

KLASIFIKASI PENYAKIT EARLY BLIGHT DAN LATE BLIGHT PADA TANAMAN TOMAT BERDASARKAN CITRA DAUN MENGGUNAKAN METODE CNN BERBASIS WEBSITE

Nining Putri Ningsih^{1*}, Emi Suryadi², Lalu Darmawan Bakti³, Bahtiar Imran⁴

^{1,2,3,4}Universitas Teknologi Mataram

Email : ^{1*}niningputriningsih2018@gmail.com, ²emisuryadi@gmail.com, ³lp2mumtm@gmail.com,
⁴bahtiarimranlombok@gmail.com

(naskah masuk : 12 desember 2022, direvisi : 12 desember 2022, dipublikasikan : 13 desember 2022)

Abstrak

Tomat merupakan salah satu tanaman hortikultura di Indonesia yang sangat rentan terserang penyakit. Petani akan mengalami kesulitan untuk mengidentifikasi penyakit pada daun tanaman tomat, jika hanya dilihat secara kasat mata saja. Hal tersebut dapat menyebabkan kesalahan dalam penanggulangannya, sehingga dapat menyebabkan turunnya hasil produksi serta memungkinkan terjadinya gagal panen pada tanaman tomat. Oleh karena itu dibutuhkan aplikasi yang membantu petani untuk mengklasifikasi Penyakit Early Blight dan Late Blight pada daun tomat. Proses klasifikasi ini menggunakan citra daun dengan metode Convolutional Neural Network. Dataset yang digunakan 4.000 citra dengan 2 jenis penyakit yaitu Early Blight dan Late Blight. Penggunaan Algoritma CNN menghasilkan akurasi yang tinggi, proses training data menggunakan learning rate 0,0001 dan batch size 20. Epoch 1 menghasilkan loss 98%, akurasi 53%, Recall 46%. Epoch 10 menghasilkan 20, loss 34%, akurasi 85%, recall 81%. Epoch 20 menghasilkan loss 22%, akurasi 94%, recall 95%. Epoch 100 menghasilkan loss 5%, akurasi 99%, dan recall 85%, akan digunakan untuk proses klasifikasi karena menghasilkan akurasi dan recall yang tinggi, serta loss yang kecil. Model CNN tersebut akan diimplementasikan ke website dengan menggunakan framework flask.

Kata kunci: Citra, CNN, deploy web, Machine Learning, Penyakit tomat

CLASSIFICATION OF EARLY BLIGHT AND LATE BLIGHT DISEASES ON TOMATO PLANTS BASED OF LEAF IMAGERY USING A WEBSITE BASED CONVOLUTIONAL NEURAL NETWORK METHOD

Abstract

Tomato is one of the horticultural crops in Indonesia which is very susceptible to disease. Farmers will find it difficult to identify diseases on tomato plant leaves, if only seen with the naked eye. This can cause errors in the handling, so that it can cause a decrease in production yields and allow crop failure in tomato plants. Therefore we need an application that helps farmers to classify Early Blight and Late Blight on tomato leaves. This classification process uses leaf images with the Convolutional Neural Network method. The dataset used is 4,000 images with 2 types of disease, namely Early Blight and Late Blight. The use of the CNN Algorithm produces high accuracy, the training data process takes a learning rate of 0.0001 and a batch size of 20. Epoch 1 produces 98% loss, 53% accuracy, and 46% recall. Epoch 10 produces 20, 34% loss, 85% accuracy, 81% recall. Epoch 20 resulted in 22% loss, 94% accuracy, 95% recall. Epoch 100 produces a loss of 5%, accuracy of 99%, and recall of 85%, will be used for the classification process because it produces high accuracy and recall, as well as small loss. The CNN model will be implemented on the website using the flask framework.

Keywords: Citra, CNN, deploy web, Machine Learning, Tomato disease

1. PENDAHULUAN

Masuknya penyakit ke tanaman tomat diperlukan untuk menghentikan kerusakan tambahan karena tomat merupakan salah satu tanaman hortikultura di Indonesia yang sangat rentan terhadap penyakit. Penyakit tanaman tomat dapat disebabkan oleh bakteri, jamur, atau virus[1]. Secara umum, penyakit tanaman tomat sebagian besar mempengaruhi daun, dan perubahan warna dan bentuk daun dapat mengungkapkan gejala penyakit. Jika hanya terlihat dengan mata telanjang, infeksi tanaman tomat akan sulit dideteksi oleh petani. Hal ini dapat mengakibatkan kesalahan penanganan yang menurunkan hasil produksi dan meningkatkan kemungkinan gagal panen tanaman tomat [2].

