

**IDENTIFIKASI VARIETAS TANAMAN JATI
BERDASARKAN DAUN
MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK(CNN)**

TUGAS AKHIR

Tugas Akhir ini sebagai salah satu syarat untuk memperoleh gelar sarjana
Teknik Informatika Universitas Pembangunan Nasional "Veteran" Yogyakarta



Disusun Oleh :

Achmad Makarim Widyanto

NIM : 123160107

**PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK INDUSTRI
UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN" YOGYAKARTA
2021**

HALAMAN PENGESAHAN PEMBIMBING

**IDENTIFIKASI VARIETAS TANAMAN JATI BERDASARKAN DAUN
MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK(CNN)**



Mengetahui,
Koordinator Tugas Akhir

Dr. Awang Hendrianto Pratomo, S.T., M.T.
NIP. 1977 07 25 2005 01 1001

SURAT PERNYATAAN
KARYA ASLI TUGAS AKHIR

Sebagai mahasiswa Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Pembangunan Nasional “Veteran” Yogyakarta, yang bertanda tangan di bawah ini, saya :

Nama : Achmad Makarim Widyanto
NIM : 123160107

Menyatakan bahwa karya ilmiah saya yang berjudul :

Identifikasi Varietas Tanaman Jati Berdasarkan Daun Menggunakan Metode Convolutional Neural Network(CNN)

Merupakan karya asli saya dan belum pernah dipublikasikan dimanapun. Apabila di kemudian hari, karya saya disinyalir bukan merupakan karya asli saya, maka saya bersedia menerima konsekuensi apa pun yang diberikan Program Studi Teknik Informatika Fakultas Teknik Industri Universitas Pembangunan Nasional “Veteran” Yogyakarta kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Yogyakarta
Pada Tanggal : 7 Januari 2021

Yang Menyatakan

Achmad Makarim Widyanto
NIM. 123160107

PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini :

Nama : Achmad Makarim Widyanto
NIM : 123160107
Fakultas/Prodi : Teknik Industri/Teknik Informatika

Dengan ini saya menyatakan bahwa judul Tugas Akhir

Indentifikasi Varietas Tanaman Jati Berdasarkan Daun Menggunakan Metode Convolutional Neural Network(CNN)

Adalah hasil kerja keras saya sendiri dan benar bebas dari plagiasi kecuali cuplikan serta ringkasan yang terdapat di dalamnya telah saya jelaskan sumbernya (Situs) dengan jelas. Apabila pernyataan ini terbukti tidak benar maka saya akan bersedia menerima sanksi sesuai peraturan Mendiknas RI No 17 Tahun 2010 dan Peraturan Perundang-undangan yang berlaku.

Demikian surat pernyataan ini saya buat dengan penuh tanggung jawab.

Yogyakarta,
Yang membuat pernyataan



Achmad Makarim Widyanto
NIM. 123160107

ABSTRAK

Kayu jati merupakan salahsatu komoditas kayu mewah diantara kayu jenis hutan lainnya. Dikatakan mewah karena kualitas kayu jati yang bagus terhadap kondisi cuaca tropis seperti indonesia dan diminati oleh banyak masyarakat Indonesia, itu terlihat dalam *trend* minat masyarakat terhadap kayu jati berdasarkan data yang bersumber dari mesin pencarian google. Banyak masyarakat menggunakan kayu jati sebagai bahan baku pembuatan pembangunan rumah seperti pondasi, pintu, jendela, selain itu juga digunakan sebagai bahan pembuatan furniture seperti meja, almari, kursi, kasur dan masih banyak lainnya.

Karena minat masyarakat terhadap kayu jati sangat tinggi maka permintaan tanaman jati pun juga ikut tinggi. Persebaran tanaman jati di Indonesia sangatlah luas diantaranya terdapat pada Pulau Kalimantan, Pulau Jawa, Pulau Sulawesi, Pulau Sumbawa, Pulau Sumatera, dan Pulau Bali. Dari persebaran jati yang sangat luas itulah menyebabkan varietas jati di Indonesia menjadi beranekaragam macamnya, diantaranya ada varietas mega, PH1, plus, belanda, platinum, solomon, dan masih banyak lainnya.

Salah satu metode klasifikasi pengolahan berupa citra menggunakan *deep learning* yaitu *convolutional neural network*. Metode tersebut mampu mengatasi jumlah data yang lebih banyak dan lebih akurat jika dibandingkan metode klasifikasi *machine learning* lainnya. Klasifikasi yang dilakukan mencakup tiga varietas jati dan satu kelas tidak diketahui. Bagian tumbuhan atau morfologi tumbuhan yang cocok dalam penelitian ini menggunakan bagian daun karena jumlahnya yang banyak dan lebih mudah jika dibandingkan dengan lainnya.

Kata Kunci: Tanaman Jati, Daun, Klasifikasi, *Deep Learning*, *Convolutional Neural Network*

KATA PENGANTAR

Bismillahirrahmanirrahim, Puji syukur kepada Allah SWT, Tuhan Yang Maha Esa yang telah melimpahkan rahmat, iman, nikmat karunia, sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul "**Identifikasi Varietas Tanaman Jati Berdasarkan Daun Menggunakan Metode Convolutional Neural Network(CNN)**" serta dapat menyusun Laporan Tugas Akhir ini tepat pada waktunya.

Penyusunan Laporan Tugas Akhir ini merupakan salah satu tahap yang wajib dilalui oleh setiap mahasiswa Teknik Informatika dalam menyelesaikan jenjang Pendidikan S1 Program Studi Teknik Informatika Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta. Penulisan Laporan Tugas Akhir tidak lepas dari kekurangan, baik aspek kualitas maupun aspek kuantitas dari materi penelitian yang disajikan. Kekurangan yang ada didasarkan pada keterbatasan yang dimiliki penulis.

Laporan Tugas Akhir ini tidak dapat terselesaikan tanpa bantuan dari berbagai pihak. Penulis menyampaikan ungkapan terima kasih kepada:

1. Bapak Drs. Witono dan Ibu Yanik Kurniawati S.T. sebagai kedua orangtua yang selalu mendoakan dan memberikan dukungan.
2. Bapak Dr. Awang Hendrianto Pratomo, S.T., M.T. selaku Ketua Jurusan Teknik Informatika UPN "Veteran" Yogyakarta.
3. Bapak Dr. Heriyanto A.Md., S.Kom., M.Cs. selaku Ketua Program Studi Informatika upn "Veteran" Yogyakarta dan sebagai dosen wali.
4. Bapak Hidayatulah Himawan, S.T., M.M., M.Eng. selaku dosen pembimbing I dan Bapak Herry Sofyan, S.T., M.Kom. selaku dosen pembimbing II atas waku, pikiran, tenaganya yang sudah sabar dalam membimbing penelitian dan penulisan ini.
5. Ibu Dr. Herlina Jayadianti, S.T., M.T., dan Bapak Oliver Samuel Simanjuntak S.Kom., M.Eng. selaku dosen penguji yang telah berkenan memberikan arahan dalam penulisan tugas akhir.

6. Seluruh dosen dan staf tata usaha Program Studi Teknik Informatika yang telah membantu selama menempuh pendidikan kuliah ini.
7. Bapak Anggit selaku Kepala bagian pembibitan Perumahan Hutan Kabupaten Gunung Kidul dan Bapak Tukiya selaku koordinator Hutan Lindung Wanagama UGM yang telah berkenan membimbing untuk mengenalkan persemaian Tanaman Jati.
8. Pihak Kantor P.T. Solusi Pembayaran Elektronik (SPE) yang telah membantu secara finansial dalam proses penelitian tugas akhir ini.
9. Alumni Informatika 2010 dan Mba Sri Rahayu Astuti, S.T., M.Kom., yang telah membantu memberikan informasi dan membantu secara finansial dalam proses penelitian tugas akhir ini.
10. Konco-konco srawung, dolan nearleka yang telah seneng dan susah bertumbuh bersama semenjak awal kuliah hingga saat ini.
11. Mas Rosadi, mas kevin, dan mas Abika yang sudah berkenan meluangkan waktunya untuk berdiskusi.
12. Kakak-Kakak senior Data Science Indonesia yang sudah berkenan mendidik adik-adiknya kemudian, teman-teman *chapter* Yogyakarta yang telah meluangkan waktunya untuk belajar dan tumbuh bersama dalam bidang *data science*.
13. Rekan-rekan Informatika angkatan 2016 yang telah mendukung dan memberikan bantuan selama dinamika menimba ilmu.
14. Berbagai pihak yang telah mendukung kelancaran penulisan tugas akhir, terkhusus kepada Verdana Fotocopy dan Desya Fotocopy.

Penulis mengharapkan adanya saran, kritik, dan tanggapan yang bersifat membangun dalam upaya pembelajaran lebih lanjut mengingat kekurangan yang ada. Penulis juga berharap Laporan Tugas Akhir ini berguna dan bermanfaat bagi semua pihak. Semoga Allah SWT, Tuhan Yang Maha Esa selalu membimbing dalam menyelami dan mengamalkan ilmu-Nya untuk menuju kehidupan yang lebih baik.

Yogyakarta, 8 Januari 2021

Penulis

Daftar Isi

Halaman Judul.....	i
Halaman Pengesahan Pembimbing	ii
Surat Pernyataan Karya Asli Tugas Akhir	iii
Pernyataan Bebas Plagiat	iv
Abstrak	v
Kata Pengantar	vi
Daftar Isi.....	viii
Daftar Gambar	xi
Daftar Tabel	xv
Daftar Modul Program	xvii
BAB I PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah	7
1.3 Batasan Masalah.....	8
1.4 Tujuan Penelitian.....	8
1.5 Manfaat Penelitian.....	8
1.6 Metodologi Penelitian.....	9
1.6.1 Metodologi Pengumpulan Data.....	9
1.6.2 Metodologi Pengembangan Sistem	10
1.7 Sistematika Penulisan.....	11
BAB II TINJAUAN PUSTAKA	13
2.1 Identifikasi	13
2.2 Jati	13
2.2.1 Perkembangbiakan Jati	14
2.2.1.1 Generatif(benih)	14
2.2.1.2 Vegetatif(stek pucuk)	14
2.3 Varietas	15
2.3.1 Varietas Mega	15
2.3.2 Varietas Perhutani 1(PH1)	16
2.3.3 Varietas Plus	17
2.4 Moroflogi Daun	18
2.4.1 Dasar	18

2.4.2 Turunan	19
2.5 Pengenalan Pola	22
2.5.1 Pengenalan Pola Secara Statistik	23
2.5.2 Pengenalan Pola Secara Sintatik	25
2.6 Citra Digital	25
2.6.1 Grayscale(citra keabuan)	26
2.6.2 Biner	26
2.6.3 Berwarna	29
2.6.4 Warna Berindeks	29
2.7 Computer Vision	30
2.8 Jaringan Syaraf Tiruan	31
2.9 Artificial Intellegence	33
2.10 Machine Learning	34
2.11 Deep Learning	36
2.12 Preprocessing	36
2.12.1 Image Augmentated	37
2.12.2 Batch	37
2.12.3 Image Size	37
2.12.4 Split Data	37
2.13 Convolutional Neural Network	37
2.13.1 Convolution Layer(konvolusi)	38
2.13.2 Pooling Layer	42
2.13.3 Fully Connected Layer	43
2.13.4 Dropout.....	50
2.14 Confusion Matrix	51
2.15 Metode Pengembangan Sistem GRAPPLE	52
BAB III METODOLOGI PENELITIAN DAN PENGEMBANGAN SISTEM	54
3.1 Metodologi Penelitian	54
3.1.1 Pengumpulan Data	56
3.1.2 Studi Pustaka	57
3.2 Metodologi Pengembangan Sistem	58
3.2.1 Kebutuhan Sistem	59
3.2.2 Analisis	60

3.2.3 Perancangan	88
BAB IV HASIL, PENGUJIAN, DAN PEMBAHASAN	103
4.1 Hasil Penelitian	103
4.1.1 Implementasi Preprocessing	103
4.1.2 Implementasi Perancangan Arsitektur	107
4.1.3 Implementasi Perancangan Pengurangan Bias	142
4.1.4 Implementasi Proses Pelatihan	142
4.1.5 Evaluasi Hasil Pelatihan	145
4.1.6 Penyimpanan Model	147
4.1.7 Implementasi Pengujian Model	147
4.2 Pengujian Sistem	157
4.2.1 Pengujian Black Box	157
4.2.2 Confusion Matrix	157
BAB V PENUTUP	179
5.1 Kesimpulan	179
5.2 Saran	179
DAFTAR PUSTAKA	181

Daftar Gambar

Gambar 1.1 Ilustrasi Persebaran Tanaman Jati di Indonesia	1
Gambar 1.2 <i>Trend</i> Pencarian <i>Keyword</i> Tanaman Jati	2
Gambar 1.3 Perbandingan <i>Trend</i> Pencarian Jenis Tanaman Hutan Lainnya	2
Gambar 1.4 Persebaran Daerah Pencarian <i>Keyword</i> Tanaman Jati	3
Gambar 1.5 <i>Trend</i> Pencarian <i>Keyword</i> Kayu Jati	4
Gambar 1.6 Perbandingan <i>Trend</i> Pencarian Jenis Kayu Hutan Lainnya	4
Gambar 1.7 Persebaran Daerah Pencarian <i>Keyword</i> Kayu Jati	5
Gambar 1.8 Alur Tahap Penelitian	9
Gambar 1.9 Alur Tahap Pengujian	10
Gambar 2.1 Daun Jati Mega	15
Gambar 2.2 Daun Jati Perhutani 1(PH1)	16
Gambar 2.3 Daun Jati Plus	17
Gambar 2.4 Area Pada daun	18
Gambar 2.5 Perimeter Pada Daun	19
Gambar 2.6 Diameter Pada Daun	19
Gambar 2.7 Physiological Length dan Physiological Width	19
Gambar 2.8 Sistem Pengenalan Pola Dengan Pendekatan Statistic	23
Gambar 2.9 Sistem Pengenalan Pola Dengan Pendekatan Sintaktik.....	25
Gambar 2.10 Representasi Citra Digital Dalam Bentuk 2 Dimensi	27
Gambar 2.11 Representasi Bentuk Matriks Citra Digital.....	27
Gambar 2.12 Hitam dan Putih (Grayscale)	28
Gambar 2.13 Binary	28
Gambar 2.14 Red Green Blue	29
Gambar 2.15 Warna Berindeks	30
Gambar 2.16 Komparasi Sudut Pandang.....	32
Gambar 2.17 Ilustrasi Proses Machine Learning.....	35
Gambar 2.18 Ilustrasi Alur Deep Learning	36
Gambar 2.19 Arsitektur Sederhana MLP	38
Gambar 2.20 Visualisasi Arsitektur CNN Secara Umum	38
Gambar 2.21 Ilustrasi Channel Red Green Blue	39
Gambar 2.22 Ilustrasi Alur Konvolusi	39
Gambar 2.23 Perhitungan Jumlah Konvolusi.....	40

Gambar 2.24 Diagram Garis Pembentukan RELU	40
Gambar 2.25 Ilustrasi Stride.....	41
Gambar 2.22 Ilustrasi Padding	42
Gambar 2.27 Ilustrasi Pengambilan Nilai.....	43
Gambar 2.28 Ilustrasi Proses pada Fully Connected Layer.....	44
Gambar 2.29 Grafik Skenario Softmax	45
Gambar 2.30 Komparasi Nilai Learning Rate (Jason Brownlee, 2019).....	47
Gambar 2.31 Ilustrasi Penghilangan Node(dropout).....	50
Gambar 2.32 Tahapan Metode Pengembangan Sistem GRAPPLE	53
Gambar 3.1 Kerangka Kerja Penelitian	55
Gambar 3.2 Lanjutan Kerangka Kerja Penelitian	56
Gambar 3.3 Alur Pengembangan Sistem GRAPPLE	57
Gambar 3.4 Tahapan Percobaan Membuat Model CNN	58
Gambar 3.5 Flowchart Dalam Preprocessing	61
Gambar 3.6 Ilustrasi merubah ukuran(resize)	62
Gambar 3.7 Ilustrasi Perbanyakkan Dataset	62
Gambar 3.8 Flowchart Pembelajaran Model	63
Gambar 3.9 Ilustrasi Arsitektur CNN	64
Gambar 3.10 Ilustrasi Perkalian Pada Lapisan Konvolusi	64
Gambar 3.11 Ilustrasi Fungsi Aktivasi RELU	65
Gambar 3.12 Ilustrasi Pada Lapisan Pooling	66
Gambar 3.13 Ilustrasi Penyederhanaan Dimensi Array	67
Gambar 3.14 Ilustrasi Tahapan Dari Input Hingga Flatten	67
Gambar 3.15 Ilustrasi Fully Connected Layer	68
Gambar 3.16 Contoh Perhitungan Fungsi Aktivasi Softmax	69
Gambar 3.17 Flowchart Pengujian Model	70
Gambar 3.18 Flowchart Arsitektur CNN 1	71
Gambar 3.19 Flowchart Arsitektur CNN 2	72
Gambar 3.20 Flowchart Arsitektur CNN 3	73
Gambar 3.21 Flowchart Arsitektur CNN 4	75
Gambar 3.22 Flowchart Arsitektur CNN 5	77
Gambar 3.23 Flowchart Arsitektur CNN 6	79
Gambar 3.24 Flowchart Arsitektur CNN 7	81

Gambar 3.25 Flowchart Arsitektur CNN 8	83
Gambar 3.26 Ilustrasi 4 Batch dalam Training	85
Gambar 3.27 Ilustrasi 4 Batch dalam Validasi	85
Gambar 3.28 Ilustrasi Menggunakan Dropout	86
Gambar 3.29 Flowchart Contoh Arsitektur CNN Menggunakan Dropout	86
Gambar 3.30 Kurva Penggunaan Learning Rate yang Berbeda lr=2(kiri) lr=0.2(kanan) ..	87
Gambar 3.31 Perancangan Arsitektur Sistem	88
Gambar 3.32 Use Case Diagram	90
Gambar 3.33 Aktivitas Diagram Unduh Sample Data Uji	91
Gambar 3.34 Aktivitas Diagram Potret Kamera	92
Gambar 3.35 Aktivitas Diagram Upload File	93
Gambar 3.36 Aktivitas Diagram Identifikasi	94
Gambar 3.37 Sequance Diagram Unduh Sample Data Uji	95
Gambar 3.38 Sequance Diagram Potret Kamera	96
Gambar 3.39 Sequance Diagram Upload File	97
Gambar 3.40 Sequance Diagram Identifikasi	97
Gambar 3.41 Class Diagram Identifikasi	98
Gambar 3.42 Class Diagram CNN	99
Gambar 3.43 Tampilan Beranda	100
Gambar 3.42 Tampilan Hasil Kelas	101
Gambar 4.1 Contoh Data Latih	106
Gambar 4.2 Contoh Augmentasi Gambar	106
Gambar 4.3 Contoh Data Validasi	107
Gambar 4.4 Summary Arsitektur Pertama	108
Gambar 4.5 Summary Arsitektur Kedua	110
Gambar 4.6 Summary Arsitektur Ketiga	114
Gambar 4.7 Summary Arsitektur Keempat	118
Gambar 4.8 Summary Arsitektur Kelima	123
Gambar 4.9 Summary Arsitektur Keenam	128
Gambar 4.10 Summary Arsitektur Ketujuh	134
Gambar 4.11 Summary Arsitektur Kedelapan	139
Gambar 4.12 Kurva Pembelajaran Model	146
Gambar 4.13 Hasil Evaluasi	147

Gambar 4.14 Halaman Antar Muka	148
Gambar 4.15 Halaman Contoh Data Sample	149
Gambar 4.16 Popup Peletakan Data Uji yang Ingin Diunduh	150
Gambar 4.17 Interface Halaman Utama	152
Gambar 4.18 Interface Hasil Klasifikasi Citra	153
Gambar 4.19 Interface Lanjutan Klasifikasi Citra	154
Gambar 4.20 Info Deploy Heroku	156
Gambar 4.21 Hasil Pengecekan Data Kosong	176
Gambar 4.22 Hasil Akhir Confusion Matrix	178

Daftar Tabel

Tabel 2.1 Contoh Pola dan Ciri	22
Tabel 2.2 Perbedaan Human Vision dengan Computer Vision.....	31
Tabel 2.3 Skema Perhitungan confusion matrix	51
Tabel 3.1 Rincian Pembagian Jumlah Data	57
Tabel 3.2 Contoh Citra Daun Jati	57
Tabel 3.3 Kebutuhan Perangkat Keras	59
Tabel 3.4 Kebutuhan Perangkat Lunak	59
Tabel 3.5 Lanjutan Kebutuhan Perangkat Lunak	60
Tabel 3.6 Arsitektur CNN yang Diuji	70
Tabel 3.7 Lanjutan Arsitektur CNN yang Diuji	71
Tabel 3.8 Pengujian Black Box	101
Tabel 3.9 Lanjutan Pengujian Black Box	102
Tabel 3.10 Daftar Kelas Klasifikasi	102
Tabel 3.11 Confusion Matrix Multi Class	102
Tabel 3.12 Hasil Confusion Matrix Multi Class	102
Tabel 4.1 Rincian Proses Pembelajaran Data	143
Tabel 4.2 Lanjutan Rincian Proses Pembelajaran Data	144
Tabel 4.3 Lanjutan Rincian Proses Pembelajaran Data	145
Tabel 4.4 Daftar Nama Penguji	157
Tabel 4.5 Sistematika Pengujian	157
Tabel 4.6 Hasil Klasifikasi Data Uji	160
Tabel 4.7 Lanjutan Hasil Klasifikasi Data Uji	161
Tabel 4.8 Lanjutan Hasil Klasifikasi Data Uji	162
Tabel 4.9 Lanjutan Hasil Klasifikasi Data Uji	163
Tabel 4.10 Lanjutan Hasil Klasifikasi Data Uji	164
Tabel 4.11 Lanjutan Hasil Klasifikasi Data Uji	165
Tabel 4.12 Lanjutan Hasil Klasifikasi Data Uji	166
Tabel 4.13 Lanjutan Hasil Klasifikasi Data Uji	167
Tabel 4.14 Lanjutan Hasil Klasifikasi Data Uji	168
Tabel 4.15 Lanjutan Hasil Klasifikasi Data Uji	169
Tabel 4.16 Lanjutan Hasil Klasifikasi Data Uji	170
Tabel 4.17 Lanjutan Hasil Klasifikasi Data Uji	171

Tabel 4.18 Lanjutan Hasil Klasifikasi Data Uji	172
Tabel 4.19 Lanjutan Hasil Klasifikasi Data Uji	173
Tabel 4.20 Lanjutan Hasil Klasifikasi Data Uji	174
Tabel 4.21 Lanjutan Hasil Klasifikasi Data Uji	175
Tabel 4.22 Hasil Pengujian Confusion Matrix	176

Daftar Modul Program

Modul Program 4.1 Inisiasi Library	103
Modul Program 4.2 Membaca Lokasi Dataset	104
Modul Program 4.3 Perbanyakkan Dataset dan Pembagian Validasi	104
Modul Program 4.4 Menampilkan Sample Data Latih	105
Modul Program 4.5 Prosedur Menampilkan Gambar Dataset	105
Modul Program 4.6 Pemanggilan Prosedur Visualisasi Data	105
Modul Program 4.7 Augmentasi Gambar	106
Modul Program 4.8 Menampilkan Sample Data Validasi	106
Modul Program 4.9 Pemanggilan Prosedur Visualisasi Data	106
Modul Program 4.10 Arsitektur Pertama	107
Modul Program 4.11 Arsitektur Kedua	109
Modul Program 4.12 Arsitektur Ketiga	111
Modul Program 4.13 Lanjutan Arsitektur Ketiga	112
Modul Program 4.14 Arsitektur Keempat	115
Modul Program 4.15 Arsitektur Kelima	119
Modul Program 4.16 Lanjut Arsitektur Kelima	120
Modul Program 4.17 Arsitektur Keenam	125
Modul Program 4.18 Arsitektur Ketujuh	130
Modul Program 4.19 Lanjutan Arsitektur Ketujuh	131
Modul Program 4.20 Arsitektur Kedelapan	136
Modul Program 4.21 Untuk Keluaran Summary Arsitektur	141
Modul Program 4.22 Pengurangan Error	142
Modul Program 4.23 Pelatihan Data	143
Modul Program 4.24 Evaluasi Pembelajaran Data Latih dengan Data Validasi	145
Modul Program 4.25 Lanjutan Evaluasi Pembelajaran Data Latih dengan Data Validasi	146
Modul Program 4.26 Nilai Akurasi dan Loss Model	147
Modul Program 4.27 Penyimpanan Model	147
Modul Program 4.28 Inisiasi Library Pengujian Model	147
Modul Program 4.29 Halaman Antar Muka	148
Modul Program 4.30 Mengarahkan Lokasi Model	150
Modul Program 4.31 Tahap Preprocessing Pengujian	150

Modul Program 4.32 Meratakan Matriks Data Uji	151
Modul Program 4.33 Proses Klasifikasi Citra	151
Modul Program 4.34 Lanjutan Proses Klasifikasi Citra	152
Modul Program 4.35 Style Tampilan	154
Modul Program 4.36 Memanggil Style Tampilan	155
Modul Program 4.37 Inisiasi Lokasi Program	155
Modul Program 4.38 Inisiasi Library yang Digunakan	155
Modul Program 4.39 Inisiasi Pengaturan Hosting Server	156
Modul Program 4.40 Inisiasi Library Pengujian	158
Modul Program 4.41 Persiapan Sebelum Pengujian	158
Modul Program 4.42 Memanggil Model	159
Modul Program 4.43 Deklarasi Varibel	159
Modul Program 4.44 Inisiasi Data Uji	160
Modul Program 4.45 Proses Klasifikasi Data	160
Modul Program 4.46 Cek Data Kosong	176
Modul Program 4.47 Proses Confusion Matrix	176
Modul Program 4.48 Visualisasi Hasil Confusion Matrix	177
Modul Program 4.49 Visualisasi Akhir	177

BAB I

PENDAHULUAN

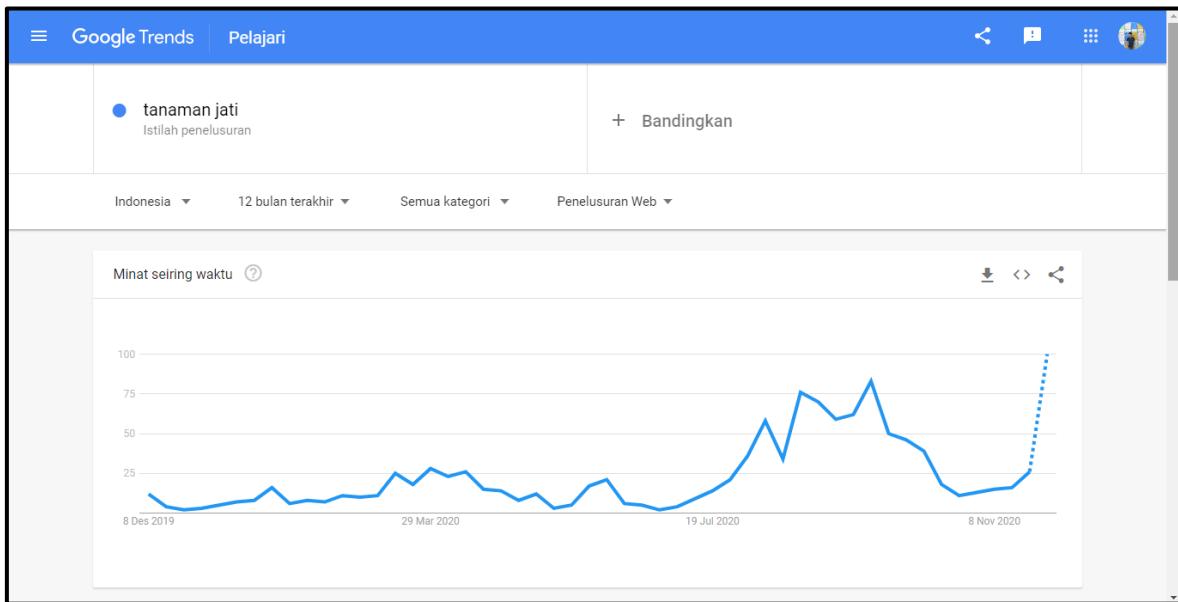
1.1 Latar Belakang

Tanaman Jati yang mempunyai nama latin *Tectona grandis Linn. F.*, merupakan salah satu tanaman hutan yang cocok untuk ditanam di iklim tropis contohnya seperti di Indonesia. Tanaman jati memiliki dua cara perkembangbiakan, yaitu yang pertama secara vegetatif, dan yang kedua adalah secara generatif. Perkembangbiakan vegetatif adalah perkembangbiakan yang berasal dari stek pucuk daun muda yang dihasilkan oleh jati yang sudah berusia dewasa, yang nantinya akan ditanam pada media tanah yang baru. Kemudian untuk perkembangbiakan generatif adalah perkembangbiakan yang melalui proses penyerbukan antara jati jantan dengan jati betina.



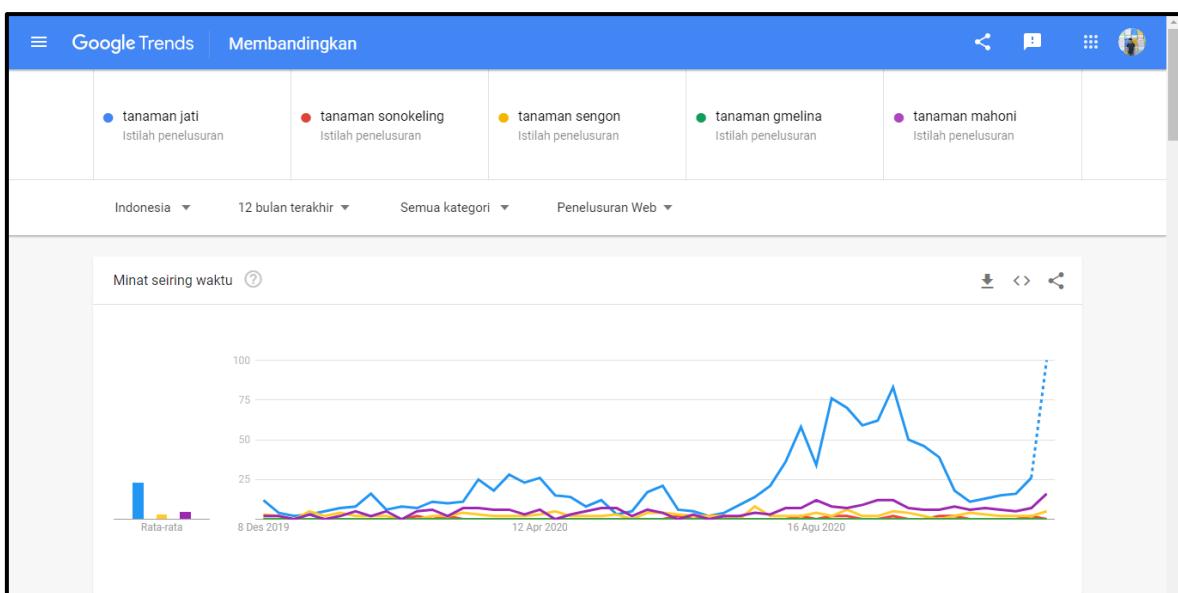
Gambar 1.1 Ilustrasi Persebaran Tanaman Jati di Indonesia

Persebaran jati di Indonesia sendiri sangatlah luas, beberapa diantaranya berada di Pulau Jawa, untuk di luar Pulau Jawa, jati ditemukan secara terbatas di beberapa tempat di Pulau Sulawesi, Pulau Muna, Pulau Sumbawa, Pulau Bali, Pulau Sumatra dan Pulau Kalimantan (Murtinah et al., 2015). Karena persebaran jati yang sangat luas itulah menjadikan varietas jati di Indonesia ini menjadi beranekaragam, diantaranya ada jati mega, jati kumbokarno, jati plus, jati perhutani 1, jati belanda, jati cina, jati emas, jati putih, jati solomon, dan masih banyak lainnya.



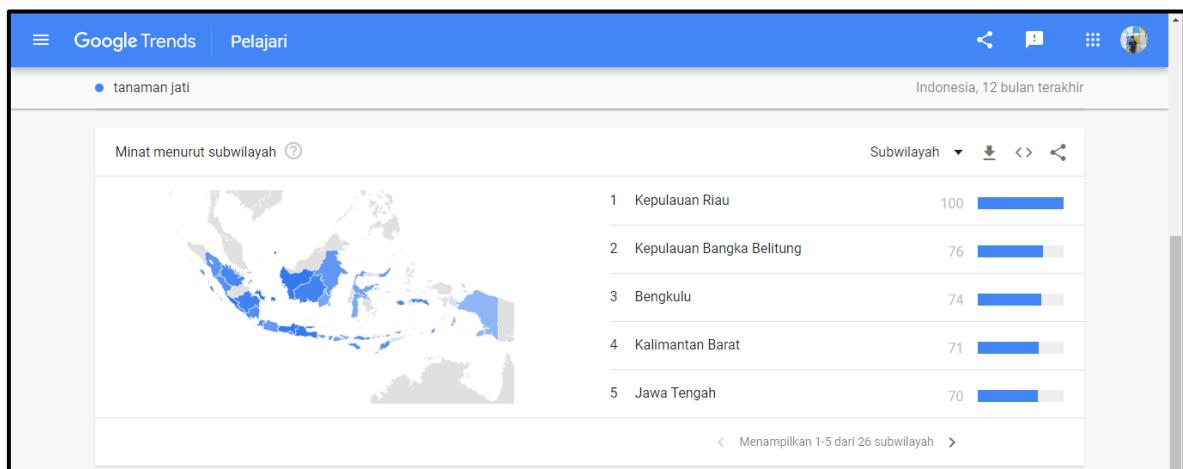
Gambar 1.2 Trend Pencarian Keyword Tanaman Jati
Sumber: Google Trends(2020)

Trend mengenai tanaman jati data yang bersumber dari *google trends* berdasarkan data satu tahun terakhir dimana dimulai dari bulan Desember tahun 2019 hingga bulan November tahun 2020 kemaren, menunjukkan adanya *trend* kenaikan pencarian mengenai keyword tanaman jati, meskipun beberapa kali mengalami penurunan tetapi kebanyakan mengalami kenaikan artinya secara keseluruhan mengalami kenaikan.



Gambar 1.3 Perbandingan Trend Pencarian Jenis Tanaman Hutan Lainnya
Sumber : Google Trends(2020)

Gambar pada grafik jika dibandingkan dengan tanaman hutan lainnya, warna biru untuk tanaman jati, warna merah untuk tanaman sonokeling, warna kuning untuk tanaman sengon, warna hijau untuk tanaman gmelina, dan warna ungu untuk tanaman mahoni, menunjukkan bahwa dalam satu tahun terakhir tanaman jati berada di posisi ter atas yang artinya paling banyak dicari oleh masyarakat jika dibandingkan dengan tanaman hutan jenis lainnya. Untuk jenis tanan hutan lainnya dalam satu tahun terakhir menunjukkan data datar dimana kurang lebih selalu sama banyaknya jumlah pencarian, kemudian untuk tanaman jati meskipun beberapa kali mengalami penurunan namun secara keseluruhan mengalami peningkatan pencarian.

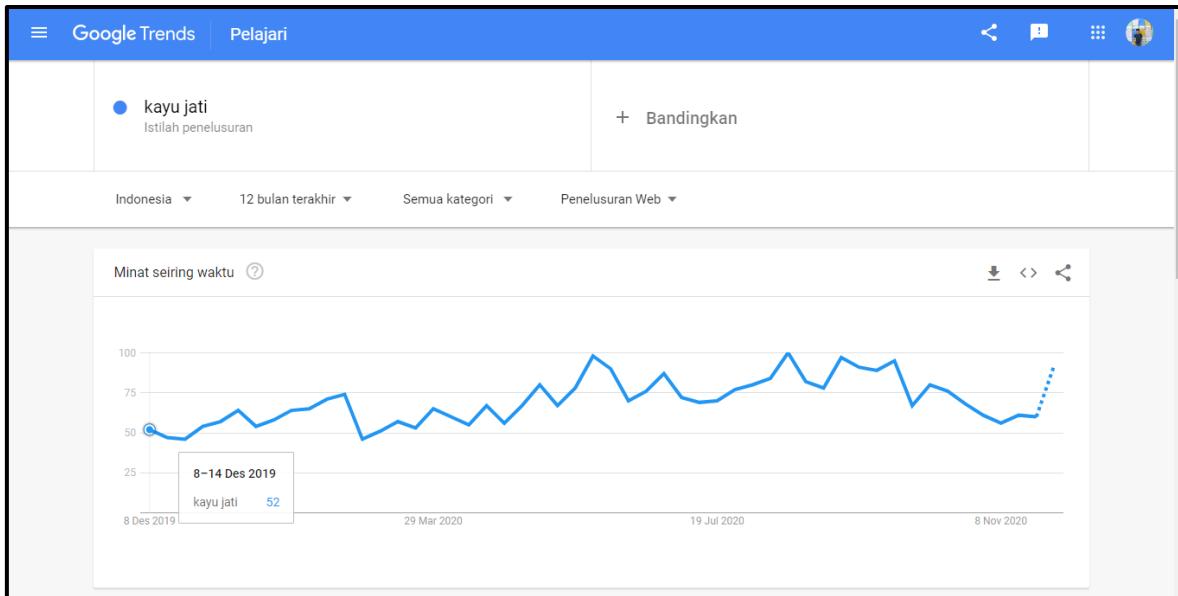


Gambar 1.4 Persebaran Daerah Pencarian Keyword Tanaman Jati
Sumber : Google Trends(2020)

Sumber data yang berasal dari Google Indonesia menjadikan kemudahan dalam pemantauan di seluruh daerah Indonesia. Gambar di atas menunjukkan daerah-daerah yang paling banyak mencari tentang *keyword* tanaman jati, yang paling banyak mencari berasal dari Kepulauan Riau, kemudian di urutan kedua berasal dari Kepulauan Bangka Belitung, di urutan ketiga berasal dari Bengkulu, urutan keempat terbanyak berasal dari Kalimantan Barat, dan posisi kelima derah yang paling banyak mencari berasal dari Jawa Tengah.

Jika sebelumnya untuk mengetahui *trend* pencarian *keyword* tanaman jati, maka kali ini akan diketahui *trend* pencarian *keyword* mengenai kayu jatinya. Sumber data masih sama

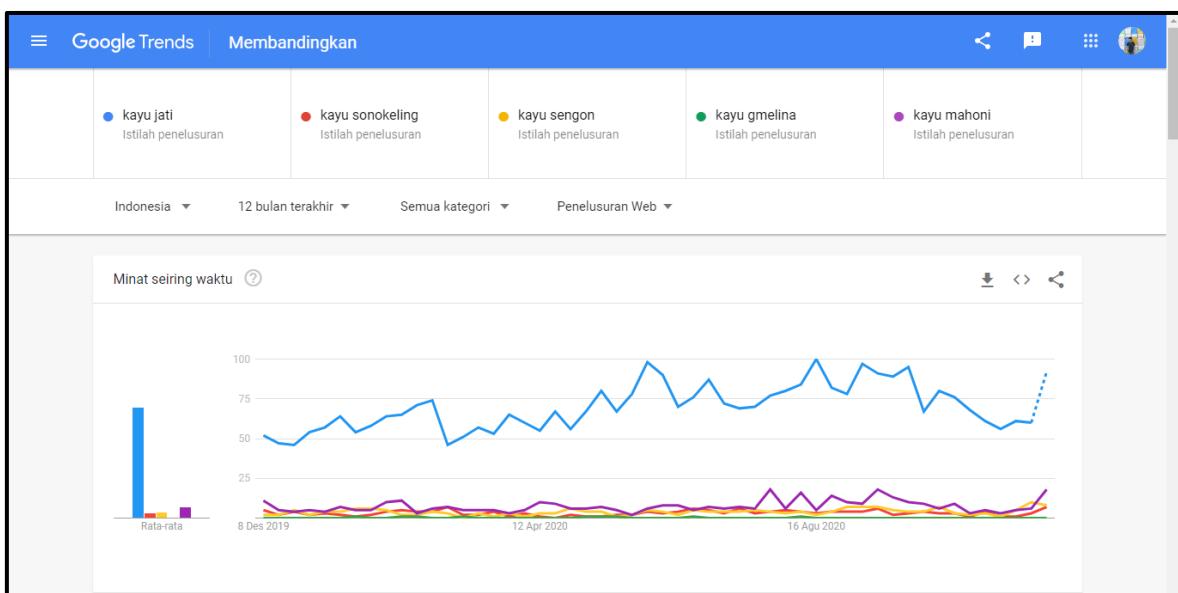
berasal dari Google Trends diambil dalam satu tahun terakhir, berawal dari bulan Desember tahun 2019 hingga bulan November tahun 2020.



Gambar 1.5 Trend Pencarian Keyword Kayu Jati

Sumber : Google Trends(2020)

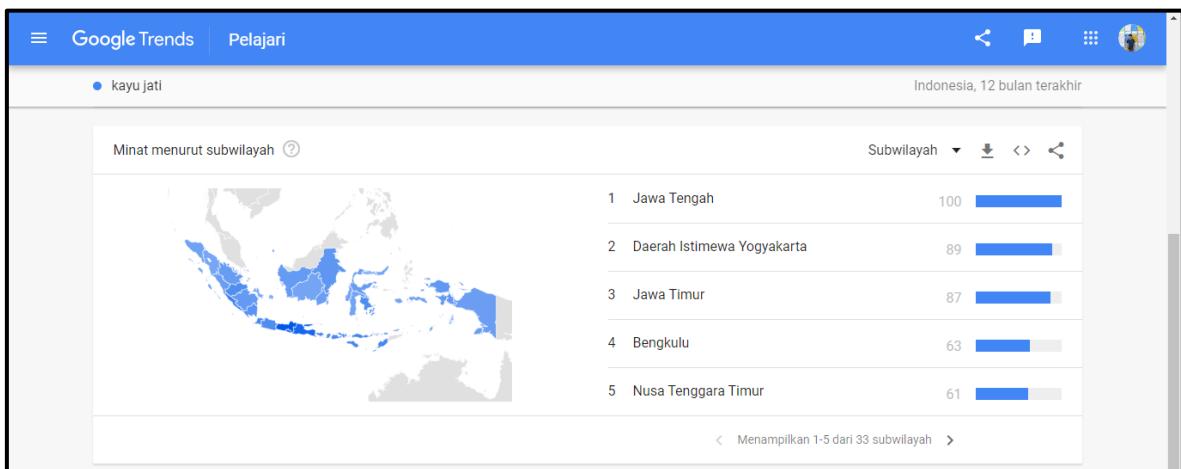
Pada grafik satu tahun terakhir menunjukkan adanya *trend* pencarian *keyword* mengenai kayu jati, meskipun dalam beberapa kali mengalami penurunan pencarian, namun secara keseluruhan cenderung mengalami peningkatan pencarian diantara rentang nilai 50 hingga nilai 100.



Gambar 1.6 Perbandingan Trend Pencarian Jenis Kayu Hutan Lainnya

Sumber : Google Trends(2020)

Kemudian, untuk perbandingan dengan jenis kayu hutan lainnya, warna biru untuk kayu jenis jati, warna merah untuk jenis kayu sonokeling, warna kuning untuk kayu sengon, warna hijau untuk kayu gmelina, dan warna ungu untuk kayu mahoni, dalam satu tahun terakhir kayu jenis jati ini cenderung lebih banyak apabila dibandingkan dengan jenis kayu hutan lainnya yang mana untuk jenis kayu hutan lainnya cenderung datar dan di bawah dari jenis kayu jati.



Gambar 1.7 Persebaran Daerah Pencarian Keyword Kayu Jati
Sumber : Google Trends(2020)

Pencarian mengenai *keyword* kayu jati dalam satu tahun terakhir paling banyak dicari berasal dari daerah Jawa tengah, kemudian paling banyak kedua dari Daerah Istimewa Yogyakarta, pencari ketiga terbanyak berasal dari Jawa Timur, pencari terbanyak keempat dari Bengkulu, dan yang kelima terbanyak berasal dari daerah Nusa Tenggara Timur.

Berdasarkan grafik-grafik tersebut yang bersumber dari data google trends, menunjukkan masih banyak masyarakat minat dan tertarik terhadap kayu jati dan tanaman jati, bahkan dalam satu tahun terakhir menunjukkan adanya kenaikan pencarian dalam dua *keywords* tersebut. Kemudian apabila dibandingkan dengan jenis kayu hutan lain dan jenis tanaman hutan lainnya, kayu dan tanaman berjenis jati ini paling banyak dicari dalam mesin pencarian google tersebut.

Kualitas kayu jati yang bagus sehingga lebih tahan dan dapat berumur panjang, hal tersebut membuat banyak masyarakat cenderung lebih tertarik dengan kayu jati sebagai bahan baku kebutuhan. Karena minat masyarakat terhadap kayu jati sangat banyak sehingga permintaan di pasar sangatlah tinggi sehingga budidaya tanaman jati sangat cocok untuk dijadikan lahan bisnis yang mana biasanya dijadikan sebagai investasi jangka panjang. Kenapa investasi jangka panjang, karena mulai awal pembibitan hingga panen kayunya memerlukan waktu yang lama.

Morfologi tumbuhan atau bagian tumbuhan diantaranya ada buah, daun, batang, akar, dan bunga, sehingga bagian-bagian tumbuhan biasanya yang dijadikan sebagai objek penelitian. Bagian batang, semakin bertambahnya umur pohon, warna dan kedalaman alurnya akan mengalami perubahan. Apabila menggunakan buah dan bunga akan sulit dilakukan karena tumbuh secara musiman, terlebih lagi tanaman jati sendiri tidak memiliki buah. Sedangkan akar akan sulit ketika pengambilan data, karena tempat tumbuhnya yang di dalam tanah sehingga harus terlebih dahulu mengeruk tanaman dari dalam tanah. Dan jika menggunakan daun, daun cenderung tersedia sebagai sumber pengamatan sepanjang waktu dan cenderung lebih mudah untuk menjadi objek pengamatan terutama berupa citra.

Persebaran jati di Indonesia yang luas menyebabkan varietas jati yang ada menjadi beranekaragam macamnya, karena keunikan dan beragamnya varietas jati yang ada menjadikan tanaman jati pernah beberapa kali diteliti, diantaranya penelitian dengan judul Identifikasi Tanaman Jati Menggunakan Probabilistic Neural Network Dengan Ekstraksi Fitur Ciri Morfologi Daun dari Institut Pertanian Bogor menghasilkan akurasi 77.5% oleh Bangun Asanurjaya. Kemudian penelitian kedua dengan judul Identifikasi Daun Tanaman Jati Menggunakan Jaringan Saraf Tiruan Backpropagation Dengan Ekstraksi Fitur Ciri Morfologi Daun dari Institut Pertanian Bogor menghasilkan akurasi 84.17%, dan penelitian yang ketiga dengan judul Identifikasi Daun Tanaman Jati Menggunakan K-Nearest neighbor

Dengan Ekstraksi Fitur Ciri Morfologi Daun dari Institut Pertanian Bogor menghasilkan akurasi 73.33% oleh Muhammad Bangkit Pratama.

Penelitian lain masih dalam seputar tanaman, namun kali ini menggunakan metode *convolutional neural network* penelitian pertama dengan judul Deteksi Hama Pada Daun Teh Dengan Metode Convolutional Neural Network (Cnn) dari Universitas Komputer Indonesia menghasilkan akurasi 95% oleh Bagja Hidayat dan Galih Hermawan. Penelitian kedua dengan judul Implementasi Convolutional Neural Networks Untuk Klasifikasi Citra Tomat Menggunakan Keras Universitas Islam Indonesia menghasilkan akurasi 90% oleh Tiara Shafira. Penelitian ketiga dengan judul Dan penelitian terakhir dengan judul Klasifikasi Citra Buah Menggunakan Convolutional Neural Network dari Universitas Negeri Surabaya menghasilkan akurasi 97.97%. Dan penelitian terakhir dengan judul Implementasi Deep Learning Menggunakan Metode Convolutional Neural Network Untuk Klasifikasi Gambar (Studi Kasus: Klasifikasi Gambar Pada Tanaman Anggrek Bulan Putih, Anggrek Dendrobium, dan Anggrek Ekor Tupai) dari Universitas Islam Indonesia menghasilkan akurasi 89% oleh Rahayu Kia Sandi Cahaya Putri.

Metode pengolahan data *convolutional neural network* termasuk *supervised learning* termasuk dalam teknik *Deep Learning* sehingga dapat mengelompokkan kelas-kelas yang sudah ditentukan dengan cara bisa tanpa memilih terlebih dahulu fitur-fitur yang akan dipelajari pola citranya, dan dapat juga mengatasi jumlah data yang lebih besar dari metode pengolahan data *backpropagation*.

1.2 Perumusan Masalah

Banyaknya varietas tanaman jati yang ada menjadikan faktor sulitnya dalam mengenali dengan pasti varietas tanaman jati yang satu dengan varietas tanaman jati yang lainnya, sehingga memerlukan keahlian dan ilmu yang mendalam untuk dapat membedakannya.

Pemilihan kayu yang tidak tepat untuk kegunaan akhir dapat diakibatkan dari kesalahan dalam mengenali atau mengidentifikasi varietas jati sejak awal, maka rumusan masalah yang dapat diperoleh sebagai berikut:

Bagaimana cara mengenali atau membaca varietas jati berdasarkan daun jati menggunakan metode *convolutional neural network* secara tepat dan lebih mudah?

1.3 Batasan Masalah

- a. Data latih dan data uji yang digunakan merupakan data primer.
- b. Terdiri atas 3 varietas tanaman jati yang digunakan yaitu varietas jati Mega, jati Perhutani 1(PH1), dan Jati Plus.
- c. Daun tanaman jati yang digunakan berasal dari daun persemaian tanaman jati yang berusia 2 - 3 bulan.
- d. Penelitian ini hanya membahas metode *Convolutional Neural Network*.
- e. Citra yang dimasukkan untuk proses *training* data adalah citra berwarna.
- f. *Output* dari penelitian ini hanya dapat membedakan varietas jati berdasarkan daun.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini sebagai berikut:

1. Memberikan cara untuk mempermudah mengenali atau mengetahui varietas tanaman jati.
2. Mengetahui cara pengklasifikasian dan ketepatan kinerja jaringan syaraf tiruan dengan menggunakan metode *Convolutional Neural Network* untuk dapat mengenali varietas tanaman jati.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini sebagai berikut:

1. Membantu masyarakat awam atau pihak - pihak terkait untuk dapat mengenali atau mengetahui varietas tanaman jati dengan lebih tepat dan mudah.

2. Penelitian yang sudah dilakukan ini dapat dijadikan refrensi penerapan penggunaan jaringan syaraf tirungan dengan metode *convolutional neural network* untuk penelitian jaringan syaraf tiruan yang akan datang.

1.6 Metodologi Penelitian

Metode pengumpulan data yang digunakan dalam penelitian ini dapat diuraikan sebagai berikut:

1.6.1 Metode Pengumpulan Data

1. Studi Pustaka

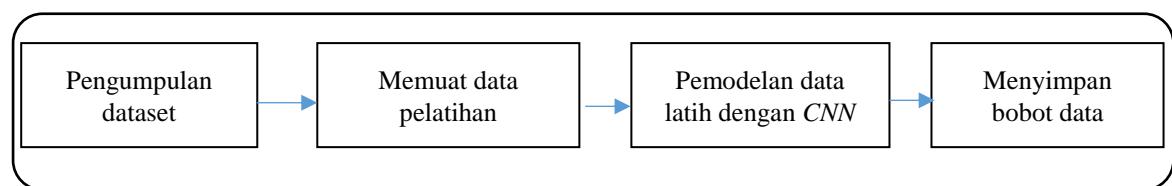
Pada tahap ini dilakukan studi kepustakaan untuk memperkuat literature penelitian yang sudah dilakukan dengan membaca buku, refrensi jurnal - jurnal, dan karya tulis ilmiah yang berhubungan dengan penelitian atau terkait dengan penelitian yang sedang dilakukan ini.

2. Pengumpulan Data

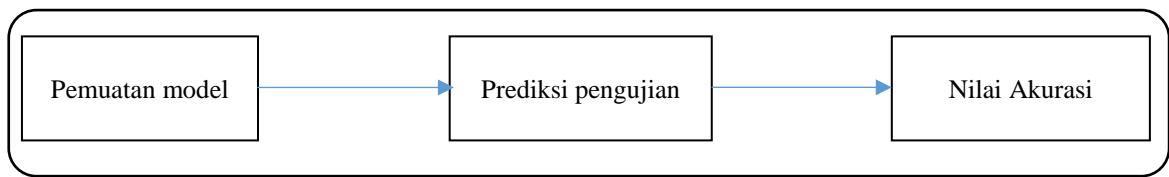
Mengumpulkan data yang digunakan dalam penelitian ini berguna untuk mendukung dan memperkuat kebutuhan selama penelitian. Data yang digunakan dalam penelitian ini merupakan data primer berupa citra daun tanaman jati. Data yang digunakan dalam penelitian ini berjumlah 1350 gambar dengan format *.jpg*.

3. Teknik Analisa Data

Dalam melakukan proses analisa pada penelitian ini terdapat dua tahapan, untuk tahapan pertama merupakan pelatihan data dengan model selanjutnya tahapan kedua pengujian data baru dari pelatihan yang telah dilakukan. Tahapan tersebut diilustrasikan sebagai berikut.



Gambar 1.8 Alur Tahap Pelatihan



Gambar 1.9 Alur Tahap Pengujian

1.6.2 Metode Pengembangan Sistem

Metode *GRAPPLE* merupakan metode pengembangan sistem yang setiap tahapannya terdiri dari beberapa tindakan, dan setiap tindakannya menghasilkan produk berupa UML (Schmuller, 1999). Adapun metodologi pengembangan sistem *GRAPPLE* terdiri dari segmen-segmen seperti berikut (Schmuller, 1999) :

1. *Requirement Gathering*, pada bagian ini bertujuan untuk merancang kebutuhan-kebutuhan apa saja yang akan digunakan dalam membangun sebuah sistem yang tepat sesuai kebutuhan perangkat lunak maupun perangkat keras.
2. *Analysis*, tahap untuk penguraian masalah-masalah yang muncul dari perumusan identifikasi kebutuhan sistem dan menganalisa pemetaan pihak aktor-aktor yang nantinya akan terlibat.
3. *Design*, setalah menjabarkan masalah pada bagian analisis lalu pada bagian desain ini membuat solusi-solusi dari penjabaran permasalahan yang ada dengan membuat perancangan sebuah *prototype user interface* untuk menilai apakah solusi yang diberikan sudah sesuai spesifikasi atau belum.
4. *Development*, tahap dimana proses pembuatan program mulai dilakukan berupa pengkodean berdasarkan diagram yang telah dirancang pada tahap sebelumnya berupa pembuatan *client side* dan juga *server side*.
5. *Deployment*, hasil dari pengkodean yang dibuat pada bagian *development* lalu diterapkan pada perangkat-perangkat lunak maupun keras untuk dilakukannya tahap

pengujian apakah perancangan hingga pengkodean sudah sesuai keinginan dan berjalan sesuai fungsinya.

1.7 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam menyusun laporan penelitian ini adalah sebagai berikut :

BAB I PENDAHULUAN

Pada bagian ini, pendahuluan membahas tentang latar belakang pemilihan topik, masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bagian ini, tinjauan pustaka memuat tentang dasar teori yang digunakan untuk analisis, perancangan serta implementasi pada penelitian ini. Selain itu juga sebagai bahan referensi dan pondasi untuk memperkuat argumentasi dalam penelitian ini. Teori - teori yang sesuai dengan penelitian ini antara lain membahas dasar-dasar tanaman jati, varietas, morfologi tumbuhan, pengolahan citra, klasifikasi, Deep Learning dan Convolutional Neural Network. Dasar-dasar tersebut nantinya akan digunakan untuk mendukung dan memperkuat teori dan konsep yang berhubungan dengan masalah yang diteliti dalam pemecahan masalah pada penelitian ini.

BAB III METODOLOGI PENELITIAN

Pada bagian ini akan membahas bahan penelitian, metode analisis data, implementasi jaringan syaraf tiruan dalam mengklasifikasi daun tanaman jati varietas Jati Mega, Jati PH1, dan Jati Plus.

