

**IDENTIFIKASI BENTUK BIJI KOPI MENGGUNAKAN DESKRIPTOR BENTUK  
DASAR DAN JARINGAN SARAF TIRUAN**

**SKRIPSI**

Diajukan untuk Memenuhi Salah Satu Syarat  
Memperoleh Gelar Sarjana Komputer  
Program Studi Teknik Informatika

Oleh:

DANIEL EKA PUTRA RISAMASU

125314068

**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNIK INFORMATIKA  
FAKULTAS SAINS DAN TEKNOLOGI  
UNIVERSITAS SANATA DHARMA**

**2017**

**COFFEE BEAN SHAPE IDENTIFICATION USING BASIC REGION  
DESCRIPTORS AND ARTIFICIAL NEURAL NETWORK**

**A THESIS**

Presented as Partial Fullfillment of The Requirements  
to Obtain *Sarjana Komputer* Degree  
In Informatics Engineering Department

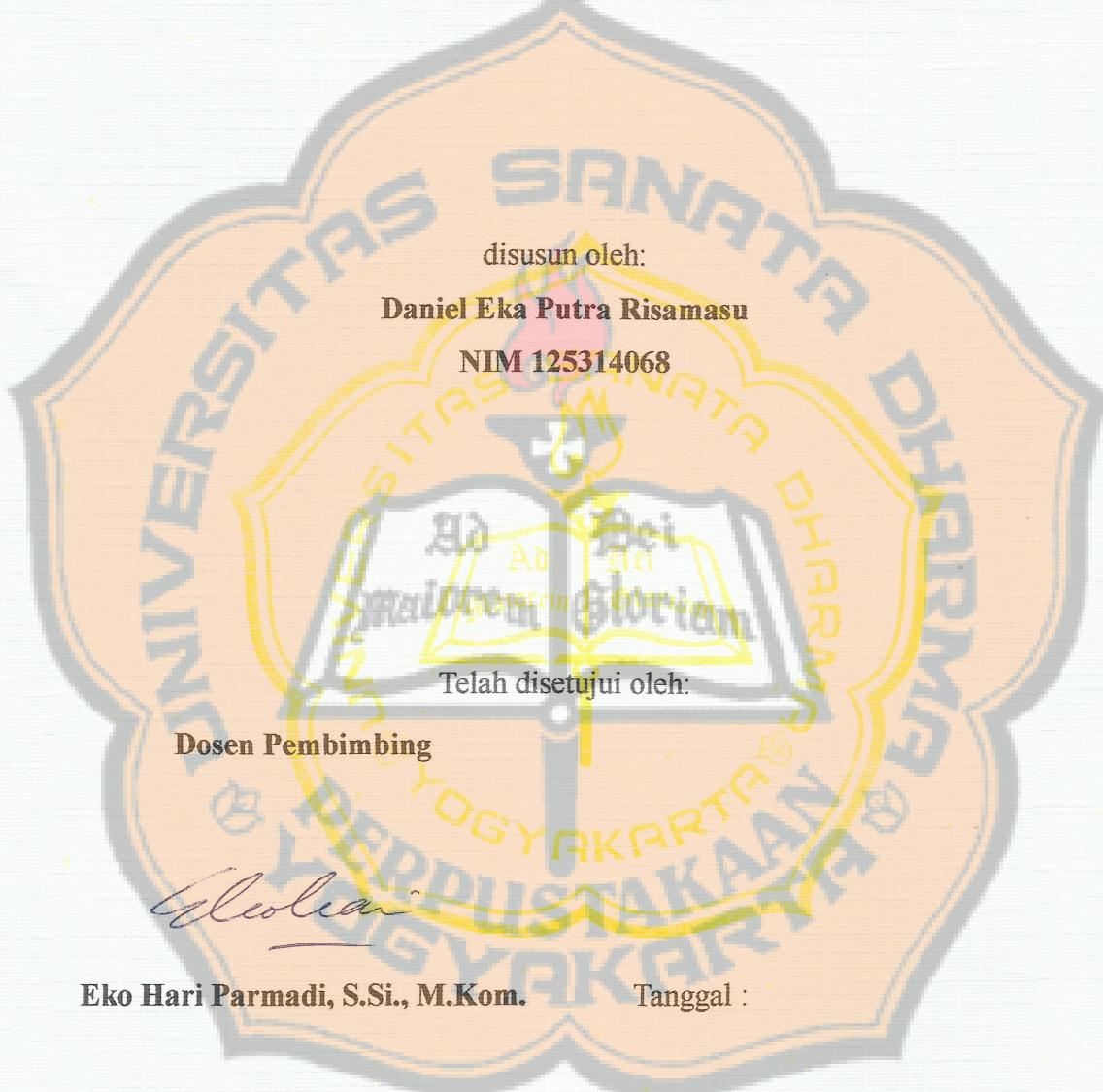


**INFORMATICS ENGINEERING STUDY PROGRAM  
DEPARTMENT OF INFORMATICS ENGINEERING  
FACULTY OF SCIENCE AND TECHNOLOGY  
SANATA DHARMA UNIVERSITY  
YOGYAKARTA**

**2017**

HALAMAN PERSETUJUAN  
SKRIPSI

IDENTIFIKASI BENTUK BIJI KOPI MENGGUNAKAN DESKRIPTOR BENTUK  
DASAR DAN JARINGAN SARAF TIRUAN



HALAMAN PENGESAHAN  
SKRIPSI

IDENTIFIKASI BENTUK BIJI KOPI MENGGUNAKAN DESKRIPTOR BENTUK  
DASAR DAN JARINGAN SARAF TIRUAN

dipersiapkan dan ditulis oleh:

Daniel Eka Putra Risamasu

NIM. 125314068

Telah dipertahankan di depan Panitia Pengaji

pada tanggal 18 Juli 2017

dan dinyatakan memenuhi syarat.

Susunan Panitia Pengaji

Nama Lengkap

Tanda Tangan

Ketua

: Alb. Agung Hadhiyatma, S.T., M.T.

Sekretaris

: Dr. Anastasia Rita Widiarti, M.Kom.

Anggota

: Eko Hari Parmadi, S.Si., M.Kom.

Yogyakarta, 22 Agustus 2017

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

Dekan



Sudi Mungkasi, S.Si., M.Math.Sc., Ph.D

**PERNYATAAN KEASLIAN KARYA**

Saya menyatakan dengan sesungguhnya bahwa skripsi yang saya tulis tidak memuat karya atau bagian karya orang lain, kecuali yang telah disebutkan dalam kutipan dan daftar pustaka sebagaimana layaknya karya ilmiah.

Yogyakarta, 18 Juli 2017

Penulis



Daniel Eka Putra Risamasu

*Malorem Gloriam*



**LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH  
UNTUK KEPENTINGAN AKADEMIS**

Saya yang bertandatangan di bawah ini, mahasiswa Universitas Sanata Dharma :

Nama : Daniel Eka Putra Risamasu

NIM : 125314068

Demi pengembangan ilmu pengetahuan, saya memberikan kepada Perpustakaan Universitas Sanata Dharma karya ilmiah saya yang berjudul :

**IDENTIFIKASI BENTUK BIJI KOPI MENGGUNAKAN DESKRIPTOR**

**BENTUK DASAR DAN JARINGAN SARAF TIRUAN**

Dengan demikian, saya memberikan kepada Perpustakaan Universitas Sanata Dharma hak untuk menyimpan, mengalihkan dalam bentuk media lain, mengelolanya dalam bentuk pengkalan data, mendistribusikan secara terbatas, dan mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya maupun memberikan royalti kepada saya selama tetap mencantumkan nama saya sebagai penulis.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di Yogyakarta

Pada tanggal 18 Juli 2017

Yang menyatakan,



Daniel Eka Putra Risamasu

## ABSTRAK

Biji kopi merupakan salah satu komoditas yang banyak diperdagangkan di era global saat ini. Biji kopi dihasilkan oleh tanaman kelompok genus *Coffea*, famili *Rubiaceace*. Biji kopi dihasilkan di lebih dari 70 negara tropis, salah satunya Indonesia. Dengan 70% produksi nasional kopi diekspor, perlu adanya penetapan standar mutu terhadap kualitas biji kopi. Hal ini diatur dalam Standar Nasional Indonesia no. 01-2907-2008

Salah satu standar mutu dalam SNI adalah penentuan biji kopi utuh dan biji kopi pecah. Biji kopi dikatakan pecah apabila ukuran tidak memenuhi  $\frac{3}{4}$  dari ukuran biji kopi yang utuh. Penelitian ini akan menggunakan deskriptor bentuk dasar yang terdiri dari luas, perimeter, panjang (diameter), lebar, rasio kebulatan, rasio kerampingan, dan fitur dispersi untuk melakukan identifikasi terhadap biji kopi utuh dan biji kopi pecah. Adapun ciri yang didapatkan akan dimasukkan dalam sebuah arsitektur jaringan saraf tiruan untuk mengenalinya

Hasil yang dicapai dari penelitian ini cukup baik dalam melakukan pengenalan terhadap biji kopi pecah dan biji kopi utuh, dengan tingkat akurasi tertinggi sebesar 97.35% atau 3 kesalahan dari 113 hasil indentifikasi.

**Kata Kunci** – Biji Kopi, Standar Nasional Indonesia, Deskriptor Bentuk Dasar, Jaringan Saraf Tiruan

## ABSTRACT

Coffee Beans was a most traded commodity in this global era. Coffee Bean was produced from Coffea genus and Rubiaceace family plants. This commodity was produced in 70 tropical countries, and Indonesia was one of them. With 70% national production was exported, it needs a standardization in quality measurement for this commodities. Which is was regulated in Indonesian National Standards (SNI) no. 01-2907-2008

One of quality standards that regulated was the determination between the broken beans and whole beans. The broken beans determined if the size was below  $\frac{3}{4}$  portion of the whole ones. This study will using basic region descriptors which consist of area, perimeter, length (diameter), width, circularity ratio, compactness ratio, and dispersion feature for coffee bean's shape identification. And those features will be inserted into artificial neural network architecture for identification process.

Achieved result from this study was good enough for coffee bean shape identification, with highest accuracy of 97.35% consist of 3 misidentification from 113 beans.