Jaringan saraf convolutional (CNN) digunakan dalam konstruksi model pembelajaran mesin studi kasus ini. Salah satu teknik machine learning untuk mengolah data dua dimensi adalah metode CNN. Karena kedalaman tingkat jaringan dan aplikasinya yang luas dalam data gambar, CNN adalah semacam jaringan saraf yang dalam. Untuk proses klasifikasi dan pengenalan, teknik CNN sering digunakan [3].

2. METODE PENELITIAN

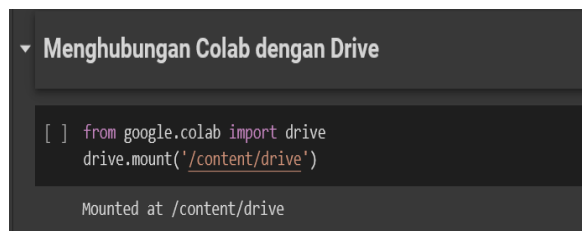
Penelitian ini menggunakan metodologi yang mencakup studi literatur, data acquisition, pre-processing, klasifikasi CNN, deployment dan dokumentasi.

2.1 Studi Literatur

Hal ini dilakukan untuk memudahkan peneliti dalam mengembangkan dan menemukan referensi baru yang sesuai dengan bidang yang diteliti.

2.2 Data Acquisition

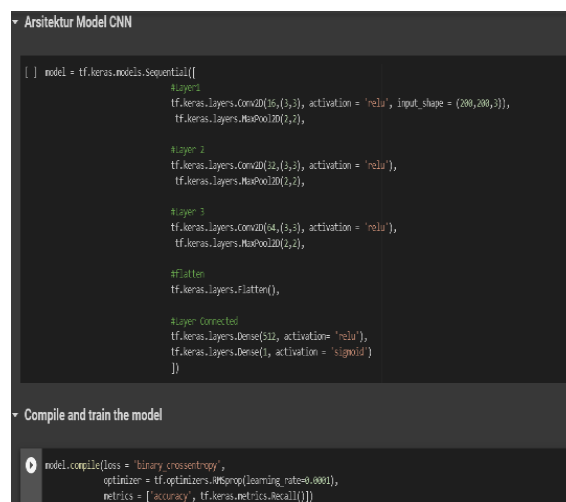
Memperoleh data melibatkan pengumpulan informasi tentang mata pelajaran tertentu[4]. Secara keseluruhan, 4.000 foto digunakan untuk analisis ini, dengan 80% digunakan untuk pelatihan, 20% untuk validasi, dan 50 sisanya sebagai data uji. Google Drive adalah tempat data pergi setelah didistribusikan. Kemudian, gunakan perintah panggilan untuk menautkan drive ke Google Collab, seperti yang diilustrasikan pada Gambar 1.



Gambar 1. Data Acquisition

2.3 Data Pre-Processing

Data Pre-Processing merupakan proses perbaikan data dan atau mempersiapkan data tersebut agar dapat di proses oleh algoritma[4]. Data yang telah dibagi kemudian peneliti melakukan preprocessing dataset dengan normalisasi menggunakan fungsi Rescale 1/255 untuk mengubah skala citra sehingga didapatkan rentang nilai dari 0-1. Target size atau ukuran gambar yang digunakan untuk proses training adalah 200 x 200. Batch size yang digunakan adalah 20, Sedangkan class mode digunakan binary. Setelah gambar di pre-processing, kode akan di run dan akan muncul jumlah dataset. Jika data set yang muncul sesuai jumlahnya dengan yang di upload pada drive berarti tidak ada gambar yang noise atau rusak.