BAB IV HASIL, PENGUJIAN DAN PEMBAHASAN

Pada bab ini akan menyajikan hasil penelitian yang berisi hasil implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Selain itu berisi pengujian terhadap hasil penelitian beserta pembahasannya.

BAB V KESIMPULAN DAN SARAN

Pada bagian ini berisi kesimpulan dari hasil penelitian yang sudah dilakukan dan saran yang diajukan oleh penulis yang dapat diterapkan sebagai masukan yang berguna untuk penelitian yang akan mendatang.

BAB II

TINJAUAN PUSTAKA

2.1 Identifikasi

Identifikasi merupakan kata yang berasal dari bahasa Inggris *identify* yang mengandung makna yaitu meneliti, menelaah dalam kegiatan yang mencari, menemukan, mengumpulkan, mendaftarkan, mencatat data dan informasi dari kebutuhan lapangan.

2.2 Jati

Jati adalah sejenis pohon penghasil kayu bermutu tinggi yang dikenal dunia dengan nama teak (bahasa Inggris). Nama ini berasal dari kata thek ku, dalam bahasa Malayalam, bahasa di negara bagian Kerala di India selatan (Al-Khairi, 2008). Jati (*Tectona grandis Linn.f.*) merupakan salah satu jenis tumbuhan yang sudah banyak dikenal dan dikembangkan oleh masyarakat luas dalam bentuk tanaman hutan maupun tanaman pekarangan rakyat. Hal ini dikarenakan hingga saat ini jati merupakan komoditas kayu mewah, berkualitas tinggi, harga jualnya mahal, dan bernilai ekonomis tinggi. Kayu jati dapat digunakan sebagai bahan dasar pembangunan rumah, konstruksi jembatan, kayu lapis, rangka kusen, pintu, jendela, kerajinan pahat yang bernilai seni tinggi juga untuk furniture.

Jati (*Tectona grandis Linn.f.*) merupakan tanaman yang sangat populer sebagai penghasil bahan baku untuk industri perkayuan karena memiliki kualitas dan nilai jual yang tinggi di masyarakat. Kekuatan dan keindahan seratnya merupakan faktor yang menjadikan kayu jati sebagai pilihan utama (Sukmadjaja & Mariska, 2003). Jati merupakan salah satu jenis kayu tropis yang sangat penting dalam pasar kayu internasional karena berbagai kelebihan yang dimilikinya dan merupakan jenis kayu yang sangat bernilai untuk tanaman kehutanan (Bermejo et al., 2004).

Di Indonesia, jati merupakan salah satu tumbuhan yang mampu memberikan kontribusi nyata dalam menyediakan bahan baku kayu. Kelebihan jati tidak hanya terletak

pada kualitas kayu yang sangat bagus dan bernilai ekonomis sangat tinggi tetapi juga karena sifat-sifat silvikulturnya yang secara umum telah dikuasai. Kayu jati dapat bertahan lama dan sifatnya yang kuat. Jati merupakan jenis yang sudah dikenal dan diusahakan sejak lama, khususnya di Pulau Jawa yang meliputi wilayah Jawa Timur, Jawa Tengah dan Jawa Barat. Di luar Pulau Jawa, jati ditemukan secara terbatas di beberapa tempat di Pulau Sulawesi, Pulau Muna, Pulau Sumbawa, Pulau Bali, Pulau Sumatra dan Pulau Kalimantan (Murtinah et al., 2015).

2.2.1 Perkembangbiakan Jati

Dalam proses keberlanjutan suatu tanaman perlu adanya perkembangbiakan yang dilakukan, supaya kelestarian tanaman dapat terus terjaga dan menghasilkan generasi baru pada suatu tumbuhan. Untuk perkembangbiakan tanaman jati terdiri atas 2 macam sumber pembitian. Diantaranya sebagai berikut:

2.2.1.1 Generatif (Benih)

Pembibitan jati dengan melalui biji yang berasal dari perkawinan jati dewasa, cara ini tidak membutuhkan biaya mahal namun dapat memakan waktu cukup lama karena proses pertumbuhan tanaman tersebut akan berlangsung dari tahap awal. Cara perkembangbiakan tumbuhan secara generatif dapat dibedakan menjadi konjugasi, isogami, anisogami dan penyerbukan.

2.2.1.2 Vegetatif (Stek Pucuk)

Pembibitan tanaman jati yang dilakukan secara konvensional seperti stek, cangkok, okulasi, dan secara modern yaitu dengan cara kultur jaringan. Pucuk daun muda ini biasa diambil dari materi kebun pangkas. Teknik pembibitan berupa stek pucuk yang unggul dihasilkan dari serangkaian tahapan pemuliaan pohon dengan melakukan beberapa proses. Proses-proses tersebut diantaranya adalah pembangunan uji klon dari materi genetik jati hasil pembiakan vegetatif dari klon-klon pohon plus atau pohon induk yang berfenotip baik.

Dari data lapangan yang menghasilkan informasi uji klon diambil materi genetik klon-klon terbaik untuk diambil dijadikan materi genetik kebun pangkas. Materi kebun pangkas berasal dari perbanyakan vegetatif klon-klon terbaik hasil uji klon. Kebun pangkas diperuntukkan untuk menghasilkan materi genetik yang bersifat juvenil sebagai bahan vegetatif stek pucuk jati.

2.3 Varietas

Varietas adalah kelompok tanaman dalam jenis atau spesies tertentu yang dapat dibedakan antara satu kelompok dengan kelompok lain berdasarkan suatu sifat atau sifat-sifat tertentu. Untuk mempertahankan kemurnian agar seragam dan keunggulannya tetap dimiliki, perlu mempelajari sifat-sifat morfologis tanaman seperti tipe tumbuh, warna hipokotil, warna bunga, warna bulu, umur berbunga, dan sifat-sifat kuantitatif seperti tinggi tanaman, ukuran biji, dan ukuran daun (Gani, 2000).

2.3.1 Varietas Mega



Gambar 2.1 Daun Jati Mega

Varietas jati yang dikembangbiakan di hutan lindung Wanagama milik Fakultas Kehutanan Universitas Gadjah Mada, yang berlokasi di Jalan Yogyakarta - Wonosari KM. 30 Gading, Kecamatan Playen, Kabupaten Gunung Kidul, Daerah Istimewa Yogyakarta. Jati

Mega ini termasuk perkembangbiakan vegetatif yang artinya tumbuh kembang biaknya berasal dari stek pucuk daun muda yang dihasilkan jati berusia dewasa yang diambil dari kebun pangkas.

Pohon Jati Mega memiliki ukuran daun dengan diameter yang panjang dan tidak lebar dengan permukaan daunnya yang tidak kasar dan tidak bergelombang, begitu juga pada sisi tepi daunnya yang tidak runcing. Persemaian yang digunakan berusia 3 bulan dengan ketinggian pohon berkisar 20 cm dengan jumlah daun dalam satu pohonnya berkisar 3 hingga 5 lembar daun. Harga jual Jati Mega satu pohnnya yaitu Rp 15,000.

2.3.2 Varietas Perhutani 1(PH1)

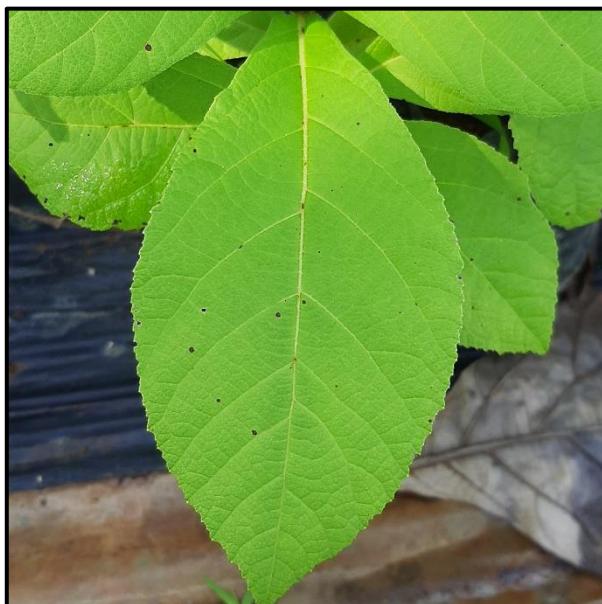


Gambar 2.2 Daun Jati Perhutani 1(PH1)

Varietas Jati yang ditumbuh kembangbiakan oleh Perumahan Hutan Kabupaten Gunung Kidul berlokasi di Persemaian Permanen BPDASHL SOP Area Hutan, Gading, Kecamatan Playen, Kabupaten Gunung Kidul, Daerah Istimewa Yogyakarta. Jati Perhutani 1 ini termasuk kategori perkembangbiakan secara vegetatif, tumbuh kembangbiaknya berasal dari stek pucuk daun muda yang dihasilkan oleh jati berusia dewasa berasal dari kebun pangkas. Bentuk daun Jati Perhutani 1 ini memiliki diameter permukaan yang pendek dan lebar serta teksturnya yang bergelombang dan tepi daun yang runcing. Persemaian yang

digunakan berusia 2 bulan dengan tinggi pohon berkisar 10 cm dengan jumlah daun berkisar 2 hingga 4 lembar daun. Harga jual satu pohonnya Rp 0 atau bisa dibilang gratis, namun harus mengurus berkas-berkas dan ijin terlebih dahulu untuk keperluan administrasi kepada pihak Persemaian Permanen BPDASHL SOP milik dinas Perumahan Hutan Kabupaten Gunung Kidul.

2.3.3 Varietas Plus



Gambar 2.3 Daun Jati Plus

Varietas jati yang dikembangbiakan di kebun persemaian swasta milik P.T. Setya Mitra Baktipersada bertepat di lokasi Ketangi, Banyusoco, Kecamatan Playen, Kabupaten Gunung Kidul, Daerah Istimewa Yogyakarta. Jati Plus termasuk kategori perkembangbiakan secara vegetatif yang artinya berasal dari stek pucuk daun muda dihasilkan dari jati yang sudah berusia dewasa diambil dari kebun pangkas.

Daun Jati Plus memiliki bentuk permukaan daun dengan diameter yang panjang dan juga lebar, serta tepi daun Jati Plus yang runcing. Persemaian yang digunakan berusia 5 bulan dengan tinggi pohon berkisar 30 cm dengan jumlah daun tiap pohnnya berkisar 3 hingga 7 lembar daun. Untuk harga jual persemaian Jati Plus satu pohnnya seharga Rp 15,000.

2.4 Morfologi Daun

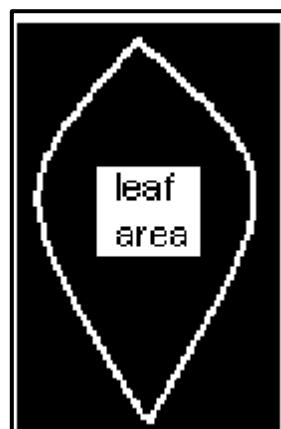
Morfologi tumbuhan merupakan ilmu yang mempelajari bagian organ dan bentuk luar tubuh dari tumbuhan. Ilmu ini mempelajari tumbuhan baik dari segi bentuk maupun fungsinya, namun tidak melakukan pembahasan hingga pada jaringan-jaringan dalam pada tumbuhan, karena pembahasan tersebut sudah termasuk dalam anatomi tumbuhan. Secara umum organ tumbuhan terdiri dari buah, akar, daun, bunga, dan batang.

Daun merupakan salah satu bagian yang ada pada tumbuhan yang terletak pada batang biasanya tipis melebar dan kaya akan zat klorofil yang memiliki tugas atau fungsi sebagai alat fotosintesis pada tumbuhan, berwarna hijau yang disebabkan oleh klorofil pada tumbuhan. Ciri ekstraksi pada daun dibedakan menjadi dua yaitu ciri dasar dan ciri turunan (Wu et al., 2007), diantaranya sebagai berikut:

2.4.1 Dasar

1. Area

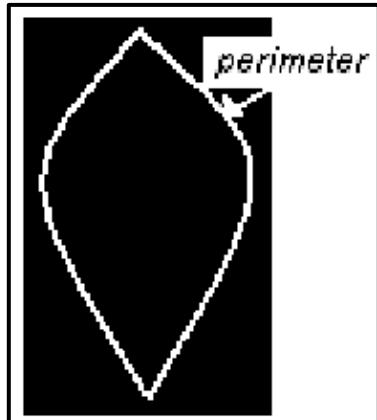
Area merupakan perhitungan jumlah piksel dari daerah yang mencakup tepi daun pada citra yang sudah dihaluskan.



Gambar 2.4 Area Pada Daun

2. Perimeter

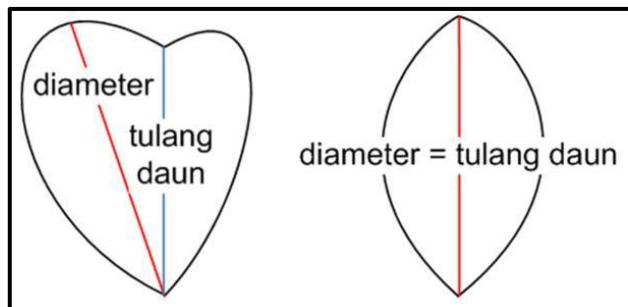
Perimeter adalah perhitungan jumlah piksel yang terdapat pada tepi daun (keliling permukaan daun)



Gambar 2.5 Perimeter Pada Daun

3. Diameter(tulang daun)

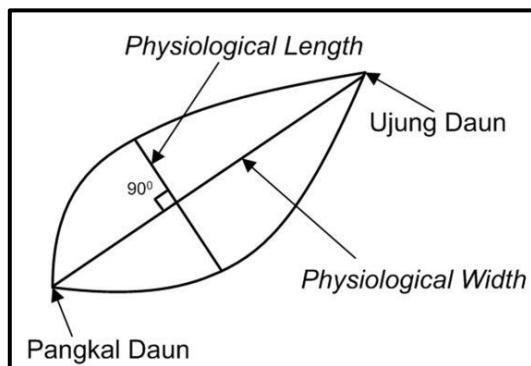
Diameter adalah jarak terpanjang pada daun antara dua titik ujung tepi.



Gambar 2.6 Diameter Pada Daun

4. *Physiological length* dan *physiological width*

Physiological length merupakan jarak antara ujung tepi daun dengan pangkal daun (panjang tulang primer). Sedangkan *physiological width* merupakan



Gambar 2.7 Physiological Length dan Physiological Width

2.4.2 Turunan

Terdapat 12 ciri turunan yang didapatkan dari 5 ciri dasar daun yaitu diantaranya:

1. Smooth Factor

Smooth Factor adalah rasio antara area citra helai daun yang dihaluskan dengan 5x5 rectangular averaging filter dan area citra helai daun yang dihaluskan dengan 2x2 rectangular averaging filter. Ciri ini digunakan untuk mengukur keteraturan tepi daun. Semakin teratur tepi daun, nilainya semakin mendekati 1. Sebaliknya, semakin tidak teratur tepi daun, nilainya semakin mendekati 0.

2. Aspect Ratio

Aspect Ratio adalah rasio antara *physiological length* dan *physiological width*. Maka rumus persamaannya dapat dilihat pada Persamaan 1.

$$\text{Aspect ratio} = \frac{L_p}{W_p} \quad \dots\dots\dots(2.1)$$

Fungsi ciri ini untuk memperkirakan bentuk helai daun. Jika benilai kurang dari 1, maka bentuk helai daun tersebut melebar. Jika benilai lebih dari 1, maka bentuk dari helai daun tersebut memanjang.

3. Form Factor

Form Factor merupakan fitur yang digunakan untuk mendskripsikan bentuk dari daun dan mengetahui seberapa bundar bentuk helai daun tersebut. Nilai *form factor* dapat dilihat pada persemaan ke-dua.

$$\text{Form factor} = \frac{4\pi A}{P^2} \quad \dots\dots\dots(2.2)$$

4. Rectangularity

Rectangularity mendeskripsikan mengenai seberapa perseginya dari masing-masing permukaan daun. Rumusnya diberikan pada persamaan ke-tiga.

$$\text{Rectangularity} = \frac{L_p W_p}{A} \quad \dots\dots\dots(2.3)$$

5. *Narrow factor*

Narrow factor didefinisikan sebagai rasio antara diameter dan physiological length. Ciri ini untuk menentukan apakah bentuk helai daun tersebut tergolong simetri atau asimetri. Jika helai daun tersebut tergolong simetri maka bernilai 1. Jika asimetri maka bernilai lebih dari 1. Nilainya dapat dicari menggunakan Persamaan 4.

$$\text{Narrow factor} = \frac{D}{Lp} \quad \dots\dots\dots(2.4)$$

6. *Perimeter Ratio Of Diameter*

Perimeter ratio of diameter yaitu untuk mengukur seberapa lonjong daun tersebut. Persamaannya dapat dilihat pada Persamaan 5.

$$\text{Perimeter ratio of diameter} = \frac{P}{D} \quad \dots\dots\dots(2.5)$$

7. *Perimeter ratio of Physiological Length And Physiological Width*

Perimeter ratio of physiological length and physiological width. Rumusnya diberikan pada Persamaan 6.

$$\text{Perimeter ratio of physiological length and physiological width} = \frac{P}{(Lp + Wp)} \quad \dots\dots\dots(2.6)$$

8. *Vein features*

Vein features. Persamaannya dapat dilihat pada Persamaan 7, 8, 9, 10 dan 11. Rasio antara area helai daun yang telah dikurangi disk-shaped structuring element dengan radius satu piksel ($Av1$) dan area daun awal (A). Rumusnya menggunakan Persamaan 7.

$$\text{Vein features 1} = \frac{Av1}{A} \quad \dots\dots\dots(2.7)$$

Rasio antara area helai daun yang telah dikurangi disk-shaped structuring element dengan radius dua piksel ($Av2$) dan area daun awal (A). Persamaan 8 merupakan rumus yang digunakan.

$$Vein \ features \ 2 = \frac{A_{v2}}{A} \quad \dots \dots \dots \quad (2.8)$$

Rasio antara area helai daun yang telah dikurangi disk-shaped structuring element dengan radius tiga piksel (Av_3) dan area daun awal (A). Untuk menghitungnya menggunakan Persamaan 9.

Rasio antara area helai daun yang telah dikurangi disk-shaped structuring element dengan radius empat piksel (A_{v4}) dan area daun awal (A). Persamaan 10 merupakan persamaan yang digunakan.

$$Vein \ features \ 4 = \frac{A_{v4}}{A} \quad \dots \dots \dots \quad (2.10)$$

Rasio antara area helai daun yang telah dikurangi disk-shaped structuring element dengan radius empat piksel (Av_4) dan area helai daun yang telah dikurangi disk-shaped structuring element dengan radius satu piksel (Av_1). Persamaannya dapat dilihat pada Persamaan 11.

2.5 Pengenalan Pola

Pola adalah entitas yang terdefinisi dan dapat diidentifikasi melalui ciri-cirinya (*features*) [HEN95]. Ciri-ciri tersebut digunakan untuk membedakan suatu pola dengan pola lainnya. Ciri yang bagus adalah ciri yang memiliki daya pembeda yang tinggi, sehingga pengelompokan pola berdasarkan ciri yang dimiliki dapat dilakukan dengan keakuratan yang tinggi (Munir, 2004).

Tabel 2.1 Contoh Pola dan Ciri

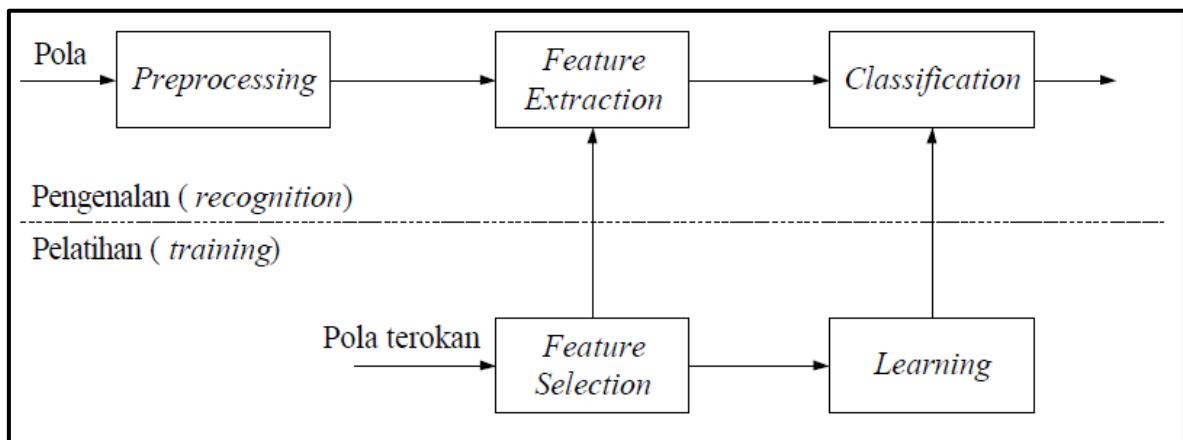
Tabel 2.1 Contoh Pola dan Ciri	
Pola	Ciri
Huruf	Tinggi, tebal, titik sudut, lengkung garis
Suara	Amplitudo, frekuensi, nada, intonasi, warna
Tanda Tangan	Panjang, kerumitan, tekanan
Sidik Jari	Lengkungan, jumlah garis

Pengenalan pola bertujuan menentukan kelompok atau kategori pola berdasarkan ciri-ciri yang dimiliki oleh pola tersebut. Dengan kata lain, pengenalan pola membedakan suatu objek dengan objek lain. Terdapat dua pendekatan yang dilakukan dalam pengenalan pola: pendekatan secara statistik dan pendekatan secara sintaktik atau struktural [HEN95] (Munir, 2004).

Terdapat dua macam pendekatan dalam pengenalan pola, diantaranya sebagai berikut:

2.5.1 Pengenalan Pola secara Statistik

Pendekatan ini menggunakan teori-teori ilmu peluang dan statistik. Ciri-ciri yang dimiliki oleh suatu pola ditentukan distribusi statistiknya. Pola yang berbeda memiliki distribusi yang berbeda pula. Dengan menggunakan teori keputusan di dalam statistik, kita menggunakan distribusi ciri untuk mengklasifikasikan pola (Munir, 2004).



Gambar 2.8 Sistem Pengenalan Pola Dengan Pendekatan Statistic

Ada dua fase dalam sistem pengenalan pola: (i) fase pelatihan dan (ii) fase pengenalan. Pada fase pelatihan, beberapa contoh citra dipelajari untuk menentukan ciri yang akan digunakan dalam proses pengenalan serta prosedur klasifikasinya. Pada fase pengenalan, citra diambil cirinya kemudian ditentukan kelas kelompoknya.

Dengan rincian keterangan proses pada system pengenalan pola dengan pendekatan statistic sebagai berikut:

1. *Preprocessing*

Proses awal yang dilakukan untuk memperbaiki kualitas citra (edge enhancement) dengan menggunakan teknik-teknik pengolahan citra yang sudah diejelaskan pada bab-bab sebelum ini.

2. *Feature Extraction*

Proses mengambil ciri-ciri yang terdapat pada objek di dalam citra. Pada proses ini objek di dalam citra mungkin perlu dideteksi seluruh tepinya, lalu menghitung properti-properti objek yang berkaitan sebagai ciri. Beberapa proses ekstraksi ciri mungkin perlu mengubah citra masukan sebagai citra biner, melakukan penipisan pola, dan sebagainya.

3. *Classification*

Proses mengelompokkan objek ke dalam kelas yang sesuai.

4. *Feature Selection*

Proses memilih ciri pada suatu objek agar diperoleh ciri yang optimum, yaitu ciri yang dapat digunakan untuk membedakan suatu objek dengan objek lainnya.

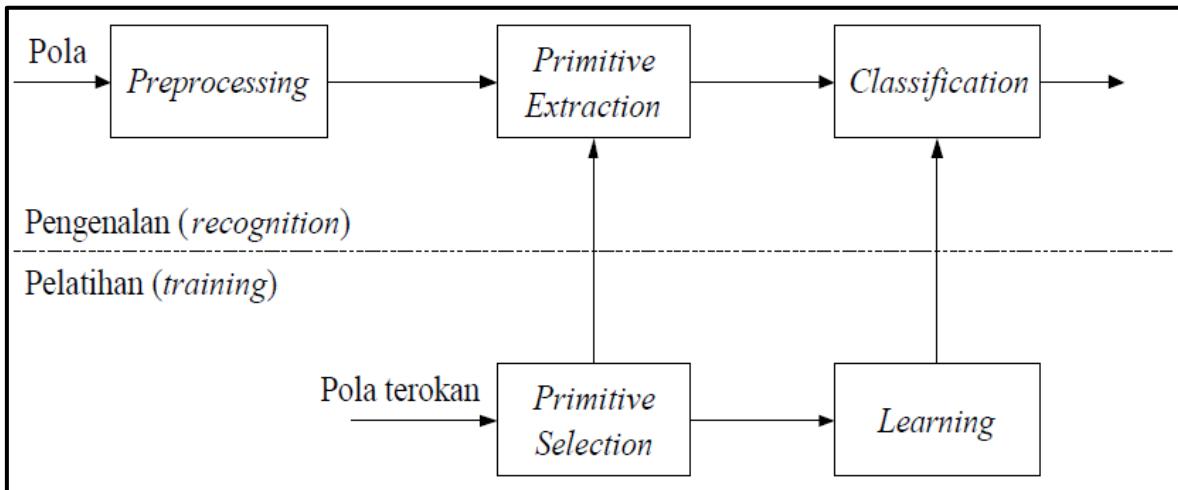
5. *Learning*

Proses belajar membuat aturan klasifikasi sehingga jumlah kelas yang tumpang tindih dibuat sekecil mungkin. Kumpulan ciri dari suatu pola dinyatakan sebagai vektor ciri dalam ruang bahanmata (multi dimensi). Jadi, setiap pola dinyatakan sebagai sebuah titik dalam ruang bahanmata. Ruang bahanmata dibagi menjadi sejumlah uparuang (sub-ruang). Tiap uparuang dibentuk berdasarkan pola-pola yang sudah dikenali kategori dan ciri-cirinya (melalui fase pelatihan).

2.5.2 Pengenalan Pola secara Sintatik

Pendekatan ini menggunakan teori bahasa formal. Ciri-ciri yang terdapat pada suatu pola ditentukan primitif dan hubungan struktural antara primitif kemudian menyusun tata bahasanya. Dari aturan produksi pada tata bahasa tersebut kita dapat menentukan kelompok

pola. Pengenalan pola secara sintaktik lebih dekat ke strategi pengenalan pola yang dilakukan manusia, namun secara praktik penerapannya relatif sulit dibandingkan pengenalan pola secara statistic (Munir, 2004).



Gambar 2.9 Sistem Pengenalan Pola Dengan Pendekatan Sintaktik

Pendekatan yang digunakan dalam membentuk tata bahasa untuk mengenali pola adalah mengikuti kontur (tepi batas) objek dengan sejumlah segmen garis terhubung satu sama lain, lalu mengkodekan setiap garis tersebut (misalnya dengan kode rantai). Setiap segmen garis merepresentasikan primitif pembentuk objek.

2.6 Citra Digital

Citra digital merupakan kombinasi antara titik, garis, bidang dan warna untuk menciptakan suatu imitasi dari suatu objek, biasanya objek fisik atau manusia (Kusumaningrum, 2018). Citra *digital* merupakan suatu matriks di mana indeks baris dan kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya (yang disebut sebagai elemen gambar atau piksel) menyatakan tingkat keabuan pada titik tersebut (Fikriya et al., 2017).

Citra *analog* tidak dapat direpresentasikan dalam komputer, sehingga tidak dapat diproses oleh komputer secara langsung. Citra analog harus dikonversi menjadi citra digital terlebih dahulu agar dapat diproses di komputer (Sutojo, 2017). Proses perubahan citra

analog menjadi citra *digital* disebut dengan digitasi. Digitasi adalah proses mengubah gambar, teks, atau suara dari benda yang dapat diliat ke dalam data elektronik yang dapat disimpan dan diproses untuk keperluan lainnya. Dalam hal lain, pengolahan citra digital merupakan pemrosesan setiap data dua dimensi (Yusri, A.C., 2019).

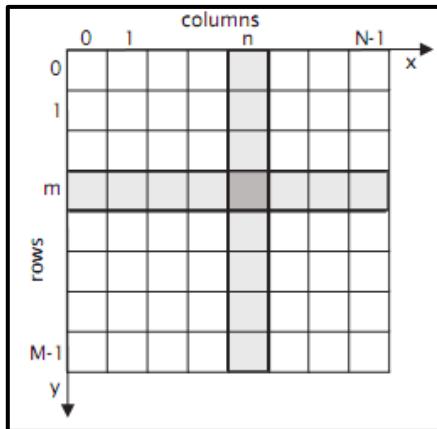
Suatu citra digital memiliki fungsi dua dimensi $f(x,y)$ dengan ukuran M sebagai baris, dan untuk N sebagai kolom, terdapat perpotongan antara kolom dan baris yang disebut dengan *pixel*. Arti dari kata *pixel* sendiri merupakan elemen terkecil dari sebuah citra. *Pixel* memiliki dua parameter, yaitu koordinat dan intensitas atau warna. Dalam sebuah citra digital terdapat sebuah elemen *pixel* berbentuk matriks 2 dimensi. Setiap *pixel-pixel* tersebut memiliki angka yang merepresentasikan nilai *channel* dari setiap warna. Angka yang terdapat pada setiap *pixel* disimpan secara beraturan dan berurutan oleh sebuah program dan memori dengan sering mengolah maupun mengurangi untuk keperluan kompresi dalam suatu pengolahan. Setiap *pixel* tersebut dinyatakan sebagai fungsi $f(x,y)$ memiliki satuan x dan y yang merupakan koordinat spasial, dan amplitudo f pada titik koordinat x dan y disebut sebagai intensitas atau kata lainnya tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai amplitudo f secara keseluruhan berhingga dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital (Putra, 2019). Berikut merupakan posisi koordinat citra digital yang dituliskan dalam bentuk matriks:

Keterangan:

M = jumlah *pixel* baris pada suatu *array* citra

N = jumlah *pixel* kolom pada suatu *array* citra

L = maksimal intensitas warna



Gambar 2.10 Representasi Citra Digital Dalam Bentuk 2 Dimensi

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Gambar 2.11 Representasi Bentuk Matriks Citra Digital

Pada saat menjalankan proses pengolahan citra terdapat empat macam dasar dalam citra digital, diantaranya sebagai berikut:

2.6.1 *Grayscale*(Citra Keabuan)

Citra *grayscale* merupakan sebuah tipe citra yang terdiri dari satu layer warna dengan derajat keabuan tertentu (RD, Kusumanto Alan, Novi, 2011). Pada citra *grayscale* warna 0 memiliki arti warna hitam pekat dan 255 merupakan arti warna putih. Diukur berdasarkan skala intensitas kecerahan yang bernilai 0-255, ini berarti bahwa setiap *pixel* memiliki ukuran 8bit atau 1 byte. Citra *grayscale* sebenarnya merupakan hasil rata rata dari *colour image*, dengan demikian maka persamaannya dapat dituliskan sebagai berikut:

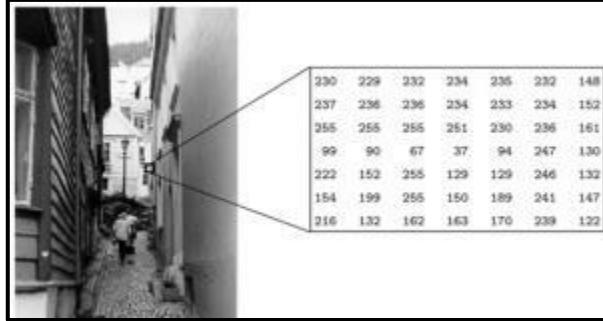
$$I_{BW}(x, y) = \frac{I_R(x, y) + I_G(x, y) + I_B(x, y)}{3} \quad \dots \dots \dots \quad (2.15)$$

Keterangan:

$IR(x, y) = \text{nilai pixel Red titik } (x, y)$

$IG(x, y) = \text{nilai pixel Green titik } (x,y)$

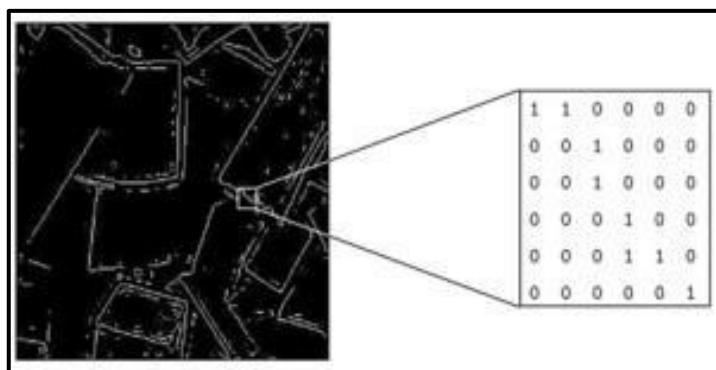
$IB(x, y) = \text{nilai pixel Blue titik } (x,y)$ (RD,Kusumanto Alan,Novi, 2011).



Gambar 2.12 Hitam dan Putih (*Grayscale*)

2.6.2 Biner

Citra biner merupakan sebuah tipe citra yang setiap pixel pada citra hanya memiliki dua nilai saja yaitu 0 dan 1. Nilai 0 mewakili warna hitam dan nilai 1 mewakili warna putih. Karena hanya memiliki 2 nilai yang mungkin untuk setiap *pixel*, maka setiap *pixel* hanya memiliki ukuran 1bit saja. Karena hanya memiliki 2 nilai yang mungkin untuk setiap *pixel*, maka setiap *pixel* hanya memiliki ukuran 1bit saja. Citra dengan tipe biner akan sangat efisien dalam proses penyimpanan karena hanya memiliki ukuran yang sangat kecil. Gambar yang direpresentasikan citra biner sangat cocok untuk teks (dicetak atau tulis tangan), sidik jari (finger print), atau gambar arsitektur (RD, Kusumanto Alan, Novi, 2011). Citra biner memiliki fungsi sebagai berikut:



Gambar 2.13 Binary

$$I_{Bin}(x, y) \begin{cases} 0 & I_{BW}(x, y) < T \\ 255 & I_{BW}(x, y) \geq T \end{cases} \dots \quad (2.16)$$

Keterangan:

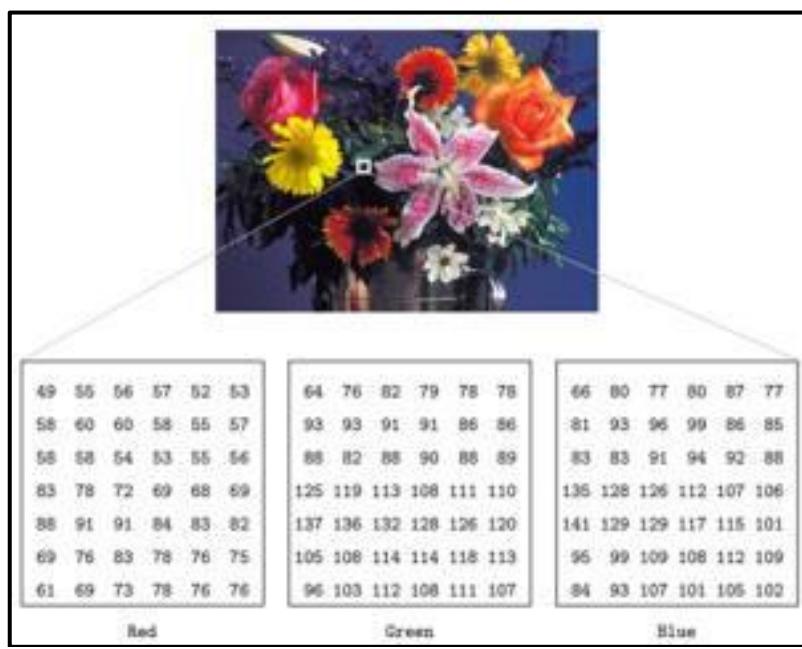
$IBW(x, y)$ = nilai pixel Gray titik(x, y)

$$IBin(x, y) = \text{nilai pixel Binary titik } (x, y)$$

T = nilai *threshold* (RD, Kusumanto Alan, Novi, 2011)

2.6.3 Berwarna(RGB)

Citra berwarna RGB atau *Red-Green-Blue* merupakan sebuah tipe citra yang setiap *pixelnya* terdiri atas 3 komponen warna, terdiri dari merah (R), hijau (G), dan biru (B). Setiap komponen warna memiliki jangkauan nilai antara 0 sampa 255 (8bit). Hal ini akan memberikan kemungkinan total warna sebanyak $255^3 = 16.581.375$. Jadi total ukuran bit untuk setiap *pixelnya* akan memiliki nilai sebanyak 24bit (8bit *Red*, 8bit *Green*, 8bit *Blue*). Citra seperti ini biasanya juga disebut dengan citra warna 24bit atau true *colour* (RD, Kusumanto Alan, Novi, 2011).

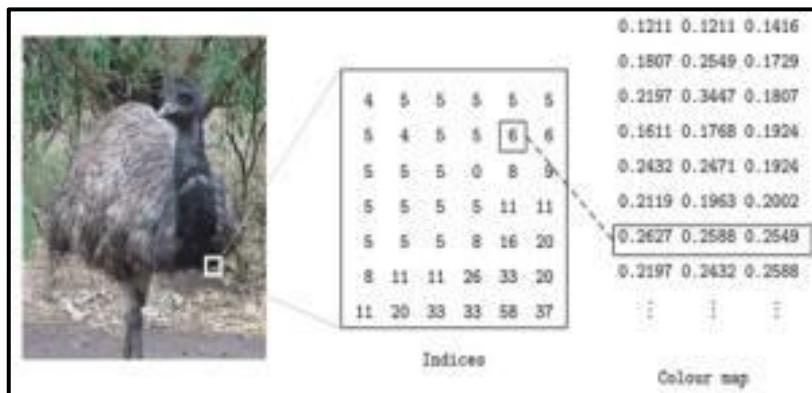


Gambar 2.14 Red Green Blue

2.6.4 Warna Berindeks

Citra warna berindeks merupakan sebuah tipe citra yang hanya memiliki sebagian kecil dari 16 juta warna yang ada. Untuk kenyamanan dalam menyimpan dan penanganan berkas file, citra warna bertipe index mempunyai sebuah peta warna terkait indeks warna yang akan ditampilkan dan hanya menyimpan daftar semua warna yang digunakan pada citra

tersebut. Setiap pixel pada citra warna berindeks mempunyai nilai yang tidak mewakili warna yang diberikan (seperti pada citra warna RGB), tetapi nilai tersebut mewakili sebuah indeks warna yang mana representasi warna tersebut tersimpan pada peta warna (Yusri, A.C., 2019).



Gambar 2.15 Warna Berindeks

2.7 Computer Vision

Computer Vision merupakan cara kerja komputer melalui sebuah program dengan meniru sistem visual yang ada pada manusia. *Computer vision* adalah bidang yang mencakup metode untuk memperoleh, mengolah, menganalisis dan memahami data visual seperti gambar dan video. Tujuan utama dari *computer vision* adalah agar komputer atau mesin dapat meniru kemampuan melihat mata manusia dan otak atau bahkan dapat mengunggulinya untuk tujuan tertentu (Yusri, A.C., 2019).

Human vision sesungguhnya sangat kompleks. Manusia melihat objek dengan indera penglihatan (mata), lalu citra objek diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek apa yang tampak dalam pandangan matanya. Hasil interpretasi ini mungkin digunakan untuk pengambilan keputusan (misalnya menghindar kalau melihat mobil melaju di depan) (Munir, 2004).

Computer vision terdiri dari teknik-teknik untuk mengestimasi ciri-ciri objek di dalam citra, pengukuran ciri yang berkaitan dengan geometri objek, dan menginterpretasi informasi

geometri tersebut. Mungkin berguna bagi anda untuk mengingat persamaan [JAI95] berikut (Munir, 2004):

$$Vision = Geometry + Measurement + Interpretation \dots\dots\dots (2.17)$$

Proses-proses di dalam computer vision dapat dibagi menjadi tiga aktivitas:

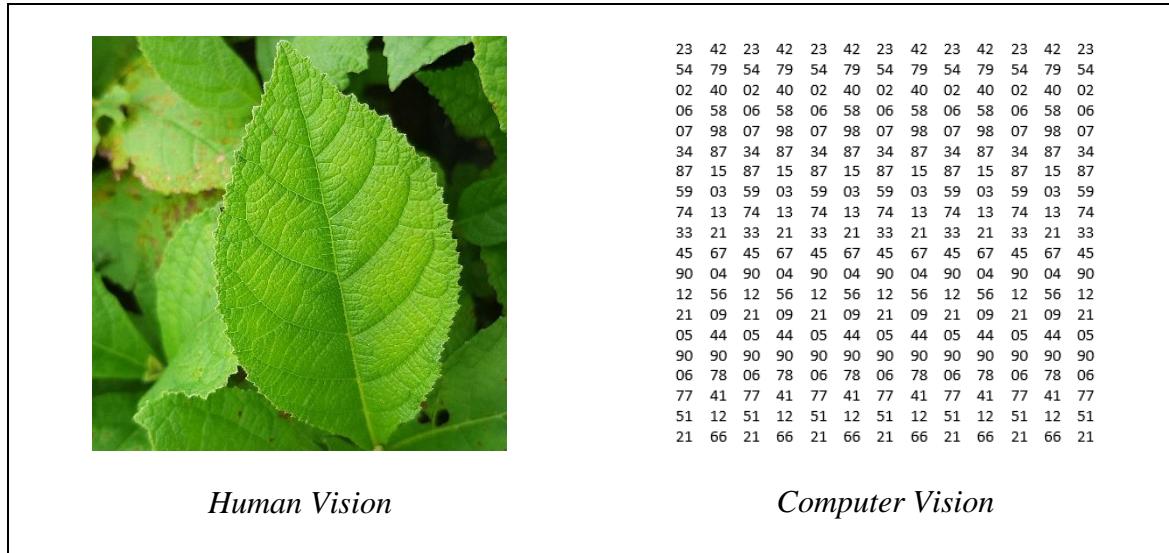
1. Memperoleh atau mengakuisisi citra digital.
2. Melakukan teknik komputasi untuk memproses atau memodifikasi data citra (operasi-operasi pengolahan citra).
3. Menganalisis dan menginterpretasi citra dan menggunakan hasil pemrosesan untuk tujuan tertentu, misalnya memandu robot, mengontrol peralatan, memantau proses manufaktur, dan lain -lain.

Tabel 2.2 Perbedaan *Human Vision* dengan *Computer Vision*

<i>Human Vision</i>	<i>Computer Vision</i>
Menggunakan mata dan visual cortex di dalam otak untuk mengenali objek.	Menggunakan kamera-kamera yang saling terhubung pada sebuah perangkat.
Menemukan dari gambar objek apa yang ada dalam penglihatan, dimana posisinya, bagaimana mereka bergerak, dan seperti apa bentuk wujudnya.	Secara otomatis menginterpretasi gambar-gambar dan mencoba menganalisa untuk menceritau isi pada <i>human vision</i>

Secara garis besar, *computer vision* adalah sebuah teknologi mesin yang mampu mengenali objek yang diamati (Novyantika, 2018). Untuk mengenali hal tersebut, computer vision merupakan kombinasi antara pengolahan citra dan pengenalan pola. Pengolahan citra (*image processing*) merupakan bidang yang berhubungan dengan proses transformasi citra/gambar (*image*).

Proses ini bertujuan untuk mendapatkan kualitas citra yang lebih baik. Sedangkan pengenalan pola (*pattern recognition*) memiliki hubungan dengan proses identifikasi objek pada citra atau interpretasi citra. Proses ini bertujuan untuk mengekstrak informasi/pesan yang akan disampaikan dari citra gambar.



Gambar 2.16 Komparasi Sudut Pandang

2.8 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (JST) adalah paradigma pemrosesan suatu informasi yang terinspirasi oleh sistem sel syaraf biologi. JST dibentuk sebagai generalisasi model matematika dari jaringan syaraf biologi, dengan asumsi bahwa (Agustin M., 2012):

1. Pemrosesan informasi terjadi pada banyak elemen sederhana (*neuron*).
2. Sinyal dikirimkan diantara neuron-neuron melalui penghubung-penghubung.
3. Penghubung antar neuron memiliki bobot yang akan memperkuat atau memperlambat sinyal.
4. Untuk menentukan output, setiap neuron menggunakan fungsi aktivasi (biasanya bukan fungsi linier) yang dikenakan pada jumlahan input yang diterima. Besarnya output ini selanjutnya dibandingkan dengan suatu batas ambang.

Sel saraf atau bisa disebut dengan *Neuron* merupakan satuan kerja utama dari sistem saraf yang berfungsi menghantarkan impuls listrik yang terbentuk akibat adanya suatu stimulus(rangsang), jutaan sel saraf ini membentuk suatu sistem saraf.

Tujuan yang ingin dicapai dengan menerapkan jaringan syaraf tiruan yaitu untuk mencapai keseimbangan antara kemampuan memorisasi dan generalisasi. Yang dimaksud

dengan kemampuan memorisasi adalah kemampuan jaringan syaraf tiruan untuk mengambil kembali secara sempurna sebuah pola yang telah dipelajari. Kemampuan generalisasi adalah kemampuan jaringan syaraf tiruan untuk menghasilkan respon yang bisa diterima terhadap pola-pola yang sebelumnya telah dipelajari (Agustin M., 2012). Oleh karena itu jaringan syaraf tiruan akan memberikan hasil respon yang bagus ketika kondisi data yang dimasukkan belum pernah dipelajari.

Jaringan syaraf tiruan menyerupai otak manusia dalam 2 hal, yaitu (Agustin M., 2012):

1. Pengetahuan diperoleh jaringan melalui proses belajar
2. Kekuatan hubungan antar sel syaraf(*neuron*) yang dikenal sebagai bobot-bobot sinaptik digunakan untuk menyimpan pengetahuan.

Jaringan syaraf tiruan ditentukan oleh 3 hal (Agustin M., 2012):

1. Pola hubungan antar neuron (disebut arsitektur jaringan).
2. Metode untuk menentukan bobot penghubung (disebut metode training/learning).
3. Fungsi aktivasi, yaitu fungsi yang digunakan untuk menentukan keluaran suatu neuron.

2.9 Artificial Intelligence

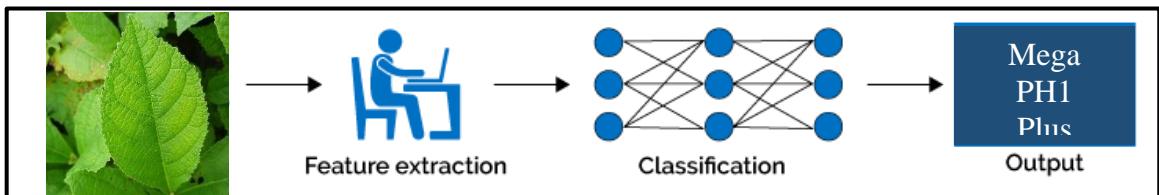
Kecerdasan buatan atau artificial intelligence merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kehidupan umat manusia. Komputer tidak lagi hanya digunakan sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia (Sri Kusumadewi, 2003).

Simon dalam Kusrini, 2006: “Kecerdasan Buatan (*Artificial Intelligence* atau *AI*) merupakan kawasan penelitian, aplikasi, dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas.”

Kecerdasan Buatan atau *Artificial Intelligence* (*AI*) adalah teknik yang digunakan untuk meniru kecerdasan yang dimiliki oleh makhluk hidup maupun benda mati untuk menyelesaikan sebuah persoalan. Untuk melakukan hal ini, setidaknya ada tiga metode yang dikembangkan (Ahmad, 2017).

1. *Fuzzy Logic*(FL). Teknik ini digunakan oleh mesin untuk mengadaptasi bagaimana makhluk hidup menyesuaikan kondisi dengan memberikan keputusan yang tidak kaku 0 atau 1. Sehingga dimunculkan sistem logika fuzzy yang tidak kaku. Penerapan logika fuzzy ini salah satunya adalah untuk sistem penggereman kereta api di Jepang.
2. *Evolutionary Computing*(EC). Pendekatan ini menggunakan skema evolusi yang menggunakan jumlah individu yang banyak dan memberikan sebuah ujian untuk menyeleksi individu terbaik untuk membangkitkan generasi selanjutnya. Seleksi tersebut digunakan untuk mencari solusi dari suatu permasalahan. Contoh dari pendekatan ini adalah Algoritme Genetika yang menggunakan ide mutasi dan kawin silang, *Particle Swarm Optimization* (PSO) yang meniru kumpulan binatang seperti burung dan ikan dalam mencari mangsa, Simulated Annealing yang menirukan bagaimana logam ditempa, dan masih banyak lagi.
3. *Machine Learning* (ML) atau dalam bahasa Indonesinya pembelajaran mesin merupakan teknik yang paling populer karena banyak digunakan untuk menggantikan atau menirukan perilaku manusia untuk menyelesaikan sebuah masalah. Sesuai namanya ML mencoba menirukan bagaimana proses manusia atau makhluk cerdas belajar dan menggeneralisasi suatu masalah untuk memecahkan sebuah masalah yang sederhana hingga rumit.

2.10 Machine Learning



Gambar 2.17 Ilustrasi Proses *Machine Learning*

Machine Learning merupakan teknik untuk melakukan inferensi terhadap data dengan pendekatan matematis. Inti *Machine Learning* adalah untuk membuat model (matematis) yang merefleksikan pola-pola data (Putra, 2019). Dapat didefinisikan *Machine Learning* yaitu ilmu atau studi yang mempelajari tentang algoritma dan model statistik yang digunakan oleh sistem komputer untuk melakukan sebuah tugas atau *task* tertentu tanpa instruksi eksplisit, bergantung pada pola dan kesimpulan yang menghasilkan model matematika yang didasari dari data sampel berupa kumpulan data (*dataset*) yang biasa disebut dengan *training data*. Terdapat tipe-tipe pada *Machine Learning*, diantaranya sebagai berikut:

1. Pembelajaran Terarah (*Supervised Learning*)

Pemrosesan pembuatan model dari kumpulan data (*dataset*) dengan penentuan kelas yang sudah ditentukan sejak sebelum dilakukannya proses pembelajaran, dengan kata lain sudah ditentukan nama-nama kelas yang akan menjadi keluaran dari dibuatnya program. Contoh metodenya adalah KNN(*K-Nearest Neighbor*), SVM(*Super Vector Machine*), *Linear Regression*, *Convolutional Neural Network*, dan lain-lain.

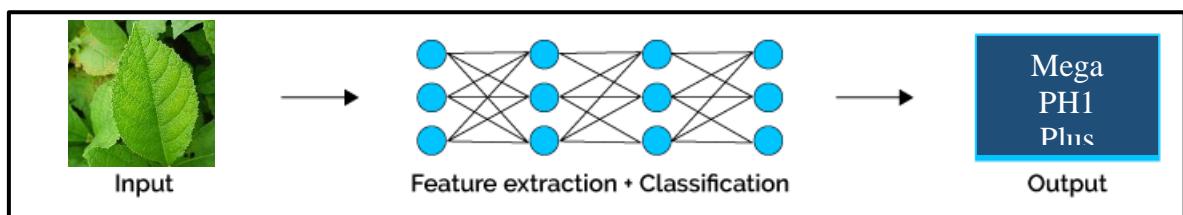
2. Pembelajaran Tidak Terarah (*Unsupervised Learning*)

Pembelajaran data yang tidak memerlukan penentuan kelas, jika *Supervised Learning* nama kelas keluarannya sudah ditentukan untuk *Unsupervised Learning* ini cukup mempelajari model tanpa harus mengetahui nama-nama kelas dari setiap polanya. Contoh metode yang terdapat pada *unsupervised learning* diantanya *K-Means*, *Principal Component Analysis*, dan lain-lain.

3. Pembelajaran Semi Pembelajaran Terarah(*Semi Supervised Learning*)

Membangun model dengan menerapkan gabungan antara *Supervised Learning* dan *Unsupervised Learning* yang artinya menerapkan beberapa data yang sudah berlabel dan lainnya belum berlabel. Teknik ini biasa diterapkan apabila memiliki data yang semuanya belum berlabel dan cocok untuk memprediksi data tanpa label dengan data learning yang ada.

2.11 *Deep Learning*



Gambar 2.18 Ilustrasi Alur *Deep Learning*

Cabang keilmuan yang lebih mendalam terdapat pada *Machine Learning* atau lebih tepatnya pengembangan dari *Machine Learning* yang berbasis Jaringan Syaraf Tiruan(JST). Ilmu ini mempelajari secara langsung dengan sendirinya mulai dari *feature extraction layer* sampai bisa mengelompokan kelas dengan menggunakan beberapa lapisan untuk mempermudah pembelajaran model baik *classification* maupun *clustering*. Arsitektur yang termasuk dalam *deep learning* adalah *deep neural networks* (DNNs), *convolutional neural networks* (CNNs), *recurrent neural networks* (RNNs), generative adversarial networks (GAN), dan masih banyak lainnya (Nguyen et al., 2019).

2.12 *Preprocessing*

Persiapan pengolahan merupakan tahap yang dilakukan sebelum melakukan pemrosesan data dimulai, berawal dengan menyiapkan terlebih dahulu data-data yang nantinya akan digunakan dalam pembelajaran sebagai data masukan untuk memberikan batasan dan memperbanyak variasi data yang telah dimiliki. Berikut merupakan proses-proses yang dilakukan pada penelitian ini:

2.12.1 *Image Augmented*

Fungsi yang disediakan dari *library tensorflow keras* untuk memperbanyak variasi dataset foto daun yang dimiliki dengan melakukan *rescale*(faktor penskalaan. Default-nya adalah Tidak Ada. Jika Tidak Ada atau 0, tidak ada penskalaan yang diterapkan, jika tidak, kami mengalikan data dengan nilai yang diberikan (setelah menerapkan semua transformasi lainnya), *width_sift*(pelabaran dan pengurangan dimensi), *height_sift*(menaikkan dan memendekkan dimensi), *horizontal_flp*(pencerminan), *shear*(Intensitas Geser (Sudut geser berlawanan arah jarum jam dalam derajat)), *rotation*(Rentang derajat untuk rotasi acak), *validation_split*(Pecahan gambar disediakan untuk validasi (antara 0 dan 1).), dan *fill mode nearest*(Poin di luar batas input diisi sesuai dengan mode yang diberikan(aaaaaaaa|abcd|dddddd)).

2.12.2 *Batch*

Jumlah dari banyaknya kelompok dalam satu kali pembelajaran data. Contoh 16 kelompok, 32, 64, 128, dan 256.

2.12.3 *Image Size*

Ukuran default gambar yang akan dijadikan data masukan. Contoh 256 pixel x 256 pixel.

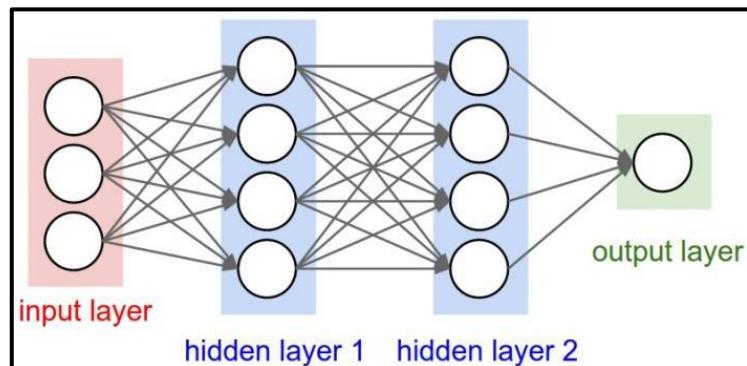
2.12.4 *Split Data*

Pembagian antara data latih dengan data validasi pada saat pembelajaran model. Data latih digunakan untuk diketahui masing-masing nilai dari setiap *array pixel* nya, sedangkan data validasi untuk mencocokan seberapa akuratkah kemiripan data ketika dicek keakuratannya. Contoh 70 % data latih dan 30% data validasi.

