

**KLASIFIKASI GEJALA PENYAKIT DAUN PADA TANAMAN  
SINGKONG BERBASIS *VISION* MENGGUNAKAN METODE  
CNN DENGAN ARSITEKTUR MOBILENET**

***(CLASSIFICATION OF LEAF DISEASES SYMPTOMS IN  
CASSAVA PLANTS BASED ON VISION USING CNN METHOD  
WITH MOBILENET ARCHITECTURE)***

**TUGAS AKHIR**

Disusun sebagai syarat mata kuliah Tugas Akhir

Program Studi S1 Teknik Telekomunikasi

Oleh

**ALEX LIANARDO**

**1101184199**



**FAKULTAS TEKNIK ELEKTRO**

**UNIVERSITAS TELKOM**

**BANDUNG**

**2022**

**LEMBAR PENGESAHAN**

**TUGAS AKHIR**

**KLASIFIKASI GEJALA PENYAKIT DAUN PADA TANAMAN  
SINGKONG BERBASIS *VISION* MENGGUNAKAN METODE  
CNN DENGAN ARSITEKTUR MOBILENET**

***(CLASSIFICATION OF LEAF DISEASES SYMPTOMS IN  
CASSAVA PLANTS BASED ON VISION USING CNN METHOD  
WITH MOBILENET ARCHITECTURE)***

**Telah disetujui dan disahkan sebagai Buku Tugas Akhir**

**Program Studi Teknik Telekomunikasi**

**Fakultas Teknik Elektro**

**Universitas Telkom**

**Oleh**

**ALEX LIANARDO**

**1101184199**

**Bandung, 03 Agustus 2022**

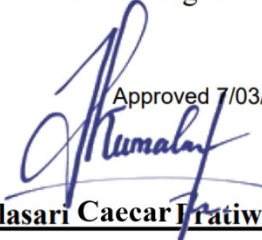
Pembimbing I



**Syamsul Rizal, S.T., M.Eng., PHD**

NIP. 19880018

Pembimbing II



Approved 7/03/2022

**Nor Kumalasari Caecar Irfatiwi, ST., M.T.**

NIP. 20890017

## LEMBAR PERNYATAAN ORISINALITAS

NAMA : Alex Lianardo

NIM : 1101184199

ALAMAT : Jalan Damang Rabu, RT/6, Desa Pujon, Kec Kapuas Tengah, Kab Kapus, Kalimantan Tengah.

EMAIL : alexlianardo9@gmail.com

No. Telp / HP : 082251389982

Menyatakan bahwa Tugas Akhir ini merupakan karya orisinal saya sendiri dengan judul:

**KLASIFIKASI GEJALA PENYAKIT DAUN PADA TANAMAN  
SINGKONG BERBASIS *VISION* MENGGUNAKAN METODE  
CNN DENGAN ARSITEKTUR MOBILENET**

***CLASSIFICATION OF LEAF DISEASES SYMPTOMS IN  
CASSAVA PLANTS BASED ON VISION USING CNN METHOD  
WITH MOBILENET ARCHITECTURE***

Atas pernyataan ini, saya siap menanggung segala resiko / sanksi yang dijatuhkan kepada saya apabila kemudian hari ditemukan adanya pelanggaran terhadap kejujuran akademik atau etika keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidakaslian karya ini.

Bandung, 20 Juli 2022



1101184199  
Alex Lianardo

## ABSTRAK

Singkong merupakan tanaman pangan yang dikonsumsi oleh mayoritas masyarakat sebagai makanan pokok. Dengan bertumbuhnya konsumsi dan produksi singkong setiap tahun, maka semakin sulitnya bagi petani untuk memeriksa kualitas dari tanaman singkong dengan kuantitas yang semakin banyak setiap tahunnya. Salah satu faktor yang dapat merusak kualitas singkong yaitu penyakit tanaman singkong, gejala dari penyakit singkong sendiri dapat dilihat melalui pengecekan secara visual.

Dengan permasalahan tersebut, penulis menggunakan pengolah citra data berbasis algoritma *convolution neural network* (CNN) di mana merupakan salah satu metode dari *deep learning* untuk mengklasifikasi gejala penyakit pada tanaman singkong melalui citra data daun singkong. Penulis membandingkan kinerja dari arsitektur CNN yaitu MobileNet V1, MobileNet V2, MobileNet V3, dan CropNet. Terdapat 5656 citra data dengan format JPG yang didapatkan dari situs web [www.kaggle.com](http://www.kaggle.com), yang telah diklasifikasikan sebagai lima kelas yaitu CBSD, CMD, CBB, CGM dan *healthy* (daun sehat).

Hasil terbaik yang didapatkan pada Tugas Akhir ini menggunakan arsitektur CropNet dengan *hyperparameter* berupa *optimizer* Adam, *learning rate* 0.001, dan *batch size* 32. Hasil yang diperoleh ialah akurasi mencapai 87.47%, presisi sebesar 87%, *recall* sebesar 82%, dan *F1-Score* sebesar 84.2%.

**Keywords:** MobileNet V1, MobileNet V2, MobileNet V3, CropNet.

## ABSTRACT

Cassava is a food crop that is consumed by the majority of people as a staple food. With the growth of consumption and production of cassava every year, it is increasingly difficult for farmers to check the quality of cassava plants with more and more quantities every year. One of the factors that can damage the quality of cassava is cassava plant disease, the symptoms of cassava disease itself can be seen through visual inspection.

The author use a data image processing based on the Convolution Neural Network (CNN) algorithm which is one method of deep learning to classify disease symptoms in cassava plants through cassava leaf data images. The author compare the performance of the CNN architecture, namely MobileNet V1, MobileNet V2, MobileNet V3, and CropNet. There are 5656 data images in JPG format obtained from the website [www.kaggle.com](http://www.kaggle.com), which have been classified into five classes, namely CBSD, CMD, CBB, CGM and healthy (healthy leaves).

The best results obtained in this Final Project are using the CropNet architecture with hyperparameters in the form of Adam *optimizer*, *learning rate* 0.001, and *batch size* 32. The results obtained are 87.47% accuracy, 87% precision, 82% *recall*, and F1-Score of 84.2%.

**Keywords:** MobileNet V1, MobileNet V2, MobileNet V3, CropNet.

## KATA PENGANTAR

Puji syukur yang penulis panjatkan kepada Tuhan Yesus Kristus, karena berkat kasih setiaNya, penulis dapat menyelesaikan tugas akhir dengan judul “Klasifikasi Gejala Penyakit Daun Pada Tanaman Singkong Berbasis *Vision* Menggunakan Metode CNN Dengan Arsitektur Mobilenet”. Tugas akhir ini merupakan syarat bagi penulis untuk dapat menyelesaikan pendidikan program studi Sarjana Teknik Telekomunikasi, Fakultas Elektro, Telkom University.

Dalam pengerjaan tugas akhir ini, banyak sekali bantuan luar biasa yang penulis terima sehingga dapat menyelesaikan tugas akhir ini dengan baik dan tepat pada waktunya. Oleh karena itu penulis ingin mengucapkan terima kasih yang teramat besar bagi semua individu yang sudah membantu penulis untuk menyelesaikan pengerjaan tugas akhir ini.

Penulis sepenuhnya sadar dari segi perancangan dan penulisan terdapat banyak kekurangan. Oleh karena itu, penulis sangat berharap pada pembaca agar memberikan kritik, saran, dan masukan dalam perbaikan dalam penulisan tugas akhir ini. Penulis juga sangat berharap penelitian ini dapat bermanfaat bagi pembaca dalam melakukan penelitian selanjutnya.

Bandung, 20 Juli 2022



1101184199  
Alex Lianardo

## UCAPAN TERIMAKASIH

Pada pengerjaan tugas akhir ini, penulis banyak mendapatkan bantuan serta dukungan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan ucapan terimakasih kepada:

1. Tuhan Yesus Kristus. Karena berkat-Nya yang luar biasa, penulis dapat menyelesaikan seluruh perkuliahan sampai dengan tugas akhir ini dengan sangat luar biasa.
2. Kedua orang tua penulis, kakak, dan adik-adik yang selalu memberi dukungan, do'a, dan juga motivasi-motivasi kepada penulis sehingga dapat menyelesaikan tugas akhir ini dengan baik.
3. Bapak Syamsul Rizal dan Ibu Nor Kumalasari Caecar Pratiwi, yang sudah menjadi pembimbing 1 dan pembimbing 2 penulis selama 1 tahun ini yang banyak memberikan dukungan, motivasi, pembelajaran hidup maupun akademis, dan arahan, sehingga tugas akhir penulis bisa berjalan dengan baik.
4. Teman-teman "Riset Singkong Keju" yang sudah bersama-sama mengerjakan tugas riset hingga penulis dapat memahami tentang cara pengerjaan tugas akhir penulis.
5. Novita Risdanti yang sudah menemani saya dalam menjalani kuliah dan juga pengerjaan tugas akhir penulis.
6. Teman-teman "Anak Korporat" (Ikhwan dan Agnes) yang sudah menemani penulis baik dalam pengerjaan tugas akhir maupun dalam menjalani kehidupan.
7. Teman-teman "Dewan Penasihat PMK" (Ezra, Angel, dan Puter) yang sudah membantu penulis dalam mengurus kepengurusan PMK sembari mengerjakan tugas akhir penulis.
8. Teman-teman UKM PMK, yang telah menjadi tempat pertumbuhan rohani bagi penulis, dan telah memberikan dukungan dan do'a bagi penulis dalam menyelesaikan studi.

## DAFTAR ISI

<b>LEMBAR PENGESAHAN .....</b>	<b>ii</b>
<b>LEMBAR PERNYATAAN ORISINALITAS .....</b>	<b>iii</b>
<b>ABSTRAK .....</b>	<b>iv</b>
<b>ABSTRACT .....</b>	<b>v</b>
<b>KATA PENGANTAR.....</b>	<b>vi</b>
<b>UCAPAN TERIMAKASIH.....</b>	<b>vii</b>
<b>DAFTAR ISI.....</b>	<b>viii</b>
<b>DAFTAR GAMBAR.....</b>	<b>xi</b>
<b>DAFTAR TABEL .....</b>	<b>xii</b>
<b>BAB I PENDAHULUAN.....</b>	<b>1</b>
1.1    Latar Belakang Masalah .....	1
1.2    Rumusan Masalah .....	2
1.3    Tujuan dan Manfaat.....	3
1.4    Batasan Masalah.....	3
1.5    Metode Penelitian.....	4
<b>BAB II KONSEP DASAR .....</b>	<b>5</b>
2.1    Penyakit Tanaman Singkong.....	5
2.1.1 <i>Cassava Brown Streak Disease (CBSD)</i> .....	5
2.1.2 <i>Cassava Mosaic Disease (CMD)</i> .....	6
2.1.3 <i>Cassava Bacterial Blight (CBB)</i> .....	6
2.1.4 <i>Cassava Green Mite (CGM)</i> .....	7
2.2 <i>Digital Image Processing</i> .....	8
2.3 <i>Machine Learning (ML)</i> .....	9
2.2.1 <i>Supervised Learning</i> .....	10
2.2.2 <i>Unsupervised Learning</i> .....	10
2.2.3 <i>Reinforcement Learning</i> .....	11
2.4 <i>Artificial Neural Network (ANN)</i> .....	11



2.5	<i>Deep Learning (DL)</i> .....	12
2.6	<i>Convolutional Neural Network (CNN)</i> .....	12
2.4.1	<i>Convolutional Layer</i> .....	13
2.4.2	<i>Pooling Layer</i> .....	16
2.4.3	<i>Fully Connected Layer</i> .....	16
2.4.4	<i>Softmax Activation</i> .....	17
2.7	<i>Transfer Learning</i> .....	17
2.8	MobileNet.....	17
2.9	MobileNet V2.....	18
2.10	MobileNet V3.....	19
2.11	CropNet .....	19
2.12	<i>Optimizer</i> .....	20
2.9.1	Adam.....	20
2.9.2	Nadam.....	20
2.9.3	RMSProp .....	21
2.13	<i>Learning rate</i> .....	22
2.14	<i>Batch size</i> .....	22
<b>BAB III MODEL SISTEM DAN PERANCANGAN.....</b>		<b>23</b>
3.1	Desain Model Sistem.....	23
3.2	Dataset .....	23
3.3	<i>Balancing Dataset</i> .....	25
3.3.1	<i>Undersampling</i> .....	25
3.3.2	<i>Oversampling</i> .....	25
3.4	Augmentasi Dataset.....	25
3.5	Pelatihan dan Pengujian Model.....	26
3.6	Parameter Pemanding Performansi Model.....	27
3.4.1	Akurasi.....	29
3.4.2	Presisi.....	29

3.4.3	<i>Recall</i> .....	29
3.4.4	<i>F1-Score</i> .....	29
3.4.5	<i>Loss Function</i> .....	29
3.7	Perancangan Skenario Pengujian .....	30
<b>BAB IV HASIL DAN ANALISIS .....</b>		<b>31</b>
4.1	Pengujian Model .....	31
4.2	Hasil dan Analisis Pengujian .....	31
4.3	Performansi Akurasi Model dengan Arsitektur MobileNet .....	31
4.4.1	Data Citra Asli .....	32
4.4.2	Data <i>Balanced</i> .....	34
4.4	Performansi Akurasi Model dengan Arsitektur MobileNet V2 .....	36
4.5.1	Data Citra Asli .....	37
4.5.2	Data <i>Balanced</i> .....	38
4.5	Performansi Akurasi Model dengan Arsitektur MobileNet V3 .....	40
4.6.1	Data Citra Asli .....	41
4.6.2	Data <i>Balanced</i> .....	42
4.6	Performansi Akurasi Model dengan Arsitektur CropNet .....	44
4.7.1	Data Citra Asli .....	45
4.7.2	Data <i>Balanced</i> .....	46
4.7	Analisa Hasil Pengujian .....	49
<b>BAB V KESIMPULAN DAN SARAN .....</b>		<b>53</b>
5.1	Kesimpulan .....	53
5.2	Saran .....	53
<b>DAFTAR PUSTAKA .....</b>		<b>53</b>

## DAFTAR GAMBAR

<b>Gambar 2.1</b> Gejala penyakit CBSD pada tanaman singkong.....	5
<b>Gambar 2.2</b> Gejala penyakit CBSD pada tanaman singkong.....	6
<b>Gambar 2.3</b> Gejala penyakit CBB pada tanaman singkong. ....	7
<b>Gambar 2.4</b> Gejala dari CGM. ....	8
<b>Gambar 2.5</b> Venn diagram dari konsep machine learning dan kelas-kelasnya...	10
<b>Gambar 2.6</b> Gambaran neural network dengan tiga lapisan.....	11
<b>Gambar 2.7</b> Proses perbedaan pembuatan model antara program eksplisit, machine learning, dan deep learning.....	12
<b>Gambar 2.8</b> Gambaran arsitektur CNN. ....	13
<b>Gambar 2.9</b> Proses pada convolutional layer dengan tiga channel (RGB). ....	14
<b>Gambar 2.10</b> Representasi input tiga dimensi dari CNN. ....	15
<b>Gambar 2.11</b> Sliding window.....	15
<b>Gambar 2.12</b> Max pooling.....	16
<b>Gambar 2.13</b> Gambaran arsitektur CropNet.....	19
<b>Gambar 3.1</b> Blok diagram sistem. ....	23
<b>Gambar 3.2</b> Contoh citra data dari tiap kelas di dataset. ....	24
<b>Gambar 3.3</b> <i>Flowchart</i> proses pelatihan dan pengujian. ....	27
<b>Gambar 4.1</b> Grafik Akurasi dan <i>Loss</i> Citra Asli (a) dan <i>Balanced</i> (b). ....	50
<b>Gambar 4.2</b> Confusion Matrix Citra Asli (a) dan <i>Balanced</i> (b). ....	50

## DAFTAR TABEL

<b>Tabel 3.1</b> <i>Confusion matrix</i> lima kelas.....	28
<b>Tabel 4.1</b> Hasil pengaruh <i>optimizer</i> ketika menggunakan data citra asli.....	32
<b>Tabel 4.2</b> Hasil pengaruh <i>learning rate</i> ketika menggunakan data citra asli.....	33
<b>Tabel 4.3</b> Pengaruh <i>batch size</i> ketika menggunakan data citra asli.....	33
<b>Tabel 4.4</b> Hasil pengaruh <i>optimizer</i> ketika menggunakan data <i>balanced</i> .....	35
<b>Tabel 4.5</b> Hasil pengaruh <i>learning rate</i> ketika menggunakan data <i>balanced</i> .....	35
<b>Tabel 4.6</b> Pengaruh <i>batch size</i> ketika menggunakan data <i>balanced</i> .....	36
<b>Tabel 4.7</b> Hasil pengaruh <i>optimizer</i> ketika menggunakan data citra asli.....	37
<b>Tabel 4.8</b> Hasil pengaruh <i>learning rate</i> ketika menggunakan data citra asli.....	37
<b>Tabel 4.9</b> Pengaruh <i>batch size</i> ketika menggunakan data citra asli.....	38
<b>Tabel 4.10</b> Hasil pengaruh <i>optimizer</i> ketika menggunakan data <i>balanced</i> .....	39
<b>Tabel 4.11</b> Hasil pengaruh <i>learning rate</i> ketika menggunakan data <i>balanced</i> ....	39
<b>Tabel 4.12</b> Pengaruh <i>batch size</i> ketika menggunakan data <i>balanced</i> .....	40
<b>Tabel 4.13</b> Hasil pengaruh <i>optimizer</i> ketika menggunakan data citra asli.....	41
<b>Tabel 4.14</b> Hasil pengaruh <i>learning rate</i> ketika menggunakan data citra asli.....	41
<b>Tabel 4.15</b> Pengaruh <i>batch size</i> ketika menggunakan data citra asli.....	42
<b>Tabel 4.16</b> Hasil pengaruh <i>optimizer</i> ketika menggunakan data <i>balanced</i> .....	43
<b>Tabel 4.17</b> Hasil pengaruh <i>learning rate</i> ketika menggunakan data <i>balanced</i> ....	43
<b>Tabel 4.18</b> Pengaruh <i>batch size</i> ketika menggunakan data <i>balanced</i> .....	44
<b>Tabel 4.19</b> Hasil pengaruh <i>optimizer</i> ketika menggunakan data citra asli.....	45
<b>Tabel 4.20</b> Hasil pengaruh <i>learning rate</i> ketika menggunakan data citra asli.....	45
<b>Tabel 4.21</b> Pengaruh <i>batch size</i> ketika menggunakan data citra asli.....	46
<b>Tabel 4.22</b> Hasil pengaruh <i>optimizer</i> ketika menggunakan data <i>balanced</i> .....	47
<b>Tabel 4.23</b> Hasil pengaruh <i>learning rate</i> ketika menggunakan data <i>balanced</i> ....	47
<b>Tabel 4.24</b> Pengaruh <i>batch size</i> ketika menggunakan data <i>balanced</i> .....	48
<b>Tabel 4.25</b> Hasil dan parameter terbaik pengujian di setiap arsitektur dan jenis data.....	49
<b>Tabel 4.26</b> Jumlah data dan nilai <i>true positive</i> per kelas.....	51
<b>Tabel 4.27</b> Nilai presisi, <i>recall</i> , dan <i>F1-Score</i> model terbaik.....	52
<b>Tabel 4.28</b> Perbedaan hasil akurasi Tugas Akhir dengan penelitian sebelumnya.....	52

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang Masalah

Singkong merupakan salah satu komoditas pertanian yang telah banyak diolah menjadi berbagai produk jadi yang memiliki nilai tambah tinggi. Indonesia menjadi salah satu negara yang memproduksi singkong dengan produksi sebesar 19 hingga 20 juta ton lebih pada tahun 2021, menjadikan Indonesia menjadi negara ke empat terbesar yang memproduksi singkong di dunia [1]. Setiap tahun berdasarkan data dari [2] tahun 1995 – hingga sekarang produksi singkong di Indonesia selalu bertambah setiap tahunnya secara kumulatif. Hal ini tentu menjadikan singkong sebagai salah satu mata pencaharian paling menjanjikan bagi petani karena banyaknya target konsumen yang bisa di jangkau baik di dalam negeri maupun di luar negeri.