Gambar 2. Data Pre-Processing

2.4 Klasifikasi CNN

Convolutional Neural Network (CNN) pembuatan multilayer perceptron (MLP) yang dioptimalkan untuk memproses foto sebagai data dua dimensi[3]. Karena kedalaman jaringan yang besar dan arsitektur berlapis-lapis, CNN termasuk dalam jaringan saraf tiruan[5].

```

Image preparation for CNN Image Classifier

[ ] # Normalize: all images will be rescaled by 1./255
train = ImageDataGenerator(rescale= 1./255)
valid = ImageDataGenerator(rescale= 1./255)
test = ImageDataGenerator(rescale= 1./255)

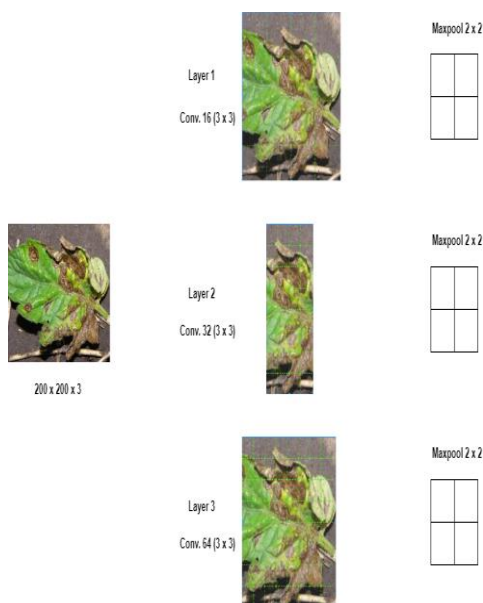
[ ] train_dataset = train.flow_from_directory('/content/drive/MyDrive/Dataset_Tomat/Training',
                                             target_size = (200,200),
                                             batch_size = 20,
                                             class_mode = 'binary')
validation_dataset = train.flow_from_directory('/content/drive/MyDrive/Dataset_Tomat/Valid',
                                              target_size = (200,200),
                                              batch_size = 20,
                                              class_mode = 'binary')

test_dataset = train.flow_from_directory('/content/drive/MyDrive/Dataset_Tomat/Test (uji)',
                                         target_size = (200,200),
                                         batch_size = 20,
                                         class_mode = 'binary')

Found 3200 images belonging to 2 classes.
Found 800 images belonging to 2 classes.
Found 50 images belonging to 2 classes.
    
```

Gambar 3. Arsitektur Model CNN

Gambar 3 diatas merupakan gambar arsitektur model CNN. Penggunaan model sekuensial membuat proses klasifikasi berurutan dari proses input sampai output. Pada fitur ekstraksi diatas terdapat layer 1, 2, dan 3. Pada layer tersebut memiliki masing-masing Conv2D yang berbeda parameter didalamnya. Pada layer 1 memiliki parameter 16, layer 2 memiliki parameter 32 dan layer 3 parameternya 64, parameter tersebut merupakan nilai dari filter. Selanjutnya parameter 3 x 3 merupakan ukuran filter dan ketiga, fungsi aktivasi yang digunakan Rectifier Function (ReLU). Parameter ke- 4 pada layer 1 adalah 200 x 200 x 3 nilai dari input shape. 200 x 200 merupakan bentuk data dan 3 merupakan channel RGB (Read, Green, Blue) yang digunakan untuk melatih citra. Pada Maxpool2D, pooling matrix yang digunakan 2 x 2 tersebut digunakan untuk menghilangkan pixel minimum dimana fitur dibagikan. Flatten merubah array 2 dimensi sebelumnya menjadi 1 dimensi dan diteruskan ke fully connected. Dense menambahkan layer ke fully connected. Parameter pertama di dalamnya 512 merupakan node yang harus ada di hidden layer dan aktivasi yang digunakan relu. Layer output hanya memiliki satu node karena termasuk klasifikasi binary. Pada proses kompilasi parameter loss menggunakan binary cross entropy untuk menentukan loss. Parameter optimizer menggunakan RMSprop dengan learning rate 0,0001 dan parameter metrics untuk menampilkan metrics. Berikut tahap feature extraction pada proses klasifikasi dengan model CNN:



Gambar 4. Ilustrasi feature extraction

Berdasarkan Gambar 4 diatas dapat dipaparkan sebagai berikut:

1. Tahap pertama menginput gambar dengan dimensi 200 x 200 x 3. Tahap selanjutnya gambar yang dipecah kecil-kecil tersebut akan dimasukkan ke small neural network dengan ukuran filter yang sama yaitu 3 x 3.
2. Stride yang digunakan diatas yaitu 2 dengan max polling 2 x 2 yang berarti filter akan bergeser sebanyak 2 x 2.
3. Selanjutnya padding atau zero padding yang digunakan adalah 0. Penggunaan zero padding bertujuan untuk memanipulasi dimensi output dari conv. layer (feature maps).

