DETEKSI HAMA PADA DAUN TEH DENGAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN)

Bagja Hidayat[1], Galih Hermawan[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur No 112-116 Bandung, 40132
E-mail : bagjahidayat48@gmail.com[1], galih.hermawan@email.unikom.ac.id[2]

ABSTRAK

Produksi dan pengembangan komoditas teh merupakan salah satu *core business* perkebunan PTPN VIII. Salah satu penyebab turunnya produksi teh di Indonesia adalah serangan hama, ada dua jenis hama, yaitu hama *blister blight* dan hama *hellopeltis*. Dalam penelitian ini dapat diperoleh melalui empat tahap, yaitu pertama tahap *preprocessing* menggunakan metode *grayscale, edge detection (Sobel)* dan *Standardization*, kedua tahap *feedforward* menggunakan metode *convolutinal layer, relu, max pooling* dan *fully connected layer*, ketiga tahap *backpropagation* dan keempat tahap *confusion matrix*. *Convolutional Neural Network* (CNN) adalah metode klasifikasi yang akan digunakan didalam penelitian ini, CNN merupakan variasi dari *Multilayer Perceptron* (MLP) yang terinsfirasi dari jaringan saraf manusia. Tujuan yang ingin dicapai yaitu, *convolutional neural network* dapat mendeteksi yang baik dalam mengenali jenis hama pada daun teh. Setelah melakukan pengujian sistem dengan metode *Black Box* dapat ditarik kesimpulan bahwa program berfungsi dengan baik. Pada proses pengujian dari data pengenalan yang telah disiapkan program mampu mendeteksi jenis hama pada daun teh dengan rata-rata akurasi sebesar 95%.

Kata Kunci : deteksi jenis hama, *convolutional neural network*, pengenalan pola, daun teh, *multilayer perceptron*

1. PENDAHULUAN

Hama pada tanaman teh merupakan pengganggu atau masalah yang sangat meresahkan bagi perkebunan. Hama-hama ini akan menghambat dan dapat mengganggu sekali pertumbuhan tanaman yang sedang dibudidayakan, termasuk pada daun.

Adapun dari penelitian sejenis yaitu Klasfikasi Jenis Pohon Mangga Gadung dan Curut Berdasarkan Tekstur Daun [1], *Clasification Of Diseased Plant Leaves Using Neural Network* Algorithms [2], *Classification and Recognition of Herbal Leaf Using SVM Algorithm* [3] dan Deteksi Cacat Daun Teh Camellia Sinensis dengan Pengolahan Citra Digital dan JST *Learning Vector Quantization* [4].

Dari penelitian sebelumnya belum adanya deteksi jenis hama pada daun teh menggunakan dengan metode *convolutional neural network*.

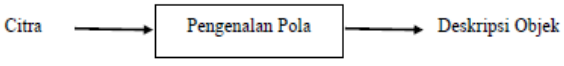
CNN adalah salah satu jenis *neural network* yang buasa digunakan pada data image. CNN bisa digunakan untuk mendeteksi dan mengenali object pada sebuah image. Secara garis besar CNN tidak jauh beda dengan *neural network* biasanya. CNN terdiri dari *neuron* yang memiliki *weight*, bisa dan *activation function*. Jaringan CNN ini memiliki lapisan khusus yang dinamakan dengan lapisan konvolusi dimana pada lapisan ini sebuah citra masukkan akan diolah berdasarkan filter yang sudah ditentukan. Dari setiap lapisan ini akan menghasilkan sebuah pola dari beberapa bagian citra yang nantinya akan lebih mudah untuk diklasifikasi. Teknik ini dapat membuat fungsi pembelajaran citra menjadi lebih efisien untuk diimplementasikan.

Berdasarkan uraian di atas dalam penelitian ini untuk mengenali suatu objek dimana metode *convolutional neural network* cocok untuk mengenali jenis hama pada daun teh sehingga pada tugas akhir ini model CNN akan digunakan sebagai pendeteksian jenis hama pada daun teh.

2. ISI PENELITIAN

2.1 Pengenalan pola

Pengenalan pola, yaitu pengklompokan data numerik dan simbolik (termasuk citra) secara otomatis oleh computer [5]. Tujuan pengklompokan adalah untuk mengenali suatu objek di dalam citra. Computer menerima masukkan berupa citra objek yang akan didentifikasi, memproses citra tersebut dan memberikan keluaran berupa deskripsi objek di dalam pengenalan pola. Gambar 1 menunjukan pengenalan pola.