Salah satu faktor yang dapat mempengaruhi baik atau buruknya kualitas dari tanaman singkong adalah penyakit pada singkong [3]. Pengklasifikasian penyakit singkong sangat memakan waktu bagi petani karena memiliki gejala yang berbeda-beda. Melakukan identifikasi penyakit tanaman singkong melalui laboratorium membutuhkan waktu yang lama, biaya yang mahal, dan terbatasnya laboratorium yang ada. Oleh karena itu, baik atau buruknya kualitas singkong dapat berdampak bagi nilai jual dari tanaman singkong itu sendiri [3].

Teknologi *machine learning* (ML) adalah subbidang dari *artificial intelligence* di mana bisa di artikan sebagai interaksi antara sejumlah data dan kondisi dengan aksi yang disediakan oleh manusia melalui sedikit pemrograman dengan meniru cara belajar manusia [4]. Penggunaan *machine learning* telah banyak digunakan di perusahaan berbasis teknologi informasi dan terbukti lebih efisien dan akurat dalam mengerjakan pekerjaan manusia sehari-hari dibandingkan manusia sendiri. Sedangkan teknologi *deep learning* (DL) adalah subbidang dari *machine learning*, perbedaannya adalah penggunaan *neural network* sebagai algoritma untuk pekerjaan-pekerjaan kompleks di mana sangat sulit bagi mesin untuk pelajari tapi bukan berarti tidak mungkin [4]. Dalam penelitian ini, penulis akan membahas bagaimana menggunakan teknologi model *machine learning* untuk

mengklasifikasi penyakit singkong melalui dataset berupa citra data daun singkong sebagai solusi untuk mendeteksi jenis penyakit menggunakan metode *deep learning* dengan algoritma *convolutional neural network* (CNN) [5].

Pada penelitian [6] Patike Kiran Rao, R Sandeep Kumar, dan Dr K Sreenivasulu klasifikasi penyakit tanaman singkong menggunakan *deep learning* dengan model arsitektur UNet terhadap dua kelas penyakit yaitu CMD dan CBB, penelitian ini mendapatkan hasil akurasi sebesar 83.9%. Klasifikasi yang dilakukan menggunakan kaggle *challenge* dataset dengan jumlah 21397 citra data yang diambil dan dilabel ke dua kelas klasifikasi yaitu CMD dan CBB.

Pada penelitian Tugas Akhir ini, penulis menggunakan metode CNN dan melakukan perbandingan terhadap beberapa arsitektur CNN berupa MobileNet [7], MobileNet V2, MobileNet V3, dan CropNet [8] dengan mengukur performa parameter akurasi, presisi, dan F1-Score. Penelitian menggunakan arsitektur MobileNet dikarenakan model arsitektur tersebut bersifat ringan, memiliki latensi rendah, dan berdaya rendah di mana cocok untuk berbagai kasus penggunaan seperti di *smartphone android* [7].

Pengklasifikasian dilakukan berdasarkan lima kelas penyakit tanaman berupa *Cassava Brown Streak Disease* (CBSD), *Cassava Mosaic Disease* (CMD), *Cassava Bacterial Blight* (CBB), *Cassava Green Mite* (CGM), daun yang sehat (*healthy*) menggunakan masukan data *training* berupa citra data yang berformat JPG untuk melatih model arsitektur. Data tersebut diperoleh dari situs web [www.kaggle.com](http://www.kaggle.com) dengan total citra data sebanyak 5656 citra data. Hasil analisis dari penelitian ini akan dilihat dari hasil prediksi dari model MobileNet, MobileNet V2, MobileNet V3, dan CropNet, hasil tersebut akan di evaluasi berdasarkan parameter yang akan digunakan untuk menentukan bagus atau tidaknya sebuah model tersebut melalui parameter akurasi, presisi, dan F1-Score.

## 1.2 Rumusan Masalah

Berdasarkan deskripsi latar belakang dan penelitian terkait yang telah dirumuskan, maka berikut beberapa rumusan masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimana implementasi Convolutional Neural Network (CNN) untuk klasifikasi penyakit daun tanaman singkong berbasis citra yang terdiri dari lima jenis penyakit daun tanaman singkong?
2. Bagaimana cara menentukan parameter yang mempengaruhi sistem klasifikasi penyakit daun singkong menggunakan *convolutional neural network*?
3. Bagaimana performansi sistem yang telah dirancang untuk mengklasifikasi penyakit daun singkong menggunakan *convolutional neural network*?

### 1.3 Tujuan dan Manfaat

Berdasarkan deskripsi rumusan masalah dan latar belakang yang telah dirumuskan, maka berikut tujuan pada penelitian ini:

1. Mengimplementasikan Convolutional Neural Network (CNN) untuk klasifikasi penyakit daun tanaman singkong berbasis citra yang terdiri dari lima jenis penyakit daun tanaman singkong.
2. Menganalisa parameter yang mempengaruhi sistem yang telah dirancang.
3. Mengukur dan menganalisis hasil akurasi, presisi, *recall*, dan *F1-Score* yang dihasilkan dari pengklasifikasian.

Adapun manfaat yang didapatkan pada penelitian ini:

1. Menerapkan sistem klasifikasi dengan metode *convolutional neural network* pada penyakit daun tanaman singkong.
2. Membuat suatu *prototype* aplikasi android menggunakan sistem klasifikasi pada penyakit daun tanaman singkong.

### 1.4 Batasan Masalah

Batasan masalah pada penelitian sebagai berikut:

1. Dataset yang digunakan diambil dari [www.kaggle.com](http://www.kaggle.com) berupa citra data.
2. Menggunakan lima kelas penyakit singkong yaitu *Cassava Brown Streak Disease* (CBSD), *Cassava Mosaic Disease* (CMD), *Cassava Bacterial Blight* (CBB), *Cassava Green Mite* (CGM), dan daun yang sehat (*healthy*).

3. Dataset berupa citra gambar yang digunakan berjumlah 5656 citra gambar dengan format JPG.
4. Parameter performansi yang digunakan meliputi akurasi, presisi, *recall*, dan *F1-Score*.
5. Menggunakan metode *Deep Learning Convolutional Neural Network* dengan arsitektur MobileNet, MobileNet V2, MobileNet V3, dan CropNet.
6. Menggunakan bahasa pemrograman *Python*.

## 1.5 Metode Penelitian

Metode yang digunakan pada penelitian ini sebagai berikut:

1. Identifikasi masalah  
Identifikasi masalah dilakukan guna mengetahui latar belakang masalah, rumusan masalah, batasan masalah, tujuan dan manfaat dari penelitian yang dilakukan.
2. Studi literatur  
Studi literatur dilakukan guna mendapatkan informasi mengenai penyakit tanaman singkong menggunakan *Convolutional Neural Network* (CNN) sebagai landasan penelitian dalam penyusunan Tugas Akhir lewat jurnal, buku, dan video *tutorial* youtube.
3. Pengambilan citra data  
Pengambilan citra data menggunakan dataset sekunder yang bersumber dari [www.kaggle.com](http://www.kaggle.com) dengan lima kelas klasifikasi yaitu *Cassava Brown Streak Disease* (CBSD), *Cassava Mosaic Disease* (CMD), *Cassava Bacterial Blight* (CBB), *Cassava Green Mite* (CGM), dan daun yang sehat (*healthy*).
4. Perancangan sistem  
Perancangan sistem dilakukan menggunakan *Pre-processing* data dan mengimplementasikan arsitektur CNN menggunakan bahasa *python* sebagai bahasa pemrograman yang digunakan di penelitian ini.
5. Analisis hasil pengujian performansi model  
Analisis performansi model dilakukan berdasarkan hasil dari tiap nilai akurasi, presisi, *recall*, dan *F1-Score* yang dihasilkan dari tiap arsitektur yang digunakan.



## **BAB II**

### **KONSEP DASAR**

#### **2.1 Penyakit Tanaman Singkong**

Singkong adalah tanaman yang dapat digunakan sebagai bahan kebutuhan pangan sehari-hari maupun sebagai obat herbal untuk berbagai macam penyakit seperti rematik [9]. Sama seperti tanaman pada umumnya, tanaman singkong rentan terhadap hama dan penyakit, di mana dapat berdampak pada kualitas hasil tanaman singkong dan juga harga jual jika dipanen secara besar [10]. Faktor-faktor yang dapat memicu terjadinya penyakit pada tanaman singkong sama seperti tanaman lainnya yaitu berupa hama, dan kondisi lingkungan yang kurang mendukung bagi tanaman singkong. Macam-macam penyakit yang ada pada singkong yaitu berupa *Cassava Brown Streak Disease (CBSD)*, *Cassava Mosaic Disease (CMD)*, *Cassava Bacterial Blight (CBB)*, dan *Cassava Green Mite (CGM)*, penyakit yang ada pada tanaman singkong memiliki gejala dan juga penanganan yang berbeda-beda [3].

##### **2.1.1 *Cassava Brown Streak Disease (CBSD)***

CBSD merupakan penyakit yang umum ditemukan di tanaman singkong dengan kondisi intensitas cuaca hujan dan suhu yang tinggi [11]. Penyebaran penyakit ini membawa spora jamur melalui medium angin dan juga air hujan yang dibawa dari tanaman singkong yang sakit ke tanaman singkong yang sehat didekatnya. Pada saat musim kemarau, jamur berlindung melalui bercak-bercak dan juga pada daun-daun yang telah rontok.



**Gambar 2.1** Gejala penyakit CBSD pada tanaman singkong

Gejala dari penyakit ini bisa dilihat melalui daun yang sudah tua karena rentan terkena penyebaran melalui medium air yang ada di tanah. Gejala awal dari penyakit ini berupa bercak kecil putih hingga coklat muda pada sisi atas daun, terkadang bercak dibatasi lingkaran berwarna sedikit ungu, dan untuk bercak coklat dikarenakan jaringan daun mati (nekrosis) [3]. Jaringan daun nekrotik mudah rontok sehingga nampak adanya lubang-lubang bekas penyakit.

### 2.1.2 *Cassava Mosaic Disease (CMD)*

Penyakit CMD disebabkan virus yang berasal dari genus *Begomovirus* di dalam keluarga *Geminiviridae* [12]. Gejala pada penyakit ini dapat dilihat dari pengurangan ukuran, distorsi lamina daun, memiliki pola mosaik klorosis, dan juga bintik-bintik. Dari semua gejala yang ada, gejala pola mosaik adalah yang paling sering terlihat pada penyakit ini, gejala tersebut dapat dilihat melalui daun di mana warnanya berupa warna hijau pucat hingga keputihan kuning [13].



**Gambar 2.2** Gejala penyakit CBSD pada tanaman singkong.

### 2.1.3 *Cassava Bacterial Blight (CBB)*

CBB merupakan serangan bakteri hawar pada bagian batang dan daun dengan gejala awal berupa kerusakan jaringan (lesion) berwarna abu-abu mirip seperti tersiram air panas. Penyakit ini dapat menular melalui perantara air, tanah, dan kontaminasi dari alat potong stek. Bakteri masuk ke dalam tanaman melalui lubang stomata dan luka pada daun atau batang. Serangga hama seperti belalang juga membantu mempercepat penularan penyakit tersebut dan juga dapat menyebar lebih cepat dengan kondisi cuaca yang hujan [14].



**Gambar 2.3** Gejala penyakit CBB pada tanaman singkong.

Gejala pada penyakit ini memiliki 4 tingkatan:

1. Lesio dengan bentuk menyudut.
2. Lesio meluas menjadi bercak nekrotik (kematian jaringan pada lokasi infeksi).
3. Perlendiran massa bakteri yang terjadi pada tangkai, helai daun, serta batang.
4. Mati pucuk.

#### 2.1.4 *Cassava Green Mite (CGM)*

CGM adalah penyakit kutu tungau hijau pada tanaman singkong, biasanya ditemukan di bagian bawah daun muda tanaman singkong, batang hijau, dan kuncup daun singkong. Tungau bertahan hidup di batang singkong dan daun singkong, tungau tersebut berwarna hijau kekuningan bintik-bintik dan dapat dilihat dengan mata telanjang [14].



**Gambar 2.4** Gejala dari CGM.

Gejala penyakit [15] ini dapat dilihat dari:

1. Klorosis (Penguningan pada daun).
2. Daun menjadi keriput dan berbintik-bintik jika dalam kasus yang serius.
3. Tungau dapat dilihat di bagian bawah daun muda.
4. Bintik-bintik kuning merata pada daun muda yang biasanya di bagian atas tanaman singkong.

## **2.2 Digital Image Processing**

*Digital image processing* adalah suatu proses untuk mendeteksi sebuah citra yang dilihat melalui citra gambar di mana diproses agar komputer dapat menghasilkan representasi yang dapat dihitung [16]. Elemen yang akan diproses dari citra gambar tersebut yaitu berupa nilai angka *pixels* suatu gambar. Untuk citra *grayscale*, nilai yang dihasilkan hanya satu jenis di mana berkisar dari 0 – 255, sedangkan untuk citra gambar yang memiliki warna, nilai warna tersebut akan direpresentasikan dengan tiga nilai yaitu berupa *Red*, *Green*, dan *Blue* (RGB). Proses dari *image processing* memiliki lima tahap teknik [16]:

1. *Image Generation*

Tahap ini mengambil gambar melalui sebuah sensor (kamera) untuk menghasilkan representasi elektronik dari dunia.

2. *Pre-processing*

Pada tahap ini, gambar yang diambil dimodifikasi untuk meningkatkan kualitas gambar. Tahap ini bisa termasuk kompresi, menghaluskan, dan perbaikan warna.

3. *Segmentation*

Selanjutnya citra data akan diproses untuk mengekstrak elemen penting seperti objek, garis, dan bagian yang ingin di proses.

4. *Feature Extraction*

Selanjutnya karakteristik citra data akan dipilih untuk mengidentifikasi sesuatu yang unik dari citra data tersebut.

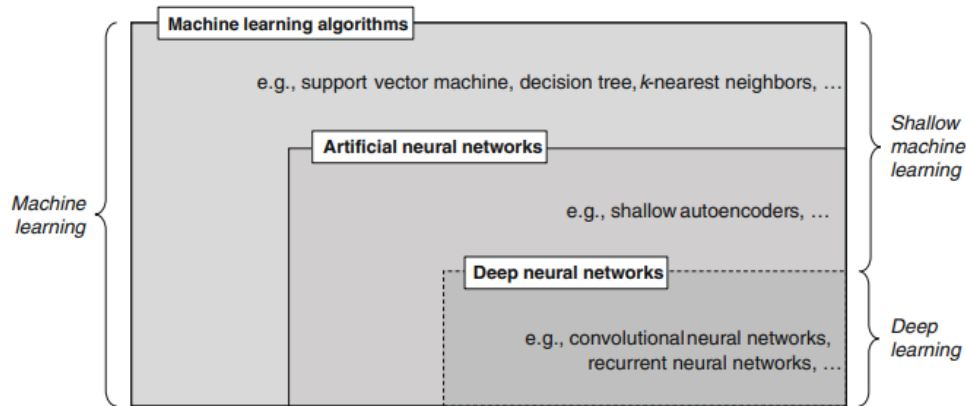
5. *Image Understanding*

Bagian terakhir adalah tahap tertinggi dalam metode *image processing*. Dibagian ini, pengetahuan dan makna dapat disimpulkan dari citra data tersebut.

### 2.3 *Machine Learning* (ML)

*Machine learning* adalah metode dari sebuah sistem mesin untuk mempelajari bagaimana cara manusia belajar untuk kegiatan sehari-hari dengan menggunakan data yang sudah di latih dari suatu subjek untuk dijadikan pondasi atau bahan pembelajaran untuk sistem *machine learning* dalam menyelesaikan tugas terkait subjek yang dipermasalahkan. Data yang sudah dilatih tersebut akan dibaca oleh mesin *machine learning* untuk mengenali suatu pola khusus di mana akan membedakan bagaimana suatu permasalahan dapat dipelajari, diselesaikan, bahkan diklasifikasikan oleh mesin *machine learning* itu sendiri. *Machine learning* sendiri memiliki banyak algoritma yang memungkinkan mesin tersebut dapat mempelajari suatu pola dalam dataset yang sudah disediakan seperti algoritma *Support Vector Machine*, *Decision Tree*, *K-Nearest Neighbors*, dan lain-lain yang memiliki keuntungan dan kekurangan tersendiri. *Machine learning* memiliki tiga tipe jenis cara belajar yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement*

*learning. machine learning* juga memiliki subbidang seperti *artificial neural networks* dan *deep neural networks* [17].



**Gambar 2.5** Venn diagram dari konsep *machine learning* dan kelas-kelasnya.

### 2.2.1 Supervised Learning

*Supervised learning* adalah metode belajar *machine learning* dengan cara mempelajari pola pada dataset yang sudah dilabeli dengan kelas-kelas terkait subjek yang ingin dipelajari, dengan dilabelinya dataset tersebut terhadap kelas yang ditentukan maka model *machine learning* bisa mempelajari dan memprediksi permasalahan tersebut [18]. *Supervised learning* biasanya digunakan untuk beberapa tipe permasalahan seperti klasifikasi, kontrol tanaman, prediksi, ramalan cuaca, dan lain-lain..