Berikut cara menghitung dimensi dari feature maps:

$$\text{Output} = \frac{W + N + 2P}{S} + 1$$

Penjelasannya:

W = Panjang / tinggi input

N = Panjang / tinggi filter

P = Padding / zero padding

S = Stride

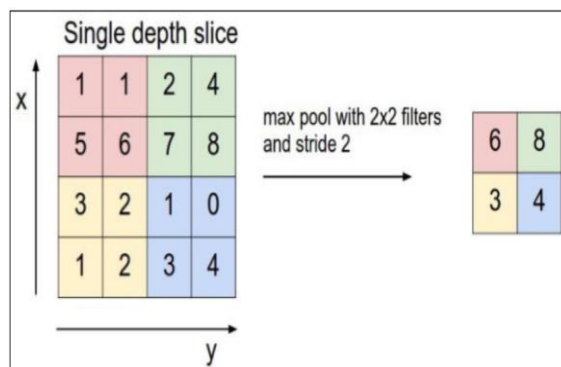
$$\text{Output} = \frac{200 + 200 + 2.0}{2} + 1$$

$$\text{Output} = \frac{401}{2} = 200,5$$

Dimensi input yang digunakan adalah 200 x 200 dan dimensi output nya adalah 200,5.

4. Polling Layer, tahap ini dilakukan setelah convolution. Max polling yang digunakan yaitu 2 x 2 dengan stride 2. Lapisan polling bekerja untuk mengurangi ukuran pada feature maps.

Berikut gambar contoh ilustrasi max polling:



Gambar 5. Contoh Ilustrasi max polling

5. Selanjutnya, lanjutkan ke lapisan yang sepenuhnya terhubung, di mana peta fitur dan hasil polling kemungkinan maksimum akan dipautkan ke jaringan saraf di lapisan FC. Anda harus meratakan peta fitur yang dihasilkan oleh ekstraksi fitur karena masih dalam bentuk array multidimensi dan tidak dapat digunakan sebagai input untuk lapisan yang sepenuhnya terhubung. Sebelum dikirim ke semua neuron di lapisan yang sepenuhnya terhubung, aktivitas dari lapisan sebelumnya perlu diubah ke keadaan satu dimensi.
6. Fungsi aktivasi sigmoid, untuk proses klasifikasi untuk menentukan kelas saat proses output apakah terdeteksi 0 atau 1 (early blight atau late blight).

2.5 Deployment

Pada penelitian ini, peneliti melakukan deploy web dengan menggunakan framework flask. Gambar 6. dan Gambar 7 dibawah ini merupakan kode pada tahap deploy web di framework flask.

```
1 from flask import Flask, render_template, request, send_from_directory
2 from tensorflow.keras.models import load_model
3 from tensorflow.keras.preprocessing.image import img_to_array, load_img
4 from tensorflow.keras.applications import vgg16
5 import numpy as np
6 import os
7
8 app = Flask(__name__)
9 app.config['UPLOAD_FOLDER'] = './static/uploads'
10 model = load_model('classification_model.h5')
11
12 class dict = {0: 'early blight', 1: 'late blight'}
13 gejala = {0: ''
14           1: 'Gejala early blight terdapat bercak kecil berwarna coklat yang dapat meluas dan menyebabkan daun berlobang, <
15           2: 'Berkas dan akar-akar menjadi layu dan layu yang menunjukkan ...',
16           3: '...selisir terapan pada daun terlihat bercak coklat hingga hitam, <
17           4: 'Analisa mengapa yang dan foto daun, kemudian action ke seluruh permukaan dan bagasi ke target daun.'
18           }
19
20 def predict_label(img_path):
21     loaded_img = load_img(img_path, target_size=(224, 224))
22     img_array = img_to_array(loaded_img) / 255.0
23     img_array = np.expand_dims(img_array, 0)
24     prediction = np.argmax(model.predict(img_array), axis=-1)
25     return class_dict[prediction]
26
27 def predict_label(img_path):
28     loaded_img = load_img(img_path, target_size=(224, 224))
29     img_array = img_to_array(loaded_img) / 255.0
30     img_array = np.expand_dims(img_array, 0)
31     prediction = np.argmax(model.predict(img_array), axis=-1)
32     return class_dict[prediction]
```