Gambar 1. Pengenalan Pola

2.2 Analisis data masukan

Data masukan yang digunakan adalah data citra daun yang diambil dari kebun teh berlokasi di Malabar Pangalengan Kabupaten Bandung milik PT. Perkebunan Nusantara VIII. Data masukan berupa citra daun terserang hama, yaitu hama *blister blight* dan hama *hellopeltis*. Jumlah data citra 440 data citra, 400 dijadikan data *training* yang terbagi 200 citra

daun yang terserang hama *blister blight* dan 200 citra daun yang terserang ham *hellopeltis*, sedangkan 40 data citra dijadikan data *testing* yang terbagi 20 citra daun hama *blister blight* dan 20 citra daun hama *hellopeltis*. Ukuran *pixel* dalam penelitian ini yaitu, 28x28 *pixel*. Contoh citra daun yang terserang oleh hama dapat dilihat pada gambar 2 dan gambar 3.

a. *Blister blight*



Gambar 2. Citra Daun Hama *Blister blight*

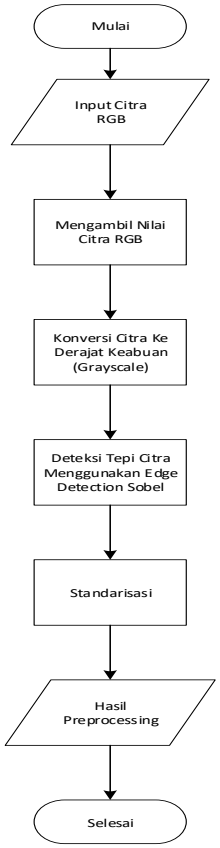
b. *Hellpeltis*



Gambar 3. Citra Daun Hama *Hellopeltis*

2.3 *Preprocessing*

Pada tahap ini citra data masukkan diproses dari citra RGB ke *grayscale*, kemudian dirubah ke *edge detection* (*sobel*) dan *standardization*. Keluaran dari proses ini yang nantinya digunakan pada tahap selanjutnya yakni *feedforward*. Tahapan proses ini dapat dilihat pada gambar 4.



Gambar 4. *Flowchart Preprocessing*

2.3.1 *Grayscale*

Dari perhitungan RGB ke *grayscale*, nilai *pixel* dari 28x28 diambil sampel 8x8 dapat dilihat pada tabel 1.

Tabel 1. Nilai *Pixel* Citra RGB

xy	1	2	3	...	8
1	R = 93 G = 131 B = 39	R = 97 G = 135 B = 41	R = 92 G = 121 B = 26	...	R = 94 G = 136 B = 25
2	R = 102 G = 139 B = 51	R = 108 G = 141 B = 54	R = 103 G = 119 B = 35	...	R = 101 G = 139 B = 27
3	R = 112 G = 148 B = 61	R = 116 G = 151 B = 66	R = 100 G = 137 B = 44	...	R = 108 G = 143 B = 29
...	...	...	...	...	...
8	R = 103 G = 140 B = 46	R = 105 G = 141 B = 44	R = 91 G = 131 B = 23	...	R = 105 G = 143 B = 32

Contoh peritungan kita ambil pixel (1,1) dengan rumus sebagai berikut.

$$(x, y) = 0.299.R + 0.587.G + 0.114.B \quad (1)$$

Maka dapat diilustrasikan sebagai berikut:

(1,1) = (0,2989*R)+(0,5870*G) + (0,1141*B)
= (0,2989*93)+(0,5870*131)+(0,1141 * 39)
= 27,7977 + 76,897 + 4,4499
= 109,1446 ≈ **109,14**

Maka *grayscale* secara keseluruhan dapat dilihat pada tabel 2.

Tabel 2. Hasil Nilai *Pixel Grayscale*

xy	1	2	3	4	5	6	7	8
1	109,14	112,92	101,49	109,57	128,90	132,73	118,86	110,78
2	117,90	121,21	104,63	99,93	124,73	130,56	115,29	114,66
3	127,31	130,84	115,33	99,35	111,99	122,64	112,79	119,53
4	132,37	132,67	123,64	106,58	102,01	114,02	121,76	115,65
5	135,89	133,25	117,69	111,41	103,60	111,75	123,36	103,32
6	135,14	132,32	114,05	105,84	106,38	117,77	111,76	103,32
7	125,08	125,57	109,18	103,38	112,29	117,18	108,34	113,85
8	118,22	119,17	106,72	106,16	119,18	102,34	100,84	118,98

2.3.2 *Edge detection* (*Sobel*)

Dari hasil *graycscale* maka dalam tahap ini *grayscale* dirubah ke dalam *edge detection* (*Sobel*) dengan rumus sebagai berikut.

