BAB II
KAJIAN LITERATUR

2.1 Citra

Pengertian umum citra adalah kombinasi antara titik, garis, bidang dan warna untuk menciptakan suatu imitasi dari suatu objek biasanya objek fisik atau manusia. Adapun pengertian lain yaitu citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek [13]. Citra adalah data dalam format gambar dan biasanya memiliki ukuran citra yang besar, semakin bagus citra yang dihasilkan maka kapasitas citra yang dimiliki semakin besar pula. Di era digitalisasi, banyak orang memberikan informasi melalui gambar karena informasi dengan menggunakan gambar lebih mudah dipahami dan efektif [14]. Citra merupakan salah satu informasi yang diperlukan manusia selain teks, suara dan video. Informasi yang terkandung dalam sebuah citra dapat diinterpretasikan berbeda-beda oleh manusia satu dengan yang lain [15]. Citra dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada citra tersebut [13].

Citra digital adalah citra yang dihasilkan melalui proses digitalisasi citra analog contoh citra digital seperti hasil foto dari kamera digital dan *scanner* [16]. Citra digital adalah gambar 2 dimensi yang dibuat dari analog 2 dimensi yang kontinu menjadi citra melalui proses sampling. Citra analog dibagi menjadi N baris dan M kolom, menjadikanya gambar diskrit. Citra digital adalah citra yang dapat diproses dan disimpan oleh komputer, citra yang terdapat di dalam komputer hanyalah angka yang menunjukkan intensitas pada setiap piksel. Karena berbentuk data numerik, maka citra digital dapat diproses oleh dengan komputer [17].

2.1.1 Jenis Citra

Beberapa jenis citra digital yang sering digunakan adalah citra *biner*, *grayscale* dan warna atau RGB [18].

1. Citra Biner

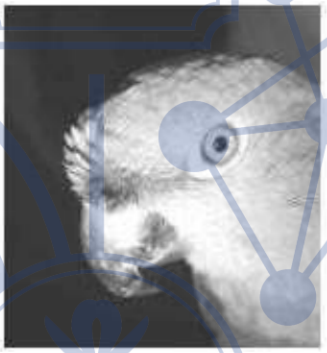
Citra biner (*biner image*) atau citra monokrom merupakan citra yang hanya memiliki dua kemungkinan nilai piksel yaitu hitam dan putih, sehingga disebut juga sebagai citra hitam putih [19,20]. Citra biner (*monochrome*) atau disebut juga *binary image*, merupakan citra digital yang setiap *pixel*-nya hanya memiliki 2 kemungkinan derajat keabuan, yaitu 0

dan 1 [21]. Nilai 0 mewakili warna hitam, dan nilai 1 mewakili warna putih, dimana setiap *pixel*-nya membutuhkan media penyimpanan sebesar 1 bit.

Gambar 2.1 Citra Biner [21]

2. Citra *Grayscale*

Citra *grayscale* adalah citra yang memiliki derajat keabuan sebanyak 256 warna. Umumnya yang digunakan adalah antara warna hitam sebagai warna minimum dan warna putih sebagai warna maksimumnya, sehingga diantara hitam dan putih adalah warna abu-abu [22].



Gambar 2.2 Citra Grayscale [21]

3. Citra RGB

Citra RGB atau citra warna merupakan citra yang menampilkan warna dalam komponen R (*red*), G (*green*), dan B (*blue*) [22]. Setiap warna memiliki masing-masing range 0 – 255, maka totalnya adalah 2553 = 16.581.375 (16 K) variasi warna berbeda pada gambar, dimana variasi warna ini cukup untuk gambar apapun [18].

Gambar 2.3 Citra RGB [21]

2.1.2 Pengolahan Citra

Pengolahan citra digital adalah suatu pengolahan citra yang bertujuan meningkatkan kualitas citra agar lebih mudah dipahami oleh manusia dan komputer. Pengolahan citra digital merupakan metode untuk mengekspos beberapa teknik pada suatu gambar dengan tujuan meningkatkan kualitas gambar atau mendapatkan informasi yang terkandung dalam gambar tersebut [19]. Pengolahan citra merupakan cabang ilmu dalam *Artifical Intelligence* yang menggunakan objek citra dalam bentuk digital untuk penyelesaian kasusnya. Metode dalam citra dapat digunakan baik perhitungan matematis pada objek secara piksel ataupun geometris. Pengolahan citra (*image processing*) memiliki hubungan yang sangat erat dengan disiplin ilmu yang jika sebuah disiplin ilmu dinyatakan dalam bentuk proses suatu input menjadikan *output*, maka pengolahan citra memiliki *input* berupa citra serta *output* berupa citra.

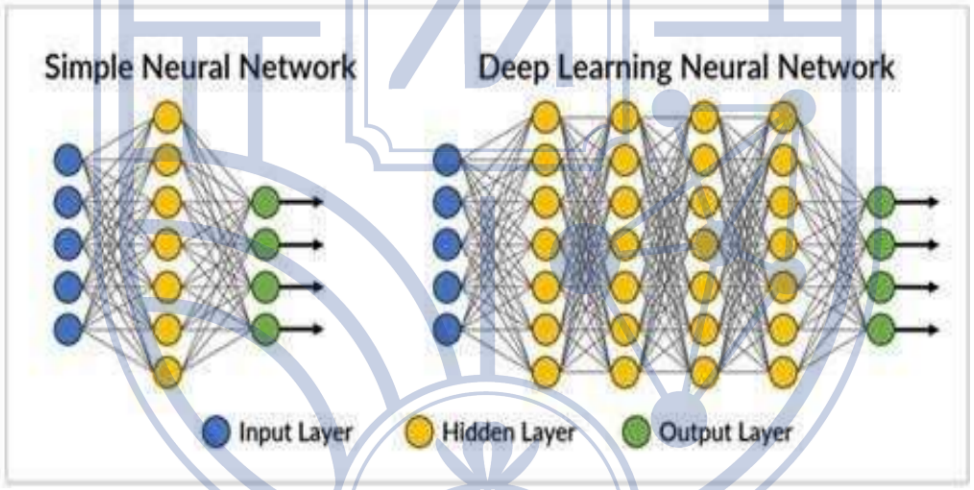
Dalam pengolahan citra digital terdapar 2 proses yaitu input dan output. Hasil dari pengolahan citra digital berupa objek dalam bentuk citra atau gambar. Pengolahan citra digital mempunyai kelebihan seperti tidak merusak objek, pengolahanya cepat dan mudah [19]. Pengolahan citra (*image processing*) adalah proses mengolah citra piksel-piksel di dalam citra digital yang digunakan untuk tujuan tertentu. Awalnya pengolahan citra dilakukan untuk memperbaiki kualitas citra, dengan berkembangnya dunia komputasi yang ditandai dengan semakin meningkatnya kapasitas dan kemampuan komputer memungkinkan manusia dapat mengambil informasi dari suatu citra [15].

