hasil bahwa tingkat akurasi model paling optimal yang dihasilkan pada penelitian ini adalah 91.66% dengan nilai loss 21.84%. setelah diketahui tingkat akurasi yang dihasilkan oleh model, hasil prediksi dari keseluruhan dataset yang digunakan sebagai data uji dapat dilihat dengan mengeksekusi program seperti pada gambar 4.8 Berikut:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder
from keras.models import load_model
from keras.preprocessing import image
```

Gambar 4. 8 Program Hasil Prediksi

4.2 Implementasi Perangkat Lunak

Bagian ini menjelaskan hasil dari pengembangan sistem yang telah dirancang sebelumnya. Sistem yang dikembangkan digunakan untuk pengujian model dan menampilkan hasil klasifikasi pada penelitian ini. Implementasi pada perangkat lunak dapat dilakukan setelah model CNN selesai dilakukan proses training, hal ini dimungkinkan apabila pengguna menyimpan hasil proses training tersebut. Model yang sudah disimpan dapat dilihat pada gambar 4.9 Berikut:

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 222, 222, 16)	448
max_pooling2d_6 (MaxPooling2D)	(None, 111, 111, 16)	0
conv2d_7 (Conv2D)	(None, 109, 109, 64)	9,280
max_pooling2d_7 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_8 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_8 (MaxPooling2D)	(None, 26, 26, 128)	0
conv2d_9 (Conv2D)	(None, 24, 24, 64)	73,792
max_pooling2d_9 (MaxPooling2D)	(None, 12, 12, 64)	0
conv2d_10 (Conv2D)	(None, 10, 10, 128)	73,856
max_pooling2d_10 (MaxPooling2D)	(None, 5, 5, 128)	0
conv2d_11 (Conv2D)	(None, 3, 3, 64)	73,792
max_pooling2d_11 (MaxPooling2D)	(None, 1, 1, 64)	0
flatten_1 (Flatten)	(None, 64)	0
dense_2 (Dense)	(None, 128)	8,320
dense_3 (Dense)	(None, 3)	387

```
Total params: 313,733 (1.20 MB)
Trainable params: 313,731 (1.20 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2 (12.00 B)
```

Gambar 4. 9 Model Cnn

Pada gambar 4. 25 Dapat kita lihat bahwa model menyimpan arsitektur yang telah kita training. Pada model ini terdapat 5 kali konvolusi dan 5 kali max pooling. File tersebut

diimplementasikan pada perangkat lunak dengan menuliskan kode seperti pada gambar 4.10 berikut:

```