Kernel Sobel:

$$x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (2)$$

Rumus:

$$x = ((1,3) + 2(2,3) + (3,3)) - ((1,1) + 2(2,1) + (3,1)) \quad (3)$$

$$y = \big((3,1) + 2(3,2) + (3,3)\big) - ((1,1) + 2(1,2) + (1,3)) \quad (4)$$

Gradien:

$$G = \sqrt{x^2 + y^2} \quad (5)$$

Keterangan:

G = Gradien Sobel

Contoh:
x = (127,3129+2*130,8400+115,3294) -
(109,1446+2*112,9164+101,4924) = 67,8525
y = (101,4924+2*104,6332+115,3294) -
(109,1446+2*117,8999+127,3129) = -46,1691
$G = \sqrt{67{,}8525^2 + -46{,}1691^2} = 82{,}0704 \approx$ **82,07**
Kernel sobe; akan digeser 1 *stride* keseluruh permukaan *pixel grayscale* sehingga menghasilkan matriks *edge detection*. Hasil keseluruhan dapat dilihat pada table 3.

Tabel 3. Hasil *Pixel Edge Detection*

xy	1	2	3	4	5	6
1	82,07	85,09	68,43	107,70	67,54	58,03
2	72,53	123,79	12,61	101,13	50,75	20,49
3	50,19	107,24	63,32	41,57	63,18	10,36
4	66,66	98,38	57,87	23,25	64,68	41,34
5	83,81	102,39	30,74	40,79	26,95	43,08
6	81,87	88,26	12,60	37,23	35,73	22,12

2.3.3 *Standardization*

Proses ini untuk merubah atau menstandarkan tingkat intensitas cahaya pada citra, citra dipetakan pada pixel dengan ukuran tertentu sehingga memberikan representasi dimensi yang tetap. Maka dapat diilustrasikan dengan rumus sebagai berikut.

$$s_j = \frac{x_j - \bar{x}}{\sigma} \quad (6)$$

Dimana:
x_j= data ke-j
$\bar{x}$ = rata – rata
σ = standar deviasi
Rata – rata:

$$\bar{x} = \frac{(1,1)+(2,1)+(3,1)+\cdots+(n,n)}{n} \quad (7)$$

Rumus untuk menghitung standar deviasi sebagai berikut:

$$\sigma = \sqrt{\frac{\sum_{j=1}^{n}(x_j - \bar{x})^2}{n-1}} \quad (8)$$

Keterangan:
σ : Standar Deviasi
x_j : nilai x ke-j
$\bar{x}$: Rata-rata
n : ukuran sampel

1. Cari nilai rata -rata.

$$\bar{x} = \frac{82{,}07+85{,}09+68{,}43+107{,}70+67{,}54+\cdots+22{,}12}{36}$$
$$= \frac{2143{,}7821}{36} = 59{,}5495 \approx 59{,}55$$

2. Cari nilai standar deviasi.

$$s_1 = \frac{82{,}0704 - 59{,}5495}{30{,}5921} = \mathbf{0{,}7362}$$

Maka hasil dari keseluruhan standarisasi dapat dilihat pada tabel 4.

Tabel 4. Hasil *Pixel Standardization*

xy	1	2	3	4	5	6
1	0,74	0,83	0,29	1,57	0,26	-0,05
2	0,42	2,10	-1,53	1,36	-0,29	-1,28
3	-0,31	1,56	0,12	-0,59	0,12	-1,61
4	0,23	1,27	-0,05	-1,19	0,17	-0,60
5	0,79	1,40	-0,94	-0,61	-1,07	-0,54
6	0,73	0,94	-1,53	-0,73	-0,78	-1,22

Hasil dari tahap *preprocessing* dapat dilihat pada gambar 5.

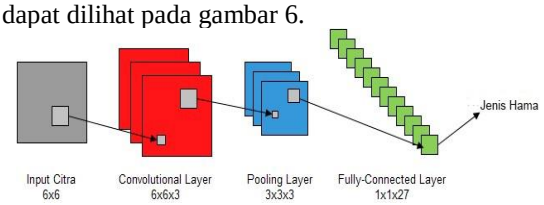


Gambar 5. Hasil *Preprocessing*

2.4 *Convolutional Neural Network*

CNN adalah variasi dai *multilayer perceptron* (MLP) yang terinspirasi dari jaringan syaraf manusia. Penelitian awal yang mendasari penemuan ini pertama kali oleh Hubel dan Wiesel [6]. *Convolutional neural network* merupakan suatu *layer* yang memiliki susunan *neuron* 3D (lebar, tinggi, kedalaman). Lebar dan tinggi merupakan ukuran *layer*, sedangkan kedalaman mengacu pada jumlah *layer*.