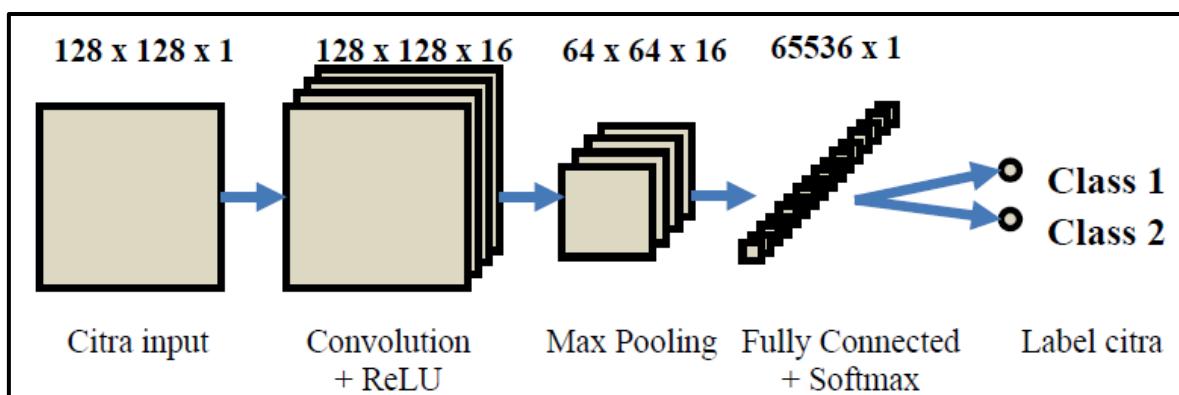
2.13 *Convolution Neural Network(CNN)*

Metode yang diterapkan pada penelitian ini adalah *convolution neural network* yang termasuk dalam pembelajaran terarah (*supervised learning*) dan memiliki banyak lapisan

pembelajaran data sehingga tergolong metode yang ada pada *deep learning*, dengan kata lain *programmer* atau pembuat kode tidak perlu menentukan fitur-fitur mana saja yang dijadikan model pembelajaran. Metode ini merupakan pengembangan dari *multi layer perceptron(MLP)* yang didesain untuk mengolah data dua dimensi dalam bentuk citra.



Gambar 2.19 Arsitektur Sederhana MLP



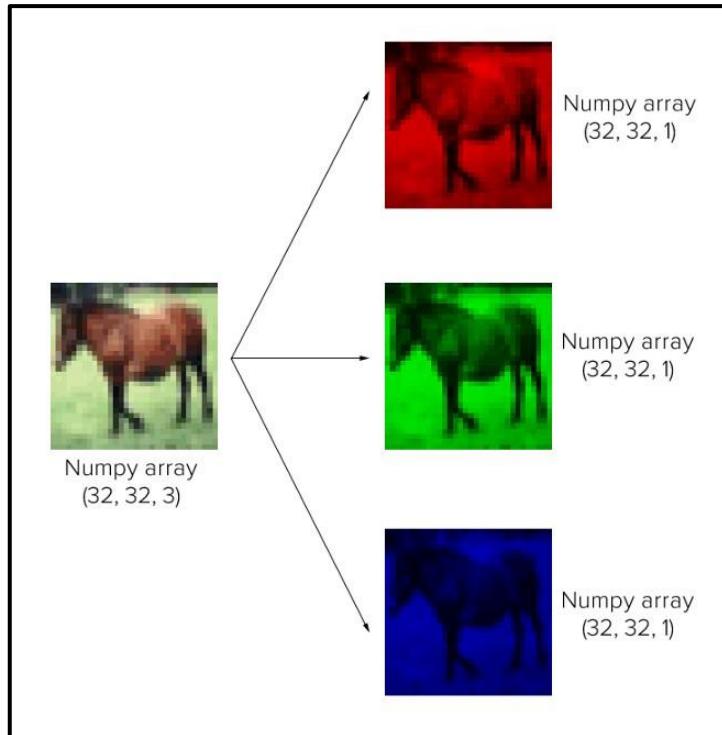
Gambar 2.20 Visualisasi Arsitektur CNN Secara Umum

Convolutional neural network merupakan jaringan yang spesial didesain untuk mengatasi data yang berupa citra gambar atau dengan kata lain dapat juga mengatasi data yang *invariant* data (Nguyen et al., 2019).

2.13.1 Convolution Layer (Konvolusi)

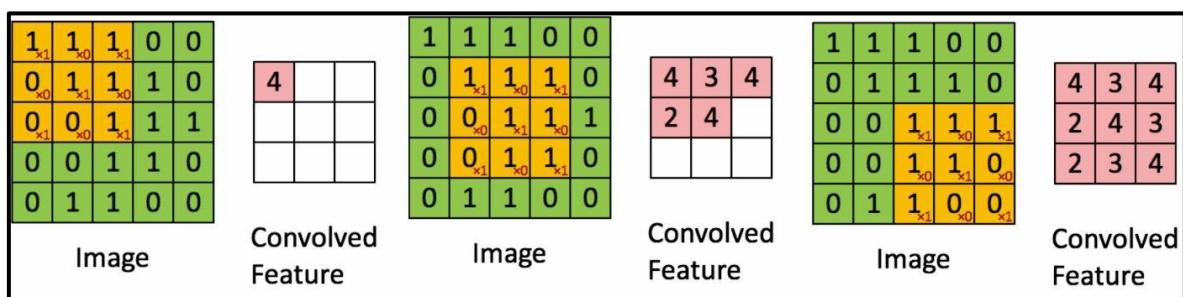
Proses pada *layer* ini melakukan operasi konvolusi yaitu dengan mengkombinasikan *linier filter* terhadap daerah lokal. *Layer* ini yang pertama kali menerima gambar yang diinputkan pada arsitektur. Bentuk *layer* ini adalah sebuah *filter* dengan panjang (*pixel*), lebar (*pixel*) dan tebal sesuai dengan channel image data yang diinputkan. Ketiga *filter* ini akan berpindah tempat keseluruh bagian gambar yang diinputkan. Pergeseran tersebut akan

melakukan operasi “dot” antara *input* dan nilai dari *filter* tersebut sehingga akan menghasilkan *output* yang disebut *activation map* atau *feature map* (Nurfita et al., 2014).



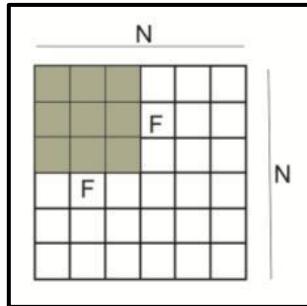
Gambar 2.21 Ilustrasi Channel Red Green Blue

Pada gambar 2.17 merupakan penjabaran dari 1 gambar yang terdapat 3 *channel* warna yaitu *red*, *green*, dan *blue* yang sebenarnya merupakan gambar *multidimensional array* dengan masing-masing ukuran *pixelnya* adalah 32x32. Maka dapat diartikan ukuran *array* pada 1 gambar kuda yaitu 32x32x3.



Gambar 2.22 Ilustrasi Alur Konvolusi

Perkalian pada setiap nilai *pixel* yang dijadikan data *input* dengan nilai konvolusi, dimulai dari array *pixel* [0][0] hingga [n][n] untuk mendapatkan nilai fitur baru atau bisa disebut *feature map*.



Gambar 2.23 Perhitungan Jumlah Konvolusi

$$\frac{(N-F+2P)}{S} + 1 \quad \dots \dots \dots \quad (2.18)$$

Keterangan:

N = Banyak pixel dalam 1 baris/1 kolom (*input*)

F = Banyak pixel dalam 1 baris/1 kolom kernel (*filter*)

P = Pelebaran banyak pixel(*padding*)

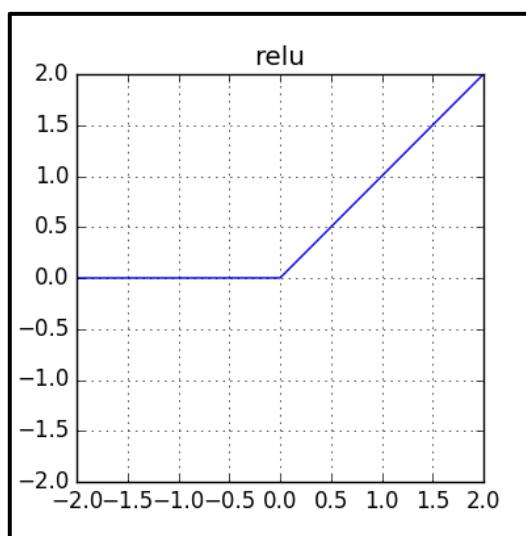
S = Pergeseran ke kanan dan kebawah (*stride*)

+1 = Nilai Bias

a. Fungsi Aktivasi (ReLU)

Fungsi ReLu adalah fungsi yang nilai keluaran suatu neuron dapat dinyatakan sebagai 0 jika nilai masukan negatif. Jika nilai masukan positif, keluaran neuron adalah aktivasi nilai masukan itu sendiri. Persamaan fungsi tersebut dapat dilihat pada persamaan (2.19).

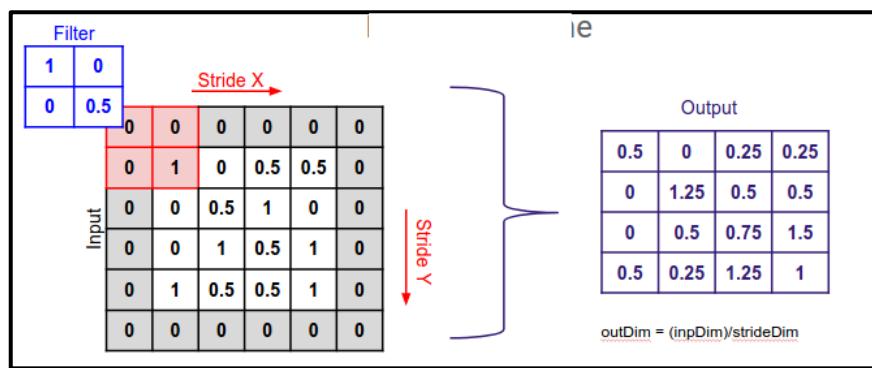
$$f(x) = \max(0, x) \quad \dots \dots \dots \quad (2.19)$$



Gambar 2.24 Diagram Garis Pembentukan RELU

Rectified Linear Unit (ReLU) adalah fungsi aktivasi yang memiliki perhitungan sederhana. Proses forward dan backward melalui ReLU hanya menggunakan kondisi if. Jika elemen bernilai negatif maka nilainya diset menjadi 0, tidak ada operasi eksponensial, perkalian atau pembagian. Dengan karakteristik seperti itu, kelebihan ReLU akan muncul saat berhadapan dengan jaringan yang memiliki neuron yang banyak sehingga dapat mengurangi waktu training dan testing dengan signifikan (Wibawa, 2017).

b. Stride



Gambar 2.25 Ilustrasi Stride

Stride adalah parameter yang menentukan berapa jumlah pergeseran *filter*. Jika nilai stride adalah 1, maka conv. filter akan bergeser sebanyak 1 pixels secara *horizontal* lalu *vertical*. Pada ilustrasi diatas, stride yang digunakan adalah 2. Semakin kecil stride maka akan semakin detail informasi yang kita dapatkan dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan stride yang besar. Namun perlu diperhatikan bahwa dengan menggunakan *stride* yang kecil kita tidak selalu akan mendapatkan performa yang bagus.

c. Padding

Sedangkan *Padding* atau *Zero Padding* adalah parameter yang menentukan jumlah pixels (berisi nilai 0) yang akan ditambahkan di setiap sisi dari input. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi *output* dari conv. layer (*Feature Map*).

Tujuan dari penggunaan *padding* adalah :

Dimensi output dari conv. *layer* selalu lebih kecil dari inputnya (kecuali penggunaan 1×1 filter dengan stride 1). *Output* ini akan digunakan kembali sebagai *input* dari conv. *layer* selanjutnya, sehingga makin banyak informasi yang terbuang. Dengan menggunakan *padding*, kita dapat mengatur dimensi output agar tetap sama seperti dimensi input atau setidaknya tidak berkurang secara drastis. Sehingga kita bisa menggunakan conv. *layer* yang lebih dalam/deep sehingga lebih banyak *features* yang berhasil di-extract.

Image						
0	0	0	0	0	0	0
0						0
0						0
0						0
0						0
0						0
0	0	0	0	0	0	0

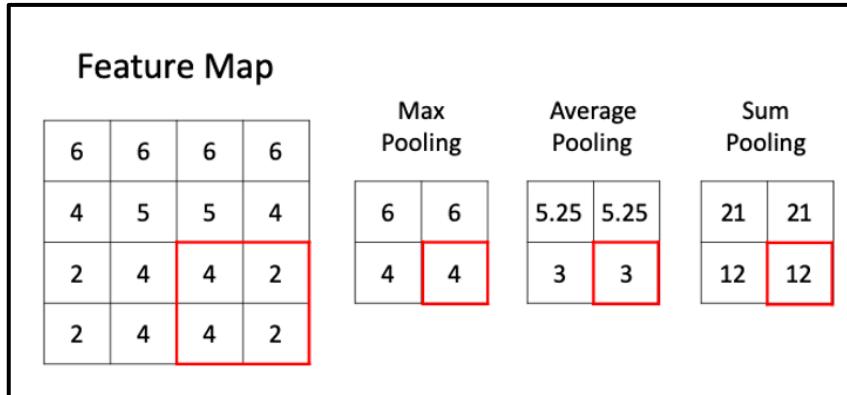
Gambar 2.26 Ilustrasi *Padding*

Meningkatkan performa dari model karena conv. filter akan fokus pada informasi yang sebenarnya yaitu yang berada diantara *zero padding* tersebut.

2.13.2 Pooling Layer

Data yang diterima pada *Pooling layer* berasal dari output yang diolah pada tahap *convolution layer*, pada *layer* ini ukuran data citra akan direduksi. Prinsipnya *pooling layer* terdiri dari *filter* dengan ukuran tertentu dan stride/langkah kemudian bergeser keseluruh area feature map. Sebagian besar arsitektur CNN, metode pooling yang digunakan adalah Max pooling (Nurfita et al., 2014). *Pooling layer* digunakan untuk mempercepat komputasi karena parameter yang harus di *update* semakin sedikit dan mengatasi *overfitting*.

Terdapat 3 macam pengambilan nilai diantaranya



Gambar 2.27 Ilustrasi Pengambilan Nilai

1. *Max Pooling*

Max yang berartikan maksimal maka pada *layer* ini menerapkan pengambilan nilai maksimal pada tiap-tiap kernel yang berjalan untuk dikumpulkan menjadi satu menjadi sebuah fitur baru.

2. *Average Pooling*

Berasal dari kata bahasa inggris *average* yang artinya adalah rata-rata maka pada tahap ini mengimplementasikan pengambilan nilai rata-rata pada setiap kernel yang berjalan untuk menghasilkan nilai fitur baru.

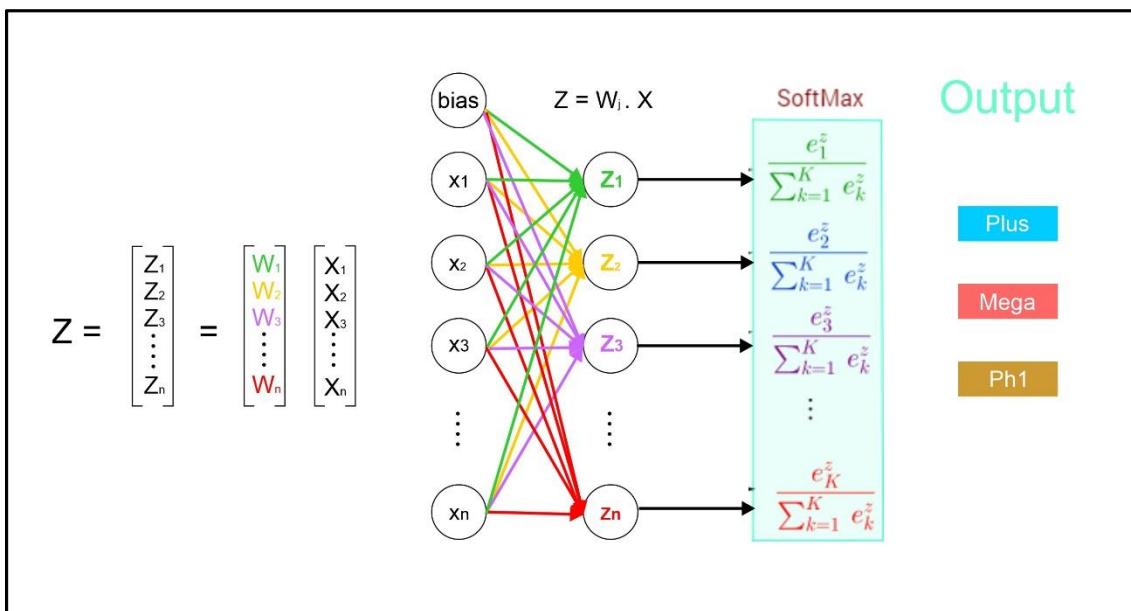
3. *Sum Pooling*

Menjumlahkan semua nilai-nilai *pixel* yang ada pada setiap kernel kemudian dikumpulkan untuk menjadi suatu fitur baru.

2.13.3 Fully Connected Layer

Lapisan terakhir dalam *Convolutional neural network* adalah *Fully Connected Layer*, yang berfungsi melakukan klasifikasi berdasarkan fitur yang diperoleh pada perhitungan lapisan sebelumnya. Apabila pada lapisan CONV dipergunakan ReLU sebagai fungsi aktivasi, maka pada lapisan FC digunakan fungsi *softmax* sebagai fungsi aktivasinya. Fungsi *softmax* dipilih karena dapat mengurangi kemungkinan nilai error yang dihasilkan oleh fungsi objektif *cross-entropy* (Rokhana et al., 2019).

Pada lapisan ini juga berfungsi mengubah multidimensional array ke dalam sebuah vektor. Lapisan *fully connected layer* merupakan kumpulan dari proses konvolusi (Hijazi et al., 2015.). Lapisan ini mendapatkan input dari proses sebelumnya untuk menentukan fitur mana yang paling berkorelasi dengan kelas tertentu. Fungsi dari lapisan ini adalah untuk menyatukan semua node menjadi satu dimensi (Albelwi & Mahmood, 2017). Activation map yang dihasilkan dari feature extraction layer masih berbentuk *multidimensional array*, sehingga mau tidak mau kita harus melakukan reshape activation map menjadi sebuah vektor agar bisa digunakan sebagai input dari *fully-connected layer*.



Gambar 2.28 Ilustrasi Proses pada *Fully Connected Layer*

Layer ini memiliki *hidden layer*, *activation function*, *output layer*, dan *loss function*. Layer ini adalah *layer* yang biasanya digunakan dalam penerapan multi layer perceptron dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear.

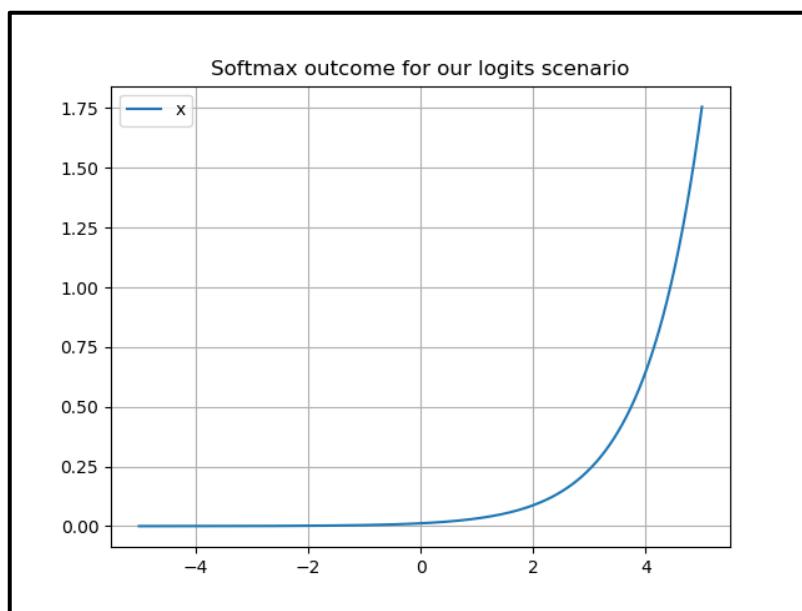
Setiap *neuron* pada *convolution layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan ke dalam sebuah *fully-connected layer* (Andika et al., 2019). Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak *reversibel*, sedangkan *fully-connected layer* hanya dapat diimplementasikan di akhir

jaringan. Setiap neuron pada lapisan konvolusi perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan ke dalam sebuah *fully-connected layer*.

Convolution layer dengan ukuran kernel 1 x 1 melakukan fungsi yang sama dengan *fully-connected layer* namun dengan tetap mempertahankan karakter spasial dari data. Sehingga hal tersebut membuat penggunaan layer ini pada CNN sekarang tidak banyak dipakai.

Selain arsitektur yang telah dipaparkan, masih banyak arsitektur lain yang dapat digunakan untuk berbagai karakteristik data maupun arsitektur yang cocok untuk karakteristik/permasalahan tertentu. Berikut merupakan beberapa parameter yang terdapat pada *fully connected layer*.

a. Fungsi Aktivasi (*Softmax*)



Gambar 2.29 Grafik Skenario *Softmax*

Fungsi aktivasi *softmax* digunakan untuk mendapatkan hasil klasifikasi. Fungsi aktivasi menghasilkan nilai yang diinterpretasi sebagai probabilitas yang belum dinormalisasi untuk tiap kelas. Nilai kelas dihitung dengan menggunakan fungsi softmax (Vedaldi & Lenc, 2015), yang ditunjukan oleh Persamaan 2.19.

$$y_{ijk} = \frac{e^{x_{ijk}}}{\sum_{t=1}^D e^{x_{ijt}}} \quad \dots \dots \dots \quad (2.19)$$

Keterangan:

y_{ijk} = vektor yang berisi nilai antara 0 dan 1.

X = vektor yang berisi nilai yang didapatkan dari lapisan *fully-connected* terakhir.

$$\ell(\mathbf{x}, c) = -\log \frac{e^{x_c}}{\sum_{k=1}^C e^{x_k}} = -x_c + \log \sum_{k=1}^C e^{x_k}. \quad \dots \dots \dots \quad (2.20)$$

Keterangan:

$l(x,c)$ = membandingkan prediksi (x) dan label (c).

x = vektor dari probabilitas akhir.

$p(k) = x_k, k = 1$.

C = banyak kelas.

b. Optimasi Adam

Optimasi pada CNN dapat membantu meningkatkan akurasi, salah satunya adalah adaptive moments atau adam (Kingma & Ba, 2015). Optimasi adam digunakan untuk meningkatkan akurasi dari model yang telah dibuat. Adam adalah algoritma optimisasi tingkat pembelajaran adaptif lain. Nama “Adam” berasal dari frasa “adaptive moments”(Kingma & Ba, 2015).

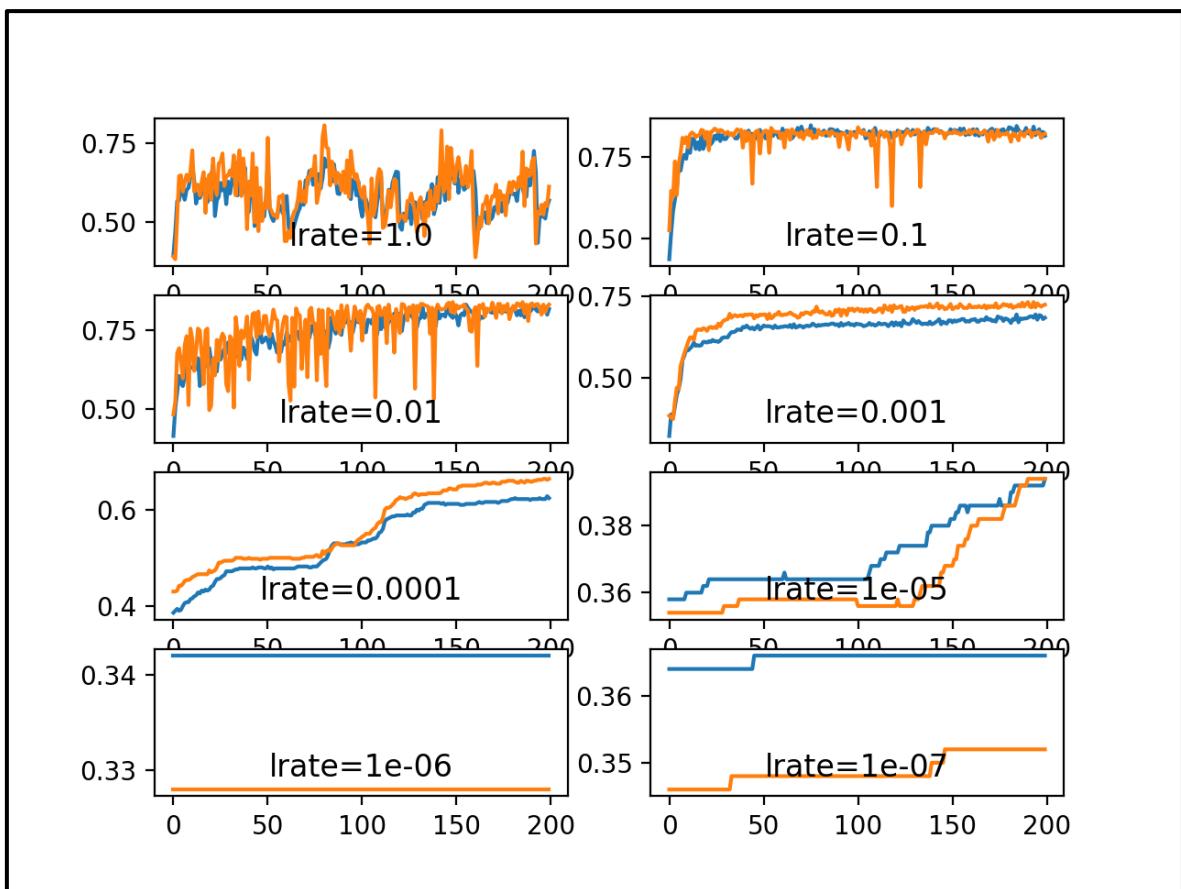
Adam merupakan kombinasi RMSProp dan momentum dengan beberapa perbedaan penting. Pertama momentum digabungkan secara langsung sebagai perkiraan momen orde pertama (dengan bobot eksponensial) dari gradien. Kedua, Adam memasukkan koreksi-koreksi bias ke estimasi momen-momen orde pertama (*momentum term*) dan momen-momen orde kedua (tidak terpusat) untuk menjelaskan inisialisasi origin. Output dari pemilihan jumlah *epochs* optimasi adalah meningkatnya akurasi serta menurunnya *loss function* pada

data training walapun menggunakan pemodelan CNN yang sama (Andika et al., 2019).

Adapun penjelasan penggunaan *learning rate* pada optimasi di *layer* ini,

Adam merupakan algoritma optimisasi stokastik berdasarkan perkiraan adaptif dari momen order rendah. Algoritma Adam pertama kali diperkenalkan oleh Kingma & Ba (Wibawa, 2017). Metode ini dapat diimplementasikan dengan mudah, memiliki komputasi yang efisien, memiliki kebutuhan memori yang kecil, invariant terhadap penskalaan gradien dan cocok diterapkan pada data atau parameter dengan jumlah yang besar. Algoritma Adam cocok diterapkan pada permasalahan data yang sangat berderau atau gradien yang menyebar (Wibawa, 2017).

1. Learning Rate



Gambar 2.30 Komparasi Nilai *Learning Rate* (Jason Brownlee, 2019).

Learning rate adalah salah satu *hyper parameter* yang sangat mempengaruhi performa suatu model CNN (Kamal Hasan et al., 2019). The learning rate adalah hyperparameter yang

mengontrol seberapa banyak model harus diubah sebagai respons terhadap estimasi error setiap kali bobot model diperbarui. Memilih kecepatan pemelajaran cukup menantang karena nilai yang terlalu kecil dapat mengakibatkan proses pelatihan yang lama yang dapat macet, sedangkan nilai yang terlalu besar dapat mengakibatkan mempelajari kumpulan bobot yang kurang optimal terlalu cepat atau proses pelatihan yang tidak stabil.

Learning rate mengontrol seberapa cepat model beradaptasi dengan masalah. Kecepatan pembelajaran yang lebih kecil memerlukan lebih banyak periode pelatihan karena perubahan yang lebih kecil dilakukan pada bobot setiap pembaruan, sedangkan kecepatan pembelajaran yang lebih besar menghasilkan perubahan yang cepat dan memerlukan waktu pelatihan yang lebih sedikit.

Learning rate yang terlalu besar dapat menyebabkan model terlalu cepat menyatu ke solusi yang kurang optimal, sedangkan kecepatan pembelajaran yang terlalu kecil dapat menyebabkan proses macet.

c. *Loss Function (Cross Entropy)*

Selama pelatihan jaringan saraf, cost function adalah kuncinya untuk menyesuaikan bobot jaringan neural untuk membuat model pembelajaran mesin yang lebih baik. Secara khusus, selama forward propagation, jaringan saraf dijalankan pada data set pelatihan, dan keluaran yang dihasilkan dalam hal klasifikasi menunjukkan probabilitas atau ketepatan pada label yang memungkinkan (Ho & Wookey, 2020).

Probabilitas ini dibandingkan dengan label target, dan, loss function menghitung penalti untuk setiap deviasi antara label target dan keluaran jaringan neural. Selama backpropagation merupakan turunan parsial dari fungsi kerugian dihitung untuk setiap bobot jaringan saraf yang dapat dilatih. Bobot disesuaikan dengan turunan parsial ini. Dibawah kondisi normal, backpropagation menyesuaikan secara berulang bobot yang dapat dilatih dari jaringan neural untuk menghasilkan model dengan nilai *loss* yang terendah

Standart binary cross-entropy loss function dapat ditulis dengan (Ho & Wookey, 2020):

$$J_{bce} = -\frac{1}{M} \sum_{m=1}^M [y_m \times \log(h_\theta(x_m)) + (1 - y_m) \times \log(1 - h_\theta(x_m))] \quad \dots \quad (2.21)$$

Keterangan:

M = Banyak contoh data latih

ym = label target contoh data latih m

xm = masukan untuk pelatihan contoh m

h_0 = model dengan *neural network weights*

Bobot tambahan dapat diatur untuk menyesuaikan tingkat kepentingannya dari label positif. Penggunaan yang umum adalah memberi lebih banyak bobot ke kelas minoritas. Untuk kasus label tunggal, klasifikasi kategorikal (mis. aktivasi softmax) cross-entropy categorical standar kerugian dapat dinyatakan dengan (Ho & Wookey, 2020):

$$J_{cce} = -\frac{1}{M} \sum_{k=1}^K \sum_{m=1}^M y_m^k \times \log(h_\theta(x_m, k)) \dots \quad (2.22)$$

Keterangan:

M = Banyak contoh data latih

K = Banyak kelas

y_{km} = label target untuk data latih m untuk kelas k

x = input untuk pelatihan contoh m

untuk standar bobot *categorical cross-entropy* dapat dinyatakan dengan (Ho & Wookey, 2020):

$$J_{wcce} = -\frac{1}{M} \sum_{k=1}^K \sum_{m=1}^M w_k \times y_m^k \times \log(h_\theta(x_m, k)) \quad \dots \quad (2.23)$$

Keterangan:

M = Banyak contoh data latih

K = Banyak kelas

w_k = bobot kelas k

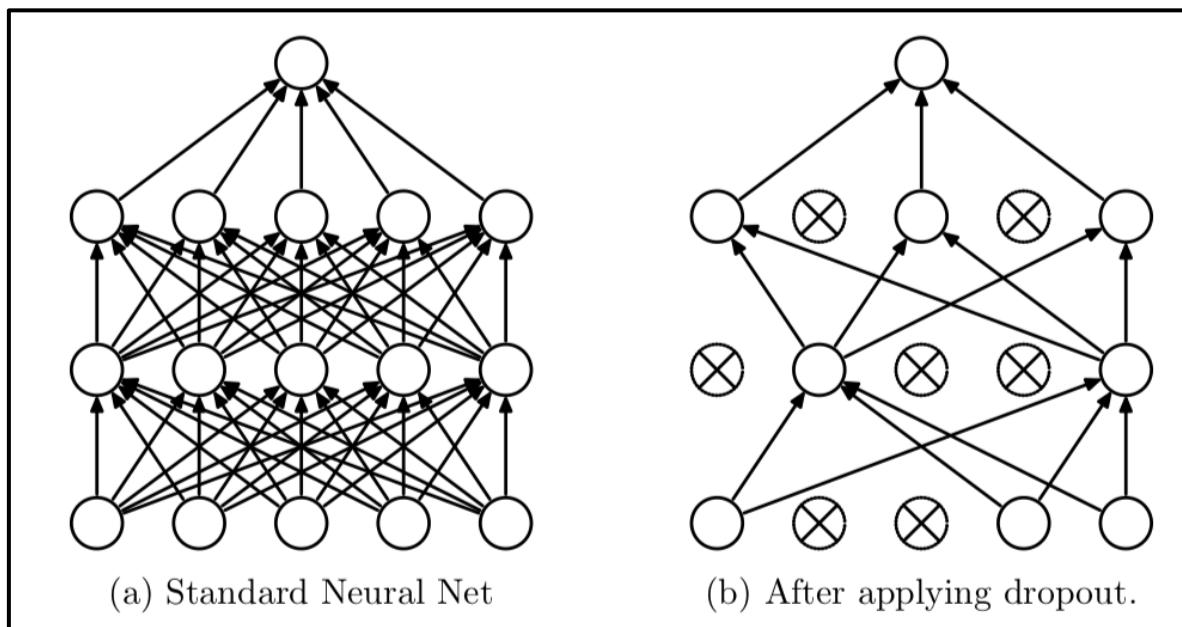
y_m = label target contoh data latih m

x_m = masukan untuk pelatihan contoh m

h_0 = model dengan *neural network weights*

2.13.4 Dropout

Dropout adalah teknik regularisasi jaringan syaraf dimana beberapa neuron akan dipilih secara acak dan tidak dipakai selama pelatihan. Neuron-neuron ini dapat dibilang dibuang secara acak. Hal ini berarti bahwa kontribusi neuron yang dibuang akan diberhentikan sementara jaringan dan bobot baru juga tidak diterapkan pada neuron pada saat melakukan *backpropagation* (Lina Q., 2018).



Gambar 2.31 Ilustrasi Penghilangan Node(*dropout*)

Dropout merupakan proses mencegah terjadinya overfitting dan juga mempercepat proses learning. Dropout mengacu kepada menghilangkan neuron yang berupa hidden mapun layer yang visible di dalam jaringan. Dengan menghilangkan suatu neuron, berarti

menghilangkannya sementara dari jaringan yang ada. Neuron yang akan dihilangkan akan dipilih secara acak. Setiap neuron akan diberikan probabilitas yang bernilai antara 0 dan 1(Lina Q., 2018).

2.14 Confusion Matrix

Pengertian dari *Confusion matrix* adalah salah satu cara yang digunakan dalam mengevaluasi metode-metode klasifikasi selama pembelajaran model. Berikut ini merupakan gambaran sketsa sedehana untuk mempermudah memahami mengenai istilah *confusion matrix* dalam keluaran klasifikasi maupun *clustering*.

Tabel 2.3 Skema Perhitungan *confusion matrix*

		Kelas Prediksi (<i>observation</i>)	
		Positif	Negative
Kelas Aktual (<i>expection</i>)	Positif	TP	TN
	Negative	FP	FN

Nilai *True Negative* (TN) adalah data yang di klasifikasi dengan tepat sebagai keluaran negatif atau salah. *True Positive* (TP) adalah data yang diklasifikasi dengan tepat sebagai keluaran positif atau benar. *False Positive* (FP) adalah data yang diklasifikasi dengan kurang tepat apabila keluaran berupa positif atau benar. *False Negative* (FN) adalah data yang diklasifikasi dengan kurang tepat.

$$Precision = \frac{\sum_i^n \frac{TP_i}{TP_i + FP_i}}{n} \quad \dots \dots \dots \quad (2.24)$$

Persamaan (2.24) merupakan perhitungan rata-rata nilai precision yaitu dari data hasil klasifikasi seberapa banyak data yang benar antara nilai sebenarnya dengan prediksi yang diberikan oleh sistem.

$$Recall = \frac{\sum_i^n \frac{TP_i}{TP_i + FN_i}}{n} \quad \dots \dots \dots \quad (2.25)$$

Persamaan (2.25) merupakan perhitungan rata-rata nilai recall yaitu dari seluruh data benar seberapa banyak data yang keluar dalam hasil klasifikasi. Evaluasi recall digunakan apabila lebih memilih nilai False Positif daripada False Negatif (Ghoenim, 2019). Seperti

pada memprediksi seseorang terkena HIV atau tidak, bahwa memprediksi seseorang yang sehat berlabel HIV lebih dipilih daripada melabelkan sehat kepada seseorang yang terkena HIV.

$$Accuracy = \frac{\sum_i^n \frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i}}{n} \quad \dots \dots \dots \quad (2.26)$$

Ini adalah rasio dari subjek yang diberi label dengan benar ke seluruh kumpulan subjek. Akurasi adalah yang paling intuitif. Accuracy merupakan perhitungan rata-rata nilai akurasi untuk menunjukkan tingkat efektifitas setiap kelas dari sebuah klasifikasi (Sokolova & Lapalme, 2009).

$$FScore = \frac{\sum_i^n \frac{recall_i \times presisi_i}{\beta(recall_i) + (1-\beta)(presisi_i)}}{n} \quad \dots \dots \dots \quad (2.27)$$

Sedangkan persamaan (2.27) merupakan perhitungan rata-rata nilai Fscore yang merupakan nilai kombinasi dari perhitungan recall dan presisi. Hubungan antara label positif data dan yang diberikan oleh pengklasifikasi (Sokolova & Lapalme, 2009).

Akurasi merupakan acuan bagus apabila data keluaran simetris. Seperti contoh berikut apabila TP = 10, FP = 9990, FN = 0, TN = 0. Maka presisinya 0.001, recallnya 1.0, dan akurasinya 0.1. Akurasinya rendah sedangkan semua data sebenarnya telah terprediksi. Jadi penggunaan Fscore dibutuhkan dari pada akurasi pada permasalahan tersebut (Rohim et al., 2019).

2.15 Metode Pengembangan Sistem GRAPPLE

Metode pengembangan sistem GRAPPLE memiliki lima bagian yaitu sebagai berikut (Schmuller, 2019) :

1. Pengumpulan Kebutuhan (*Requirement Gathering*)

Bagian awal dari metode GRAPPLE merupakan bagian untuk memperoleh informasi lengkap tentang sistem yang akan dibangun melalui wawancara atau kuesioner. Pada bagian ini pula didefinisikan fungsi dan kebutuhan sistem yang akan dibangun berdasarkan pada

permasalahan yang ada. Pada penelitian ini, kuesioner digunakan sebagai bentuk pengumpulan data awal berupa aksara Jawa yang ditulis manual oleh responden.

2. Analisis (*Analysis*)

Bagian kedua merupakan penggalian lebih dalam dari hasil yang telah diperoleh dalam tahap sebelumnya. Penggalian yang dilakukan dapat berupa pengembangan data dan informasi serta pembuatan diagram awal dengan tujuan untuk dapat memperoleh solusi dari permasalahan yang ada.

3. Perancangan (*Design*)

Bagian ketiga yaitu design merupakan proses penyesuaian perancangan solusi yang telah dilakukan pada tahap analysis agar dapat diperoleh rancangan yang tepat. Bagian ini mencakup implementasi model dan diagram yang telah dianalisis dan dibuat rancangannya.

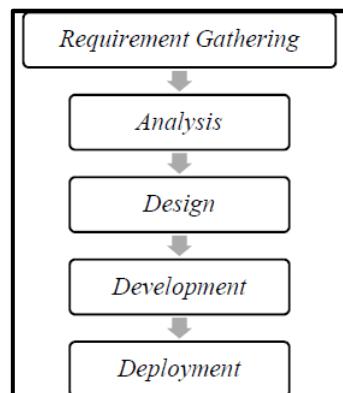
4. Pengembangan (*Development*)

Bagian keempat dari metode GRAPPLE adalah *development* yang bertujuan untuk membangun kode program dan antarmuka pengguna. Pada bagian ini pula dilakukan pengujian program dan dokumentasi sistem.

5. Penyebaran (*Deployment*)

Bagian terakhir merupakan pendistribusian produk yang dihasilkan kepada pengguna. Tahap ini mencakup instalasi dan perencanaan cadangan data jika diperlukan.

Bagan tahapan umum dari metode GRAPPLE ini disajikan pada Gambar



Gambar 2.32 Tahapan Metode Pengembangan Sistem GRAPPLE

BAB III

METODOLOGI PENELITIAN DAN PENGEMBANGAN SISTEM

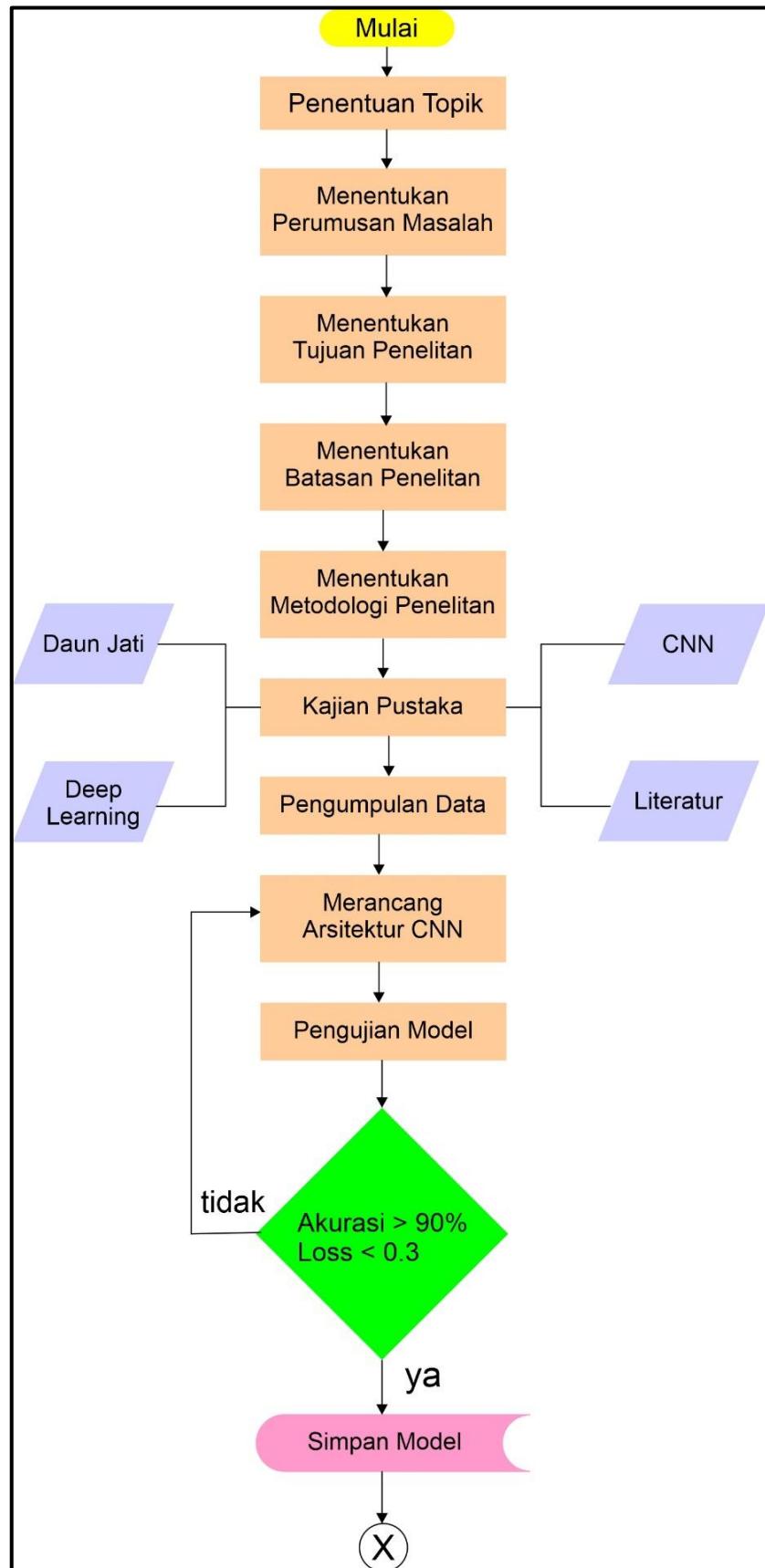
3.1 Metodologi Penelitian

Topik yang dibahas pada bab ini mengenai rancangan dalam melakukan penelitian yang sudah dilakukan dan rancangan mengenai pengembangan kedepannya dari metode yang digunakan pada penelitian ini. Penelitian ini termasuk penelitian kuantitatif dimana penelitian yang sudah dilakukan bersifat sistematis dan menggunakan model-model atau tahapan-tahapan yang bersifat matematis. Teknik pengambilan sampel pada umumnya dilakukan secara acak pada suatu populasi, pengumpulan data menggunakan *instrument* penelitian, analisis data bersifat kuantitatif/static dengan tujuan untuk menguji hipotesis yang ditetapkan (Sugiyono, 2015). Sampel adalah bagian dari jumlah dan karakteristik yang dimiliki oleh populasi tersebut (Sugiyono, 2015).

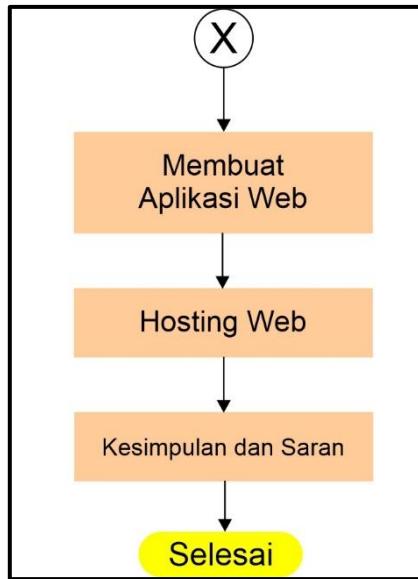
Penelitian ini melakukan pengumpulan data dengan menggunakan studi pustaka dan mengumpulkan data citra yang dibutuhkan. Pada studi pustaka dilakukan agar peneliti dapat menguatkan landasan teori dan mengumpulkan informasi dari refrensi penelitian terdahulu sebagai penunjang penelitian. Informasi yang didapat nantinya akan dijadikan acuan dalam membangun hipotesa selama penelitian.

Sumber data citra yang digunakan pada penelitian ini menggunakan data primer, artinya data-data citra daun varietas jati yang diperoleh bersumber secara langsung dari kebun persemaian jati, lokasi pengambilan sendiri berlokasi di daerah Kabupaten Gunung Kidul Daerah Istimewa Yogyakarta. Data sampel yang digunakan pada penelitian yang sudah dilakukan ini adalah citra daun varietas jati mega, jati PH1(perhutani), dan jati plus yang sudah diberikan pelabelan dan dilakukannya proses pembelajaran model supaya sistem dapat membaca dan mengenali nilai-nilai vektor yang ada pada setiap data citra foto daun varietas jati.

Untuk kerangka kerja penelitian ini dapat digambarkan sebagai berikut pada gambar 3.1



Gambar 3.1 Kerangka Kerja Penelitian



Gambar 3.2 Lanjutan Kerangka Kerja Penelitian

3.1.1 Pengumpulan Data

Citra foto sebagai data yang digunakan dalam penelitian ini berupa data primer. Data primer sendiri diartikan sebagai data-data yang digunakan berasal dari sumbernya secara langsung, bukan berasal dari buku atau dari penelitian sebelumnya. Data primer dalam suatu penelitian diperoleh langsung dari sumbernya dengan melakukan pekukuran, menghitung sendiri dalam bentuk angket, observasi, wawancara dan lain-lain (Ahyar et al., 2020).

Proses mengumpulkan data citra foto daun varietas jati sendiri menggunakan perangkat *smartphone* kemudian mempotret satu persatu daun dari setiap masing-masing varietas jati yang sudah ditentukan. Jumlah yang berhasil didapat dari masing-masing citra foto daun jati pada setiap varietasnya berbeda-beda jumlahnya, dengan rincian 500 data foto citra daun jati varietas plus, 500 data foto citra daun jati varietas mega, dan yang terakhir adalah 511 data foto citra daun jati varietas PH1(perhutani).

Daun jati yang didapatkan berasal dari kebun pangkas jati yang berlokasi di daerah Kabupaten Gunung Kidul, Daerah Istimewa Yogyakarta. Lokasi untuk pengambilan data yang pertama berada di Persemaian Permanen milik Perumahan Hutan(PERHUTANI) Kabupaten Gunung Kidul, yang kedua berlokasi di Hutan Lindung Wanagama milik

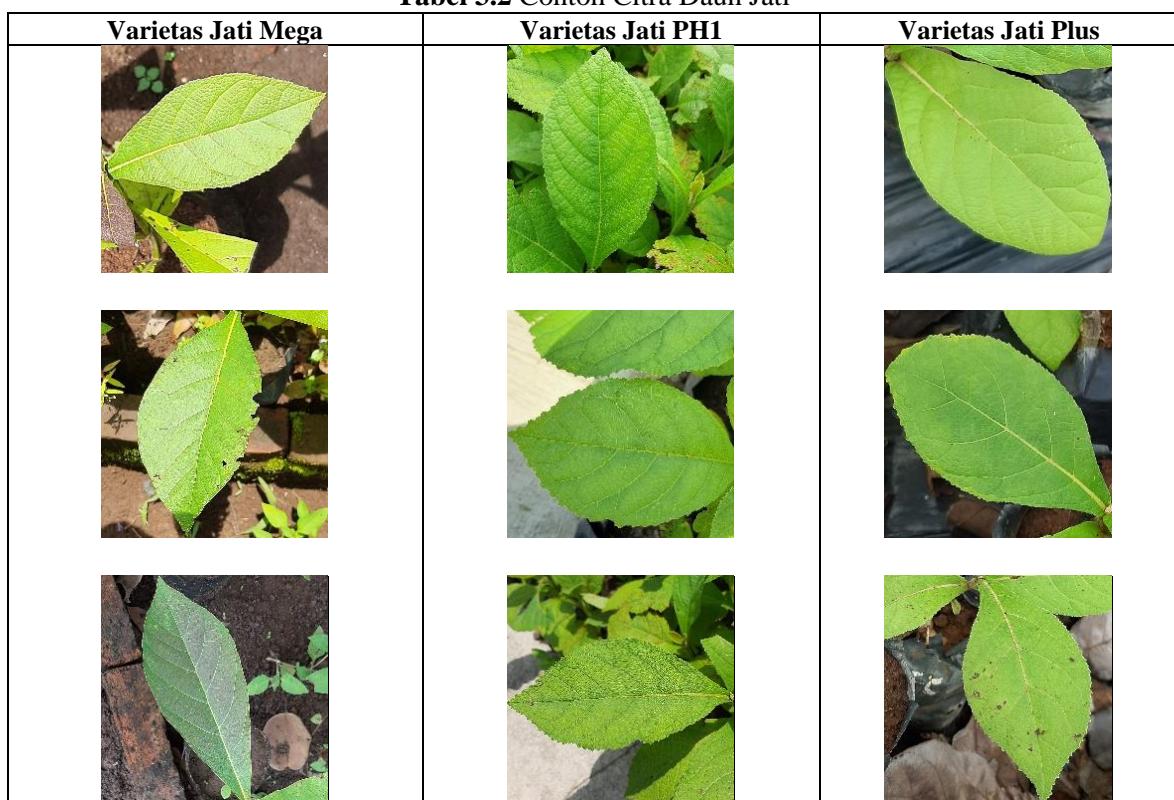
Fakultas Kehutanan Universitas Gadjah Mada, dan yang terakhir berlokasi di persemaian Jati milik P.T. Mitra Baktipersada. Kumpulan data yang berhasil didapatkan nantinya akan menjadi dataset untuk digunakan sebagai data latih(*training*), data validasi(*validation*), dan data uji(*testing*).

Tabel 3.1 Rincian Pembagian Jumlah Data

Data Citra	Data Latih	Data Validasi	Data Uji	Total
Daun varietas Plus	315	135	61	511
Daun Varietas Mega	315	135	50	500
Daun Varietas PH1	315	135	50	500
Tidak Diketahui	315	135	60	510
Total	1260	540	221	2021

Berikut ini merupakan beberapa contoh citra foto dari masing-masing varietas jati yang berhasil didapat:

Tabel 3.2 Contoh Citra Daun Jati



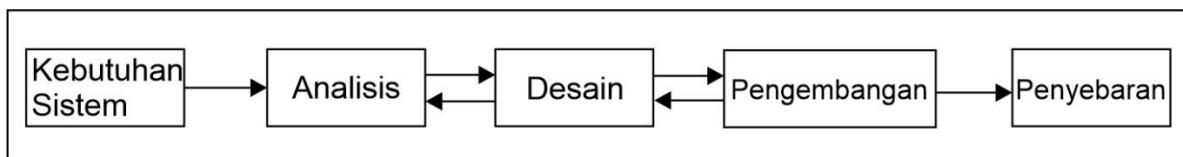
3.1.2 Studi Pustaka

Studi pustaka atau kepustakaan dapat diartikan sebagai serangkaian kegiatan yang berkenaan dengan metode pengumpulan data pustaka, membaca dan mencatat serta mengolah bahan penelitian (Zed, 2004). Informasi dari hasil penelitian yang sudah dilakukan

sebelumnya akan menjadi landasan teori pada penelitian ini. Pada penelitian ini menggunakan refrensi seperti jurnal-jurnal, skripsi, buku dan refrensi-refrensi lainnya yang berkaitan dengan penelitian ini membahas seputar identifikasi, tanaman jati, dan metode *convolutional neural network*.

Adapun penelitian yang berkaitan dengan penelitian ini diantaranya yang pertama dengan judul “Convolutional Neural Network untuk Pendekripsi Patah Tulang Femur pada Citra Ultrasonik B–Mode” oleh (Rokhana et al., 2019). Kemudian penelitian dengan judul “Klasifikasi Citra Multi-Kelas Menggunakan Convolutional Neural Network” oleh (Kamal Hasan et al., 2019). Yang ketiga dengan judul “Implementasi Metode Convolutional Neural Network Untuk Klasifikasi Tanaman Pada Citra Resolusi Tinggi” (Nour, 2018), kemudian penelitian dengan judul “Identifikasi Daun Tanaman Jati Menggunakan K-Nearest Neighbour Dengan Ekstraksi Fitur Ciri Morfologi Daun” oleh (Pratama B.M., 2013), dan lainnya.

3.2 Metodologi Pengembangan Sistem



Gambar 3.3 Alur Pengembangan Sistem GRAPPLE

Penelitian ini menggunakan metodologi pengembangan sistem yaitu *Guidelines for Rapid Application Engineering*(GRAPPLE). GRAPPLE Terdiri atas tiga bagian yaitu yang pertama pengumpulan kebutuhan, yang kedua analisis, dan yang terakhir adalah perancangan. Pengumpulan kebutuhan membahas mengenai bahan-bahan yang digunakan baik perangkat keras maupun perangkat lunak selama proses penelitian berlangsung. Bagian analisis menggambarkan bahasan proses yang dilakukan oleh sistem dari awal hingga berakhirnya proses. Dan bagian perancangan membahas proses simulasi dari apa yang ingin

dibuat sebelum kita merealisasikan produk jadinya, berkali-kali sehingga memungkinkan kita merasa puas dengan hasil akhirnya.