2.2 Deep Learning

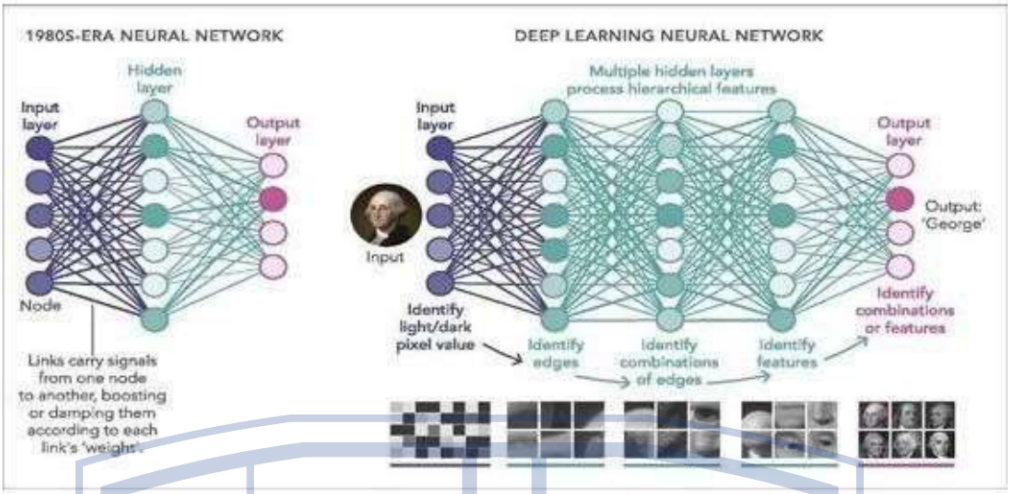
Deep learning adalah area pembelajaran mesin yang menggunakan jaringan syaraf tiruan untuk menyelesaikan masalah dengan kumpulan data besar [23]. Pembelajaran mendalam didefinisikan sebagai sebuah pendekatan yang menekankan pada pemahaman konseptual dan penerapan pengetahuan secara kritis. Untuk banyak aplikasi, model *deep learning* mengungguli

model pembelajaran mesin yang dangkal dan pendekatan analisis data tradisional [24]. Dalam berbagai disiplin ilmu, termasuk keamanan siber, pemrosesan bahasa alami, bioinformatika, robotika dan kontrol, serta pemrosesan informasi medis, *deep learning* telah menggungguli pendekatan pembelajaran mesin yang sudah terkenal [23,24]. *Deep learning* mampu belajar dan beradaptasi menyelesaikan berbagai permasalahan, *deep learning* dapat mempelajari metode komputasinya sendiri, menganalisis data secara terus menerus dengan struktur logika yang mirip dengan bagaimana manusia mengambil keputusan [25].

Selain itu, algoritma *deep learning* berfungsi untuk menjalankan data dengan layer jaringan neural. Jaringan saraf akan mengirimkan data yang sebelumnya disederhanakan ke lapisan berikutnya. Lapisan pertama akan memperoleh kemampuan untuk mengidentifikasi karakteristik halus dalam gambar, sedangkan berikutnya akan mengintegrasikan elemen yang diekstraksi dari lapisan sebelumnya untuk membentuk representasi atau konfigurasi yang lebih halus [26]. Dibawah ini adalah contoh gambar struktur dari *deep learning* seperti contoh Gambar 2.4 dan Gambar 2.5.



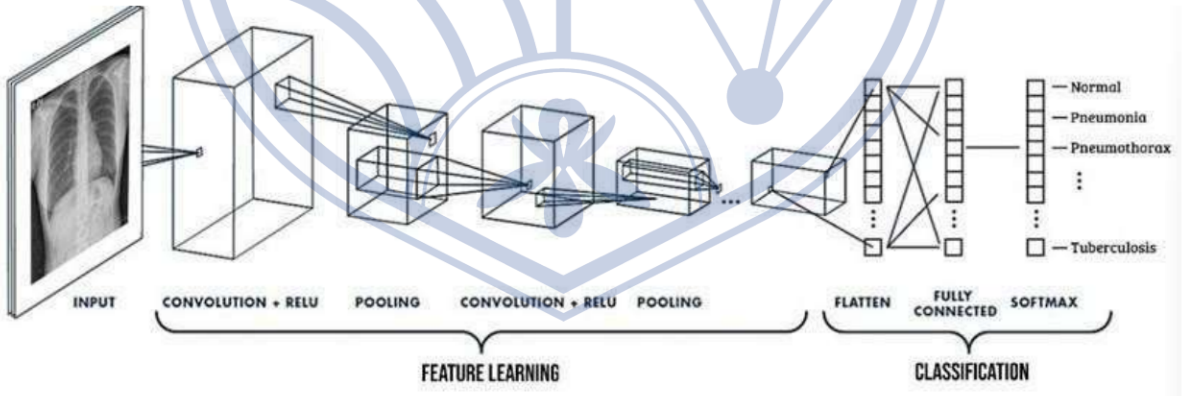
Gambar 2.4 Perbedaan Arsitektur Simple Neural Network dan Arsitektur Deep Learning Neural Network [26]



Gambar 2.5 Contoh Struktur Deep Learning [28]

2.3 CNN

Convolutional Neural Network (CNN) merupakan salah satu model dari *deep learning* yang dirancang khusus untuk mengolah data dua dimensi seperti memproses inputan gambar dan menentukan kepentingan (bobot dan bias yang dapat dipelajari) ke berbagai aspek dalam gambar dan berfungsi untuk membedakan objek satu dengan objek lainnya [27,28]. Model CNN sangat penting dalam membangun pengklasifikasi gambar yang mampu memprediksi dan mengkategorikan gambar secara akurat [26]. CNN terdiri dari lapisan arsitektur, yakni *feature learning* dan *classification layer* [28] seperti terlihat pada Gambar 2.6.