**Kata Kunci** – *Coffee Beans, Indonesian National Standards, Basic Region Descriptors, Artificial Neural Networks*

## KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa yang telah memberikan berkat dan karuniaNya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul “IDENTIFIKASI BENTUK BIJI KOPI MENGGUNAKAN DESKRIPTOR BENTUK DASAR DAN JARINGAN SARAF TIRUAN” ini. Penulisan skripsi ini diajukan sebagai syarat untuk memperoleh gelar Sarjana Komputer di Program Studi Teknik Informatika Universitas Sanata Dharma.

Penulis menyampaikan terimakasih kepada:

Eko Hari Parmadi, S.Si, S.Kom. (*Dosen Pembimbing*), Simon Johannes Harold Risamasu (*Ayah*), Winasih (*Ibu*), Andre Dwi Nugroho Risamasu dan Yohanes Kurniadi Risamasu (*Adik*). Selain itu untuk teman-teman penulis : Laurensia Eva, Octaviani, Elizabeth Febrina Cornelia, Raden Alexander Purbo Widianto, Kevindha Mahatma Arjun Conbravura, Yustinus Adrian Nada, Dian Saktian Tobias, Gregorius Hugo Himawan, Cahyo Wicaksono, Alvin Christianto Nugroho, Thomas Wiga Heru Prasetya, Anjar Nugraha Jati, Agustinus Komang Mariadi Saputra, Yosua Astutakari Gultom, Rekiyan Seto, Yohanes Baptis Christian Bayu, Nyoman Wisnu Wardana, Pius Juan Pratama, Henrycus Bagus Handoko, Romualdus Vanadio Yoga Setiawan, Stephanus Wijaya Nata Kusuma, Willybrodus Ranggga Khaisar Purnama, Young Queen Putra Nugroho, Mikael Aditya Wahyu, Michael Kevin, Dhesa Ardhyanta, Laurensius Haris Chrisanda, Lorencius Echo, Mikael Fajar Jati, Vina Puspitasasri, FX Dwi Kurniawan, Dyonisius Agung Wicaksono, Paulus Edi Gunawan, Tria Mahardhika, Hervian Jatmika Bimantoro, Laurensius Pandu Mahardika, Robertus Heru Santoso, Kadek Edo Jashinta, Lintang

Rigia Mukti, Ajeng Anggraeni Putri, Aloysius Rangga, Angela Priskalina Fridawanti, Cecilia Riskawati, Dina Febriyani, Guerika Yucky Fandera Widanna, Gusti Ayu Dara Bintang Kejora, Ken Sulanjari, Patricia Kiti Puspitaningrum, Patrisius Gerdian Bimawiratma, Monica Tri Irianti, Venny Valeria, Indah Rahayu, Engelbertus Vione, Engelbert Eric, Euclides Wahyu Nugroho, Vinsen Muliadi dan masih banyak dan tak tersebutkan, yang menjadi bagian dari dinamika penulis selama menjalani kehidupan akademik di Universitas Sanata Dharma, mulai dari komunitas Sanata Dharma Open Source, kepanitiaan-kepanitiaan, Dewan Perwakilan Mahasiswa, serta menjadi dinamika kehidupan pribadi penulis.

Penulis menyadari ketidak sempurnaan dari naskah ini. Oleh karena itu, penulis mengharapkan masukan dan saran yang membangun, sehingga dapat memperkaya kajian dan metode untuk topik yang masih memiliki penerapan teori dan metode yang sejenis. Semoga tugas akhir ini memiliki manfaat dan dapat berguna bagi pihak yang membutuhkan.

Yogyakarta, 18 Juli 2017

Penulis

## DAFTAR ISI

<b>Bab I PENDAHULUAN</b>	1
1.1.Latar Belakang.....	1
1.2.Rumusan Masalah.....	3
1.3.Maksud Tujuan Penelitian.....	3
1.4.Batasan Masalah.....	3
1.5.Sistematika Penulisan.....	5
 <b>Bab II LANDASAN TEORI</b>	 6
2.1.Kopi.....	6
2.2.Standar Nasional Indonesia No. 01-2907-2008.....	7
2.3.Citra Digital.....	11
2.4.Pemrosesan Citra	12
2.4.1. <i>Grayscale</i> .....	12
2.4.2. Deteksi Tepi : Sobel Operator .....	12
2.4.3. Operasi Morfologi : Dilasi .....	13
2.5.Ekstraksi Ciri : <i>Basic Region Descriptor</i>	14
2.5.1. Kontur.....	14
2.5.2. Kontur Internal.....	15
2.5.3. <i>Chain Code</i> .....	20
2.5.4. Ciri dari Penghitungan Kontur Internal dan <i>Chain Code</i>	26
2.5.4.1.Luas.....	26
2.5.4.2.Perimeter.....	32
2.5.4.3.Panjang, Lebar dan Diameter.....	33
2.5.5. Ciri dari Penghitungan Perimeter, Luas, Panjang, dan Lebar	37
2.5.5.1.Rasio Kebulatan.....	37
2.5.5.2.Rasio Kerampingan.....	38
2.5.6. Pusat Massa Objek ( <i>Centroid</i> ).....	39
2.5.7. Ciri dari Penghitungan <i>Centroid</i> dan Kontur	40
2.5.7.1.Dispersi.....	40
2.6.Jaringan Saraf Tiruan	42
2.6.1. Arsitektur Jaringan Saraf Tiruan.....	42
2.6.1.1.Jaringan Layar Tunggal.....	43
2.6.1.2.Jaringan Layar Jamak.....	43
2.6.2. Fungsi Aktivasi.....	44
2.6.3. Jaringan Saraf Tiruan <i>Backpropagation</i>	45
2.6.3.1.Arsitektur <i>Backpropagation</i> .....	45
2.6.3.2.Fungsi Aktivasi <i>Backpropagation</i> .....	45
2.6.4. Pelatihan Standar <i>Backpropagation</i> .....	47

<b>Bab III METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM</b>	49
3.1. Metodologi Penelitian.....	49
3.1.1. Gambaran Umum.....	49
3.1.2. Desain Penelitian	49
3.1.2.1. Studi Literatur.....	49
3.1.2.2. Data Penelitian.....	50
3.1.2.2.1. Kopi.....	50
3.1.2.2.2. Skenario Pengambilan Data.....	50
3.1.2.2.3. Hasil Pengelompokan.....	51
3.1.2.3. Metode Perancangan Alat Uji.....	51
3.1.3. Spesifikasi Perangkat Penelitian	52
3.1.3.1. Perangkat Keras.....	52
3.1.3.2. Perangkat Lunak.....	52
3.2. Perancangan Sistem Alat Uji	53
3.2.1. Alur Penggunaan Alat Uji.....	53
3.2.1.1. <i>Preprocessing Data</i> .....	53
3.2.1.2. Ekstraksi Ciri.....	54
3.2.1.3. Proses Klasifikasi dengan Jaringan Saraf Tiruan	55
3.2.1.3.1. Proses <i>Training</i> .....	55
3.2.1.3.2. Proses <i>Testing</i> .....	55
3.2.2. Diagram Konteks	56
3.2.2.1. DFD Level 1 Sisi Pengguna.....	56
3.2.2.2. DFD Level 2 Sisi Pengguna Proses Pelatihan JST .....	56
3.2.2.3. DFD Level 2 Sisi Pengguna Proses Testing .....	57
3.2.2.4. DFD Level 3 Sisi Pengguna <i>Preprocessing Citra</i> (1.1, 2.1) .....	57
3.2.2.5. DFD Level 3 Sisi Pengguna Proses Ekstraksi Ciri (1.2, 2.2) .....	58
3.2.3. Rancangan Implementasi <i>Preprocessing Data</i>	61
3.2.3.1. <i>Cropping Manual</i> .....	61
3.2.3.2. <i>Grayscale</i> .....	62
3.2.3.3. Deteksi Tepi Sobel.....	62
3.2.3.4. Dilasi <i>Structure Element</i> .....	62
3.2.3.5. <i>Image Filling</i> .....	63
3.2.3.6. <i>Image Center &amp; Resize</i> .....	63
3.2.3.7. Tepi Objek.....	64
3.2.4. Rancangan Implementasi Ekstraksi Ciri : <i>Basic Region Descriptor</i>	64
3.2.4.1. Pencarian Kontur.....	64
3.2.4.2. Pencarian Rantai Kode ( <i>Chain Code</i> ).....	65
3.2.4.3. Ciri 1 : Luas.....	65
3.2.4.4. Ciri 2 : Perimeter.....	66

3.2.4.5.Ciri 3, 4, 5 : Panjang Lebar, Diameter.....	66
3.2.4.6.Ciri 6 : Rasio Kebulatan.....	67
3.2.4.7.Ciri 7 : Rasio Kerampingan.....	67
3.2.4.8.Penghitungan <i>Centroid</i> .....	68
3.2.4.9.Ciri 8 dan 9 : Dispersi I dan Dispersi IR.....	69
3.2.5. Perancangan Jaringan Saraf Tiruan	69
3.2.5.1.Lapisan Tersembunyi.....	69
3.2.5.2.Neuron Tersembunyi.....	69
3.2.5.3.Fungsi Aktivasi.....	69
3.2.5.4.Fungsi Training.....	69
3.2.5.5.Target.....	69
3.2.5.6. <i>Epoch</i> (Iterasi).....	70
3.2.6. Pemilihan Data <i>Training-Testing-Validation</i>	70
3.2.6.1.3 <i>Fold Cross Validation</i> .....	70
3.2.6.2.5 <i>Fold Validation</i> .....	71
3.2.7. Model Desain	72
3.2.7.1.User Interface Sistem	72
3.2.7.1.1. Halaman <i>Preprocessing</i> (Pembuatan Dataset).....	72
3.2.7.1.2. Halaman Test Group.....	72
3.2.7.1.3. Halaman Test Tunggal.....	73
3.2.8. Analisis Hasil.....	74
<b>Bab IV IMPLEMENTASI SISTEM DAN HASIL</b>	
4.1.Implementasi Coding Sistem.....	75
4.1.1. <i>Preprocessing</i> .....	75
4.1.1.1. <i>Resize – Centering</i> .....	75
4.1.1.2.Ekstraksi Ciri	76
4.1.2.1.Cari Kontur.....	77
4.1.2.2.Hitung <i>Centoid</i> .....	78
4.1.2.3.Hitung Kode Rantai.....	79
4.1.2.4.Hitung Panjang, Lebar, Diameter.....	79
4.1.2.5.Hitung Fitur Dispersi.....	81
4.1.2.6.Hitung Perimeter.....	82
4.1.2.7.Hitung Luas.....	82
4.1.2.8.Hitung Rasio Kerampingan.....	83
4.1.2.9.Hitung Rasio Kebulatan.....	83
4.1.3. Training dan Identifikasi.....	83
4.2.Desain Tampilan Sistem	85
4.2.1. Tampilan Jendela Pembuatan Dataset.....	85
4.2.2. Tampilan Jendela Penyusunan Arsitektur JST.....	86