3.2.1 Kebutuhan Sistem

Sub bab kebutuhan sistem membahas mengenai perangkat apa saja yang digunakan selama penelitian maupun untuk pengembangan selanjutnya mulai dari perangkat keras dan juga pada perangkat lunak. Berikut ini merupakan rincian kebutuhan perangkat keras dan perangkat lunak yang digunakan:

1. Perangkat keras

Tabel 3.3 Kebutuhan Perangkat Keras

No.	Perangkat Keras	Keterangan
1	<i>Notebook</i>	Fujitsu E-754
	<i>Processor</i>	Intel Core i5-4210M CPU 2.6 GHz
	<i>Solid State Drive</i>	512 GB
	<i>Hard Disk Drive</i>	512 GB
	<i>Video Graphic Array</i>	Intel Graphic HD 4600
2	<i>Smartphone</i>	Samsung A30S
	<i>Processor</i>	Octa-core (2x1.8 GHz Cortex-A73 & 6x1.6 GHz Cortex-A53)
	<i>GPU</i>	Mali-G71 MP2
	<i>Camera</i>	25 MP, f/1.7, 27mm (wide), PDAF
3	<i>Tripod</i>	
4	<i>Mouse</i>	

2. Perangkat lunak

Tabel 3.4 Kebutuhan Perangkat Lunak

No.	Perangkat Lunak	Keterangan
1	<i>Sistem Operasi Notebook</i>	Windows 10 Profesional
	<i>Sistem Operasi Smartphone</i>	Android 10 One UI 2.0
	<i>Bahasa Pemrograman</i>	Python 3.8
	<i>Integrated Development Environment (IDE) untuk Python</i>	Spyder dan Visual Studio Code
	<i>Video Graphic Array</i>	Intel Graphic HD 4600
2	<i>Cloud Notebok</i>	Google Colaboration
	<i>Cloud Notebok</i>	Kaggle
	<i>Cloud Drive</i>	Google Drive
	<i>Cloud Hosting</i>	Heroku
3	<i>end-to-end open source machine learning platform</i>	Tensorflow
	<i>Deep Learning API</i>	Keras
	<i>Import Image</i>	PIL
	<i>Numerical computing tools</i>	Numpy
	<i>open source data analysis and manipulation tool</i>	Pandas
	<i>framework for Machine Learning and Data Science teams</i>	Streamlit

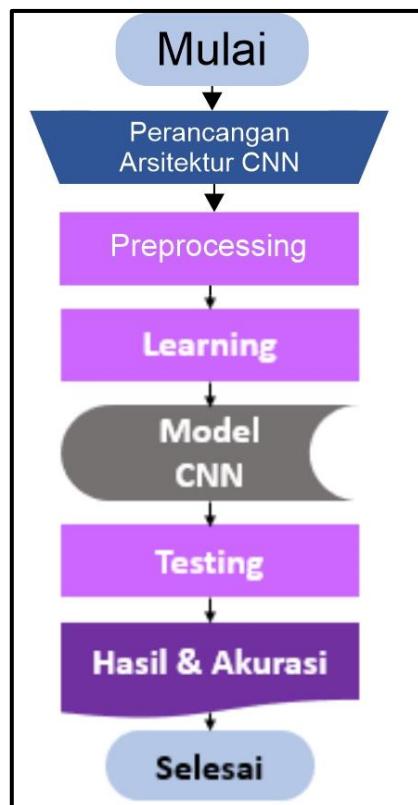
Tabel 3.5 Lanjutan Kebutuhan Perangkat Lunak

No.	Perangkat Lunak	Keterangan
	<i>comprehensive library for creating static, animated, and interactive visualizations</i>	matplotlib
	<i>data visualization library based on matplotlib</i>	seaborn
	<i>mapping is captured the first time the os module is imported</i>	os
	<i>module may not handle dates and times before the epoch or far in the future</i>	time
	<i>module also automatically generates help and usage</i>	argparse

3.2.2 Analisis

Menurut Dwi Prastowo Darminto, Analisis sebagai suatu aksi memerinci suatu subjek/objek tertentu menjadi beberapa bagian. Setiap bagian tersebut diamati, lalu dicari hubungannya dengan bagian lain yang terhubung, dengan tujuan mendapatkan definisi yang tepat dan agar mudah dipahami secara keseluruhan.

3.2.2.1 Analisis pembuatan model CNN

**Gambar 3.4** Tahapan Percobaan Membuat Model CNN

Proses pembuatan sistem klasifikasi CNN dimulai dari perancangan arsitektur CNN. Perancangan model CNN dilakukan menggunakan platform google colaboratory. Desain dari

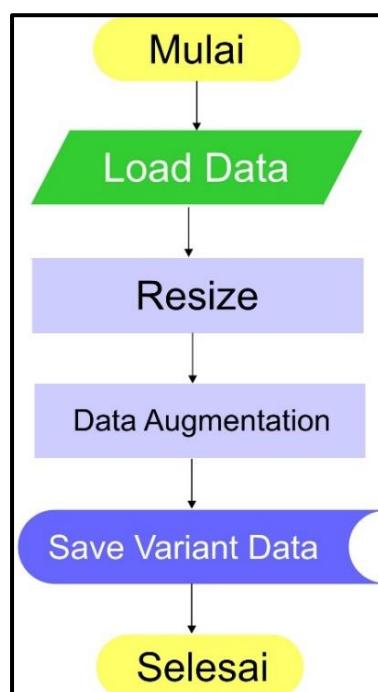
arsitektur CNN yang dibuat pada penelitian ini ditunjukkan dengan rincian pada tabel sub bab 3.2.2.2 analisis arsitektur model CNN.

Setelah arsitektur CNN selesai dirancang atau ditentukan, maka tahap berikutnya yaitu masuk dalam proses pembelajaran(*learning*). Proses pembelajaran(*learning*) dilakukan supaya model CNN nantinya dapat mengenali objek citra daun varietas jati berdasarkan indeks dari tiap kelas yang diinputkan. Proses pembelajaran(*learning*) dilakukan hingga ditemukan model CNN yang memenuhi target yang telah ditentukan.

Model CNN yang memenuhi target tersebut nantinya akan disimpan dan akan digunakan pada tahap proses pengujian(*testing*). Proses pengujian(*testing*) dilakukan untuk menguji performa dari model CNN yang telah dirancang untuk mengklasifikasi citra daun varietas jati. Setelah dilakukan pengujian(*testing*) model CNN, tahap selanjutnya adalah melakukan penghitungan akurasi untuk mengukur ketepatan model CNN dalam mengklasifikasi citra daun varietas jati.

Berikut ini rincian dari beberapa proses yang ada:

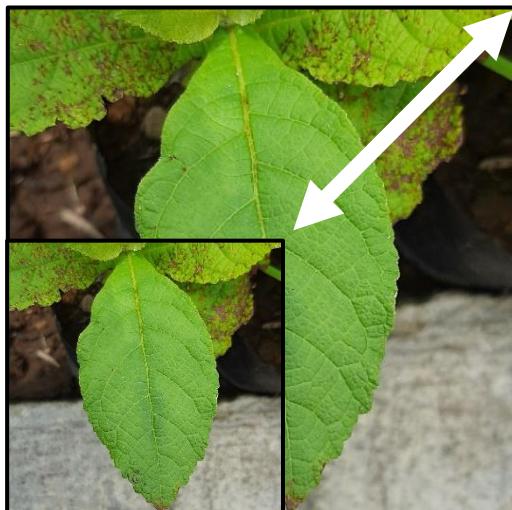
1. *Preprocessing*



Gambar 3.5 Flowchart Dalam *Preprocessing*

Perlu adanya persiapan data-data sebelum melakukan proses pembelajaran, pada tahap ini bertujuan memberi ukuran citra foto yang sama besarnya dan memvariasi dataset yang ada. Ukuran citra foto daun jati yang dijadikan data masukan dan data uji harus berukuran sama besar maka dari itu pada penelitian menggunakan patokan ukuran *pixel* yaitu $256 \text{ pixel} \times 256 \text{ pixel}$, setelah itu supaya model yang dibuat kaya akan ilmu atau pengetahuan mengenai data citra foto daun jati maka data-data yang ada perlu divariasi macamnya supaya model bisa mengenali lebih kaya lagi akan informasi. Memvariasi data citra foto daun jati pada penelitian ini menggunakan *data augmentation* dari Keras.

a. *Resize*



Gambar 3.6 Ilustrasi merubah ukuran(*resize*)

Data citra foto yang masuk harus memiliki ukuran yang sama besar supaya tidak ada perbedaan ukuran antara data citra foto yang satu dengan data citra foto yang lainnya. Penelitian yang dilakukan ini menggunakan ukuran yang sama besar yaitu dengan ukuran $256 \times 256 \text{ pixel}$.

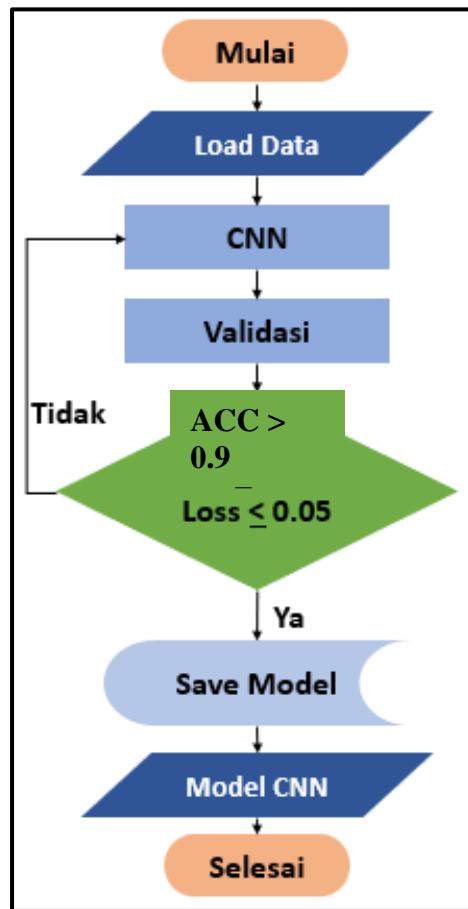
b. *Data Augmentation*



Gambar 3.7 Ilustrasi Perbanyak Dataset

Menggunakan bantuan *library* dari keras dapat memperbanyak varian data yang dimiliki, diantaranya memutar citra foto, mencerminkan secara horizontal, dan menambah sudut

2. Pembelajaran (*Learning*)

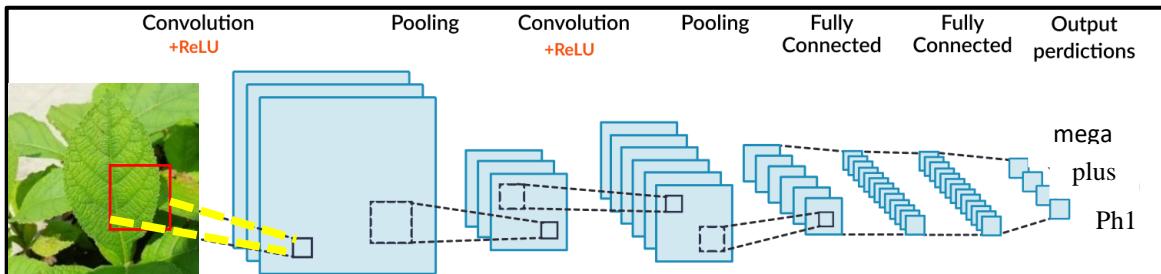


Gambar 3.8 Flowchart Pembelajaran Model

Setelah data kaya akan informasi maka pada tahap ini proses yang dilakukan adalah mempelajari atau mengenali pola dari data citra foto secara satu persatu sebanyak dataset yang dimiliki kemudian dipelajari dengan menggunakan metode *CNN* lalu perlu dilakukannya proses validasi, jika akurasi yang dihasilkan sudah lebih dari 0.9 atau 90% dan *loss* kurang dari 0.25 maka akan lanjut tahap selanjutnya yaitu menyimpan model kemudian jadi model *CNN* yang sudah mempelajari pola citra foto daun jati, apabila kurang dari 0.9 atau 90% dan *loss* lebih dari 0.25 maka akan disusun ulang arsitektur *CNN* sampai nilai validasi dari *error* dan akurasi memenuhi target yang telah ditentukan. Dan jika sudah

memenuhi target maka akan menjadi model paten yang akan digunakan pada tahap pengujian(*testing*).

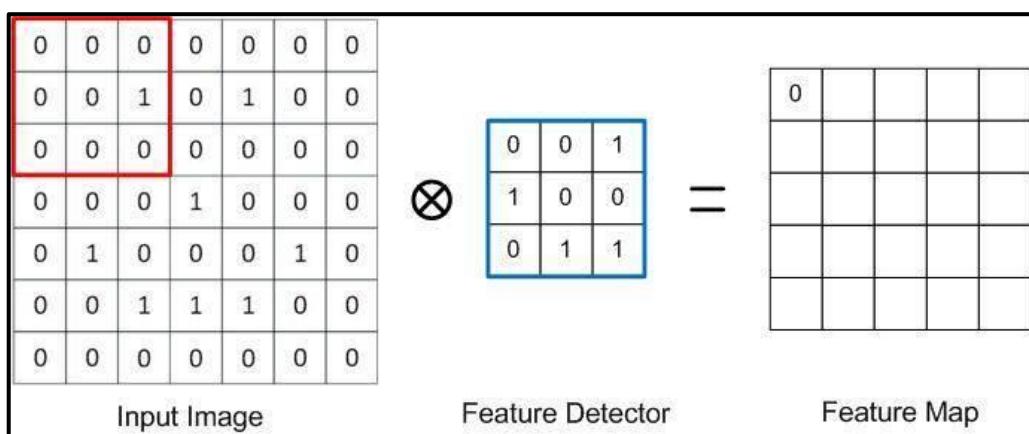
a. CNN



Gambar 3.9 Ilustrasi Arsitektur CNN

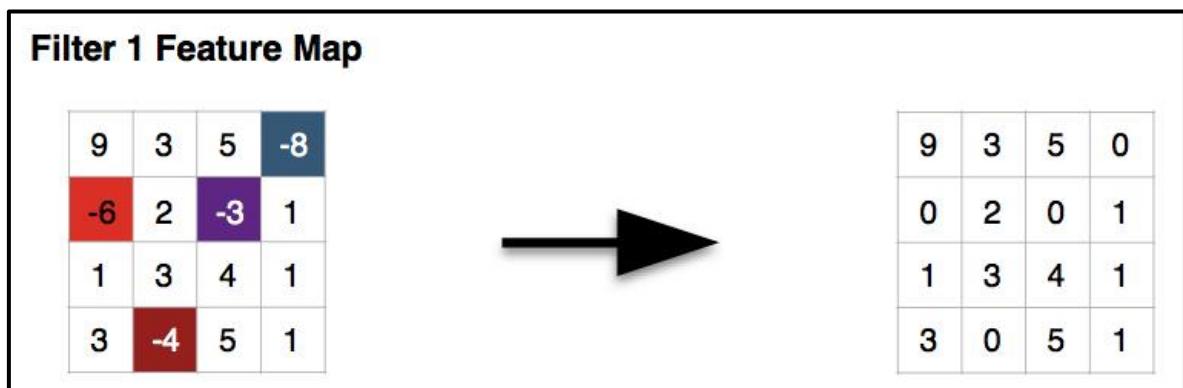
Metode *Convolutional Neural Network* merupakan salah satu algoritma yang ada pada *deep learning*. Ciri khas pada metode ini yaitu penambahan *layer* konvolusi untuk meningkatkan pengambilan fitur di *multi layer preceptron*. Beberapa lapisan yang dapat diatur sesuai kebutuhan menambah keunggulan pada *CNN*. Lapisan-lapisan yang ada pada *CNN* diantaranya ada lapisan konvolusi, lapisan pooling, dan lapisan *fully connected*. Berikut ini merupakan penjelasan dari lapisan-lapisan yang terdapat pada *CNN* dan beberapa fungsi aktivasi yang dipakai.

Lapisan-lapisan yang terdapat dalam *feature learning* berguna untuk mentranslasikan suatu input menjadi features berdasarkan ciri dari input tersebut yang berbentuk angka-angka dalam vektor. Lapisan ekstraksi fitur ini terdiri dari *convolutional layer* dan *pooling layer*. Berikut ini merupakan penjelasan lebih jelasnya mengenai *feature learning*.



Gambar 3.10 Ilustrasi Perkalian Pada Lapisan Konvolusi

Lapisan konvolusi bertugas untuk mengkalikan nilai-nilai yang ada pada data masukan(*input image*) dengan nilai konvolusi yang sudah ditentukan, bertujuan untuk menyederhanakan nilai-nilai yang masuk tanpa harus menghilangkan maksud dari foto tersebut untuk menjadi nilai baru yang disebut dengan *feature map*. Data masukan citra foto akan menjadi beberapa lapisan baru lagi sebanyak nilai yang diinginkan, pada penelitian ini menggunakan konvolusi 32, 64, 128, dan 256. Dengan kata lain lapisan konvolusi menghitung output dari neuron yang terhubung ke daerah lokal dalam input, masing-masing menghitung produk titik antara bobot mereka dan wilayah kecil yang terhubung ke dalam *volume input*.

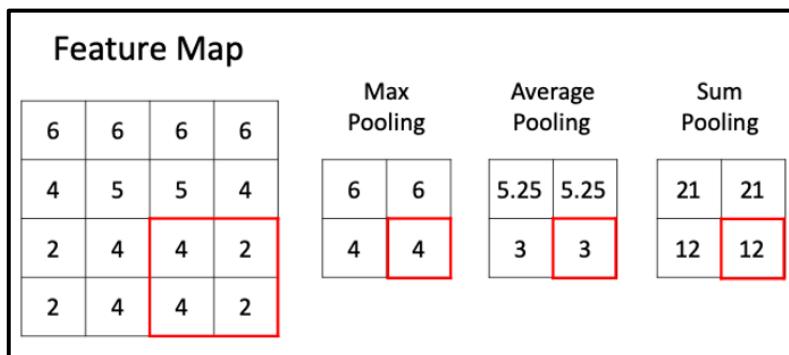


Gambar 3.11 Ilustrasi Fungsi Aktivasi RELU

Rectified Linear Unit (ReLU) akan menghilangkan vanishing gradient dengan cara menerapkan fungsi aktivasi element sebagai $f(x)=\max[0,x]$ alias aktivasi elemen akan dilakukan saat berada di ambang batas 0. Kelebihan dan kekurangan dalam penggunaan ReLU :

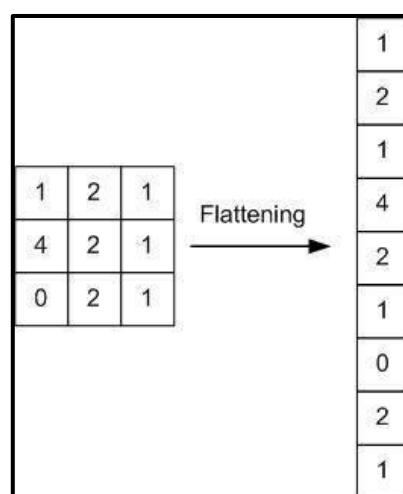
- (+) : Bisa mempercepat gradien stokastik dibandingkan dengan fungsi sigmoid / tan h karena ReLU berbentuk linear
- (+) : Tidak menggunakan operasi eksponensial seperti sigmoid/tan h, sehingga bisa melakukan dengan pembuatan matriks aktivasi saat ambang batas berada pada nilai 0.
- (-) : ReLU bisa rapuh saat masa training dan mati karena gradien besar yang mengalir melalui ReLU menyebabkan update bobot, sehingga neuron tidak aktif pada datapoint lagi.

Jika ini terjadi, maka gradien yang mengalir melalui unit akan selamanya nol dari titik itu. Artinya, unit ReLU dapat mati secara ireversibel selama pelatihan karena mereka dapat melumpuhkan data manifold. Misalnya, Anda mungkin menemukan bahwa sebanyak 40% dari jaringan Anda dapat “mati” (yaitu neuron yang tidak pernah aktif di seluruh dataset pelatihan) jika tingkat pembelajaran ditetapkan terlalu tinggi. Dengan pengaturan tingkat pembelajaran yang tepat, ini lebih jarang menjadi masalah.



Gambar 3.12 Ilustrasi Pada Lapisan Pooling

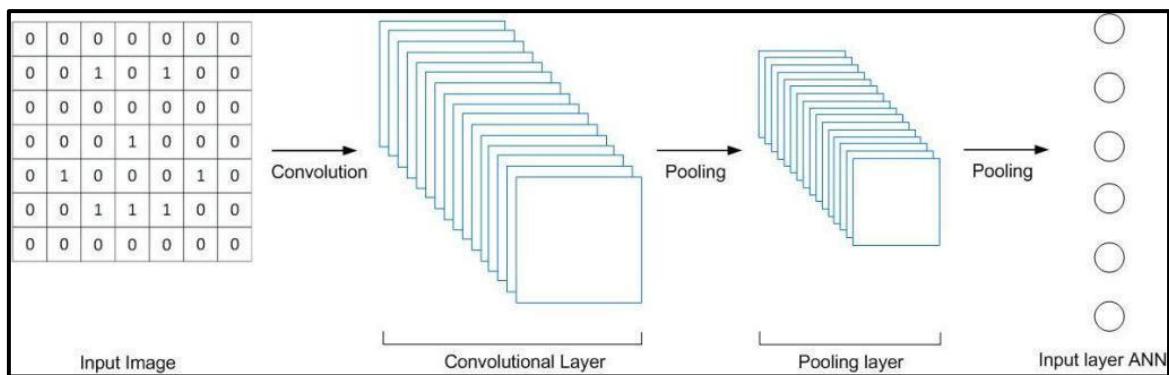
Pooling layer adalah lapisan yang mengurangi dimensi dari feature map atau lebih dikenal dengan langkah untuk *downsampling*, sehingga mempercepat komputasi karena parameter yang harus diupdate semakin sedikit dan mengatasi *overfitting*. Pooling yang biasa digunakan adalah *max pooling* dan *average pooling*. *Max pooling* untuk menentukan nilai maksimum tiap pergeseran *filter*, sementara *average pooling* akan menentukan nilai rata-ratanya, dan untuk *sum pooling* akan menentukan nilai total pada tiap pergeseran *filter*.



Gambar 3.13 Ilustrasi Penyederhanaan Dimensi Array

Ini adalah tahapan yang sangat sederhana. Jika sebelumnya kita sudah melakukan proses konvolusi, dilanjutkan dengan pooling, maka kali ini kita lakukan proses *flattening*. Tahapan *flattening* adalah merubah dari matriks yang ada di *pooling layer* menjadi satu kolom saja (sebuah vektor tunggal). Nantinya vektor ini akan menjadi bagian dari *input layer* di *artificial neural networks* (ANN). Dari *pooled feature map* (di sebelah kiri) dari proses sebelumnya, kita cukup mengambil baris demi baris dan menggabungkannya menjadi 1 kolom. Baris pertama (1, 2, 1) digabung dengan baris kedua (4, 2, 1) kemudian baris ketiga (0, 2, 1).

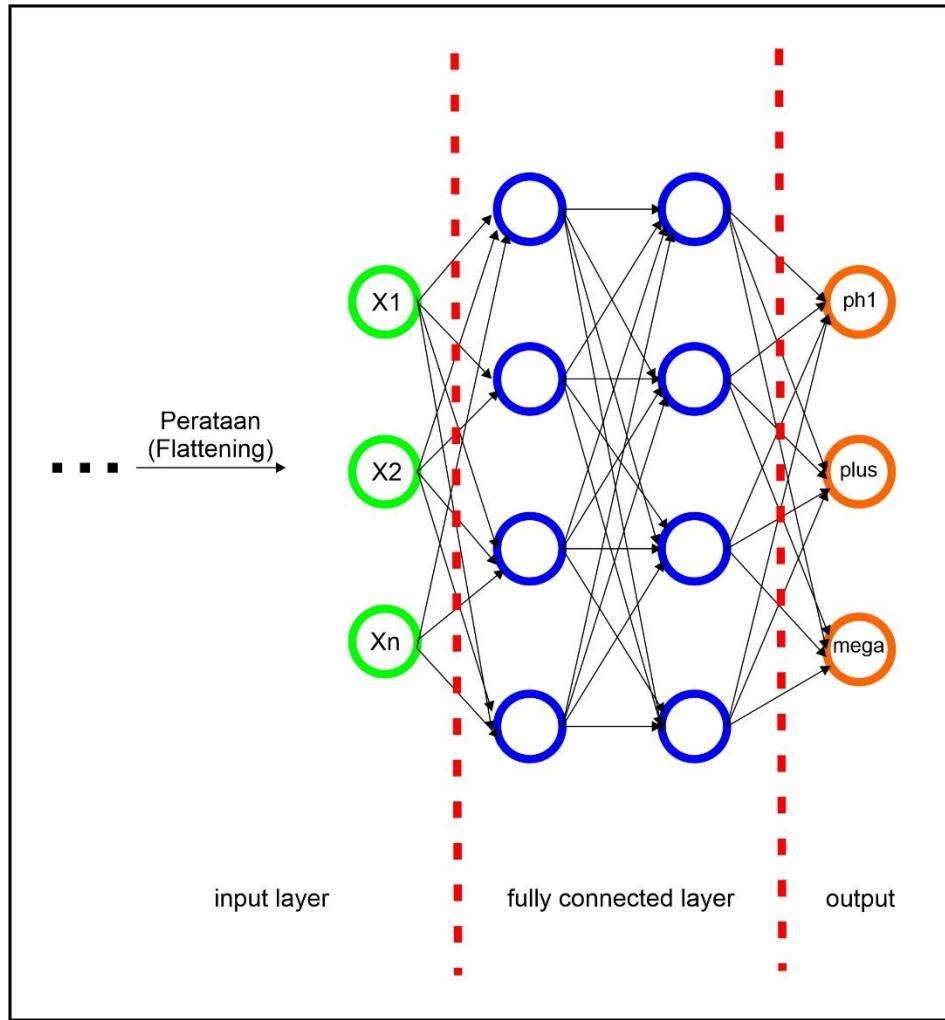
Setelah kita lakukan *flattening* (bahasa indonesia= pemipihan/perataan), kita masukkan hasilnya ke dalam *input layer* sebuah ANN. Tampilannya sebagai berikut:



Gambar 3.14 Ilustrasi Tahapan Dari Input Hingga Flatten

Semua *pooling layer* akan menjadi 1 vektor saja. Jadi jika sebuah *pooling layer* berukuran 3×3 matriks, maka ia akan menjadi 1 vektor dengan 9 baris. Jika ada 10 *pooling layer* dengan ukuran yang sama (3×3), maka akan ada 1 vektor dengan 90 baris sebagai input untuk ANN, dan seterusnya sampai semua *feature map* yang ada pada *pooling layer*.

Lapisan selanjutnya adalah *fully connected layer*, hal yang perlu diperhatikan adalah dalam pembahasan ANN sebelumnya setiap *neuron (nodes)* tidaklah harus saling terhubung dengan nodes di depannya. Dalam konteks CNN, maka semua nodes harus terhubung dengan nodes di depan dan belakangnya. Oleh karena itu disebut dengan *full connection*.



Gambar 3.15 Ilustrasi Fully Connected Layer

Pada ilustrasi di atas kita bisa lihat bahwa hasil dari flattening menjadi input dari ANN.

Seperti biasa, ANN terdiri dari 3 bagian, yaitu *input layer*, *hidden layer*, dan *output layer*.

Dalam konteks CNN, maka semua nodes saling terhubung dengan nodes di depan dan belakang.

Kemudian di bagian output layer ada terdiri dari beberapa class (kategori). Di gambar atas, kita memiliki 3 kelas, misalnya apakah gambar masuk ke dalam kategori varietas $ph1$, $plus$, atau $mega$. Jika lebih dari 3 kategori, misal 5 kategori, maka nodes di output layer ada 5 buah.

Hasil dari 3 kategori pada gambar di atas (apakah gambarnya adalah varietas $ph1$ atau $plus$ atau $mega$) berupa probabilitas. Ia akan mengeluarkan nilai probabilitas untuk masing-

masing kategori. Jika ia menebak bahwa gambarnya adalah gambar daun jati varietas ph1, bisa jadi nilai probabilitas akhirnya adalah plus 0.1, ph1 0.87, dan mega adalah 0.1.

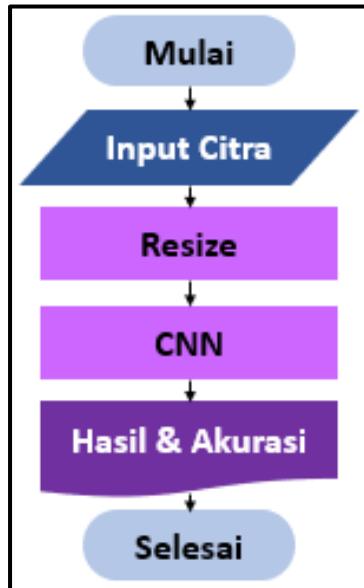
Cara kerja ANN di CNN juga sama. Ia akan maju (*forward propagation*) dan mundur (*back propagation*). Yang paling berperan dalam penentuan apakah input gambar masuk ke kategori A, B, dan seterusnya adalah bobot yang dimiliki setiap node. Bobot ini akan selalu disesuaikan setiap epochnya.

Logit value	Softmax computation	Softmax outcome
2.0	$\frac{\exp(x_i)}{\sum_j \exp(x_j)} = \frac{\exp(2.0)}{\exp(2.0) + \exp(4.3) + \exp(1.2) + \exp(-3.1)}$	0.087492
4.3	$\frac{\exp(x_i)}{\sum_j \exp(x_j)} = \frac{\exp(4.3)}{\exp(2.0) + \exp(4.3) + \exp(1.2) + \exp(-3.1)}$	0.872661
1.2	$\frac{\exp(x_i)}{\sum_j \exp(x_j)} = \frac{\exp(1.2)}{\exp(2.0) + \exp(4.3) + \exp(1.2) + \exp(-3.1)}$	0.039312
-3.1	$\frac{\exp(x_i)}{\sum_j \exp(x_j)} = \frac{\exp(-3.1)}{\exp(2.0) + \exp(4.3) + \exp(1.2) + \exp(-3.1)}$	0.000292
Sum		0.999757
(rounded)		1

Gambar 3.16 Contoh Perhitungan Fungsi Aktivasi *Softmax*

Fungsi *Softmax* yang terdapat pada lapisan *fully connected layer* menghitung probabilitas dari setiap kelas target atas semua kelas target yang memungkinkan dan akan membantu untuk menentukan kelas target untuk input yang diberikan. Keuntungan utama menggunakan *Softmax* adalah rentang probabilitas output dengan nilai 0 hingga 1, dan jumlah semua probabilitas akan sama dengan satu. Jika fungsi softmax digunakan untuk model multi-klasifikasi, dia akan mengembalikan peluang dari masing-masing kelas dan kelas target akan memiliki probabilitas tinggi. Softmax menggunakan eksponensial (*e-power*) dari nilai input yang diberikan dan jumlah nilai eksponensial dari semua nilai dalam input. Maka rasio eksponensial dari nilai input dan jumlah nilai eksponensial adalah output dari fungsi *softmax*.

3. Pengujian(*Testing*)



Gambar 3.17 Flowchart Pengujian Model

Tahap ini merupakan akhir dari proses yang ada, menguji model yang sudah mempelajari pola citra foto data daun dari masing-masing varietas jati yang ditentukan. Dimulai dari memasukan data uji citra foto daun jati kemudian ukuran foto disama ratakan yaitu ukuran *256 pixel x 256 pixel* sesuai ukuran citra foto yang digunakan data masukan ketika proses pembelajaran model.

Setelah ukuran disesuaikan selanjutnya membaca pola daun data uji lalu dicek dengan model CNN yang telah ada untuk dikenali citra foto daun data uji tadi untuk mencari tahu termasuk kelas varietas jati apa citra foto data daun uji tadi, dan akan juga diketahui tingkat kemiripan dari masing-masing kelas.

a. Analisis Pembangunan Model CNN

1. Arsitektur CNN

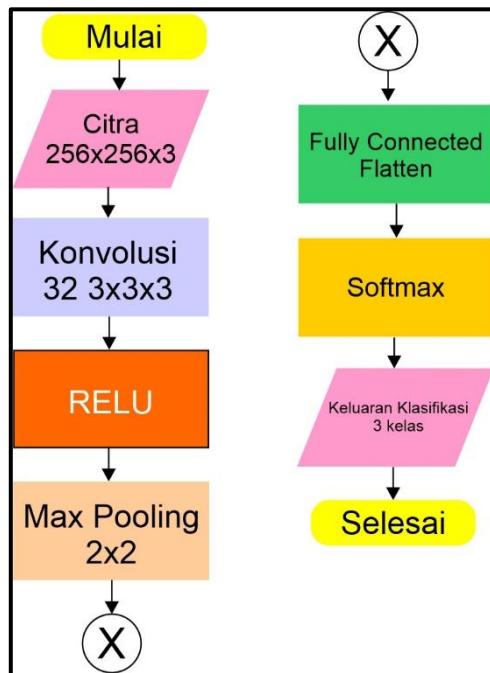
Tabel 3.6 Arsitektur CNN yang Diuji

CNN 1	CNN 2	CNN 3	CNN 4	CNN 5	CNN 6	CNN 7
Conv 32						
		Pool	Pool	Pool	Pool	Pool
	Pool	Conv 64				

Tabel 3.7 Lanjutan Arsitektur CNN yang Diuji

CNN 1	CNN 2	CNN 3	CNN 4	CNN 5	CNN 6	CNN 7
Pool	Conv 64	Pool	Pool	Pool	Pool	Pool
			Conv 128	Conv 128	Conv 128	Conv 128
	Conv 128	Pool	Pool	Pool	Pool	Pool
		Conv 256				
FC	Pool	Pool	Pool	Pool	Pool	Pool
				Conv 512	Conv 512	Conv 512
	FC	FC	FC	Pool	Pool	Pool
				FC	FC	FC
Softmax	Softmax	Softmax	Softmax	FC 32	FC 1024	
					Softmax	Softmax

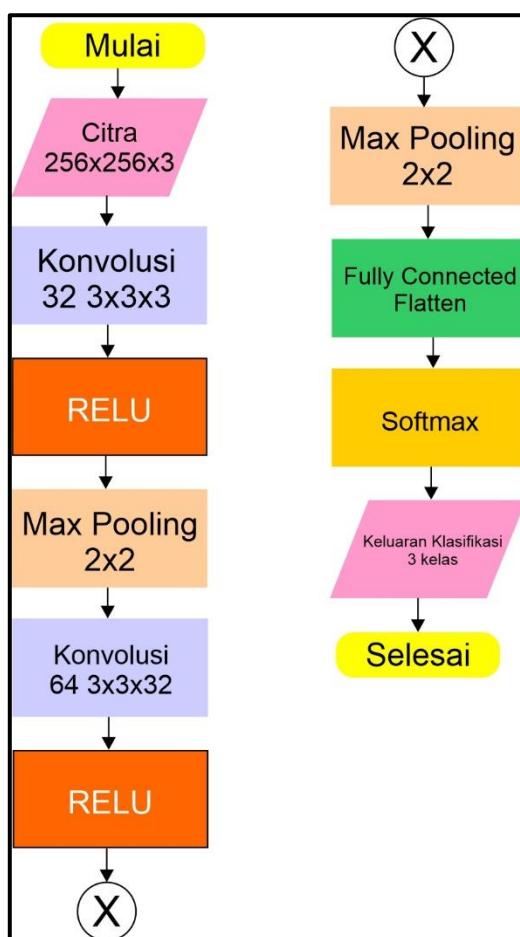
i. Arsitektur CNN 1

**Gambar 3.18 Flowchart Arsitektur CNN 1**

Jika disusun, arsitektur CNN pertama masukan citra akan memiliki ukuran yang tidak sama maka pada tahap ini citra akan *direshape* menjadi $256 \times 256 \text{ pixel}$ yang terdiri dari 3 bagian yaitu *red*, *green*, *blue* kemudian masuk dalam proses konvolusi berukuran kernel 3×3 dengan 3 bagian yaitu *red*, *green*, *blue* sebanyak 32 menggunakan pergeseran *stride* 1 dan pelebaran *padding* 0. Kemudian masuk dalam proses *thresholding* dengan nilai nol terhadap

nilai piksel pada input citra menggunakan fungsi aktivasi *relu*. Kemudian masuk dalam proses *max pooling* berukuran kernel 2x2 untuk diambil nilai maksimal dari setiap kernel pergeserannya. Setelah mendapatkan *featured map* baru kemudian masuk dalam proses *fully connected* untuk disederhanakan menjadi satu layer dengan cara *mereshape* hasil *featured map* yang didapat dari *max pooling*. Dan tahap terakhir adalah penggunaan fungsi aktivasi *softmax* untuk menangani *multiclass classification* yang berisikan 3 keluaran *neuron*.

ii. Arsitektur CNN 2



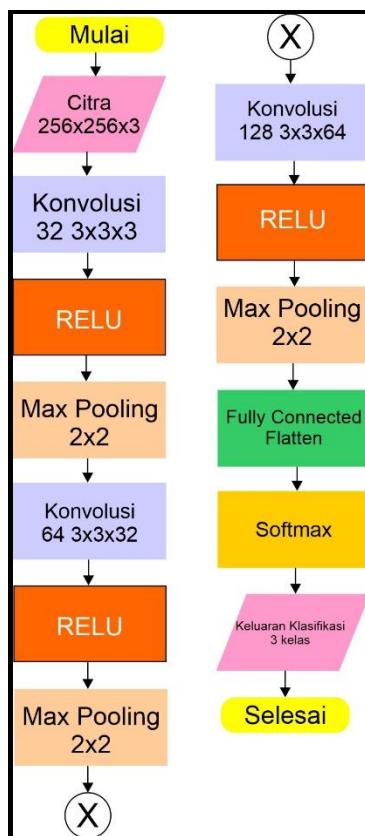
Gambar 3.19 Flowchart Arsitektur CNN 2

Jika disusun, arsitektur CNN kedua sebagai pengembangan dari arsitektur CNN 1, masukan citra akan memiliki ukuran yang tidak sama maka pada tahap ini citra akan *direshape* menjadi 256x256 *pixel* yang terdiri dari 3 bagian yaitu *red, green, blue* kemudian masuk dalam proses konvolusi berukuran kernel 3x3 dengan 3 bagian yaitu *red, green, blue*

sebanyak 32 menggunakan pergeseran *stride* 1 dan pelebaran *padding* 0. Kemudian masuk dalam proses *thresholding* dengan nilai nol terhadap nilai piksel pada input citra menggunakan fungsi aktivasi *relu*. Kemudian masuk dalam proses *max pooling* berukuran kernel 2x2 untuk diambil nilai maksimal dari setiap kernel pergeserannya. Berikutnya akan masuk dalam proses konvolusi dengan ukuran kernel 3x3 sebanyak 32 yang akan menjadi data masukan konvolusi 64.

Kemudian akan diproses menggunakan fungsi aktivasi RELU untuk melakukan proses *thresholding* setelah selesai akan diambil nilai maksimalnya kembali menggunakan *max pooling*. Setelah mendapatkan *featured map* baru kemudian masuk dalam proses *fully connected* untuk disederhanakan menjadi satu layer dengan cara *reshape* hasil *featured map* yang didapat dari *max pooling*. Dan tahap terakhir adalah penggunaan fungsi aktivasi *softmax* untuk menangani *multiclass classification* yang berisikan 3 keluaran *neuron*.

iii. Arsitektur CNN 3



Gambar 3.20 Flowchart Arsitektur CNN 3

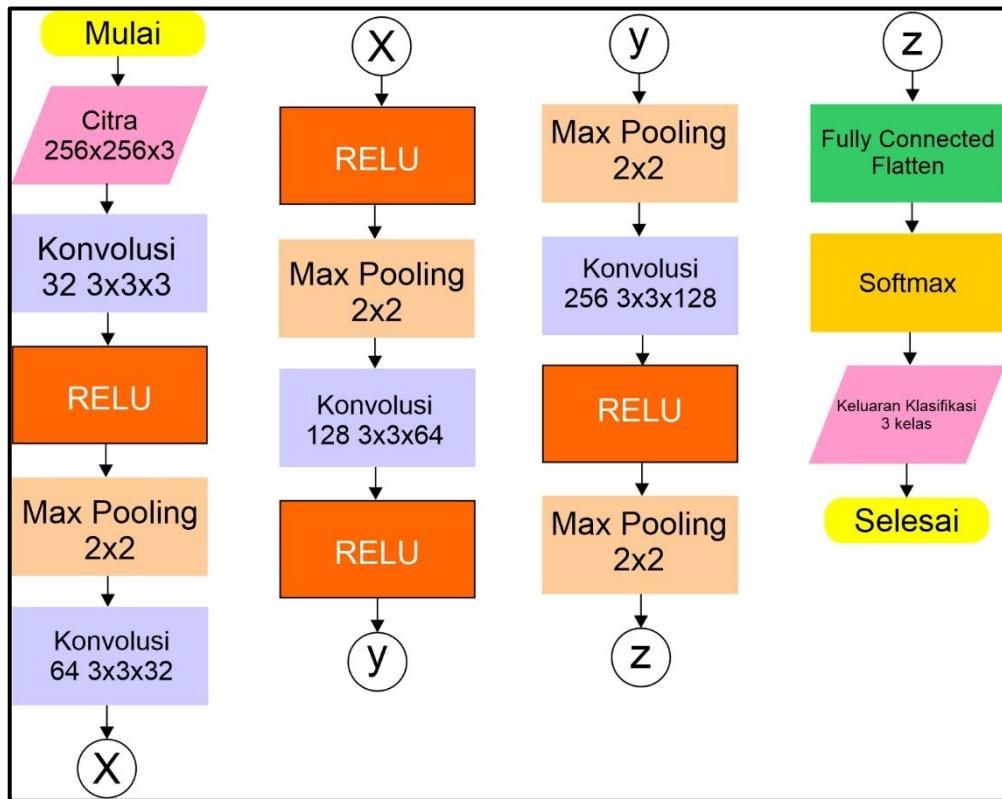
Jika disusun, arsitektur CNN ketiga ini sebagai pengembangan dari arsitektur CNN kedua, masukan citra akan memiliki ukuran yang tidak sama maka pada tahap ini citra akan *direshape* menjadi 256x256 *pixel* yang terdiri dari 3 bagian yaitu *red, green, blue* kemudian masuk dalam proses konvolusi berukuran kernel 3x3 dengan 3 bagian yaitu *red, green, blue* sebanyak 32 menggunakan pergeseran *stride* 1 dan pelebaran *padding* 0. Kemudian masuk dalam proses *thresholding* dengan nilai nol terhadap nilai piksel pada input citra menggunakan fungsi aktivasi *relu*. Kemudian masuk dalam proses *max pooling* berukuran kernel 2x2 untuk diambil nilai maksimal dari setiap kernel pergeserannya.

Berikutnya akan masuk dalam proses konvolusi dengan ukuran kernel 3x3 sebanyak 32 yang akan menjadi data masukan konvolusi dan akan dilakukan perulangan sebanyak 64. Kemudian akan diproses menggunakan fungsi aktivasi RELU untuk melakukan proses *thresholding* setelah selesai akan diambil nilai maksimalnya kembali menggunakan *max pooling*.

Bagian ini yang akan menjadi pengembangan dari arsitektur CNN kedua, dimana akan ditambah 1 lapisan konvolusi, RELU, dan *pooling* untuk semakin mendalam ketika proses pengambilan fitur-fiturnya. Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 64 lapisan akan dikonvolusi kembali sebanyak 128 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Setelah mendapatkan *featured map* baru dari *pooling layer* kemudian masuk dalam proses *fully connected* untuk disederhanakan menjadi satu layer dengan cara *mereshape* hasil *featured map* yang didapat dari *max pooling*. Dan tahap terakhir adalah penggunaan fungsi aktivasi *softmax* untuk menangani *multiclass classification* yang berisikan 3 keluaran *neuron*.

iv. Arsitektur CNN 4



Gambar 3.21 Flowchart Arsitektur CNN 4

Jika disusun, arsitektur CNN keempat ini sebagai pengembangan dari arsitektur CNN ketiga, masukan citra akan memiliki ukuran yang tidak sama maka pada tahap ini citra akan *direshape* menjadi 256x256 pixel yang terdiri dari 3 bagian yaitu red, green, blue kemudian masuk dalam proses konvolusi berukuran kernel 3x3 dengan 3 bagian yaitu red, green, blue sebanyak 32 menggunakan pergeseran *stride* 1 dan pelebaran *padding* 0. Kemudian masuk dalam proses *thresholding* dengan nilai nol terhadap nilai piksel pada input citra menggunakan fungsi aktivasi *relu*. Kemudian masuk dalam proses *max pooling* berukuran kernel 2x2 untuk diambil nilai maksimal dari setiap kernel pergeserannya.

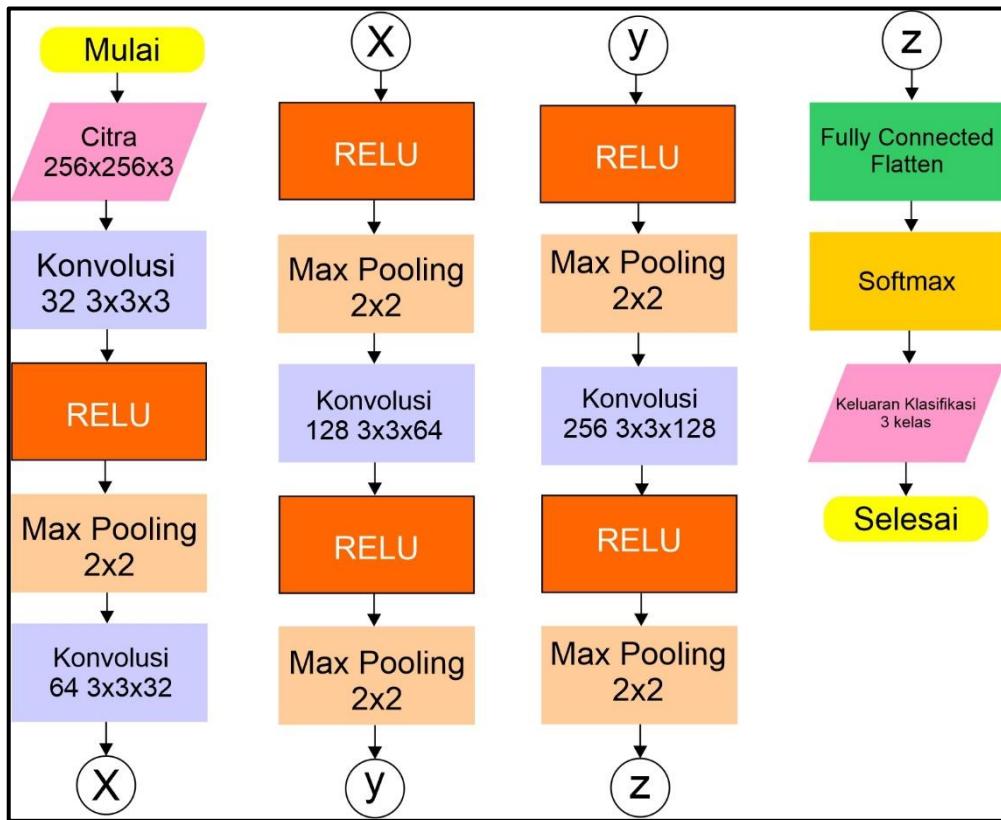
Berikutnya akan masuk dalam proses konvolusi dengan ukuran kernel 3x3 sebanyak 32 kali yang akan menjadi data masukan konvolusi dan akan dilakukan perulangan sebanyak 64. Kemudian akan diproses menggunakan fungsi aktivasi RELU untuk melakukan proses *thresholding* setelah selesai akan diambil nilai maksimalnya kembali menggunakan *max*

pooling. Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 64 lapisan akan dikonvolusi kembali sebanyak 128 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Bagian ini yang akan menjadi pengembangan dari arsitektur CNN ketiga, dimana akan ditambah 1 lapisan konvolusi, RELU, dan *pooling* untuk semakin mendalam ketika proses pengambilan fitur-fiturnya. Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 128 lapisan dan akan dikonvolusi kembali sebanyak 256 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Setelah mendapatkan *featured map* baru yang diperoleh dari *pooling layer* kemudian masuk dalam proses *fully connected* untuk disederhanakan menjadi satu layer dengan melakukan *reshape* hasil *featured map* yang didapat dari *max pooling*. Dan tahap terakhir adalah penggunaan fungsi aktivasi *softmax* untuk menangani *multiclass classification* yang berisikan 3 keluaran *neuron*.

v. Arsitektur CNN 5



Gambar 3.22 Flowchart Arsitektur 5

Jika disusun, arsitektur CNN kelima ini sebagai pengembangan dari arsitektur CNN keempat, masukan citra akan memiliki ukuran yang tidak sama maka pada tahap ini citra akan *reshape* menjadi 256x256 *pixel* yang terdiri dari 3 bagian yaitu *red*, *green*, *blue* kemudian masuk dalam proses konvolusi berukuran kernel 3x3 dengan 3 bagian yaitu *red*, *green*, *blue* sebanyak 32 menggunakan pergeseran *stride* 1 dan pelebaran *padding* 0. Kemudian masuk dalam proses *thresholding* dengan nilai nol terhadap nilai piksel pada input citra menggunakan fungsi aktivasi *relu*. Kemudian masuk dalam proses *max pooling* berukuran kernel 2x2 untuk diambil nilai maksimal dari setiap kernel pergeserannya. Berikutnya akan masuk dalam proses konvolusi dengan ukuran kernel 3x3 sebanyak 32 kali yang akan menjadi data masukan konvolusi dan akan dilakukan perulangan sebanyak 64. Kemudian akan diproses menggunakan fungsi aktivasi RELU untuk melakukan proses

thresholding setelah selesai akan diambil nilai maksimalnya kembali menggunakan *max pooling*.

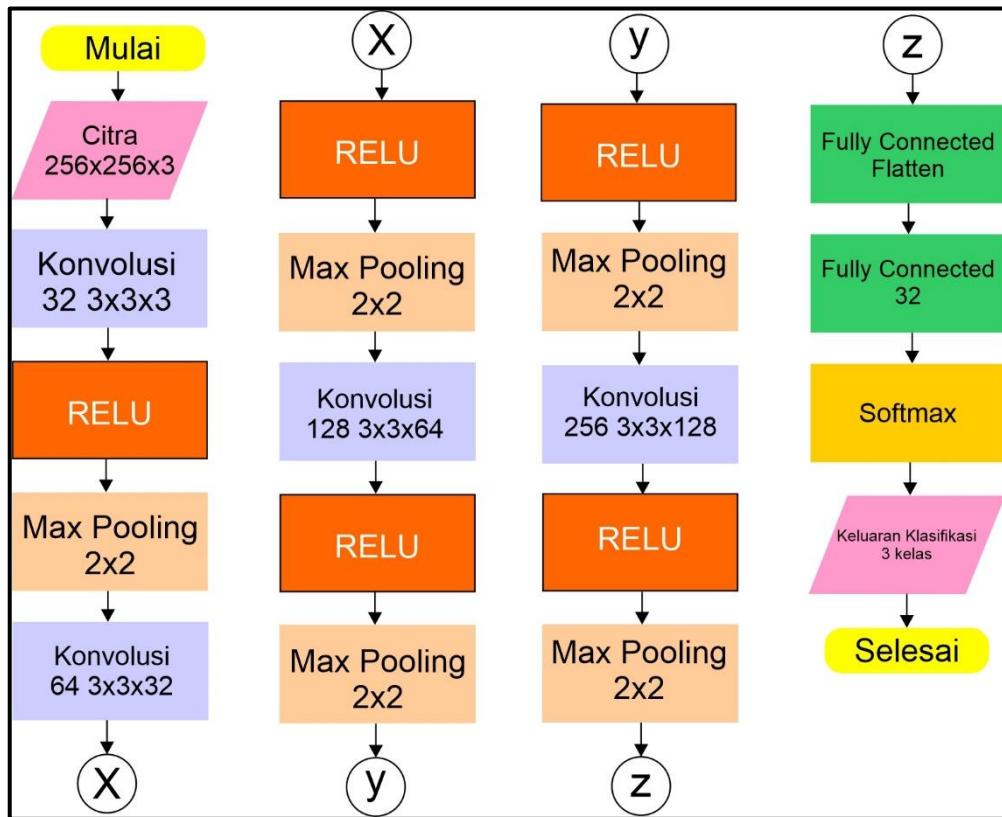
Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 64 lapisan akan dikonvolusi kembali sebanyak 128 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 128 lapisan dan akan dikonvolusi kembali sebanyak 256 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Bagian ini yang akan menjadi pengembangan dari arsitektur CNN keempat, dimana akan ditambah 1 lapisan konvolusi, RELU, dan *pooling* untuk semakin mendalam ketika proses pengambilan fitur-fiturnya. Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 256 lapisan dan akan dikonvolusi kembali sebanyak 512 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Setelah mendapatkan *featured map* baru yang diperoleh dari *pooling layer* kemudian masuk dalam proses *fully connected* untuk disederhanakan menjadi satu layer dengan melakukan *reshape* hasil *featured map* yang didapat dari *max pooling*. Dan tahap terakhir adalah penggunaan fungsi aktivasi *softmax* untuk menangani *multiclass classification* yang berisikan 3 keluaran *neuron*.

vi. Arsitektur CNN 6



Gambar 3.23 Flowchart Arsitektur 6

Jika disusun, arsitektur CNN keenam ini sebagai pengembangan dari arsitektur CNN kelima, masukan citra akan memiliki ukuran yang tidak sama maka pada tahap ini citra akan *dreshape* menjadi 256x256 *pixel* yang terdiri dari 3 bagian yaitu *red, green, blue* kemudian masuk dalam proses konvolusi berukuran kernel 3x3 dengan 3 bagian *red, green, blue* sebanyak 32 menggunakan pergeseran *stride* 1 dan pelebaran *padding* 0. Kemudian masuk dalam proses *thresholding* dengan nilai nol terhadap nilai piksel pada input citra menggunakan fungsi aktivasi *relu*. Kemudian masuk dalam proses *max pooling* berukuran kernel 2x2 untuk diambil nilai maksimal dari setiap kernel pergeserannya.

Berikutnya akan masuk dalam proses konvolusi dengan ukuran kernel 3x3 sebanyak 32 kali yang akan menjadi data masukan konvolusi dan akan dilakukan perulangan sebanyak 64. Kemudian akan diproses menggunakan fungsi aktivasi *RELU* untuk melakukan proses

thresholding setelah selesai akan diambil nilai maksimalnya kembali menggunakan *max pooling*.

Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 64 lapisan akan dikonvolusi kembali sebanyak 128 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

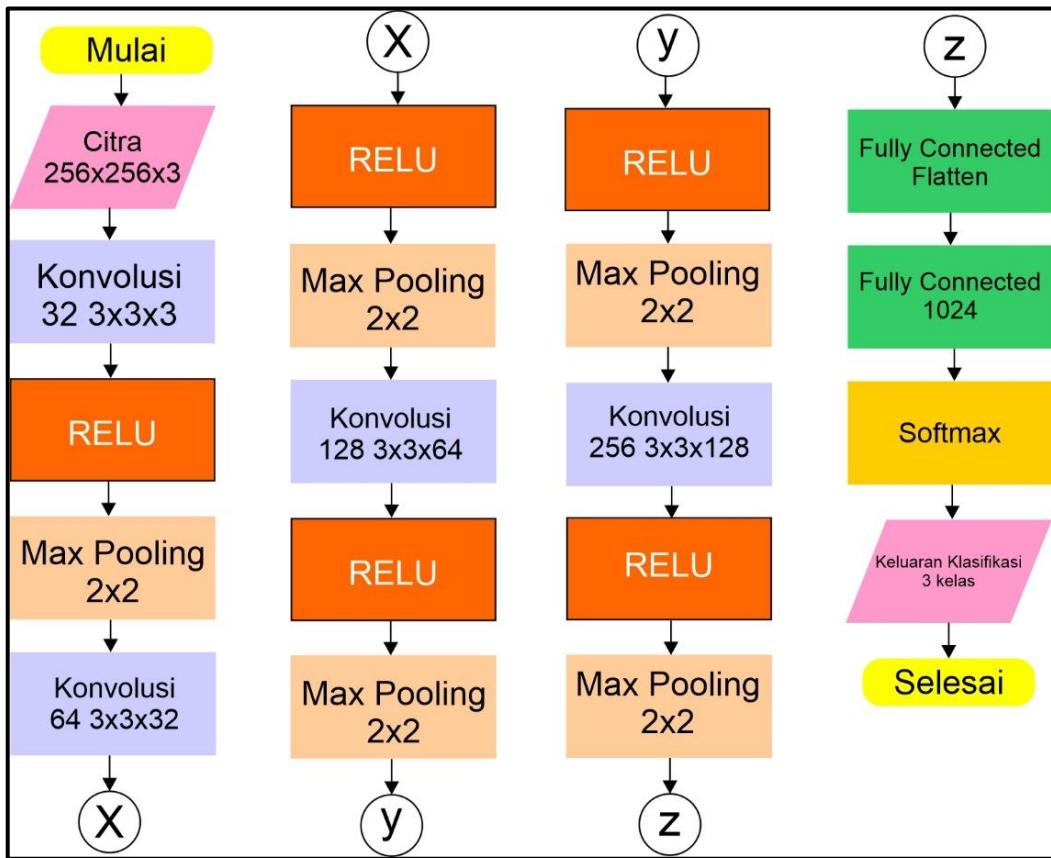
Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 128 lapisan dan akan dikonvolusi kembali sebanyak 256 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 256 lapisan dan akan dikonvolusi kembali sebanyak 512 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Setelah mendapatkan *featured map* baru yang diperoleh dari *pooling layer* kemudian masuk dalam proses *fully connected* untuk disederhanakan menjadi satu layer dengan melakukan *reshape* hasil *featured map* yang didapat dari *max pooling*. Pada bagian ini yang mebedakan antara arsitektur CNN keenam dengan arsitektur sebelumnya, menambahkan 1 layer *fully connected* untuk dapat lebih menyederhanakan lapisan yang dimasukkan. Layer ini akan mengkalikan 32 *neuron* sehingga variasi data dalam 1 dimensi *array* akan semakin bervariasi.