Gambar 2.6 Struktur Arsitektur *Convolutional Neural Network* [29]

Pada bagian *feature learning*, bagian untuk mengekstrak fitur penting dari sebuah *input* gambar secara langsung diawal dan memprosesnya sampai menghasilkan output data [28,29]. Lapisan yang ada pada proses ini terdiri dari lapisan konvolusi dan lapisan *pooling* dan setiap proses lapisan tersebut menghasilkan *feature maps* yang berupa angka-angka.

Sedangkan *classification,* lapisan ini menerima *input* dari *output* bagian *feature learning* yang akan diproses pada *flatten* dengan tambahan beberapa *hidden layer* pada *fully connected* hingga menghasilkan *output* berupa akurasi klasifikasi dari setiap kelas [29]. Lapisan *classification* terdiri dari beberapa lapisan yang berisi *neuron* yang terkoneksi penuh (*fully connected*) dengan lapisan lain [28]. Bagian dari lapisan arsitektur CNN dijelaskan pada Tabel 2.1.

Tabel 2.1 Algoritma *Convolutional Neural Network* [28]

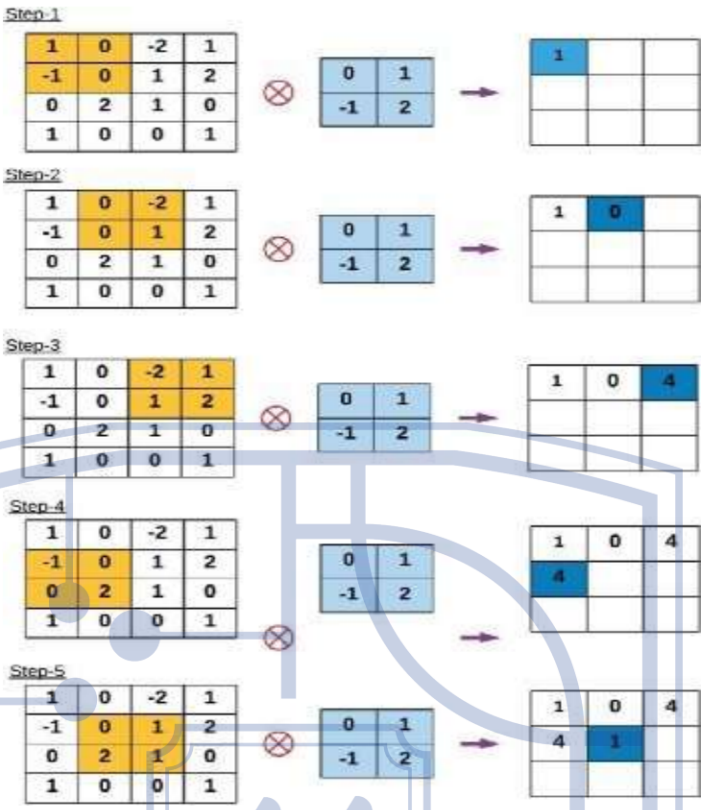
Algoritma *Convolutional Neural Network*	
Feature learning	*Input Layer*
	Convolution Layer
	Activation Layer
	Pooling Layer
Classification	*Fully Connected Layer*
	Output Layer

1. *Input Layer*

   Lapisan ini berguna untuk menampung pixel value dari citra yang diinputkan [28]. *Input layer* berfungsi untuk menerima nilai masukan dari setiap rekaman dalam data. Jumlah simpul *input* pada layer ini sama dengan jumlah variable prediktor yang ada [30].

2. *Convolution Layer*

   Convolution layer merupakan lapisan pertama yang menerima input citra yang dimasukkan ke dalam arsitektur. *Convolution layer* digunakan untuk ekstraksi fitur dan terdiri dari filter yang membagi vektor *input* menjadi blok-blok kecil [31]. Pada layer konvolusional, sebuah matriks kernel digeser sebanyak jumlah stride melalui matriks input untuk menghasilkan fitur pada layer berikutnya. Konvolusi dilakukan dengan menggeser matriks kernel di atas matriks masukan [32]. Di setiap lokasi, dilakukan operasi penjumlahan dari perkalian *element-wise* pada matriks kernel dan sebagian dari matriks *input*. Gambar berikut ini proses konvolusi dengan jumlah *stride* = 1, jumlah padding = 0, ukuran kernel = 2 dan ukuran input = 4 [33].



Gambar 2.7 Proses Konvolusi [33]

Perkalian *element-wise* adalah operasi perkalian antara dua matriks yang beroperasi pada elemen yang sesuai dalam masing-masing matriks. Dua elemen dikatakan bersesuaian jika kedua elemen tersebut menempati posisi yang sama di dalam matriks [34]. Perkalian *element-wise* dapat digambarkan seperti berikut ini.

Gambar 2.8 Perkalian Element-wise [34]

Pada layer konvolusional matriks hasil perkalian *element-wise* akan dijumlahkan dan satu piksel dari fitur. Ukuran dari fitur yang dihasilkan pada layer konvolusi dapat dihitung menggunakan rumus berikut ini [35].

$$S_o = \left[\frac{S_i + 2_p - K}{S}\right] + 1 \quad \text{..... (2.1)}$$

Dimana:

S_o = ukuran dari *output*

S_i = ukuran dari *input*

p = jumlah *padding*

K = *ukuran kernel*

S = jumlah *stride*

3. *Activation layer*

Fungsi aktivasi sebagai salah satu parameter penting dalam arsitektur *deep learning* yang menentukan *output*, akurasi, dan efisiensi model pelatihan [32]. Ada beberapa contoh fungsi aktivasi, seperti *Sigmoid*, *Tanh*, *ReLU*, *Leaky ReLU*, *Parametrized ReLU*, *Exponential Linear Unit, Swish,* dan *SoftMax* [36]. Fungsi aktivasi yang umum dan populer digunakan adalah ReLU. Fungsi aktivasi *ReLu* dapat dihitung dengan menggunakan rumus berikut [32].

$$ReLU(x^{(i,j)}) = \max(0, x^{(i,j)}) \quad (2.2)$$

Dimana:

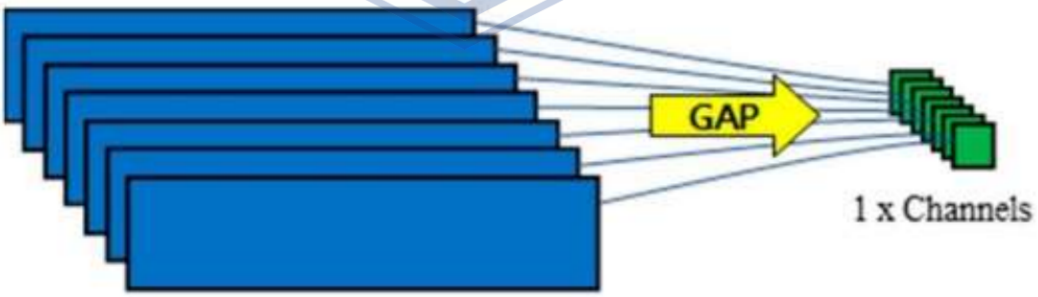
$x^{(i,j)} = nilai\ input\ pada\ posisi\ ke$-i,j (misalnya hasil konvolusi pada titik tersebut.