### 2.2.2 Unsupervised Learning

*Unsupervised learning* adalah metode belajar *machine learning* dengan cara mencari pola tersembunyi dari dataset yang diberikan. Pola tersembunyi tersebut akan dibandingkan dengan dataset yang lain sehingga model dari *machine learning* tersebut dapat mengetahui apa perbedaan pola dari setiap dataset yang diberikan. Komputasi yang dilakukan dari metode ini biasanya melakukan pembelajaran secara mandiri di mana model *machine learning* akan mempelajari kembali jika pola yang awalnya didapatkan tidak terlalu berguna untuk digunakan kedepannya sehingga model akan melakukan pengecekan ulang dalam pencarian pola tersembunyi yang lainnya sehingga mendapatkan hasil yang maksimal [18].

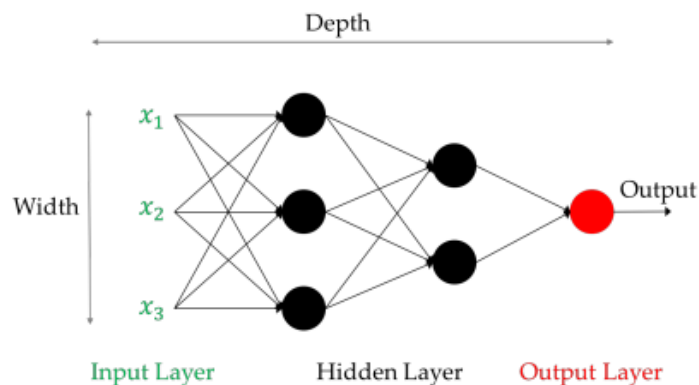
Biasanya metode ini digunakan untuk tipe permasalahan seperti *clustering*, *speech recognition*, deteksi anomali, dan lain-lain.

### 2.2.3 Reinforcement Learning

*Reinforcement learning* adalah metode belajar *machine learning* di mana memiliki basis berupa *agent* dan *environment* di mana dua hal tersebut harus ada dalam metode pembelajaran ini, *environment* (lingkungan) adalah entitas dimana seorang *agent* bisa berinteraksi terhadap lingkungan tersebut. *Agent* dapat melakukan *action* (aksi yang bisa dilakukan seorang *agent* di dalam *environment*) dengan tujuan untuk mendapatkan *reward* (hasil positif dari suatu *action*). Jika hasil yang didapatkan kurang bagus maka *agent* akan mengulang kembali *action* sudah dilakukan dengan harapan mendapatkan hasil terbaik dengan *action* yang berbeda dari sebelumnya, kejadian tersebut biasanya disebut dengan konsep *trial and error* [19].

### 2.4 Artificial Neural Network (ANN)

ANN adalah subbagian dari *machine learning* yang meniru cara kerja otak manusia bekerja atau bisa dikatakan meniru bagaimana proses belajar manusia. ANN memiliki unit yang disebut *neuron* di mana memiliki banyak lapisan *neuron* yang saling berhubungan satu sama lain, terorganisir secara hirarki, dan memiliki nilai atau bobot yang berbeda tiap lapisannya dimana tiap lapisan melakukan transformasi yang berbeda pada masukan untuk mendapatkan tingkat abstraksi dan ekstraksi fitur yang berbeda pada *output* [20].

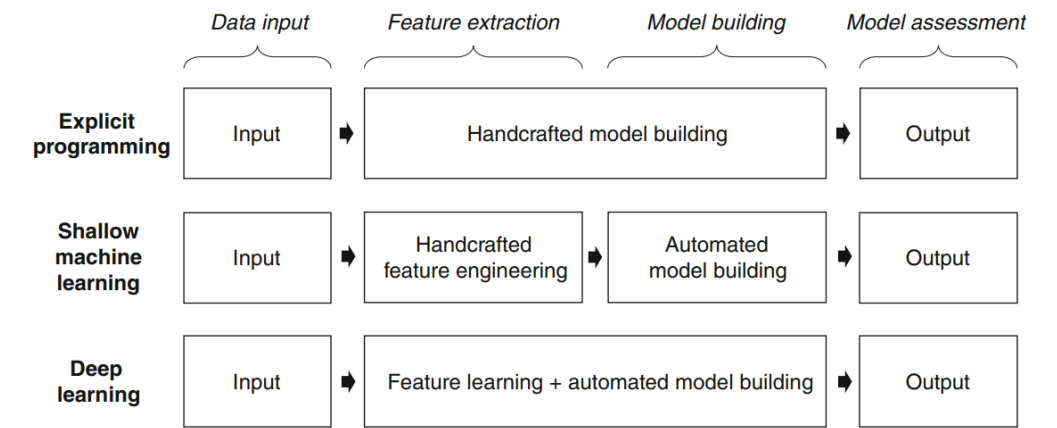


**Gambar 2.6** Gambaran *neural network* dengan tiga lapisan.



## 2.5 Deep Learning (DL)

*Deep learning* adalah sebuah bagian dari *machine learning* di mana metode ini menggunakan struktur *multi-layer* untuk melakukan belajar otomatis dan melakukan ekstraksi tingkat tinggi dari dataset mentah yang disediakan. Perbedaan *deep learning* dengan *machine learning* terletak pada cara belajarnya yang super efisien tanpa adanya manual pembuatan algoritma untuk sebuah model di mana pastinya sangat kompleks dan memakan sangat banyak waktu. Kata dari “*deep*” merupakan representasi dari banyaknya lapisan pada *neuron* di mana meniru cara kerja otak manusia dengan menghubungkan antar *neuron*. Lapisan-lapisan tersebut disebut *artificial neural network* (ANN) di mana merupakan basis dari semua model *deep learning* [20].



**Gambar 2.7** Proses perbedaan pembuatan model antara program eksplisit, *machine learning*, dan *deep learning*.

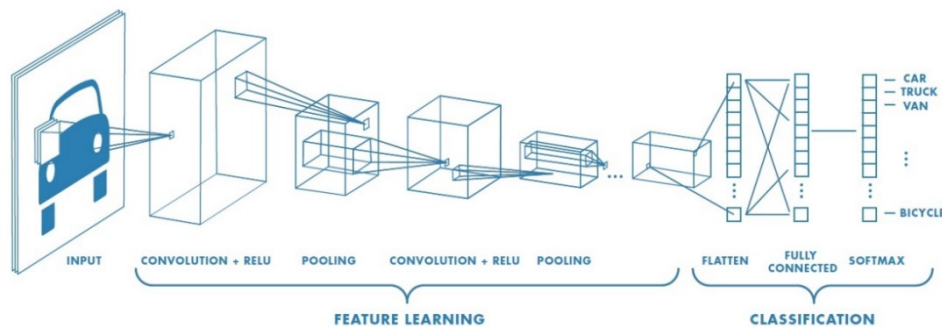
## 2.6 Convolutional Neural Network (CNN)

CNN adalah salah satu tipe *multi-layer neural network* dan juga arsitektur *deep learning* yang terinspirasi dari cara melihat makhluk hidup. CNN sangat cocok untuk berbagai subjek model *machine learning* yang berhubungan dengan pengenalan pola seperti *computer vision* dan *natural language processing* [21]. Sebuah *deep CNN* umumnya membahas tentang sebuah struktur yang berisi *convolutional layers*, *pooling layers*, dan *fully connected layers*. Di bagian *feature learning*, operasi konvolusi digunakan untuk fitur ekstraksi yang berguna menghitung suatu bagian dari citra data untuk mendapatkan nilai agar bisa



dikomputasi, sedangkan untuk *fully connected* berguna untuk pengklasifikasi dari fitur ekstraksi [22]. Untuk *fully connected layers* biasanya digunakan bersamaan dengan fungsi aktivasi *Softmax* untuk tujuan klasifikasi. Berbagai macam arsitektur CNN yang bisa digunakan langsung tanpa harus membuat dari awal untuk bagian *feature learning* dan klasifikasinya, contoh dari arsitektur tersebut seperti AlexNet, OverFeat, GoogleNet, dan lain-lain [22].

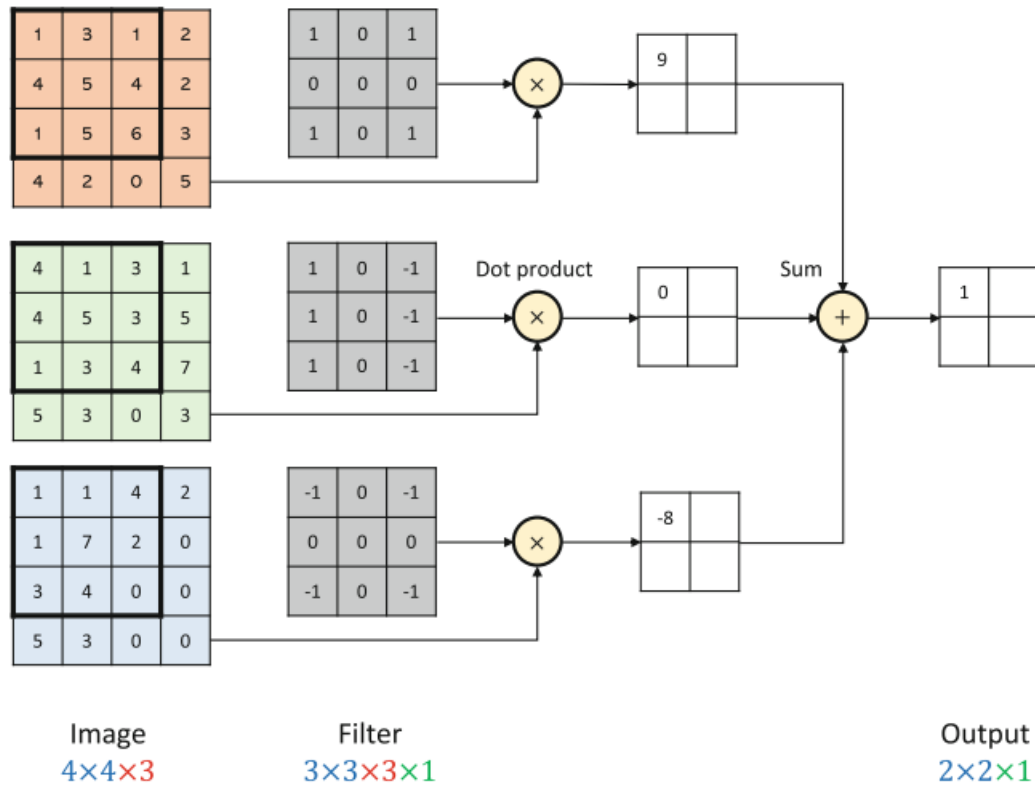
Perbedaan yang bisa dilihat dari ANN dan CNN adalah perlu diketahui jika CNN adalah arsitektur lanjutan dari ANN, dimana CNN memiliki *convolutional layers*, *stride*, *padding*, *pooling layers*, dan *full-connected layers* [5]. Biasanya di dalam arsitektur CNN, setiap *convolution layers* terdapat ReLU dan *Pooling layer* secara berulang-ulang seperti yang direpresentasikan pada Gambar 2.8, lalu diikuti oleh lapisan untuk klasifikasi seperti *flatten*, *fully connected*, dan *activation layer* seperti *softmax*.



**Gambar 2.8** Gambaran arsitektur CNN.

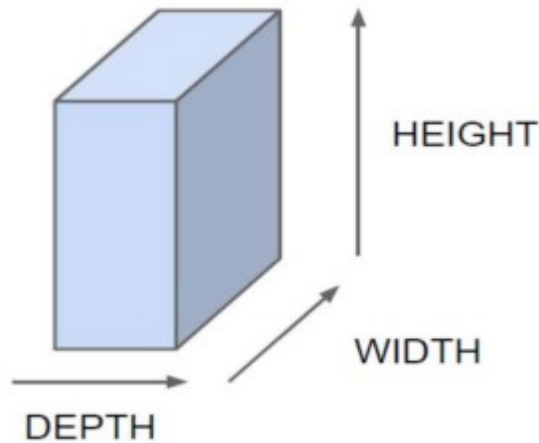
### 2.4.1 Convolutional Layer

*Convolutional layer* adalah lapisan pertama yang bertugas melakukan konvolusi pada citra data di *layer input* [23]. Pada *convolutional layer* proses konvolusi menggunakan filter bisa dilakukan ke citra gambar (dengan panjang *pixel*, tinggi *pixel*, dan juga kedalaman tiga *pixel* berupa RGB *channel*) atau sebuah video (video *grayscale* di mana memiliki resolusi tinggi dan lebar, sedangkan kedalamannya berupa *frames*) [5]. Jumlah dari kanal dan ukuran filter bisa disesuaikan dengan jumlah dari *input* citra data.



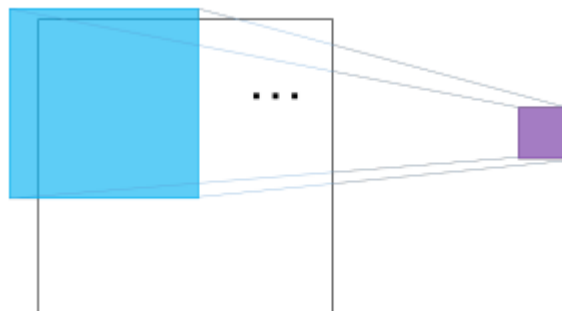
**Gambar 2.9** Proses pada *convolutional layer* dengan tiga channel (RGB).

Kita ambil contoh dengan jaringan yang mendapatkan nilai *pixels* mentah sebagai *input*. Kemudian, untuk menghubungkan *input layers* ke satu *neuron*, akan ada  $32 \times 32 \times 3$  parameter jika menggunakan dataset CIFAR-10 [5] [24]. Jika kita menambahkan satu *neuron* lagi ke dalam *hidden layer*, maka kita memerlukan  $32 \times 32 \times 3$  parameter lagi, di mana akan menjadikan parameter dari kedua *neuron* tersebut menjadi  $32 \times 32 \times 3 \times 2$  parameter [5]. Untuk memperjelas lagi, lebih dari 6000 parameter digunakan untuk menghubungkan *input* ke dua *nodes neurons* saja [5]. Dengan nilai parameter tersebut kita dapat membuat mesin menganalisa hingga tahapan lapisan *output*.



**Gambar 2.10** Representasi *input* tiga dimensi dari CNN.

Perlu diingat jika tujuan awal CNN adalah untuk mengenali aspek informatif atau spesial yang ada pada region tertentu.



**Gambar 2.11** *Sliding window*.

Untuk mengenali objek pada banyak regional maka memerlukan yang namanya *sliding window* seperti pada Gambar 2.10, dengan ilustrasi pada bagian warna biru merupakan representasi satu *window*, kemudian kotak ungu merupakan representasi dari aspek paling spesial (disebut *filter*) dari *window* tersebut. Setiap operasi pada *window* bertujuan untuk mencari aspek spesial yang ada pada *window* tersebut. Dengan kata lain kita ingin mendapatkan nilai numerik (*filter*) dari mentransformasikan suatu *window*. Kita juga bisa mendapatkan nilai  $d$  ( $d$ -channels) numerik dari mentransformasikan suatu *window*. *Window* ini kemudian digeser sebanyak  $T$  kali, sehingga mendapatkan vektor dengan panjang  $d \times T$  [25].

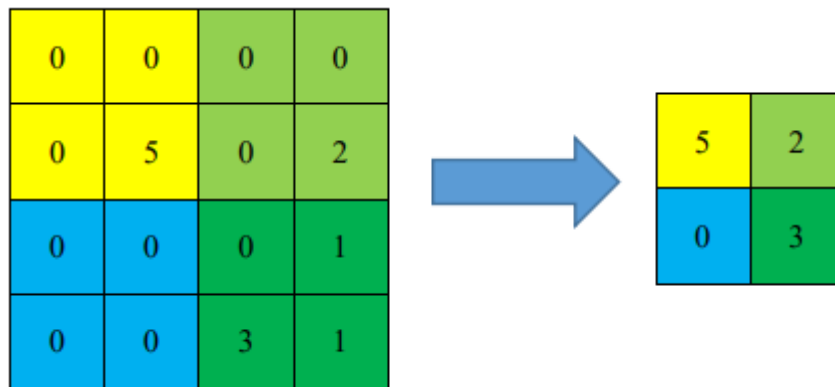
### 2.4.2 RelU

*RelU* berguna untuk mengubah nilai negatif menjadi nol dan nilai positif menjadi nilai tetap [20].

$$\text{RelU } f(x) = \max(0, x) \quad (2.1)$$

### 2.4.3 Pooling Layer

Tujuan utama dari *pooling layer* adalah *downsampling* untuk mengurangi kompleksitas lapisan selanjutnya. Di dalam *image processing*, fitur ini berfungsi untuk mengurangi resolusi citra data [5]. Salah satu metode *pooling* yang sering digunakan adalah *max pooling* di mana cara kerja *max pooling* adalah mempartisi citra data ke *sub-region* persegi panjang dan mengembalikan nilai maksimum dari dalam *sub-region* tersebut [5]. Setelah melewati berbagai macam operasi *convolution* dan *pooling* akan dihasilkan satu vektor yang kemudian akan dilewatkan pada *fully connected layer* untuk melakukan identifikasi suatu permasalahan tertentu [25].



**Gambar 2.12** Max pooling.

### 2.4.4 Fully Connected Layer

*Fully connected layer* sama seperti dengan *convolutional layer*, yang membedakannya adalah *fully connected layer* terhubung ke semua *neuron* sedangkan *convolutional layer* hanya menghubungkan beberapa saja pada *layer input* [5]. Fungsinya adalah mengubah atau melakukan *flatten feature map* agar

hasil fitur berupa *array multi-dimensional* menjadi *array* satu dimensi sehingga data dapat diklasifikasikan [23].

#### 2.4.5 Softmax Activation

*Softmax activation* berfungsi sebagai lapisan akhir dari *neural network*, *softmax* sendiri memiliki fungsi normalisasi agar probabilitas tiap kelas dapat ditemukan di rentang nilai nol hingga satu [19].

$$\text{Softmax } \sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (2.2)$$

Keterangan:

e = konstanta agar memberikan nilai positif;

Z = vektor input ke fungsi softmax (nilai *weight*);

K = jumlah kelas;

#### 2.7 Transfer Learning

*Transfer learning* bertujuan untuk menggunakan pengetahuan dari satu model arsitektur *machine learning* atau *deep learning* yang digunakan untuk berbagai hal seperti klasifikasi dan deteksi. Metode ini digunakan untuk menghemat waktu, memiliki performansi yang lebih baik pada *neural networks* (dalam sebagian besar kasus), dan tidak memerlukan banyak data [26]. Metode ini digunakan pada *pre-trained* model atau model *machine learning* yang sudah dibuat sebelumnya, penggunaan metode ini dalam penelitian bertujuan menggunakan model arsitektur seperti MobileNet untuk digunakan dalam pembelajaran klasifikasi penyakit tanaman singkong.