4.2.3. Tampilan Jendela Klasifikasi.....	88
<b>4.3. Contoh Hasil <i>Preprocessing</i> Data dan Ekstraksi Ciri</b>	<b>89</b>
4.3.1. Preprocessing.....	89
4.3.1.1. <i>Grayscale</i> .....	89
4.3.1.2.Tepi Sobel.....	90
4.3.1.3.Dilasi.....	90
4.3.1.4. <i>Image Filling</i> .....	90
4.3.1.5. <i>Centering &amp; Resizing</i> .....	91
4.3.1.6.Kontur Tepi.....	91
4.3.2. Ekstraksi Ciri	91
4.3.2.1. Kontur.....	91
4.3.2.2.Kode Rantai.....	92
4.3.2.3.Luas.....	92
4.3.2.4.Perimeter.....	92
4.3.2.5.Panjang, Lebar, Diameter.....	92
4.3.2.6.Rasio Kebulatan.....	92
4.3.2.7.Rasio Kerampingan.....	92
4.3.2.8.Dispersi I dan IR.....	92
<b>4.4.Uji Coba dan Analisa Penggunaan dengan Parameter</b>	<b>93</b>
4.4.1. Pengaturan Arsitektur Awal.....	93
4.4.2. Pencarian jaringan optimal : Jumlah <i>Neuron</i> dan <i>Hidden Layer</i> Pertama	94
4.4.2.1.Pemilihan jenis penggunaan data, k-fold, dan jumlah <i>hidden neuron</i>	94
4.4.2.2.Pemilihan fungsi <i>training</i> , k-fold, dan 10 15 20 25 <i>hidden neuron</i> ...	96
4.4.3. Pencarian jaringan optimal : jumlah <i>neuron</i> pada <i>hidden layer</i> kedua	102
4.4.3.1.Pemilihan jenis penggunaan data, kfold dan jumlah <i>hidden neuron</i>	102
4.4.3.2.Pemilihan fungsi <i>training</i> dengan kfold pada <i>hidden neuron</i> 5 15 25	104
4.5.Uji coba data tunggal.....	108
4.6.Analisa hasil.....	108
<b>Bab V PENUTUP</b>	<b>112</b>
5.1.Kesimpulan.....	112
5.2.Saran.....	113
Daftar Pustaka.....	114
Lampiran	

## DAFTAR GAMBAR

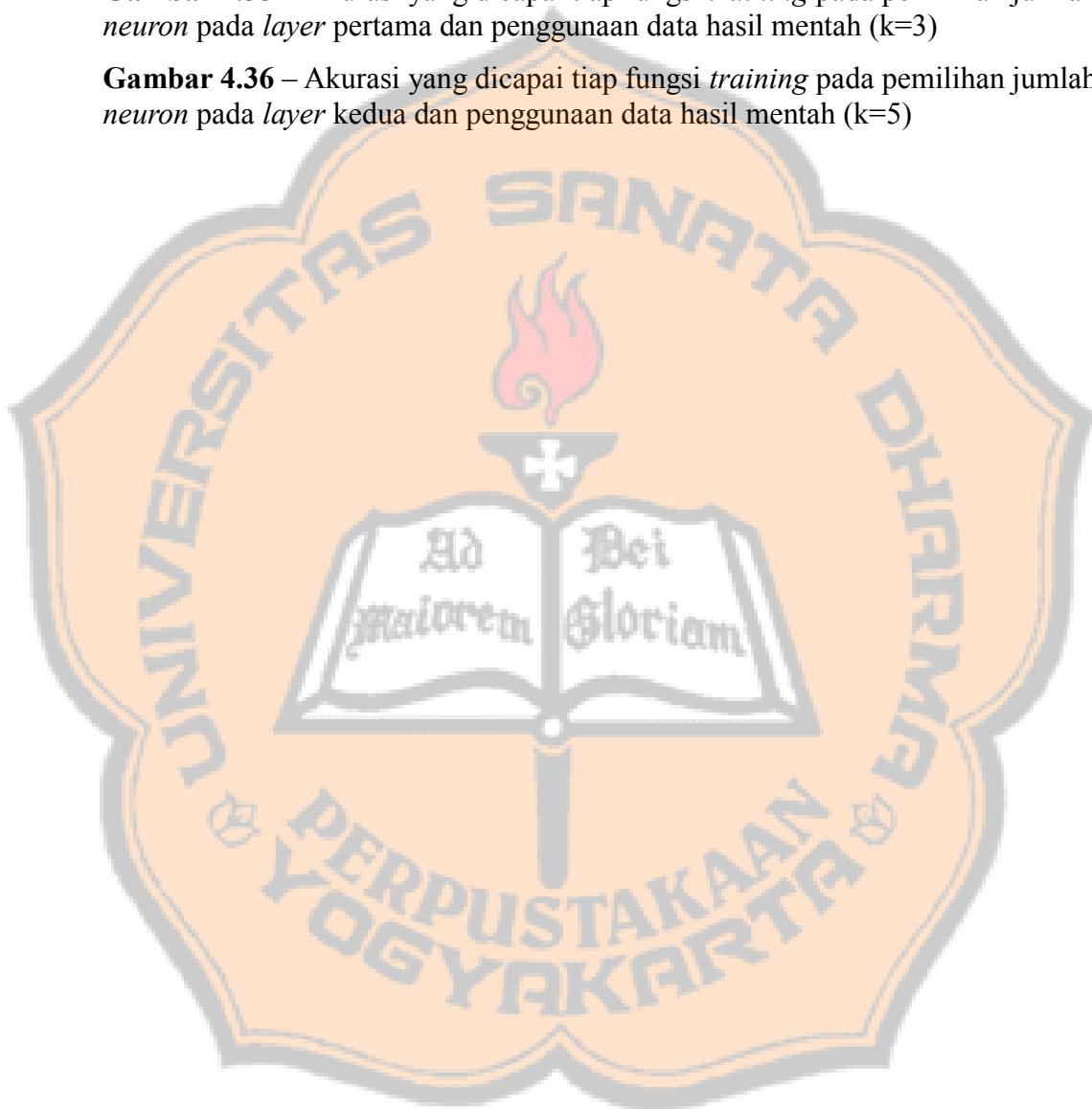
<b>Gambar 2.1 – Kernel Konvolusi Sobel</b>	12
<b>Gambar 2.2 - Representasi Bentuk (Kadir, 2013)</b>	14
<b>Gambar 2.3 - Gambar objek asli (<b>kiri</b>), kontur eksternalnya (<b>tengah</b>), dan kontur internal (<b>kanan</b>) (Kadir, 2013)</b>	14
<b>Gambar 2.4 - Visualisasi jalan kerja algoritma pelacakan kontur Moore. Dari <b>kiri ke kanan</b> : citra objek, proses pelacakan kontur dimulai dari titik (2,5), kontur moore yang dihasilkan, dan template 8 arah ketetanggan yang digunakan dalam pelacakan kontur (Kadir, 2013)</b>	15
<b>Gambar 2.5 - Visualisasi Pelacakan Kontur Moore <b>kiri ke kanan</b> : Citra Asli, Titik awal pelacakan kontur, iterasi ke-1</b>	15
<b>Gambar 2.6 - Visualisasi Pelacakan Kontur Moore <b>kiri ke kanan</b> : Iterasi ke-2, 3</b>	16
<b>Gambar 2.7 - Visualisasi Pelacakan Kontur Moore <b>kiri ke kanan</b> : Iterasi ke-4, 5, dan 6</b>	16
<b>Gambar 2.8 - Visualisasi Pelacakan Kontur Moore <b>kiri ke kanan</b> : Iterasi ke-7, 8, dan 9</b>	17
<b>Gambar 2.9 - Visualisasi Pelacakan Kontur Moore <b>kiri ke kanan</b> : Iterasi ke-10, 11, dan 12</b>	18
<b>Gambar 2.10 - Visualisasi Pelacakan Kontur Moore <b>kiri ke kanan</b> : Iterasi ke-13, 14, dan 15</b>	18
<b>Gambar 2.11 - Visualisasi Pelacakan Kontur Moore <b>kiri ke kanan</b> : Iterasi ke-16, 17, dan 18</b>	19
<b>Gambar 2.12 - Visualisasi Pelacakan Kontur Moore <b>kiri ke kanan</b> : Iterasi ke-19, 20, dan objek dengan kontur internalnya</b>	19
<b>Gambar 2.13 - Kontur internal objek hasil pelacakan</b>	20
<b>Gambar 2.14 - Arah rantai kode beserta kodennya (Kadir, 2013)</b>	20
<b>Gambar 2.15 - <b>Kiri</b> : Representasi indeks kode rantai, <b>Kanan</b> : kode rantai</b>	20
<b>Gambar 2.16 - <b>Kiri</b> : Kontur Objek, <b>Kanan</b> : Titik awal penelusuran</b>	21
<b>Gambar 2.17 - <b>Kiri ke Kanan</b> : Visualisasi pencarian kode rantai iterasi ke-1, 2, dan 3</b>	22
<b>Gambar 2.18 - <b>Kiri ke Kanan</b> : Visualisasi pencarian kode rantai iterasi ke-4, 5, dan 6</b>	22
<b>Gambar 2.19 - <b>Kiri ke Kanan</b> : Visualisasi pencarian kode rantai iterasi ke-7, 8, dan 9</b>	23
<b>Gambar 2.20 - <b>Kiri ke Kanan</b> : Visualisasi pencarian kode rantai iterasi ke-10, 11, dan 12</b>	23
<b>Gambar 2.21 - <b>Kiri ke Kanan</b> : Visualisasi pencarian kode rantai iterasi ke-13, 14, dan 15</b>	24
<b>Gambar 2.22 - <b>Kiri ke Kanan</b> : Visualisasi pencarian kode rantai iterasi ke-16, 17, dan 18</b>	24
<b>Gambar 2.23 - <b>Kiri ke Kanan</b> : Visualisasi pencarian kode rantai iterasi ke-19 dan 20</b>	25