Fungsi max(0,$x^{(i,j)}$) berarti :

- Jika $x^{(i,j)} > 0$, maka $ReLU(x^{(i,j)}) = x^{(i,j)}$
- Jika $x^{(i,j)} \leq 0$, maka $ReLU(x^{(i,j)}) = 0$

4. *Pooling layer*

Lapisan *pooling* bekerja di setiap tumpukan *feature map* dan melakukan pengurangan ukuran [28]. *Pooling layer* memiliki fungsi untuk mempercepat operasi dan meningkatkan kinerja seluruh *convolutional layer* [31]. Salah satu metode dalam *pooling layer* adalah *Global Average Pooling* (GAP), dimana ukuran *stride* tidak perlu ditentukan sehingga *overfitting* dapat dihindari. Berbeda dengan *average pooling*, yang hanya mengekstrak rata-rata dari area tertentu dan diterapkan setelah setiap lapisan konvolusi, GAP menghitung rata-rata nilai node untuk setiap peta fitur [37].



Gambar 2.9 *Global Average Pooling* [37]

Seperti yang ditunjukkan pada Gambar 2.9 metode *global average pooling* melakukan pengurangan dimensi *feature map* dari $h \times w \times d$ menjadi $1 \times 1 \times d$ di mana *h, w,* dan *d* adalah tinggi, lebar, dan ukuran channel dari masing-masing *feature map* [37]. Operasi GAP menghasilkan angka satu skalar yang kemudian diteruskan ke lapisan *fully connected layer* untuk menghasilkan prediksi akhir. Secara matematis, GAP dapat didefiniskan dengan [38]:

$$GAP(F_c) = \frac{1}{n_c} \sum_{i,j} x_{c,i,j} \quad \text{..... (2.3)}$$

Dimana:

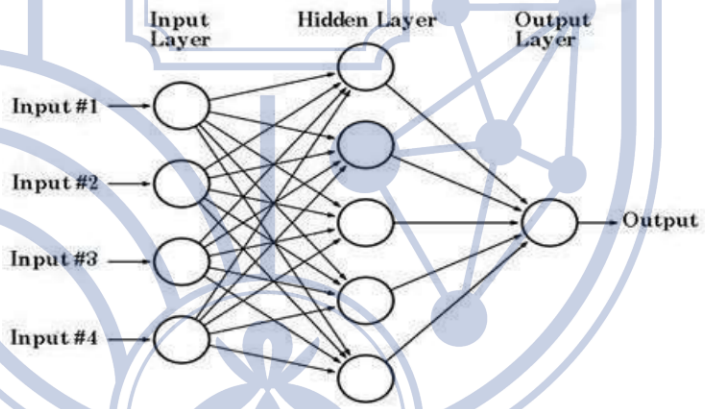
F_c = *feature map*

$x_{c,i,j}$ = nilai aktivasi pada posisi (*i*,*j*) dari *feature map* tersebut

n_c = jumlah total elemen spasial (piksel) dalam *feature map*

5. *Fully connected layer*

Lapisan *Fully-connected* atau lebih dikenal dengan *dense layer* adalah lapisan dimana semua *neuron* aktivitas dari lapisan sebelumnya terhubung semua dengan *neuron* di lapisan selanjutnya seperti halnya jaringan saraf tiruan [28,29].



Gambar 2.10 *Fully Connected Layer* [28]

a. *Forward Propagation*

Forward propagation adalah proses inti dalam jaringan saraf tiruan dimana informasi mengalir secara berurutan dari lapisan input menuju *output* untuk menghasilkan prediksi. Perhitungan pada tahapan ini dapat dilihat pada persamaan berikut [39].

$$R_t = \sigma(X_t W_x + H_{t-1} W_h + b_r) \quad \text{.....(2.4)}$$

Keterangan:

X_t= nilai *input*

σ = fungsi aktivasi *sigmoid*

W = bobot

b = bias

H_{t-1} = *hidden state* sebelumnya

R_t = *reset gate*

b. *Softmax*

Softmax adalah fungsi yang digunakan untuk menghitung probabilitas untuk menentukan klasifikasi multi kelas dengan *output* kelas yang memiliki nilai probabilitas yang paling tinggi. Hasil *output* yang dihasilkan *softmax* memiliki nilai antara 0 sampai 1. Untuk rumus perhitungan *softmax* ditunjukkan pada persamaan berikut [40]:

$$f(Xi) = \frac{Exp(Xi)}{\sum_{j=0}^{k} Exp(xi)}, nilai\ i = 1,2,3,..,k \quad \text{..... (2.5)}$$

c. *Cross-Entropy Loss*

Cross-Entropy Loss atau kerugian entropi silang juga disebut "*log loss*" dan "*soft-max loss"* didefinisikan sebagai persamaan berikut [41]:

$$L(p, y) = -\sum_n y_n \log(p_n), n \in [1, N] \quad \text{.....(2.6)}$$

Dimana y mewakili *output* yang inginkan dan *p* adalah probabilitas untuk setiap kategori *output*. Ada total N *neuron* di lapisan *output layer*, sehingga $p, y, \in \mathbb{R}^N$. Probabilitas setiap kelas dapat dihitung menggunakan fungsi *soft-max* : $p_n = \frac{Exp(pn)}{\sum_k \exp(pk)}$, dimana *pn* adalah skor *output* yang belum dinormalisasikan dari lapisan sebelumnya dalam jaringan. Karena bentuk fungsi normalisasi dalam kerugian ini, kerugian ini juga disebut kerugian *soft-max*. Menarik untuk dicatat bahwa mengoptimalkan parameter jaringan menggunakan *cross-entropy loss* setara dengan meminimalkan divergensi KL antara keluaran yang diprediksi distribusi yang dihasilkan *p* dan keluaran yang diinginkan distribusi sebenarnya *y*. Divergensi KL antara *p* dan *y* dapat diekspresikan sebagai selisih antara *cross-entropy* dilambangkan dengan *L*(-) dan *entropy* dilambangkan dengan *H*(-) seperti persamaan berikut

$$KL(p\ y) = L\ (p, y) - H(p) \quad \text{.....(2.7)}$$

Entropy hanyalah nilai konstan, meminimumkan antropi silang setara dengan meminimumkan divergensi KL antara dua distribusi.