Gambar 6. Kode Deploy Flask

```
14
15
16 @app.route('/')
17 def main():
18     return render_template("index.html")
19
20 @app.route("/about")
21 def about():
22     return render_template("about.html")
23
24 @app.route("/help")
25 def help():
26     return render_template("help.html")
27
28 @app.route("/upload", methods = ['GET', 'POST'])
29 def get_output():
30     if request.method == 'POST':
31         img = request.files['my_image']
32
33         img_path = 'static/' + img.filename
34         img.save(img_path)
35
36         p = predict_label(img_path)
37         n = predict_label(img_path)
38         return render_template("about.html", prediction = p, img_path = img_path, gejala = n )
39
40 @app.route("/display/cnn")
41 def display_cnn():
42     return send_from_directory(app.config['UPLOAD_FOLDER'], filename)
43
44 if __name__ == '__main__':
45     app.run(debug=True)
```

Gambar 7. Kode Deploy Flask

Pada Gambar 6 kode deploy tersebut pada line 1 sampai 7 merupakan pemanggilan library yang sebelumnya digunakan di collab untuk menjalankan kode python pada model CNN. Line 8 sampai 10 merupakan keyword dari flask dimana di isi dengan pemanggilan folder dan pemanggilan model CNN yang save h5. Sedangkan line 12 sampai 18 kode untuk gejala yang terjadi jika tanaman terserang 2 penyakit tersebut. Line 20 sampai 33 merupakan kode fungsi logika untuk memanggil prediksi yang dilakukan oleh model CNN. Sedangkan pada gambar 3.7 merupakan kode logika yang menghubungkan flask dengan desain html yang dibuat sebelumnya sehingga model CNN yang dibuat dapat diimplementasikan dalam bentuk website dan mempermudah user dalam menggunakannya.

2.6 Dokumentasi

Pada tahap ini peneliti melakukan dokumentasi dengan menulis laporan hasil penelitian serta hasil dari pembuatan aplikasi.

3. HASIL DAN PEMBAHASAN

3.1 Training

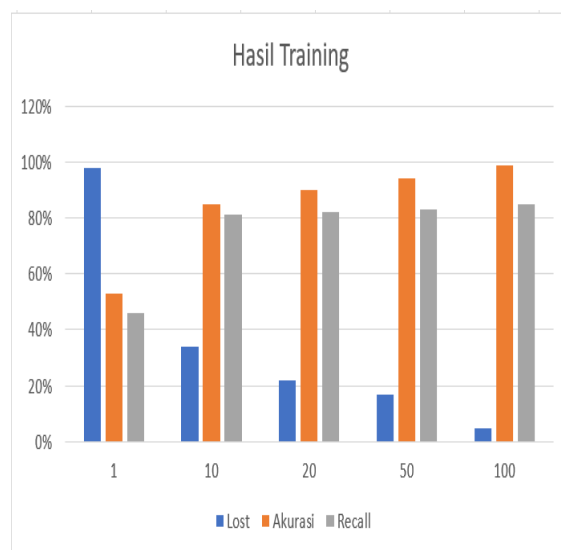
Tujuan dari prosedur pembelajaran (atau pelatihan) adalah untuk menyesuaikan model CNN yang dikembangkan untuk mengenali dan mengkategorikan gambar daun [6].

Tabel 1. Hasil Training Model CNN

Epoch	Learning Rate	Batch Size	Lost	Akurasi	Recall
1	0.0001	20	98%	53%	46%
10	0.0001	20	34%	85%	81%
20	0.0001	20	22%	90%	82%
50	0.0001	20	17%	94%	83%
100	0.0001	20	5%	99%	85%

Pada tabel 1 diatas merupakan tabel hasil training model CNN. Epoch 1 dengan learning rate 0,0001 dan batch size 20 akan menghasilkan loss 98%, akurasi 53%, Recall 46%. Epoch 10 dengan learning rate 0,0001 dan batch size menghasilkan 20, loss 34%, akurasi 85%, recall 81%. Epoch 20 dengan learning rate 0,0001 dan batch size 20 menghasilkan loss 22%, akurasi 94%, recall 95%. Epoch 100 dengan learning rate 0,0001 dan batch size 20 menghasilkan loss 5%, akurasi 99%, dan recall 85%.

Untuk grafik dari tabel 3.1 tersebut dapat dilihat pada gambar 8 grafik hasil training model. Pada grafik dibawah tertera tiga warna pertama loss berwarna biru, kedua akurasi dengan warna orange dan terakhir dengan warna abu adalah recall. Berdasarkan hasil training tersebut Epoch 100 dengan learning rate 0,0001 dan batch size 20 menghasilkan loss 5%, akurasi 99%, dan recall 85%, akan digunakan untuk proses klasifikasi karena menghasilkan akurasi dan recall yang tinggi, serta loss yang kecil.