from tensorflow.keras.models import load_model

# Load model
model = load_model("vgg_lost.h5")

# Cek arsitektur model (terutama layer terakhir)
model.summary()

# Cek output layer (jumlah kelas)
output_layer = model.layers[-1]
print("Layer terakhir:", output_layer)
# print("Aktivasi:", output_layer.activation)
# print("Jumlah Output:", output_layer.output_shape)
```

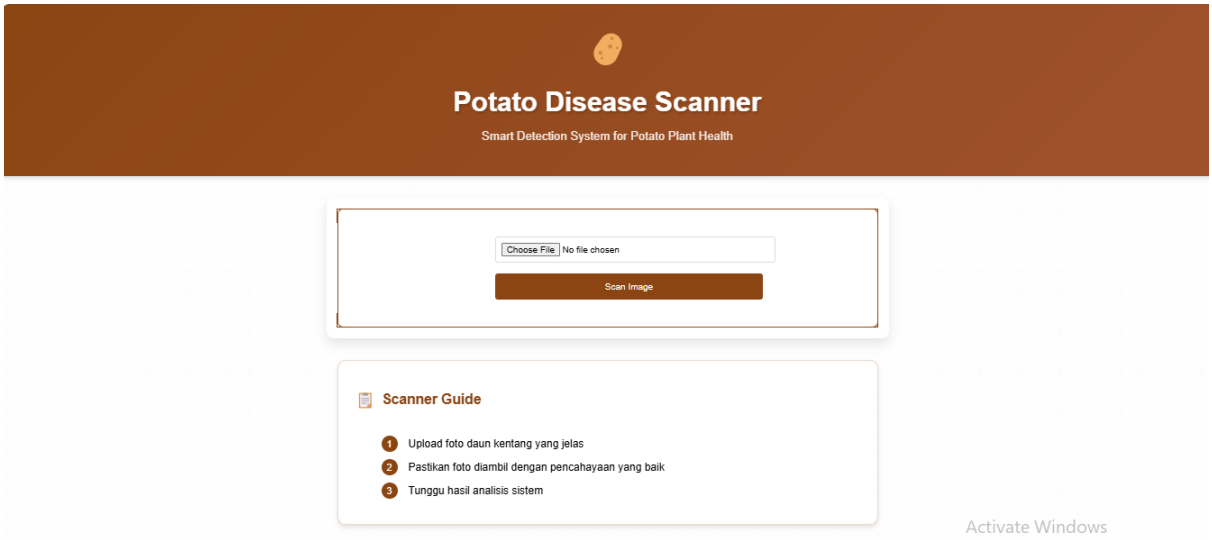
Gambar 4. 10 Proses Load Model Pada Perangkat Lunak

Citra yang akan di klasifikasikan tersebut disimpan lalu dilakukan proses klasifikasi pada funcion preict. Proses klasifikasi tersebut dilakukan dengan mengeksekusi kode pada gambar 4.11 berikut:

```
# Daftar nama kelas
classes = ['Potato_Early_blight', 'Potato_Late_blight', 'Potato_healthy']

# Inisialisasi label encoder
label_encoder = LabelEncoder()

# Inisialisasi list untuk menyimpan true labels dan predicted labels
true_labels = []
predicted_labels = []

# Loop melalui setiap subfolder kelas di direktori test
for class_name in classes:
    class_dir = os.path.join(test_dir, class_name)
    for image_name in os.listdir(class_dir):