Dengan demikian CNN merupakan metode untuk mentransformasikan gambar original *layer* ke *layer* dari nilai *pixel* gambar kedalam nilai skoring kelas untuk klasifikasi. Analisis yang digunakan dalam implementasi *convolutional neural netwok* dapat dilihat pada gambar 6.



Gambar 6. Analisis *Convolutional Neural Network*

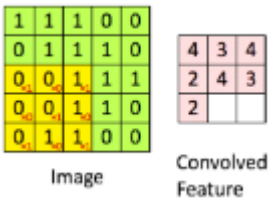
2.5 Arsitektur Jaringan CNN

Pada Tahap ini akan dilakukan proses CNN dari awal masukan melewati *convolution layer*, fungsi aktivasi (*relu*), *pooling* dan *fully connected layer* hingga dihasilkan sebuah *vector one-hot* klasifikasi jenis hama.

2.5.1 *Convolution layer*

Pada *convolution layer* akan dilakukan operasi konvolusi antara matriks citra masukan dengan matriks *kernel*. *Kernel* ini akan digeser keseluruh permukaan gambar. Tujuan dilakukannya konvolusi pada citra adalah untuk mengekstraksi fitur dari citra *input*. Konvolusi akan menghasilkan tranformasi *linear* dari data *input* sesuai informasi spasial pada data. Bobot pada *layer* tersebut menspesifikasikan

kernel konvolusi yang digunakan, sehingga *kernel* konvolusi dapat dilatih berdasarkan *input* pada CNN [8].



Gambar 7. Operasi Konvolusi

2.5.2 Fungsi Aktivasi *ReLU*

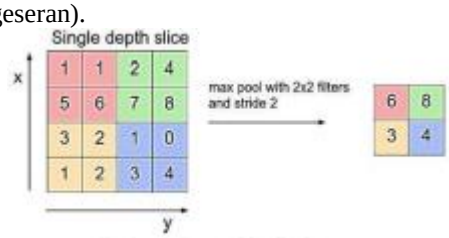
Fungsi aktivasi adalah fungsi *non linear* yang memungkinkan sebuah JST untuk dapat mentransformasi data *input* menjadi dimensi yang lebih tinggi sehingga dapat dilakukan, pemotongan *hyperline* sederhana yang memungkinkan dilakukan klasifikasi. Dalam CNN terdapat fungsi aktivasi digunakan yaitu fungsi *ReLU*.

Fungsi aktivasi *ReLU*, dimana hasil dari konvolusi setiap *pixel* yang memiliki nilai < 0 akan dijadikan 0 dan yang memiliki nilai ≥ 0 nilai tersebut ditetapkan, sehingga fungsi aktivasi *relu* memiliki bentuk persamaan sebagai berikut:

$$f(x) = max(0, x) \quad (9)$$

2.5.3 *Pooling (Max pooling)*

Max pooling membagi output dai *relu* menjadi beberapa *grid* kecil lalu mengambil nilai maksimal dari setiap *grid* untuk menyusun matriks citra yang telah direduksi seperti yang ditunjukan pada gambar 8. *Grid* yang berwarna merah, hijau, kuning dan biru merupakan kelompok *grid* yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan *grid* disebelah kanannya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi (pergeseran).

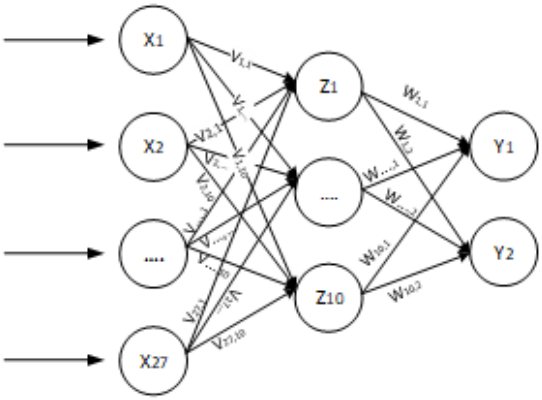


Gambar 8. Operasi *Max Pooling*

2.5.4 *Fully Connected Layer*

Layer tersebut adalah *layer* yang biasanya digunakan dalam penerapan MLP dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasi secara *linear*.

Setiap *neuron* pada *convolution layer* perlu ditranformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan ke dalam sebuah *fully connected* hanya dapat diimplementasikan di akhir jaringan. Gambar 9 adalah arsitektur *fully connected layer* yang digunakan dalam analisis ini.