#### 2.8 MobileNet

Berdasarkan [27] MobileNet menjadi model arsitektur CNN yang memiliki kelebihan seperti *low-cost*, *stable*, dan *high precision*. MobileNet adalah arsitektur CNN yang dibuat oleh *Google* dimana pada awalnya mencoba mengadaptasi arsitektur InceptionNet ke perangkat selular. Tujuannya adalah untuk mengurangi

jumlah parameter dan komputasi sehingga model menjadi lebih ringan, sementara pada saat yang sama tetap mempertahankan kinerja sebanyak mungkin.

**Tabel 2.1** Struktur arsitektur MobileNet.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5×	Conv dw / s1	$3 \times 3 \times 512$ dw
	Conv / s1	$1 \times 1 \times 512 \times 512$
	Conv dw / s2	$3 \times 3 \times 512$ dw
	Conv / s1	$1 \times 1 \times 512 \times 1024$
	Conv dw / s2	$3 \times 3 \times 1024$ dw
	Conv / s1	$1 \times 1 \times 1024 \times 1024$
	Avg Pool / s1	Pool $7 \times 7$
	FC / s1	$1024 \times 1000$
	Softmax / s1	Classifier

MobileNet memperkenalkan sebuah filter yang bernama *depthwise separable convolution*. MobileNet berhasil mengurangi jumlah parameter secara signifikan dan sebagai hasilnya komputasi yang dibutuhkan biasanya hanya 1/9 dari jaringan asli, dengan sedikit pengurangan akurasi [28].

## 2.9 MobileNet V2

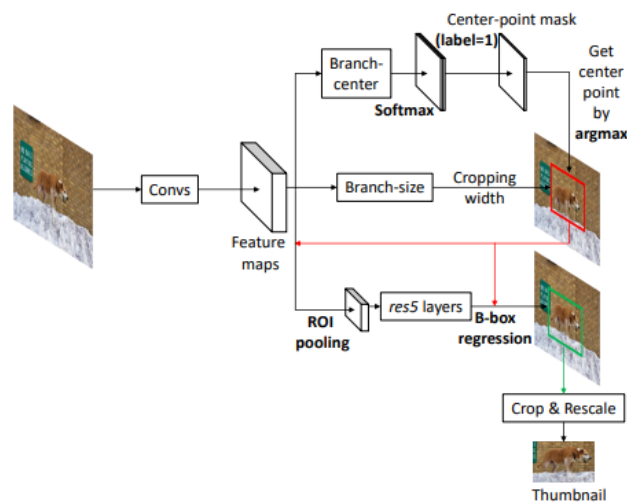
MobileNet V2 adalah arsitektur lanjutan dari MobileNet V1 dimana pada arsitektur ini memperkenalkan dua jenis fitur baru yaitu *inverted residuals* dengan *linear bottlenecks*. Arsitektur ini banyak digunakan untuk objek deteksi dan segmentasi di mana lebih cepat 35% dalam mendeteksi objek dibandingkan dengan MobileNet V1 [29].

## 2.10 MobileNet V3

MobileNet V3, model ringan yang cocok untuk perangkat seluler dirancang melalui arsitektur *network-aware network* dan algoritma NetAdapt [30]. MobileNet V3 sendiri merupakan bentuk paling baru dari MobileNet dimana arsitektur ini sudah sangat jauh melampaui pendahulunya yaitu MobileNet V1 baik dari segi akurasi dan latensi. Arsitektur ini memiliki dua versi yaitu arsitektur MobileNet V3 *small* dan MobileNet V3 *large-crafted* untuk memenuhi kebutuhan yang digunakan pengguna pada arsitektur ini.

## 2.11 CropNet

CropNet adalah arsitektur CNN yang digunakan sebagai arsitektur yang berfokus melakukan deteksi dan klasifikasi penyakit tanaman singkong [31]. CropNet sendiri memiliki empat buah bagian. Lapisan paling awal yaitu merupakan *feature extraction* di mana bisa mengeluarkan hasil *feature maps* dengan citra data sebagai masukan yang sama seperti cara kerja *deep learning* lainnya di mana *branch-center* untuk memprediksi *center-point mask* dan *branch-size* untuk memprediksi ukuran citra data (lebar). Setelah *branch-center* dan *branch-size* dilewatkan, sebuah *cropping window* dilakukan untuk memotong ukuran citra data masukan dan mengimplementasikan *bounding box* pada citra data input untuk meningkatkan performansi [8].



**Gambar 2.13** Gambaran arsitektur *CropNet*.

## 2.12 Optimizer

*Optimizer* merupakan salah satu metode dalam CNN untuk mengoptimalkan *training machine learning* dengan memperkecil nilai *loss* [32]. Untuk *optimizer* yang digunakan pada penelitian ini menggunakan tiga jenis *optimizer*, yaitu:

### 2.9.1 Adam

Adam (*Adaptive Moment Estimation*) adalah *optimizer* yang menghitung adaptasi *learning rate* di setiap parameter dalam jaringan lapisan dan menggabungkan momentum dan *optimizer* RMSProp dengan mempertahankan nilai rata-rata dari gradien dan RMSProp [21].

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \left[ \frac{\delta L}{\delta w_t} \right] \quad (2.3)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[ \frac{\delta L}{\delta w_t} \right]^2 \quad (2.4)$$

Keterangan:

$m_t$  = agregat gradien pada waktu t [saat ini] (mulanya,  $m_t = 0$ );

$v_t$  = jumlah kuadrat gradien masa lalu [i.e  $\text{sum}(\partial L / \partial w_{t-1})$ ] (mulanya,  $v_t = 0$ );

$m_{t-1}$  &  $v_{t-1}$  = agregat gradien pada waktu t-1 [sebelumnya];

$\delta L$  = turunan dari *loss function*;

$\delta w_t$  = turunan dari *weights* terhadap waktu t;

$\beta_1$  &  $\beta_2$  = *decay rate* rata-rata gradien dalam dua formula di atas ( $\beta_1 = 0.9$  &  $\beta_2 = 0.999$ ).

### 2.9.2 Nadam

Nadam (*Nesterov-accelerated Adam*) adalah *optimizer* gabungan dari Adam dan NAG (*Nesterov Accelerated Gradient*) di mana cara kerjanya mirip seperti Adam akan tetapi ditambahkan dengan *optimizer* NAG yaitu metode *momentum-based SGD optimizer* [32].



$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{S}_t} + \epsilon} \left( \beta_1 \hat{V}_t + \frac{1 - \beta_1}{1 - \beta_1^t} \cdot \frac{\partial L}{\partial w_t} \right) \quad (2.5)$$

dimana,

$$\hat{V}_t = \frac{V_t}{1 - \beta_1^t} \quad (2.6)$$

$$\hat{S}_t = \frac{S_t}{1 - \beta_2^t} \quad (2.7)$$

$$V_t = \beta_1 V_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial w_t} \quad (2.8)$$

$$S_t = \beta_2 S_{t-1} + (1 - \beta_2) \left[ \frac{\partial L}{\partial w_t} \right]^2 \quad (2.9)$$

Keterangan:

$\alpha = 0.002$ ;

$\beta_1 = 0.9$ ;

$\beta_2 = 0.999$ ;

$\epsilon = 10^{-7}$ ;

$\delta L$  = turunan dari *loss function*;

$\delta w_t$  = turunan dari *weights* terhadap waktu  $t$ ;

$V_t$  &  $S_t$  = inialisasi dengan 0;

### 2.9.3 RMSProp

RMSProp (*Root Mean Square Propagation*) adalah salah satu *optimizer* yang dirancang untuk menutupi kekurangan dari *optimizer* Adagrad yang secara radikal mengurangi masalah *learning rate*, di mana maksudnya ketika melakukan *training* dengan *epoch* yang sangat besar maka jumlah kuadrat dari atau *sum of square* dari semua gradien masa lalu menjadi sangat besar sehingga hasilnya membuat *learning rate* menjadi sangat-sangat kecil [21].

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t} \quad (2.10)$$

$$S_t = \beta S_{t-1} + (1 - \beta) \left[ \frac{\partial L}{\partial w_t} \right]^2 \quad (2.11)$$

Keterangan:

$\alpha = 0.001$ ;

$\beta = 0.9$ ;

$\delta L$  = turunan dari *loss function*;

S = inialisasi dengan 0;

$\delta w_t$  = turunan dari *weights* terhadap waktu t;

### 2.13 *Learning rate*

*Learning rate* adalah *hyperparameter* yang mengontrol seberapa banyak perubahan model *machine learning* dalam menanggapi kesalahan yang diperkirakan setiap kali bobot model diperbarui. *Learning rate* umumnya memiliki rentang 0.0001 hingga 1. *Learning rate* yang terlalu kecil dapat menyebabkan model *machine learning* menjadi macet dalam suatu kondisi, sedangkan jika *learning rate* terlalu besar akan menyebabkan model *machine learning* mengalami *overfitting* [20].

### 2.14 *Batch size*

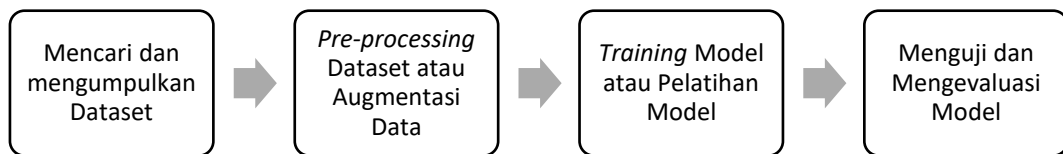
*Batch size* adalah *hyperparameter* yang digunakan untuk mengatur kecepatan komputasi dari model *machine learning*. *Batch size* secara umum digunakan di rentang 8 hingga 1024. Semakin besar *batch size* yang digunakan maka semakin cepat pula komputasi yang dilakukan, sedangkan *batch size* semakin kecil maka komputasi yang dilakukan akan semakin lambat [20].

## BAB III

### MODEL SISTEM DAN PERANCANGAN

#### 3.1 Desain Model Sistem

Pada Tugas Akhir ini penulis akan melakukan penelitian terhadap perbandingan kinerja dari empat model arsitektur CNN untuk pengklasifikasian penyakit tanaman singkong, yaitu dengan menggunakan arsitektur MobileNet, MobileNet V2, MobileNet V3 dan CropNet. Kedua arsitektur tersebut akan dilatih menggunakan dataset berupa citra data penyakit tanaman singkong yang telah diproses melalui tahap *pre-processing*, dataset juga sudah dilabeli untuk diklasifikasi dengan dibagi menjadi empat kelas jenis penyakit tanaman singkong dan satu kelas untuk tanaman singkong yang sehat. Setelah itu penulis akan melakukan evaluasi terhadap model yang akan digunakan berdasarkan parameter pengujian seperti akurasi, presisi, *recall*, dan juga *F1-Score*. Gambar 3.1 berikut merupakan skema umum dari blok diagram sistem.



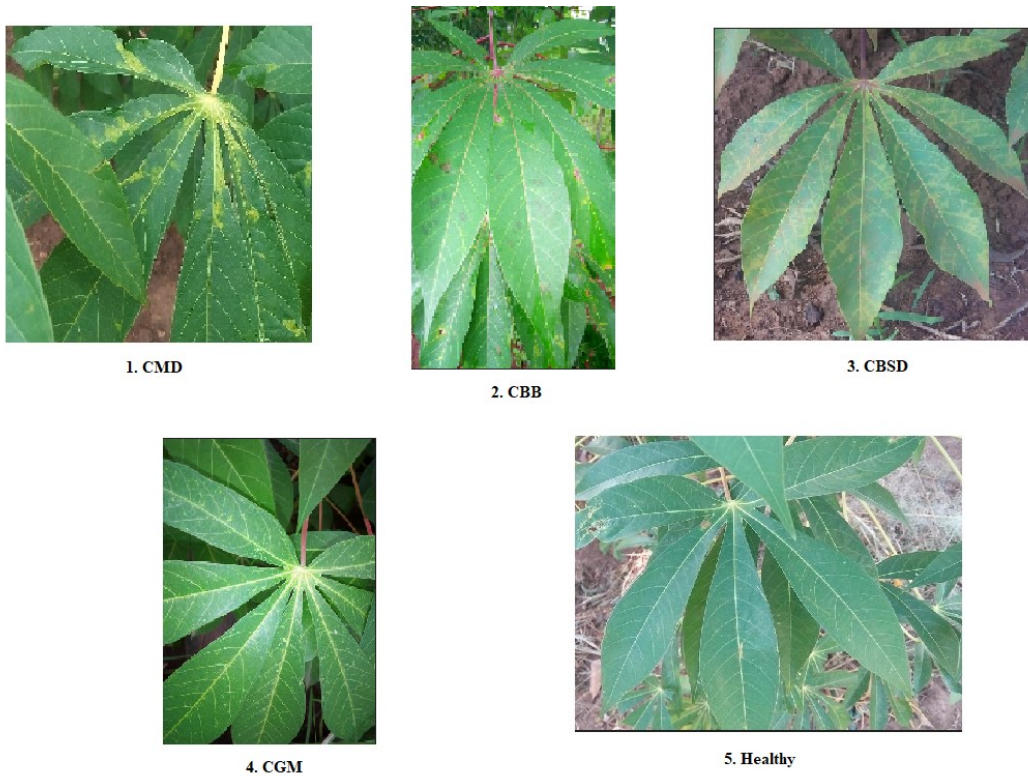
**Gambar 3.1** Blok diagram sistem.

#### 3.2 Dataset

Dataset yang digunakan berasal dari situs web [www.kaggle.com](http://www.kaggle.com) berjudul *Cassava Disease Classification* yang awalnya digunakan untuk kebutuhan kompetisi dari *Community Prediction Competition* dimana kompetisi ini merupakan bagian dari *fine-grained visual-categorization workshop* (FGVC6 *workshop*) di konferensi CVPR 2019. Dataset ini berjumlah 5656 citra data daun singkong yang sudah dilabeli menjadi lima label, dataset tersebut dikumpulkan selama survey regular di Uganda, sebagian besar bersumber dari petani yang mengambil gambar kebun singkong mereka dan dijelaskan oleh para ahli di *National Crops Resources Research Institute* (NaCRRI) bekerjasama dengan lab AI di *Makerere University*, kota *Kampala*. Dataset menggunakan format JPG dan

memiliki warna RGB dengan resolusi yang berbeda-beda di setiap citra data. Dataset sudah terbagi menjadi lima kelas klasifikasi tanaman singkong, yaitu berupa penyakit tanaman singkong berjenis CBSD, CMD, CBB, CGM, dan *healthy* (daun yang sehat). Jumlah dataset latih tiap kelas memiliki jumlah yang tidak seimbang dengan perbandingan:

1. CMD: 2658 citra data,
2. CBB: 466 citra data,
3. CBSD: 1443 citra data,
4. CGM: 773 citra data,
5. *Healthy*: 316 citra data.



**Gambar 3.2** Contoh citra data dari tiap kelas di dataset.

Dataset ini akan dibagi menjadi tiga jenis dataset yaitu dataset *training* dengan jumlah 70% dari total keseluruhan data, dataset validasi dengan jumlah 20% dari total keseluruhan data, dan dataset *testing* dengan jumlah 10% dari total keseluruhan data.

### **3.3 Balancing Dataset**

Pada penelitian ini jumlah dataset per kelas yang digunakan yaitu 1000 citra data per kelas dimana total keseluruhan dataset yaitu 5000 citra data gambar dimana dataset ini akan dibagi menjadi tiga jenis dataset yaitu dataset *training* dengan jumlah 70% dari total keseluruhan data, dataset validasi dengan jumlah 20% dari total keseluruhan data, dan dataset *testing* dengan jumlah 10% dari total keseluruhan data. Teknik yang digunakan untuk menyeimbangkan dataset tiap kelas yaitu menggunakan teknik:

#### **3.3.1 Undersampling**

*Undersampling* adalah pengurangan jumlah dataset tiap kelas dengan jumlah mayoritas ke jumlah dataset kelas dengan jumlah terkecil atau mengurangi dataset dengan jumlah terbesar ke jumlah dataset yang kita inginkan, dalam penelitian ini yaitu 1000 citra data per kelas. Dalam penelitian ini kelas dataset yang mengalami *undersampling* yaitu CMD (2658 citra data) dan CBSD (1443 citra data) [33].

#### **3.3.2 Oversampling**

*Oversampling* adalah penambahan jumlah dataset tiap kelas dengan jumlah minoritas ke jumlah dataset kelas dengan jumlah terbesar atau menambah dataset dengan jumlah terkecil ke jumlah dataset yang kita inginkan, dalam penelitian ini yaitu 1000 citra data per kelas. Dalam penelitian ini kelas dataset yang akan mengalami *oversampling* yaitu *Healthy* (316 citra data), *CGM* (773 citra data), dan *CBB* (466 citra data) [33].

### **3.4 Augmentasi Dataset**

Pada bagian ini, citra data akan melalui tahap augmentasi guna untuk menyeimbangkan ketidakseimbangan dari jumlah data yang ada pada kelas data *training* di dataset. Augmentasi sendiri berupa proses merubah citra data dengan upaya menambah citra data yang tidak seimbang jumlahnya dari kelas yang lain menjadi seimbang dengan berbagai macam variasi citra data dengan bentuk yang baru. Hal tersebut dilakukan agar model CNN semakin mudah dalam mengenali gejala penyakit yang ada pada tanaman singkong karena dengan dataset yang

semakin banyak sebagai dasar untuk penentuan klasifikasi. Pada tahap ini jenis augmentasi yang digunakan sebagai berikut:

- **Resize**

*Resize* merupakan teknik untuk mengubah dimensi citra data sesuai dengan yang dibutuhkan sehingga. Citra data dapat diperbesar ataupun diperkecil.

- **Flipping**

*Flipping* merupakan teknik untuk merotasi citra data dengan derajat tertentu.