Dan tahap terakhir adalah penggunaan fungsi aktivasi *softmax* untuk menangani *multiclass classification* yang berisikan 3 keluaran *neuron*.

vii. Arsitektur CNN 7



Gambar 3.24 Flowchart Arsitektur CNN 7

Jika disusun, arsitektur CNN ketujuh ini sebagai pengembangan dari arsitektur CNN keenam, masukan citra akan memiliki ukuran yang tidak sama maka pada tahap ini citra akan *reshape* menjadi 256x256 pixel yang terdiri dari 3 bagian yaitu red, green, blue kemudian masuk dalam proses konvolusi berukuran kernel 3x3 dengan 3 bagian yaitu red, green, blue sebanyak 32 menggunakan pergeseran *stride* 1 dan pelebaran *padding* 0. Kemudian masuk dalam proses *thresholding* dengan nilai nol terhadap nilai piksel pada input citra menggunakan fungsi aktivasi *relu*. Kemudian masuk dalam proses *max pooling* berukuran kernel 2x2 untuk diambil nilai maksimal dari setiap kernel pergeserannya.

Berikutnya akan masuk dalam proses konvolusi dengan ukuran kernel 3x3 sebanyak 32 kali yang akan menjadi data masukan konvolusi dan akan dilakukan perulangan sebanyak 64. Kemudian akan diproses menggunakan fungsi aktivasi RELU untuk melakukan proses

thresholding setelah selesai akan diambil nilai maksimalnya kembali menggunakan *max pooling*.

Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 64 lapisan akan dikonvolusi kembali sebanyak 128 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

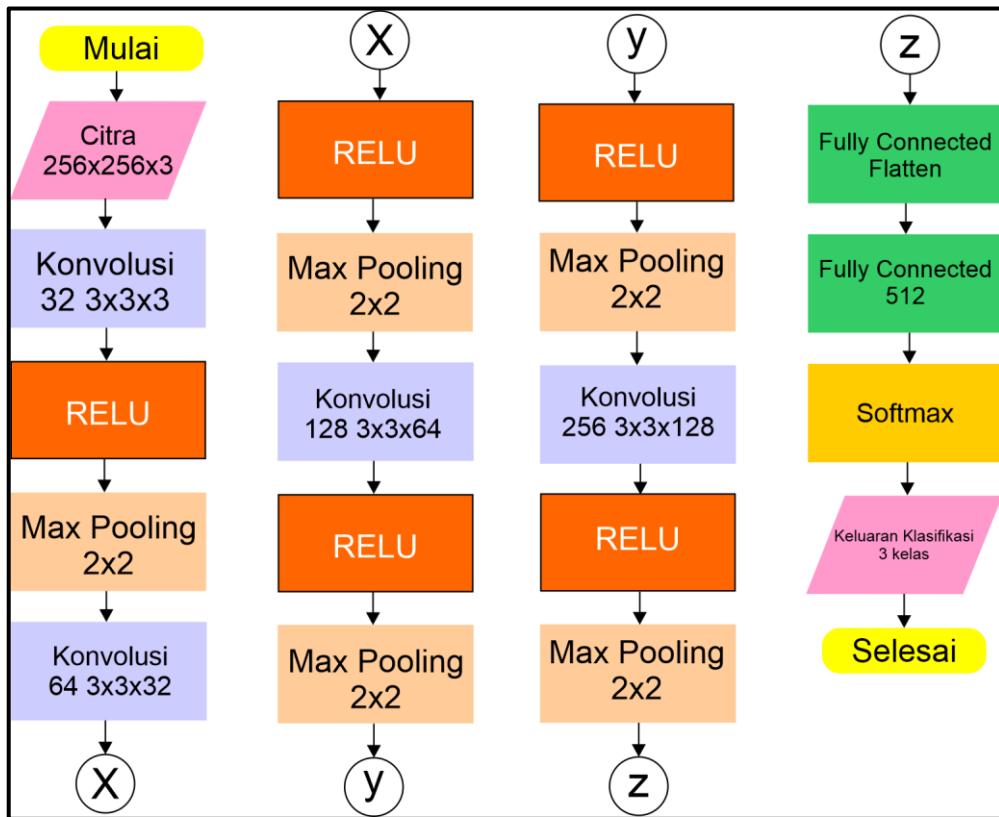
Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 128 lapisan dan akan dikonvolusi kembali sebanyak 256 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 256 lapisan dan akan dikonvolusi kembali sebanyak 512 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Setelah mendapatkan *featured map* baru yang diperoleh dari *pooling layer* kemudian masuk dalam proses *fully connected* untuk disederhanakan menjadi satu layer dengan melakukan *reshape* hasil *featured map* yang didapat dari *max pooling*. Pada bagian ini yang mebedakan antara arsitektur CNN ketujuh dengan arsitektur keenam, menggunakan 1 *layer* *fully connected* untuk dapat lebih menyederhanakan lapisan yang dimasukkan. *Layer* ini akan mengkalikan 1024 *neuron* sehingga variasi data dalam 1 dimensi *array* akan semakin bervariasi.

Dan tahap terakhir adalah penggunaan fungsi aktivasi *softmax* untuk menangani *multiclass classification* yang berisikan 3 keluaran *neuron*.

viii. Arsitektur CNN 8



Gambar 3.25 Flowchart Arsitektur CNN 8

Jika disusun, arsitektur CNN kedelapan ini sebagai pengembangan dari arsitektur CNN keenam, masukan citra akan memiliki ukuran yang tidak sama maka pada tahap ini citra akan *reshape* menjadi 256x256 pixel yang terdiri dari 3 bagian yaitu *red*, *green*, *blue* kemudian masuk dalam proses konvolusi berukuran kernel 3x3 dengan 3 bagian yaitu *red*, *green*, *blue* sebanyak 32 menggunakan pergeseran *stride* 1 dan pelebaran *padding* 0. Kemudian masuk dalam proses *thresholding* dengan nilai nol terhadap nilai piksel pada input citra menggunakan fungsi aktivasi *relu*. Kemudian masuk dalam proses *max pooling* berukuran kernel 2x2 untuk diambil nilai maksimal dari setiap kernel pergeserannya.

Berikutnya akan masuk dalam proses konvolusi dengan ukuran kernel 3x3 sebanyak 32 kali yang akan menjadi data masukan konvolusi dan akan dilakukan perulangan sebanyak 64. Kemudian akan diproses menggunakan fungsi aktivasi RELU untuk melakukan proses

thresholding setelah selesai akan diambil nilai maksimalnya kembali menggunakan *max pooling*.

Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 64 lapisan akan dikonvolusi kembali sebanyak 128 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 128 lapisan dan akan dikonvolusi kembali sebanyak 256 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

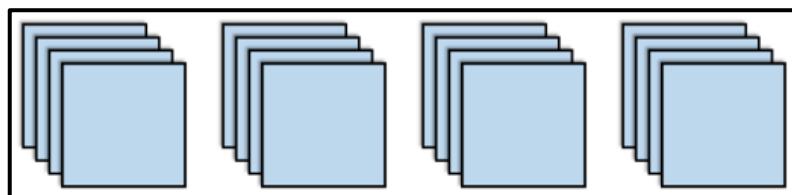
Menggunakan data *input* yang berasal dari hasil *pooling layer* sebelumnya yaitu kernel berukuran 3x3 sebanyak 256 lapisan dan akan dikonvolusi kembali sebanyak 512 kali, kemudian akan dilakukan *thresholding* dengan fungsi aktivasi *relu* dilanjutkan dengan pengambilan nilai maksimal pada *pooling layer* ukuran kernel 2x2 untuk menghasilkan *featured map* baru.

Setelah mendapatkan *featured map* baru yang diperoleh dari *pooling layer* kemudian masuk dalam proses *fully connected* untuk disederhanakan menjadi satu layer dengan melakukan *reshape* hasil *featured map* yang didapat dari *max pooling*. Pada bagian ini yang mebedakan antara arsitektur CNN ketujuh dengan arsitektur keenam, menggunakan 1 *layer* *fully connected* untuk dapat lebih menyederhanakan lapisan yang dimasukkan. *Layer* ini akan mengkalikan 1024 *neuron* sehingga variasi data dalam 1 dimensi *array* akan semakin bervariasi.

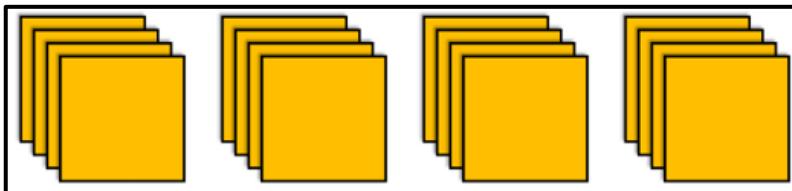
Dan tahap terakhir adalah penggunaan fungsi aktivasi *softmax* untuk menangani *multiclass classification* yang berisikan 3 keluaran *neuron*.

2. Pengaruh Jumlah *Batch*

Ukuran *batch* adalah *hyperparameter* yang menentukan jumlah sampel yang harus dikerjakan sebelum memperbarui parameter model *internal*. Bayangkan *batch* sebagai pengulangan untuk satu atau beberapa sampel dan buat prediksi. Pada akhir *batch*, prediksi dibandingkan dengan *variabel* keluaran yang diharapkan dan kesalahan dihitung. Dari kesalahan ini, algoritma pembaruan digunakan untuk memperbaiki model, mis. bergerak ke bawah sepanjang gradien kesalahan.



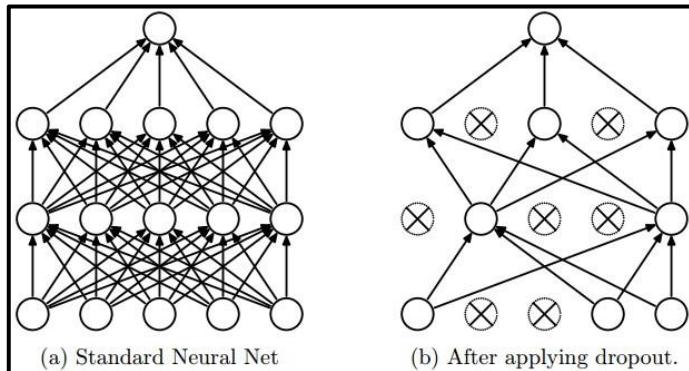
Gambar 3.26 Ilustrasi 4 *Batch* dalam *Training*



Gambar 3.27 Ilustrasi 4 *Batch* dalam *Validasi*

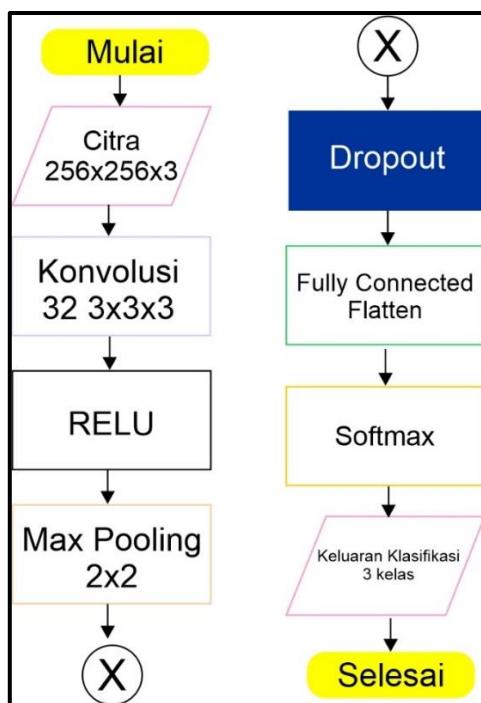
Set data pelatihan dapat dibagi menjadi satu atau beberapa kelompok. Jika semua sampel pelatihan digunakan untuk membuat satu *batch*, algoritma pembelajaran ini disebut penurunan gradien *batch*. Jika *batch* berukuran satu sampel, algoritma pembelajarannya disebut penurunan gradien stokastik. Jika ukuran tumpukan lebih dari satu sampel dan kurang dari ukuran set data pelatihan, algoritme pembelajaran disebut penurunan gradien tumpukan mini. Penurunan *Gradien Batch* dengan ukuran *batch* samadengan ukuran set pelatihan. Penurunan *Gradien Stochastic* dengan ukuran *batch* bernilai 1. Penurunan *Gradien Batch Mini* bernilai minimal 1 dari Ukuran *batch* hingga ukuran *set* pelatihan. Dalam kasus penurunan *Gradien Batch Mini*, ukuran tumpukan populer mencakup 32, 64,dan 128 sampel kemudian jumlah tersebut yang digunakan dalam percobaan pembuatan model.

3. Pengaruh Jumlah *Dropout*



Gambar 3.28 Ilustrasi Menggunakan *Dropout*

Dropout Layer adalah *layer* yang berfungsi melakukan regularisasi pada JST dengan mematikan *neuron* secara acak dengan kemungkinan *drop_rate* pada tiap iterasi pelatihan (Hinton et al., 2012).



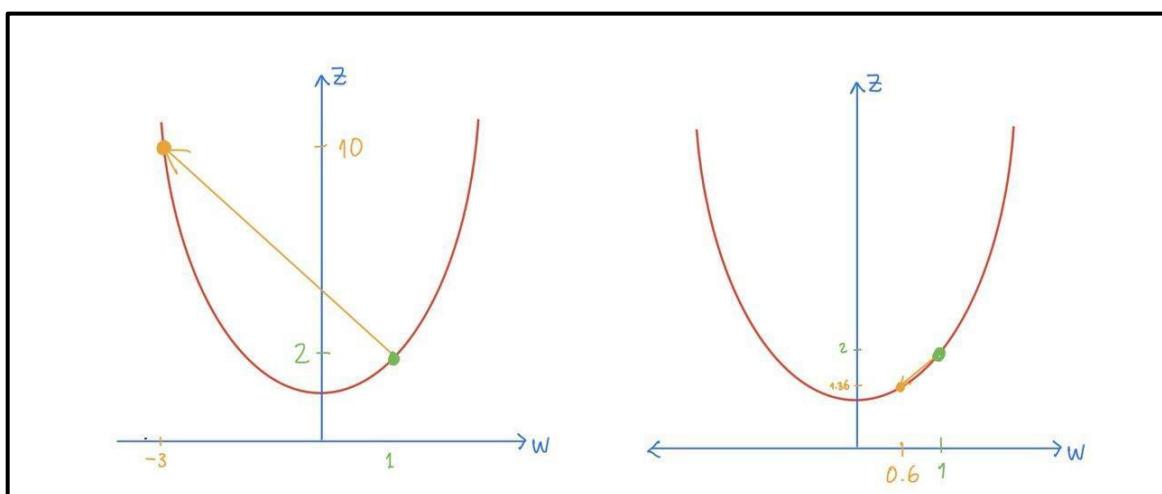
Gambar 3.29 Flowchart Contoh Arsitektur CNN Menggunakan *Dropout*

Dropout berfungsi dengan mencegah adanya ketergantungan antar *neuron* pada jaringan. Ketika tahap pengujian, *dropout* dinonaktifkan untuk mendapatkan hasil menyeluruh, namun output dari masing-masing neuron yang menggunakan *dropout* dikalikan dengan *drop_rate* agar outputnya proporsional dengan ketika tahap pelatihan (Putra, 2015).

4. Pengaruh Jumlah *Learning Rate*

Learning rate menentukan perubahan bobot yang dihasilkan di setiap pelatihan siklik. Besar kecepatan pembelajaran dapat mengakibatkan sistem yang tidak pasti, tetapi sistem yang kecil dapat menyebabkan sistem yang lebih lama waktu pelatihan dan konvergensi yang lambat. Jadi cenderung memilih kecepatan pembelajaran kecil untuk memastikan stabilitas sistem. Rentang nilai *learning rate* pada umumnya adalah 0 hingga 0,0005.

Dengan menggunakan fungsi $z = w^2 + 1$ dan inisialisasi nilai $w=1$ kita akan mencari nilai z paling minimal. Nilai yang diinginkan adalah saat z mencapai optimal yaitu $z=1$. Dengan menggunakan limit tangent, kita mendapatkan nilai dz/dw yaitu $2w$, yang dalam kasus ini berarti $2*1 = 2$. Turunan ini dapat digunakan untuk melakukan update nilai w sesuai dengan learning rate. Nilai w setelah dilakukan update dengan menggunakan $lr=2$ dan $lr=0.2$ masing-masing menjadi -3 dan 0.6. Sehingga menghasilkan nilai z baru yaitu 10 dan -0.6. Ilustrasi hasil proses update pada nilai w dengan menggunakan 2 learning rate yang berbeda dapat dilihat pada gambar 3.29.



Gambar 3.30 Kurva Penggunaan *Learning Rate* yang Berbeda $lr=2$ (kiri) $lr=0.2$ (kanan)

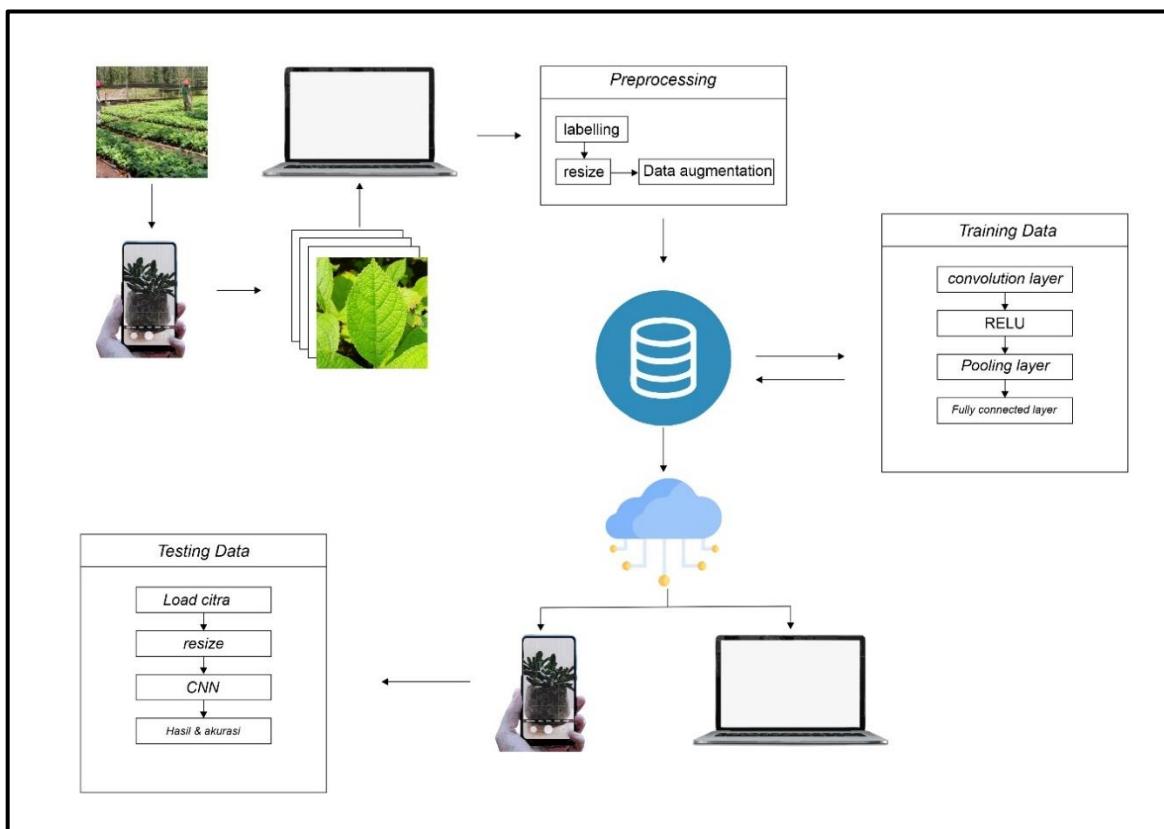
Dengan demikian, dapat dilihat bahwa penggunaan learning rate yang terlalu besar akan menyebabkan perubahan yang besar pula, sehingga menyebabkan dilewatkannya solusi optimal yang diinginkan. Namun, learning rate yang terlalu kecil juga akan menyebabkan

proses mencapai solusi optimal menjadi sangat lambat. Jadi kita harus benar-benar menentukan nilai learning rate yang tepat dalam melakukan training model neural network.

3.2.3 Perancangan

Perancangan pada aplikasi identifikasi varietas jati ini memiliki beberapa rancangan(*design*). Rancangan tersebut dibuat berdasarkan analisa sistem yang akan dibangun. Rancangan pada penitian ini dibagi menjadi lima, yaitu perancangan arsitektur sistem, *unified modelling language* (UML), perancangan tampilan antarmuka, dan perancangan pengujian.

1. Perancangan Arsitektur Sistem



Gambar 3.31 Perancangan Arsitektur Sistem

Gambar 3.30 merupakan skema perancangan arsitektur sistem yang dibuat dalam penelitian ini. Arsitektur sistem menggambarkan hubungan antar perangkat lunak dan perangkat keras dengan aplikasi identifikasi varietas tanaman jati. Tahap pertama pengambilan data dari kebun persemaian jati dengan cara memotret satu-persatu daun

varietas jati yang sudah ditentukan. Setelah mengumpulkan data citra foto daun varietas jati kemudian diolah pada *laptop* atau perangkat sejenisnya untuk dikumpulkan dan diolah lebih lanjut. Data citra foto daun jati kemudian akan dilakukan proses *preprocessing* dimana diolah terlebih dahulu sebelum masuk dalam pemrosesan lebih lanjut. Setelah dilakukannya *preprocessing* kemudian akan disimpan dan digunakan dalam tahap pelatihan(*training*) atau pembelajaran model yang kemudian setelah selesai akan disimpan kembali dalam bentuk file adalah .h5 .

Data yang sudah disimpan akan dirancang sebuah program berbasis web yang sudah terkoneksi dengan *cloud* yaitu *hosting* menggunakan jasa layanan *herokuapp*, digunakan layanan *cloud* supaya dapat diakses pada perangkat ponsel, laptop maupun perangkat lainnya yang dapat mengaksesnya.

2. Perancangan *Unified Modeling Language*(UML)

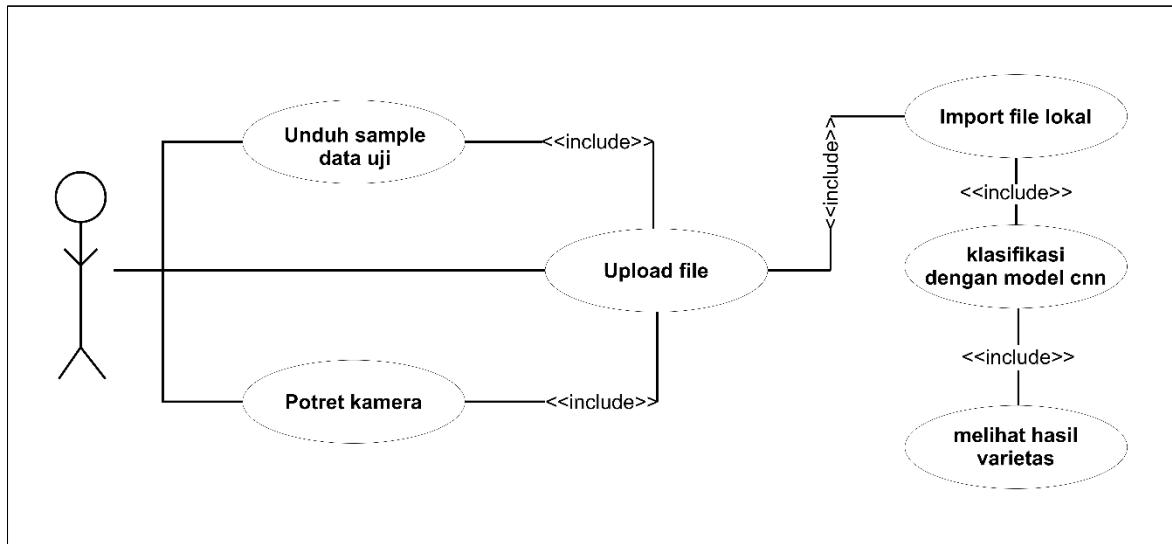
Unified Modelling Language (UML) adalah modelling untuk merinci, menggambarkan, dan mendokumentasikan suatu sistem perangkat lunak yang dibangun dengan pendekatan objek, dimana model-model disajikan dalam bentuk diagram-diagram (Nugroho, 2009).

Unified Modelling Language (UML) berfungsi dalam membantu memvisualisasi pemodelan skema sistem pada aplikasi yang akan digambarkan dalam bentuk diagram. Tujuan atau fungsi pembuatan UML diantaranya:

- a. Dapat memberikan bahasa permodelan visual kepada pengguna dari berbagai macam pemerograman maupun proses rekayasa.
- b. Dapat menyatukan praktek-praktek terbaik yang ada dalam permodelan.
- c. Dapat memberikan model yang siap untuk digunakan, merupakan bahasa permodelan visual yang ekspresif untuk mengembangkan sistem dan untuk saling menukar model secara mudah.

Berikut ini merupakan kerangka isi dari *unified modelling language* (UML):

i. *Use Case Diagram*



Gambar 3.32 Use Case Diagram Identifikasi Varietas

Untuk dapat menggunakan identifikasi citra foto daun jati maka dibuatkan aplikasi web dengan 1 aktor yaitu pengguna dengan studi kasus masyarakat awam. Terdapat tiga aktivitas yang dapat dilakukan oleh aktor yaitu yang pertama mengunduh contoh data uji(testing) apabila belum memiliki data uji, yang kedua yaitu melakukan pengunggahan gambar daun, dan yang ke tiga ketika pengguna berada langsung di kebun persemaian jati dengan cara melakukan pemotretan menggunakan aplikasi kamera bawaan dari smartphone pengguna.

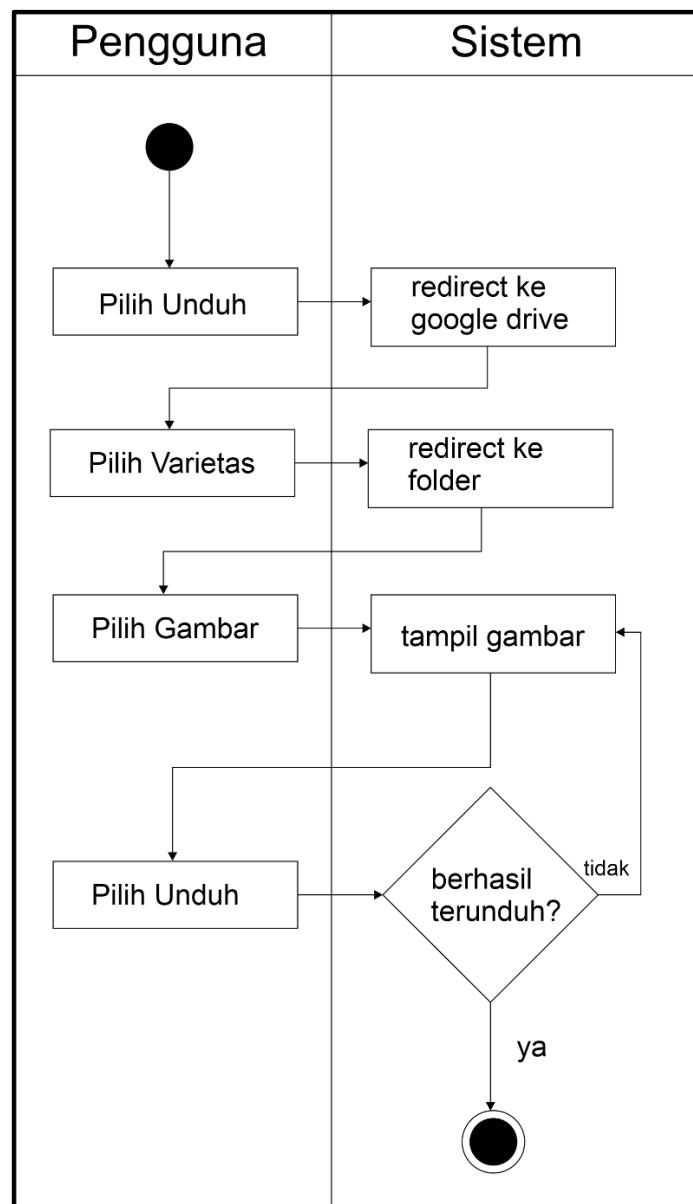
Untuk dapat melakukan hasil kelas citra daun yang diuji harus melakukan terlebih dahulu dengan menyediakan gambar citra daun dengan cara mengunggah citra daun jati kemudian memilih gambar dari galeri atau file lokal, setelah itu dilakukan proses pengenalan menggunakan model CNN yang sudah dilakukan proses pembelajaran dan yang terakhir dapat melihat hasil kelas dari daun yang diunggah tersebut. Case yang kedua yaitu mengunduh data uji, dilakukan apabila pengguna belum punya data citra daun jati, yang kemudian akan diarahkan pada link url google drive dimana lokasi kumpulan data uji berada,

setelah itu memilih citra daun jati yang diingkan dan terakhir adalah mengunduh untuk kemudian disimpan dijadikan pengujian.

ii. *Activity diagram*

Pada bagian ini aktivitas diagram(*activity diagram*) menggambarkan berbagai alur aktivitas yang dilakukan masing-masing objek dalam sistem yang akan dirancang. Menggambarkan dimana masing-masing alur berawal, aktivitas yang dilakukan saat pertama kali, serta bagaimana alur aktivitas tersebut berakhir atau selesai.

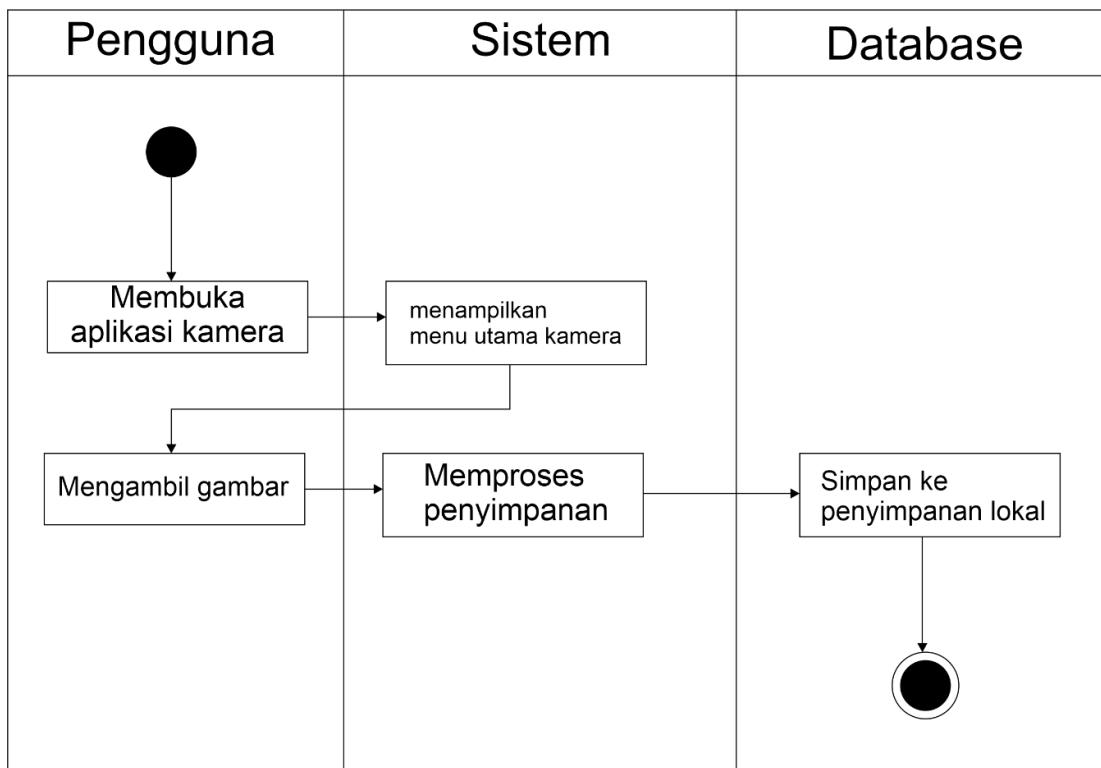
a. *Activity diagram* unduh sample data uji



Gambar 3.33 Aktivitas Diagram Unduh Sample Data Uji

Apabila pengguna belum memiliki data citra daun jati maka ada data citra yang disiapkan untuk bisa dicoba. Terdapat menu unduh data uji yang nantinya akan diarahkan ke sebuah link url yaitu ke *google drive* yang berisikan kumpulan data-data citra daun jati. Pengguna dapat memilih kelas varietas sesuai keinginannya sesuai folder yang sudah disediakan. Nama folder yang tersedia hanya untuk menandai kelas varietas jati yang diunduh dan sebagai patokan apakah nanti setelah diuji kelasnya sudah sesuai atau belum. Kemudian memilih gambar *sample* daun yang ingin diuji, jika sudah memilih langsung bisa diunduh dengan menggunakan menu unduh.

b. *Activity diagram* potret kamera

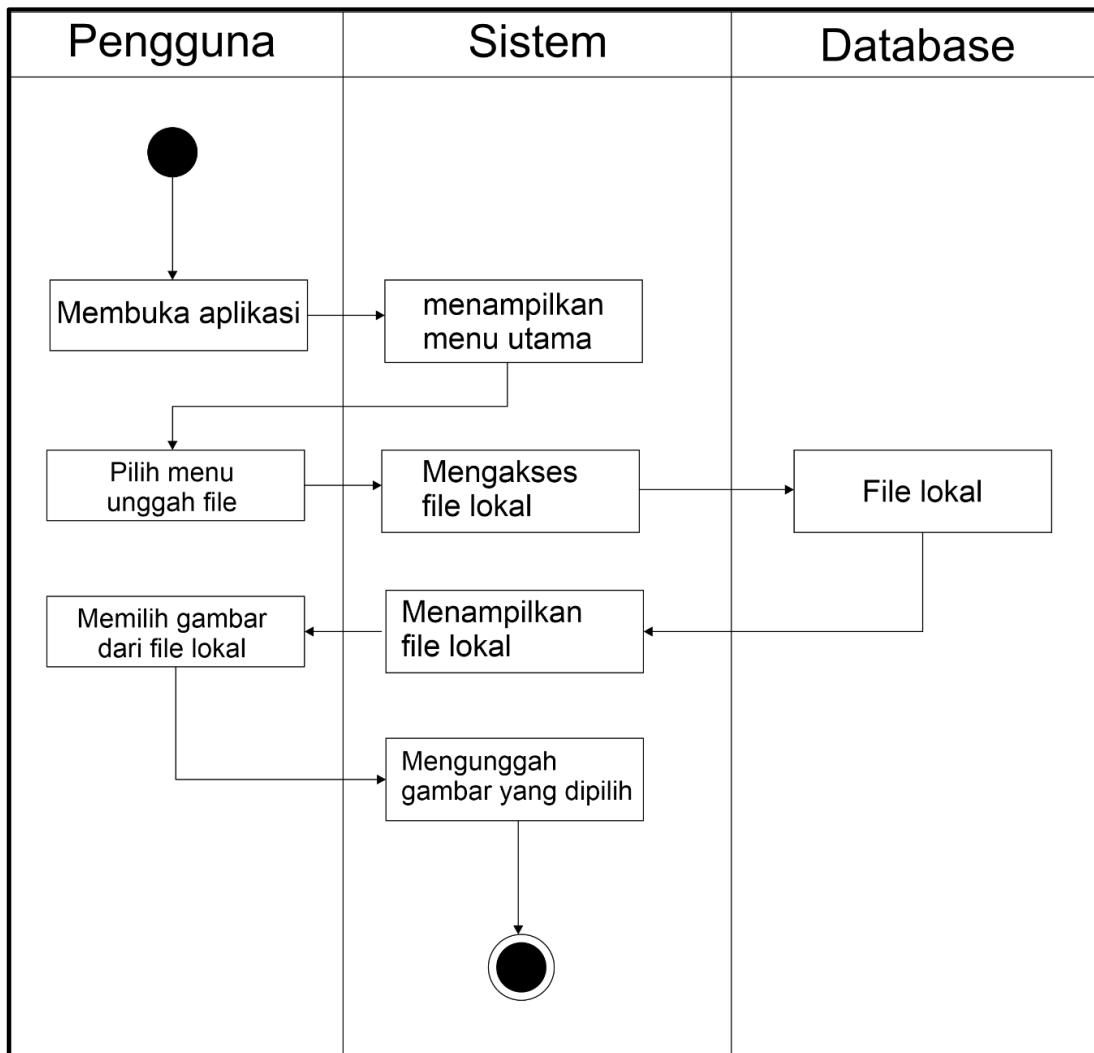


Gambar 3.34 Aktivitas Diagram Potret Kamera

Langkah awal apabila pengguna ingin melakukan pengidentifikasi varietas jati, langkah-langkah ini hanya untuk penggunaan pada *smartphone*, tidak dianjurkan apabila menggunakan laptop maupun *personal computer*. Pengguna membuka aplikasi kamera bawaan dari *smartphone* milik masing-masing pengguna. Setelah aplikasi kamera terbuka maka akan muncul tampilan menu utama dari aplikasi kamera tersebut, tampilan yang ada

dari kamera masing-masing pengguna akan berbeda tampilan utamanya, berikutnya pengguna mengambil gambar atau memotret seperti biasa dengan mengarahkan smarphone mendekat kearah daun jati yang ingin diambil gambarnya.

c. *Activity diagram upload file*

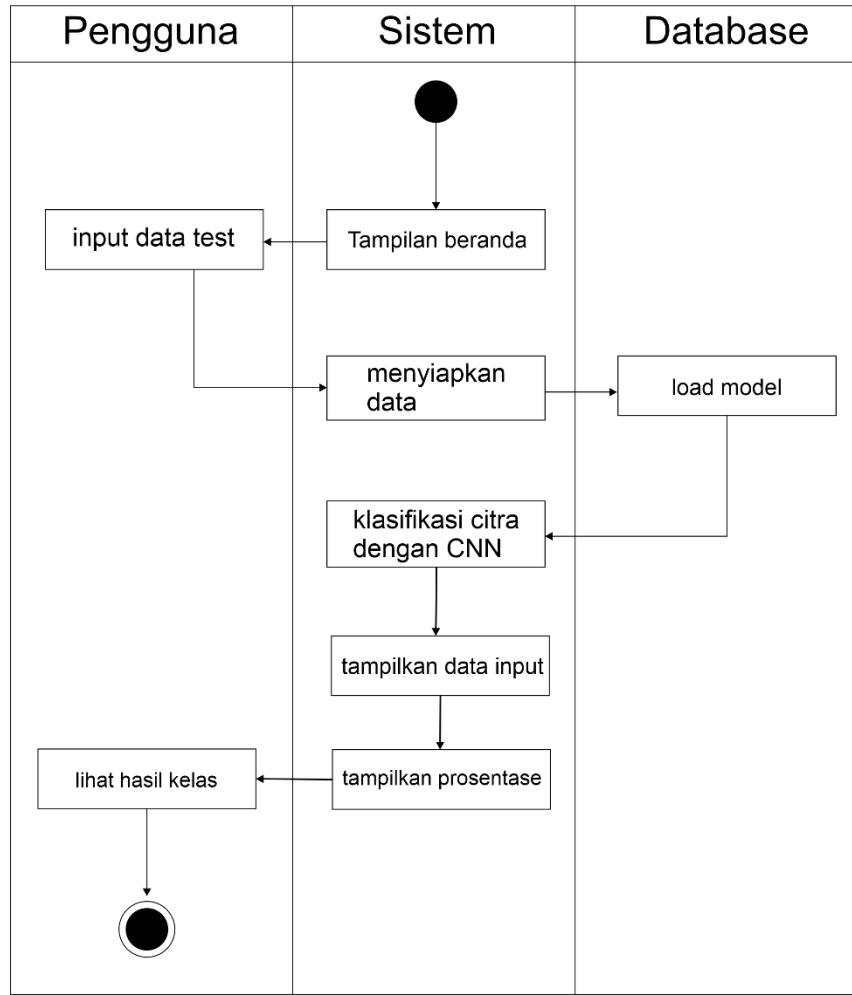


Gambar 3.35 Aktivitas Diagram Upload File

Tahap ini untuk memasukkan data ke dalam aplikasi, pengguna lebih terdahulu mengunggah gambar yang ingin dicari tau nama varietas jatinya, dengan memanfaatkan file lokal yang ada gambar-gambar yang telah dipotret maupun data sampel yang sudah diunduh pada tahap sebelumnya selanjutnya akan tersimpan pada file lokal yang ada baik itu pada *smartphone* maupun personal *computer*. Langkah awal pengguna terlebih dahulu membuka aplikasi web identifikasi varietas jati, kemudian pilih menu unggah file yang sudah tersedia.

Aplikasi akan memunculkan popup lokasi file lokal berada dan akan memunculkan data-data gambar yang ada pada lokal. Setelah muncul pengguna baru bisa memilih gambar yang ingin diketahui nama varietas jatinya baik hasil dari pemotretan gambar atau dari data sampel.

d. *Activity diagram identifikasi daun*



Gambar 3.36 Aktivitas Diagram Identifikasi

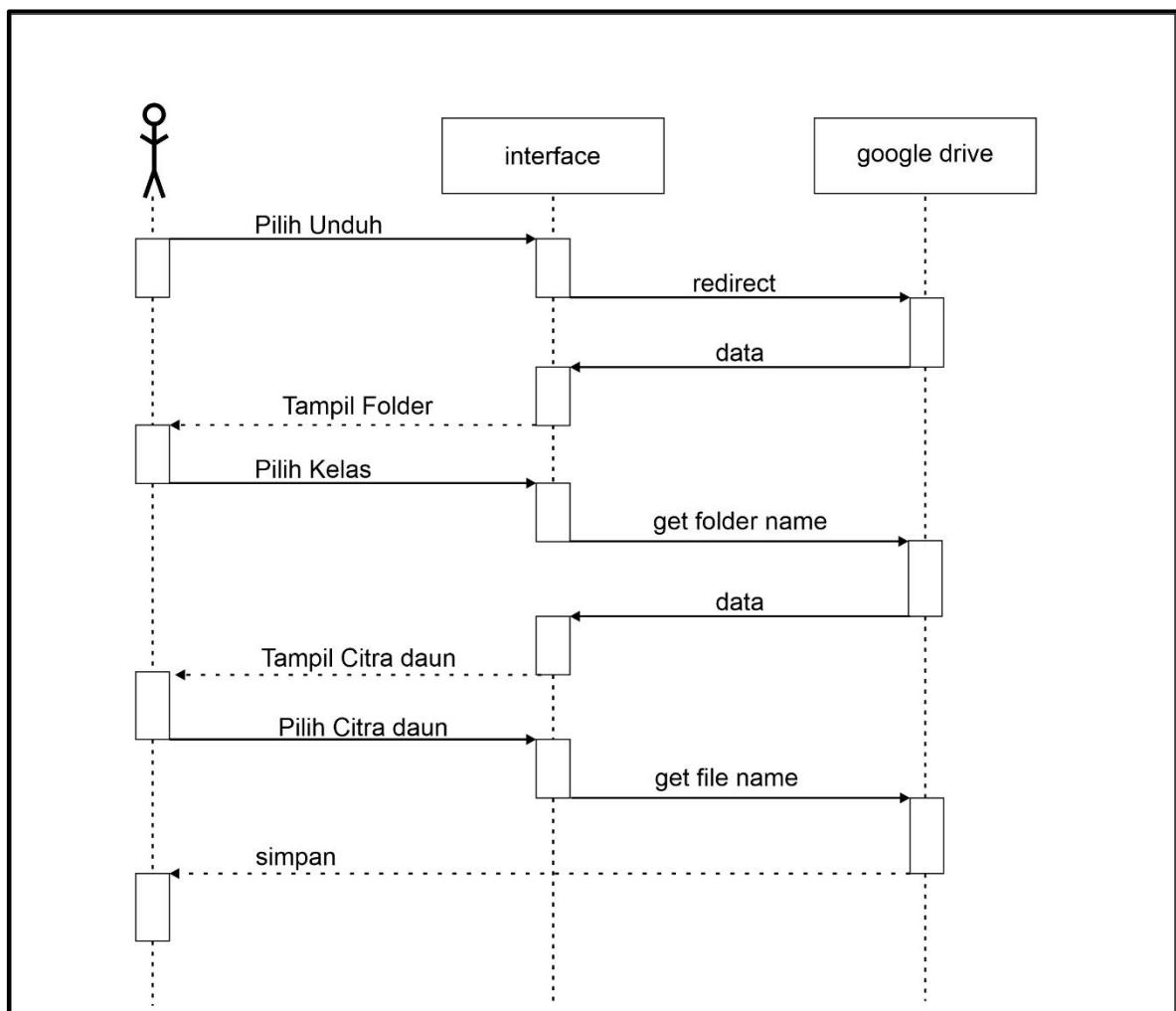
Aktivitas diagram yang terakhir adalah aktivitas diagram identifikasi varietas jati yang dapat digunakan oleh pengguna. Pertama aplikasi akan menampilkan terlebih dahulu halaman utama atau beranda yang nantinya akan digunakan oleh pengguna untuk memasukkan citra foto daun jati baru yang akan memproses data citra daun masukan tadi, kemudian sistem akan memanggil data model yang sudah disediakan, jika foto baru yang diuji tidak sesuai maka akan kembali kepada pengguna untuk menampilkan error supaya

vektorisasi. Setelah mendapatkan nilai vektornya maka akan masuk dalam CNN untuk diolah dengan model dan arsitektur yang sudah dimiliki untuk dibaca pola dari nilai-nilai vektor yang masuk yang berguna untuk memprediksi citra daun jati yang sedang diuji.

iii. *Sequance Diagram*

Sequence diagram ini adalah diagram yang menggambarkan kolaborasi dinamis antara sejumlah object. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara object juga interaksi antara *object*. Sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

a. *Sequance diagram* unduh *sample* data uji

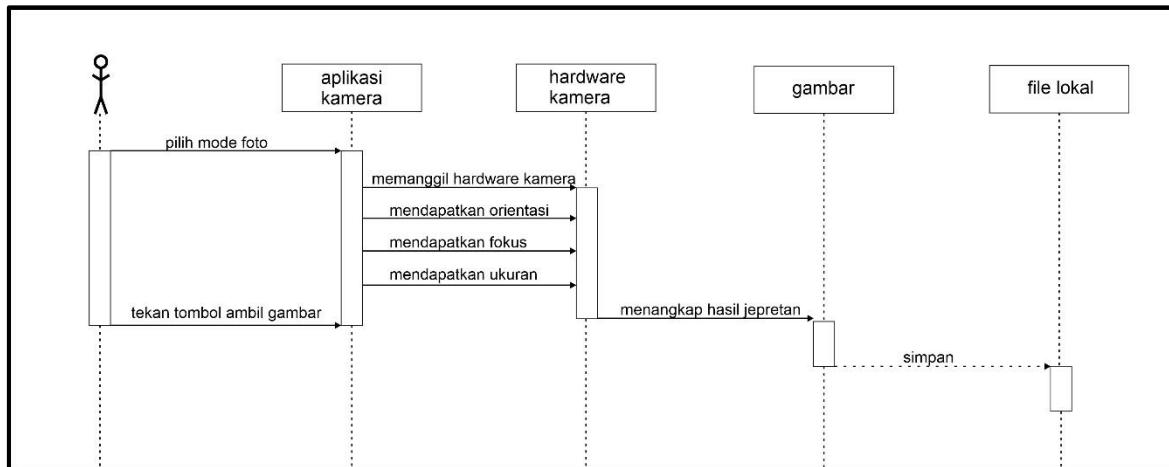


Gambar 3.37 *Sequance Diagram* Unduh Sample Data Uji

Setelah penggambaran alur aktivitas diagaram maka kali masuk dalam *sequance* diagram. Aktor melakukan pemilihan menu unduh yang ada pada tampilan beranda, kemudian aplikasi akan mengarahkan pada *google drive* dimana lokasi folder yang berisikan data-data citra daun jati. Setelah berhasil pindah maka tampilan akan berubah menjadi daftar dari folder nama varietas jati, kemudian pengguna dapat memilih terlebih dahulu kelas varietas yang diinginkan, selanjutnya *google drive* akan mendapatkan nama folder kelas yang dipilih tadi, dan setelah itu akan muncul daftar dari citra-citra daun sesuai kelas varietas yang dipilih sebelumnya.

Tahap terakhir pengguna dapat memilih gambar daun yang diinginkan yang nantinya bisa langsung diunduh untuk disimpan kedalam galeri atau penyimpanan internal milik pengguna, untuk dijadikan contoh gambar data uji.

b. *Sequance diagram* potret kamera

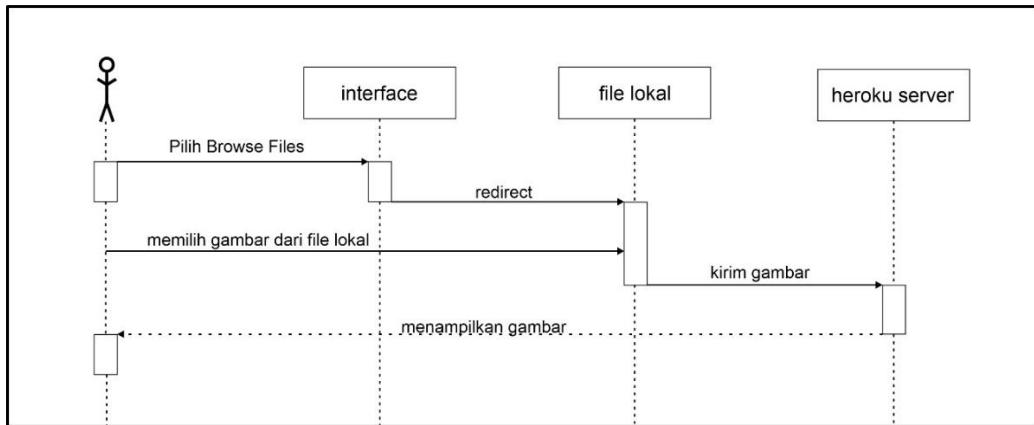


Gambar 3.38 *Sequance Diagram* Potret Kamera

Terdiri dari satu aktor dimana pengguna dapat menggunakan aplikasi kamera yang ada pada masing-masing telepon pintarnya. Kemudian pilih mode potret atau foto pada isi aplikasi yang nantinya akan mengaktifkan *hardware* kamera yang tertanam pada ponsel, dan kemudian posisikan terlebih dahulu ponsel pintar terhadap daun yang diinginkan, atur orientasi apakah landscape atau portrait, atur fokus kamera, dan jika sudah pengguna dapat

menekan tombol ambil gambar yang nantinya akan dirubah menjadi gambar dan tersimpan pada file lokal ponsel.