<b>Gambar 2.24 - Kiri ke Kanan :</b> Visualisasi pencarian luas dengan kode rantai, Kontur objek (kiri) dan Titik (4,8) sebagai titik awal (kanan)	27
<b>Gambar 2.25 - Kiri ke Kanan :</b> Visualisasi pencarian luas dengan kode rantai, Iterasi 1, 2 dan 3	27
<b>Gambar 2.26 - Kiri ke Kanan :</b> Visualisasi pencarian luas dengan kode rantai, Iterasi 4, 5 dan 6	28
<b>Gambar 2.27 - Kiri ke Kanan :</b> Visualisasi pencarian luas dengan kode rantai, Iterasi 7, 8 dan 9	29
<b>Gambar 2.28 - Kiri ke Kanan :</b> Visualisasi pencarian luas dengan kode rantai, Iterasi 10, 11 dan 12	29
<b>Gambar 2.29 - Kiri ke Kanan :</b> Visualisasi pencarian luas dengan kode rantai, Iterasi 13, 14 dan 15	30
<b>Gambar 2.30 - Kiri ke Kanan :</b> Visualisasi pencarian luas dengan kode rantai, Iterasi 16, 17 dan 18	30
<b>Gambar 2.31 - Kiri ke Kanan :</b> Visualisasi pencarian luas dengan kode rantai, Iterasi 19 dan 20	31
<b>Gambar 2.32 -</b> Jarak antar piksel pada 8 ketetanggan, bernilai 1 pada piksel horizontal dan vertikal ( <b>kiri</b> ), dan bernilai $\sqrt{2}$ pada piksel yang diagonal ( <b>kanan</b> ) (Kadir, 2013)	32
<b>Gambar 2.33</b> Visualisasi penghitungan perimter (keliling) menggunakan kode rantai. Kontur Objek (kiri) dan representasi kode genap-ganjil (kanan)	32
<b>Gambar 2.34 -</b> Objek dengan kontur internal yang akan dicari panjang (diameter) dan lebar	34
<b>Gambar 2.35 -</b> Garis Panjang (Diameter) objek dengan gradien 1/8, pada titik (10,6), (2,5) dengan jarak 8,1	37
<b>Gambar 2.36 -</b> Garis Lebar objek dengan gradien -6, pada titik (4,8)-(5,2),(5,8)-(6,2), dan (6,8)-(7,2) dengan jarak 6,3	37
<b>Gambar 2.37 –</b> Visualisasi penghitungan pusat massa objek, dengan hasil (5.95, 5.23)	39
<b>Gambar 2.38 -</b> Tiga objek dengan kekompakan berbeda. Lingkaran ( <b>kiri</b> ) dan Oval ( <b>tengah</b> ) teratur, dan objek random ( <b>kanan</b> ) tidak teratur (Kadir, 2013)	40
<b>Gambar 2.39 –</b> Visualisasi penghitungan jarak sentroid dengan kontur, Sentroid-(10,6) merupakan jarak terpanjang dan Sentroid-(4,4) merupakan jarak terpendek	41
<b>Gambar 2.40 –</b> Jaringan layar tunggal (Siang, 2005)	43
<b>Gambar 2.41 –</b> Jaringan layar jamak (Siang, 2005)	43
<b>Gambar 2.42 –</b> Arsitektur <i>Backpropagation</i> (Siang, 2005)	45
<b>Gambar 3.1</b> Skenario pengambilan data. (1) : Kamera Digital, (2) dan (3) : Sumber Cahaya, (4) : Jarak sumber cahaya dengan obyek penampang kopi, (5) : Jarak kamera digital dengan obyek penampang kopi (6) : Penampang biji kop	51
<b>Gambar 3.2 –</b> Garis Besar Alur Alat Uji	53
<b>Gambar 3.3 –</b> <i>Preprocessing</i> Data Citra Biji Kopi yang akan digunakan	54
<b>Gambar 3.4 –</b> Proses Ekstraksi ciri citra biji kopi yang dilakukan	54
<b>Gambar 3.5 –</b> Proses pelatihan jaringan saraf tiruan untuk identifikasi	55
<b>Gambar 3.6 –</b> Proses Uji klasifikasi dari jaringan saraf tiruan hasil pelatihan	55

<b>Gambar 3.7 – Diagram Konteks Sistem</b>	56
<b>Gambar 3.8 – Diagram Alur Data, Level 1 Sistem</b>	56
<b>Gambar 3.9 – Diagram alur data, Level 2 Proses Pelatihan sistem</b>	57
<b>Gambar 3.10 – Diagram alur data, Level 2 proses <i>Testing</i></b>	58
<b>Gambar 3.11 – Diagram alur data, level 3 <i>preprocess</i> data</b>	59
<b>Gambar 3.12 – Diagram alur data, level 3 ekstraksi ciri</b>	60
<b>Gambar 3.13 – Contoh hasil foto biji kopi pada penampang</b>	61
<b>Gambar 3.14 - Citra biji kopi hasil cropping, dengan ukuran 256 x 256 piksel</b>	61
<b>Gambar 3.15 - Arsitektur Jaringan Saraf Tiruan yang akan digunakan dalam penelitian ini</b>	70
<b>Gambar 3.16 – Home Screen <i>Preprocessing</i></b>	72
<b>Gambar 3.17 – Halaman Set Data <i>Training</i></b>	73
<b>Gambar 3.18 – Halaman Hasil</b>	74
<b>Gambar 4.1 – Tampilan awal untuk <i>preprocessing</i> dan ekstraksi ciri</b>	85
<b>Gambar 4.2 – Pemilihan data untuk <i>preprocessing</i></b>	85
<b>Gambar 4.3 – Tampilan <i>preview</i> dan dataset yang dihasilkan.</b>	86
<b>Gambar 4.4 – Tampilan Jendela penyusunan jaringan saraf tiruan</b>	86
<b>Gambar 4.5 – file ‘strBijidanCiriUtuh.mat’ dan ‘strBijidanCiriPecah.mat’</b>	87
<b>Gambar 4.6 – Tampilan perancangan Jaringan Saraf Tiruan</b>	87
<b>Gambar 4.7 – Grafik peforma dan status pelatihan</b>	88
<b>Gambar 4.8 – Tampilan jendela identifikasi tunggal</b>	88
<b>Gambar 4.9 – Tampilan jendela identifikasi tunggal dan hasilnya</b>	89
<b>Gambar 4.10 – Citra Asli (kiri), Citra <i>Grayscale</i> (kanan)</b>	89
<b>Gambar 4.11 – Citra <i>Grayscale</i> (kiri), Citra Biner (kanan)</b>	90
<b>Gambar 4.12 – Citra Biner (kiri), Dilasi Vertikal (tengah), Dilasi Horizontal (kanan)</b>	90
<b>Gambar 4.13 – Citra Biner (kiri), Hasil mengisi lubang objek (kanan)</b>	90
<b>Gambar 4.14 – Citra Biner (kiri), Hasil <i>centering</i> dan <i>resizing</i> (kanan)</b>	91
<b>Gambar 4.15 – Citra Biner (kiri), Piksel Kontur (kanan)</b>	91
<b>Gambar 4.16 – Uji 1 : Hasil akurasi identifikasi dengan arsitektur: jumlah <i>hidden neuron</i> pada <i>hidden layer</i> ke-1 dengan k <i>value</i> = 3 pada data yang dinormalisasi dan data mentah</b>	94
<b>Gambar 4.17 – Uji 2 : Hasil akurasi identifikasi dengan arsitektur: jumlah <i>hidden neuron</i> pada <i>hidden layer</i> ke-1 dengan k <i>value</i> = 5 pada data yang dinormalisasi dan data mentah</b>	95

<b>Gambar 4.18</b> – Uji 3 : Hasil akurasi identifikasi dengan arsitektur: 10 <i>Hidden Neuron</i> , K=3, dan fungsi <i>training</i> pada data hasil normalisasi dan data mentah	96
<b>Gambar 4.19</b> – Uji 4 : Hasil akurasi identifikasi dengan arsitektur: 10 <i>Hidden Neuron</i> , K=5, dan fungsi <i>training</i> pada data hasil normalisasi dan data mentah	97
<b>Gambar 4.20</b> – Uji 5 : Hasil akurasi identifikasi dengan arsitektur: 15 <i>Hidden Neuron</i> , K=3, dan fungsi <i>training</i> pada data hasil normalisasi dan data mentah	98
<b>Gambar 4.21</b> – Uji 6 : Hasil akurasi identifikasi dengan arsitektur: 15 <i>Hidden Neuron</i> , K=5, dan fungsi <i>training</i> pada data hasil normalisasi dan data mentah	99
<b>Gambar 4.22</b> – Uji 7 : Hasil akurasi identifikasi dengan arsitektur: 20 <i>Hidden Neuron</i> , K=3, dan fungsi <i>training</i> pada data hasil normalisasi dan data mentah	99
<b>Gambar 4.23</b> – Uji 8 : Hasil akurasi identifikasi dengan arsitektur: 20 <i>Hidden Neuron</i> , K=5, dan fungsi <i>training</i> pada data hasil normalisasi dan data mentah	100
<b>Gambar 4.24</b> – Uji 9 dan 10: Hasil akurasi identifikasi dengan arsitektur: 25 <i>Hidden Neuron</i> , K=3 dan 5 , dan fungsi <i>training</i> pada data hasil normalisasi dan data mentah	101
<b>Gambar 4.25</b> – Uji 11 : Hasil akurasi identifikasi dengan arsitektur: jumlah <i>hidden neuron</i> 15 pada <i>layer</i> ke-1 dan <i>layer</i> -2 dengan k <i>value</i> = 3 pada data yang dinormalisasi dan data mentah	102
<b>Gambar 4.26</b> – Uji 12 : Hasil akurasi identifikasi dengan arsitektur: jumlah <i>hidden neuron</i> 15 pada <i>layer</i> ke-1 dan <i>layer</i> -2 dengan k <i>value</i> = 5 pada data yang dinormalisasi dan data mentah	103
<b>Gambar 4.27</b> – Uji 13 : Hasil akurasi identifikasi dengan arsitektur: 15 <i>neuron</i> pada <i>layer</i> ke-1 dan 5 <i>neuron</i> pada <i>layer</i> -2 dengan k <i>value</i> =3 pada data yang dinormalisasi dan data mentah	104
<b>Gambar 4.28</b> – Uji 14 : Hasil akurasi identifikasi dengan arsitektur: 15 <i>neuron</i> pada <i>layer</i> ke-1 dan 5 <i>neuron</i> pada <i>layer</i> -2 dengan k <i>value</i> =5 pada data yang dinormalisasi dan data mentah	105
<b>Gambar 4.29</b> – Uji 15: Hasil akurasi identifikasi dengan arsitektur: 15 <i>neuron</i> pada <i>layer</i> ke-1 dan 15 <i>neuron</i> pada <i>layer</i> -2 dengan k <i>value</i> = 3 pada data yang dinormalisasi dan data mentah	106
<b>Gambar 4.30</b> – Uji 16: Hasil akurasi identifikasi dengan arsitektur: 15 <i>neuron</i> pada <i>layer</i> ke-1 dan 15 <i>neuron</i> pada <i>layer</i> -2 dengan k <i>value</i> = 5 pada data yang dinormalisasi dan data mentah	106
<b>Gambar 4.31</b> – Uji 17 : Hasil akurasi identifikasi dengan arsitektur: 15 <i>neuron</i> pada <i>layer</i> ke-1 dan 25 <i>neuron</i> pada <i>layer</i> -2 dengan k <i>value</i> = 3 pada data yang dinormalisasi dan data mentah	107
<b>Gambar 4.32</b> – Uji 18 : Hasil akurasi identifikasi dengan arsitektur: 15 <i>neuron</i> pada <i>layer</i> ke-1 dan 25 <i>neuron</i> pada <i>layer</i> -2 dengan k <i>value</i> = 5 pada data yang dinormalisasi dan data mentah	108