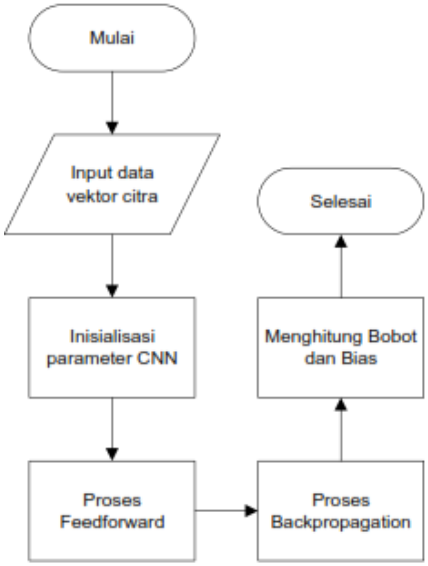
Gambar 9. Arsitektur *Fully Connected Layer*

Keluaran yang dihasilkan adalah *vector one-hot* jenis hama, uang berisi citra masukkan jenis hama [7]. Karen jenis hama yang digunakan dua buah (*blister blight* dan *hellopeltis*), maka *vector one-hot* yang digunakan memiliki dua komponen, yaitu y_1 dan y_2, dimana y_1 jenis hama *blister blight* dan y_2 jenis hama *hellopeltis*.

2.6 *Proses Training*

Proses *training* merupakan tahapan dimana CNN dilatih untuk memperoleh akurasi yang tinggi dari klasifikasi yang dilakukan. Tahapanini terdiri dari proses *feedforward* dan proses *backpropagation*. Untuk memulai proses *feedforward* diperlukan jumlah dan ukuran *layer* yang akan dibentuk, dimana citra vektor akan melalui proses konvolusi dan *Max pooling* untuk mereduksi ukuran citranya dan memperbanyak neuronnya. Sehingga terbentuk banyak jaringan yang mana menambah variant data untuk dipelajari [8].

Hasil dari proses *feedforward* berupa bobot yang akan digunakan untuk mengevaluasi proses *neural network* tadi. Alur prosesnya seperti pada gambar 10.



Gambar 10. Alur Proses *Training*

2.6.1 Proses *Feedforward*

Proses *feedforward* merupakan tahap pertama dalam proses *training*. Proses ini akan menghasilkan beberapa lapisan untuk mengklasifikasi data citra yang mana menggunakan bobot dan bias yang telah diperbarui dari proses *backpropagation*. Tahap ini juga akan digunakan kembali saat proses *testing*.

2.6.2 Proses *Backpropagation*

Proses *backpropagation* merupakan tahap kedua dari proses *training*. Hasil proses dari *feedforward* di-*trace* kesalahannya dari lapisan *output* sampai lapisan pertama. Untuk menandai bahwa data tersebut telah di-*trace* diperoleh bobot dan bias yang baru.

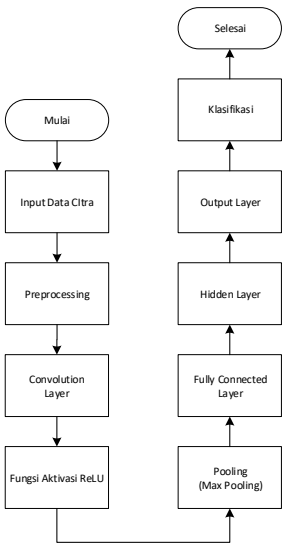
2.6.3 Perhitungan *Gradient*

Pada proses *gradient* untuk jaringan konvolusi merupakan proses untuk memperoleh nilai bobot dan bias baru yang akan diperlukan saat *training* dan proses *testing*.

2.7 Proses Testing

Proses *testing* merupakan proses untuk memperoleh nilai bobot dan bias dari hasil proses *training*. Proses ini tidak jauh berbeda dengan proses *training* yang membedakannya tidak terdapat proses *backpropagation* setelah proses *feedforward*. Sehingga hasil akhir dari proses ini menghasilkan akurasi dari klasifikasi yang dilakukan, data yang gagal diklasifikasi, nomor citra yang gagal diklasifikasi dan bentuk network yang terbentuk dari proses *feedforward*.

Dengan bobot dan bias yang baru proses *feedforward* diterapkan yang kemudian menghasilkan lapisan *output*. Lapisan *output* sudah *fully connected* dengan label yang disediakan. Hasil *fully connected* tersebut diperoleh data yang gagal dan berhasil diklasifikasi. Dari penjelasan diatas bentuk alur proses *testing* berbentuk seperti pada gambar 11.



Gambar 11. Alur Proses *Testing*

2.8 Pengujian

Pengujian system adalah bagian penelusuran dalam rangka menemukan kesalahan pada system yang dibangun. Tujuan pengujian sistem adalah untuk mencari kesalahan pada sistem yang dibangun sebelum sistem digunakan oleh pengguna. Pengujian sistem yang dilakuakan meliputi tahap pengujian fungsionalitas, pengujiain detail perancangan dan pengujian performansi.