d. *Backward Propagation*

Backward propagation (*backpropagation*) adalah algoritma kunci dalam pelatihan jaringan saraf tiruan yang berfungsi untuk menyesuaikan bobot dan bias dalam jaringan agar mengurangi tingkat kesalahan (*error*) prediksi yang terjadi setelah *forward propagation*. Pada tahapan ini bermaksud untuk mencari nilai tiap bobot dan bias yang optimal berdasarkan *error* yang didapat pada saar *forward propagation* [39]. Perhitungan *update* bobot dan bias pada tahapan ini ditunjukkan pada persamaan berikut.

$$W_{baru} = W_{lama} - \alpha \times \frac{\partial E}{\partial w} \quad (2.8)$$

Keterangan:

α = *learning rate*

$\frac{\partial E}{\partial w}$ = turunan parsial *error* terhadap parameter (bobot dan bias)

W_{lama} = parameter (bobot atau bias) lama

W_{baru} = parameter (bobot atau bias) baru

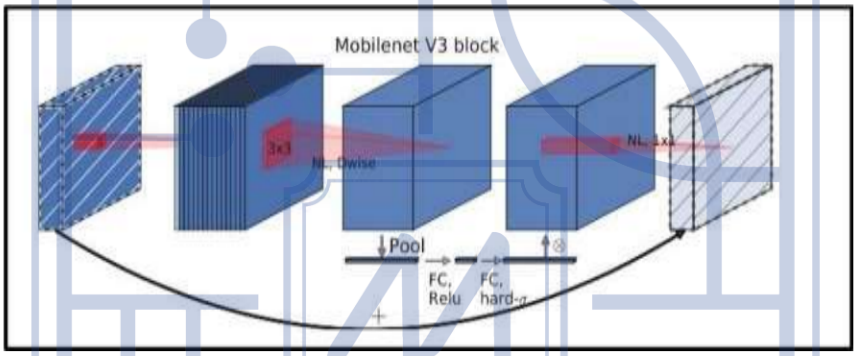
6. *Output layer*

Setelah itu dilanjutkan pada proses klasifikasi, yang mana dengan bantuan aktivasi *softmax* akan diklasifikasi *input* sesuai dengan target kategorinya [28]. Garis yang menghubungkan dengan *output layer* berasal dari *hidden layer* atau *input layer,* dan menghasilkan nilai keluaran yang sesuai dengan variabel prediksi. Keluaran dari *output layer* umumnya merupakan nilai numerik *float* antara 0 hingga 1 [30].

2.4 MobileNetV3

MobileNet adalah keluarga arsitektur CNN yang dirancang khusus untuk perangkat mobile dan edge dengan keterbatasan komputasi. MobileNetV1 memperkenalkan *depthwise separable convolution* sebagai pengganti yang efisien untuk lapisan konvolusi tradisional, MobileNetV2 mengusulkan *linear bottleneck* dan *inverted residual structure* untuk membuat struktur lapisan yang efisien, dan MobileNetV3 menggunakan kombinasi struktur MobileNetV2 dan *Squeeze-and-excite* sebagai *residual layer* untuk membangun model yang lebih efektif [42]. MobileNetV1 memperkenalkan *depthwise separable convolution* sebagai pengganti yang efisien untuk lapisan konvolusi tradisional, MobileNetV2 mengusulkan *linear bottleneck* dan *inverted*

residual structure untuk membuat struktur lapisan yang efisien, dan MobileNetV3 menggunakan kombinasi struktur MobileNetV2 dan *Squeeze-and-excite* sebagai *residual layer* untuk membangun model yang lebih efektif [42]. MobileNetV3 memiliki 2 jenis model yaitu MobileNetV3 *Small* dan MobileNetV3 *Large* kedua model tersebut bisa dimanfaatkan untuk penggunaan sumber daya yang rendah ataupun sumber daya tinggi. MobileNetV3-*Small* adalah varian ringan dari MobileNetV3 yang dioptimalkan untuk perangkat mobile dan edge dengan keterbatasan memori dan daya. Arsitektur ini menggabungkan teknik seperti *depthwise convolution, squeeze-and-excitation*, dan *activation function h-swish*. MobileNetV3-*Small* memiliki performa yang kompetitif dengan ukuran model dan konsumsi daya yang rendah [42].



Gambar 2.11 MobileNetv3 Block [42]

Pada Gambar 2.11 menunjukkan struktur blok MobileNetV3, yaitu unit dasar pada arsitektur MobileNetV3 yang dirancang agar ringan dan efisien untuk perangkat mobile. Prosesnya dimulai dengan *depthwise convolution* (3×3) untuk mengekstraksi fitur spasial secara hemat komputasi, kemudian dilanjutkan dengan *pointwise convolution* (1×1) untuk menggabungkan informasi antar channel. Blok ini juga dilengkapi dengan *Squeeze-and-Excitation* (SE) module, yang menggunakan *global average pooling* dan dua *fully connected layer* untuk memberikan bobot perhatian pada channel yang penting. Aktivasi yang digunakan biasanya ReLU atau *Hard-Swish* agar tetap efisien namun akurat. Jika ukuran *input* dan *output* sama, blok ini menggunakan *residual connection* (*skip connection*) untuk menjaga aliran informasi dan mempermudah proses pelatihan jaringan.