        # Mendapatkan true label dari nama subfolder
        true_label = class_name
        true_labels.append(true_label)

        # Memuat dan melakukan preprocessing pada gambar
        img_path = os.path.join(class_dir, image_name)
        img = image.load_img(img_path, target_size=(224, 224, 3))
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array /= 255.  # Normalisasi

        # Lakukan prediksi pada gambar
        predicted_label = model.predict(img_array)
        predicted_label = classes[np.argmax(predicted_label)]
        predicted_labels.append(predicted_label)
```

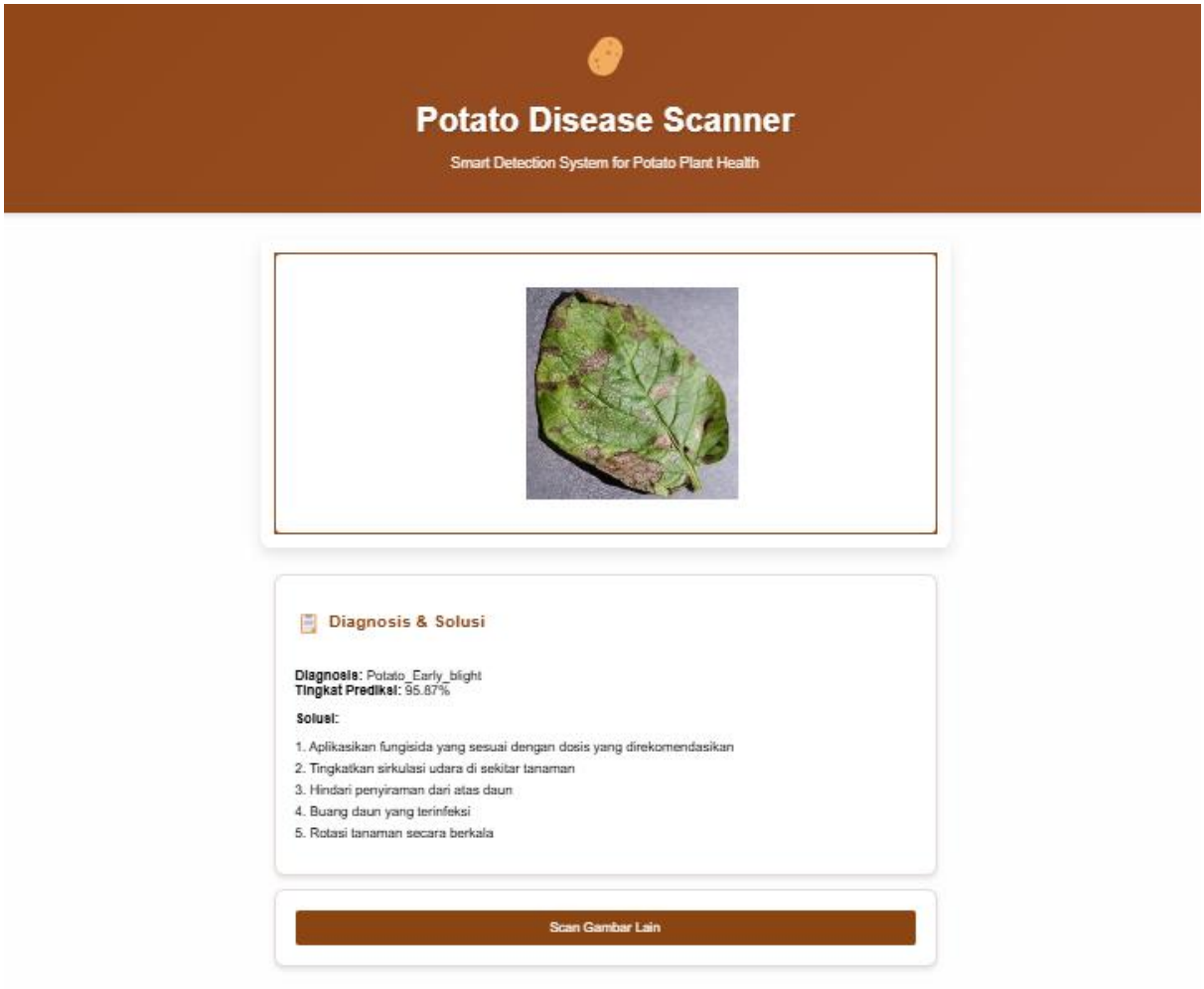
Gambar 4. 11 Proses Klasifikasi Dan Menampilkan Hasil

Sehingga apabila program website dijalankan dan dilakukan proses klasifikasi, akan terlihat pada gambar 4.12, 4.13, 4.14, dan 4.15 berikut:



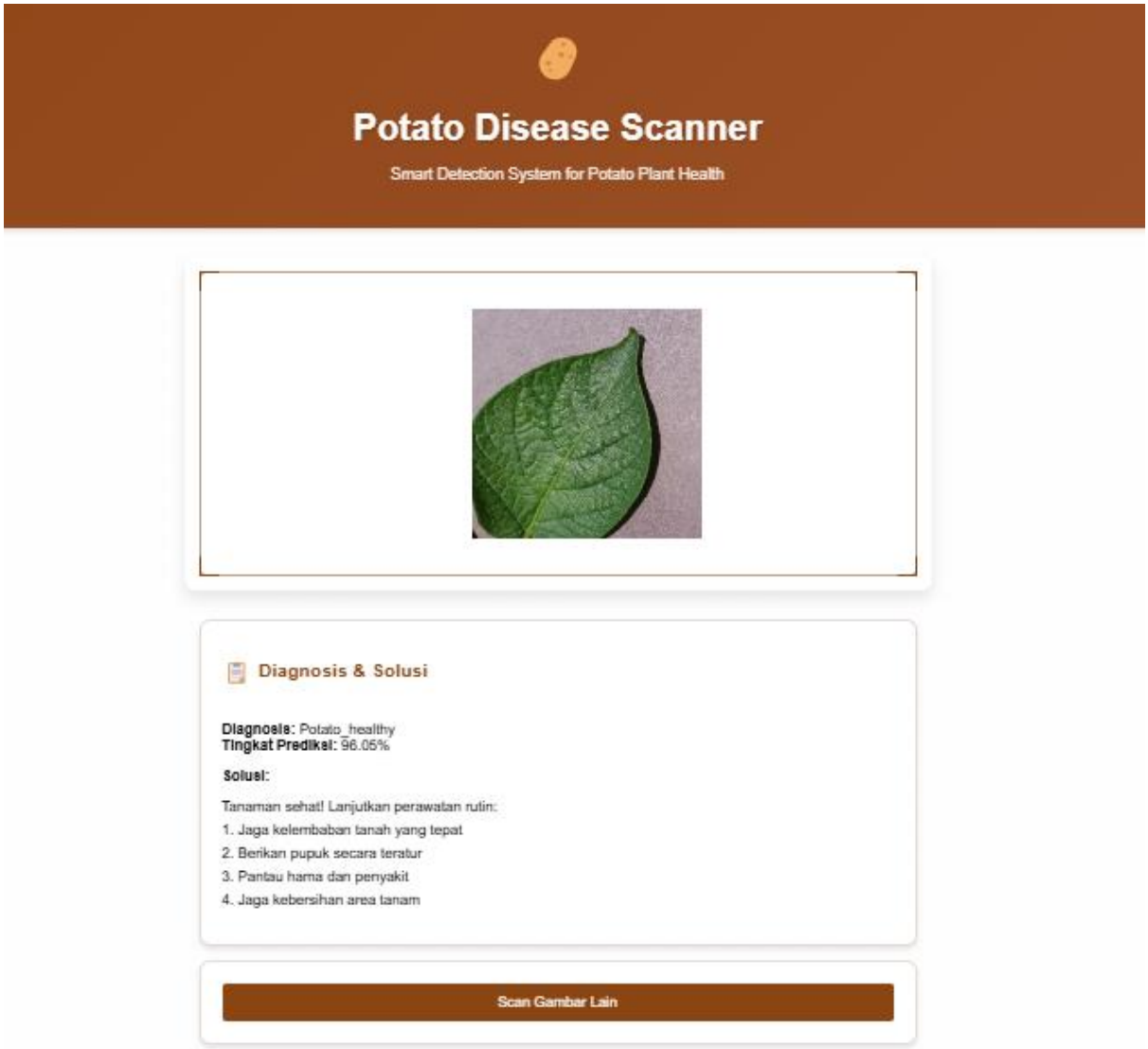
Gambar 4. 12 Halaman Awal Untuk Menginput Data Citra