2.8.1 Pengujian Performasi

Pengujian performansi merupakan tahap yang memiliki tujuan untuk mengetahui performa atau tingkat akurasi dari hasil metode *Convolutional Neural Network* (CNN) dengan cara menghitung jumlah data *training* yang kelasnya diprediksi secara benar. Cara mengukur kinerja dari sitem akan dilakukan evaluasi dengan menggunakan *confusion matrix*.

Confusion matrix adalah matriks yang berisi data aktual dan data prediksi yang telah diklasifikasikan oleh sistem [9]. Pada penelitian ini, pengujian performansi dilakukan dengan menggunakan 1 metode dengan ketentuannya dapat dilihat pada tabel 5.

Tabel 5. Ketentuan Pengujian *Confusion Matriks*

Actual	Predict	
	0	1
0	TP	FN
1	FP	TN

Keterangan:
0 = Positif
1 = Negatif
TP dan TN memberikan informasi ketika klasifikasi benar, sedangkan FP dan FN memberitahu ketika klasifikasi salah.

2.8.2 Menguji data masukan yang tidak termasuk ke dalam data *Training*

Pengujian ini dilakukan dengan menguji data yang tidak termasuk ke dalam data *pembelajaran*, pengujian ini bertujuan untuk mengetahui tingkat pengenalan data *pengenalan* terhadap data *pembelajaran* yang telah dipilih. Data *pembelajaran* yang digunakan sebanyak 400 data, 200 data *hellopeltis* dan 200 data *blister blight* sedangkan data *pengenalan* sebanyak 40 data, 20 data *hellopeltis* dan 20 data *blister blight*. Untuk *leraning rate* ditetapkan 0,01 dan *momentum* 0,9 dengan *epoch* yang berbeda-beda. Ketentuan pengujian performansi data pengenalan dapat dilihat pada tabel 6.

Tabel 6. Ketentuan Performansi Data *Testing*

No	Nama File	Prediksi	Fakta	Status
1	blister_1.jpg	0 / *Positive*	0 / *Positive*	TP
2	blister_2.jpg	1 / *Negative*	0 / *Positive*	FN
3	hellopeltis_1.jpg	0 / *Positive*	1 / *Negative*	FP
4	hellopeltis_2.jpg	1 / *Negative*	1 / *Negative*	TN

2.8.2.1 Pengujian performansi data *testing* pertama

Pada pengujian ini menggunakan 400 data *training* dan 40 data *testing* dengan epoch = 5. Adapun pengujian performansi data *testing* pertama dapat dilihat pada tabel 7.

Tabel 7. Confusion Matrix Data *Testing* Pertama

Predict	*Actual*	
	0	1
0	TP = 9	FP = 0
1	FN = 11	TN = 20

Dari tabel 7 *confusion matrix* data *testing* pertama di atas kemudian dapat juga diukur tingkat akurasi dari prediksi dengan menggunakan persamaan sebagai berikut.

$$Accuracy = \frac{coreectly\ predict\ data}{total\ testing\ data} \times 100\% \quad (9)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (10)$$

$$Accuracy = \frac{9+20}{9+20+0+11} \times 100\%$$

$$= \frac{29}{40} \times 100\% = 75\ \%$$

2.8.2.2 Pengujian performansi data *testing* kedua

Pada pengujian ini menggunakan 400 data *training* dan 40 data *testing* dengan epoch = 10. Adapun pengujian performansi data *testing* kedua dapat dilihat pada tabel 8.

Tabel 7. Confusion Matrix Data *Testing* Pertama

Predict	*Actual*	
	0	1
0	TP = 0	FP = 0
1	FN = 20	TN = 20

Dari tabel 8 *confusion matrix* data *testing* pertama di atas kemudian dapat juga diukur tingkat akurasi dari prediksi dengan menggunakan persamaan sebagai berikut.

$$Accuracy = \frac{coreectly\ predict\ data}{total\ testing\ data} \times 100\%$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%$$

$$Accuracy = \frac{0+20}{0+20+0+20} \times 100\%$$

$$= \frac{20}{40} \times 100\% = 50\ \%$$

2.8.2.3 Pengujian performansi data *testing* ketiga

Pada pengujian ini menggunakan 400 data *training* dan 40 data *testing* dengan epoch = 15. Adapun pengujian performansi data *testing* kedua dapat dilihat pada tabel 9.