Tabel 2.2 Spesifikasi MobileNetV3 Small [42]

Input	Operator	Exp size	*#out*	SE	NL	*s*
224^2 x 3	conv2d, 3x3	-	16	-	HS	2
122^2 x 16	bneck, 3x3	16	16	✓	RE	2
56^2 x 16	bneck, 3x3	72	24	-	RE	2

28^2 x 24	bneck, 3x3	88	24	-	RE	1
28^2 x 24	bneck, 5x5	96	40	✓	HS	2
14^2 x 40	bneck, 5x5	240	40	✓	HS	1
14^2 x 40	bneck, 5x5	240	40	✓	HS	1
14^2 x 40	bneck, 5x5	120	48	✓	HS	1
14^2 x 48	bneck, 5x5	144	48	✓	HS	1
14^2 x 48	bneck, 5x5	288	96	✓	HS	2
7^2 x 96	bneck, 5x5	576	96	✓	HS	1
7^2 x 96	bneck, 5x5	576	96	✓	HS	1
7^2 x 96	conv2d, 1x1	-	576	✓	HS	1
7^2 x 576	pool, 7x7	-	-	-	-	1
1^2 x 576	conv2d 1x1, NBN	-	1024	-	HS	1
1^2 x 1024	conv2d 1x1, NBN	-	k	-	-	1

MobileNetv3 juga dirancang menggunakan teknik *neural architecture search* (NAS) dan algoritma NetAdapt untuk menyeimbangkan akurasi dan efisiensi pada perangkat mobile. Arsitektur ini juga memperkenalkan fungsi aktivasi *h-swish* yang lebih efisien dibandingkan ReLU6 dan memberikan peningkatan akurasi pada tugas klasifikasi gambar [42]. *#Out* adalah jumlah filter yang digunakan, *exp size* adalah singkatan dari *expansion size*, SE adalah *Squeeze-And-Excite*. HS adalah singkatan dari aktivasi *hard-swish* dan RE adalah singkatan dari aktivasi *Rectified Linear Unit* (ReLU), *No Batch Normalization* (NBN), sedangkan *s* adalah singkatan dari *stride*. MobileNetV3 telah digunakan dalam berbagai aplikasi pengolahan gambar di perangkat mobile, seperti deteksi objek, pengenalan wajah, klasifikasi gambar, dan sebagainya. Model ini telah menjadi populer dalam komunitas penelitian dan industri karena kemampuannya dalam menyediakan solusi yang efisien dan akurat untuk pengolahan gambar di perangkat mobile dengan sumber daya terbatas [43].

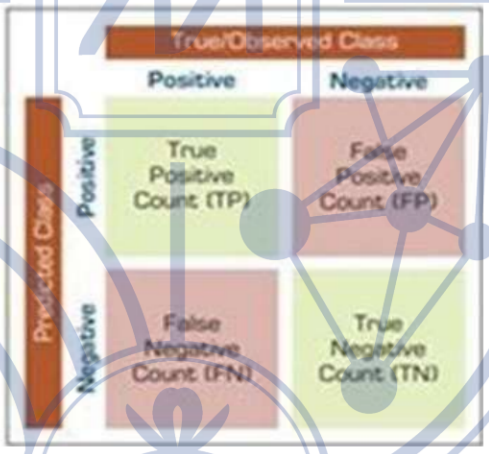
2.5 Evaluasi Model

Evaluasi model adalah tahapan penting karena membantu mengukur sejauh mana dan seberapa baik suatu model bekerja, mampu mencapai tujuan, standar, dan harapan yang telah ditetapkan [44]. Evaluasi model dilakukan untuk mengevaluasi performa model yang

telah dihasilkan. Matriks evaluasi yang digunakan antara lain yaitu, akurasi, presisi, *recall* dan f1 *score*. Evaluasi ini penting untuk menentukan kualitas dan keandalan model dalam menganalisis serta untuk mengidentifikasi area yang memerlukan perbaikan [45].

1. *Confusion Matrix*

Confusion matrix merupakan salah satu metode data mining berupa prediksi tipe klasifikasi yang juga merupakan metode *supervised learning*, dimana terdapat label yang dapat dijadikan acuan dalam mengukur performansi sebuah model dan mewakili jumlah dari nilai yang diprediksi dan nilai aktual [46]. Selain itu, *confusion matrix* merupakan bagian dari keluarga *machine learning* yang mempelajari data yang sudah tersedia dan mengklasifikasikan sebagai data baru serta menghasilkan *output* variabel bersifat kategorial (nominal atau ordinal) [46]. Terdapat 4 istilah sebagai representasi hasil proses klasifikasi pada *confusion matrix*. Keempat istilah tersebut adalah *True Positive* (TP) merupakan data positif yang di prediksi benar, *True Negative* (TN) merupakan data negatif yang diprediksi benar, *False Positive* (FP) merupakan data negatif namun diprediksi sebagai data positif dan *False Negative* (FN) merupakan data positif namun diprediksi sebagai data negatif.



Gambar 2.12 *Confusion Matrix* [47]

Selain itu *confusion matrix* juga digunakan untuk mendapatkan nilai *accuracy, precision* dan *recall* [46]. Untuk mendapatkan nilai masing-masing dapat dihitung dengan rumus di bawah ini [48].

a. *Accuracy*

Accuracy merupakan rasio prediksi benar (*positive* dan *negative*) dengan keseluruhan data.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{.....(2.9)}$$

b. *Precision*

Precision merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif.

$$Precision = \frac{TP}{TP+FP} \quad (2.10)$$

c. *Recall*

Recall merupakan rasio prediksi benar positif dibandingkan keseluruhan data yang benar positif.

$$Recall = \frac{TP}{TP+FN} \quad (2.11)$$

d. F1 *Score*

F1 *score* merupakan salah satu perhitungan evaluasi dalam informasi yang mengkombinasi recall dan presisi yang kemudian hasilnya disebut sebagai nilai pengukuran [47,50]. Nilai f1 *score* menunjukkan perbandingan rata-rata recall dan presisi yang dibobotkan. F1 *score* dapat diperoleh menggunakan persamaan berikut ini [49].

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (2.12)$$

2.6 Metodologi Pengembangan Sistem

2.6.1 Agile Development

Agile development merupakan pendekatan lebih lanjut dari SDLC (*System Development Life Cycle*) untuk memfasilitasi pengembangan aplikasi yang membutuhkan waktu yang singkat dan memberikan tingkat keberhasilan pengembangan aplikasi lebih baik dari metode desain terstuktur. *Agile development* menekankan alur iterasi sehingga jika dalam satu alur terjadi revisi maka akan dilakukan iterasi atau perulangan tanpa menunggu proses selesai terlebih dahulu [50]. *Agile* merupakan metode yang digunakan untuk pengembangan *software* yang dilakukan dengan cara bertahap. Metode *agile* dalam pengunaannya mampu membuat keputusan dalam perubahan *software* sesuai dengan kondisi pasar yanag dituju, hasil *software* metode *agile* ini akan lebih fleksibel dan efisien.