<b>Gambar 4.33</b> – Akurasi yang dicapai tiap fungsi <i>training</i> pada pemilihan jumlah <i>neuron</i> pada <i>layer</i> pertama dan penggunaan data hasil normalisasi (k=3)	109
<b>Gambar 4.34</b> – Akurasi yang dicapai tiap fungsi <i>training</i> pada pemilihan jumlah <i>neuron</i> pada <i>layer</i> kedua dan penggunaan data hasil normalisasi (k=5)	110
<b>Gambar 4.35</b> – Akurasi yang dicapai tiap fungsi <i>training</i> pada pemilihan jumlah <i>neuron</i> pada <i>layer</i> pertama dan penggunaan data hasil mentah (k=3)	110
<b>Gambar 4.36</b> – Akurasi yang dicapai tiap fungsi <i>training</i> pada pemilihan jumlah <i>neuron</i> pada <i>layer</i> kedua dan penggunaan data hasil mentah (k=5)	111



## DAFTAR TABEL

<b>Tabel 2.1 – Syarat Mutu Umum (BSNI, 2008)</b>	7
<b>Tabel 2.2 – Syarat Mutu Khusus Berdasarkan Sistem Nilai Cacat (BSNI, 2008)</b>	7
<b>Tabel 2.3 – Penentuan Nilai Cacat (BSNI, 2008)</b>	8
<b>Tabel 2.4 - Indeks dan kode rantai dua piksel bertetangga (Kadir, 2013)</b>	21
<b>Tabel 2.5 - Tabel hasil kode rantai dan indeksnya dari kontur objek masukan</b>	25
<b>Tabel 2.6 - Tabel hasil penghitungan luas berdasarkan kode rantai dan koordinat penyusun kontur</b>	31
<b>Tabel 2.7 - . Hasil penghitungan perimeter (keliling) menggunakan kode rantai</b>	33
<b>Tabel 2.8 - . Pengitungan Jarak <i>Euclidean</i> antar titik kontur objek</b>	35
<b>Tabel 2.9 - . Pengitungan gradien garis yang melewati titik kontur objek</b>	36
<b>Tabel 2.10 – Tabel jarak antara titik penyusun kontur objek dan titik pusat massa</b>	41
<b>Tabel 3.1 – Target Jaringan Saraf Tiruan <i>Backpropagation</i> untuk kelas</b>	70
<b>Tabel 3.2 – 3 Fold Cross Validation</b>	71
<b>Tabel 3.3 – 5 Fold Cross Validation</b>	71
<b>Tabel 3.4 – Tabel <i>Confusion Matrix</i> untuk akurasi klasifikasi jenis cacat</b>	74
<b>Tabel 4.1 – 15 koordinat kontur pertama ‘pecah.3.jpg’</b>	91
<b>Tabel 4.2 – kode rantai citra ‘pecah.3.jpg’</b>	92
<b>Tabel 4.3 – Hasil Uji Data Tunggal</b>	108

## DAFTAR ALGORITMA

<b>Algoritma 2.1</b> - Pelatihan jaringan saraf tiruan <i>backpropagation</i>	47
<b>Algoritma 3.1</b> – Proses <i>Grayscale</i> ‘rgb2gray’ (MATLAB documentation)	62
<b>Algoritma 3.2</b> – Proses deteksi biner Sobel dengan <i>toolbox image</i> MATLAB	62
<b>Algoritma 3.3</b> – Proses dilasi dengan <i>toolbox image</i> MATLAB	63
<b>Algoritma 3.4</b> – Proses dilasi imfill dengan <i>toolbox image</i> MATLAB	63
<b>Algoritma 3.5</b> – Proses <i>image center &amp; resize</i>	64
<b>Algoritma 3.6</b> – Proses tepi biner	64
<b>Algoritma 3.7</b> – Memperoleh kontur internal Moore	65
<b>Algoritma 3.8</b> – Proses pencarian kode rantai	65
<b>Algoritma 3.9</b> – Proses pencarian Luas	66
<b>Algoritma 3.10</b> – Proses pencarian perimeter	66
<b>Algoritma 3.11</b> – Estimasi panjang (diameter), lebar bentuk	67
<b>Algoritma 3.12</b> – Proses pencarian Rasio Kebulatan	67
<b>Algoritma 3.13</b> – Proses pencarian Rasio Kerampingan	68
<b>Algoritma 3.14</b> – Estimasi penghitungan sentroid	68
<b>Algoritma 3.15</b> – Estimasi penghitungan Dispersi I dan IR	69

## DAFTAR PERSAMAAN

<b>Persamaan 2.1</b> - Intensitas Cahaya dalam bidang 2 dimensi	11
<b>Persamaan 2.2</b> – Konversi warna ke citra keabuan	12
<b>Persamaan 2.3</b> – Formula konversi <i>grayscale</i> NTSC	12
<b>Persamaan 2.4</b> – Proses Dilasi	13
<b>Persamaan 2.5</b> – Indeks Kode Rantai	21
<b>Persamaan 2.6</b> – Hitung Perimeter dengan kode rantai	32
<b>Persamaan 2.7</b> – Jarak <i>Euclidean</i>	33
<b>Persamaan 2.8</b> – Gradien Garis Panjang	34
<b>Persamaan 2.9</b> - Gradien Garis Lebar	34
<b>Persamaan 2.10</b> – Rasio Kebulatan	38
<b>Persamaan 2.11</b> – Rasio Kerampingan	38
<b>Persamaan 2.12</b> – Pencarian Sentroid	39
<b>Persamaan 2.13</b> – Dispersi I	40
<b>Persamaan 2.14</b> – Dispersi IR	40
<b>Persamaan 2.15</b> – Kombinasi linear masukan dan bobot neuron	44
<b>Persamaan 2.16</b> – Fungsi <i>threshold</i>	44
<b>Persamaan 2.17</b> – Fungsi <i>Sigmoid</i>	44
<b>Persamaan 2.18</b> – Turunan Fungsi <i>Sigmoid</i>	44
<b>Persamaan 2.19</b> – Fungsi Identitas	44
<b>Persamaan 2.20</b> – Fungsi <i>Sigmoid</i> Biner	46
<b>Persamaan 2.21</b> – Turunan <i>sigmoid</i> biner	46
<b>Persamaan 2.22</b> – Fungsi <i>sigmoid</i> bipolar	46
<b>Persamaan 2.23</b> – Turunan <i>sigmoid</i> bipolar	46
<b>Persamaan 2.24</b> – Fungsi identitas pada keluaran	46
<b>Persamaan 2.25</b> – Jumlah <i>output</i> dari masukan dari <i>hidden layer</i>	47
<b>Persamaan 2.26</b> – Keluaran <i>hidden layer</i> dari fungsi aktivasi	47
<b>Persamaan 2.27</b> – Jumlah <i>output</i> dari masukan pada <i>layer output</i>	47
<b>Persamaan 2.28</b> – Keluaran <i>layer output</i> dari fungsi aktivasi	47

<b>Persamaan 2.29 – Faktor kesalahan dari <i>layer output</i></b>	47
<b>Persamaan 2.30 – Perubahan bobot</b>	48
<b>Persamaan 2.31 – faktor <math>\delta</math> <i>hidden layer</i> berdasarkan kesalahan</b>	48
<b>Persamaan 2.32 - Faktor <math>\delta</math> di <i>hidden layer</i></b>	48
<b>Persamaan 2.33 - perubahan bobot <math>v_{ij}</math></b>	48
<b>Persamaan 2.34 - Perubahan bobot garis ke <i>output layer</i></b>	48
<b>Persamaan 2.35 - Perubahan bobot garis ke <i>hidden layer</i></b>	48
<b>Persamaan 3.1 – Penghitungan Akurasi</b>	74

## DAFTAR LISTING PROGRAM

<b>Listing Program 4.1 – Preprocessing Data</b>	75
<b>Listing Program 4.2 – Centering - Resizing</b>	76
<b>Listing Program 4.3 – Ekstraksi Ciri</b>	77
<b>Listing Program 4.4 – Kontur Internal</b>	78
<b>Listing Program 4.5 – Cari Sentroid</b>	79
<b>Listing Program 4.6 – Hitung Kode Rantai</b>	79
<b>Listing Program 4.7 – Pencarian Panjang (Diameter), Lebar</b>	80
<b>Listing Program 4.8 – Penghitungan Dispersi</b>	81
<b>Listing Program 4.9 – Pencarian Perimeter</b>	82
<b>Listing Program 4.10 – Pencarian Luas</b>	82
<b>Listing Program 4.11 – Hitung Rasio Kerampingan</b>	83
<b>Listing Program 4.12 – Hitung Rasio Kebulatan</b>	83
<b>Listing Program 4.13 – Proses <i>Training</i> dan Identifikasi</b>	84

## BAB I

### PENDAHULUAN

#### 1.1. Latar Belakang

Biji kopi merupakan salah satu komoditas yang banyak diperdagangkan di era global saat ini. Biji kopi dihasilkan oleh tanaman kelompok genus *Coffea*, famili *Rubiaceace*. Biji kopi dihasilkan lebih dari 70 negara yang sebagian besar terletak di daerah tropis, yakni di benua Amerika Selatan, Afrika, India, dan Asia Tenggara. Dengan total produksi dunia yang mencapai 8.920.840 (FAOSTAT, 2013) ton pada 2013, membuat biji kopi ini menjadi komoditas yang paling banyak dicari.