Tabel 7. Confusion Matrix Data *Testing* ketiga

Predict	*Actual*	
	0	1
0	TP = 18	FP = 6
1	FN = 2	TN = 14

Dari tabel 9 *confusion matrix* data *testing* pertama di atas kemudian dapat juga diukur tingkat akurasi dari prediksi dengan menggunakan persamaan sebagai berikut.

$$Accuracy = \frac{coreectly\ predict\ data}{total\ testing\ data} \times 100\%$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%$$

$$Accuracy = \frac{18+14}{18+14+6+2} \times 100\%$$

$$= \frac{32}{40} \times 100\% = 80\ \%$$

2.8.2.4 Pengujian performansi data *testing* keempat

Pada pengujian ini menggunakan 400 data *training* dan 40 data *testing* dengan epoch = 20. Adapun pengujian performansi data *testing* kedua dapat dilihat pada tabel 10.

Tabel 10. Confusion Matrix Data *Testing* Keempat

Predict	*Actual*	
	0	1
0	TP = 20	FP = 2
1	FN = 0	TN = 18

Dari tabel 10 *confusion matrix* data *testing* pertama di atas kemudian dapat juga diukur tingkat akurasi dari prediksi dengan menggunakan persamaan sebagai berikut.

$$Accuracy = \frac{coreectly\ predict\ data}{total\ testing\ data} \times 100\%$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%$$

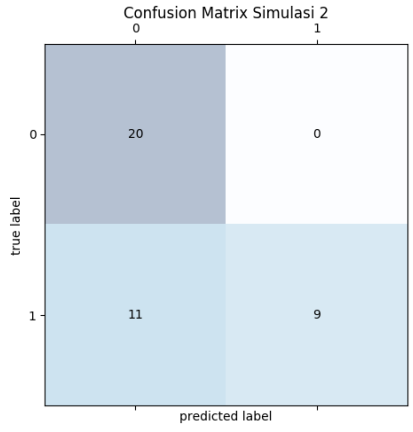
$$Accuracy = \frac{20+18}{20+18+2+0} \times 100\%$$

$$= \frac{38}{40} \times 100\% = 95\ \%$$

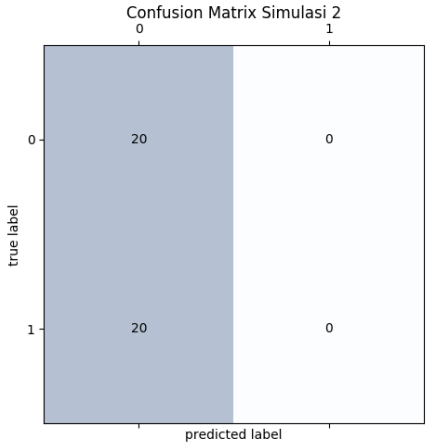
2.8.3 Pengujian Simulator

Berikut merupakan pengujian yang dilakukan terhadap data *testing*. Pengujian yang dilakukan untuk melihat keberhasilan dan keakuratan dalam deteksi jenis hama pada daun teh pada simulator.

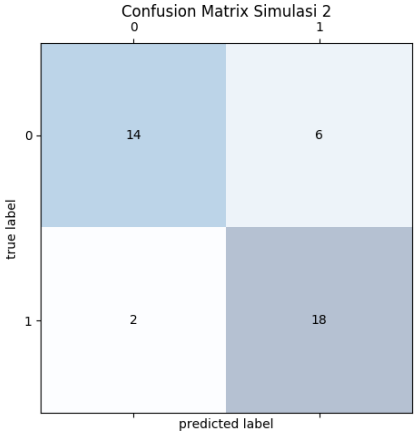
Pengujian pada tahap ini merupakan hasil prediksi dari data *testing*. Prediksi didapat dari hasil klasifikasi dengan metode *convolutional neural network* (CNN). Simulator pengujian *testing* pertama sampai *testing* keempat dapat dilihat pada gambar 12, 13, 14 dan 15.



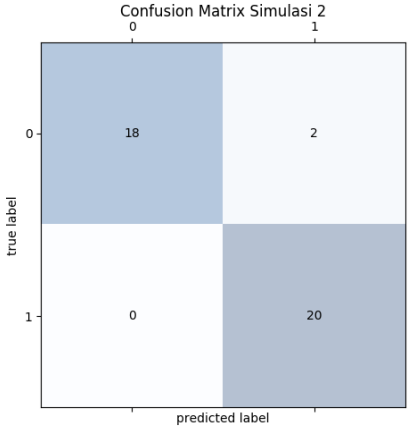
Gambar 12. Pengujian *Testing* Pertama



Gambar 13. Pengujian *Testing* Kedua



Gambar 14. Pengujian *Testing* Ketiga



Gambar 15. Pengujian *Testing* Keempat

2.8.4 Kesimpulan Pengujian Performansi dan Simulator

Kesimpulan dari pengujian simulasi yang dilakukan menunjukkan bahwa simulasi yang dibuat untuk menghasilkan prediksi jenis hama pada daun teh, dari preprocessing, klasifikasi dengan metode *convolutional neural network* (CNN) dan prediksi dari hasil klasifikasi CNN, sudah memenuhi persyaratan fungsional, akan tetapi pada prosesnya masih memungkinkan ditemukan kekurangannya.