- **Zoom**

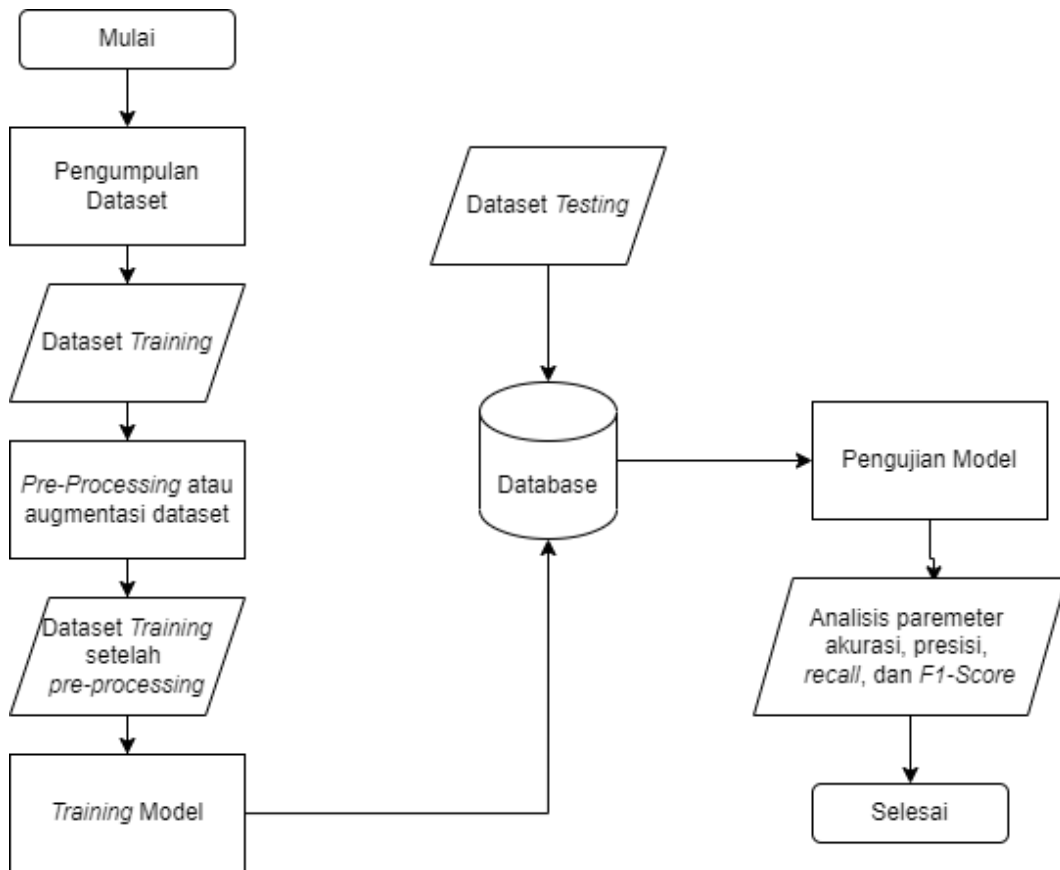
*Zoom* merupakan teknik yang memperbesar citra data sehingga didekatkan ke objek yang ingin diklasifikasi dengan skala tertentu.

- **Shear**

*Shear* merupakan teknik yang mendistorsikan citra data ke sumbu tertentu sehingga mendapatkan sudut pandang yang berbeda pada citra data dari sudut yang berbeda.

### 3.5 Pelatihan dan Pengujian Model

Sebelum melakukan pelatihan, dataset *training* yang akan digunakan sebagai data latih model akan melalui tahap proses *pre-processing* atau augmentasi citra data. Selanjutnya setelah selesai melalui tahap *pre-processing*, model akan melakukan *training* yang nantinya akan menghasilkan model yang sudah dilatih agar disimpan di *database*. Setelah selesai melalui tahap *training model* dan masuk kedalam *database*, model akan melalui tahap pengujian dengan melakukan klasifikasi terhadap dataset citra data uji yang belum diketahui oleh model yang sudah dilatih. Setelah melakukan tahap pengujian, model akan dianalisis melalui parameter pengujian yaitu akurasi, presisi, *recall*, dan *F1-Score*.



**Gambar 3.3** Flowchart proses pelatihan dan pengujian.

Proses ini berlaku untuk keempat arsitektur CNN yang digunakan dan akan dilakukan perubahan terhadap *hyperparameter* berupa *learning rate* dan *batch size* dari setiap pelatihan yang akan dilakukan dan dianalisis.

### 3.6 Parameter Pemanding Performansi Model

Parameter pembanding dari performansi model CNN dapat dilihat dari nilai parameter pada akurasi, presisi, *recall*, dan *F1-Score*. Untuk mendapatkan nilai parameter tersebut, diperlukannya *confusion matrix* di mana berisi nilai *True positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) [34]. Untuk penjelasan masing-masing nilai *confusion matrix* sebagai berikut:

- ***True positive* (TP)**  
*True positive* didapatkan ketika nilai yang diprediksi benar memang benar.
- ***True Negative* (TN)**  
*True negative* didapatkan ketika nilai yang diprediksi salah memang salah.

- **False Positive (FP)**  
*False positive* didapatkan ketika nilai yang diprediksi benar ternyata salah.
- **False Negative (FN)**  
*False negative* didapatkan ketika nilai yang diprediksi salah ternyata benar.  
Contoh *confusion matrix* dapat dilihat di Tabel 3.1 [35].

**Tabel 3.1** *Confusion matrix* lima kelas.

		Prediksi				
		CBB	CBSD	CMD	CGM	Healthy
Aktual	CBB	$x_{11}$	$x_{12}$	$x_{13}$	$x_{14}$	$x_{15}$
	CBSD	$x_{21}$	$x_{22}$	$x_{23}$	$x_{24}$	$x_{25}$
	CMD	$x_{31}$	$x_{32}$	$x_{33}$	$x_{34}$	$x_{35}$
	CGM	$x_{41}$	$x_{42}$	$x_{43}$	$x_{44}$	$x_{45}$
	Healthy	$x_{51}$	$x_{52}$	$x_{53}$	$x_{54}$	$x_{55}$

Dari *confusion matrix* tersebut, kita dapat mengevaluasi nilai dari *False Negative* (FN), *False Positive* (FP), *True Negative* (TN), dan *True positive* (TP) [35]:

$$FN_i = \sum_{j=1, j \neq i}^5 x_{ij} \quad (3.1)$$

$$FP_i = \sum_{j=1, j \neq i}^5 x_{ji} \quad (3.2)$$

$$TN_i = \sum_{k=1, k \neq i}^5 x_{jk} \quad (3.3)$$

$$TP_{all} = \sum_{j=1}^5 x_{jj} \quad (3.4)$$

Keterangan:

i = kelas yang di prediksi;

j = kelas actual;

k = kelas actual selain kelas yang prediksi;



### 3.4.1 Akurasi

Parameter akurasi digunakan sebagai nilai yang menentukan kebenaran suatu model dalam mengklasifikasi suatu kelas dengan melakukan perbandingan terhadap rasio prediksi yang benar [34]. Akurasi adalah metrik yang berguna ketika *error* dalam memprediksi semua kelas sama pentingnya [34]. Perhitungan akurasi dapat dilakukan menggunakan rumus pada persamaan 3.5.

$$Akurasi = \frac{(TTP_{all} + TTN_{all})}{Total\ Number\ of\ Testing\ Entries} \times 100\% \quad (3.5)$$

### 3.4.2 Presisi

Parameter presisi digunakan sebagai nilai yang menunjukkan seberapa benar suatu nilai benar atau positif yang dihasilkan [34]. Presisi sendiri berupa rasio prediksi positif yang benar (*true*) dengan jumlah keseluruhan prediksi positif seperti yang ada pada persamaan 3.6.

$$presisi = \frac{TTP_{all}}{TTP_{all} + TFP_i} \times 100\% \quad (3.6)$$

### 3.4.3 Recall

Parameter *recall* digunakan untuk mendapatkan nilai rasio prediksi positif yang benar (*true*) dengan jumlah keseluruhan contoh positif dalam kumpulan data [34]. Untuk mencari nilai *recall* dapat menggunakan persamaan 3.7.

$$Recall = \frac{TTP_{all}}{TTP_{all} + TFN_i} \times 100\% \quad (3.7)$$

### 3.4.4 F1-Score

Parameter *F1-Score* digunakan untuk mencari nilai rata-rata dari presisi dan *recall* [25]. Untuk persamaan *F1-Score* bisa dilihat dipersamaan 3.8.

$$F1 - score = 2 \frac{presisi \times recall}{presisi + recall} \times 100\% \quad (3.8)$$

### 3.4.5 Loss Function

Parameter *loss* berfungsi untuk mengetahui seberapa besar *error rate* atau *loss* yang dilakukan oleh model *machine learning*. Fungsi *loss* yang digunakan

dalam dalam penelitian ini yaitu *crossentropy* di mana cocok untuk nilai *output* yang berupa probabilitas [34].

### 3.7 Perancangan Skenario Pengujian

Penelitian ini menggunakan data citra daun penyakit tanaman singkong yang dikategorikan menjadi dua jenis, data pertama yaitu data citra asli yang merupakan data citra sebelum melalui *pre-processing* dan *balancing* data, lalu data kedua adalah data citra yang sudah melalui *pre-processing* dan juga *balancing* data citra. Penulis menggunakan tiga skenario yang akan diuji ke dua jenis data citra tersebut. *Hyperparameter* yang diubah dalam pengujian berupa:

1. *Optimizer*

*Optimizer* yang digunakan yaitu Adam, Nadam, dan RMSProp.

2. *Learning rate*

*Learning rate* yang digunakan yaitu 0.01, 0.001, dan 0.0001.

3. *Batch size*

*Batch size* yang digunakan yaitu 16, 32, dan 64.

Skenario yang digunakan seperti berikut:

1. Skenario pertama, yaitu mencari *optimizer* terbaik dengan *hyperparameter* awal yang sudah ditentukan yaitu *learning rate* 0.001, *batch size* 32, *epoch* 30, dan *input size*  $224 \times 224$  pixels.
2. Skenario kedua, setelah mendapatkan *optimizer* terbaik, selanjutnya mencari *learning rate* terbaik dari hasil skenario pertama.
3. Skenario ketiga, mendapatkan *batch size* terbaik setelah mendapatkan *optimizer* dan *learning rate* terbaik tiap *batch size* di skenario pertama dan kedua.

Agar dapat mengetahui hasil dari skenario terbaik dapat dilihat melalui parameter grafik hasil akurasi dan grafik hasil *loss*. Nilai parameter dari akurasi, presisi, *recall*, *loss*, dan F1-Score juga digunakan untuk menentukan hasil terbaik dari setiap skenario.

## **BAB IV**

### **HASIL DAN ANALISIS**

Pada bab ini dilakukan pemaparan analisis dan hasil yang didapatkan terhadap klasifikasi penyakit tanaman singkong menggunakan *Convolutional Neural Network* (CNN) dengan arsitektur MobileNet, MobileNet V2, MobileNet V3, dan CropNet. Capaian yang didapatkan terhadap bab ini berupa hasil akurasi, presisi, *loss*, *recall*, dan *F1-Score*.

#### **4.1 Pengujian Model**

Pengujian model yang dilakukan dimulai dari proses *training* data validasi data. Proses training data sendiri dimaksudkan untuk melatih model dari arsitektur yang digunakan yaitu MobileNet, MobileNet V2, MobileNet V3, dan CropNet menggunakan dataset yang sudah disediakan sebelumnya. Proses validasi data sendiri berupa pengujian model menggunakan data validasi agar mendapatkan nilai parameter validasi akurasi dan validasi *loss* untuk mengetahui apakah model yang di latih mengalami *overfitting*, *underfitting*, atau normal. Pengujian model bertujuan untuk mendapatkan nilai performa terbaik dari arsitektur dan juga parameter-parameter yang akan digunakan. Pengujian model menggunakan *library* Keras yang disediakan bahasa pemrograman *python* dengan *tensorflow* yang dilakukan di *Google Colaboratoy*.

#### **4.2 Hasil dan Analisis Pengujian**

Hasil pengujian klasifikasi penyakit pada tanaman singkong menggunakan empat jenis arsitektur yaitu MobileNet, MobileNet V2, MobileNet V3, dan CropNet yang masing-masing di uji berdasarkan tiga skenario pengujian dan dua jenis dataset.

#### **4.3 Performansi Akurasi Model dengan Arsitektur MobileNet**

Model pertama yang digunakan yaitu MobileNet, pelatihan model ini menggunakan dua jenis dataset yaitu data citra asli dan data *balanced*, serta melalui tiga skenario.

#### 4.4.1 Data *Unbalanced*

Pengujian pertama menggunakan data citra asli yang merupakan dataset yang tidak melalui *balancing*, *pre-processing* atau augmentasi apapun. *Hyperparameter* yang digunakan dalam pengujian skenario yaitu *optimizer* Adam, Nadam, dan RMSProp lalu menggunakan masing-masing *learning rate* 0.01, 0.001, dan 0.0001 lalu menggunakan masing-masing *batch size* 16, 32, dan 64. Skenario pertama menggunakan hasil skenario pertama yaitu dengan mencari hasil *optimizer* terbaik di tiap masing-masing *learning rate* dan *batch size*. Skenario kedua dilanjutkan menggunakan hasil skenario pertama yaitu dengan mencari hasil *learning rate* terbaik di tiap masing-masing *batch size*. Skenario ketiga menentukan hasil terbaik arsitektur yang digunakan yaitu dengan mencari hasil *batch size* terbaik setelah skenario kedua dijalankan.

##### 4.4.1.1 *Optimizer*

Skenario pertama mencari hasil *optimizer* terbaik di *learning rate* 0.001 dan *batch size* 32. Hasil dari pengujian dapat dilihat pada Tabel 4.1.

**Tabel 4.1** Hasil pengaruh *optimizer* ketika menggunakan data citra asli.

<i>Optimizer</i>	<i>Learning rate</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
Adam	0.001	32	84.34	77.5	0.4381	0.738
Nadam			83.17	76.26	0.4622	0.748
RMSProp			78.85	71.83	0.5778	0.9466

Seperti yang dilihat pada **Tabel 4.1**, terlihat *optimizer* terbaik yang didapatkan yaitu Adam dengan nilai akurasi sebesar 84.34%, dapat dilihat jika *optimizer* yang lain memiliki perbandingan akurasi sebesar 1 hingga 5 %, dapat dikatakan jika hasil skenario ini berpengaruh terhadap dataset ini. Dengan hasil yang didapatkan, maka *optimizer* Adam digunakan di skenario selanjutnya.

#### 4.4.1.2 Learning rate

Skenario kedua dilanjutkan menggunakan hasil skenario pertama yaitu dengan mencari hasil *learning rate* terbaik. Hasil dari pengujian dapat dilihat pada Tabel 4.2.

**Tabel 4.2** Hasil pengaruh *learning rate* ketika menggunakan data citra asli.

<i>Learning rate</i>	<i>Optimizer</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
0.01	Adam	32	80.55	76.71	0.8875	1.9007
0.001			84.34	77.5	0.4381	0.738
0.0001			76.43	73.07	0.645	0.7635

Seperti yang dilihat pada Tabel 4.2, terlihat *learning rate* terbaik yang didapatkan yaitu 0.001 di mana masih sama dengan hasil skenario pertama yaitu dengan akurasi sebesar 84.34%, dapat dilihat jika *learning rate* yang lain memiliki perbandingan akurasi sebesar 3 hingga 8 % dibandingkan dengan *learning rate* 0.001, dimana semakin besar atau semakin kecil *learning rate* yang digunakan maka akurasi yang didapatkan juga semakin kecil. Dengan hasil yang didapatkan, maka *learning rate* 0.001 digunakan di skenario selanjutnya.

#### 4.4.1.3 Batch size

Skenario ketiga menentukan hasil terbaik arsitektur yang digunakan yaitu dengan mencari hasil *batch size* terbaik setelah skenario dua dijalankan. Hasil dari pengujian dapat dilihat pada Tabel 4.3.

**Tabel 4.3** Pengaruh *batch size* ketika menggunakan data citra asli.

<i>Batch size</i>	<i>Optimizer</i>	<i>Learning rate</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
16	Adam	0.001	84.21	75.47	0.4406	0.7705
32			84.34	77.5	0.4381	0.738
64			83.25	77.86	0.4627	0.7909

Seperti yang dilihat pada Tabel 4.3, terlihat *batch size* terbaik yang didapatkan yaitu 32 dengan akurasi sebesar 84.34%, dapat dilihat jika *batch size* yang lain memiliki perbandingan akurasi sebesar 0.1 hingga 1 %, akurasi dari *batch size* 16 sendiri lebih kecil 0.1 % dibandingkan *batch size* 32. Dengan hasil yang didapatkan, maka *batch size* 32 digunakan sebagai hasil akhir untuk skenario ini.

#### **4.4.1.4 Hasil Simulasi Menggunakan Data *Unbalanced***

Hasil simulasi terbaik yang didapatkan dengan dataset ini berdasarkan tiga skenario yang sudah dijalankan yaitu dengan menggunakan *hyperparameter*:

1. *Learning rate* : 0.001
2. *Batch size* : 32
3. *Optimizer* : Adam

#### **4.4.2 Data *Balanced***

Pengujian kedua menggunakan data *balanced* yang merupakan dataset yang sudah melalui *balancing*, *pre-processing* dan augmentasi. *Balancing* yang dilakukan berupa *oversampling* dan *undersampling* mana membuat dataset menjadi 1000 sampel di tiap kelas penyakit tanaman singkong. Augmentasi yang digunakan berupa *resize* citra data menjadi  $224 \times 224$  *pixels*, *flipping* citra data menjadi vertikal dan horizontal, *zoom* citra data sebesar 15%, dan *shear* citra data sebesar 15 derajat. *Hyperparameter* yang digunakan dalam pengujian skenario yaitu *optimizer* Adam, Nadam, dan RMSprop lalu menggunakan masing-masing *learning rate* 0.01, 0.001, dan 0.0001 lalu menggunakan masing-masing *batch size* 16, 32, dan 64.

##### **4.4.2.1 *Optimizer***

Skenario pertama mencari hasil *optimizer* terbaik di *learning rate* 0.001 dan *batch size* 32. Hasil dari pengujian dapat dilihat pada Tabel 4.4.

**Tabel 4.4** Hasil pengaruh *optimizer* ketika menggunakan data *balanced*.

<i>Optimizer</i>	<i>Learning rate</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
Adam	0.001	32	82.57	71.8	0.4677	0.8073
Nadam			83	72.4	0.4556	0.8336
RMSProp			78.86	70.3	0.5685	0.932

Seperti yang dilihat pada Tabel 4.4, terlihat *optimizer* terbaik yang didapatkan yaitu Nadam dengan akurasi sebesar 83%, dapat dilihat jika *optimizer* yang lain memiliki perbandingan akurasi sebesar 1 hingga 5%, dapat dikatakan jika hasil skenario ini berpengaruh terhadap dataset ini. Dengan hasil yang didapatkan, maka *optimizer* Nadam digunakan di skenario selanjutnya.