Di Indonesia, kopi merupakan salah satu produk unggulan dalam sektor perkebunan. Untuk dapat bersaing dengan negara-negara penghasil kopi lainnya, maka mutu kopi Indonesia harus sesuai dengan standar yang telah ditetapkan. Dengan 70 % total produksi nasional dijadikan komoditas ekspor, maka perlu adanya standardisasi kualitas biji kopi. Dalam melakukan penilaian kualitas biji kopi, dapat dilakukan berbagai macam metode, salah satunya adalah melakukan pemutuan kualitas terhadap biji kopi.

Sejarah penerapan standar mutu terhadap komoditas kopi sudah ada sejak jaman pendudukan Belanda. Pada waktu itu dikenal dengan nama OVEIP atau *Organisatie Verenigde Eksporteurs Van Indonesische Producten*. Organisasi ini merupakan lembaga yang melakukan standarisasi dari produk-produk komoditas yang diekspor dari Indonesia. Selanjutnya dierapkan sistem TRIAGE, atau lebih dikenal dengan sistem nilai kotor. Nilai kotor yang dimaksud adalah biji kopi warna hitam, warna cokelat, dan feksel (biji pecah/hancur). Dengan adanya perkembangan selera dan permintaan akan komoditas kopi, maka pada tanggal 1 Oktober 1983 ditetapkan standar mutu kopi dengan nama “Sistem Nilai Cacat” (*defect system*). Pada sistem mutu kopi ini dikenal kopi mutu 1 sampai dengan mutu 6. Pembaharuan standar mutu dengan sistem ini dimaksudkan untuk menyesuaikan standar mutu kopi Indonesia dengan sistem pemutuan kualitas di negara khususnya negara penghasil kopi. Sistem inilah yang diadopsi Badan Standarisasi Nasional

Indonesia (BSNI) dalam menetapkan Standar Nasional Indonesia (SNI) untuk pemutuan kualitas biji kopi. Seiring berjalannya waktu, SNI Biji Kopi mengalami beberapa pengubahan, dan SNI terbaru yang digunakan saat ini adalah SNI no. 01-2907-2008. dalam standar ini terdapat definisi dan penetapan nilai cacat untuk jenis cacat dari biji kopi, dan penggolongan mutu berdasarkan nilai cacat yang didapatkan.

Penelitian sebelumnya telah dilakukan oleh beberapa peneliti untuk melakukan identifikasi atau klasifikasi dari kualitas kopi berdasarkan citra biji kopi. Sofi'i (2005) melakukan pemutuan biji kopi dengan pengolahan citra digital biji kopi dan jaringan saraf tiruan. Penelitian tersebut menghasilkan akurasi identifikasi dari 20 jenis cacat berdasarkan standar SNI sebesar 72.6%. Selain itu, Madi (2010) dalam penelitiannya berhasil melakukan klasifikasi kualitas biji kopi menggunakan citra kedalam 4 kelas kualitas dengan akurasi rerata sistem sebesar 81.1%. Dengan menggunakan penghitungan entropi, energi, kontras, dan homogenitas, Parikesit, dkk (2011) melakukan klasifikasi kualitas sampel dari biji kopi kedalam 7 kelas sesuai standar SNI dengan akurasi rerata yang dihasilkan sebesar 74,28%. Sedangkan Ayitenfsu (2014) menggunakan salahsatu deskriptor ciri bentuk dasar ‘kebulatan’ dalam melakukan klasifikasi biji kopi menjadi 2 kelas. Dengan menetapkan ambang batas dari ciri yang didapatkan, akurasi klasifikasi yang dicapai sebesar 98% (dengan 2% misklasifikasi).

Salah satu jenis cacat pada standar SNI untuk biji kopi adalah biji pecah. Biji pecah merupakan biji kopi yang tidak utuh, yang besarnya sama atau kurang dari  $\frac{3}{4}$  biji yang utuh. Untuk setiap 1 (satu) biji yang pecah dalam sampel 300 gr, dikenakan nilai 1/5 untuk nilai cacatnya (*defect value*). Pada penelitian sebelumnya digunakan beberapa parameter seperti luas, parimeter, panjang, lebar, diameter, kerampingan, kebulatan, dan dispersi sebagai ciri, dan jaringan saraf tiruan sebagai alat klasifikasi dalam melakukan identifikasi kualitas dari biji kopi. Penelitian ini akan menggunakan ciri-ciri tersebut dan melakukan perancangan arsitektur jaringan saraf tiruan sebagai alat untuk melakukan klasifikasi jenis bentuk biji. Diharapkan penelitian ini dapat melakukan klasifikasi terhadap bentuk biji kopi utuh dan biji kopi pecah dengan tingkat akurasi yang baik.

## 1.2. Rumusan Masalah

Masalah yang akan diselesaikan melalui penelitian ini adalah:

1. Bagaimana cara kerja ekstraksi menggunakan Deskriptor Bentuk Dasar (*Basic Region Descriptor*) dalam proses ekstraksi ciri pada masing-masing citra biji kopi?
2. Bagaimana arsitektur jaringan saraf tiruan yang optimal dalam melakukan identifikasi?
3. Berapakah akurasi hasil identifikasi jenis bentuk biji kopi yang dihasilkan dari pemrosesan citra dan klasifikasi dari jaringan saraf tiruan yang sudah dibentuk?

## 1.3. Maksud dan Tujuan Penelitian

Penelitian ini dilakukan dengan maksud dan tujuan:

1. Mengetahui cara kerja ekstraksi menggunakan Deskriptor Bentuk Dasar (*Basic Region Descriptor*) dalam proses ekstraksi ciri pada masing-masing citra biji kopi.
2. Menentukan arsitektur jaringan saraf tiruan yang optimal dalam melakukan identifikasi
3. Melakukan pengujian hasil keluaran dari jaringan saraf tiruan yang sudah dibentuk, dan mengukur akurasi yang dihasilkan.

## 1.4. Batasan Masalah

Batasan (*constraint*) masalah dari penelitian ini adalah

1. Sampel biji kopi yang digunakan adalah sampel biji kopi jenis robusta dan arabica dengan berat sampel 300gr robusta yang telah disortasi berdasarkan aturan SNI atau sortasi berdasarkan bagian-bagian kopi
2. Pengambilan citra sampel biji kopi menggunakan kamera digital Nikon D3300 dengan penampang latar belakang warna putih, kecepatan lensa 1/40 detik, dan ISO *speed rating* 100
3. Citra yang digunakan diambil dengan luas penampang 21 x 29.7 cm dengan resolusi kamera 6000 x 4000 piksel

4. Segmentasi, *cropping* citra, dan pembersihan latar obyek kopi dilakukan manual perbutir dengan ukuran citra masing-masing biji 256 x 256 piksel
5. Citra disimpan dalam file berekstensi .jpg
6. Pemisahan citra hasil foto menjadi 2 kelas, yakni biji pecah dan utuh dilakukan dengan melakukan pengamatan visual secara manual.
7. *Preprocessing* data citra dilakukan untuk mendapatkan bentuk kontur dari biji kopi. Proses ini meliputi operasi deteksi tepi *Sobel*, dilasi, *image filling*, *center-resizing*, dan operasi tepi biner.
8. Proses binerisasi menggunakan *toolbox Image Processing Toolbox* dari Matlab 2012b. Proses ini meliputi deteksi tepi *Sobel* (*edge('sobel')*) dan operasi pembanjiran *image fill* ('*imfill*')
9. Ekstraksi ciri yang digunakan adalah deskriptor bentuk dasar (*Basic Region Descriptor*) yang terdiri dari penghitungan luas, perimeter, panjang, lebar, diameter, rasio kebulatan, rasio kerampingan, dispersi minimum dan dispersi maksimum, dimana ciri tersebut merupakan hasil penghitungan operasi geometri citra.
10. Data akan diberikan 2 perlakuan yakni proses training akan dilakukan dengan normalisasi dan tanpa normalisasi. Dalam proses normalisasi akan digunakan tool *mapminmax* sebagai fungsi untuk normalisasi data dari Matlab R2012b
11. Proses klasifikasi menggunakan jaringan saraf propagasi balik. Perancangan arsitektur menggunakan 9 buah fungsi *training*, dimana *hidden neuron* akan dibatasi 5 s.d 25 pada setiap *layernya* (sejumlah 2), *epoch* (iterasi) tiap *training* akan dilakukan sebanyak 100 kali
12. Dalam proses training jaringan saraf tiruan, penentuan data *training*, *testing*, dan *validation* dilakukan dengan membagi data per kelas menjadi 3 dan 5 *fold*, dimana *fold* pertama adalah data *training*, *fold* kedua adalah data *testing*, dan *fold* ketiga adalah data *validation*, dan seterusnya.
13. Penentuan jaringan saraf tiruan paling optimal ditentukan dengan hasil akurasi identifikasi tertinggi dari data testing dalam proses training

14. Perangkat lunak (*software*) yang digunakan adalah Matlab R2012b yang digunakan untuk perancangan antarmuka (GUI), proses *preprocessing*, ekstraksi ciri, dan klasifikasi.

### **1.5. Sistematika Penulisan**

Sistematika penulisan dalam tugas akhir yang berjudul Identifikasi Bentuk Biji Kopi Menggunakan Deskriptor Bentuk Dasar dan Jaringan Saraf Tiruan ini dijelaskan sebagai berikut:

#### **Bab I Pendahuluan**

Pada bab ini dijelaskan latar belakang masalah yang mendorong dilakukannya penelitian ini, rumusan masalah, tujuan dan maksud penelitian, batasan masalah penelitian, serta sistematika penulisan yang digunakan dalam menyelesaikan laporan tugas akhir ini.