Gambar 8. Grafik Hasil Training

3.2 Testing

Pengujian terjadi pada akhir proses penelitian. Untuk memastikan kategorisasi seakurat mungkin, indeks model CNN yang dilatih dievaluasi [6].

Data testing yang digunakan sebanyak 50 citra dengan rincian masing-masing kelas 25 citra. Citra uji yang di input akan dilakukan preprocessing oleh sistem, setelahnya citra tersebut akan menuju feature ekstraksi, kemudian ke flatten, dari flatten array sebelumnya dijadikan citra satu dimensi untuk diteruskan ke layer connected dan ke layer output disana binary akan bekerja untuk menentukan kelas apakah 0 atau 1. Kemampuan model CNN dalam membaca data testing dapat dilihat pada Gambar 9 berikut ini.

```
[ ] model.evaluate(test_dataset)

3/3 [=====] - 5s 3s/step - loss: 0.6404 - accuracy: 0.8000
[0.6403847336769104, 0.800000011920929]
```

Gambar 9. Model Evaluasi Data Testing

3.3 Hasil Pengujian

Berikut merupakan tabel hasil pengujian penyakit early blight dan late blight pada citra tomat:

Tabel 2 Hasil pengujian penyakit early blight dan late blight

Jenis Penyakit	Citra ke	Keberhasilan Uji
Early Blight	1	Berhasil
	2	Berhasil
	3	Berhasil
	4	Berhasil
	5	Berhasil
	6	Berhasil
	7	Berhasil
	8	Berhasil
	9	Berhasil
	10	Berhasil
	11	Tidak Berhasil
	12	Berhasil
	13	Berhasil
	14	Berhasil
	15	Tidak Berhasil
	16	Berhasil
	17	Berhasil
	18	Tidak Berhasil
	19	Berhasil
	20	Tidak Berhasil
	21	Berhasil
	22	Tidak Berhasil
	23	Tidak Berhasil
	24	Berhasil
	25	Tidak Berhasil
Late Blight	26	Berhasil
	27	Berhasil
	28	Berhasil
	29	Berhasil
	30	Tidak Berhasil
	31	Berhasil
	32	Berhasil
	33	Tidak Berhasil
	34	Berhasil
	35	Berhasil
	36	Berhasil
	37	Berhasil
	38	Berhasil
	39	Berhasil
	40	Berhasil
	41	Berhasil
	42	Berhasil
	43	Berhasil
	44	Tidak Berhasil
	45	Berhasil
	46	Berhasil
	47	Berhasil
	48	Berhasil
	49	Berhasil
	50	Berhasil

Setelah pengujian dilakukan, evaluasi akurasi adalah langkah selanjutnya. Perhitungan akurasi dilakukan untuk mengetahui persentase keberhasilan model CNN dalam mengklasifikasikan buah jeruk. Akurasi juga dapat digunakan sebagai tolok ukur untuk membandingkan model atau mengembangkan sistem CNN baru di masa mendatang[6].

Berikut adalah prosedur untuk menentukan akurasi dengan menggabungkan data.

$$\text{Akurasi} = \frac{\text{Jumlah Data Benar}}{\text{Jumlah Data Uji}} \times 100\%$$

Berdasarkan Tabel 2 diatas jumlah data benar adalah 40 citra dengan jumlah data uji yang digunakan 50 citra dan berdasarkan rumus diatas hasil akurasi adalah sebagai berikut.

$$\text{Akurasi} = \frac{40}{50} \times 100\% = 80\%$$

Akurasi delapan puluh persen pada 50 set data uji.

TP (True Positive) menunjukkan sistem memprediksi positif dan hasilnya benar; ini adalah hasil kategorisasi yang paling positif. Ketika sistem memprediksi hasil negatif dan hasil itu benar-benar terjadi ('Benar Negatif'), sistem telah bekerja sebagaimana dimaksud; ketika memprediksi hasil positif dan prediksi salah ('False Positive'), dan ketika memprediksi hasil negatif dan prediksi salah ('False Negative'), sistem telah melakukan sebagaimana dimaksud tetapi dengan prediksi yang salah [7].