#### 4.4.2.2 *Learning rate*

Skenario kedua dilanjutkan menggunakan hasil skenario pertama yaitu dengan mencari hasil *learning rate* terbaik.

**Tabel 4.5** Hasil pengaruh *learning rate* ketika menggunakan data *balanced*.

<i>Learning rate</i>	<i>Optimizer</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
0.01	Nadam	32	79.86	69.4	1.2069	2.5467
0.001			83	72.4	0.4556	0.8336
0.0001			71.77	69.1	0.7658	0.8489

Seperti yang dilihat pada Tabel 4.5, terlihat *learning rate* terbaik yang didapatkan yaitu 0.001 dengan akurasi sebesar 83%, dapat dilihat jika *learning rate* yang lain memiliki perbandingan akurasi sebesar 3 hingga 11 % dimana semakin besar atau semakin kecil *learning rate* yang digunakan maka akurasi yang didapatkan juga semakin kecil. Dengan hasil yang didapatkan, maka *learning rate* 0.001 digunakan di skenario selanjutnya.

#### 4.4.2.3 Batch size

Skenario ketiga menentukan hasil terbaik arsitektur yang digunakan yaitu dengan mencari hasil *batch size* terbaik setelah skenario dua dijalankan.

**Tabel 4.6** Pengaruh *batch size* ketika menggunakan data *balanced*.

<i>Batch size</i>	<i>Optimizer</i>	<i>Learning rate</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
16	Nadam	0.001	83.03	71.6	0.4662	0.783
32			83	72.4	0.4556	0.8336
64			79.74	73.5	0.554	0.7508

Seperti yang dilihat pada Tabel 4.6, terlihat *batch size* terbaik yang didapatkan yaitu 16 dengan akurasi 83.03%, dapat dilihat jika *batch size* yang lain memiliki perbandingan akurasi sebesar 0 hingga 3 %, akurasi dari *batch size* 32 sendiri lebih kecil 0.03 % dibandingkan *batch size* 16. Dengan hasil yang didapatkan, maka *batch size* 16 digunakan sebagai hasil akhir untuk skenario ini.

#### 4.4.2.4 Hasil Simulasi Menggunakan Data *Balanced*

Hasil simulasi terbaik yang didapatkan dengan dataset ini berdasarkan tiga skenario yang sudah dijalan yaitu dengan menggunakan *hyperparameter*:

4. *Learning rate* : 0.001
5. *Batch size* : 16
6. *Optimizer* : Nadam

#### 4.4 Performansi Akurasi Model dengan Arsitektur MobileNet V2

Model kedua yang digunakan yaitu MobileNet V2, pelatihan model ini menggunakan dua jenis dataset yaitu data citra asli dan data *balanced* dan augmentasi, serta melalui tiga skenario.



#### 4.5.1 Data Unbalanced

##### 4.5.1.1 Optimizer

Skenario pertama mencari hasil *optimizer* terbaik di *learning rate* 0.001 dan *batch size* 32. Hasil dari pengujian dapat dilihat pada Tabel 4.7.

**Tabel 4.7** Hasil pengaruh *optimizer* ketika menggunakan data citra asli.

<i>Optimizer</i>	<i>Learning rate</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
Adam	0.001	32	66.35	60.5	0.8922	1.0825
Nadam			60.94	53.8	0.9798	1.1726
RMSProp			61.32	56.95	1.0143	1.1228

Seperti yang dilihat pada Tabel 4.7, terlihat *optimizer* terbaik yang didapatkan yaitu Adam dengan akurasi sebesar 66.35%, dapat dilihat jika *optimizer* yang lain memiliki perbandingan akurasi sebesar 5 hingga 6 %, dapat dikatakan jika hasil skenario ini berpengaruh terhadap dataset ini. Dengan hasil yang didapatkan, maka *optimizer* Adam digunakan di skenario selanjutnya.

##### 4.5.1.2 Learning rate

Skenario kedua dilanjutkan menggunakan hasil skenario pertama yaitu dengan mencari hasil *learning rate* terbaik di tiap masing-masing *batch size*.

**Tabel 4.8** Hasil pengaruh *learning rate* ketika menggunakan data citra asli.

<i>Learning rate</i>	<i>Optimizer</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
0.01	Adam	32	60.33	48.27	1.2695	1.5621
0.001			66.35	60.5	0.8922	1.0825
0.0001			62.46	60.14	0.9938	1.0548

Seperti yang dilihat pada Tabel 4.8, terlihat *learning rate* terbaik yang didapatkan yaitu 0.001 dengan akurasi sebesar 66.35%, dapat dilihat jika *learning rate* yang lain memiliki perbandingan akurasi sebesar 4 hingga 6 % dimana semakin besar atau semakin kecil *learning rate* yang digunakan maka akurasi yang

didapatkan juga semakin kecil. Dengan hasil yang didapatkan, maka *learning rate* 0.001 digunakan di skenario selanjutnya.

#### 4.5.1.3 *Batch size*

Skenario ketiga menentukan hasil terbaik arsitektur yang digunakan yaitu dengan mencari hasil *batch size* terbaik setelah skenario dua dijalankan.

**Tabel 4.9** Pengaruh *batch size* ketika menggunakan data citra asli.

<i>Batch size</i>	<i>Optimizer</i>	<i>Learning rate</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
16	Adam	0.001	64.91	55.89	0.9101	1.1329
32			66.35	60.5	0.8922	1.0825
64			66.3	61.29	0.8971	1.0354

Seperti yang dilihat pada Tabel 4.9, terlihat *batch size* terbaik yang didapatkan yaitu 32 dengan akurasi sebesar 66.35%, dapat dilihat jika *batch size* yang lain memiliki perbandingan akurasi sebesar 0.05% hingga 1.54%. Dengan hasil yang didapatkan, maka *batch size* 32 digunakan sebagai hasil akhir untuk skenario ini.

#### 4.5.1.4 Hasil Simulasi Menggunakan Data *Unbalanced*

Hasil simulasi terbaik yang didapatkan dengan dataset ini berdasarkan tiga skenario yang sudah dijalan yaitu dengan menggunakan *hyperparameter*:

1. *Learning rate* : 0.001
2. *Batch size* : 32
3. *Optimizer* : Adam

### 4.5.2 *Data Balanced*

#### 4.5.2.1 *Optimizer*

Skenario pertama mencari hasil *optimizer* terbaik di *learning rate* 0.001 dan *batch size* 32. Hasil dari pengujian dapat dilihat pada Tabel 4.10.

**Tabel 4.10** Hasil pengaruh *optimizer* ketika menggunakan data *balanced*.

<i>Optimizer</i>	<i>Learning rate</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
Adam	0.001	32	61.26	54.3	0.9756	1.1112
Nadam			61.06	55.8	0.9835	1.1189
RMSProp			56.69	44.6	1.083	1.3324

Seperti yang dilihat pada Tabel 4.10, terlihat *optimizer* terbaik yang didapatkan yaitu Adam dengan akurasi sebesar 61.26%, dapat dilihat jika *optimizer* yang lain memiliki perbandingan akurasi sebesar 0.002 hingga 3.457, dapat dikatakan jika hasil skenario ini berpengaruh terhadap dataset ini. Dengan hasil yang didapatkan, maka *optimizer* Adam digunakan di skenario selanjutnya.

#### 4.5.2.2 *Learning rate*

Skenario kedua dilanjutkan menggunakan hasil skenario pertama yaitu dengan mencari hasil *learning rate* terbaik di tiap masing-masing *batch size*.

**Tabel 4.11** Hasil pengaruh *learning rate* ketika menggunakan data *balanced*.

<i>Learning rate</i>	<i>Optimizer</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
0.01	Adam	32	58.77	51.6	1.1593	1.3456
0.001			61.26	54.3	0.9756	1.1112
0.0001			54.89	52.4	1.1167	1.137

Seperti yang dilihat pada Tabel 4.11, terlihat *learning rate* terbaik yang didapatkan yaitu 0.001 dengan akurasi sebesar 61.26%, dapat dilihat jika *learning rate* yang lain memiliki perbandingan akurasi sebesar 2.49 hingga 6.37 % dimana semakin besar atau semakin kecil *learning rate* yang digunakan maka akurasi yang didapatkan juga semakin kecil. Dengan hasil yang didapatkan, maka *learning rate* 0.001 digunakan di skenario selanjutnya.

#### 4.5.2.3 Batch size

Skenario ketiga menentukan hasil terbaik arsitektur yang digunakan yaitu dengan mencari hasil *batch size* terbaik setelah skenario dua dijalankan.

**Tabel 4.12** Pengaruh *batch size* ketika menggunakan data *balanced*.

<i>Batch size</i>	<i>Optimizer</i>	<i>Learning rate</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
16	Adam	0.001	61.57	53.7	0.9754	1.1304
32			61.26	54.3	0.9756	1.1112
64			78.94	73.9	0.5794	0.7331

Seperti yang dilihat pada Tabel 4.12, terlihat *batch size* terbaik yang didapatkan yaitu 64 dengan akurasi 78.94 %, dapat dilihat jika *batch size* yang lain memiliki perbandingan akurasi sebesar 17.37 hingga 17.68 % dimana semakin besar jumlah *batch size* akurasi yang didapatkan semakin tinggi pula. Dengan hasil yang didapatkan, maka *batch size* 64 digunakan sebagai hasil akhir untuk skenario ini.

#### 4.5.2.4 Hasil Simulasi Dataset *Balanced*

Hasil simulasi terbaik yang didapatkan dengan dataset ini berdasarkan tiga skenario yang sudah dijalan yaitu dengan menggunakan *hyperparameter*:

1. *Learning rate* : 0.001
2. *Batch size* : 64
3. *Optimizer* : Adam

### 4.5 Performansi Akurasi Model dengan Arsitektur MobileNet V3

Model ketiga yang digunakan yaitu MobileNet V3, pelatihan model ini menggunakan dua jenis dataset yaitu data citra asli dan data *balanced*, serta melalui tiga skenario.

#### 4.6.1 Data Unbalanced

##### 4.6.1.1 Optimizer

Skenario pertama mencari hasil *optimizer* terbaik di *learning rate* 0.001 dan *batch size* 32. Hasil dari pengujian dapat dilihat pada Tabel 4.13.

**Tabel 4.13** Hasil pengaruh *optimizer* ketika menggunakan data citra asli.

<i>Optimizer</i>	<i>Learning rate</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
Adam	0.001	32	82.87	77.68	0.4747	0.6642
Nadam			83.35	77.15	0.4713	0.6764
RMSProp			81.53	74.4	0.5036	0.7316

Seperti yang dilihat pada Tabel 4.13, terlihat *optimizer* terbaik yang didapatkan yaitu Nadam dengan akurasi sebesar 83.35%, dapat dilihat jika *optimizer* yang lain memiliki perbandingan akurasi sebesar 0.48 hingga 1.82 %, dapat dikatakan jika hasil skenario ini berpengaruh terhadap dataset ini. Dengan hasil yang didapatkan, maka *optimizer* Nadam digunakan di skenario selanjutnya.

##### 4.6.1.2 Learning rate

Skenario kedua dilanjutkan menggunakan hasil skenario pertama yaitu dengan mencari hasil *learning rate* terbaik di tiap masing-masing *batch size*.

**Tabel 4.14** Hasil pengaruh *learning rate* ketika menggunakan data citra asli.

<i>Learning rate</i>	<i>Optimizer</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
0.01	Nadam	32	78.93	75.64	0.8311	1.3518
0.001			83.35	77.15	0.4713	0.6764
0.0001			76.5	74.31	0.6585	0.7045

Seperti yang dilihat pada Tabel 4.14, terlihat *learning rate* terbaik yang didapatkan yaitu 0.001 dengan akurasi sebesar 83.35%, dapat dilihat jika *learning rate* yang lain memiliki perbandingan akurasi sebesar 4.42 hingga 6.85 % dimana semakin besar atau semakin kecil *learning rate* yang digunakan maka akurasi yang

didapatkan juga semakin kecil. Dengan hasil yang didapatkan, maka *learning rate* 0.001 digunakan di skenario selanjutnya.

#### 4.6.1.3 *Batch size*

Skenario ketiga menentukan hasil terbaik arsitektur yang digunakan yaitu dengan mencari hasil *batch size* terbaik setelah skenario dua dijalankan.

**Tabel 4.15** Pengaruh *batch size* ketika menggunakan data citra asli.

<i>Batch size</i>	<i>Optimizer</i>	<i>Learning rate</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
16	Nadam	0.001	83.02	75.82	0.4761	0.6862
32			83.35	77.15	0.4713	0.6764
64			82.74	76.35	0.4752	0.6779

Seperti yang dilihat pada Tabel 4.15, terlihat *batch size* terbaik yang didapatkan yaitu 32 dengan akurasi sebesar 83.35%, dapat dilihat jika *batch size* yang lain memiliki perbandingan akurasi sebesar 0.33 hingga 0.61 %. Dengan hasil yang didapatkan, maka *batch size* 32 digunakan sebagai hasil akhir untuk skenario ini.

#### 4.6.1.4 Hasil Simulasi Menggunakan Data *Unbalanced*

Hasil simulasi terbaik yang didapatkan dengan dataset ini berdasarkan tiga skenario yang sudah dijalan yaitu dengan menggunakan *hyperparameter*:

1. *Learning rate* : 0.001
2. *Batch size* : 32
3. *Optimizer* : Nadam

### 4.6.2 *Data Balanced*

#### 4.6.2.1 *Optimizer*

Skenario pertama mencari hasil *optimizer* terbaik di *learning rate* 0.001 dan *batch size* 32. Hasil dari pengujian dapat dilihat pada Tabel 4.16.

**Tabel 4.16** Hasil pengaruh *optimizer* ketika menggunakan data *balanced*.

<i>Optimizer</i>	<i>Learning rate</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
Adam	0.001	32	83.14	75.4	0.4708	0.6792
Nadam			83.54	76.9	0.4693	0.6457
RMSProp			81.83	77.1	0.4985	0.6634

Seperti yang dilihat pada Tabel 4.16, terlihat *optimizer* terbaik yang didapatkan yaitu Nadam dengan akurasi sebesar 83.54%, dapat dilihat jika *optimizer* yang lain memiliki perbandingan akurasi sebesar 0.4 hingga 1.71 %, dapat dikatakan jika hasil skenario ini berpengaruh terhadap dataset ini. Dengan hasil yang didapatkan, maka *optimizer* Nadam digunakan di skenario selanjutnya.

#### 4.6.2.2 *Learning rate*

Skenario kedua dilanjutkan menggunakan hasil skenario pertama yaitu dengan mencari hasil *learning rate* terbaik di tiap masing-masing *batch size*.

**Tabel 4.17** Hasil pengaruh *learning rate* ketika menggunakan data *balanced*.

<i>Learning rate</i>	<i>Optimizer</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
0.01	Nadam	32	79.17	70.9	0.7709	1.5185
0.001			83.54	76.9	0.4693	0.6457
0.0001			74.23	71.6	0.7053	0.7514

Seperti yang dilihat pada Tabel 4.17, terlihat *learning rate* terbaik yang didapatkan yaitu 0.001 dengan akurasi sebesar 83.54%, dapat dilihat jika *learning rate* yang lain memiliki perbandingan akurasi sebesar 4.37 hingga 9.31 % dimana semakin besar atau semakin kecil *learning rate* yang digunakan maka akurasi yang didapatkan juga semakin kecil. Dengan hasil yang didapatkan, maka *learning rate* 0.001 digunakan di skenario selanjutnya.

#### 4.6.2.3 Batch size

Skenario ketiga menentukan hasil terbaik arsitektur yang digunakan yaitu dengan mencari hasil *batch size* terbaik setelah skenario dua dijalankan.

**Tabel 4.18** Pengaruh *batch size* ketika menggunakan data *balanced*.

<i>Batch size</i>	<i>Optimizer</i>	<i>Learning rate</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
16	Nadam	0.001	82.77	77.1	0.4829	0.6495
32			83.54	76.9	0.4693	0.6457
64			82.86	76.1	0.4812	0.6631

Seperti yang dilihat pada Tabel 4.15, terlihat *batch size* terbaik yang didapatkan yaitu 32 dengan akurasi sebesar 83.54%, dapat dilihat jika *batch size* yang lain memiliki perbandingan akurasi sebesar 0.77 hingga 0.68 %. Dengan hasil yang didapatkan, maka *batch size* 32 digunakan sebagai hasil akhir untuk skenario ini.

#### 4.6.2.4 Hasil Simulasi Menggunakan Data *Balanced*

Hasil simulasi terbaik yang didapatkan dengan dataset ini berdasarkan tiga skenario yang sudah dijalan yaitu dengan menggunakan *hyperparameter*:

1. *Learning rate* : 0.001
2. *Batch size* : 32
3. *Optimizer* : Nadam

### 4.6 Performansi Akurasi Model dengan Arsitektur CropNet

Model keempat yang digunakan yaitu CropNet, pelatihan model ini menggunakan dua jenis dataset yaitu data citra asli dan data *balanced*, serta melalui tiga skenario.



#### 4.7.1 Data Unbalanced

##### 4.7.1.1 Optimizer

Skenario pertama mencari hasil *optimizer* terbaik di *learning rate* 0.001 dan *batch size* 32. Hasil dari pengujian dapat dilihat pada Tabel 4.19.

**Tabel 4.19** Hasil pengaruh *optimizer* ketika menggunakan data citra asli.

<i>Optimizer</i>	<i>Learning rate</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
Adam	0.001	32	86.56	88.04	0.5603	0.5356
Nadam			87.47	88.13	0.5297	0.5099
RMSProp			87.11	87.78	0.4954	0.4769

Seperti yang dilihat pada Tabel 4.19, terlihat *optimizer* terbaik yang didapatkan yaitu Nadam dengan akurasi sebesar 87.47%, dapat dilihat jika *optimizer* yang lain memiliki perbandingan akurasi sebesar 0.36 hingga 0.91 %, dapat dikatakan jika hasil skenario ini berpengaruh terhadap dataset ini. Dengan hasil yang didapatkan, maka *optimizer* Nadam digunakan di skenario selanjutnya.

##### 4.7.1.2 Learning rate

Skenario kedua dilanjutkan menggunakan hasil skenario pertama yaitu dengan mencari hasil *learning rate* terbaik di tiap masing-masing *batch size*.