Untuk tahapan penerapan proses-proses dan implementasi metode *convolutional neural network* dalam mendeteksi jenis hama pada daun teh. Membuktikan bahwa proses-proses dan metode yang diterapkan dapat berjalan dan terimplementasi, dengan menghasilkan sebuah prediksi jenis hama dengan akurasi diatas 95% menggunakan epoch = 20. Proses pengujian performansi pertama sampai terakhir bisa dilihat pada tabel 11.

Tabel 11. Hasil Pengujian Performansi

Pengujian	*Epoch*	*Akurasi*
1	**5**	**75%**
2	**10**	**50%**
3	**15**	**80%**
4	**20**	**95%**

3 PENUTUP

Kesimpulan dan saran dari penelitian yang dilakukan dalam implementasi metode *convolutional neural network* (CNN) dalam mendeteksi jenis hama pada daun teh.

3.1 Kesimpulan

Hasil penelitian yang telah dilakukan, dapat diketahui prediksi jenis hama berdasarkan data citra hama *hellopeltis* dan hama *blister blight*. Hasil prediksi berupa pengenalan jenis hama *hellopeltis* dan *blister blight*.

Berdasarkan hasil implementasi dan pengujian, akurasi prediksi jenis hama yang dihasilkan simulator cukup baik dengan nilai akurasi rata-rata 95% dengan epoch = 20. Dari hasil yang didapatkan tersebut, maka dapat disimpulkan bahwa penggunaan metode *convolutional neural network* (CNN) dapat diimplementasikan dalam pengenalan citra dalam mendeteksi jenis hama pada daun teh berdasarkan data citra hama *hellopeltis* dan data citra *blister blight* tersebut, sehingga menghasilkan akurasi prediksi jenis hama yang cukup baik.

3.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, masih terdapat beberapa kajian lebih lanjut yang diperlukan dalam penelitian ini. Adapun saran untuk penelitian lebih lanjut yaitu:

1. Dalam meningkatkan akurasi prediksi, perlu dikaji lebih lanjut mengenai penambahan aspek atau faktor-faktor lain yang mempengaruhi akurasi deteksi jenis hama pada daun teh, seperti faktor data pembelajaran, faktor data pengenalan, dan sebagainya.
2. Perlu dikaji lebih lanjut mengenai metode-metode yang dapat diterapkan untuk meningkatkan akurasi deteksi jenis hama pada daun teh.

DAFTAR PUSTAKA

[1] Agustin, Soffiana dan E. Prasetyo, "Klasifikasi Jenis Pohon Mangga Gadung dan Curut Berdasarkan Tekstur Daun", Sistem Informasi ITS, Tesis 2011.

[2] K. Muthukaman, P. Latha, R. P. Selvi and P. Nisha,"*Classification Of Diseased Plant Leaves Using Neural Network ALgorithms*", *ARPN Journal of Engineering and Applied Scinces*, vol.10, No. 4, 2015.

[3] A. Maheswari., N. Bharathi., P. Neelamegan., and T. Gayathridevi, "*Classification and Recognition of Herbal Leaf Using SVM Algorithm", Research Journal of Pharmaceutical, Biological and Chemical Sciences*, pp. 0975-8585, 2014.

[4] P. Lestari., H. Bambang., dan E. Susatio, "Deteksi Cacat Daun Teh Camellia Sinensis dengan Pengolahan Citra Digital dan JST *Learning Vector Quantization"*. Tugas Akhir. Universitas Telkom, 2011.

[5] A. Camuri dan G. Volpe, "*Gesture-Based Communication in Human-Computer Interaction". Pp. 978-3-540-21072-6*, 2004.

[6] D. Hubel dan T. Wiesel, "*Receptive fields an functional architecture of monkey striate cortex*". *Journal of Physiology*. Tesis 1986.

[7] P. Sadowski, "*Notes on Backpropagation*", https://www.ics.uci.edu/~pjsadows/notes.pdf, Diakses 6 Juli 2018.

[8] I. Wayan Suartika, A. Y Wijaya dan R. Soelaiman, "Klasifikasi Citra Menggunakan *Convolutional Neural Network* (Cnn) pada Caltech 101", Jurnal Teknik ITS, Vol. 5, No. 1, 2016.

[9] R. Sebastian, "*Python Machine Learnig*", pp. 978-1-78355-513-0, September 2015.