Pada gambar 4.12 Merupakan halaman awal untuk input data citra yang akan diklasifikasikan. Pada gambar 4.12 Pada gambar diatas dapat dilihat menu untuk browser files yang akan digunakan untuk input gambar kemudian setelah gambar yang ingin diklasifikasikan diinput, user harus menekan tombol ”Prediksi Penyakit” sehingga halaman akan berubah seperti gambar 4.13 Berikut:



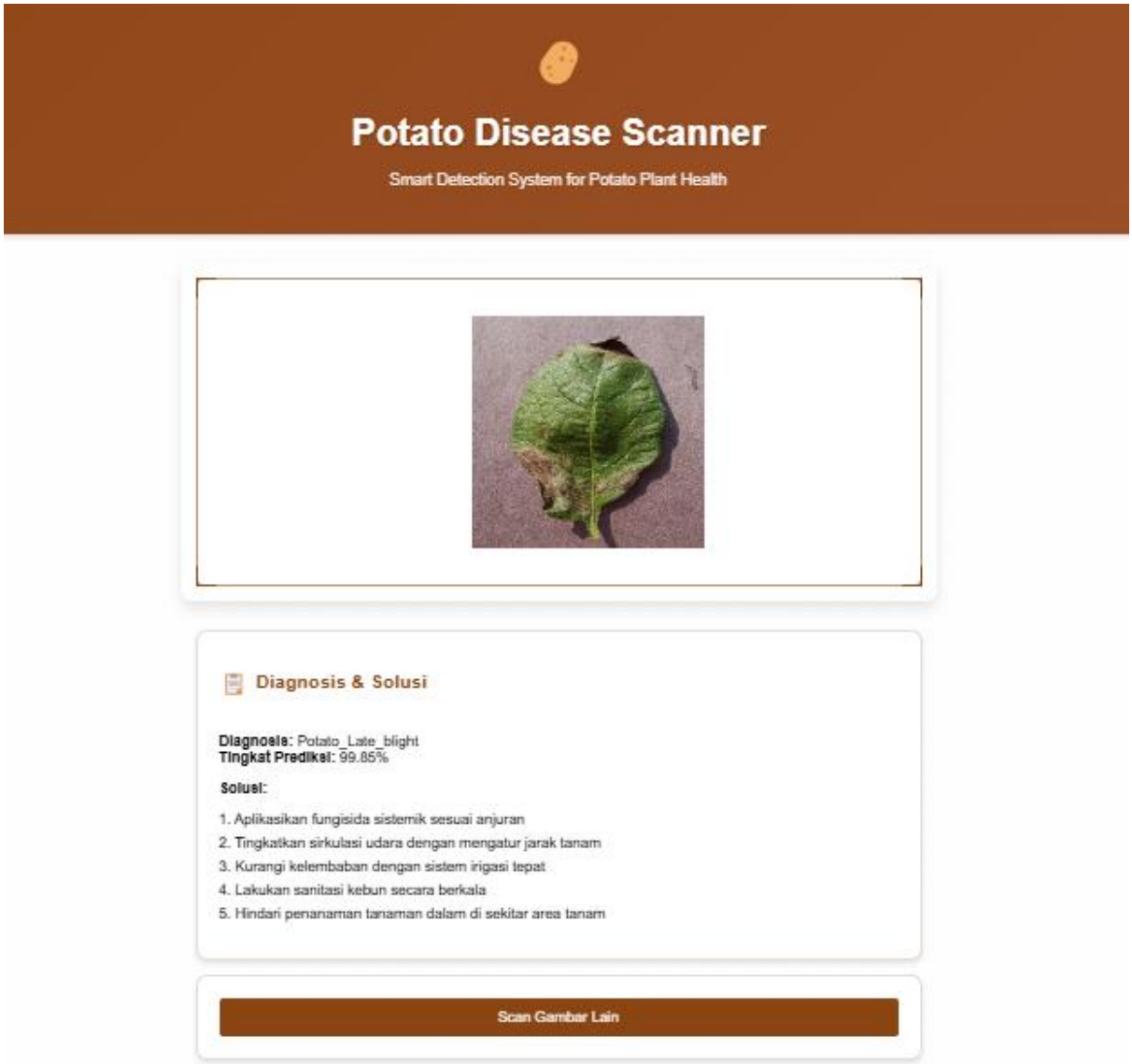
Gambar 4. 13 Hasil Proses Klasifikasi Kentang Early Blight

Pada gambar diatas hasil klasifikasi citra daun kentang tersebut adalah daun kentang yang terkena penyak early blight dengan akurasi 95.87% berikut dengan solusi perawatannya. Untuk klasifikasi citra daun yang sehat dapat dilihat pada gambar 4.14 berikut:



Gambar 4. 14 Hasil Proses Klasifikasi Kentang Healty

Pada gambar 4.14 hasil klasifikasi citra daun kentang tersebut adalah daun kentang yang healty dengan akurasi 96.06% berikut dengan solusi perawatannya. Untuk klasifikasi citra daun late blight dapat dilihat pada gambar 4.15 berikut:



Gambar 4. 15 Hasil Proses Klasifikasi Kentang Late Blight

Pada gambar diatas hasil klasifikasi citra daun kentang tersebut adalah daun kentang late blight dengan akurasi 99.85% berikut dengan solusi perawatannya.