Gambar 2.13 *Agile Development Methodology* [51]

Berikut adalah tahapan dari metode Agile, yaitu [51]:

1. Perencanaan, dalam tahapan disini pihak pengembangan sistem dan klien dapat melakukan perencanaan kebutuhan yang akan dikerjakan.
2. Rancangan, dalam tahapan ini pihak pengembang sistem dapat merancanag terlebih dahulu alur dan sistem seperti bagaimana yang akan dibuat.
3. Tes perangkat lunak, dalam tahapan ini pihak pengembang sudah membuat sistem dan dapat melakukan pengecekan sistem apakah ada eror dari sistem yang sudah dibuat, jika ada *error* maka harus diperbaiki.
4. Dokumentasi dalam tahapan ini memberikan kemudahan untuk pengguna untuk pemeliharaan sistem kedepannya.
5. *Deployment* dalam tahapan ini pengembang dapat menjamin kualitas sistem yang telah dibuat dengan menguji kualitas, keamanan, kecepatan dari sistem yang telah dibuat.

2.6.2 FURPS

Model FURPS merupakan akronim dari *Functional*, *Usability*, *Reliability*, *Performance*, *Supportability*. Kerangka kerja tersebut dapat digunakan baik sebagai persyaratan sistem maupun di penilaian kualitas sistem. Berikut adalah penjelasan dari setiap indikator yang ada pada model FURPS [52]:

1. *Functionality* (Fungsionalitas) merupakan kemampuan perangkat lunak untuk memenuhi kebutuhan dan persyaratan yang harus dimiliki dalam perangkat lunak

tersebut untuk memenuhi harapan pengguna.

2. *Usability* (Kegunaan) yaitu menggambarkan sejauh mana perangkat lunak mudah digunakan dan dipahami oleh pengguna.
3. *Reliability* (Keandalan) mencakup bagaimana tingkat keandalan perangkat lunak untuk berfungsi secara konsisten tanpa mengalami gangguan atau kegagalan.
4. *Performance* (Performa) merupakan kemampuan perangkat lunak untuk berfungsi dengan cepat, efisien, dan responsif sesuai kebutuhan pengguna.
5. *Supportability* (Daya dukung) merupakan sejauh mana perangkat lunak dapat didukung, diperbaiki, diadaptasi, dan diperluas di masa depan. Hal ini meliputi kemampuan untuk di uji, dikembangkan, adaptasi terhadap perbaruan, pemeliharaan, kemudahan servis dan kemampuan instalasi.

2.6.3 Pengujian Fungsional

Tahapan kebutuhan fungsional merupakan pernyataan yang menjelaskan fungsi, aksi, atau layanan yang wajib disediakan oleh suatu sistem untuk memenuhi kebutuhan pengguna. Dalam jurnal Fariz Zakaria dan Utami dijelaskan bahwa kebutuhan fungsional mendeskripsikan bagaimana sistem harus berperilaku dalam menjalankan operasi tertentu dan biasanya dinyatakan secara spesifik serta dapat diuji. Kebutuhan ini menjadi dasar utama dalam pengujian fungsional menggunakan metode *black box* testing, di mana pengujian difokuskan pada kesesuaian *input* dan *output* sistem tanpa memperhatikan struktur internal perangkat lunak [53].

Pengujian fungsional merupakan tahapan pengujian perangkat lunak yang bertujuan untuk memastikan bahwa sistem bekerja sesuai dengan fungsi yang telah ditentukan pada tahap perancangan. Ayuningtyas et al, menyatakan bahwa *black box testing* merupakan metode pengujian yang digunakan untuk menguji fungsionalitas *input* dan *output* perangkat lunak tanpa memperhatikan struktur logika internal sistem. Pengujian dilakukan dengan memberikan data uji berdasarkan spesifikasi perangkat lunak, kemudian keluaran yang dihasilkan diverifikasi untuk memastikan kesesuaiannya dengan kebutuhan fungsional yang diharapkan [54].

2.7 Flutter

Flutter adalah *Software Development Kit* (SDK) untuk pengembangan aplikasi seluler yang bersifat *open source*. Dikembangkan dan disponsori oleh Google, *Flutter*

digunakan untuk mengembangkan aplikasi untuk *Android* dan iOS, serta sistem operasi *Google Fuchsia*. Flutter ditulis dalam bahasa C, C++, dan *Dart* dan menggunakan *Skia Graphics Engine*. Flutter menawarkan satu set *widget* yang dapat disesuaikan sepenuhnya untuk membangun antarmuka asli, termasuk perpustakaan Desain Material yang indah dan widget *Cupertino*. Fitur-fitur ini membantu pengembang dengan cepat membangun *user interface* (UI) tanpa kehilangan status pada emulator, simulator, atau perangkat keras apa pun untuk iOS dan Android [55]. *Flutter* mendukung fitur *cross-platform* yang memungkinkan *developer* untuk menulis kode satu kali, memelihara (*maintain)*, dan dapat berjalan di platform yang berbeda [56]. *Flutter* memungkinkan pengembangan dengan hanya menggunakan satu kode dasar yang mencakup tampilan pengguna dan logika aplikasi [57].

Flutter memiliki beberapa kelebihan dalam pengembangan aplikasi *mobile*. Salah satu keunggulannya adalah kemampuannya untuk membuat proses pengembangan aplikasi sangat cepat karena fitur *hot-reload.* Fitur ini memungkinkan pengembang untuk mengubah kode yang direfleksikan segera setelah perubahan dilakukan. Berikut beberapa kelebihan *Flutter* [56]:

a. *Flutter* memberikan pengalaman menggulir (*scrolling)* yang lebih mulus dan lancar saat menggunakan aplikasi, dengan lebih sedikit hang dan terputus-putus. Hal ini membuat aplikasi berjalan lebih cepat daripada kerangka kerja aplikasi seluler lainnya.
b. *Flutter* mengurangi waktu dan upaya pengujian. Karena aplikasi *Flutter* bersifat lintas platform, penguji tidak perlu menjalankan serangkaian pengujian yang sama pada platform yang berbeda untuk aplikasi yang sama.
c. *Flutter* menawarkan antarmuka pengguna yang sangat baik berkat widget yang berpusat pada desain, alat pengembangan yang canggih, dan API (*Application Programming Interface)* yang canggih. Mirip dengan kerangka kerja reaktif, *Flutter* memungkinkan pengembang untuk menghindari memperbarui konten UI (*user interface)* secara manual. Karena proses pengembangannya yang cepat dan sifatnya yang lintas *platform*, *framework* ini cocok untuk aplikasi MVP (M*inimum Viable Product*).