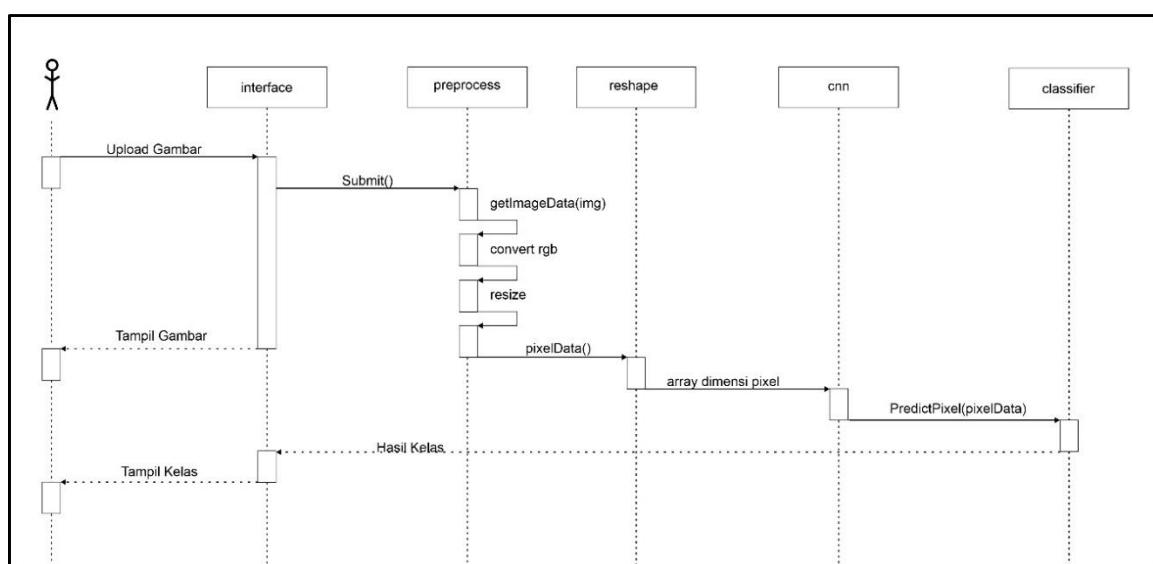
c. *Sequance diagram upload file*



Gambar 3.39 Sequance Diagram Upload File

Aktor yang terdapat pada *sequance diagram upload file* hanyalah satu yaitu pengguna atau masyarakat umum, dimulai dari pengguna memilih menu *browse file* yang ada pada menu halaman utama kemudian akan diarahkan menuju penyimpanan lokal pada ponsel pintar atau *personal computer* untuk memilih gambar yang akan diunggah, dan setelah memilih akan langsung terunggah menuju server milik heroku. Dan jika berhasil gambar akan muncul pada aplikasi.

d. *Sequance diagram identifikasi*

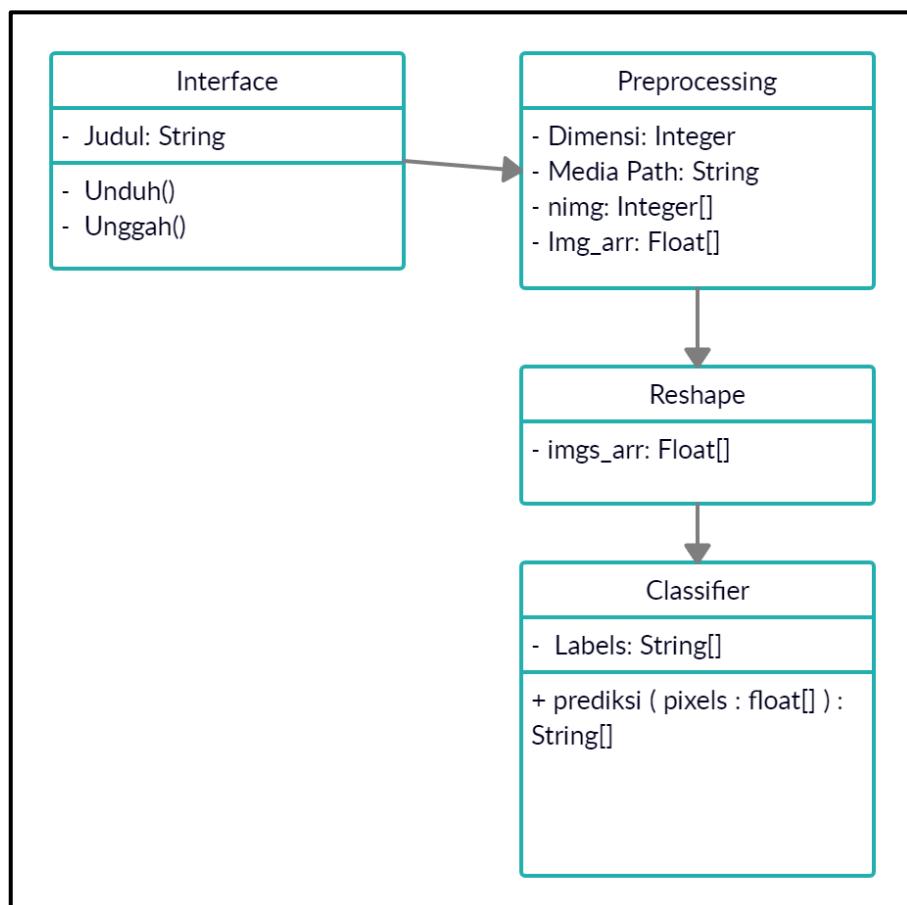


Gambar 3.40 Sequance Diagram Identifikasi

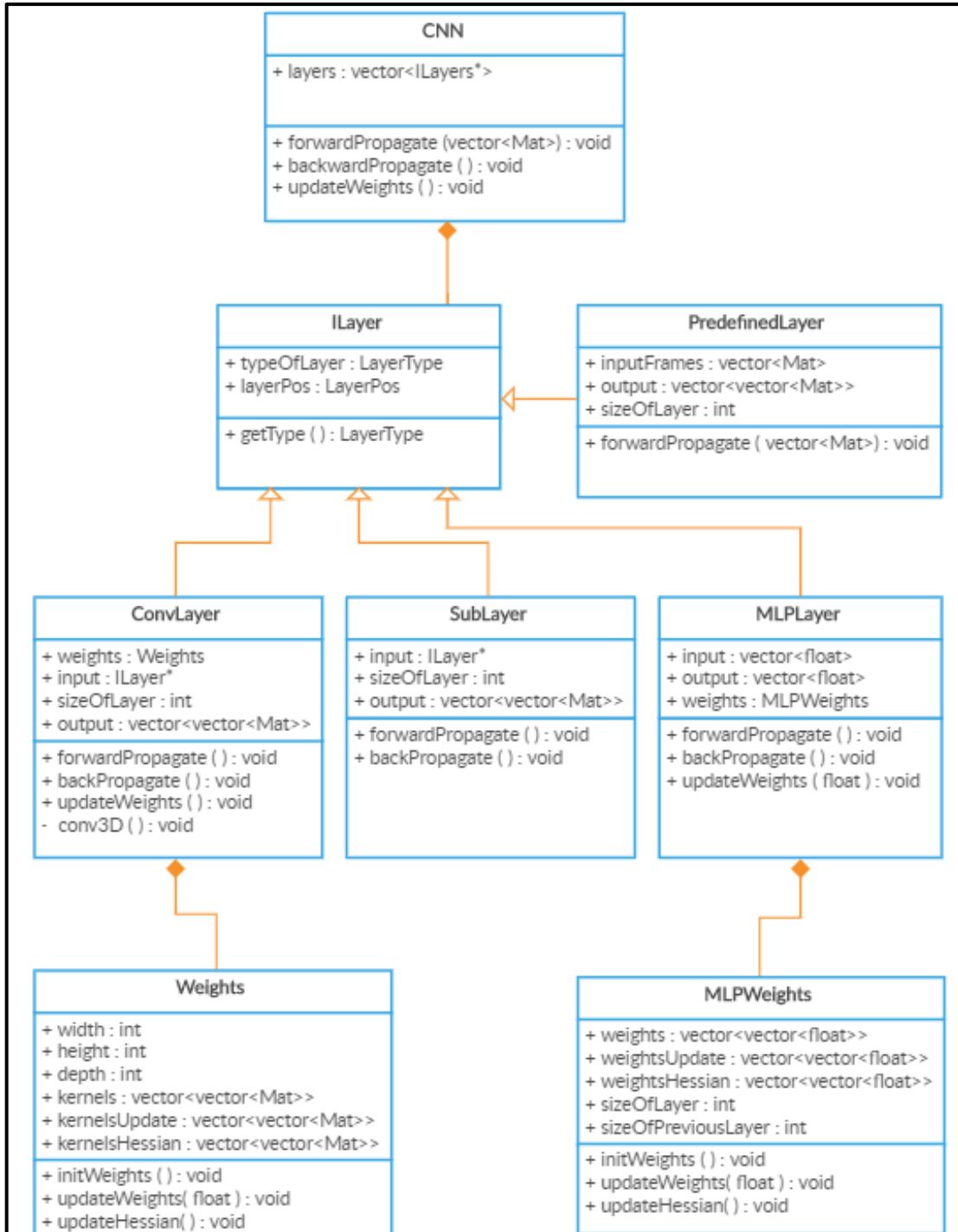
Pada kali ini menampilkan *sequence diagram* untuk melakukan proses identifikasi varietas. Pengguna melakukan aktifitas pengunggahan citra daun jati ke aplikasi kemudian nantinya akan tampil pada aplikasi, setelah tampil maka akan dirubah menjadi rgb dan resize ukuran kemudian dapat nilai pixel setiap dimensi arraynya. Setelah mendapatkan nilai pixel maka akan dilakukan perubahan nilai-nilai menjadi vektor atau bisa disebut dengan proses vektorisasi. Setelah mendapatkan nilai vektornya maka akan masuk dalam CNN untuk diolah dengan model dan arsitektur yang sudah dimiliki untuk dibaca pola dari nilai-nilai vektor yang masuk yang berguna untuk memprediksi citra daun jati yang sedang diuji.

iv. *Class diagram*

Class Diagram merupakan diagram yang memberikan ilustrasi struktur dan deskripsi hubungan antara *class diagram* tersebut. Karakteristik dasar dari *Class Diagram* yaitu pada kumpulan kelas-kelas.



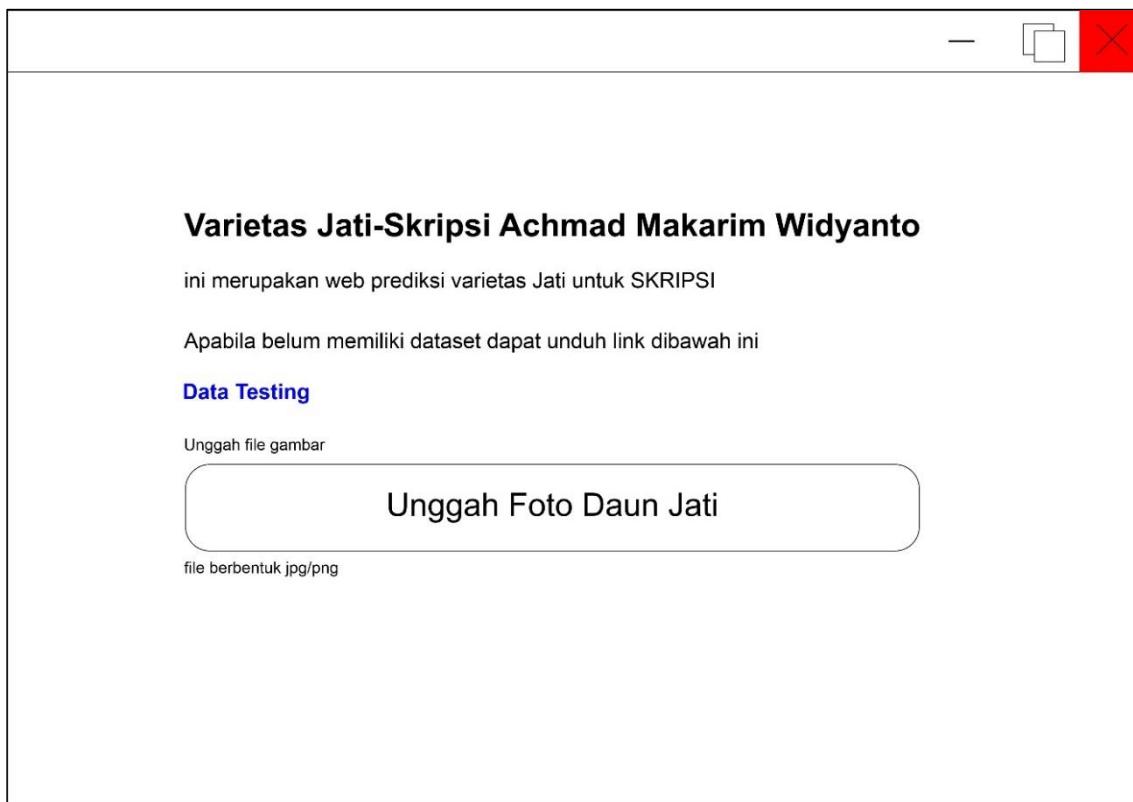
Gambar 3.41 *Sequance Diagram* Identifikasi Varietas Untuk Pengguna



Gambar 3.42 Class Diagram CNN

3. Perancangan Tampilan Antarmuka

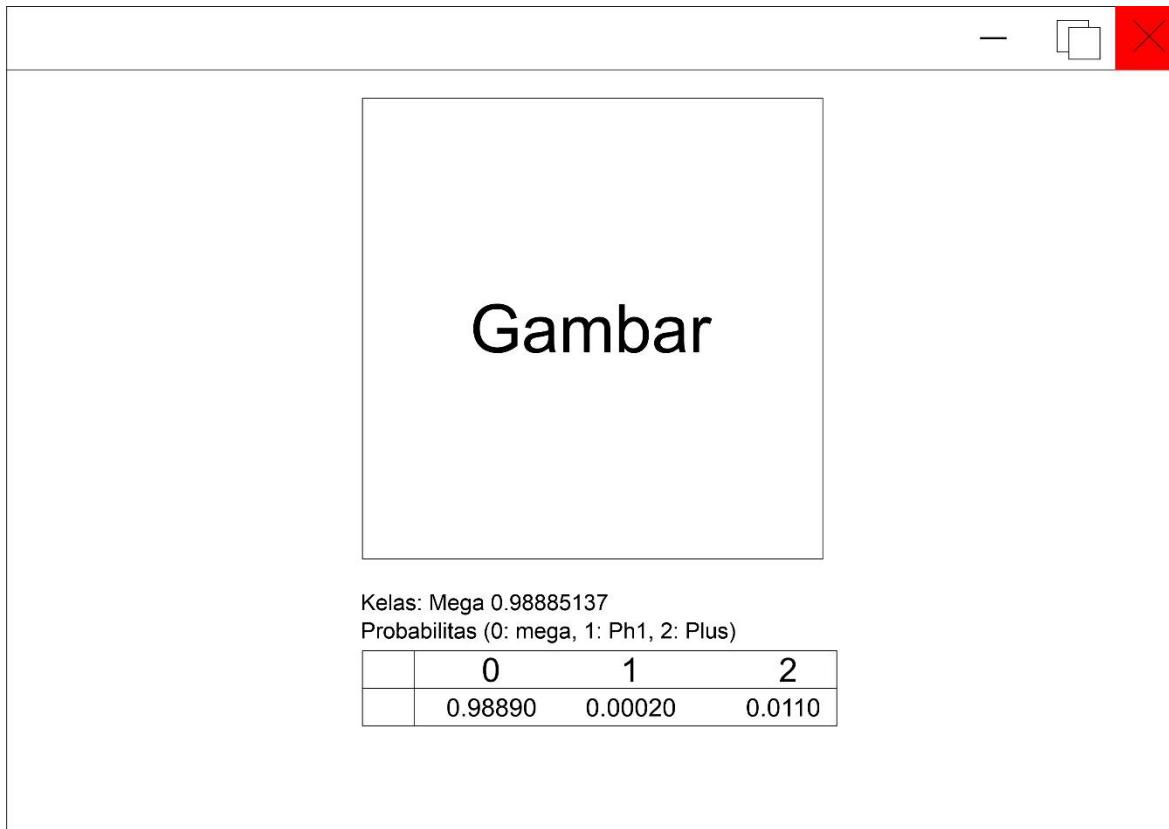
a. Antarmuka Beranda



Gambar 3.43 Tampilan Beranda

Rancangan antar muka beranda ini yang nantinya akan muncul ketika pengguna membuka pertama kali aplikasi web identifikasi varietas jatinya. Akan menampilkan judul, dan ada petunjuk jika belum memiliki data uji, jika belum memiliki data uji maka akan diarahkan pada penyimpanan *cloud* yaitu *google drive*, dan kemudian jika sudah mengunduh citra foto daun dapat memencet tombol untuk mengunggah citra foto daun yang akan dicaritau kelas varietas jatinya.

b. Antarmuka Hasil Kelas



Gambar 3.44 Tampilan Hasil Kelas

Pada rancangan antarmuka hasil kelas, akan menampilkan hasil dari masukan citra foto daun jati yang ingin diketahui kelas varietasnya, kemudian juga akan ditampilkan probabilitas dari masing-masing kelas varietas yang ada, dan juga hasil akhir kelas varietas jatinya.

4. Perancangan Pengujian

Tabel 3.8 Pengujian Black Box

NO	Halaman	Detail Pengujian	Pengujian	
			Berhasil	Gagal
1.	Menu Utama	Menampilkan daftar menu		
		Menuju link unduh data Uji		
		Berpindah ke unggah gambar		
2.	Unduh Data Uji	Menampilkan Google Drive		
3.	Unggah Gambar	Menampilkan sub menu unggah gambar		
		Menampilkan daftar gambar dari directory		
		Gambar dapat dipilih		
		Menampilkan gambar yang dipilih		

Tabel 3.9 Lanjutan Pengujian *Black Box*

4.	Hasil Klasifikasi	Menampilkan hasil klasifikasi		
		Menampilkan tingkat kemiripan		
		Menampilkan perbandingan tingkat kemiripan dari masing-masing kelas		

Tabel 3.10 Daftar Kelas Klasifikasi

No kelas	Kelas
0	Mega
1	PH1
2	Plus
3	Tidak Diketahui

Tabel 3.11 *Confusion Matrix Multi Class*

Kelas	0	1	2	3
0				
1				
2				
3				

Tabel 3.12 Hasil *Confusion Matrix Multi Class*

	Precision	Recall	F1-Score	Support
0				
1				
2				
3				
Accuracy				
Macro avg				
Weighted avg				

BAB IV

HASIL, PENGUJIAN, DAN PEMBAHASAN

Hasil, pengujian, dan pembahasan merupakan bab yang membahas mengenai sistem pada aplikasi ini bisa diterapkan pada sebenarnya. Isi dari bab ini juga membahas mengenai aplikasi ini dapat sesuai menghasilkan tujuan yang sudah diinginkan berdasarkan pada analisis dan perancangan yang sudah dibuat sebelum melakukan penelitian.

4.1 Hasil Penelitian

Hasil penelitian merupakan penerapan dari metode *convolutional neural network* untuk mempelajari pola citra daun varietas jati menjadi kelas sesuai yang telah ditentukan, nantinya digunakan sebagai model untuk bisa mengidentifikasi varietas jati berdasar daun melalui aplikasi web.

4.1.1 Implementasi Preprocessing

a. Inisiasi library training yang digunakan

```
import os
import time
import datetime
import tensorflow as tf
import matplotlib.pyplot as plt
from google.colab import drive
from tensorflow.keras import optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Modul Program 4.1 Inisiasi Library

Penggunaan library dalam pembuatan model untuk mempermudah dan mempercepat dalam penyusunan program. Isi dari *library* sendiri terdiri dari beberapa *package* yang berbeda sesuai dengan fungsi penggunaannya. *Library* yang digunakan diantaranya ada *os* untuk mengakses directory lokal, kemudian *time* untuk mengambil waktu yang ada pada sistem, *tensorflow* untuk pengolahan pembuatan model dengan menggunakan *cnn*, *google colab* dengan *package* *drive* untuk dapat mengakses *google drive*, *keras package* *optimizers* untuk penggunaan optimasi pembelajaran, dan yang terakhir *ImageDataGenerator* untuk melakukan *preprocessing* perbanyak data yang dimiliki.

b. Membaca lokasi dataset

```
drive.mount('/content/drive')
base_dir = '/content/drive/My Drive/Jati_v3'
print(os.listdir(base_dir))
```

Modul Program 4.2 Membaca Lokasi Dataset

Lokasi penyimpanan dataset yang berada di google drive perlu diketahui terlebih dahulu keberadaannya, untuk bisa mengetahui lokasi keberadaannya atau sistem dapat membaca lokasinya maka perlu dikenali atau ditulis dengan menggunakan library google colab *package* drive secara *default* supaya dapat langsung mengarah pada lokasi, dan jika berhasil diketahui lokasinya maka akan dibaca isi dari lokasi tersebut dengan menggunakan library os.

c. Perbanyakkan dataset dan pembagian validasi

```
train_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 20,
    validation_split=0.3,
    horizontal_flip = True,
    vertical_flip = True,
    fill_mode = 'nearest'
)

train_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size = (256, 256),
    batch_size = 16,
    subset = 'training',
    class_mode = 'categorical'
)

validation_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size = (256, 256),
    batch_size = 16,
    subset = 'validation',
    class_mode = 'categorical'
)
```

Modul Program 4.3 Perbanyakkan Dataset dan Pembagian Validasi

Memvariasi data untuk memperkaya informasi yang ada pada dataset dapat dilakukan dengan cara mengaugmentasi gambar menjadi beberapa macam, pada program ini dengan cara merotasi setiap 20 derajat, dibalik secara horizal dan vertika, dan fill mode secara default adalah nearest: aaaaaaaaa|abcd|ddddddddd.

Langkah *preprocessing* yang dilakukan hanya memberikan patokan ukuran data inputnya menjadi 256 pixel x 256 pixel, lalu mengatur banyaknya kelompok ketika proses pembelajaran berlangsung, dan memberikan tanda sebagai data latih atau sebagai data validasi.

c. Menampilkan sample data pelatihan(*training*)

```
sample_training_images, _ = next(train_generator)
```

Modul Program 4.4 Menampilkan Sample Data Latih

Apabila ingin membaca beberapa contoh gambar isi yang dijadikan sebagai data latih dapat menggunakan fungsi *next* dan langsung memanggil variabel pembentukan data augmentasi.

```
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 5, figsize=(256,256))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()
```

Modul Program 4.5 Prosedur Menampilkan Gambar Dataset

Apabila sebelumnya program untuk membaca data, kali ini untuk menampilkan data-data atau untuk memvisualisasi gambar yang telah dibaca tadi. Menggunakan library matplotlib akan menjadi lebih mudah dalam menampilkannya. Deklarasikan terlebih dahulu banyaknya foto yang akan muncul, lalu atur ukuran gambarnya kemudian tampilkan datanya secara mendatar, kemudian hilangkan sumbu x dan y, dan terakhir mengeksekusi atau memvisualisasikan semua perintah yang sudah ditulis.

```
plotImages(sample_training_images[:5])
```

Modul Program 4.6 Pemanggilan Prosedur Visualisasi Data

Setelah dideklarasikan atau dikenali maka langkah selanjutnya dapat memanggil prosedur yang sudah dibuat tadi dengan merincikan banyaknya nilai data yang ingin ditampilkan, dengan contoh mengisi nilainya adalah 5 maka nantinya akan ditampilkan 5

gambar yang ada di *sample* data latih. Setelah ini merupakan beberapa gambar contoh data latih yang dapat ditampilkan.



Gambar 4.1 Contoh Data Latih

```
augmented_images = [train_generator[0][0][0] for i in range(5)]
plotImages(augmented_images)
```

Modul Program 4.7 Augmentasi Gambar

Apabila program sebelumnya merupakan untuk menampilkan data latih yang dimiliki, kali ini akan menampilkan data yang telah diaugmentasi, deklarasikan terlebih dahulu variabel data augmentasi lalu berikan nilai banyaknya gambar yang akan ingin ditampilkan dengan contoh disini adalah lima gambar. Dibawah ini merupakan data contoh gambar yang sudah teraumentasi.



Gambar 4.2 Contoh Augmentasi Gambar

d. Menampilkan sample data validasi(*validation*)

```
sample_validasi_images, _ = next(validation_generator)
```

Modul Program 4.8 Menampilkan Sample Data Validasi

Kali ini akan menampilkan data-data atau gambar yang digunakan untuk validasi, dengan menggunakan fungsi yaitu next.

```
plotImages(sample_validasi_images[5:])
```

Modul Program 4.9 Pemanggilan Prosedur Visualisasi Data

Setelah dideklarasikan atau dikenali maka langkah selanjutnya dapat memanggil prosedur yang sudah dibuat tadi dengan merincikan banyaknya nilai data validasi yang ingin

ditampilkan, dengan contoh mengisi nilainya adalah 5 maka nantinya akan ditampilkan 5 gambar yang ada di *sample* data latih. Setelah ini merupakan beberapa gambar contoh data validasi yang dapat ditampilkan. Dibawah ini merupakan contoh data validasi yang digunakan.



Gambar 4.3 Contoh Data Validasi

4.1.2 Implementasi Perancangan Arsitektur

a. Arsitektur CNN 1

```
model = tf.keras.models.Sequential([
    #konvolusi pertama
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(256, 256, 3)),
    #pooling pertama
    tf.keras.layers.MaxPooling2D(2, 2),
    #perataan
    tf.keras.layers.Flatten(),
    #keluaran
    tf.keras.layers.Dense(3, activation='softmax')
])
```

Modul Program 4.10 Arsitektur Pertama

Dalam pembuatan model memerlukan beberapa kali percobaan untuk mencaritau arsitektur cnn seperti apa yang cocok untuk digunakan mempelajari pola gambar dataset daun yang dimiliki. Ini merupakan arsitektur cnn pertama yaitu dengan menggunakan library tensorflow untuk penggunaan cnnya dengan rincian yaitu satu lapisan konvolusi, satu lapisan *pooling*, dan satu lapisan *fully connected*. Ukuran gambar inputannya sebesar 256 pixel dikali dengan 256 pixel dengan 3 lapisan merah, hijau, dan biru(RGB).

Gambar berwarna yang masuk akan diambil fitur-fiturnya dengan cara mengalikan(konvolusi) nilai setiap pixel yang ada dengan nilai filter yaitu berukuran 3x3 dengan jumlah 32 varian. Hasil dari proses konvolusi tersebut akan diseleksi terlebih dahulu

menggunakan fungsi aktivasi relu, apabila nilai dari masing-masing pixel bernilai kurang dari 0 maka nilai-nilai tersebut nantinya akan dirubah menjadi nol, dan jika masing-masing nilai setiap pixelnya sudah lebih dari nol maka akan tetap menjadi nilai itu sendiri.

Proses berikutnya yaitu pengambilan nilai tertentu pada setiap kelompok-kelompok(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan ukuran filter yang berukuran 2x2 pixel, dari pengambilan nilai tersebut nantinya akan menghasilkan nilai-nilai baru atau bisa disebut dengan *featured map*.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
flatten (Flatten)	(None, 516128)	0
dense (Dense)	(None, 3)	1548387
<hr/>		
Total params: 1,549,283		
Trainable params: 1,549,283		
Non-trainable params: 0		

Gambar 4.4 Summary Arsitektur Pertama

Setelah menyusun arsitektur pertama maka akan diperoleh banyaknya jumlah parameter yang digunakan. Dengan data awal berukuran 256 pixel kali 256 pixel dan dengan tiga lapisan yaitu merah,hijau,biru(RGB) diambil fiturnya dengan konvolusi berubah ukuran matriksnya menjadi 254 piksel kali 254 piksel kemudian jumlah parameter pada konvolusi pertama sebanyak 896, berikutnya disederhanakan tanpa menghilangkan isi informasi dari gambar dengan menggunakan pooling ukuran filter yaitu 2x2 maka ukurannya berubah menjadi 127 pixel kali 127 pixel.

Sebelum masuk dalam layer *fully connected* atau layer keluaran data-data yang masih berbentuk matriks n kali n dirubah menjadi n kali 1 atau bisa disebut proses perataan maka menjadi rata dengan banyak parameter menjadi 516.128. Dan pada layer terakhir atau layer untuk keluaran yaitu menjadi 3 bagian sesuai banyaknya kelas yang diinginkan.

b. Arsitektur CNN 2

```
model = tf.keras.models.Sequential([
    #konvolusi pertama
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(256, 256, 3)),
    #pooling pertama
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi kedua
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
input_shape=(256, 256, 3)),
    #pooling kedua
    tf.keras.layers.MaxPooling2D(2, 2),

    #perataan
    tf.keras.layers.Flatten(),
    #keluaran
    tf.keras.layers.Dense(3, activation='softmax')
])
```

Modul Program 4.11 Arsitektur Kedua

Selanjutnya merupakan arsitektur cnn kedua yaitu penggunaan library tensorflow yang masih sama seperti arsitektur pertama untuk penggunaan cnnya, pada arsitektur ini dengan rincian yaitu dua lapisan konvolusi, dua lapisan *pooling*, dan satu lapisan *fully connected*. Ukuran gambar harus sama sisi yaitu ukuran inputannya berukuran 256x256 pixel dengan 3 lapisan bawaan yaitu lapisan merah, hijau, dan biru(RGB).

Gambar berwarna yang masuk akan diambil fitur-fiturnya dengan cara mengalikan(kovolusi) nilai setiap pixel yang ada dengan nilai filter yaitu berukuran 3x3 dengan jumlah 32 varian. Hasil dari proses konvolusi tersebut akan diseleksi terlebih dahulu menggunakan fungsi aktivasi relu, apabila nilai dari masing-masing pixel bernilai kurang dari 0 maka nilai-nilai tersebut nantinya akan dirubah menjadi nol, dan jika masing-masing nilai setiap pixelnya sudah lebih dari nol maka akan tetap menjadi nilai itu sendiri.

Proses pengambilan fitur berikutnya yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan nilai-nilai baru atau bisa disebut dengan *featured map* yang nantinya akan digunakan untuk pengambilan fitur kembali pada lapisan berikutnya.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 64 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 64 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan nilai-nilai baru atau bisa disebut dengan *featured map*.

Dari hasil pengambilan fitur yang sudah dilakukan, maka berikutnya adalah meratakan matriks yang masih berukuran n kali n menjadi matriks berukuran n kali 1. Dan yang paling terakhir adalah *layer* dimana *layer* sebagai keluaran, yaitu sebanyak 3 kelas.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_1 (MaxPooling2 (None, 62, 62, 64)		0
flatten (Flatten)	(None, 246016)	0
dense (Dense)	(None, 3)	738051
<hr/>		
Total params: 757,443		
Trainable params: 757,443		
Non-trainable params: 0		

Gambar 4.5 Summary Arsitektur Kedua

Jumlah paramater pada arsitektur kedua cenderung lebih sedikit jika dibandingkan dengan arsitektur yang pertama, mengapa demikian karena jumlah lapisan pada arsitektur kedua ini lebih banyak daripada arsitektur pertama yang menyebabkan bertambahnya jumlah penyaring untuk mengambil fitur.

Dengan data awal berukuran 256 pixel kali 256 pixel dan dengan tiga lapisan yaitu merah,hijau,biru(RGB) diambil fiturnya dengan konvolusi berubah ukuran matriksnya menjadi 254 piksel kali 254 piksel kemudian jumlah parameter pada konvolusi pertama sebanyak 896, berikutnya disederhanakan tanpa menghilangkan isi informasi dari gambar dengan menggunakan pooling ukuran filter yaitu 2x2 maka ukurannya berubah menjadi 127 pixel kali 127 pixel.

Bertambah satu lapisan konvolusi maka hasil dari *pooling layer* dijadikan data masukan kembali untuk pengambilan fitur, yang mana ukuran awalnya adalah 127x127 pixel pada konvolusi kedua dikalikan dengan filter 3x3 sebanyak 64 maka akan menjadi 125x125 pixel sebanyak 64 varian juga dengan nilai parameternya yaitu 18.496.

Setelah dari konvolusi akan masuk kembali pada *pooling layer* dengan matriks masukan berukuran 125x125 pixel menjadi matriks berukuran 62x62 pixel dengan banyak varian masih sama yaitu 64 varian.

Sebelum masuk dalam layer *fully connected* atau layer keluaran data-data yang masih berbentuk matriks n kali n dirubah menjadi n kali 1 atau bisa disebut proses perataan maka nilai masukan awalnya sebanyak 64 varian akan menjadi 1 saja dengan banyak parameter yaitu menjadi 246.016. Dan pada layer terakhir atau layer untuk keluaran yaitu menjadi 3 bagian sesuai banyaknya kelas yang diinginkan.

c. Arsitektur CNN 3

```
model = tf.keras.models.Sequential([
    #konvolusi pertama
    tf.keras.layers.Conv2D(32,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling pertama
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi kedua
    tf.keras.layers.Conv2D(64,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling kedua
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi ketiga
```

```

tf.keras.layers.Conv2D(128,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
#pooling ketiga
tf.keras.layers.MaxPooling2D(2, 2),

#perataan
tf.keras.layers.Flatten(),
#keluaran
tf.keras.layers.Dense(3, activation='softmax')
])

```

Modul Program 4.13 Lanjutan Arsitektur Ketiga

Selanjutnya merupakan arsitektur cnn ketiga yaitu penggunaan library tensorflow yang masih sama seperti arsitektur pertama untuk penggunaan cnannya, pada arsitektur ini dengan rincian yaitu tiga lapisan konvolusi, tiga lapisan *pooling*, dan satu lapisan *fully connected*. Ukuran gambar harus sama sisi yaitu ukuran inputanya berukuran 256x256 pixel dengan 3 lapisan bawaan yaitu lapisan merah, hijau, dan biru(RGB).

Gambar berwarna yang masuk akan diambil fitur-fiturnya dengan cara mengalikan(konvolusi) nilai setiap pixel yang ada dengan nilai filter yaitu berukuran 3x3 dengan jumlah 32 varian. Hasil dari proses konvolusi tersebut akan diseleksi terlebih dahulu menggunakan fungsi aktivasi relu, apabila nilai dari masing-masing pixel bernilai kurang dari 0 maka nilai-nilai tersebut nantinya akan dirubah menjadi nol, dan jika masing-masing nilai setiap pixelnya sudah lebih dari nol maka akan tetap menjadi nilai itu sendiri.

Proses pengambilan fitur berikutnya yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan nilai-nilai baru atau bisa disebut dengan *featured map* yang nantinya akan digunakan untuk pengambilan fitur kembali pada lapisan berikutnya.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 64 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada

nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 64 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 128 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 128 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru lagi.

Dari hasil pengambilan fitur yang sudah dilakukan, maka berikutnya adalah meratakan matriks yang masih berukuran n kali n menjadi matriks berukuran n kali 1. Dan yang paling terakhir adalah *layer* dimana *layer* sebagai keluaran, yaitu sebanyak 3 kelas.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten (Flatten)	(None, 115200)	0
dense (Dense)	(None, 3)	345603
<hr/>		
Total params: 438,851		
Trainable params: 438,851		
Non-trainable params: 0		

Gambar 4.6 Summary Arsitektur Ketiga

Jumlah paramater pada arsitektur ketiga akan cenderung lebih sedikit jika dibandingkan dengan arsitektur yang pertama dan kedua, mengapa demikian karena jumlah lapisan pada arsitektur ketiga ini lebih banyak daripada arsitektur pertama dan kedua yang menyebabkan bertambahnya jumlah penyaring untuk mengambil fitur.

Dengan data awal berukuran 256 pixel kali 256 pixel dan dengan tiga lapisan yaitu merah,hijau,biru(RGB) diambil fiturnya dengan konvolusi berubah ukuran matriksnya menjadi 254 piksel kali 254 piksel kemudian jumlah parameter pada konvolusi pertama sebanyak 896, berikutnya disederhanakan tanpa menghilangkan isi informasi dari gambar dengan menggunakan pooling ukuran filter yaitu 2x2 maka ukurannya berubah menjadi 127 pixel kali 127 pixel.

Bertambah satu lapisan konvolusi maka hasil dari *pooling layer* dijadikan data masukan kembali untuk pengambilan fitur, yang mana ukuran awalnya adalah 127x127 pixel pada konvolusi kedua dikalikan dengan filter 3x3 sebanyak 64 maka akan menjadi 125x125 pixel sebanyak 64 varian juga dengan nilai parameternya yaitu 18.496.

Setelah dari konvolusi akan masuk kembali pada *pooling layer* dengan matriks masukan berukuran 125x125 pixel menjadi matriks berukuran 62x62 pixel dengan banyak varian masih sama yaitu 64 varian.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 62x62 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 128 varian maka akan menjadi 60x60 pixel sebanyak 128 varian dengan jumlah parameter sebanyak 73.856.

Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 60x60 pixel menjadi matriks berukuran 30x30 pixel.

Sebelum masuk dalam layer *fully connected* atau layer keluaran data-data yang masih berbentuk matriks n kali n dirubah menjadi n kali 1 atau bisa disebut proses perataan maka nilai masukan awalnya sebanyak 128 varian akan menjadi 1 saja dengan banyak parameter yaitu menjadi 438.851. Dan pada layer terakhir atau layer untuk keluaran yaitu menjadi 3 bagian sesuai banyaknya kelas yang diinginkan

d. Arsitektur CNN 4

```

        #konvolusi pertama dan pooling pertama
        tf.keras.layers.Conv2D(32, (3,3), activation='relu',
input_shape=(256, 256, 3)),
        tf.keras.layers.MaxPooling2D(2, 2),
        #konvolusi kedua dan pooling kedua
        tf.keras.layers.Conv2D(64,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
        tf.keras.layers.MaxPooling2D(2, 2),
        #konvolusi ketiga
        tf.keras.layers.Conv2D(128,     (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
        #pooling ketiga
        tf.keras.layers.MaxPooling2D(2, 2),
        #konvolusi keempat
        tf.keras.layers.Conv2D(256,     (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
        #pooling keempat
        tf.keras.layers.MaxPooling2D(2, 2),
        #perataan
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(3, activation='softmax')
    )

```

Selanjutnya merupakan arsitektur cnn keempat yaitu penggunaan library tensorflow yang masih sama seperti arsitektur sebelum-sebelumnya untuk penggunaan cnnnya, pada arsitektur ini dengan rincian yaitu empat lapisan konvolusi, empat lapisan *pooling*, dan satu lapisan *fully connected*. Ukuran gambar harus sama sisi yaitu ukuran inputannya berukuran 256x256 pixel dengan 3 lapisan bawaan yaitu lapisan merah, hijau, dan biru(RGB).

Gambar berwarna yang masuk akan diambil fitur-fiturnya dengan cara mengalikan(konvolusi) nilai setiap pixel yang ada dengan nilai filter yaitu berukuran 3x3 dengan jumlah 32 varian. Hasil dari proses konvolusi tersebut akan diseleksi terlebih dahulu menggunakan fungsi aktivasi *relu*, apabila nilai dari masing-masing pixel bernilai kurang dari 0 maka nilai-nilai tersebut nantinya akan dirubah menjadi nol, dan jika masing-masing nilai setiap pixelnya sudah lebih dari nol maka akan tetap menjadi nilai itu sendiri.

Proses pengambilan fitur berikutnya yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan nilai-nilai baru atau bisa disebut dengan *featured map* yang nantinya akan digunakan untuk pengambilan fitur kembali pada lapisan berikutnya.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 64 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 64 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya hasil dari pengambilan

nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 128 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 128 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 256 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 256 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Dari hasil pengambilan nilai-nilai fitur yang sudah dilakukan, maka langkah berikutnya adalah meratakan ukuran matriks menjadi satu dimensi yang masih berukuran n kali n menjadi matriks berukuran n kali 1. Dan yang paling terakhir adalah *layer* dimana *layer* sebagai keluaran, yaitu sebanyak 3 kelas.

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 256)	295168
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 3)	150531
<hr/>		
Total params: 538,947		
Trainable params: 538,947		
Non-trainable params: 0		

Gambar 4.7 Summary Arsitektur Keempat

Jumlah parameter pada arsitektur keempat akan cenderung lebih sedikit jika dibandingkan dengan arsitektur yang pertama dan kedua namun lebih banyak dari arsitektur ketiga, mengapa demikian karena jumlah lapisan pada arsitektur keempat ini lebih banyak daripada arsitektur pertama dan kedua yang menyebabkan bertambahnya jumlah penyaring untuk mengambil fitur.

Dengan data awal berukuran 256 pixel kali 256 pixel dan dengan tiga lapisan yaitu merah,hijau,biru(RGB) diambil fiturnya dengan konvolusi berubah ukuran matriksnya menjadi 254 piksel kali 254 piksel kemudian jumlah parameter pada konvolusi pertama sebanyak 896, berikutnya disederhanakan tanpa menghilangkan isi informasi dari gambar dengan menggunakan pooling ukuran filter yaitu 2x2 maka ukurannya berubah menjadi 127 pixel kali 127 pixel.

Bertambah satu lapisan konvolusi maka hasil dari *pooling layer* dijadikan data masukan kembali untuk pengambilan fitur, yang mana ukuran awalnya adalah 127x127 pixel

pada konvolusi kedua dikalikan dengan filter 3x3 sebanyak 64 maka akan menjadi 125x125 pixel sebanyak 64 varian juga dengan nilai parameternya yaitu 18.496.

Setelah dari konvolusi akan masuk kembali pada *pooling layer* dengan matriks masukan berukuran 125x125 pixel menjadi matriks berukuran 62x62 pixel dengan banyak varian masih sama yaitu 64 varian.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 62x62 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 128 varian maka akan menjadi 60x60 pixel sebanyak 128 varian dengan jumlah parameter sebanyak 73.856.

Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 60x60 pixel menjadi matriks berukuran 30x30 pixel.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 30x30 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 256 varian maka akan menjadi 28x28 pixel sebanyak 256 varian dengan jumlah parameter sebanyak 295.168.

Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 28x28 pixel menjadi matriks berukuran 14x14 pixel.

Sebelum masuk dalam layer *fully connected* atau layer keluaran data-data yang masih berbentuk matriks n kali n dirubah menjadi n kali 1 atau bisa disebut proses perataan maka nilai masukan awalnya sebanyak 256 varian akan menjadi 1 saja dengan banyak parameter yaitu menjadi 548947. Dan pada layer terakhir atau layer untuk keluaran yaitu menjadi 3 bagian sesuai banyaknya kelas yang diinginkan

e. Arsitektur CNN 5

```
#konvolusi pertama
    tf.keras.layers.Conv2D(32,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
```

```

#pooling pertama
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi kedua
    tf.keras.layers.Conv2D(64,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling kedua
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi ketiga
    tf.keras.layers.Conv2D(128,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling ketiga
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi keempat
    tf.keras.layers.Conv2D(256,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling keempat
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi kelima
    tf.keras.layers.Conv2D(512,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling kelima
    tf.keras.layers.MaxPooling2D(2, 2),

    #perataan
    tf.keras.layers.Flatten(),
    #keluaran
    tf.keras.layers.Dense(3, activation='softmax')
)

```

Modul Program 4.16 Lanjutan Arsitektur Kelima

Selanjutnya merupakan arsitektur cnn kelima yaitu penggunaan library tensorflow yang masih sama seperti arsitektur sebelum-sebelumnya untuk penerapan pembuatan arsitektur cnnya, pada arsitektur ini dengan rincian yaitu lima lapisan konvolusi, lima lapisan *pooling*, dan satu lapisan *fully connected*. Ukuran gambar harus sama sisi yaitu ukuran inputannya berukuran 256x256 pixel dengan 3 lapisan bawaan yaitu lapisan merah, hijau, dan biru(RGB).

Gambar berwarna yang masuk akan diambil fitur-fiturnya dengan cara mengalikan(kovolusi) nilai setiap pixel yang ada dengan nilai filter yaitu berukuran 3x3 dengan jumlah 32 varian. Hasil dari proses konvolusi tersebut akan diseleksi terlebih dahulu menggunakan fungsi aktivasi relu, apabila nilai dari masing-masing pixel bernilai kurang dari 0 maka nilai-nilai tersebut nantinya akan dirubah menjadi nol, dan jika masing-masing nilai setiap pixelnya sudah lebih dari nol maka akan tetap menjadi nilai itu sendiri.

Proses pengambilan fitur berikutnya yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan nilai-nilai baru atau bisa disebut dengan *featured map* yang nantinya akan digunakan untuk pengambilan fitur kembali pada lapisan berikutnya.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 64 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 64 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 128 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 128 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 256 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 256 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 512 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 512 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Dari hasil pengambilan fitur yang sudah dilakukan, maka berikutnya adalah meratakan matriks yang masih berukuran $n \times n$ menjadi matriks berukuran $n \times 1$. Dan yang paling terakhir adalah *layer* dimana *layer* sebagai keluaran, yaitu sebanyak 3 kelas.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 256)	295168
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
conv2d_4 (Conv2D)	(None, 12, 12, 512)	1180160
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 512)	0
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 3)	55299
<hr/>		
Total params: 1,623,875		
Trainable params: 1,623,875		
Non-trainable params: 0		

Gambar 4.8 Summary Arsitektur Kelima

Jumlah paramater pada arsitektur kelima cenderung lebih banyak jika dibandingkan dengan arsitektur sebelum-sebelumnya, mengapa demikian karena jumlah lapisan pada arsitektur kelima ini menjadi lebih banyak daripada arsitektur sebelumnya namun pada lapisan terakhir konvolusi variasi filter menjadi bertambah maka jumlah parameter jauh lebih banyak untuk pengambilan fitur-fiturnya.

Dengan data awal berukuran 256 pixel kali 256 pixel dan dengan tiga lapisan yaitu merah,hijau,biru(RGB) diambil fiturnya dengan konvolusi berubah ukuran matriksnya menjadi 254 piksel kali 254 piksel kemudian jumlah parameter pada konvolusi pertama sebanyak 896, berikutnya disederhanakan tanpa menghilangkan isi informasi dari gambar dengan menggunakan pooling ukuran filter yaitu 2x2 maka ukurannya berubah menjadi 127 pixel kali 127 pixel.

Bertambah satu lapisan konvolusi maka hasil dari *pooling layer* dijadikan data masukan kembali untuk pengambilan fitur, yang mana ukuran awalnya adalah 127x127 pixel pada konvolusi kedua dikalikan dengan filter 3x3 sebanyak 64 maka akan menjadi 125x125 pixel sebanyak 64 varian juga dengan nilai parameternya yaitu 18.496.

Setelah dari konvolusi akan masuk kembali pada *pooling layer* dengan matriks masukan berukuran 125x125 pixel menjadi matriks berukuran 62x62 pixel dengan banyak varian masih sama yaitu 64 varian.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 62x62 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 128 varian maka akan menjadi 60x60 pixel sebanyak 128 varian dengan jumlah parameter sebanyak 73.856.

Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 60x60 pixel menjadi matriks berukuran 30x30 pixel.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 30x30 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 256 varian maka akan menjadi 28x28 pixel sebanyak 256 varian dengan jumlah parameter sebanyak 295.168. Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 28x28 pixel menjadi matriks berukuran 14x14 pixel.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 14x14 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 512 varian maka akan menjadi 12x12 pixel sebanyak 512 varian dengan jumlah parameter sebanyak 1.180.160. Pengambilan fitur belum juga selesai, akan berlanjut lagi ke tahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 28x28 pixel menjadi matriks berukuran 14x14 pixel.

Sebelum masuk dalam layer *fully connected* atau layer keluaran data-data yang masih berbentuk matriks n kali n dirubah menjadi n kali 1 atau bisa disebut proses perataan maka nilai masukan awalnya sebanyak 512 varian akan menjadi 1 saja dengan banyak parameter yaitu menjadi 1.623.875. Dan pada layer terakhir atau layer untuk keluaran yaitu menjadi 3 bagian sesuai banyaknya kelas yang diinginkan.

f. Arsitektur CNN 6

```
#konvolusi pertama
    tf.keras.layers.Conv2D(32,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling pertama
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi kedua
    tf.keras.layers.Conv2D(64,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling kedua
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi ketiga
    tf.keras.layers.Conv2D(128,     (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling ketiga
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi keempat
    tf.keras.layers.Conv2D(256,     (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling keempat
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi kelima
    tf.keras.layers.Conv2D(512,     (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling kelima
    tf.keras.layers.MaxPooling2D(2, 2),

    #perataan
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(32, activation='relu'),
    #keluaran
    tf.keras.layers.Dense(3, activation='softmax')
])
```

Modul Program 4.17 Arsitektur Keenam

Selanjutnya merupakan arsitektur cnn keenam yaitu penggunaan library tensorflow yang masih sama seperti arsitektur sebelum-sebelumnya untuk penggunaan cnnnya, pada arsitektur ini dengan rincian yaitu lima lapisan konvolusi, lima lapisan *pooling*, dan dua lapisan *fully connected*. Ukuran gambar harus sama sisi yaitu ukuran inputannya berukuran 256x256 pixel dengan 3 lapisan bawaan yaitu lapisan merah, hijau, dan biru(RGB).

Gambar berwarna yang masuk akan diambil fitur-fiturnya dengan cara mengalikan(kovolusi) nilai setiap pixel yang ada dengan nilai filter yaitu berukuran 3x3 dengan jumlah 32 varian. Hasil dari proses konvolusi tersebut akan diseleksi terlebih dahulu menggunakan fungsi aktivasi *relu*, apabila nilai dari masing-masing pixel bernilai kurang dari 0 maka nilai-nilai tersebut nantinya akan dirubah menjadi nol, dan jika masing-masing nilai setiap pixelnya sudah lebih dari nol maka akan tetap menjadi nilai itu sendiri.

Proses pengambilan fitur berikutnya yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan nilai-nilai baru atau bisa disebut dengan *featured map* yang nantinya akan digunakan untuk pengambilan fitur kembali pada lapisan berikutnya.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 64 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter sebanyak 64 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali pengambilan fiturnya menggunakan lapisan konvolusi dengan banyak vilter sebanyak 128 filter yang berukuran 3x3, dari hasil tersebut akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilai-nilai itu akan dirubah secara permanen menjadi nol

dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 128 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 256 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 256 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 512 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 512 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan

berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Dari hasil pengambilan fitur yang sudah dilakukan, maka berikutnya adalah meratakan matriks yang masih berukuran $n \times n$ menjadi matriks berukuran $n \times 1$. Apabila pada arsitektur sebelum-sebelumnya langsung berujung pada *layer* keluaran, maka pada arsitektur kali ini akan masuk 1 *hidden layer* dengan bantuan 32 *neurons* yang kemudian terakhir adalah *layer* dimana *layer* sebagai keluaran, yaitu sebanyak 3 kelas.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 256)	295168
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
conv2d_4 (Conv2D)	(None, 12, 12, 512)	1180160
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 512)	0
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 32)	589856
dense_1 (Dense)	(None, 3)	99
<hr/>		
Total params: 2,158,531		
Trainable params: 2,158,531		
Non-trainable params: 0		

Gambar 4.9 Summary Arsitektur Keenam

Jumlah parameter pada arsitektur keenam cenderung lebih banyak jika dibandingkan dengan arsitektur sebelumnya. Banyak lapisan pengambilan fitur tidak menyebabkan berkurangnya jumlah parameter akhir yang ada, hal ini disebabkan karena satu lapisan akhir

yang menggunakan bantuan perbanyak jumlah neurons sebanyak 1024 sebelum *layer* keluaran menjadi 3 kelas.

Dengan data awal berukuran 256 pixel kali 256 pixel dan dengan tiga lapisan yaitu merah,hijau,biru(RGB) diambil fiturnya dengan konvolusi berubah ukuran matriksnya menjadi 254 piksel kali 254 piksel kemudian jumlah parameter pada konvolusi pertama sebanyak 896, berikutnya disederhanakan tanpa menghilangkan isi informasi dari gambar dengan menggunakan pooling filter yaitu 2x2 maka ukurannya berubah menjadi 127 pixel kali 127 pixel.

Bertambah satu lapisan konvolusi maka hasil dari *pooling layer* dijadikan data masukan kembali untuk pengambilan fitur, yang mana ukuran awalnya adalah 127x127 pixel pada konvolusi kedua dikalikan dengan filter 3x3 sebanyak 64 maka akan menjadi 125x125 pixel sebanyak 64 varian juga dengan nilai parameternya yaitu 18.496.

Setelah dari konvolusi akan masuk kembali pada *pooling layer* dengan matriks masukan berukuran 125x125 pixel menjadi matriks berukuran 62x62 pixel dengan banyak varian masih sama yaitu 64 varian.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 62x62 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 128 varian maka akan menjadi 60x60 pixel sebanyak 128 varian dengan jumlah parameter sebanyak 73.856.

Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 60x60 pixel menjadi matriks berukuran 30x30 pixel.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 30x30 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 256 varian maka akan menjadi 28x28 pixel sebanyak 256 varian dengan jumlah parameter sebanyak 295.168. Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling layer* dengan cara

mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 28x28 pixel menjadi matriks berukuran 14x14 pixel.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 14x14 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 512 varian maka akan menjadi 12x12 pixel sebanyak 512 varian dengan jumlah parameter sebanyak 1.180.160. Pengambilan fitur belum juga selesai, akan berlanjut lagi ke tahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 28x28 pixel menjadi matriks berukuran 14x14 pixel.

Sebelum masuk dalam layer *fully connected* atau layer keluaran data-data yang masih berbentuk matriks n kali n dirubah menjadi n kali 1 atau bisa disebut proses perataan maka nilai masukan awalnya sebanyak 512 varian akan menjadi 1 saja dengan banyak parameter yaitu menjadi 1.623.875.

Dan pada layer *fully connected* ada satu *hidden layer* dengan bantuan *neuron* sebanyak 32 untuk membantu mengecilkan terlebih dahulu sebelum masuk *layer* terakhir atau layer untuk keluaran yaitu menjadi 3 bagian sesuai banyaknya kelas yang diinginkan. Dengan demikian jumlah parameter secara keseluruhan pada arsitektur keenam ini adalah 2.158.531.

g. Arsitektur CNN 7

```
#konvolusi pertama dan pooling pertama
tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(256, 256, 3)),
tf.keras.layers.MaxPooling2D(2, 2),
#konvolusi kedua
tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
input_shape=(256, 256, 3)),
#pooling kedua
tf.keras.layers.MaxPooling2D(2, 2),
#konvolusi ketiga
tf.keras.layers.Conv2D(128, (3, 3), activation='relu',
input_shape=(256, 256, 3)),
#pooling ketiga
tf.keras.layers.MaxPooling2D(2, 2),
#konvolusi keempat
tf.keras.layers.Conv2D(256, (3, 3), activation='relu',
input_shape=(256, 256, 3)),
```

```

    #pooling keempat
        tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi kelima
        tf.keras.layers.Conv2D(512, (3, 3), activation='relu',
input_shape=(256, 256, 3)),
            #pooling kelima
            tf.keras.layers.MaxPooling2D(2, 2),

            #perataan
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(1024, activation='relu'),
            #keluaran
            tf.keras.layers.Dense(3, activation='softmax')
    ]
)

```

Modul Program 4.19 Lanjutan Arsitektur Ketujuh

Selanjutnya merupakan arsitektur cnn ketujuh yaitu penggunaan library tensorflow yang masih sama seperti arsitektur sebelum-sebelumnya untuk penggunaan cnnnya, pada arsitektur ini dengan rincian yaitu lima lapisan konvolusi, lima lapisan *pooling*, dan dua lapisan *fully connected*. Ukuran gambar harus sama sisi yaitu ukuran inputannya berukuran 256x256 pixel dengan 3 lapisan bawaan yaitu lapisan merah, hijau, dan biru(RGB).

Gambar berwarna yang masuk akan diambil fitur-fiturnya dengan cara mengalikan(konvolusi) nilai setiap pixel yang ada dengan nilai filter yaitu berukuran 3x3 dengan jumlah 32 varian. Hasil dari proses konvolusi tersebut akan diseleksi terlebih dahulu menggunakan fungsi aktivasi relu, apabila nilai dari masing-masing pixel bernilai kurang dari 0 maka nilai-nilai tersebut nantinya akan dirubah menjadi nol, dan jika masing-masing nilai setiap pixelnya sudah lebih dari nol maka akan tetap menjadi nilai itu sendiri.

Proses pengambilan fitur berikutnya yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan nilai-nilai baru atau bisa disebut dengan *featured map* yang nantinya akan digunakan untuk pengambilan fitur kembali pada lapisan berikutnya.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 64 varian filter yang berukuran 3x3, dan

hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 64 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 128 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 128 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 256 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 256 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada

setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 512 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian filter sebanyak 512 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Dari hasil pengambilan fitur yang sudah dilakukan, maka berikutnya adalah meratakan matriks yang masih berukuran n kali n menjadi matriks berukuran n kali 1. Apabila pada arsitektur sebelum-sebelumnya langsung berujung pada *layer* keluaran, maka pada arsitektur kali ini sama halnya seperti arsitektur keenam yang mana akan masuk 1 *hidden layer* untuk penguraian fitur dengan bantuan 1024 *neurons* yang kemudian terakhir adalah *layer* dimana *layer* sebagai keluaran, yaitu sebanyak 3 kelas.

Layer (type)	Output Shape	Param #
<hr/>		
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 256)	295168
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
conv2d_4 (Conv2D)	(None, 12, 12, 512)	1180160
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 512)	0
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 1024)	18875392
dense_1 (Dense)	(None, 3)	3075
<hr/>		
Total params:	20,447,043	
Trainable params:	20,447,043	
Non-trainable params:	0	

Gambar 4.10 Summary Arsitektur Ketujuh

Jumlah parameter pada arsitektur ketujuh akan lebih banyak jika dibandingkan dengan arsitektur sebelum-sebelumnya, meskipun jumlah lapisannya lebih banyak, namun pada lapisan terakhir menggunakan bantuan neurons dalam pengambilan fiturnya.

Dengan data awal berukuran 256 pixel kali 256 pixel dan dengan tiga lapisan yaitu merah,hijau,biru(RGB) diambil fiturnya dengan konvolusi berubah ukuran matriksnya menjadi 254 piksel kali 254 piksel kemudian jumlah parameter pada konvolusi pertama sebanyak 896, berikutnya disederhanakan tanpa menghilangkan isi informasi dari gambar dengan menggunakan pooling ukuran filter yaitu 2x2 maka ukurannya berubah menjadi 127 pixel kali 127 pixel.

Bertambah satu lapisan konvolusi maka hasil dari *pooling layer* dijadikan data masukan kembali untuk pengambilan fitur, yang mana ukuran awalnya adalah 127x127 pixel pada konvolusi kedua dikalikan dengan filter 3x3 sebanyak 64 maka akan menjadi 125x125 pixel sebanyak 64 varian juga dengan nilai parameternya yaitu 18.496.

Setelah dari konvolusi akan masuk kembali pada *pooling layer* dengan matriks masukan berukuran 125x125 pixel menjadi matriks berukuran 62x62 pixel dengan banyak varian masih sama yaitu 64 varian.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 62x62 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 128 varian maka akan menjadi 60x60 pixel sebanyak 128 varian dengan jumlah parameter sebanyak 73.856.

Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 60x60 pixel menjadi matriks berukuran 30x30 pixel.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 30x30 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 256 varian maka akan menjadi 28x28 pixel sebanyak 256 varian dengan jumlah parameter sebanyak 295.168. Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 28x28 pixel menjadi matriks berukuran 14x14 pixel.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 14x14 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 512 varian maka akan menjadi 12x12 pixel sebanyak 512 varian dengan jumlah parameter sebanyak 1.180.160. Pengambilan fitur belum juga selesai, akan berlanjut lagi ke tahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 28x28 pixel menjadi matriks berukuran 14x14 pixel.

Sebelum masuk dalam layer *fully connected* atau layer keluaran data-data yang masih berbentuk matriks n kali n dirubah menjadi n kali 1 atau bisa disebut proses perataan maka nilai masukan awalnya sebanyak 512 varian akan menjadi 1 saja dengan banyak parameter yaitu menjadi 1.623.875.

Dan pada layer *fully connected* ada satu *hidden layer* dengan bantuan *neuron* sebanyak 1024 untuk membantu mengecilkan terlebih dahulu sebelum masuk *layer* terakhir atau layer untuk keluaran yaitu menjadi 3 bagian sesuai banyaknya kelas yang diinginkan. Dengan demikian jumlah parameter secara keseluruhan pada arsitektur keenam ini adalah 20.447.043

h. Arsitektur CNN 8

```
#konvolusi pertama
    tf.keras.layers.Conv2D(32,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling pertama
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi kedua
    tf.keras.layers.Conv2D(64,      (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling kedua
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi ketiga
    tf.keras.layers.Conv2D(128,     (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling ketiga
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi keempat
    tf.keras.layers.Conv2D(256,     (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling keempat
    tf.keras.layers.MaxPooling2D(2, 2),
    #konvolusi kelima
    tf.keras.layers.Conv2D(512,     (3,      3),      activation='relu',
input_shape=(256, 256, 3)),
    #pooling kelima
    tf.keras.layers.MaxPooling2D(2, 2),