4.3 Hasil Pengujian Model CNN

Pada skenario ini adalah membandingkan nilai akurasi pada saat menggunakan oprimizer adam,RMSprop, dan SGD dengan epoch 15, 25 dan 50 dengan convolution layer menggunakan 7 layer. Sehingga total pengujian yang dilakukan pada penelitian ini sebanyak 15 kali. Tujuan dari penentuan paramenter model ini ingin membandingkan

model yang memiliki akurasi terbaik dengan memperhatikan nilai paramenternya. Serta mengetahui apakah dengan dilakukannya penambahan epoch, optimizer serta mengubah layer konvolusi dapat meningkatkan kualitas model CNN dalam mengklasifikasikan citra daun kentang. Keseluruhan hasil dapat dilihat pada tabel 4.2 Berikut:

Tabel 4. 2 Hasil Pngujian Model CNN

No	Optimizer	Epoch	Convolution Layer	Akurasi
1	Adam	15	4	87,83%
2	Adam	15	5	91,82%
3	Adam	25	4	91,08%
4	Adam	25	5	91,62%
5	**Adam**	**50**	**4**	**92,70%**
6	Adam	50	5	90,15%
7	RMSprop	15	4	86,73%
8	RMSprop	15	5	85,50%
9	RMSprop	25	4	85,77%
10	RMSprop	25	5	87,96%
11	RMSprop	50	4	87,09%
12	RMSprop	50	5	87,11%
13	SGD	15	4	20,56%
14	SGD	15	5	20,56%
15	SGD	25	4	20,56%
16	SGD	25	5	47,75%
17	SGD	50	4	20,56%
18	SGD	50	5	20,56%

Dari 18 percobaan yang telah dilakukan, didapatkan kombinasi terbaik yaitu pada pengujian ke 2 dengan menggunakan *Optimizer* Adam, dengn epoch 50, dan menggunakan 4 dan 5 layer konvolusi. Akurasi yang didapatkan sama yaitu 92,70%, akurasi ini adalah yang paling tinggi. Opimizer RMSProp memberikan akurasi yang cukup baik dengan akurasi tertinggi 87,96% pada epoch 25 dan 5 convolution layer. Akurasi dengan Optimizer SGD adalah yang paling rendah dengan akurasi maksimal 47,75% pada epoch 25 dan convolution layer, sedangkan sebagian besar lainnya hanya mencapai 20,56%.

Berdasarkan kombinasi parameter yang teah diujikan pada tabel 4.9 dapat disimpulkan bahwa epoch yang lebih tinggi (50) cenderung meningkatkan akurasi pada adam dan RMSProp, namun peningkatannya tidak signifikan setelah 25 epoch. Dapat dilihat pada

pengujian ke 1 dengan epoch 15 mendapatkan akurasi 87,83%, pengujian ke 3 dengan epoch 25 mendapatkan akurasi 91,08%, pengujian ke 5 dengan epoch 50 mendapatkan akurasi 92,70%.

Jumlah convolution layer tidak selalu berbanding lurus dengan akurasi, namun pada adam, 5 convolution layer cenderung memberikan hasil lebih baik dibandingkan 4 convolution layer. Adam adalah optimizer terbaik dalam eksperimen ini, diikuti oleh RMSprop, sementara SGD memiliki performa yang sangat buruk.