**Tabel 4.20** Hasil pengaruh *learning rate* ketika menggunakan data citra asli.

<i>Learning rate</i>	<i>Optimizer</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
0.01	Nadam	32	86.84	88.13	0.4112	0.3753
0.001			87.47	88.13	0.5297	0.5099
0.0001			17.36	18.07	1.4385	1.4249

Seperti yang dilihat pada Tabel 4.20, terlihat *learning rate* terbaik yang didapatkan yaitu 0.001 dengan akurasi sebesar 87.47%, dapat dilihat jika *learning rate* yang lain memiliki perbandingan akurasi sebesar 0.63 hingga 70.11 % dimana semakin besar atau semakin kecil *learning rate* yang digunakan maka akurasi yang

didapatkan juga semakin kecil. Dengan hasil yang didapatkan, maka *learning rate* 0.001 digunakan di skenario selanjutnya.

#### 4.7.1.3 *Batch size*

Skenario ketiga menentukan hasil terbaik arsitektur yang digunakan yaitu dengan mencari hasil *batch size* terbaik setelah skenario dua dijalankan.

**Tabel 4.21** Pengaruh *batch size* ketika menggunakan data citra asli.

<i>Batch size</i>	<i>Optimizer</i>	<i>Learning rate</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
16	Nadam	0.001	86.94	88.22	0.5589	0.5339
32			87.47	88.13	0.5297	0.5099
64			87.01	89.19	0.5402	0.506

Seperti yang dilihat pada Tabel 4.21, terlihat *batch size* terbaik yang didapatkan yaitu 32 dengan akurasi sebesar 87.47%, dapat dilihat jika *batch size* yang lain memiliki perbandingan akurasi sebesar 0.46 hingga 0.53 %. Dengan hasil yang didapatkan, maka *batch size* 32 digunakan sebagai hasil akhir untuk skenario ini.

#### 4.7.1.4 Hasil Simulasi Menggunakan Data *Unbalanced*

Hasil simulasi terbaik yang didapatkan dengan dataset ini berdasarkan tiga skenario yang sudah dijalan yaitu dengan menggunakan *hyperparameter*:

1. *Learning rate* : 0.001
2. *Batch size* : 32
3. *Optimizer* : Nadam

### 4.7.2 *Data Balanced*

#### 4.7.2.1 *Optimizer*

Skenario pertama mencari hasil *optimizer* terbaik di *learning rate* 0.001 dan *batch size* 32. Hasil dari pengujian dapat dilihat pada Tabel 4.22.

**Tabel 4.22** Hasil pengaruh *optimizer* ketika menggunakan data *balanced*.

<i>Optimizer</i>	<i>Learning rate</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
Adam	0.001	32	81.69	83.7	0.7328	0.6931
Nadam			81.94	82.9	0.6812	0.6416
RMSProp			81.77	83.7	0.6328	0.584

Seperti yang dilihat pada Tabel 4.22, terlihat *optimizer* terbaik yang didapatkan yaitu Nadam dengan akurasi sebesar 81.94%, dapat dilihat jika *optimizer* yang lain memiliki perbandingan akurasi sebesar 0.17 hingga 0.25 %, dapat dikatakan jika hasil skenario ini berpengaruh terhadap dataset ini. Dengan hasil yang didapatkan, maka *optimizer* Nadam digunakan di skenario selanjutnya.

#### 4.7.2.2 *Learning rate*

Skenario kedua dilanjutkan menggunakan hasil skenario pertama yaitu dengan mencari hasil *learning rate* terbaik di tiap masing-masing *batch size*.

**Tabel 4.23** Hasil pengaruh *learning rate* ketika menggunakan data *balanced*.

<i>Learning rate</i>	<i>Optimizer</i>	<i>Batch size</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
0.01	Nadam	32	82.09	84.3	0.5599	0.5114
0.001			81.94	82.9	0.6812	0.6416
0.0001			48.91	49.4	1.3963	1.3919

Seperti yang dilihat pada Tabel 4.23, terlihat *learning rate* terbaik yang didapatkan yaitu 0.01 dengan akurasi sebesar 82.09%, dapat dilihat jika *learning rate* yang lain memiliki perbandingan akurasi sebesar 0.15 hingga 33.03 % dimana semakin kecil *learning rate* yang digunakan maka akurasi yang didapatkan juga semakin kecil. Dengan hasil yang didapatkan, maka *learning rate* 0.01 digunakan di skenario selanjutnya.

### 4.7.2.3 Batch size

Skenario ketiga menentukan hasil terbaik arsitektur yang digunakan yaitu dengan mencari hasil *batch size* terbaik setelah skenario dua dijalankan.

**Tabel 4.24** Pengaruh *batch size* ketika menggunakan data *balanced*.

<i>Batch size</i>	<i>Optimizer</i>	<i>Learning rate</i>	<i>Train Accuracy (%)</i>	<i>Val Accuracy (%)</i>	<i>Train Loss</i>	<i>Val Loss</i>
16	Nadam	0.01	81.86	84.2	0.5535	0.4955
32			82.09	84.3	0.5599	0.5114
64			81.97	85	0.5482	0.4864

Seperti yang dilihat pada Tabel 4.25, terlihat *batch size* terbaik yang didapatkan yaitu 32 dengan akurasi sebesar 87.47%, dapat dilihat jika *batch size* yang lain memiliki perbandingan akurasi sebesar 0.12 hingga 0.23 % di mana semakin besar atau kecil *batch size* yang digunakan maka akurasi semakin kecil. Dengan hasil yang didapatkan, maka *batch size* 32 digunakan sebagai hasil akhir untuk skenario ini.

### 4.7.2.4 Hasil Simulasi Menggunakan Data *Balanced*

Hasil simulasi terbaik yang didapatkan dengan dataset ini berdasarkan tiga skenario yang sudah dijalan yaitu dengan menggunakan *hyperparameter*:

1. *Learning rate* : 0.01
2. *Batch size* : 32
3. *Optimizer* : Nadam

#### 4.7 Analisa Hasil Pengujian

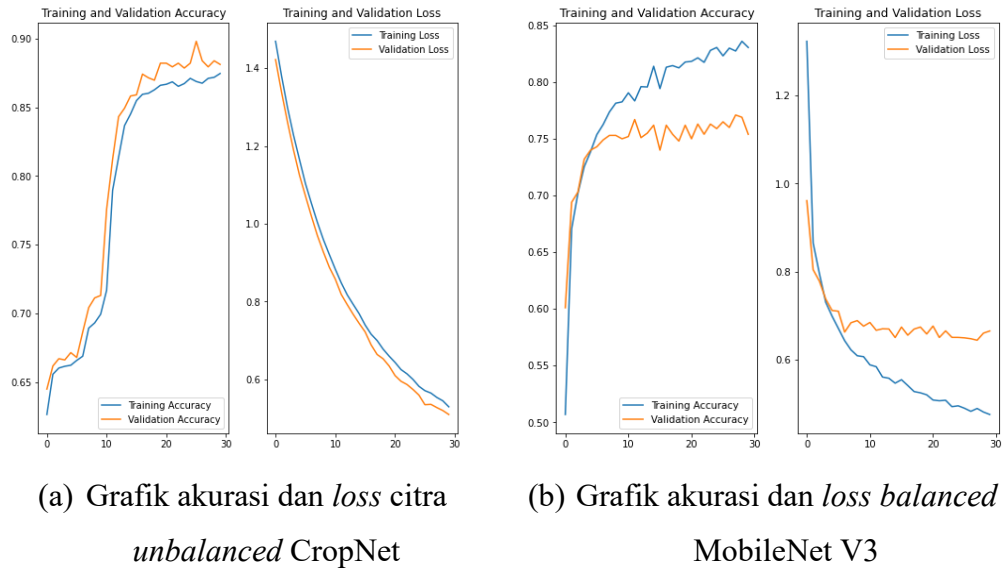
**Tabel 4.25** Hasil dan parameter terbaik pengujian di setiap arsitektur dan jenis data.

HYPERPARAMETER TERBAIK TIAP SKENARIO								
Arsitektur	MobileNet V1		MobileNet V2		MobileNet V3		CropNet	
Data	Citra Asli	<i>Balanced</i>	Citra Asli	<i>Balanced</i>	Citra Asli	<i>Balanced</i>	Citra Asli	<i>Balanced</i>
<i>Optimizer</i>	Adam	Nadam	Adam	Adam	Nadam	Nadam	Nadam	Nadam
<i>Learning rate</i>	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.01
<i>Batch size</i>	32	16	32	64	32	32	32	32
<i>Train Akurasi (%)</i>	84.34	83.03	66.35	78.94	83.35	83.54	87.47	82.09
<i>Val Akurasi (%)</i>	77.5	71.6	60.5	73.9	77.15	76.9	88.13	84.3
<i>Train Loss</i>	0.4381	0.4662	0.8922	0.5794	0.4713	0.4693	0.5297	0.5599
<i>Val Loss</i>	0.738	0.783	1.0825	0.7331	0.6764	0.6457	0.5099	0.5114

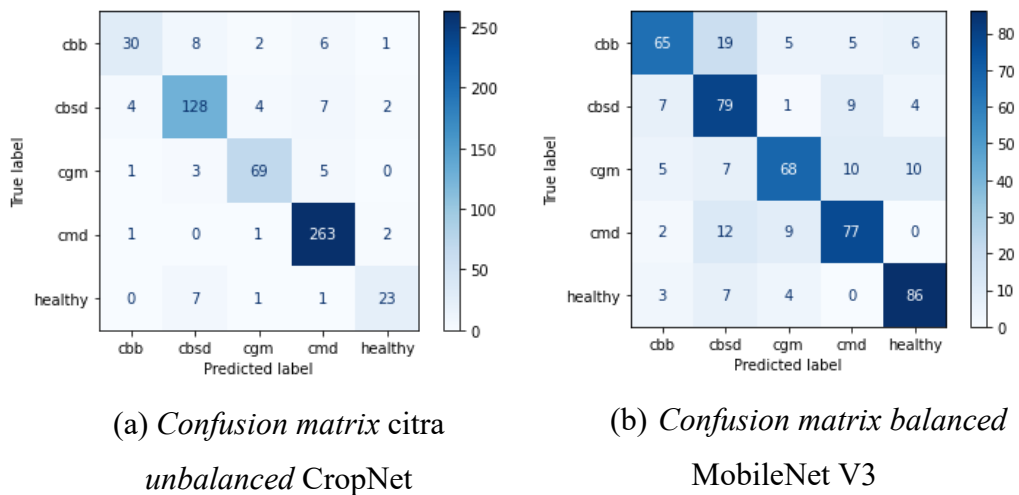
Terlihat pada Tabel 4.25 hasil terbaik yang didapatkan baik dari data citra asli maupun data *balanced*, hasil terbaik didapatkan oleh arsitektur CropNet dengan menggunakan data citra asli dan parameter *optimizer* Adam, *learning rate* 0.001, dan *batch size* 32. *Optimizer* Nadam mendominasi semua *optimizer* yang lain dalam mendapatkan hasil terbaik di setiap arsitektur, sedangkan untuk *learning rate* didominasi oleh *learning rate* sebesar 0.001 didalam setiap arsitektur, dan untuk *batch size* sendiri didominasi oleh *batch size* 32. Dapat disimpulkan jika *optimizer* Nadam, *learning rate* 0.001, dan *batch size* 32 sangat cocok digunakan dalam penggunaan proses *training* citra data daun singkong.

Dapat dilihat pada Tabel 4.25, untuk hasil terbaik menggunakan data citra asli yaitu menggunakan arsitektur CropNet menggunakan parameter *optimizer* Nadam, *learning rate* 0001, dan *batch size* 32 dengan hasil akurasi dan validasi terbaik dari semua arsitektur. Hasil akurasi yang didapatkan dari data citra asli yaitu sebesar 87,47% dan *loss* sebesar 0,5297. Hasil terbaik menggunakan data *balanced* yaitu menggunakan arsitektur MobileNet V3 menggunakan parameter *optimizer* Nadam,

learning rate 0.001, dan batch size 32 dengan hasil akurasi terbaik dari semua arsitektur. Hasil akurasi yang didapatkan dari data *balanced* yaitu sebesar 83.54% dan loss sebesar 0.4693 di mana hasil akurasi yang didapatkan dari data *balanced* cenderung lebih rendah dibandingkan data dari citra asli.



**Gambar 4.1** Grafik Akurasi dan *Loss* Citra Asli (a) dan *Balanced* (b).



**Gambar 4.2** *Confusion Matrix* Citra Asli (a) dan *Balanced* (b).

Dapat dilihat pada Gambar 4.1 perbandingan grafik antara penggunaan citra asli dan *balance* pada arsitektur CropNet dan MobileNet V3, terlihat jika grafik pada citra asli (a) cenderung selalu meningkat baik dari grafik akurasi dan grafik *loss*, sedangkan grafik pada data *balanced* (b) akurasi dan *loss* cenderung tidak

mengalami adanya peningkatan setelah melewati *epoch* ke 5 dan 10. Hal ini menandakan jika pada data *balanced*, ditemukannya homogenitas terhadap data yang sudah di augmentasi dapat menimbulkan kecenderungan model pada arsitektur ini untuk mempelajari hal yang sama berulang-ulang dengan tidak adanya perbedaan di setiap dataset sehingga model terjebak terhadap peningkatan akurasi dan *loss* pada *training* model. Akan tetapi, pada kedua jenis data ini dapat dilihat jika tidak terjadinya *overfitting* ataupun *underfitting* terhadap data ini sehingga model bisa dikatakan stabil.

Pada Gambar 4.2 *confusion matrix* yang dibuat menggunakan data *testing* dimana sebelumnya data tersebut menggunakan 10% dari keseluruhan data di setiap kelas data. Pada Gambar 4.2 *confusion matrix* yang dihasilkan citra asli (a) cenderung memiliki nilai *True positive* paling banyak melakukan kesalahan pada kelas CBB (46 citra data) sebesar 57% dan *healthy* (31 citra data) sebesar 78% dimana kedua kelas data tersebut adalah data yang paling sedikit jumlahnya dibandingkan kelas yang lain, dan kelas yang memiliki nilai *True positive* sedikit melakukan kesalahan didapatkan pada kelas CMD (265 citra data) dengan jumlah *True positive* sebesar 99% dimana kelas data tersebut adalah data yang paling banyak jumlahnya dibandingkan data kelas yang lain. Sedangkan data *balanced* terlihat *inferior* dibandingkan menggunakan data citra asli dimana kelas yang ada pada data *balanced* kecuali kelas CBB (100 citra data) sebesar 63% dan kelas *healthy* (100 citra data) sebesar 88% memiliki nilai *True positive* yang lebih rendah dibandingkan data citra asli, hal ini dikarenakan jumlah data pada kelas mayoritas (CBSD dan CMD) mengalami *undersampling* dan pada kelas minoritas (CBB, CGM, dan *healthy*) mengalami *oversampling*.

**Tabel 4.26** Jumlah data dan nilai *true positive* per kelas.

Kelas	Jumlah Data <i>Testing</i> Citra Asli CropNet	Jumlah Data <i>Testing</i> <i>Balanced</i> MobileNet V3	<i>True positive</i> Data Citra Asli CropNet	<i>True positive</i> Data <i>Balanced</i> MobileNet V3
CBB	47	100	57%	63%
CBSD	145	100	88%	69%
CGM	78	100	88%	74%
CMD	267	100	99%	84%
healthy	32	100	78%	88%

Dapat disimpulkan dari Tabel 4.27 pada analisis ini jika model terbaik pada penelitian ini yaitu menggunakan arsitektur CropNet dengan parameter uji *optimizer* Nadam, *learning rate* 0.001, dan *batch size* 32.

**Tabel 4.27** Nilai presisi, *recall*, dan F1-Score model terbaik.

Kelas	Jumlah Data <i>Testing</i>	Presisi	<i>Recall</i>	F1-Score
CBB	47	0.84	0.57	0.68
CBSD	145	0.87	0.88	0.88
CGM	78	0.87	0.88	0.88
CMD	267	0.94	0.99	0.96
healthy	32	0.83	0.78	0.81
	Rata-rata	0.87	0.82	0.842

Hasil yang didapatkan pada Tugas Akhir ini dapat disimpulkan memiliki hasil yang lebih baik dibandingkan dengan penelitian sebelumnya [6] di mana menggunakan arsitektur UNet dengan dataset yang berbeda, berikut perbandingan dari hasil yang didapatkan dari penelitian sebelumnya:

**Tabel 4.28** Perbedaan hasil akurasi Tugas Akhir dengan penelitian sebelumnya.

Model	Jumlah Dataset	Kelas	<i>Epoch</i>	Akurasi
CropNet	5656	5	30	87.47%
UNet	21397	2	4	83.9%

Dapat dilihat pada Tabel 4.28 jika hasil yang didapatkan dalam penelitian ini lebih baik dibandingkan dengan penelitian sebelumnya di mana penelitian sebelumnya hanya meneliti dua kelas penyakit dan menggunakan dataset yang lebih banyak, akan tetapi *epoch* yang digunakan lebih kecil kecil.



## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Pada penelitian Tugas Akhir ini penulis membuat dan merancang sistem dengan parameter awal yaitu menggunakan *epoch* sebesar 30, *input size* sebesar  $224 \times 224$  *pixels*, dan pembagian data menjadi 70% data *training*, 20% data validasi, dan 10% data *testing* untuk mengklasifikasi penyakit tanaman singkong menggunakan dua jenis citra data daun yaitu citra data asli dan citra data *balanced* dengan metode *convolutional neural network* menggunakan arsitektur MobileNet V1, MobileNet V2, MobileNet V3, dan CropNet. Dengan diperolehnya beberapa kesimpulan diantaranya sebagai berikut:

1. Model yang dibuat pada Tugas Akhir ini dapat mengklasifikasi lima kelas penyakit tanaman singkong.
2. *Balancing* dataset memiliki pengaruh terhadap peningkatan ataupun penurunan akurasi *training* tiap arsitektur, begitu pula dengan penurunan *loss training*, terbukti dari meningkatnya nilai *true positive* dari kelas jumlah data paling rendah yaitu CBB dan *healthy*, begitu pula dengan menurunnya nilai *true positive* pada kelas dengan jumlah data paling tinggi yaitu CBSD, CGM, dan CMD.
3. Model pengujian menggunakan data citra asli mendapatkan hasil terbaik dibandingkan dengan data *balanced* dengan menggunakan arsitektur CropNet dengan *hyperparameter optimizer* Adam, *learning rate* 0.001, dan *batch size* 32. Nilai akurasi dan validasi akurasi yang didapatkan yaitu secara berturut-turut 87.47% dan 88.13%.