    #perataan
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    #keluaran
    tf.keras.layers.Dense(3, activation='softmax')
])
```

Modul Program 4.20 Arsitektur CNN 8

Selanjutnya merupakan arsitektur cnn kedelapan yaitu penggunaan library tensorflow yang masih sama seperti arsitektur sebelum-sebelumnya untuk penggunaan cnnnya, pada

arsitektur ini dengan rincian yaitu menggunakan lima lapisan konvolusi, lima lapisan *pooling*, dan dua lapisan *fully connected*. Ukuran gambar harus sama sisi yaitu ukuran inputannya berukuran 256x256 pixel dengan 3 lapisan bawaan yaitu lapisan merah, hijau, dan biru(RGB).

Gambar berwarna yang masuk akan diambil fitur-fiturnya dengan cara mengalikan(konvolusi) nilai setiap pixel yang ada dengan nilai filter yaitu berukuran 3x3 dengan jumlah 32 varian. Hasil dari proses konvolusi tersebut akan diseleksi terlebih dahulu menggunakan fungsi aktivasi *relu*, apabila nilai dari masing-masing pixel bernilai kurang dari 0 maka nilai-nilai tersebut nantinya akan dirubah menjadi nol, dan jika masing-masing nilai setiap pixelnya sudah lebih dari nol maka akan tetap menjadi nilai itu sendiri.

Proses pengambilan fitur berikutnya yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan nilai-nilai baru atau bisa disebut dengan *featured map* yang nantinya akan digunakan untuk pengambilan fitur kembali pada lapisan berikutnya.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 64 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter sebanyak 64 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 128 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 128 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 256 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian fileter seabanyak 256 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Setelah dari *pooling layer* akan berlanjut kembali untuk pengambilan fiturnya menggunakan lapisan konvolusi yaitu sebanyak 512 varian filter yang berukuran 3x3, dan hasilnya akan diseleksi kembali menggunakan fungsi aktivasi *relu* yang mana apabila ada nilai yang kurang dari nol maka nilainya akan dirubah secara permanen menjadi nol, dan apabila nilainya sudah lebih dari nol, maka nilai yang dikembalikan adalah nilai itu sendiri.

Hasil pengambilan fitur pada lapisan konvolusi dengan varian filter sebanyak 512 akan masuk kembali pada *pooling layer*, yaitu dengan cara mengambil nilai tertentu pada setiap kelompok filter(*pooling*) dengan cara mengambil nilai tertinggi atau maksimal berdasarkan luasan ukuran filter yang berukuran 2x2 pixel, nantinya dari pengambilan nilai tersebut akan menghasilkan *featured map* yang baru.

Dari hasil pengambilan fitur yang sudah dilakukan, maka berikutnya adalah meratakan matriks yang masih berukuran n kali n menjadi matriks berukuran n kali 1. Apabila pada arsitektur sebelum-sebelumnya langsung berujung pada *layer* keluaran, maka pada arsitektur kali ini sama halnya seperti arsitektur keenam yang mana akan masuk 1 *hidden layer* untuk penguraian fitur dengan bantuan 512 *neurons* yang kemudian terakhir adalah *layer* dimana *layer* sebagai keluaran, yaitu sebanyak 3 kelas.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 256)	295168
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
conv2d_4 (Conv2D)	(None, 12, 12, 512)	1180160
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 512)	0
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 512)	9437696
dense_1 (Dense)	(None, 4)	2052
<hr/>		
Total params: 11,008,324		
Trainable params: 11,008,324		
Non-trainable params: 0		

Gambar 4.11 Summary Arsitektur Kedelapan

Jumlah paramater pada arsitektur kedelapan akan lebih banyak jika dibandingkan dengan arsitektur pertama hingga keenam, dan lebih sedikit jika dibandingkan arsitektur ketujuh, meskipun jumlah lapisannya lebih banyak, namun pada lapisan terakhir menggunakan bantuan neurons dalam pengambilan fiturnya.

Dengan data awal berukuran 256 pixel kali 256 pixel dan dengan tiga lapisan yaitu merah,hijau,biru(RGB) diambil fiturnya dengan konvolusi berubah ukuran matriksnya menjadi 254 piksel kali 254 piksel kemudian jumlah parameter pada konvolusi pertama sebanyak 896, berikutnya disederhanakan tanpa menghilangkan isi informasi dari gambar dengan menggunakan pooling filter yaitu 2x2 maka ukurannya berubah menjadi 127 pixel kali 127 pixel.

Bertambah satu lapisan konvolusi maka hasil dari *pooling layer* dijadikan data masukan kembali untuk pengambilan fitur, yang mana ukuran awalnya adalah 127x127 pixel pada konvolusi kedua dikalikan dengan filter 3x3 sebanyak 64 maka akan menjadi 125x125 pixel sebanyak 64 varian juga dengan nilai parameternya yaitu 18.496.

Setelah dari konvolusi akan masuk kembali pada *pooling layer* dengan matriks masukan berukuran 125x125 pixel menjadi matriks berukuran 62x62 pixel dengan banyak varian masih sama yaitu 64 varian.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 62x62 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 128 varian maka akan menjadi 60x60 pixel sebanyak 128 varian dengan jumlah parameter sebanyak 73.856.

Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling layer* dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 60x60 pixel menjadi matriks berukuran 30x30 pixel.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 30x30 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 256 varian maka

akan menjadi 28x28 pixel sebanyak 256 varian dengan jumlah parameter sebanyak 295.168. Pengambilan fitur belum juga selesai, akan berlanjut lagi ketahap *pooling* layer dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 28x28 pixel menjadi matriks berukuran 14x14 pixel.

Disaring kembali menggunakan lapisan konvolusi dengan masukan yang berukuran 14x14 pixel dilakukan konvolusi dengan filter berukuran 3x3 sebanyak 512 varian maka akan menjadi 12x12 pixel sebanyak 512 varian dengan jumlah parameter sebanyak 1.180.160. Pengambilan fitur belum juga selesai, akan berlanjut lagi ke tahap *pooling* layer dengan cara mengambil nilai maksimalnya dalam setiap filter yang berukuran 2x2, matriks masukan berukuran 28x28 pixel menjadi matriks berukuran 14x14 pixel.

Sebelum masuk dalam layer *fully connected* atau layer keluaran data-data yang masih berbentuk matriks n kali n dirubah menjadi n kali 1 atau bisa disebut proses perataan maka nilai masukan awalnya sebanyak 512 varian akan menjadi 1 saja dengan banyak parameter yaitu menjadi 1.623.875.

Dan pada layer *fully connected* ada satu *hidden layer* dengan bantuan *neuron* sebanyak 512 untuk membantu mengecilkan terlebih dahulu sebelum masuk *layer* terakhir atau layer untuk keluaran yaitu menjadi 3 bagian sesuai banyaknya kelas yang diinginkan. Dengan demikian jumlah parameter secara keseluruhan pada arsitektur keenam ini adalah 11.008.324

```
model.summary()
```

Modul Program 4.21 Untuk Keluaran Summary Arsitektur

Fungsi yang sudah disediakan library tensorflow untuk mengeluarkan rincian susunan arsitektur yang digunakan, arsitektur yang sudah dibuat susunannya dipanggil kembali variabelnya kemudian ditambahkan fungsi summary. Tujuan dari penggunaan ini untuk mengetahui seberapa banyak jumlah parameter yang ada, dan supaya lebih mudah pengecekannya apabila nantinya selama proses pembelajaran terjadi kendala dan berlebih.

4.1.3 Implementasi Perancangan Pengurangan Bias

```
model.compile(optimizer = tf.keras.optimizers.Adam(),
              loss = 'categorical_crossentropy',
              metrics = ['accuracy']
)
```

Modul Program 4.22 Pengurangan Error

Setelah membuat arsitektur dari CNN, model kita belum bisa melakukan apa-apa. Agar model bisa belajar, maka perlu memanggil fungsi compile pada model kita dan menspesifikasikan optimizer dan loss function. Untuk optimizer menggunakan Adam. Selanjutnya untuk loss function kita dapat menggunakan categorical crossentropy pada kasus klasifikasi 3 kelas atau lebih. Parameter metrics berfungsi untuk menampilkan metrik yang dipilih pada proses pelatihan model.

4.1.4 Proses Pelatihan(Training)

```
# 6. pembelajaran model(training)
start = datetime.datetime.now()
history = model.fit(
    train_generator,
    steps_per_epoch = 1260//15,
    epochs = 50,
    validation_data = validation_generator,
    validation_steps = 539//15,
)
end = datetime.datetime.now()
```

Modul Program 4.23 Pelatihan Data

Inisiasi terlebih dahulu pengambilan waktu saat sebelum memulai pembelajaran data untuk digunakan nantinya sebagai penanda awalan perhitungan durasi pembelajaran data secara keseluruhan. Kemudian inisiasi variabel penyimpan yaitu *history* apabila ingin melihat kurva pembelajaran nantinya. Isi dari history sendiri yaitu arsitektur yang dirancang atau disusun sebelumnya dipanggil kembali untuk dijadikan dasar pengambilan fitur-fiturnya, kemudian panggil data latih yang ingin dipelajari, berikan nilai steps_epoch sebanyak total data latih secara keseluruhan dibagi dengan banyaknya jumlah batch, kemudian berikan nilai epoch atau banyaknya putaran selama proses pembelajaran berlangsung, panggil data validasi yang digunakan, kemudian berikan nilai validasi_steps

dengan membagi antara nilai data validasi dengan banyaknya jumlah epoch. Dan di akhir program tandai waktu selesainya proses pembelajaran.

Tabel 4.1 Rincian Proses Pembelajaran Data

Epoch 1/50
84/84 [=====] - 607s 7s/step - loss: 1.2918 - accuracy: 0.3706 - val_loss: 0.9668 - val_accuracy: 0.5581
Epoch 2/50
84/84 [=====] - 88s 1s/step - loss: 0.8278 - accuracy: 0.6484 - val_loss: 0.6025 - val_accuracy: 0.7943
Epoch 3/50
84/84 [=====] - 88s 1s/step - loss: 0.5767 - accuracy: 0.7844 - val_loss: 0.6473 - val_accuracy: 0.7600
Epoch 4/50
84/84 [=====] - 88s 1s/step - loss: 0.5247 - accuracy: 0.7994 - val_loss: 0.5500 - val_accuracy: 0.7867
Epoch 5/50
84/84 [=====] - 87s 1s/step - loss: 0.4503 - accuracy: 0.8431 - val_loss: 0.4119 - val_accuracy: 0.8419
Epoch 6/50
84/84 [=====] - 87s 1s/step - loss: 0.3912 - accuracy: 0.8716 - val_loss: 0.4626 - val_accuracy: 0.8248
Epoch 7/50
84/84 [=====] - 88s 1s/step - loss: 0.3244 - accuracy: 0.8861 - val_loss: 0.5523 - val_accuracy: 0.8152
Epoch 8/50
84/84 [=====] - 88s 1s/step - loss: 0.3309 - accuracy: 0.8903 - val_loss: 0.5844 - val_accuracy: 0.7733
Epoch 9/50
84/84 [=====] - 88s 1s/step - loss: 0.3120 - accuracy: 0.9028 - val_loss: 0.4251 - val_accuracy: 0.8533
Epoch 10/50
84/84 [=====] - 89s 1s/step - loss: 0.3032 - accuracy: 0.8968 - val_loss: 0.6858 - val_accuracy: 0.7829
Epoch 11/50
84/84 [=====] - 88s 1s/step - loss: 0.3177 - accuracy: 0.8845 - val_loss: 0.3649 - val_accuracy: 0.8819
Epoch 12/50
84/84 [=====] - 88s 1s/step - loss: 0.2623 - accuracy: 0.9150 - val_loss: 0.4396 - val_accuracy: 0.8438
Epoch 13/50
84/84 [=====] - 88s 1s/step - loss: 0.2523 - accuracy: 0.9052 - val_loss: 0.5130 - val_accuracy: 0.8324
Epoch 14/50
84/84 [=====] - 88s 1s/step - loss: 0.2541 - accuracy: 0.9059 - val_loss: 0.3786 - val_accuracy: 0.8686
Epoch 15/50
84/84 [=====] - 87s 1s/step - loss: 0.3101 - accuracy: 0.8966 - val_loss: 0.5659 - val_accuracy: 0.8000
Epoch 16/50
84/84 [=====] - 88s 1s/step - loss: 0.2282 - accuracy: 0.9175 - val_loss: 0.5093 - val_accuracy: 0.8248
Epoch 17/50
84/84 [=====] - 88s 1s/step - loss: 0.1820 - accuracy: 0.9405 - val_loss: 0.4543 - val_accuracy: 0.8610
Epoch 18/50
84/84 [=====] - 88s 1s/step - loss: 0.1879 - accuracy: 0.9366 - val_loss: 0.4992 - val_accuracy: 0.8343

Tabel 4.2 Lanjutan Proses Pembelajaran Data

Epoch 19/50
84/84 [=====] - 87s 1s/step - loss: 0.2110 - accuracy: 0.9298 - val_loss: 0.4562 - val_accuracy: 0.8590
Epoch 20/50
84/84 [=====] - 87s 1s/step - loss: 0.1592 - accuracy: 0.9467 - val_loss: 0.4236 - val_accuracy: 0.8705
Epoch 21/50
84/84 [=====] - 88s 1s/step - loss: 0.1753 - accuracy: 0.9455 - val_loss: 0.6301 - val_accuracy: 0.7848
Epoch 22/50
84/84 [=====] - 88s 1s/step - loss: 0.2389 - accuracy: 0.9200 - val_loss: 0.3964 - val_accuracy: 0.8743
Epoch 23/50
84/84 [=====] - 88s 1s/step - loss: 0.1394 - accuracy: 0.9520 - val_loss: 0.5177 - val_accuracy: 0.8667
Epoch 24/50
84/84 [=====] - 87s 1s/step - loss: 0.1794 - accuracy: 0.9400 - val_loss: 0.4756 - val_accuracy: 0.8590
Epoch 25/50
84/84 [=====] - 85s 1s/step - loss: 0.1352 - accuracy: 0.9563 - val_loss: 0.4155 - val_accuracy: 0.8743
Epoch 26/50
84/84 [=====] - 84s 1s/step - loss: 0.1268 - accuracy: 0.9616 - val_loss: 0.6494 - val_accuracy: 0.7867
Epoch 27/50
84/84 [=====] - 84s 1s/step - loss: 0.2208 - accuracy: 0.9346 - val_loss: 0.3689 - val_accuracy: 0.8629
Epoch 28/50
84/84 [=====] - 84s 1s/step - loss: 0.0953 - accuracy: 0.9748 - val_loss: 0.5164 - val_accuracy: 0.8438
Epoch 29/50
84/84 [=====] - 84s 999ms/step - loss: 0.1183 - accuracy: 0.9595 - val_loss: 0.4438 - val_accuracy: 0.8590
Epoch 30/50
84/84 [=====] - 83s 992ms/step - loss: 0.1133 - accuracy: 0.9572 - val_loss: 0.3805 - val_accuracy: 0.8800
Epoch 31/50
84/84 [=====] - 83s 996ms/step - loss: 0.1171 - accuracy: 0.9619 - val_loss: 0.4084 - val_accuracy: 0.8914
Epoch 32/50
84/84 [=====] - 83s 996ms/step - loss: 0.1115 - accuracy: 0.9652 - val_loss: 0.4317 - val_accuracy: 0.8705
Epoch 33/50
84/84 [=====] - 83s 991ms/step - loss: 0.0943 - accuracy: 0.9608 - val_loss: 0.4072 - val_accuracy: 0.8914
Epoch 34/50
84/84 [=====] - 83s 989ms/step - loss: 0.0683 - accuracy: 0.9848 - val_loss: 0.3756 - val_accuracy: 0.8952
Epoch 35/50
84/84 [=====] - 83s 988ms/step - loss: 0.1708 - accuracy: 0.9372 - val_loss: 0.6605 - val_accuracy: 0.8648
Epoch 36/50
84/84 [=====] - 83s 989ms/step - loss: 0.1413 - accuracy: 0.9597 - val_loss: 0.4195 - val_accuracy: 0.8724
Epoch 37/50
84/84 [=====] - 82s 986ms/step - loss: 0.1033 - accuracy: 0.9696 - val_loss: 0.3721 - val_accuracy: 0.8800
Epoch 38/50

Tabel 4.3 Lanjutan Proses Pembelajaran Data

```

84/84 [=====] - 82s 976ms/step - loss: 0.0885 -
accuracy: 0.9717 - val_loss: 0.5216 - val_accuracy: 0.8629
Epoch 39/50
84/84 [=====] - 82s 981ms/step - loss: 0.0962 -
accuracy: 0.9672 - val_loss: 0.3739 - val_accuracy: 0.9010
Epoch 40/50
84/84 [=====] - 81s 974ms/step - loss: 0.0698 -
accuracy: 0.9767 - val_loss: 0.3860 - val_accuracy: 0.8857
Epoch 41/50
84/84 [=====] - 81s 970ms/step - loss: 0.0778 -
accuracy: 0.9732 - val_loss: 0.5230 - val_accuracy: 0.8724
Epoch 42/50
84/84 [=====] - 82s 975ms/step - loss: 0.1054 -
accuracy: 0.9594 - val_loss: 0.4504 - val_accuracy: 0.8990
Epoch 43/50
84/84 [=====] - 82s 983ms/step - loss: 0.0641 -
accuracy: 0.9769 - val_loss: 0.5182 - val_accuracy: 0.8686
Epoch 44/50
84/84 [=====] - 82s 980ms/step - loss: 0.1477 -
accuracy: 0.9558 - val_loss: 0.5672 - val_accuracy: 0.8381
Epoch 45/50
84/84 [=====] - 82s 976ms/step - loss: 0.0851 -
accuracy: 0.9683 - val_loss: 0.5682 - val_accuracy: 0.8629
Epoch 46/50
84/84 [=====] - 81s 966ms/step - loss: 0.0760 -
accuracy: 0.9674 - val_loss: 0.4210 - val_accuracy: 0.8819
Epoch 47/50
84/84 [=====] - 81s 967ms/step - loss: 0.0719 -
accuracy: 0.9766 - val_loss: 0.5144 - val_accuracy: 0.8533
Epoch 48/50
84/84 [=====] - 83s 996ms/step - loss: 0.0679 -
accuracy: 0.9705 - val_loss: 0.4578 - val_accuracy: 0.8686
Epoch 49/50
84/84 [=====] - 84s 1s/step - loss: 0.0506 -
accuracy: 0.9826 - val_loss: 0.4030 - val_accuracy: 0.8933
Epoch 50/50
84/84 [=====] - 84s 1s/step - loss: 0.0941 -
accuracy: 0.9738 - val_loss: 0.3462 - val_accuracy: 0.9124

```

4.1.5 Evaluasi Hasil Pelatihan

a. Kurva pembelajaran model training dan validasi

```

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

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

epochs = 50
epochs_range = range(epochs)

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

```

Modul Program 4.24 Evaluasi Pembelajaran Data Latih dengan Data Validasi

```

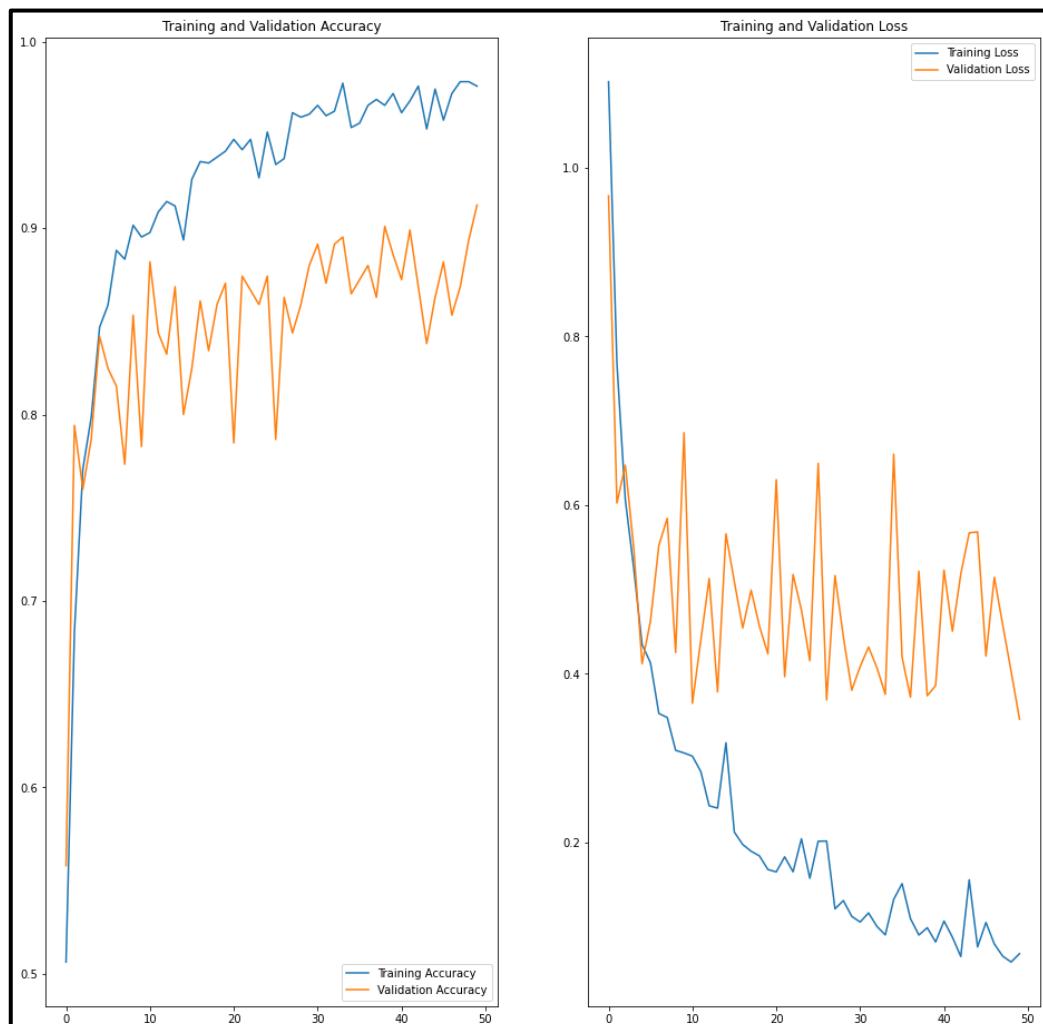
plt.title('Training and Validation Accuracy')

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

```

Modul Program 4.25 Lanjutan Evaluasi Pembelajaran Data Latih dengan Data Validasi

Setelah proses pembelajaran data berlangsung maka dapat dilihat *trend* perkembangan dari setiap tahapannya dengan cara memvisualisasikan pada gambar, cara yang paling mudah yaitu menggunakan library matplotlib yang fungsi-fungsinya sudah tersedia. Panggil terlebih dahulu data latih selama pembelajaran, kemudian panggil juga data validasi selama pembeajaran, nantinya akan diketahui pergerakan akurasi dan loss. Untuk hasil akhirnya seperti dibawah ini



Gambar 4.12 Kurva Pembelajaran Model

b. Hasil evaluasi

```
loss, acc = model.evaluate(validation_generator)
print("Accuracy: {:.2f}%".format(100 * acc))
print("Loss: ", loss)
elapsed = end-start
print ('Time: ', elapsed)
```

Modul Program 4.26 Nilai Akurasi dan Loss Model

Melihat hasil pembelajaran sangat penting dalam pembuatan model, deklarasikan variabel akurasi dan loss untuk mengambil hasil pembelajaran yang telah dilakukan. Tampilkan hasil akurasi dengan mencetak nilai rata-rata secara keseluruhan berdasarkan dengan data validasi yang digunakan. Waktu start dan akhir digunakan untuk mengetahui berapa lama proses pembelajaran berjalan, dengan melakukan operasi pengurangan maka akan dapat hasilnya, untuk hasil akhirnya seperti di bawah ini.

```
36/36 [=====] - 27s 734ms/step - loss: 0.3541 - accuracy: 0.9184
Accuracy: 91.84%
Loss: 0.354060560464859
Time: 1:19:41.003246
```

Gambar 4.13 Hasil Evaluasi

4.1.6 Penyimpanan Model

```
model.save('/content/drive/My Drive/Colab Notebooks/Makarim/Model/Percobaan 3/Learning Rate/L00005/upgrade/L00005 model bagus.h5')
```

Modul Program 4.27 Penyimpanan Model

Setelah proses pembentukan arsitektur kemudian dikenali menggunakan compile, proses pembelajaran dapat dijalankan, dan dirasa semua arsitektur, waktu, akurasi, dan loss sudah berjalan sesuai keinginan maka berikutnya adalah menyimpan hasil pembelajaran tadi yang gunanya untuk digunakan pada pengujian atau implementasi pada aplikasi.

4.1.7 Implementasi Pengujian Model

a. Inisiasi library

```
import tensorflow as tf
import streamlit as st
from PIL import Image
import numpy as np
from load_css import local_css
local_css("style.css")
```

Modul Program 4.28 Inisiasi Library Pengujian Model

Panggil semua library yang akan digunakan pada penerapan aplikasi, diantaranya library tensorflow untuk pengolahan model, streamlit untuk visualisasi, PIL untuk mengunggah gambar, numpy untuk operasi matematika, dan yang terakhir memanggil package style untuk memberi keindahan pada aplikasi.

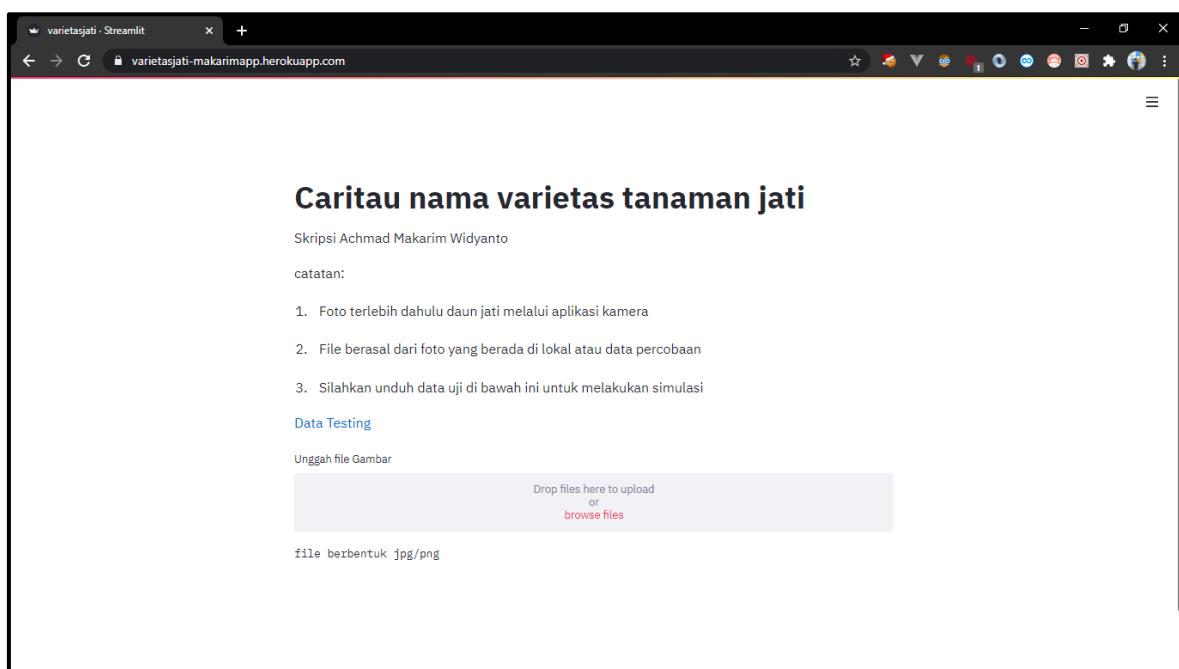
b. Halaman antarmuka awal

```
st.write("""
    # Caritau nama varietas tanaman jati
"""
)
st.write("Skripsi Achmad Makarim Widhyanto")

link = '[Data Testing] (https://drive.google.com/drive/folders/1EaP1X0d-E8iFykyv\_5a8f\_grgnAXn4Ja?usp=sharing)'
st.write("catatan: ")
st.write("1. Foto terlebih dahulu daun jati melalui aplikasi kamera")
st.write("2. File berasal dari foto yang berada di lokal atau data percobaan")
st.write("3. Silahkan unduh data uji di bawah ini untuk melakukan simulasi")
st.markdown(link, unsafe_allow_html=True)
file = st.file_uploader("Unggah file Gambar", type = ["jpg", "png"])
```

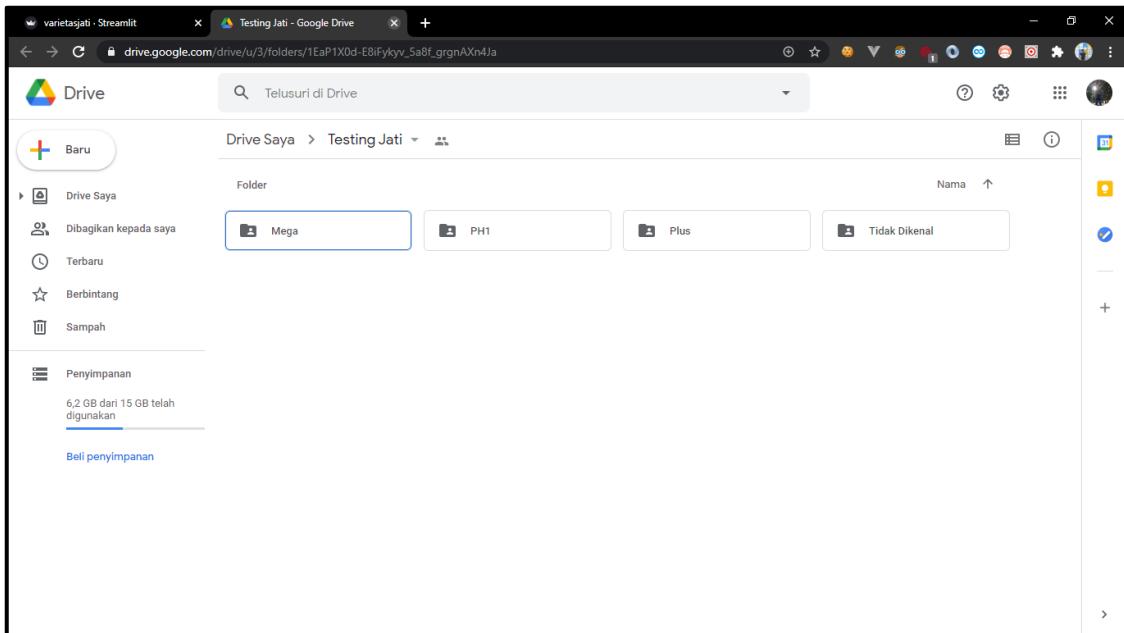
Modul Program 4.29 Halaman Antar Muka

Menggunakan library streamlit akan sangat mempermudah memvisualisasikan semua data yang dimiliki. Berikut ini merupakan tampilan beranda aplikasi web menggunakan streamlit. Alamat tautan aplikasi web yaitu varietasjati-makarimapp.



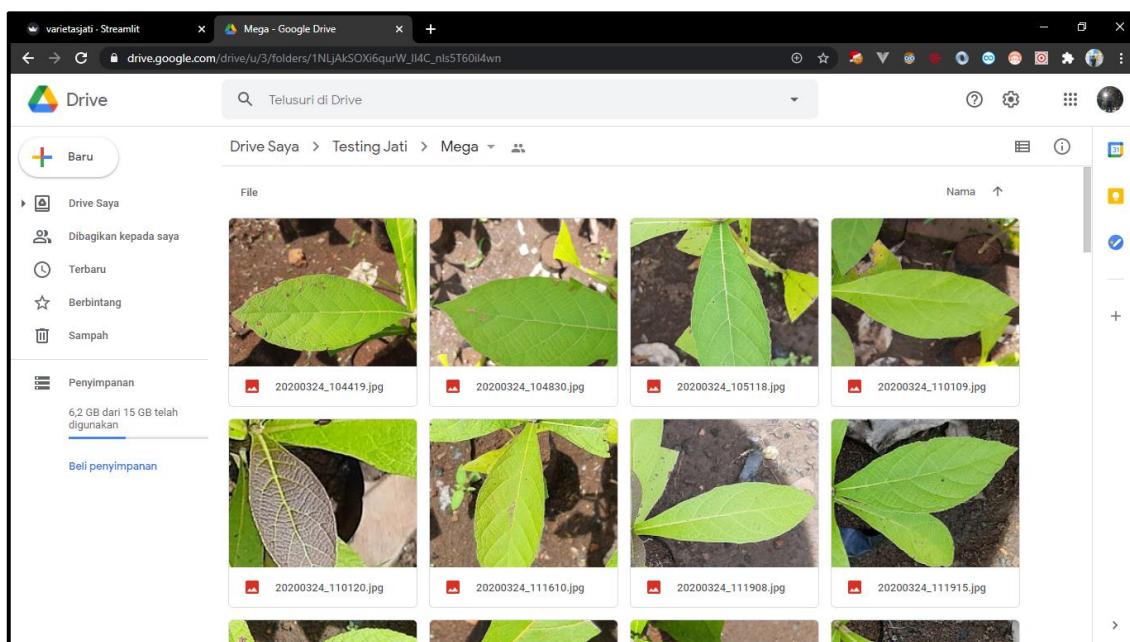
Gambar 4.14 Halaman Antar Muka

Aplikasi web ini dapat digunakan dengan dua cara yaitu penggunaan sample data dan penggunaan data secara langsung dari pengguna. Apabila ingin menggunakan sample data dapat akses link *data testing* yang sudah disediakan.

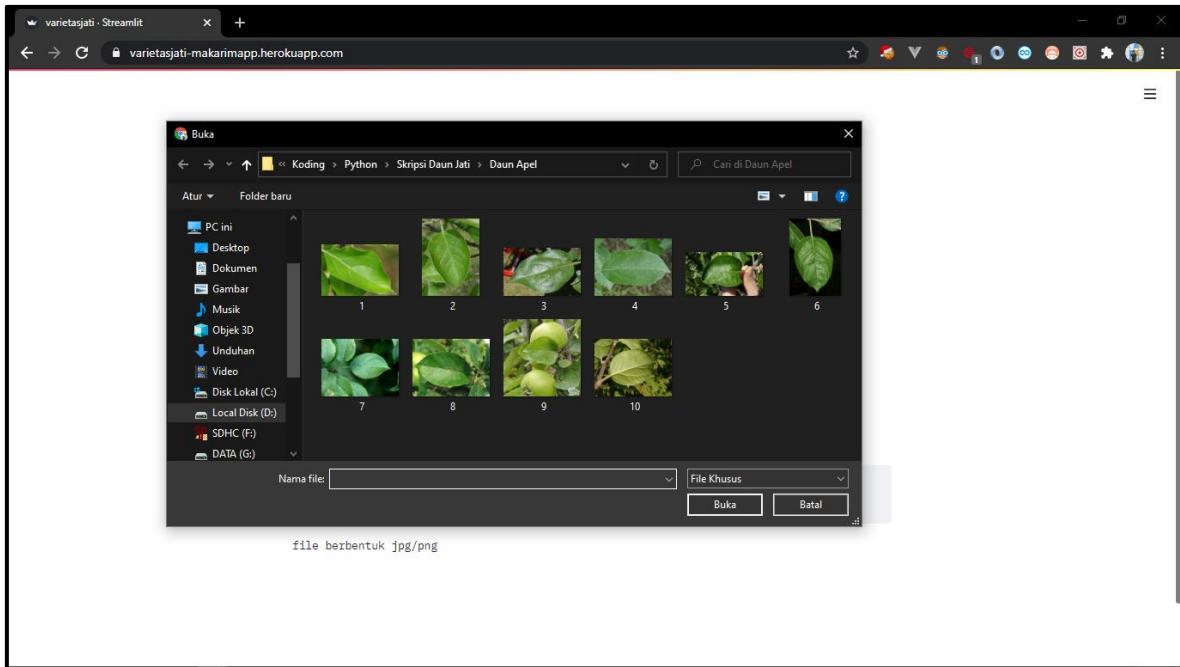


Gambar 4.14 Halaman Penyimpanan Data Sample

Data sample digunakan apabila pengguna ingin mencoba-coba aplikasi dan belum memiliki data sesungguhnya, isi terdiri dari foto-foto varietas jati yang digunakan mega, plus, dan PH1, dan juga kelas tidak diketahui apabila data selain 3 kelas varietas.



Gambar 4.15 Halaman Contoh Data Sample



Gambar 4.16 Popup Peletakan Data Uji yang Ingin Diunduh

Untuk bisa menggunakan data-data sample tersebut dapat diunduh terlebih dahulu foto-foto daun yang sudah disediakan, arahkan pada penyimpanan lokal yang dapat diingat dan mudah diakses.

c. Load model

```
model = tf.keras.models.load_model('L00005_HL512_bagus.h5')
```

Modul Program 4.30 Mengarahkan Lokasi Model

Panggil model yang sudah dipelajari pada tahap sebelumnya dengan cara load model.

d. *Preprocessing* data uji

```
dimensi_gambar = (256, 256)
channel = (3,)
input_shape = dimensi_gambar + channel
labels = ['mega', 'phl', 'plus', 'tidak diketahui']

def preprocess(gambar, dimensi_gambar):

    # sebelum preprocess
    nimg = gambar.convert('RGB').resize(dimensi_gambar, resample= 0)
    # print('sebelum preprocess:', gambar)

    # setelah preprocess
    img_arr = (np.array(nimg))/255
    # print('setelah preprocess: \n', img_arr)
    # st.image(img_arr, use_column_width=True)
    return img_arr
```

Modul Program 4.31 Tahap Preprocessing Pengujian

Pengujian dapat dilakukan dengan baik apabila persiapan pengolahan data sudah sesuai dengan persiapan pengolah data ketika pembelajaran model. Langkah-langkah persiapan diantaranya ada memberikan patokan ukuran data gambar beresolusi 256 x 256 pixel, kemudian dirubah menjadi tiga *channel* atau tiga lapisan warna yaitu merah, hijau, biru(RGB). Hasil dari perubahan menjadi nilai RGB akan dirubah skalanya yang awalnya nol hingga duaratus lima puluh lima menjadi skala yang berukuran nol hingga satu, yang nantinya hasil tersebut akan dikembalikan nilanya.

e. Perataan data uji

```
def reshape(imgs_arr):
    return np.stack(imgs_arr, axis=0)
```

Modul Program 4.32 Meratakan Matriks Data Uji

Karena dalam arsitektur cnn berbentuk satu dimensi maka gambar data uji harus juga dirubah menjadi satu dimensi, dengan menggunakan library numpy dan package yang ada yaitu stack dengan isian matriks gambar data uji.

f. Proses klasifikasi citra

```
if file is None:
    st.text("file berbentuk jpg/png")
else:
    'Memulai pembacaan data'

    # Add a placeholder
    latest_iteration = st.empty()
    bar = st.progress(0)
    for i in range(100):
        # Update the progress bar with each iteration.
        latest_iteration.text(f'Proses... {i+1}%')
        bar.progress(i + 1)
        time.sleep(0.1)
    gambar = Image.open(file)
    st.image(gambar, use_column_width=True)

    X = preprocess(gambar, dimensi_gambar)
    X = reshape([X])
    y = model.predict(X)

    kategori = labels[np.argmax(y)]

    'Selesai!'
    st.subheader('Hasil')
    if kategori == 'tidak diketahui':
```

Modul Program 4.33 Proses Klasifikasi Citra

```

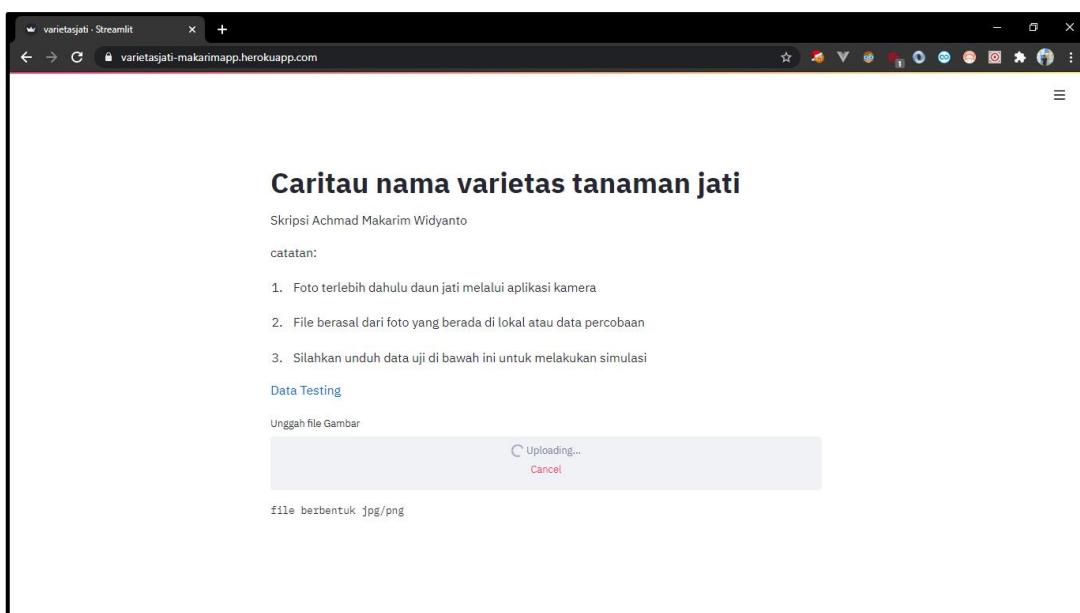
    st.markdown(f"Varietas <span class='highlight merah'>tidak </span>
diketahui", unsafe_allow_html=True)
else :
    st.markdown(f"Tanaman jati ini termasuk varietas <span
class='highlight ijo'>{kategori}</span> dengan nilai kemiripan <span
class='highlight
{(np.max(y)*100).astype(int)}%</span>", unsafe_allow_html=True)
    st.subheader('Nilai Probabilitas')
    st.text("(0: Mega, 1: PH1, 2: Plus, 3: Tidak Diketahui)")
    st.write(y)

```

Modul Program 4.34 Lanjutan Proses Klasifikasi Citra

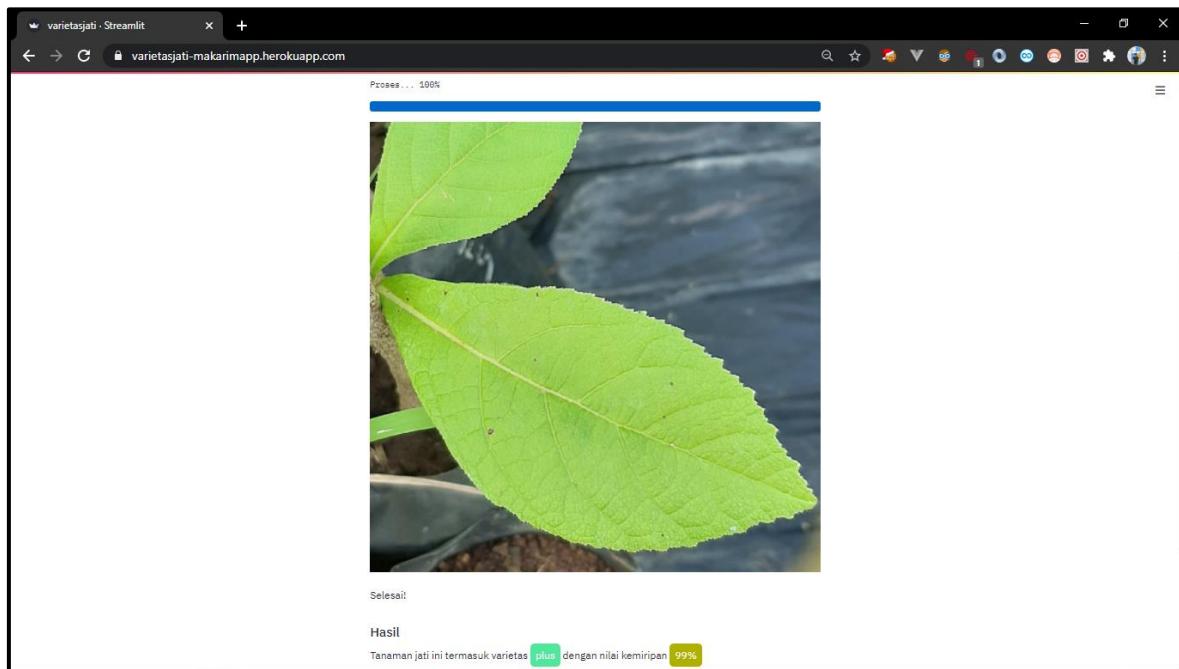
Setelah proses persiapan pengelahan data, maka kali ini adalah proses pengklasifikasian gambar data uji untuk menentukan apakah gambar data uji termasuk kelas varietas jati mega, ph1, plus atau bukan. File yang diunggah harus berbentuk jpg atau png, selain jenis file ekstensi dua tersebut tidak akan bisa digunakan.

Selama proses pengunggahan, proses *praprocess*, dan perataan berlangsung akan ditampilkan tampilan *loading* data dari satu hingga seratus supaya pengguna dapat menunggu hasil dari pemrosesan yang sedang berjalan. Ketika sudah berhasil maka akan memanggil fungsi yang sudah disediakan oleh library tensorflow untuk diprediksi untuk mendapatkan hasil dari pengklasifikasian, hasil dari probabilitas dari masing-masing kelas akan diambil nilai tertingginya menggunakan *library* numpy yang artinya bahwa nilai tertinggi tersebut merupakan kelas dari gambar yang sedang diunggah.



Gambar 4.17 Interface Halaman Utama

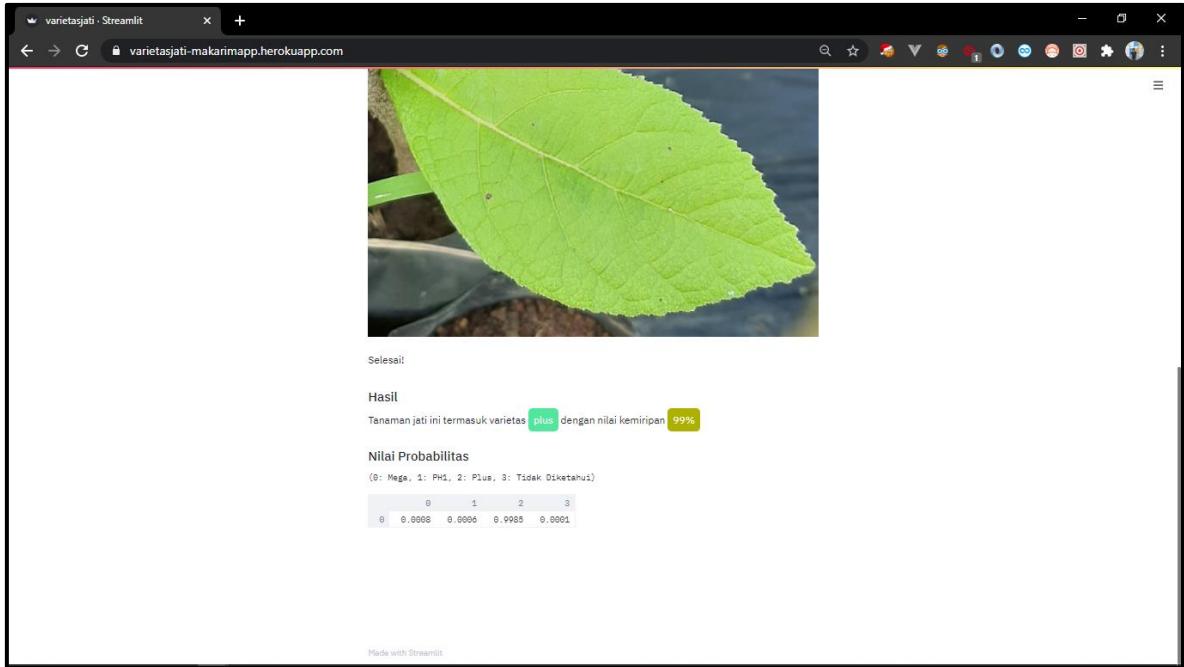
Hasil dari program yang sudah dibuat maka akan menjadi seperti halaman sebelumnya, yaitu dimulai dari halaman utama aplikasi yang berisikan judul aplikasi, petunjuk penggunaan aplikasi, lokasi penyimpanan contoh data uji, dan tombol untuk mengunggah gambar data ujinya.



Gambar 4.18 Interface Hasil Klasifikasi Citra

Untuk mengunggah gambar data uji yang ingin diketahui kelas varietasnya maka gunakan tombol *browse file* yang nantinya akan diarahkan pada lokasi penyimpanan lokal pengguna, baik contoh data uji yang sudah diunduh maupun gambar data uji sesungguhnya apabila digunakan pada kebun.

Hasil pengunggahan gambar dari lokasi penyimpanan lokal menuju aplikasi akan tampak seperti gambar 4.18, dimana gambar yang berhasil diunggah akan tampil kepada pengguna terlebih dahulu, supaya memastikan kepada pengguna apakah gambar yang diunggah sudah sesuai atau belum.



Gambar 4.19 Interface Lanjutan Klasifikasi Citra

Setelah gambar yang diunggah muncul, maka hasil dari pengklasifikasian kelas akan muncul di tampilan juga beserta probabilitas dari masing-masing kelas. Dengan contoh pada gambar menunjukkan gambar daun yang diunggah termasuk varietas jati plus dengan tingkat kemiripan yaitu sebesar 99% dimana probabilitas dari masing-masing kelas yang ada yaitu untuk kelas varietas jati mega adalah 0.000, kemudia kelas varietas jati ph1 adalah 0.0006, kemudian kelas varietas jati plus 0.9995, dan kelas tidak diketahui adalah 0.0001.

g. Style tampilan

```
.highlight {
    border-radius: 0.4rem; color: white;
    padding: 0.5rem; margin-bottom: 1rem;
}
.bold {
    padding-left: 1rem;
    font-weight: 700;
}
.ijo {
    background-color: rgb(81, 231, 156);
}
.biru {
    background-color: rgb(174, 177, 0);
}
.merah {
    background-color: rgb(211, 23, 23);
}
```

Modul Program 4.35 Style Tampilan

Supaya tampilan aplikasi lebih informatif maka dibuatkan sebuah *style* kusus menggunakan css dengan rincian *highlight* digunakan untuk mempertegas tulisan kelas varietas, kemudian *bold* untuk mempertebal tulisan, kelas ijo untuk memberikan warna ijo pada tulisan kelas varietas apabila benar, kelas biru untuk menuliskan prosentase besaran tingkat kemiripan, dan kelas warna merah untuk menandai apabila gambar yang diunggah bukan termasuk kelas varietas jati mega, ph1, atau plus.

h. Load style tampilan

```
import streamlit as st

def local_css(file_name):
    with open(file_name) as f:
        st.markdown('<style>{}</style>'.format(f.read()), unsafe_allow_html=True)
```

Modul Program 4.36 Memanggil Style Tampilan

Untuk dapat menggunakan *file* ekstensi css yang sudah dibuat tadi maka perlu dipanggil terlebih dahulu pada program utama supaya dapat dikenali, dengan cara menggunakan *library* streamlit kemudian tuliskan formatnya.

i. Procfile heroku

```
web: sh setup.sh && streamlit run varietasjati.py
```

Modul Program 4.37 Inisiasi Lokasi Program

Karena lokasi *hosting* menggunakan heroku maka supaya aplikasi dapat diunggah atau dijalankan pada internet secara luas maka heroku perlu mengenali terlebih dahulu lokasi nama program utama yang akan dijalankan, dengan cara tersebut maka heroku dapat mengetahui nama file varietasjati.py yang akan djalankan pada aplikasi.

j. Requirements library heroku

```
tensorflow==2.3.0
streamlit==0.62.0
numpy==1.16.5
pillow==6.2.0
```

Modul Program 4.38 Inisiasi Library yang Digunakan

Heroku juga perlu mengetahui *library* apa saja yang akan dipakai pada aplikasi ini

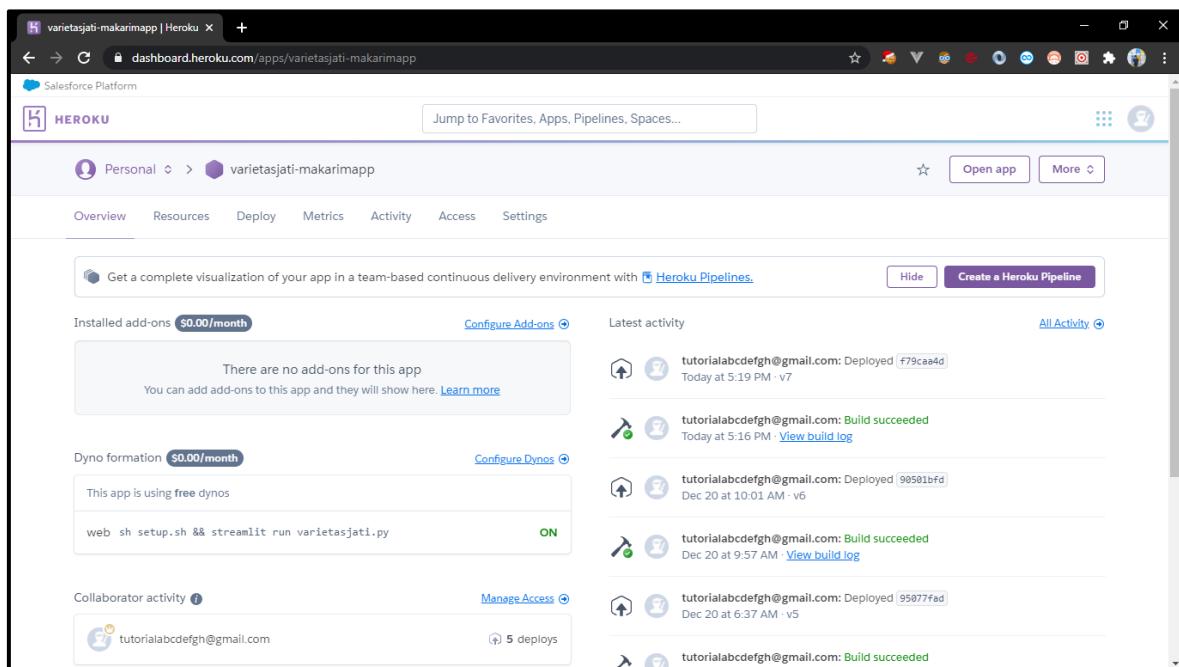
library yang digunakan diantaranya ada tensorflow, streamlit, numpy, pillow, pandas, dan time. Untuk pandas dan time tidak perlu dideklarasikan karena *library* tersebut sudah tersedia pada heroku.

k. Setup heroku