4.4 Hasil Penelitian

Penelitian ini menyimpulkan bahwa, parameter terbaik yang dapat digunakan pada perancangan model CNN untuk klasifikasi cira daun kentang ditampilkan pada tabel 4.3 berikut:

Tabel 4. 3 Kombinasi Optimal Hyperparameter dan Optimizer

Optimizer	Epoch	Convolutional Layer	Akurasi
Adam	50	4	92,70%

Sehingga dari keseluruhan percobaan pertama hingga akhir, kombinasi hyperparameter dan optimizer yang memberikan hasil paling optimal adalah kombinasi pada percobaan 5 seperti pada tabel diatas, untuk mendapatkan nilai akurasi dapat digunakan confusion matrix seperti tabel 4.4 Berikut:

Tabel 4. 4 Confusion Matrix Model optimal

Matriks	Prediksi Kelas		
	Early Blight	Healthy	Late Blight
Early Blight	100	0	20
Healthy	0	96	0
Late Blight	0	0	58

Berdasarkan tabel 4.4 hasil klasifikasi dari model menunjukkan hasil yang baik. Klasifikasi terhadap citra daun kentang berpenyakit Early Blight diklasifikasikan ke dalam eraly blight sebnyak 100 data dan data yang gagal menjadi late blight sebanyak 20 data citra. Klasifikasi pada citra daun kentang tak berpenyakit (Healthy diklasifikasikan sebagai Healthy sebanyak 96 data. Kemudia yang terakhir adalah klasifikasi terhadap

citra daun kentang berpenyakit Late Blight diklasifikasikan benar kedalam late bligh sebanyak 58. Perhitungan akurasi dari keseluuhan matrix diatas apabila dihitung menggunakan persamaan berikut:

$$Akurasi = \frac{Jumlah\ Prediksi\ Benar}{Jumlah\ Prediksi\ Salah} = \frac{254}{274} = 0{,}9270 = 92{,}70\%$$

4.5 Hasil Pengujian Sistem

dari hasil pengembangan sistem diatas, selanjutnya dilakukan uji cba format file dan uji remove background pada sistem. Hasilnya dapa dilihat pada tabel 4.5 Dan 4.6 Berikut:

Tabel 4. 5 Hasil pengujian format file pada sistem

Data Testing	Perkiraan Hasil	Hasil	Kesimpulan
Daun sehat.jpg	True	True	Success
Daun berpenyakit.png	True	True	Success
Daun sehat.jpeg	True	True	Success
Daun berpenyakit.jpg	True	True	Success
Daunlain.pdf	False	False	Success

Tabel 4. 6 Hasil pengujian remove background

Data Testing	Perkiraan Hasil	Hasil	Kesimpulan
DaunBG.jpg	True	True	Success
DaunNoBG.png	True	True	Success

Dataset dipilih secara acak kemudia di augmentasi, dataset yang dipilih dipilih dimasukkan masing-masing 7 gambar untuk setiap penyakit, yang terdiri dari 4 gambar memiliki background dan 3 lainnya tidak memiliki background. Proses augmentasi database testing yang dilakukan pada penelitian ini merotasi gambar, melakukan penambahan brighness 40%, dan menurunkan brightness 40%. Setelah proses augmentasi selesai kemudia data di klasifikasi. Penelitian mencatat seiap prediksi yang benar dan prediksi yang salah, kemudian hasil tersebut dikumpulkan dan digunakan untuk menghitung nilai akurasi pada setiap tabel pengujian. Hasil dari proses klasifikasi dataset yang sudah di augmentasi dengan rotasi 40 derajat dapat dilihat pada tabel 4.7 berikut:

Tabel 4. 7 Citra Daun Kentang dalam Pengujian

No	Jenis Penyakit	Normal	Rotasi 40°	Brightness +40	Brightness -40