#### **Bab II Landasan Teori**

Pada bab ini dijelaskan teori dan metode yang digunakan dalam penelitian ini. Teori-teori tersebut seperti Teori dan penjelasan tentang biji kopi, Sistematika Standar Nasional Indonesia, Teori Citra dan Pemrosesan Citra, penjelasan tentang ekstraksi ciri menggunakan *Basic Region Descriptor*, serta Penjelasan tentang metode jaringan saraf tiruan dengan *backpropagation*.

#### **Bab III Metodologi Penelitian dan Perancangan Sistem Uji**

Pada bab ini dijelaskan beberapa hal seperti metodologi penelitian yang digunakan, meliputi studi literatur, data penelitian, pengembangan alat uji penelitian, serta analisa hasil penelitian.

#### **Bab IV Implementasi Sistem dan Analisa Hasil**

Pada bab ini dijelaskan beberapa hal seperti implementasi program yang sudah dijelaskan pada bagian metode penelitian, serta analisa tentang hasil dari identifikasi dari program yang telah dirancang.

#### **Bab V Penutup**

Pada bab ini berisi kesimpulan, dan saran dari hasil penelitian yang sudah dilakukan.

## BAB II

### LANDASAN TEORI

#### 2.1. Kopi

Tanaman kopi termasuk dalam famili Rubiaceae dan terdiri dari banyak jenis antara lain *Coffea arabica*, *Coffea robusta*, dan *Coffea liberica*. Kopi banyak diyakini berasal dari sebuah kerajaan kuno di Ethiopia bernama Abessinia. Dan disana, tanaman kopi tumbuh di dataran tinggi. Berikut merupakan sistematika klasifikasi spesies kopi arabika, robusta dan liberika (Armansyah, 2010):

Kingdom	:	Plantae
Subkingdom	:	Tracheobionta
Super Divisi	:	Spermatophyta
Divisi	:	Magnoliophyta
Kelas	:	Magnoliophyta
Sub Kelas	:	Asteridae
Ordo	:	Rubiales
Famili	:	Rubiaceae
Genus	:	<i>Coffea</i>
Species	:	<i>Coffea arabica</i> (Kopi Arabika), <i>Coffea robusta</i> (Kopi Robusta) dan <i>Coffea liberica</i> (Kopi Liberika)

Di Indonesia, yang banyak dibudidayakan adalah *Coffea arabica* dan *Coffea robusta*. Banyak varietas unggul yang dikembangkan seperti tipe kloning dan beberapa sifat unggul seperti tahan penyakit karat daun dan nematoda (*Radopholus similis, cobb*) – Andungsari, dan lain-lain. Adapun varietas kopi arabika lain seperti Abisina 3, Kartika 1, Kartika 2, S 795 dan USDA 762 yang dikembangkan sesuai dengan kebutuhan produksi (ICCR, 20--).

## 2.2. Standar Nasional Indonesia No 01-2907-2008

Standar Nasional Indonesia (SNI) biji kopi (No. 01-2907-2008) merupakan revisi dari upaya standardisasi yang sudah dilakukan sebelumnya (01-2907-1999). Standar ini dirumuskan oleh Panitia Teknis SNI 65-03 bagian Pertanian. Standar ini disusun dan direvisi berdasarkan perkembangan pasar global, salah satunya Resolusi ICO yang berisi : “*Negara pengekspor tidak boleh mengekspor kopi dengan (a) Untuk Arabika, tidak melebihi 86 nilai cacat per 300 gr sampel (b) Robusta tidak melebihi 150 nilai cacat per 300 gr*” yang berlaku sejak 1 Oktober 2002. Untuk mengantisipasi hal tersebut perlu dilakukan peningkatan standar mutu kopi Indonesia yang disesuaikan dengan standar mutu kopi dunia. Standar ini sudah dibahas melalui rapat dan konsensus Panitia Teknis dan pada 15 September 2007 sampai 21 Agustus 2007 dilakukan jejak pendapat dan disetujui menjadi RASNI (Rancangan Standar Nasional Indonesia).

Berdasarkan dokumen SNI no. 01-2907-2008 terdapat beberapa beberapa syarat mutu umum dan syarat mutu khusus berdasarkan nilai cacat yang disajikan dalam tabel berikut.

**Tabel 2.1 – Syarat Mutu Umum (BSNI, 2008)**

No.	Kriteria	Satuan	Persyaratan
1.	Serangga Hidup	-	Tidak ada
2.	Biji berbau busuk/kapang	-	Tidak ada
3.	Kadar Air	% fraksi massa	Maks. 12.5
4.	Kadar kotoran	% fraksi massa	Maks. 0.5

**Tabel 2.2 – Syarat Mutu Khusus Berdasarkan Sistem Nilai Cacat (BSNI, 2008)**

Mutu	Persyaratan
Golongan I	Jumlah nilai cacat maksimum 11
Golongan II	Jumlah nilai cacat 12 sampai dengan 25
Golongan III	Jumlah nilai cacat 26 sampai dengan 44
Golongan IVA	Jumlah nilai cacat 45 sampai dengan 60
Golongan IVB	Jumlah nilai cacat 61 sampai dengan 80
Golongan V	Jumlah nilai cacat 81 sampai dengan 150

Golongan VI	Jumlah nilai cacat 151 sampai dengan 225
<b>CATATAN</b> Untuk kopi arabika Golongan IV tidak dibagi menjadi IVA dan IVB	

**Tabel 2.3 – Penentuan Nilai Cacat (BSNI, 2008)**

No.	Jenis Cacat	Nilai Cacat
1.	1 (satu) biji hitam	1
2.	1 (satu) biji hitam sebagian	$\frac{1}{2}$
3.	1 (satu) biji hitam pecah	$\frac{1}{2}$
4.	1 (satu) biji kopi gelondong	1
5.	1 (satu) biji cokelat	$\frac{1}{4}$
6.	1 (satu) kulit kopi ukuran besar	1
7.	1 (satu) kulit kopi ukuran sedang	$\frac{1}{2}$
8.	1 (satu) kulit kopi ukuran kecil	$\frac{1}{5}$
9.	1 (satu) biji berkulit tanduk	$\frac{1}{2}$
10.	1 (satu) kulit tanduk ukuran besar	$\frac{1}{2}$
11.	1 (satu) kulit tanduk ukuran sedang	$\frac{1}{5}$
12.	1 (satu) kulit tanduk ukuran kecil	$\frac{1}{10}$
13.	1 (satu) biji pecah	$\frac{1}{5}$
14.	1 (satu) biji muda	$\frac{1}{5}$
15.	1 (satu) biji berlubang satu	$\frac{1}{10}$
16.	1 (satu) biji berlubang lebih dari satu	$\frac{1}{5}$
17.	1 (satu) biji bertutul-tutul	$\frac{1}{10}$
18.	1 (satu) ranting/tanah/batu ukuran besar	5
19.	1 (satu) ranting/tanah/batu ukuran sedang	2
20.	1 (satu) ranting/tanah/batu ukuran kecil	1

**KETERANGAN** Jumlah nilai cacat dihitung dari contoh uji seberat 300 gr. Jika satu biji mempunyai lebih dari satu nilai cacat, maka penentuan nilai cacat tersebut didasarkan pada bobot nilai cacat terbesar.

Adapun definisi dari 20 kelas jenis cacat sesuai Standar Nasional Indonesia No 01-2907-2008 adalah sebagai berikut,

1. Biji hitam  
biji kopi yang setengah atau lebih dari bagian luarnya berwarna hitam baik yang mengkilap maupun keriput.
2. Biji hitam sebagian  
biji kopi yang kurang dari setengah bagian luarnya berwarna hitam, atau satu bintik hitam kebiru-biruan tetapi tidak berlubang atau ditemukan lubang dengan warna hitam yang lebih besar dari lubang tersebut
3. Biji hitam pecah  
biji kopi yang berwarna hitam tidak utuh, berukuran sama dengan atau kurang dari 3/4 bagian biji utuh, atau biji hitam sebagian yang pecah.
4. Kopi gelondong  
buah kopi kering yang masih terbungkus dalam kulit majemuknya, baik dalam keadaan utuh maupun besarnya sama atau lebih dari 3/4 bagian kulit majemuk yang utuh
5. Biji coklat  
biji kopi yang setengah atau lebih bagian luarnya berwarna coklat, yang lebih tua dari populasinya, baik yang mengkilap maupun keriput. Biji coklat yang pecah dinilai sebagai biji pecah
6. Kulit kopi (*husk*) ukuran besar  
kulit majemuk (*pericarp*) dari kopi gelondong dengan atau tanpa kulit ari (*silver skin*) dan kulit tanduk (*parchment*) di dalamnya, yang berukuran lebih besar dari 3/4 bagian kulit majemuk yang utuh
7. Kulit kopi (*husk*) ukuran sedang  
kulit majemuk dari kopi gelondong dengan atau tanpa kulit ari dan kulit tanduk di dalamnya, yang berukuran 1/2 sampai dengan 3/4 bagian kulit majemuk yang utuh
8. Kulit kopi (*husk*) ukuran kecil  
kulit majemuk dari kopi gelondong dengan atau tanpa kulit ari dan kulit tanduk di dalamnya, yang berukuran kurang dari 1/2 bagian kulit majemuk yang utuh

9. Biji berkulit tanduk  
biji kopi yang masih terbungkus oleh kulit tanduk, yang membungkus biji tersebut dalam keadaan utuh maupun besarnya sama dengan atau lebih besar dari 3/4 bagian kulit tanduk utuh
10. Kulit tanduk ukuran besar  
kulit tanduk yang terlepas atau tidak terlepas dari biji kopi, yang berukuran lebih besar dari 3/4 bagian kulit tanduk utuh
11. Kulit tanduk ukuran sedang  
kulit tanduk yang terlepas atau tidak terlepas dari biji kopi yang berukuran 1/2 sampai 3/4 bagian kulit tanduk utuh
12. Kulit tanduk ukuran kecil  
kulit tanduk yang terlepas dari biji kopi yang berukuran kurang dari 1/2 bagian kulit tanduk yang utuh
13. Biji pecah  
biji kopi yang tidak utuh yang besarnya sama atau kurang dari 3/4 bagian biji yang utuh
14. Biji muda  
biji kopi yang kecil dan keriput pada seluruh bagian luarnya
15. Biji berlubang satu  
biji kopi yang berlubang satu akibat serangan serangga
16. Biji berlubang lebih dari satu  
biji kopi yang berlubang lebih dari satu akibat serangan serangga
17. Biji bertutul-tutul  
biji kopi yang bertutul-tutul pada 1/2 (setengah) atau lebih bagian luarnya.  
Ketentuan ini hanya berlaku untuk kopi yang diolah dengan cara pengolahan basah
18. Ranting, tanah, atau batu berukuran besar  
ranting, tanah, atau batu berukuran panjang atau diameter lebih dari 10 mm
19. Ranting, tanah, atau batu berukuran sedang  
ranting, tanah, atau batu berukuran panjang atau diameter 5 mm -10 mm

20. Ranting, tanah, atau batu berukuran kecil  
ranting, tanah, atau batu berukuran panjang atau diameter kurang dari 5 mm

### 2.3. Citra Digital

Citra atau gambar merupakan kata yang berasal dari kata *image* dalam bahasa Inggris. Citra sebagai salah satu komponen multimedia memegang peranan sangat penting sebagai bentuk informasi visual. Citra memiliki karakteristik yang tidak dimiliki oleh data textual dikarenakan lebih kaya akan informasi. Sebuah gambar dapat memberikan informasi dapat memberikan informasi yang lebih banyak daripada informasi yang disajikan dalam bentuk textual.