```
mkdir -p ~/.streamlit
echo "[server]
headless = true
port = $PORT
enableCORS = false
" > ~/.streamlit/config.toml
```

Modul Program 4.39 Inisiasi Pengaturan Hosting Server

Untuk bisa mengkoneksikan aplikasi dengan server maka pada aplikasi perlu dideklarasikan terlebih dahulu supaya dapat diarahkan pada server heroku yang tersedia. Penggunaan *port* untuk mengarahkan pada tempat yang bisa dipakai supaya aplikasi dapat diakses secara global pada internet. Apabila semua sudah sesuai dan tidak ada kendala maka semua akan masuk pada heroku dengan tampilan seperti dibawah ini, akan muncul berhasil atau *succeeded* apabila tidak terjadi kendala, dan apabila terjadi kendala akan muncul info bahwa aplikasi *failed*.



Gambar 4.20 Info Deploy Heroku

4.2 Pengujian Sistem

Pengujian pada penelitian ini terdiri dari dua macam pengujian, yaitu yang pertama pengujian terhadap pemakaian aplikasi dari pengguna menggunakan pengujian *black box* dan pengujian yang kedua terhadap model menggunakan *confusion matrix*.

4.2.1 Pengujian Black Box

- Daftar nama penguji

Tabel 4.4 Daftar Nama Penguji

No	Nama Responden	Keterangan
1.	Yanik Kurniawati	Ibu rumah tangga
2.	Witono	Pegawai Swasta
3.	Faizah Novi Widyani	Pegawai Negri Sipil
4.	Nabila Puteri Widyani	Mahasiswa
5.	Arsyad Finand	Mahasiswa
6.	Alek Setya Nugroho	Mahasiswa
7.	Maudya Kumhaida Savitri	Mahasiswa
8.	Ahmad Dzakiyyul Fuad	Mahasiswa
9.	Muhammad Fathur Pitayandanu	Mahasiswa
10.	Uyun Navita	Mahasiswa

- Pengujian

tabel 4.5 Sistematika Pengujian

No	Halaman	Detail Pengujian	Pengujian	
			Berhasil	Gagal
1	Halaman Utama	Tampil saat pertama kali membuka aplikasi	6/6	-
		Menampilkan menu-menu yang tersedia	6/6	-
2	Upload File	Mengakses file lokal	6/6	-
		Mengunggah gambar	6/6	-
3	Unduh Sample Data Uji	Berpindah ke lokasi google drive	6/6	-
		Memilih kelas varietas jati	6/6	-
		Memilih gambar contoh data uji	6/6	-
4	Hasil Klasifikasi	Menampilkan gambar yang diunggah	6/6	-
		Menampilkan kelas varietas	6/6	-
		Menampilkan probabilitas kelas	6/6	-

4.2.2 Confusion Matrix

Pengujian menggunakan *confusion matrix* merupakan pengujian yang dilakukan untuk mengetahui tingkat keberhasilan dari suatu arsitektur dalam model yang sudah dibuat. Pengukuran tingkat keberhasilan dengan melihat parameter nilai akurasi, *recall*, dan *f1-score*. Parameter-parameter tersebut dihitung berdasarkan nilai yang terdapat pada *confusion*

matrix diantaranya ada *true positive*(TP) , *true negative*(TN), *false positive*(FP), dan *false negative*(FN). Berikut merupakan langkah-langkahnya.

a. Inisiasi library

```
Import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from PIL import Image
from io import BytesIO
from google.colab import files
from google.colab import drive
from keras.preprocessing import image
from tensorflow.keras.utils import plot_model
from tensorflow.keras.models import load_model
%matplotlib inline
from sklearn.metrics import confusion_matrix, classification_report
```

Modul Program 4.40 Inisiasi Library Pengujian

Seperti biasa deklarasikan terlebih dahulu *library-libray* yang akan digunakan, diantaranya ada tensorflow untuk pengolahan model, os untuk mengkases lokasi sistem, numpy untuk operasi matematika, pandas untuk operasi data, seaborn untuk visualisasi, matplotlib untuk visualisasi, PIL untuk menampilkan gambar, dan google colab untuk menghubungkan menuju google drive.

b. Preprocessing

```
dimensi_gambar = (256, 256)
channel = (3,)
input_shape = dimensi_gambar + channel
labels = ['Mega', 'PH1', 'Plus', 'Tidak Diketahui']
print(labels)

def preprocess(gambar, dimensi_gambar):
    nimg = gambar.convert('RGB').resize(dimensi_gambar, resample= 0)
    #print(nimg)
    img_arr = (np.array(nimg))/255
    #print(img_arr)
    return img_arr
def reshape(imgs_arr):
    #bantu = np.stack(imgs_arr, axis=0)
    #print(bantu)
    return np.stack(imgs_arr, axis=0)
```

Modul Program 4.40 Persiapan Sebelum Pengujian

Data gambar yang akan diuji terlebih dahulu disesuaikan dengan data latih, dengan memberikan ukuran dimensi yang sama besar yaitu 256x256 pixel, memiliki 3 *channel* warna(RGB), merubah skala menjadi nol hingga satu, dan diratakan satu dimensi.

c. Load model

```
drive.mount('/content/drive')
MODEL_PATH           =           '/content/drive/My Drive/Colab 3/Learning Notebooks/Makarim/Model/Percobaan Rate/L00005/upgrade/L00005_model_bagus_v3.h5'
model = load_model(MODEL_PATH,compile=False)
model.summary()
```

Modul Program 4.41 Memanggil Model

Setelah persiapan data telah dilakukan selanjutnya memanggil model data latih yang telah dibuat, akses google drive terlebih dahulu kemudian arahkan pada model tersebut. Lihat terlebih dahulu model yang dipanggil sudah sesuai atau belum dengan summary.

d. Inisiasi jawaban benar

```
jawaban_benar = pd.DataFrame()
jawaban_benar['nama'] = os.listdir('/content/drive/My Drive/Data Testing Heroku')
kategori           =           ['3','3','3','3','3','3','3','3','3','3',
'3','3','3','3','3','3','3','3','3','3','3','3','3','3','3','3',
'3','3','3','3','3','3','3','3','3','3','3','3','3','3','3','3',
'3','3','3','3','3','3','3','3','3','3','3','3','3','3','3','3',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
'1','1','1','1','1','1','1','1','1','1','1','1','1','1','1','1',
'1','1','1','1','1','1','1','1','1','1','1','1','1','1','1','1',
'1','1','1','1','1','1','1','1','1','1','1','1','1','1','1','1',
'1','1','1','1','1','1','1','1','2','2','2','2','2','2','2','2',
'2','2','2','2','2','2','2','2','2','2','2','2','2','2','2','2',
'2','2','2','2','2','2','2','2','2','2','2','2','2','2','2','2',
'2','2','2','2','2','2','2','2','2','2','2','2','2','2','2','2',
'2','2','2','2','2','2','2','2','2','2','2','2','2','2','2','2']
```

Modul Program 4.42 Deklarasi Varibel

Gunakan variabel jawaban sebagai patokan jawaban yang sesungguhnya, jadikan jawaban benar menjadi sebuah *dataframe* dengan menggunakan *library* pandas. Arahkan lokasi penyimpanan data uji pada google drive untuk bisa menggunakan semua data uji yang sudah disiapkan.

e. Inisiasi data uji

```
labels_bantu = ['0','1','2','3']
data_testing = pd.DataFrame()
data_testing['nama'] = os.listdir('/content/drive/My Drive/Data Testing Heroku')
```

Modul Program 4.43 Inisiasi Data Uji

Sama halnya jawaban benar, maka kali data uji harus dikenali terlebih dahulu dan diarahkan pada lokasi data uji yang ada pada google drive.

f. Proses pengujian confusion matrix

```
tampung = []
for fn in data_testing['nama']:

    path = ('/content/drive/My Drive/Testing Jati/' + fn)
    #predicting images
    img = image.load_img(path, target_size=(256,256))
    #imgplot = plt.imshow(img)
    im = Image.open(path)
    X = preprocess(im, dimensi_gambar)
    X = reshape([X])
    y = model.predict(X)
    if np.max(y) < 0.5:
        tampung.append('-')
    elif np.max(y) > 0.5:
        print(y)
        print(labels_bantu[np.argmax(y)])
        tampung.append(labels_bantu[np.argmax(y)])
        print(fn)
        print('-----')
```

Modul Program 4.44 Proses Klasifikasi Data

Sama halnya dengan pengklasifikasian kelas, namun kali jumlahnya sangat banyak karena jumlah data uji yang begitu banyak hingga mencapai duaratus duapuluhan data uji maka perlu dilakukan perulangan pengujian sebanyak data yang ada pada lokasi penyimpanan.

Tabel 4.6 Hasil Klasifikasi Data Uji

```
[[1.0774006e-06 2.7054220e-08 8.7656730e-07 9.9999797e-01]]
3
20210105_073754.jpg
-----
[[8.4224790e-03 2.7865465e-09 4.1518451e-06 9.9157339e-01]]
3
20210105_073748.jpg
-----
[[5.7186463e-07 5.4943537e-12 1.5484944e-09 9.9999940e-01]]
3
20210105_073742.jpg
-----
[[4.2613260e-09 4.4194862e-14 5.6635391e-11 1.0000000e+00]]
3
```

Tabel 4.7 Lanjutan Hasil Klasifikasi Data Uji

20210105_073733.jpg	-----
[[7.7253247e-08 2.1474484e-11 5.1819566e-09 9.9999988e-01]]	3
20210105_073821.jpg	-----
[[6.2466525e-09 2.7267242e-12 2.9871478e-10 1.0000000e+00]]	3
20210105_073810.jpg	-----
[[3.9588117e-06 1.9715510e-13 1.2497260e-09 9.9999607e-01]]	3
20210105_073806.jpg	-----
[[7.8084497e-11 1.0698738e-12 4.2999895e-11 1.0000000e+00]]	3
20210105_073900.jpg	-----
[[3.6486133e-03 9.4571266e-09 6.1048161e-07 9.9635077e-01]]	3
20210105_073832.jpg	-----
[[1.3965049e-05 2.6451751e-12 1.5440145e-09 9.9998605e-01]]	3
20210105_073829.jpg	-----
[[2.4538292e-07 3.7162900e-13 4.7198218e-11 9.9999976e-01]]	3
20210105_073825.jpg	-----
[[4.7423929e-02 4.6106313e-11 1.2934727e-07 9.5257598e-01]]	3
20210105_073929.jpg	-----
[[5.7142717e-03 3.0537453e-10 1.3849072e-07 9.9428564e-01]]	3
20210105_073923.jpg	-----
[[8.7138674e-10 2.7073772e-13 1.3413710e-10 1.0000000e+00]]	3
20210105_073914.jpg	-----
[[7.5014075e-03 4.0357349e-07 1.2390153e-06 9.9249697e-01]]	3
20210105_073855.jpg	-----
[[1.9999939e-05 7.2319180e-13 1.4064476e-08 9.9997997e-01]]	3
20210105_073914.jpg	-----
[[7.5014075e-03 4.0357349e-07 1.2390153e-06 9.9249697e-01]]	3
20210105_073855.jpg	-----
[[1.9999939e-05 7.2319180e-13 1.4064476e-08 9.9997997e-01]]	3
[[1.5950814e-06 4.5353197e-09 3.1786300e-08 9.9999833e-01]]	3
20210105_073946.jpg	

Tabel 4.8 Lanjutan Hasil Klasifikasi Data Uji

```
-----[[1.7532589e-14 1.6318316e-17 8.4786954e-14 1.0000000e+00]]  
3  
20210105_073942.jpg  
-----[[2.8763668e-07 3.1033962e-12 4.8246925e-08 9.9999964e-01]]  
3  
20210105_074054.jpg  
-----[[6.6557813e-01 1.1834677e-04 8.3885221e-03 3.2591495e-01]]  
0  
20210105_074043.jpg  
-----[[7.997068e-11 5.818848e-12 6.453378e-10 1.000000e+00]]  
3  
20210105_074050.jpg  
-----[[1.0350553e-05 6.3850279e-14 4.6117581e-08 9.9998963e-01]]  
3  
20210105_074009.jpg  
-----[[1.6575625e-04 3.0902787e-12 7.5561593e-09 9.9983418e-01]]  
3  
20210105_074154.jpg  
-----[[1.5147053e-10 1.3747446e-11 4.2308565e-10 1.0000000e+00]]  
3  
20210105_074120.jpg  
-----[[1.8731873e-08 1.1584042e-11 2.5571492e-10 1.0000000e+00]]  
3  
20210105_074115.jpg  
-----[[2.1938521e-03 1.1510679e-13 5.6187246e-06 9.9780053e-01]]  
3  
20210105_074105.jpg  
-----[[1.3574360e-06 1.4328111e-10 1.0743918e-06 9.9999762e-01]]  
3  
20210105_074058.jpg  
-----[[2.8507287e-07 1.3346929e-06 6.2853062e-05 9.9993551e-01]]  
3  
20210105_074305.jpg  
-----[[9.61003389e-05 7.10696042e-07 1.05771884e-04 9.99797404e-01]]  
3  
20210105_074253.jpg  
-----[[1.3239498e-01 2.0470133e-04 2.8247604e-02 8.3915263e-01]]  
3  
20210105_074221.jpg  
-----[[4.115701e-07 5.713810e-06 7.893692e-05 9.999149e-01]]  
3  
20210105_074214.jpg  
-----[[4.6160372e-07 1.8457862e-10 4.2803188e-07 9.9999917e-01]]  
3
```

Tabel 4.9 Lanjutan Hasil Klasifikasi Data Uji

20210105_074428.jpg

[[5.3881124e-08 7.7345277e-09 6.9696216e-05 9.9993026e-01]]
3
20210105_074328.jpg

[[2.0947954e-08 3.0521204e-16 4.3757531e-10 1.0000000e+00]]
3
20210105_074318.jpg

[[6.5605910e-08 4.1877685e-10 2.4377787e-05 9.9997556e-01]]
3
20210105_074312.jpg

[[5.272036e-11 6.191098e-15 2.651415e-12 1.000000e+00]]
3
20210105_074511.jpg

[[4.80485998e-11 7.12840800e-16 1.02364665e-10 1.00000000e+00]]
3
20210105_074520.jpg

[[3.2469585e-12 1.2879942e-15 2.4772038e-09 1.0000000e+00]]
3
20210105_074438.jpg

[[7.3040520e-09 5.4538829e-10 4.1497486e-08 1.0000000e+00]]
3
20210105_074433.jpg

[[3.35943732e-05 1.16654959e-07 1.04259314e-04 9.99861956e-01]]
3
20210105_073340.jpg

[[1.3437005e-07 8.4124839e-11 9.0355911e-08 9.9999976e-01]]
3
20210105_073332.jpg

[[5.4749488e-13 2.9613492e-20 2.0157172e-12 1.0000000e+00]]
3
20210105_073247.jpg

[[1.2227528e-12 7.7897785e-19 6.5066805e-15 1.0000000e+00]]
3
20210105_073241.jpg

[[7.1743754e-16 8.9283683e-19 2.0370926e-14 1.0000000e+00]]
3
20210105_073416.jpg

[[9.243513e-08 6.452034e-06 5.354620e-06 9.999881e-01]]
3
20210105_073404.jpg

[[3.8203680e-08 1.8160396e-12 1.5965134e-07 9.9999976e-01]]
3
20210105_073352.jpg

[[7.9873527e-05 1.3669604e-06 3.9725241e-05 9.9987900e-01]]

Tabel 4.10 Lanjutan Hasil Klasifikasi Data Uji

3	20210105_073347.jpg	----- [[4.5879826e-02 4.4411810e-10 1.1684179e-04 9.5400339e-01]]
3	20210105_073456.jpg	----- [[4.209159e-04 7.587415e-07 7.837595e-05 9.994999e-01]]
3	20210105_073500.jpg	----- [[1.1234904e-14 4.6067250e-19 1.1002015e-15 1.0000000e+00]]
3	20210105_073432.jpg	----- [[3.0512109e-17 3.8528803e-20 1.4376344e-15 1.0000000e+00]]
3	20210105_073422.jpg	----- [[3.0393348e-06 4.4343676e-14 3.4465997e-10 9.9999702e-01]]
3	20210105_073517.jpg	----- [[1.01720964e-07 1.17415148e-14 7.79504170e-11 9.99999881e-01]]
3	20210105_073512.jpg	----- [[6.5355055e-02 7.2192363e-10 2.0682386e-05 9.3462425e-01]]
3	20210105_073506.jpg	----- [[1.4370588e-11 3.2026486e-12 9.4917785e-10 1.0000000e+00]]
3	20210105_073652.jpg	----- [[8.2047954e-08 5.8554850e-09 3.6254977e-09 9.9999988e-01]]
3	20210105_073648.jpg	----- [[6.4236425e-15 3.0446186e-17 1.6314234e-14 1.0000000e+00]]
3	20210105_073607.jpg	----- [[1.7901783e-08 1.8597923e-14 4.1094816e-09 1.0000000e+00]]
3	20210105_073558.jpg	----- [[5.5703940e-04 7.6806350e-10 1.7782810e-07 9.9944276e-01]]
0	20200324_104419.jpg	----- [[9.9980634e-01 2.6825811e-07 4.2475960e-05 1.5102398e-04]]
0	20200324_104830.jpg	----- [[9.9913174e-01 2.6657281e-08 4.4091299e-05 8.2419638e-04]]

Tabel 4.11 Lanjutan Hasil Klasifikasi Data Uji

[[9.9997938e-01 4.4169246e-07 1.9288647e-05 7.8871045e-07]]
0
20200324_105118.jpg

[[9.9942905e-01 6.3131637e-08 5.7011907e-04 7.2592138e-07]]
0
20200324_110109.jpg

[[9.9990201e-01 1.9735564e-06 7.5700955e-05 2.0254040e-05]]
0
20200324_111915.jpg

[[8.9051926e-01 2.4641049e-06 7.0433272e-03 1.0243496e-01]]
0
20200324_110120.jpg

[[9.99984622e-01 5.89538338e-08 1.60054503e-06 1.37832985e-05]]
0
20200324_111610.jpg

[[9.9999976e-01 4.5251580e-12 2.3348221e-07 3.3682557e-08]]
0
20200324_111908.jpg

[[9.99981880e-01 4.35252971e-08 6.72383385e-06 1.13994265e-05]]
0
20200324_111948.jpg

[[9.9999559e-01 9.7546016e-09 4.4618637e-06 4.1133394e-08]]
0
20200324_111954.jpg

[[9.9999440e-01 3.8332526e-08 5.2893488e-06 3.6387169e-07]]
0
20200324_112004.jpg

[[9.8252523e-01 7.1784324e-04 1.4986514e-02 1.7703964e-03]]
0
20200324_112016.jpg

[[9.9998617e-01 2.1113722e-09 1.3414877e-05 3.4259449e-07]]
0
20200324_112104.jpg

[[9.9990904e-01 1.4512336e-08 5.7303937e-06 8.5175350e-05]]
0
20200324_112052.jpg

[[9.9968016e-01 7.3017550e-06 2.3336854e-04 7.9202866e-05]]
0
20200324_112040.jpg

[[9.9996316e-01 1.7625105e-08 2.1634196e-05 1.5251490e-05]]
0
20200324_112109.jpg

[[9.9984109e-01 2.2229333e-11 1.5890358e-04 2.7022336e-08]]
0
20200324_112033.jpg

Tabel 4.12 Lanjutan Hasil Klasifikasi Data Uji

```
-----[[5.1718433e-03 7.8590875e-03 9.8695159e-01 1.7358239e-05]]  
2  
20200527_10154922.jpg  
-----[[9.9994051e-01 7.6230695e-11 5.9352802e-05 8.6355136e-08]]  
0  
20200324_112119.jpg  
-----[[9.9978691e-01 8.4209396e-06 1.6307217e-04 4.1640604e-05]]  
0  
20200324_112133.jpg  
-----[[9.9999714e-01 8.3901441e-10 2.1435480e-06 7.3760259e-07]]  
0  
20200324_112746.jpg  
-----[[9.8983824e-01 3.6030827e-04 6.3024840e-05 9.7384183e-03]]  
0  
20200324_112803.jpg  
-----[[9.9954128e-01 2.3854420e-06 4.4229405e-04 1.4096845e-05]]  
0  
20200324_112816.jpg  
-----[[9.9999964e-01 4.6610649e-10 1.2680691e-07 2.9105277e-07]]  
0  
20200324_112826.jpg  
-----[[9.3697125e-01 6.1899931e-03 5.6815587e-02 2.3248440e-05]]  
0  
20200324_112831.jpg  
-----[[9.978569e-01 1.400927e-03 6.541793e-04 8.808508e-05]]  
0  
20200324_112838.jpg  
-----[[9.9996352e-01 1.9250355e-09 3.9911429e-06 3.2493233e-05]]  
0  
20200324_112851.jpg  
-----[[9.9935144e-01 3.5605077e-05 4.4007556e-05 5.6895212e-04]]  
0  
20200324_112859.jpg  
-----[[9.9998832e-01 6.8025288e-09 3.6290572e-07 1.1267251e-05]]  
0  
20200324_112908.jpg  
-----[[1.0000000e+00 7.1200927e-12 3.1470847e-08 4.2193285e-08]]  
0  
20200324_112917.jpg  
-----[[9.9999654e-01 5.5770788e-10 3.2839378e-06 1.2225395e-07]]  
0  
20200324_112924.jpg  
-----[[9.9996591e-01 2.5881775e-09 9.8158878e-08 3.3992412e-05]]  
0
```

Tabel 4.13 Lanjutan Hasil Klasifikasi Data Uji

0	20200324_112933.jpg	-----	[[9.9986541e-01 1.0610567e-05 6.5298875e-05 5.8671572e-05]]
0	20200324_112937.jpg	-----	[[9.9999809e-01 2.6260929e-09 1.4132307e-06 4.5848691e-07]]
0	20200324_112947.jpg	-----	[[9.9997365e-01 7.9234006e-08 1.8104012e-05 8.1577446e-06]]
0	20200324_112953.jpg	-----	[[9.9999547e-01 4.4636542e-10 3.7107625e-06 8.7117974e-07]]
0	20200324_113004.jpg	-----	[[9.9994004e-01 1.1705464e-06 5.7573390e-05 1.1767729e-06]]
0	20200324_113013.jpg	-----	[[9.9987853e-01 1.9715558e-06 5.9585673e-05 5.9778224e-05]]
0	20200324_113027.jpg	-----	[[9.9999845e-01 1.2718145e-11 2.4470292e-08 1.5060147e-06]]
0	20200324_113039.jpg	-----	[[9.9998498e-01 6.5009897e-07 6.6375392e-06 7.7606610e-06]]
0	20200324_113043.jpg	-----	[[9.9577123e-01 1.4524630e-03 1.8452872e-04 2.5916905e-03]]
0	20200324_113106.jpg	-----	[[1.6629877e-02 6.7187351e-01 3.1146419e-01 3.2437478e-05]]
1	20200324_113113.jpg	-----	[[9.9994147e-01 2.2854540e-09 5.7466116e-07 5.7979334e-05]]
0	20200324_113124.jpg	-----	[[9.9999583e-01 2.2416115e-08 8.7531572e-07 3.2956568e-06]]
0	20200324_113136.jpg	-----	[[9.999933e-01 3.029002e-08 5.631690e-06 1.084614e-06]]
0	20200324_113147.jpg	-----	[[9.9998021e-01 5.8986640e-08 9.4163979e-06 1.0350425e-05]]
	20200324_113152.jpg	-----	[[9.9999404e-01 1.6593094e-10 5.8587389e-06 7.4799459e-08]]

Tabel 4.14 Lanjutan Hasil Klasifikasi Data Uji

0	20200324_113205.jpg	-----
		[[9.9999726e-01 3.0666412e-09 1.7456033e-06 9.3481850e-07]]
0	20200324_113215.jpg	-----
		[[9.9977988e-01 1.1062753e-08 1.5851471e-04 6.1578670e-05]]
0	20200324_113225.jpg	-----
		[[9.1355687e-01 7.4826960e-07 8.6440966e-02 1.4079677e-06]]
0	20200324_113235.jpg	-----
		[[1.22399115e-05 9.98535872e-01 1.31941610e-03 1.32484740e-04]]
1	20200423_095239.jpg	-----
		[[9.9999201e-01 3.4828360e-10 7.8739813e-06 1.5561739e-07]]
0	20200324_113241.jpg	-----
		[[3.9899696e-06 9.9969923e-01 2.9428446e-04 2.5965878e-06]]
1	20200423_095321.jpg	-----
		[[8.2036686e-06 9.9986815e-01 7.0695696e-06 1.1653900e-04]]
1	20200423_100704.jpg	-----
		[[3.1973832e-05 9.9424875e-01 4.5355619e-03 1.1837392e-03]]
1	20200423_105803.jpg	-----
		[[4.5609627e-07 9.9999917e-01 4.1138426e-07 5.3419530e-08]]
1	20200423_110018.jpg	-----
		[[6.9820875e-04 7.4454367e-01 2.5472924e-01 2.8843400e-05]]
1	20200423_110120.jpg	-----
		[[6.0317712e-04 3.0013904e-01 6.9925481e-01 2.9695248e-06]]
2	20200423_110103.jpg	-----
		[[6.4636075e-01 3.3732715e-01 8.7899760e-05 1.6224220e-02]]
0	20200423_110033.jpg	-----
		[[9.9619734e-01 3.1670346e-04 3.1632781e-03 3.2268726e-04]]
0	20200423_110048.jpg	-----
		[[6.1117915e-09 1.0000000e+00 3.0046763e-08 4.6054506e-08]]
1	20200423_110133.jpg	

Tabel 4.15 Lanjutan Hasil Klasifikasi Data Uji

```
-----[[1.1148933e-04 9.9986732e-01 2.0734440e-05 5.1437735e-07]]  
1  
20200423_110220.jpg  
-----[[1.41190385e-05 9.99869823e-01 1.88974336e-05 9.71914124e-05]]  
1  
20200423_110148.jpg  
-----[[4.6647463e-05 9.9994922e-01 4.2171878e-06 1.1794465e-08]]  
1  
20200423_110325.jpg  
-----[[2.1548567e-06 9.9999750e-01 4.1708756e-07 9.0038721e-10]]  
1  
20200423_110226.jpg  
-----[[5.5636451e-06 9.9998796e-01 6.3844896e-06 1.0590155e-08]]  
1  
20200423_110231.jpg  
-----[[7.1899530e-07 9.9999857e-01 6.8342342e-07 2.7180165e-09]]  
1  
20200423_110259.jpg  
-----[[5.962974e-09 1.000000e+00 6.440970e-10 4.861132e-09]]  
1  
20200423_110307.jpg  
-----[[8.5167959e-04 4.8551522e-03 9.9429250e-01 7.4039565e-07]]  
2  
20200423_110330.jpg  
-----[[1.0740913e-09 9.9999976e-01 2.0752229e-10 2.1359614e-07]]  
1  
20200423_110338.jpg  
-----[[2.5268098e-06 9.9999678e-01 6.8998088e-07 1.1858084e-10]]  
1  
20200423_110343.jpg  
-----[[4.5383621e-07 9.9999940e-01 8.0719047e-08 1.2925797e-09]]  
1  
20200423_110352.jpg  
-----[[1.2635824e-06 9.9997187e-01 2.6220056e-05 6.5112602e-07]]  
1  
20200423_110358.jpg  
-----[[6.6050245e-07 9.9999583e-01 4.1807127e-09 3.4342922e-06]]  
1  
20200423_110408.jpg  
-----[[1.1425012e-04 9.9964559e-01 1.9000410e-04 5.0133582e-05]]  
1  
20200423_110418.jpg  
-----[[1.11322976e-07 9.99999762e-01 6.87385011e-08 2.27151054e-09]]  
1
```

Tabel 4.16 Lanjutan Hasil Klasifikasi Data Uji

```

20200423_110435.jpg
-----
[[3.2300657e-06 9.9999535e-01 1.2619176e-06 6.6591959e-08]]
1
20200423_110443.jpg
-----
[[4.4170349e-07 9.9998999e-01 7.8325556e-06 1.7020184e-06]]
1
20200423_110451.jpg
-----
[[1.8899243e-06 9.9996316e-01 3.4412929e-05 4.6369433e-07]]
1
20200423_111737.jpg
-----
[[8.7748896e-07 9.9999893e-01 2.2568041e-07 3.9019450e-09]]
1
20200423_111746.jpg
-----
[[2.4318958e-09 1.0000000e+00 7.7035844e-10 4.4848605e-08]]
1
20200423_111748.jpg
-----
[[6.6689245e-07 9.9999928e-01 1.2147764e-08 3.1648609e-10]]
1
20200423_111758.jpg
-----
[[1.3968406e-08 1.0000000e+00 5.3265253e-11 1.4038538e-09]]
1
20200423_111802.jpg
-----
[[1.0008652e-06 9.9999905e-01 8.2434664e-09 1.7803451e-09]]
1
20200423_111817.jpg
-----
[[4.4263325e-07 9.9999940e-01 1.0152282e-07 2.6717933e-10]]
1
20200423_111827.jpg
-----
[[1.7729773e-08 1.0000000e+00 2.1718848e-10 4.0269383e-13]]
1
20200423_111922.jpg
-----
[[3.4134777e-04 9.9963343e-01 1.8974999e-05 6.3201546e-06]]
1
20200423_112220.jpg
-----
[[7.0038368e-06 9.9903774e-01 9.2162361e-04 3.3535161e-05]]
1
20200423_114405.jpg
-----
[[4.4235403e-06 9.9985135e-01 3.4109999e-05 1.1013593e-04]]
1
20200423_114412.jpg
-----
[[2.5307925e-05 9.9544382e-01 4.0994541e-04 4.1209604e-03]]
1
20200423_114415.jpg
-----
```

Tabel 4.17 Lanjutan Hasil Klasifikasi Data Uji

[[1.32935775e-05 9.99848366e-01 1.35745737e-04 2.55779332e-06]]
1
20200423_114422.jpg

[[3.6213194e-07 9.9996340e-01 2.9530063e-05 6.6734665e-06]]
1
20200423_114440.jpg

[[3.4361742e-06 9.9999535e-01 1.1283706e-07 1.0623019e-06]]
1
20200423_114445.jpg

[[9.9924614e-07 9.9998701e-01 2.0955615e-06 9.9451399e-06]]
1
20200423_114449.jpg

[[3.0559838e-07 9.9998343e-01 5.3321533e-06 1.0882593e-05]]
1
20200423_114454.jpg

[[9.9946155e-06 9.9995220e-01 3.7134076e-05 5.8774339e-07]]
1
20200423_114623.jpg

[[6.2228779e-08 9.9999893e-01 1.7914434e-08 9.9304509e-07]]
1
20200423_115037.jpg

[[1.1995214e-08 1.0000000e+00 1.6042311e-08 2.0134005e-08]]
1
20200423_115211.jpg

[[2.0823235e-07 9.9999976e-01 1.0424174e-10 2.4684751e-11]]
1
20200423_115242.jpg

[[4.7707840e-07 9.9970812e-01 4.2371557e-06 2.8708114e-04]]
1
20200423_121228.jpg

[[7.3275583e-06 3.3092925e-03 9.9667871e-01 4.6289847e-06]]
2
20200527_095005.jpg

[[0.8851275 0.01177339 0.04683637 0.05626271]]
0
20200527_093603.jpg

[[1.9549763e-05 3.4345044e-07 9.9997997e-01 1.2837356e-07]]
2
20200527_095021.jpg

[[1.2167528e-01 8.9411708e-03 8.6935538e-01 2.8231370e-05]]
2
20200527_095207.jpg

[[6.2760550e-06 1.6851099e-04 9.9982399e-01 1.3241737e-06]]
2
20200527_095212.jpg

Tabel 4.18 Lanjutan Hasil Klasifikasi Data Uji

```
-----[[1.7114701e-04 2.4199949e-04 9.9957412e-01 1.2768022e-05]]  
2  
20200527_095310.jpg  
-----[[5.6539134e-10 8.4181753e-05 9.9991584e-01 1.4913723e-10]]  
2  
20200527_095344.jpg  
-----[[3.9761220e-05 1.5938067e-05 9.9994421e-01 1.2970365e-07]]  
2  
20200527_095443.jpg  
-----[[2.6225592e-07 1.2207355e-05 9.9998748e-01 4.5145423e-08]]  
2  
20200527_095544.jpg  
-----[[9.4488556e-11 5.8276342e-08 1.0000000e+00 1.1238866e-11]]  
2  
20200527_095559.jpg  
-----[[4.8521457e-07 6.4569758e-05 9.9993396e-01 9.1321795e-07]]  
2  
20200527_095748.jpg  
-----[[9.706641e-09 1.832703e-04 9.998160e-01 6.982023e-07]]  
2  
20200527_095757.jpg  
-----[[1.4233656e-06 2.9322950e-04 9.9970526e-01 9.6733942e-08]]  
2  
20200527_095912.jpg  
-----[[8.9075334e-09 1.0575150e-05 9.9998939e-01 1.2178174e-08]]  
2  
20200527_095916.jpg  
-----[[3.8721369e-06 4.7940712e-02 9.5205253e-01 2.9886730e-06]]  
2  
20200527_100248.jpg  
-----[[0.00067765 0.6543682 0.3395565 0.00539758]]  
1  
20200527_100547.jpg  
-----[[3.82744111e-05 1.20939754e-01 8.68920863e-01 1.01011368e-02]]  
2  
20200527_100554.jpg  
-----[[2.3874478e-08 2.1648455e-05 9.9997830e-01 5.4942677e-09]]  
2  
20200527_100558.jpg  
-----[[5.0799841e-05 3.1681186e-01 6.7959559e-01 3.5416791e-03]]  
2  
20200527_100601.jpg  
-----[[2.1330503e-01 1.0786035e-05 7.8668088e-01 3.3202089e-06]]  
2
```

Tabel 4.19 Lanjutan Hasil Klasifikasi Data Uji

20200527_100621.jpg

[[6.2114516e-08 1.0662360e-07 9.9999988e-01 5.2912591e-10]]
2
20200527_100612.jpg

[[1.16164316e-07 2.03346470e-04 9.99795616e-01 9.91703814e-07]]
2
20200527_100605.jpg

[[0.00197115 0.7200163 0.24565282 0.03235976]]
1
20200527_100616.jpg

[[2.3073815e-06 3.8677852e-05 9.9995863e-01 3.6343661e-07]]
2
20200527_100624.jpg

[[1.2596033e-03 2.1361653e-02 9.7729129e-01 8.7478584e-05]]
2
20200527_100631.jpg

[[1.4870318e-08 1.8739072e-05 9.9998128e-01 3.9584279e-08]]
2
20200527_100639.jpg

[[1.1072381e-04 2.8755746e-04 9.9957210e-01 2.9638617e-05]]
2
20200527_100641.jpg

[[8.8502895e-07 2.0601056e-03 9.9793786e-01 1.0298493e-06]]
2
20200527_100702.jpg

[[9.1222655e-06 1.8055767e-05 9.9997258e-01 2.2082821e-07]]
2
20200527_100648.jpg

[[1.6453952e-09 4.4931412e-06 9.9999547e-01 3.8983603e-09]]
2
20200527_100643.jpg

[[8.3947788e-07 2.0639009e-03 9.9792933e-01 5.9223512e-06]]
2
20200527_100653.jpg

[[0.00275301 0.08238272 0.9131504 0.00171383]]
2
20200527_100731.jpg

[[3.0794545e-05 3.0050010e-03 9.9696249e-01 1.6199605e-06]]
2
20200527_100712.jpg

[[2.3342909e-05 1.6601350e-02 9.8337144e-01 3.9159831e-06]]
2
20200527_100706.jpg

[[3.2764933e-07 1.6147150e-04 9.9983811e-01 1.1859975e-07]]

Tabel 4.20 Lanjutan Hasil Klasifikasi Data Uji

2	20200527_100714.jpg	----- [[2.8842096e-03 3.0959272e-04 9.9672061e-01 8.5511987e-05]]
2	20200527_100811.jpg	----- [[2.0067280e-07 8.4247444e-07 9.9999881e-01 6.7290344e-08]]
2	20200527_100801.jpg	----- [[5.5344701e-10 5.8596455e-05 9.9994135e-01 1.8590576e-08]]
2	20200527_100737.jpg	----- [[4.5959123e-08 9.3895842e-06 9.9999046e-01 1.0456293e-07]]
2	20200527_100805.jpg	----- [[8.0643280e-05 3.0685147e-02 9.6923143e-01 2.7559820e-06]]
2	20200527_100826.jpg	----- [[5.9746834e-04 1.2231817e-03 9.9811339e-01 6.5961503e-05]]
2	20200527_100820.jpg	----- [[5.6570448e-02 2.3913904e-05 9.4299257e-01 4.1315090e-04]]
2	20200527_100816.jpg	----- [[1.3587889e-03 9.5448077e-02 9.0318304e-01 1.0150658e-05]]
2	20200527_100822.jpg	----- [[2.0584851e-07 2.2167980e-03 9.9778277e-01 2.5119596e-07]]
2	20200527_100845.jpg	----- [[2.5940031e-08 6.5730563e-07 9.9999928e-01 6.8221069e-09]]
2	20200527_100837.jpg	----- [[2.5745374e-04 8.3320051e-02 8.9497101e-01 2.1451533e-02]]
2	20200527_100828.jpg	----- [[9.5749772e-07 4.4233033e-07 9.9999857e-01 1.6377925e-08]]
2	20200527_100842.jpg	----- [[2.2135939e-06 5.9140712e-07 9.9999714e-01 6.6825909e-09]]
2	20200527_0955592.jpg	----- [[7.2101392e-09 1.8669080e-05 9.9998128e-01 1.3044301e-08]]
2	20200527_0955442.jpg	-----

Tabel 4.21 Lanjutan Hasil Klasifikasi Data Uji

[[8.0522892e-05 1.6467779e-06 9.9991751e-01 3.8989006e-07]]
2
20200527_0950212.jpg

[[4.1031817e-05 3.0296464e-02 9.6940619e-01 2.5630265e-04]]
2
20200527_101310.jpg

[[4.3510067e-06 9.2688127e-04 9.9904293e-01 2.5881473e-05]]
2
20200527_1001062.jpg

[[5.1561562e-07 1.0724276e-02 9.8927301e-01 2.0816403e-06]]
2
20200527_0959162.jpg

[[2.3212926e-06 4.0568649e-03 9.9500614e-01 9.3466416e-04]]
2
20200527_0957572.jpg

[[1.0061282e-04 1.2969319e-04 9.9967432e-01 9.5383206e-05]]
2
20200527_0957482.jpg

[[5.2649397e-05 1.1725551e-05 9.9993563e-01 2.5441294e-08]]
2
20200527_1015492.jpg

[[3.1249490e-07 3.0100060e-04 9.9969876e-01 1.9101437e-08]]
2
20200527_1007062.jpg

[[5.5450486e-04 2.6747570e-04 9.9917787e-01 9.1811714e-08]]
2
20200527_1009492.jpg

[[1.8799466e-04 6.1279349e-03 9.8438823e-01 9.2958016e-03]]
2
20200527_1005472.jpg

[[1.2993481e-04 1.3587032e-06 9.9986839e-01 2.5411276e-07]]
2
20200527_10094922.jpg

Hasil klasifikasi nantinya akan berupa kelas yang sesuai dengan kemiripan pada model yang telah dibuat. Prediksi yang dilakukan pada data uji untuk *confusion matrix* memerlukan jumlah data uji yang banyak, pada pengujian ini menggunakan dua ratus dua puluh satu gambar data uji, apabila gambar yang diuji lebih mirip varietas jati mega akan termasuk kelas 0, apabila data uji lebih mirip varietas jati PH1 maka akan termasuk kelas 1, apabila lebih mirip varietas plus maka termasuk kelas 2, dan jika tidak diketahui termasuk kelas 3.

g. Cek Data Kosong

```
jawaban_testing['kategori'] = tampung
jawaban_testing.isna().sum()
```

Modul Program 4.45 Cek Data Kosong

Menghindari adanya hasil data uji kosong maka perlu dilihat terlebih dahulu apakah semua hasil sudah terisi atau belum, dengan menggunakan *library pandas package* isna maka dapat diketahui jumlah data kosongnya.

nama	0
kategori	0
dtype:	int64

Gambar 4.21 Hasil Pengecekan Data Kosong

Isi dari dataframe yang ada dapat dilihat pada gambar 4.21 menunjukkan tidak adanya data kosong.

h. Proses pengujian confusion matrix

```
figure = classification_report(kategori, jawaban_testing['kategori'])
figure
```

Modul Program 4.46 Proses Confusion Matrix

Tahap ini merupakan inti dari pengujian menggunakan *confusion matrix*, hasil pengujian klasifikasi yang sudah dilakukan sebelumnya dibandingkan dengan data jawaban yang benar. Menggunakan *library sklearn* dapat sangat mempermudah penerapan *confusion matrix*, maka hasilnya sebagai berikut.

Tabel 4.22 Hasil Pengujian Confusion Matrix

	Precision	Recall	F1-Score	Support
0	0.91	0.96	0.93	50
1	0.92	0.90	0.91	50
2	0.95	0.93	0.94	61
3	1.00	0.98	0.99	60
Accuracy			0.95	221
Macro avg	0.94	0.94	0.94	221
Weighted avg	0.95	0.95	0.95	221

Hasil dari pengujian *confusion matrix* berdasarkan parameter-parameter yaitu akurasi, *recall*, dan *f1-score* menunjukkan nilai-nilainya diatas 90% semua, ini menandakan bahwa susunan arsitektur cnn dapat sangat baik mempelajari pola data daun yang ada.

i. Fungsi visualisasi hasil pengujian

```
def plot_confusion_matrix(data, labels):

    sns.set(color_codes=True)
    plt.figure(1, figsize=(9, 6))

    plt.title("Confusion Matrix")

    sns.set(font_scale=1.4)
    ax = sns.heatmap(data, annot=True, cmap="YlGnBu", cbar_kws={'label': 'Scale'})

    ax.set_xticklabels(labels)
    ax.set_yticklabels(labels)

    ax.set(ylabel="True Label", xlabel="Predicted Label")
```

Modul Program 4.47 Visualisasi Hasil Confusion Matrix

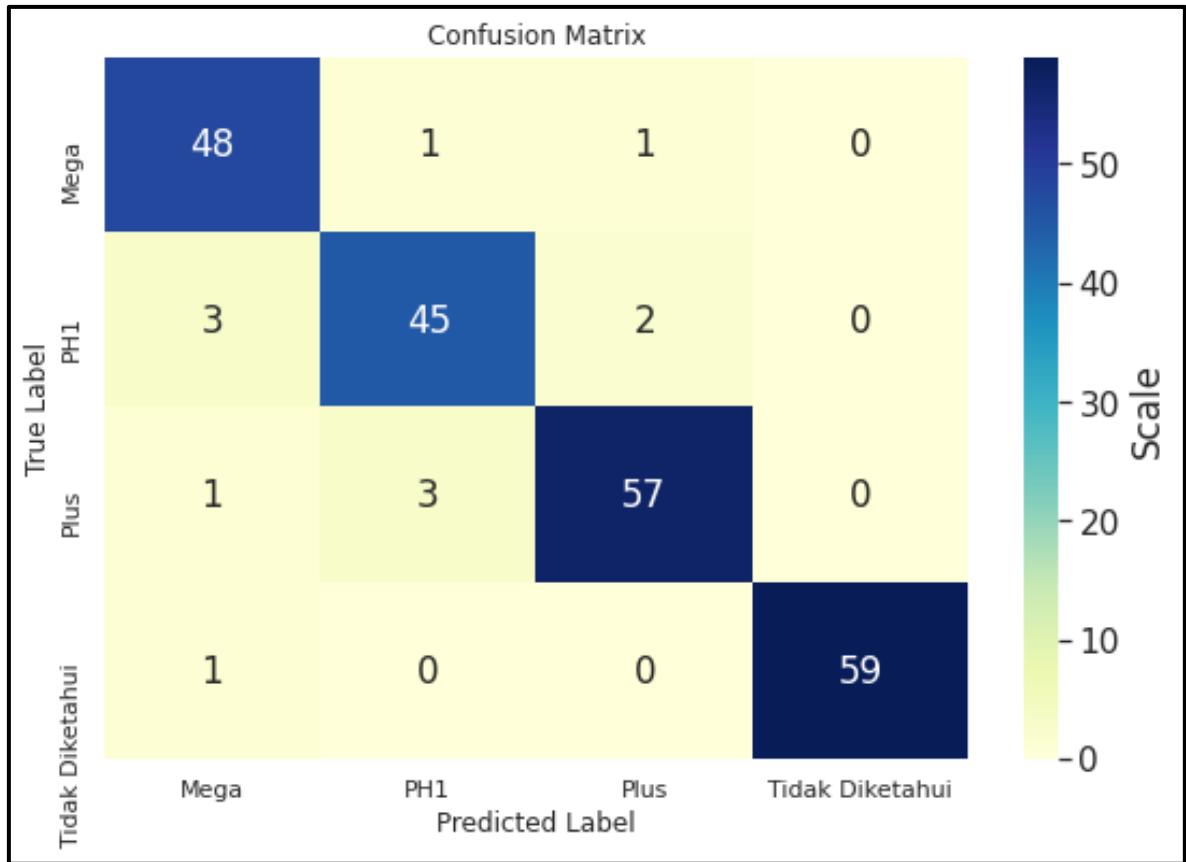
Menampilkan hasil data uji *confusion matrix* dapat mengevaluasi proses pengujian yang telah dilakukan, hasil pengujian yang kurang informatif dapat diubah menjadi visualisasi hasil pengujian yang lebih mudah untuk dilihat, oleh karena itu penggunaan *library* seaborn dengan memanfaatkan *heatmaps* akan dapat lebih mudah memetakan hasil dari data pengujian.

j. Visualisasi hasil pengujian

```
labels = ['Mega', 'PH1', 'Plus']
confusion_mat      = confusion_matrix(jawaban_benar['kategori'],
                                       submission['kategori'])
plot confusion matrix(confusion mat, labels)
```

Modul Program 4.4 Visualisasi Akhir

Setelah membuat prosedur visualisasi hasil pengujian *cofusion matrix* maka kali ini program penggunaanya, panggil prosedur yang sudah dibuat sebelumnya kemudia masukkan dengan jawaban yang benar dan hasil dari klasifikasi varietas data uji. Setelah dideklarasikan semua maka selanjutnya tampilkan hasilnya pada layar dengan menggunakan plotting.



Gambar 4.22 Hasil Akhir Confusion Matrix

BAB V

PENUTUP

5.1 Kesimpulan

Berawal dari analisis kemudian melakukan perancangan, pengambilan sampel, pembuatan, pengujian dan hingga implementasi, maka dalam penelitian ini dapat dikemukakan kesimpulan dari penulis sebagai berikut:

1. Hasil penelitian ini mampu menghasilkan aplikasi mengidentifikasi varietas jati berdasar daun dengan penerapan metode pembelajaran *convolutional neural network*.
2. Metode *convolutional neural network* mampu mengenali pola dan mengklasifikasikan dari ketiga varietas jati yaitu PH1, plus, dan mega dengan nilai akurasi 98%, precision 98%, dan recall 97%.
3. Arsitektur *convolutional neural network* yang dapat menghasilkan akurasi paling baik yaitu pada arsitektur ke lima, dengan menggunakan lima lapisan konvolusi, lima lapisan *pooling*, dan satu lapisan *fully connected*.
4. Perangkat keras yang digunakan dalam proses pelatihan mempengaruhi kecepatan dalam pembentukan model yang dibangun.
5. Kekurangan pada penelitian ini yaitu kondisi cuaca pada dataset yang kurang beragam.

5.2 Saran

Adapun saran dan masukan dari penulis yang dapat digunakan untuk pengembangan penelitian lebih lanjut sebagai berikut:

1. Pengembangan dapat dilakukan memperbanyak dataset dengan menambah macam varietas jati
2. Dalam mempelajari pola daun perlu memperhatikan *preprocessing* deteksi tepi supaya kedepannya dapat mendeteksi tepian dari daun ketika pengujian di lapangan.
3. Memperbanyak dataset dengan kondisi cuaca yang berbeda.

4. Platform yang digunakan bisa menggunakan aplikasi mobile untuk mempermudah pengujian di lapangan nantinya.

Daftar Pustaka

- Agustin M. (2012). Penggunaan Jaringan Syaraf Tiruan *Backpropagation* Untuk Seleksi Penerimaan Mahasiswa Baru Pada Jurusan Teknik Komputer Di Politeknik Negeri Sriwijaya. THESIS. Universitas Diponegoro Semarang.
- Ahmad, A. (2017). *Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning. October.*
- Ahyar, H., Maret, U. S., Andriani, H., Sukmana, D. J., Mada, U. G., Hardani, S.Pd., M. S., Nur Hikmatul Auliya, G. C. B., Helmina Andriani, M. S., Fardani, R. A., Ustiawaty, J., Utami, E. F., Sukmana, D. J., & Istiqomah, R. R. (2020). *Buku Metode Penelitian Kualitatif & Kuantitatif* (Issue March).
- Albelwi, S., & Mahmood, A. (2017). A framework for designing the architectures of deep Convolutional Neural Networks. *Entropy*, 19(6). <https://doi.org/10.3390/e19060242>.
- Al Khairi. (2008). Keragaman Genetik Jati Rakyat Di Jawa Berdasarkan Penanda Random Amplified Polymorphic DNA(RAPD). Bogor. Skripsi: Institut Pertanian Bogor.
- Andika, L. A., Pratiwi, H., & Handajani, S. S. (2019). *Lingga Aji Andika 1 , Hasih Pratiwi 2 , and Sri Sulistijowati Handajani 3 1.* 331–340.
- Bermejo, I., Cañellas, I., & San Miguel, A. (2004). Growth and yield models for teak plantations in Costa Rica. *Forest Ecology and Management*, 189(1–3), 97–110. <https://doi.org/10.1016/j.foreco.2003.07.031>.
- Brownlee J., (2019). Understand the Impact of Learning Rate on Neural Network Performance. Retrieved from medium website: <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>.
- Gani, jafar abdul. (2000). *kedelai Varietas unggul baru. 07.*
- Ghoneim, S. (2019, October 2). Accuracy, Recall, Precision, F-Score & Specificity, which to optimize on? Retrieved from Towards Data Science.
- Haryono, A., & Kustiyo, A. (2013). *Identifikasi Daun Tanaman Jati Menggunakan Jaringan Syaraf Tiruan Backpropagation dengan Ekstraksi Fitur Ciri Morfologi Daun.*
- Hidayat, B. (2018). Deteksi Hama Pada Daun Teh Dengan Metode Convolutional Neural Network (CNN). *SKRIPSI: Program Studi Teknik Informatika UNIKOM, 112.*
- Hijazi, S., Kumar, R., & Rowen, C. (2015). *What Is a CNN? Using Convolutional Neural Networks for Image Recognition.* 1–12.
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). *Improving Neural Networks By Preventing Co-Adaptation Of Feature Detectors.* 1–18. <Http://Arxiv.Org/Abs/1207.0580>.

- Ho, Y., & Wookey, S. (2020). The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling. *IEEE Access*, 8, 4806–4813. <https://doi.org/10.1109/ACCESS.2019.2962617>.
- Kamal Hasan, M., Adiwijaya, & Said, A. F. (2019). Klasifikasi Citra Multi-Kelas Menggunakan Convolutional Neural Network. *E-Proceeding of Engineering*, 6(1), 2127–2136.
- Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.
- Kusumaningrum, T.F. (2018). *Implementasi Convolutional Neural Network(CNN) untuk Klasifikasi Jamur Konsumsi Di Indonesia Menggunakan Keras*. SKRIPSI: Universitas Islam Indonesia, 300.
- Lina, Q. (2018). Apa itu *Convolutional Neural Network*? Retrieved from medium. website: <https://medium.com/@16611110/apa-itu-convolutional-neural-network-836f70b193a4>.
- Munir, R. (2004). Pengolahan Citra Digital dengan Pendekatan Algoritmik. Bandung: Informatika.
- Murtinah, V., Marjenah, M., Ruchaemi, A., & Ruhiyat, D. (2015). PERTUMBUHAN HUTAN TANAMAN JATI (*Tectona grandis* Linn.f.) DI KALIMANTAN TIMUR. *Agrifor*, 14(2), 287–292.
- Nugroho, Adi. (2009). Rekayasa Perangkat Lunak Menggunakan UML dan JAVA. Andi. Yogyakarta.
- Nurfita, R. D., Ariyanto, G., Learning, D., Network, C. N., & Pendahuluan, I. (2014). *IMPLEMENTASI DEEP LEARNING BERBASIS TENSORFLOW UNTUK PENGENALAN SIDIK JARI*.
- Nguyen, G., Dlugolinsky, S., Bobák, M., Tran, V., López García, Á., Heredia, I., Malík, P., & Hluchý, L. (2019). Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey. *Artificial Intelligence Review*, 52(1), 77–124. <https://doi.org/10.1007/s10462-018-09679-z>.
- Nour, E. (2018). *IMPLEMENTASI METODE CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI TANAMAN PADA CITRA RESOLUSI TINGGI* (*The Implementation of Convolutional Neural Network Method for Agricultural Plant Classification in High Resolution Imagery*). 61–68.
- Novyantika, R. D. (2018). *DETEKSI TANDA NOMOR KENDARAAN BERMOTOR PADA MEDIA STREAMING DENGAN ALGORITMA CONVOLUTIONAL NEURAL NETWORK MENGGUNAKAN Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Jurusan Statistika Disusun Oleh : Rizky Dwi Novyantika. March*.
- Putra, J. W. G. (2019). *Pengenalan Konsep Pembelajaran Mesin dan Deep Learning*. 4, 1–235.

Putra, S. R. (2015). *Implementasi Convolutional Neural Network Untuk Klasifikasi Objek Pada Citra*. Http://Repository.Its.Ac.Id/71292/1/5111100076-Undergraduate Thesis.Pdf

Rahmadewi, R., Efeline, V., Purwanti, E., & Ext, T. F. (n.d.). *IDENTIFIKASI JENIS TUMBUHAN MENGGUNAKAN CITRA DAUN BERBASIS JARINGAN SARAF TIRUAN (ARTIFICIAL NEURAL NETWORKS)*. VII(2), 38–43.

RD,Kusumanto Alan,Novi, T. (2011). Pengolahan Citra Digital Untuk Mendeteksi Warna Model Normalisasi Rgb. *Semantik*, 17(C), 329–332. [https://doi.org/10.1016/S0166-1116\(08\)71924-1](https://doi.org/10.1016/S0166-1116(08)71924-1).

Rohim, A., Sari, Y. A., & Tibyani. (2019). Convolution Neural Network (CNN) untuk Pengklasifikasian Citra Makanan Tradisional. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 3(7), 7038–7042. <http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/5851/2789>

Rokhana, R., Priambodo, J., Karlita, T., Sunarya, I. M. G., Yuniarno, E. M., Purnama, I. K. E., & Purnomo, M. H. (2019). Convolutional Neural Network untuk Pendekripsi Patah Tulang Femur pada Citra Ultrasonik B–Mode. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi (JNTETI)*, 8(1), 59. <https://doi.org/10.22146/jnteti.v8i1.491>

Schmuller, J. (1999). *Sams Teach Yourself UML in 24 Hours*, Third edition. Indianapolis : Sams Publishing.

Shafira T. (2018). *Implementasi Convolutional Neural Network untuk Klasifikasi Citra Tomat Menggunakan Keras*.[Skripsi]. Statistika Universitas Islam Indonesia.

Sri Kusumadewi. (2003). Artificial Intelligence. *Artificial Intelligence (Teknik Dan Aplikasinya)*.

Sugiyono.(2015). Metode Penelitian Kuantitatif, kualitatif dan R & D. (Cetakan ke-21) Bandung:Alfabeta.

Sukmadjaja, D., & Mariska, I. (2003). *ISBN 979-95627-8-3 Balai Penelitian Bioteknologi dan Sumberdaya Genetik Pertanian*.

Sutojo, T, E Mulyanto, V Suhartono.(2017). Teori Pengolahan Citra Digital. Semarang : Universitas Dian Nuswantoro.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4), 427–437. <https://doi.org/10.1016/j.ipm.2009.03.002>

Vedaldi, A., & Lenc, K. (2015). MatConvNet: Convolutional neural networks for MATLAB. *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*, 689–692. <https://doi.org/10.1145/2733373.2807412>.

Wibawa, M. S. (2017). Pengaruh Fungsi Aktivasi , Optimisasi dan Jumlah Epoch Terhadap Performa Jaringan Saraf Tiruan. *Jurnal Sistem Dan Informatika*, 11(2), 167–174. <https://doi.org/10.13140/RG.2.2.21139.94241>

- Wu, S. G., Bao, F. S., Xu, E. Y., Wang, Y.-X., Chang, Y.-F., & Xiang, Q.-L. (2007). A Leaf Recognition Algorithm for Plant Classification Using. *Probabilistic Neural Network*", *IEEE 7th Interantional Symposium on Signal Processing and Information Technology, November.*
- Yusniar, E., & Kustiyo, A. (2014). *Identifikasi Daun Shorea menggunakan KNN dengan Ekstraksi Fitur 2DPCA Shorea Leaves Identification using KNN with 2DPCA Feature Extraction.* 3.
- Yusri, A.C. 20019. Implementasi Jaringan Syaraf Tiruan dengan Deteksi Objek Kanker Kulit Melanoma menggunakan Metode Convolutional Neural Network [skripsi]. Yogyakarta (ID): Universitas Pembangunan Nasional "Veteran" Yogyakarta.
- Zed, M. (2004). *Koleksi Buku 2004 Zed , Mestika " Metode penelitian kepustakaan / Mestika Zed "* 2004. 2004.