2.8 Singkong

Singkong (Ubi Kayu) merupakan salah satu bahan pangan pokok dan makanan produk yang ada di Indonesia penghasil energi setelah padi dan jagung [58,59]. Singkong berasal dari benua Amerika, tepatnya dari negara Brazil [59,60]. Penyebarannya hampir ke seluruh dunia, antara lain: Afrika, Madagaskar, India, Tiongkok. Singkong berkembang di negara-negara yang terkenal wilayah pertaniannya dan masuk ke Indonesia pada tahun 1852. Saat ini Indonesia merupakan negara penghasil singkong terbanyak keempat di dunia [60]. Daun, batang dan kulit singkong (*Manihot utillisima*) mengandung kadar serat kasar tinggi dan protein yang rendah [61]. Namun banyak nutrisi penting yang terkandung di dalam daun singkong sendiri, daun singkong kaya akan kandungan vitamin, asam amino essensial, dan juga protein [62]. Ubi singkong memiliki nilai gizi yang positif untuk kesehatan tubuh seperti energi, air, protein, lemak, karbohidrat, kalsium, fosfor, besi, vitamin A, vitamin B, dan vitamin C [63].

Gambar 2.14 Perkebunan Singkong [64]

Adapun penyakit-penyakit utama yang menyerang tanaman singkong antara lain:

1. *Cassava Brown Streak Disease* (CBSD)

Cassava Brown Streak Disease (CBSD) merupakan penyakit virus yang sangat merugikan pada tanaman singkong, khususnya di wilayah Afrika Timur dan Tengah. Penyakit ini disebabkan oleh dua spesies virus, yaitu *Cassava Brown Streak Virus* (CBSV) dan *Ugandan Cassava Brown Streak Virus* (UCBSV), yang keduanya termasuk dalam genus Ipomovirus dari famili Potyviridae [64]. Gejala khas dari penyakit ini meliputi klorosis pada tulang daun, bercak cokelat pada batang, serta nekrosis gabus berwarna cokelat pada umbi

penyimpanan yang dapat menyebabkan umbi tidak layak konsumsi dan mengakibatkan kerugian panen secara keseluruhan. Meski beberapa varietas singkong Afrika menunjukkan gejala yang lebih ringan, seluruh varietas tetap rentan terhadap infeksi. Studi dalam jurnal ini juga menunjukkan bahwa sebagian varietas singkong dari Amerika Selatan memiliki ketahanan yang tinggi terhadap CBSV, termasuk varietas yang menunjukkan tidak adanya gejala pada seluruh jaringan tanaman dan bebas virus berdasarkan hasil uji qRT-PCR [65].

Gambar 2.15 Gejala CBSD Pada Daun Singkong [64]

2. *Cassava Mosaic Disease* (CMD)

Cassava Mosaic Disease (CMD) merupakan penyakit utama yang berdampak besar terhadap penurunan produktivitas tanaman singkong di berbagai wilayah tropis, khususnya Afrika. Penyakit ini disebabkan oleh sekelompok virus yang ditularkan melalui bahan tanam terinfeksi dan juga oleh serangga vektor seperti lalat putih (Bemisia tabaci). Di wilayah Komoro, hasil survei menunjukkan bahwa CMD memiliki tingkat kejadian yang sangat tinggi, bahkan mencapai lebih dari 95% pada beberapa area tanam. Gejala yang ditimbulkan meliputi perubahan warna daun menjadi belang kuning dan hijau (mosaik), serta adanya distorsi atau perubahan bentuk daun. Berdasarkan pengamatan di lapangan, tingkat keparahan penyakit ini rata-rata mencapai tingkat sedang, yang dapat memengaruhi hasil panen secara signifikan [66].

Gambar 2.16 Gejala CMD Pada Daun Singkong [64]

3. *Cassava Bacterial Blight* (CBB)

Cassava Bacterial Blight (CBB) merupakan salah satu penyakit bakteri paling merugikan pada tanaman singkong, yang disebabkan oleh patogen Xanthomonas axonopodis pv. manihotis. CBB menimbulkan gejala nekrotik pada jaringan daun dan batang, seperti bercak-bercak berwarna cokelat hingga kehitaman yang menyebabkan terganggunya fungsi fotosintesis tanaman. Gejala ini dapat berkembang menjadi luka yang lebih luas dan menyebabkan kematian jaringan, terutama pada varietas singkong yang rentan. Dalam laporan Srivathsan et al. (2025), dijelaskan bahwa penyakit ini dapat menurunkan hasil produksi tanaman singkong dalam kisaran 12% hingga 95%, tergantung pada tingkat keparahan dan kondisi lingkungan. CBB juga diketahui menyebar dengan cepat melalui cipratan air hujan dan peralatan pertanian yang tidak steril, serta melalui bahan tanam yang terkontaminasi [67].

Gambar 2.17 Gejala CBB Pada Daun Singkong [65]

4. *Cassava Green Mottle* (CGM)

Cassava Green Mottle Disease (CGM) adalah penyakit virus pada tanaman singkong yang disebabkan oleh *Cassava Green Mottle* Virus (CGMV). Tanaman yang terinfeksi CGMV menunjukkan gejala awal berupa belang sistemik dengan nekrosis pada daun muda. Gejala yang paling mencolok terlihat pada daun yang baru tumbuh, yaitu berupa daun mengeriting dengan tepi yang mengalami distorsi, serta pola belang berwarna hijau dan kuning. Meskipun gejala awal cukup jelas, daun-daun berikutnya seringkali tidak menunjukkan gejala meskipun masih mengandung virus. Dalam beberapa kasus, tanaman tampak pulih tetapi tetap menunjukkan pertumbuhan yang kerdil, dan umbinya menjadi sangat kecil atau bahkan tidak terbentuk. Jika terbentuk, umbi memiliki tekstur berkayu dan tidak layak dikonsumsi. Selain menyerang singkong, virus ini juga dapat ditularkan secara mekanis ke lebih dari 30 spesies tanaman lain dari 12 famili, seperti Amaranthaceae, Cucurbitaceae, dan Solanaceae [64].

Gambar 2.18 Gejala CGM Pada Daun Singkong [68]