#### **5.2 Saran**

Dengan didapatkannya hasil dari Tugas Akhir ini, masih ada beberapa kekurangan yang masih bisa dikembangkan, maka dari itu penulis memberi beberapa saran untuk membantu penelitian selanjutnya terkait topik Tugas Akhir ini, diantaranya adalah:

1. Mencari data yang lebih seimbang di tiap kelasnya tanpa menggunakan augmentasi data agar model dapat mempelajari secara langsung data asli yang ada kehidupan nyata.
2. Melakukan perubahan *hyperparameter* lain seperti *input size* dan penambahan jumlah *epoch*.
3. Menggunakan *optimizer* lain seperti SGD, Adadelta, Adagrad, dan lain-lain.
4. Menambah dataset dari sumber yang berbeda ke dalam dataset yang sudah ada.
5. Menggunakan filter seperti Gaussian pada data *oversampling* agar citra data dapat menjadi lebih halus, mengurangi ukuran citra data, dan membuat citra data menjadi lebih beragam jenisnya.

## DAFTAR PUSTAKA

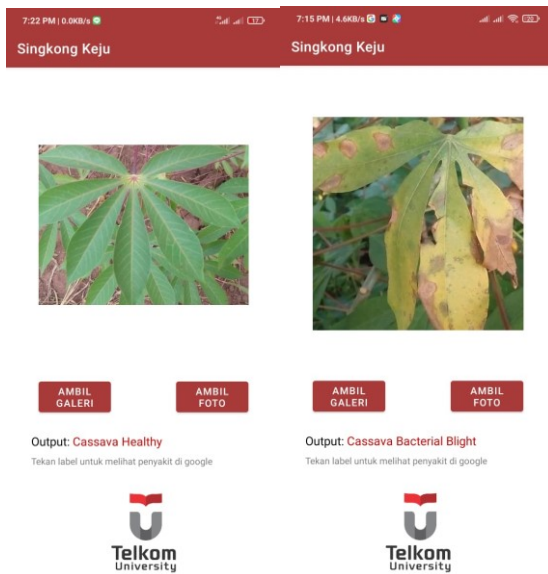
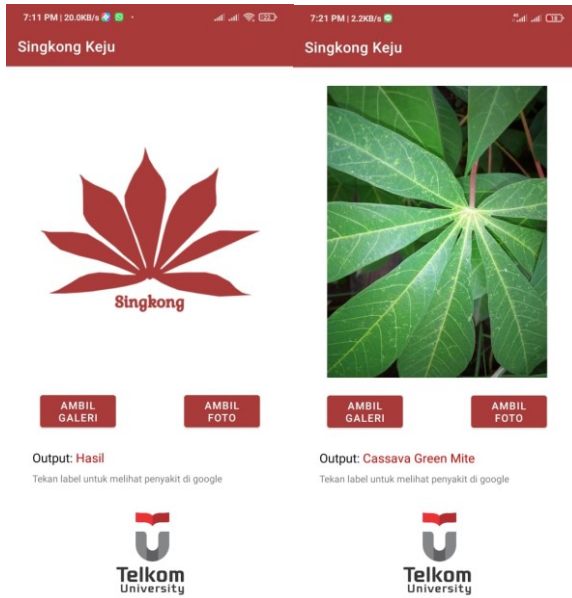
- [1] “Indonesia Negara Penghasil Singkong Terbanyak Keempat Dunia,” *Dinas Komun. dan Inform. Pemerintah Provinsi Jawa Timur*, 2021, [Daring]. Tersedia pada: <http://kominfo.jatimprov.go.id/read/umum/indonesia-negara-penghasil-singkong-terbanyak-keempat-dunia>.
- [2] “Produksi Ubi Kayu Menurut Provinsi (ton), 1993-2015,” *Badan Pusat Statistik*, 2015. <https://www.bps.go.id/linkTableDinamis/view/id/880>.
- [3] N. Saleh, M. Rahayu, S. W. Indiaty, B. S. Radjit, dan S. Wahyuningsih, “Hama, Penyakit, dan Gulma pada Tanaman Ubi Kayu,” *Badan Penelit. Dan Pengemb. Pertan. Kementeri. Pertan.*, hal. 48, 2013.
- [4] P. Ongsulee, “Artificial intelligence, machine learning and deep learning,” *Int. Conf. ICT Knowl. Eng.*, hal. 1–6, 2018, doi: 10.1109/ICTKE.2017.8259629.
- [5] S. Albawi, T. A. M. Mohammed, dan S. Alzawi, “Understanding of a Convolutional Neural Network,” *Ieee*, hal. 16, 2017.
- [6] P. K. Rao, R. S. Kumar, dan K. Sreenivasulu, “Cassava leaf disease classification using separable convolutions UNet,” *Turkish J. Comput. Math. Educ.*, vol. 12, no. 7, hal. 140–145, 2021.
- [7] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017, [Daring]. Tersedia pada: <http://arxiv.org/abs/1704.04861>.
- [8] H. Chen, B. Wang, T. Pan, L. Zhou, dan H. Zeng, “CropNet: Real-time thumbnailing,” *MM 2018 - Proc. 2018 ACM Multimed. Conf.*, hal. 81–89, 2018, doi: 10.1145/3240508.3240517.
- [9] S. E. Rikomah, E. Elmitra, dan D. G. Yunita, “EFEK EKSTRAK ETANOL DAUN SINGKONG (Manihot Utilissima Pohl) SEBAGAI OBAT ALTERNATIF ANTI REMATIK TERHADAP RASA SAKIT PADA MENCIT,” *J. Ilm. Manuntung*, vol. 3, no. 2, hal. 133, 2018, doi: 10.51352/jim.v3i2.119.
- [10] G. Howeler, Reinhardt; Lutaladio, NeBambi; Thomas, *Save and grow: Cassava*. FAO, 2013.
- [11] G. Jackson, “Cassava brown streak disease,” *Pacific Pests Pathog.*, no. 329, hal. 3–6, 2019.
- [12] M. Padidam, R. N. Beachy, dan C. M. Fauquet, “Classification and identification of geminiviruses using sequence comparisons,” *J. Gen. Virol.*, vol. 76, no. 2, hal. 249–263, 1995, doi: 10.1099/0022-1317-76-2-249.
- [13] P. C. Chikoti, R. M. Mulenga, M. Tembo, dan P. Sseruwagi, “Correction to: Cassava mosaic disease: a review of a threat to cassava production in Zambia

- (Journal of Plant Pathology, (2019), 101, 3, (467-477), 10.1007/s42161-019-00255-0),” *J. Plant Pathol.*, vol. 101, no. 3, hal. 479, 2019, doi: 10.1007/s42161-019-00267-w.
- [14] B. James *et al.*, *Starting a cassava farm*, no. 229. International Institute of Tropical Agriculture, 2000.
- [15] “Cassava (manioc),” *PlantVillage*, 2020. <https://plantvillage.psu.edu/topics/cassava-manioc/infos>.
- [16] D. Oliva, M. A. Elaziz, dan S. Hinojosa, *Metaheuristic Algorithms for Image Segmentation: Theory and Applications*. Springer Nature Switzerland AG, 2019.
- [17] A. Chahal dan P. Gulia, “Machine learning and deep learning,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 12, hal. 4910–4914, 2019, doi: 10.35940/ijitee.L3550.1081219.
- [18] R. Sathya dan A. Abraham, “Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification,” *Int. J. Adv. Res. Artif. Intell.*, vol. 2, no. 2, hal. 34–38, 2013, doi: 10.14569/ijarai.2013.020206.
- [19] H. Zhang dan T. Yu, *Deep Reinforcement Learning*. Springer Nature Singapore, 2020.
- [20] L. Chen, *Deep Learning and Practice with MindSpore*. Tsinghua University Press, 2021.
- [21] D. Saravanan, D. Joseph, dan S. Vaithyasubramanian, *Recent Trends and Advances in Artificial Intelligence and Internet of Things*, vol. 172. 2019.
- [22] M. Sarigül, B. M. Ozyildirim, dan M. Avcı, “Differential convolutional neural network,” *Neural Networks*, vol. 116, hal. 279–287, 2019, doi: 10.1016/j.neunet.2019.04.025.
- [23] A. Ajit, K. Acharya, dan A. Samanta, “A Review of Convolutional Neural Networks,” *Int. Conf. Emerg. Trends Inf. Technol. Eng. ic-ETITE 2020*, hal. 1–5, 2020, doi: 10.1109/ic-ETITE47903.2020.049.
- [24] T. Ho-Phuoc, “CIFAR10 to Compare Visual Recognition Performance between Deep Neural Networks and Humans,” 2018, [Daring]. Tersedia pada: <http://arxiv.org/abs/1811.07270>.
- [25] J. W. G. Putra, “Pengenalan konsep pembelajaran mesin dan deep learning,” *Comput. Linguist. Nat. Lang. Process. Lab.*, vol. 4, no. August, hal. 1–235, 2019, [Daring]. Tersedia pada: <https://www.researchgate.net/publication/323700644>.
- [26] M. Shaha dan M. Pawar, “Transfer Learning for Image Classification,” *Proc. 2nd Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2018*, no. Iceca, hal. 656–660, 2018, doi: 10.1109/ICECA.2018.8474802.

- [27] C. Bi, J. Wang, Y. Duan, B. Fu, J. R. Kang, dan Y. Shi, “MobileNet Based Apple Leaf Diseases Identification,” *Mob. Networks Appl.*, 2020, doi: 10.1007/s11036-020-01640-1.
- [28] S.-H. Tsang, “Review: MobileNetV1 — Depthwise Separable Convolution (Light Weight Model),” *Towards Data Science*, 2018. <https://towardsdatascience.com/review-mobilenetv1-depthwise-separable-convolution-light-weight-model-a382df364b69>.
- [29] M. Sandler, A. Howard, M. Zhu, dan A. Zhmoginov, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” hal. 4510–4520, 2018.
- [30] P. S. P. Kavyashree dan M. El-Sharkawy, “Compressed MobileNet V3:A Light Weight Variant for Resource-Constrained Platforms,” *2021 IEEE 11th Annu. Comput. Commun. Work. Conf. CCWC 2021*, hal. 104–107, 2021, doi: 10.1109/CCWC51732.2021.9376113.
- [31] G. M. Oktavian dan H. Santoso, “Leveraging MobileNet , InceptionV3 , and CropNet to Classify Cassava Plant Disease,” vol. 6, no. 2, hal. 183–193, 2021.
- [32] M. N. Halgamuge, E. Daminda, dan A. Nirmalathas, “Best optimizer selection for predicting bushfire occurrences using deep learning,” *Nat. Hazards*, 2020, doi: 10.1007/s11069-020-04015-7.
- [33] A. Indrawati, L. Ilmu, P. Indonesia, dan P. I. Diabetes, “PENERAPAN TEKNIK KOMBINASI OVERSAMPLING DAN UNDERSAMPLING HYBRID OVERSAMPLING AND UNDERSAMPLING TECHNIQUES TO HANDLING IMBALANCED DATASET,” vol. 4, no. 1, hal. 38–43, 2021, doi: 10.33387/jiko.
- [34] Andriy Burkov, *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.
- [35] N. K. C. Pratiwi, Y. N. Fu’adah, dan E. Edwar, “Early Detection of Deforestation through Satellite Land Geospatial Images based on CNN Architecture,” *J. Infotel*, vol. 13, no. 2, hal. 54–62, 2021, doi: 10.20895/infotel.v13i2.642.

# LAMPIRAN

## Tampilan aplikasi Mobile dan klasifikasi *Healthy*, CGM dan CBB.



## Klasifikasi Penyakit CBSD dan CMD di aplikasi Mobile

7:11 PM | 20.6KB/s | Singkong Keju

7:11 PM | 2.4KB/s | Singkong Keju

AMBIL GALERI

AMBIL FOTO

Output: **Cassava Brown Streak Disease**  
Tekan label untuk melihat penyakit di google

AMBIL GALERI

AMBIL FOTO

Output: **Cassava Mosaic Disease**  
Tekan label untuk melihat penyakit di google

Telkom University

Telkom University

## Source Code Arsitektur MobileNet dengan Dataset Unbalanced

```
# -*- coding: utf-8 -*-
"""MobileNet Original.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/1aq1CM1tWGINXL5vtm8uyJ0m
IWScCzB0w

# Load Dataset from Kaggle using Kaggle API Token
"""

from google.colab import files
files.upload()

! pip install kaggle
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

```

!kaggle competitions download -c cassava-disease

! unzip cassava-disease.zip
! mkdir train
! unzip train.zip -d train

"""# Data Collection and Preprocessing

## Prerequisites :
1. Tf.dataset Input Pipeline
2. Optimize Tensorflow Pipeline
3. Data Augmentation
"""

import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

import os
length = 0
lengths = {}
train_dir = '/content/train/train'
for folder in os.listdir(train_dir):
    lengths[folder] = len(os.listdir(os.path.join(train_dir,
folder)))
print(lengths)

import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
x = ['CMD', 'CGM', 'CBB', 'CBSD', 'Healthy']
y = [lengths['cmd'], lengths['cgm'], lengths['healthy'],
lengths['cbb'], lengths['cbstd']]
ax.set_title('Jumlah Gambar per Kelas', fontsize=18)
ax.bar(x, y)
plt.show()

!pip install split-folders

import splitfolders

splitfolders.ratio("/content/train/train", output="datasett",
seed=1337, ratio=(0.7, 0.2, 0.1),
group_prefix=None, move=False) # default values

TRAINING_DIR = '/content/datasett/train'
TEST_DIR = '/content/datasett/test'
VAL_DIR = '/content/datasett/val'
BATCH_SIZE_64 = 64
BATCH_SIZE_32 = 32

```



```

BATCH_SIZE_16 = 16
TARGET_SIZE = 224
EPOCHS = 30
CHANNELS = 3

from tensorflow.keras.preprocessing.image import
ImageDataGenerator

traingen = ImageDataGenerator(
    rescale = 1./255,
    # rotation_range = 180,
    # horizontal_flip = True,
    # vertical_flip = True
    # zoom_range = 0.3
)

valgen = ImageDataGenerator(
    rescale = 1./255
)

testgen = ImageDataGenerator(
    rescale = 1./255
)

train_ds = traingen.flow_from_directory(TRAINING_DIR,
    batch_size =
BATCH_SIZE_64,
    shuffle = True,
    target_size =
(TARGET_SIZE, TARGET_SIZE),
    class_mode =
'categorical')
test_ds = testgen.flow_from_directory(TEST_DIR,
    batch_size =
BATCH_SIZE_64,
    shuffle = True,
    target_size =
(TARGET_SIZE, TARGET_SIZE),
    class_mode =
'categorical')
val_ds = valgen.flow_from_directory(VAL_DIR,
    batch_size =
BATCH_SIZE_64,
    shuffle = True,
    target_size =
(TARGET_SIZE, TARGET_SIZE),
    class_mode =
'categorical')

class_names = list(train_ds.class_indices)
print(class_names)

n_classes = len(class_names)
print(n_classes)

for image_batch, label_batch in train_ds :

```

```

print(image_batch.shape)
print(label_batch[0])
print(image_batch[0])
break

train_ds.image_shape
val_ds.image_shape
test_ds.image_shape

import seaborn as sns
plt.figure(figsize=(20,8))
sns.countplot(x = train_ds.classes)

x,y = next(train_ds)

# function to plot images
def plotImages(x,y):
    plt.figure(figsize=(10,10))
    for i in range(16):
        plt.subplot(4,4,i+1)
        plt.imshow(x[i])
        plt.title(class_names[np.argmax(y[i])])
        plt.axis("off")
    plt.show()

plotImages(x,y)

!zip -r dataset_defisiensi.zip /content/dataset/

"""# Model Building

## Prerequisites :
1. Convolutional Neural Network
"""

!pip install --upgrade tensorflow_hub
!pip install tensorflow-addons

import tensorflow_addons as tfa
import tensorflow_hub as hub
from tensorflow.keras.callbacks import EarlyStopping,
ModelCheckpoint

pretrained_model =
'https://tfhub.dev/google/imagenet/mobilenet_v1_100_224/classifi
cation/5'

base_model = hub.KerasLayer(
                                pretrained_model,
                                input_shape = (224,224,3),
                                trainable = False
)

model = tf.keras.Sequential([

```

```

        base_model,
        tf.keras.layers.Flatten(),

tf.keras.layers.Dense(5,activation = 'softmax')
])

model.summary()

my_calls = [EarlyStopping(monitor="val_loss",patience=5),
            ModelCheckpoint("Model.h5",verbose= 1
,save_best_only=True)]

"""# ***MobileNet V1***

## **Batch Size 64**

### **Adam LR 0.01**
"""

METRICS = [
    tf.keras.metrics.CategoricalAccuracy(name =
'accuracy'),
    tf.keras.metrics.Recall(name = 'recall'),
    tf.keras.metrics.Precision(name = 'precision'),
    tfa.metrics.F1Score(num_classes=5)
]

model.compile(
    tf.keras.optimizers.Adam(learning_rate = 0.01),
    loss = tf.keras.losses.CategoricalCrossentropy(from_logits =
False),
    metrics = METRICS
)

history = model.fit(
    train_ds,
    epochs = EPOCHS,
    batch_size = BATCH_SIZE_64,
    verbose = 1,
    validation_data = val_ds,
    # callbacks = my_calls
)

model.save("Model.h5")

from sklearn.metrics import classification_report,
confusion_matrix, ConfusionMatrixDisplay
import numpy as np

Y_pred = model.predict_generator(test_ds, 500)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
cm = confusion_matrix(test_ds.classes, y_pred, normalize='true')
print('Classification Report')

```

```

target_names = ['cbb', 'cbds', 'cgm', 'cmd', 'healthy']
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=target_names)
disp.plot(cmap=plt.cm.Blues)
plt.show()

import tensorflow

from tensorflow.keras.preprocessing.image import load_img,
img_to_array
from tensorflow.keras.models import load_model

import tensorflow_hub as hub
from tensorflow import keras
from keras.models import model_from_json

loaded_model = load_model("Model.h5", custom_objects =
{"KerasLayer" : hub.KerasLayer})

scores = model.evaluate(test_ds)

from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score, KFold
from sklearn.linear_model import LogisticRegression
iris=load_iris()
x=iris.data
y=iris.target
logreg=LogisticRegression(max_iter=1000)
score=cross_val_score(logreg,x,y,cv=5)
print("Cross Validation Scores are {}".format(score))
print("Average Cross Validation score :{}".format(score.mean()))

history.params

history.history.keys()

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8,8))
plt.subplot(1, 2, 1)
plt.plot(range(EPOCHS), acc, label = 'Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label = 'Validation Accuracy')
plt.legend(loc = 'lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(range(EPOCHS), loss, label = 'Training Loss')
plt.plot(range(EPOCHS), val_loss, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.title('Training and Validation Loss')

```