1	Early Blight	93.77%	61.5%	56.81%	93.86%
2	Early Blight	53.97%	99.42%	74.23%	97.61%
3	Early Blight	99.52%	55.81%	96.21%	50.81%
4	Early Blight	97.68%	89.34%	99.63%	91.1%
5	Early Blight	99.99%	99.91%	99.99%	58.47%
6	Early Blight	65.17%	97.16%	96.21%	86.89%
7	Early Blight	99.64%	94.78%	99.78%	81.96%
8	Healthy	80.65%	71.18%	98.02%	71.43%
9	Healthy	71.69%	98.99%	76.03%	64.76%
10	Healthy	97.54%	65.6%	53.52%	94.7%
11	Healthy	97.18%	98.76%	95.05%	94.77%
12	Healthy	62.32%	76.41%	54.49%	95.95%
13	Healthy	96.18%	97.68%	92.95%	53.59%
14	Healthy	93.31%	52.58%	83.21%	99.14%
15	Late Blight	99.78%	99.18%	97.54%	99.82%
16	Late Blight	98.19%	99.97%	83.04%	99.46%
17	Late Blight	98.39%	99.16%	88.84%	99.81%
18	Late Blight	78.09%	99.71%	74.31%	95.26%
19	Late Blight	99.16%	99.15%	99.12%	99.28%
20	Late Blight	99.68%	99.95%	98.06%	99.81%
21	Late Blight	99.71%	99.6%	99.64%	98.6%
Akurasi		100%	66,66%	100%	38%

Berdasarkan hasil pengujian menghasilkan informasi pada tabel 4.7. Arti warna hijau pada tabel yaitu berarti hasil klasifikasi berhasil dan yang berwana merah berarti gagal. Untuk menentukan akurasi pada klasifikasi citra daun kentang yaitu:

1. Merotasi gambar 40 derajat

   Akurasi : 14/21 * 100%= 66,66%

   Akurasi dari pengujian merotasi citra 40 derajat yaitu sebesar 66,66%

2. Menambahkan brightness 40%

Akurasi: 21/21 * 100%= 100%

Akurasi dari pengujian pada tabel 4. Sebesar 100%

3. Menurunkan brightness 40%

   Akurasi: 8/21 * 100% = 38%

   Akurasi dari pengujian pada tabel 4. Sebesar 38%

4.6 Pembahasan

Berdasarkan hasil dari pengujian kombinasi hyperparameter pada setiap model CNN yang dirancang. Setiap Hyperparameter dan Optimizer yang diujikan memberi pengaruh yang besar pada model yang dihasilkan. Kombinasi paling optimal terjadi pada percobaan ke 5 dengan kombinasi Optimizer Adam, 4 Convolution Layer, dan 50 epochs menghasilkan tingkat akurasi tertinggi 92,70%. Akurasi terendah pada percobaan ke 13,14,15,17 dan 18 dengan kombinasi Optimizer SGD, 4 dan 5 Convolution Layer, 15, 25 dan 50 epochs.

Berdasarkan hasil pengujian sistem, variasi rotasi dan brightness digunakan dalam proses augmentasi data untuk meningkatkan kinerja model CNN dalam mendeteksi penyakit pada tanaman kentang. Dasar pemilihan variasi ini adalah untuk meningkatkan keragaman data latih, sehingga model dapat lebih robust terhadap perubahan posisi dan pencahayaan pada gambar input. Disimpulkan bahwa perubahan yang terjadi pada citra daun kentang mempengaruhi ketepatan dari proses klasifikasi,yaitu dengan merotasi citra, melakukan perubahan brightness,dan melakukan shear range. Pengujian 21 kali menggunakan rotasi 40° mendapatkan akurasi 71%. Pengujian 21 kali penambahan brightness 40% mendapatkan akurasi 100% dan pengurangan brightness 40% mendapatkan akurasi 38%. Tujuan utama dari augmentasi ini adalah untuk mencegah overfitting, yaitu ketika model hanya belajar dari pola tertentu dalam dataset tanpa mampu mengenali variasi baru dari citra yang serupa.