Tabel 3. Confusion matrix

	PREDICTED	
ACTUAL	1	0
1	True Positive (TP)	False Positive (FP)
0	False Negative (FN)	True Negative (TN)

$$\text{Akurasi} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{TP} + \text{FN}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Dari rumus dan dapat tabel 3 diatas dapat dijabarkan dalam tabel 4 dan hitungan dibawah ini.

Tabel 4. Hasil confusion matrix

	PREDICTED	
ACTUAL	Early Blight	Late Blight
Early Blight	18	7
Late Blight	3	22

$$\text{Akurasi} = \frac{18 + 22}{18 + 22 + 3 + 7} = \frac{40}{50} = 80\%$$

$$\text{Recall} = \frac{18}{18 + 3} = \frac{18}{21} = 85\%$$

Hasil akurasi dan recall dari 50 sampel data uji pada tabel 4 adalah akurasi nya 80% dan recall 85%. Akurasi merupakan hasil ketepatan atau kemiripan data uji dan data sebenarnya (data valid). Sedangkan, data recall adalah ketepatan suatu sistem menemukan kembali suatu informasi.

4. KESIMPULAN

Berdasarkan hasil pengujian dengan metode Convolutional Neural Network (CNN) dalam proses pengolahan citra klasifikasi penyakit pada daun tomat, dapat di simpul kan bahwa pada penelitian ini proses training data menggunakan learning rate 0,0001 dan batch size 20. Penggunaan Algoritma CNN menghasilkan akurasi yang tinggi, proses training data menggunakan learning rate 0,0001 dan batch size 20. Epoch 1 menghasilkan loss 98%, akurasi 53%, Recall 46%. Epoch 10 menghasilkan 20, loss 34%, akurasi 85%, recall 81%. Epoch 20 menghasilkan loss 22%, akurasi 94%, recall 95%. Epoch 100 menghasilkan loss 5%, akurasi 99%, dan recall 85%, akan digunakan untuk proses klasifikasi karena menghasilkan akurasi dan recall yang tinggi, serta loss yang kecil. Proses training ini menggunakan data sebanyak 4.000 citra dan 2 jenis penyakit pada daun tomat. Hasil akurasi

dan recall dari 50 sampel data uji adalah akurasi nya 80% dan recall 85%. Setelah didapatkan hasil akurasi dan recall yang tinggi, serta loss yang kecil pada epoch 100. Peneliti melakukan proses implementasi model CNN ke website dengan framework Flask.

5. REFERENSI

- [1] F. Alviansyah, I. Ruslianto, and M. Diponegoro, "Identifikasi Penyakit Pada Tanaman Tomat Berdasarkan Warna Dan Bentuk Daun Dengan Metode Naive Bayes," *J. Coding Sist. Komput. Untan*, vol. 05, no. 1, pp. 23–32, 2017.
- [2] M. Astiningrum, P. P. Arhandi, N. A. Ariditya, and A. T. Tomat, "Berdasarkan Fitur Warna dan Tekstur," pp. 227–230.
- [3] T. NURHIKMAT, "IMPLEMENTASI DEEP LEARNING UNTUK IMAGE CLASSIFICATION MENGGUNAKAN ALGORITMA CONVOLUTIONAL NEURAL NETWORK (CNN) PADA CITRA WAYANG GOLEK," vol. 7, no. 2, pp. 44–68, 2018.
- [4] P. Choirunisa, "Implementasi Artificial Intelligence Untuk Memprediksi Harga Penjualan Rumah Menggunakan Metode Random Forest dan Flask (Tugas Akhir)," 2020.
- [5] H. Hambali, M. Mahayadi, and ..., "Classification of Lombok Songket Cloth Image Using Convolution Neural Network Method (Cnn)," *Pilar Nusa Mandiri ...*, no. 85, pp. 149–156, 2021, doi: 10.33480/pilar.v17i2.2705.
- [6] F. F. Maulana and N. Rochmawati, "Klasifikasi Citra Buah Menggunakan Convolutional Neural Network," *J. Informatics Comput. Sci.*, vol. 1, no. 02, pp. 104–108, 2020, doi: 10.26740/jinacs.v1n02.p104-108.
- [7] Felix, S. Faisal, T. F. M. Butarbutar, and P. Sirait, "Implementasi CNN dan SVM untuk Identifikasi Penyakit Tomat via Daun," *Issn 2622-8130*, vol. 20, no. 2, pp. 117–134, 2019.