Secara harfiah, citra merupakan sebuah data visual dalam bidang dwimatra (dimensi) (Widiarti, 2013). Citra dapat didefinisikan sebagai sebuah fungsi kontinyu dari intensitas dalam bidang dua dimensi dimana setiap titik dapat dituliskan

$$0 < f(x,y) < \infty \quad 2.1$$

dimana  $f(x,y)$  merupakan intensitas cahaya pada lokasi  $(x,y)$  (Gonzales dan Woods, 1992). Dikarenakan citra merupakan data visual dwimatra yang dapat direpresentasikan dalam dimensi panjang, dan lebar, maka seperti halnya matriks, citra dapat direpresentasikan kedalam sebuah matriks  $m \times n$  dengan intensitas piksel sebagai komponen penyusunnya.

## 2.4. Pemrosesan Citra

### 2.4.1. Grayscale

Secara umum, proses konversi dari citra warna ke citra keabuan menggunakan persamaan

$$I = axR + bxG + cxB \quad 2.2$$

Dimana R adalah nilai yang menunjukkan komponen merah, G adalah nilai yang menunjukkan komponen Hijau, dan B menunjukkan komponen Biru. Jika nilai a, b dan c dibuat sama, dan sebagai contoh digunakan nilai R=50 G=70 dan B=60 maka akan didapatkan

$$I = \frac{50 + 70 + 60}{3}$$

Dan berikut merupakan rumus umum yang biasa dipakai dalam konversi ke citra keabuan yang merupakan formula yang dikembangkan oleh *National Television System Committee* (NTSC)

$$I = 0.2989 \times R + 0.5870 \times G + 0.1141 \times B \quad 2.3$$

### 2.4.2. Deteksi Tepi : Sobel Operator

Merupakan salah satu operator yang digunakan dalam proses pendekripsi tepi pada citra. Operator sobel lebih sensitif terhadap tepi diagonal daripada tepi vertikal dan horizontal. Hal ini berbeda dengan operator Prewitt, yang lebih sensitif terhadap tepi vertikal dan horizontal. Matriks operator Sobel dapat dilihat pada gambar berikut.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Gambar 2.1 – Kernel Konvolusi Sobel

### 2.4.3. Operasi Morfologi : Dilasi

Morfologi citra merupakan sebuah bentuk dan struktur dari sebuah obyek yakni sebuah cara menggambarkan atau menganalisa sebuah objek digital dalam sebuah citra (Parker, 1997). Proses morfologi dilakukan dengan cara mengkombinasikan titik-titik dalam citra digital dengan sebuah *structuring element* (kernel).

Dilasi merupakan sebuah proses untuk memperbesar ukuran obyek dalam citra. Dilasi merupakan operasi yang berkebalikan dengan erosi. Dilasi merupakan sebuah proses dalam merubah piksel yang berupa latar (bernilai 0 dalam citra biner) menjadi objek (bernilai 1 dalam citra biner).

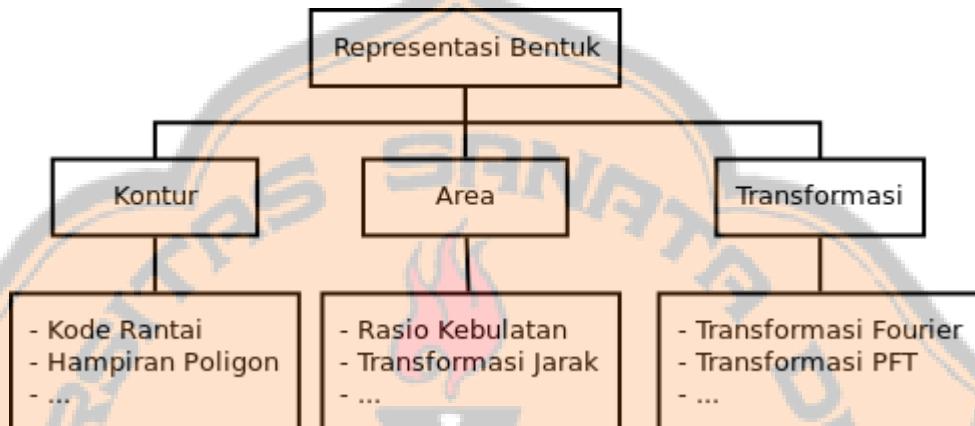
$$D(A, B) = A \oplus B$$

2.4

Dimana A merupakan citra, dan B merupakan elemen penyusunnya. Secara garis besar, proses yang dilakukan adalah melakukan proses dilasi pada setiap titik pada citra A sesuai dengan elemen penyusun B.

## 2.5. Ekstraksi Ciri : Basic Region Descriptor

Beberapa pemrosesan citra mengacu pada citra biner. Fitur suatu objek dalam citra biner merupakan karakteristik yang melekat pada objek. Fitur bentuk merupakan suatu fitur (ciri) yang diperoleh melalui bentuk objek, dan dapat dinyatakan melalui kontur, area, dan transformasi.

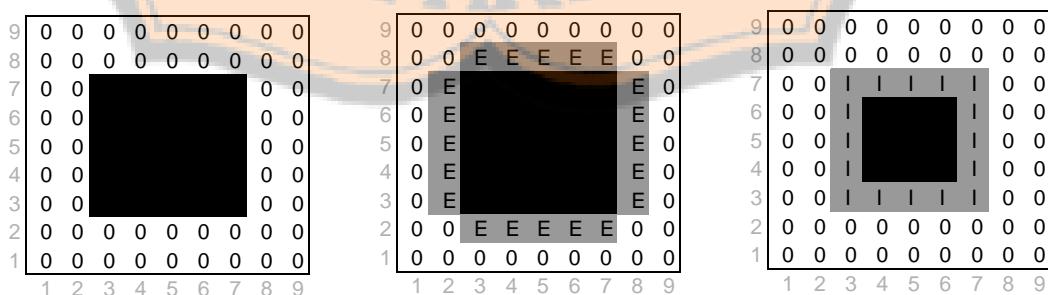


Gambar - 2.2 Representasi Bentuk (Kadir, 2013)

Gambar 2.2 merupakan jenis-jenis dari representasi bentuk melalui kontur, area, atau transformasi yang dilakukan terhadap citra biner guna mendapatkan fitur bentuk dari objek pada citra tersebut.

### 2.5.1. Kontur

Mengikuti kontur (*contour following*) merupakan suatu metode yang digunakan untuk mendapatkan tepi objek. Dalam penerapannya, terdapat 2 jenis kontur, yakni kontur internal dan kontur eksternal.

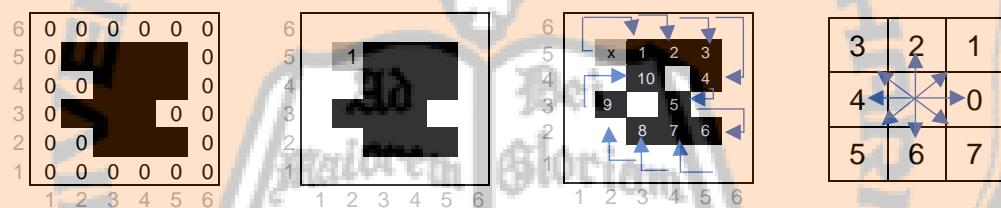


Gambar 2.3 - Gambar objek asli (**kiri**), kontur eksternalnya (**tengah**), dan kontur internal (**kanan**) (Kadir, 2013)

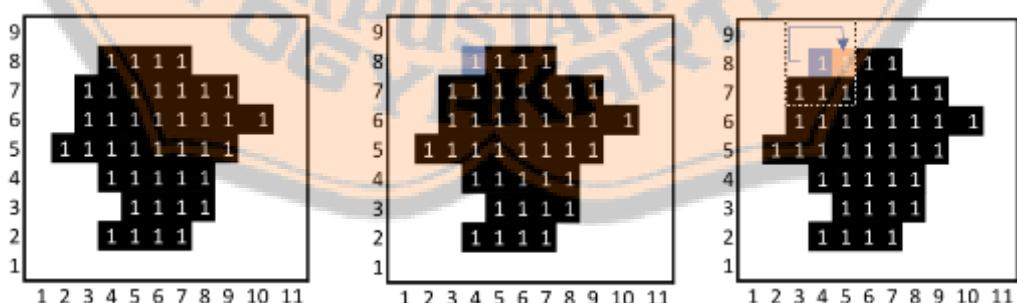
Pada gambar 2.3, ‘E’ menunjukkan kontur eksternal dari sebuah objek, dan ‘I’ merupakan kontur internal dari objek. Terlihat bahwa piksel yang menjadi bagian kontur eksternal terletak di luar objek, sedangkan piksel yang menjadi bagian kontur internal terletak didalam objek itu sendiri.

### 2.5.2. Kontur Internal

Salah satu cara untuk mendapatkan kontur internal yang telah diurutkan menurut letak piksel adalah dengan menggunakan algoritma pelacakan kontur Moore. Secara umum, algoritma pelacakan kontur Moore menggunakan konsep ketetanggan dari sebuah piksel dalam mencari konturnya. Dengan menggunakan 8 ketetanggaan, penelusuran akan dimulai dari titik paling atas kanan dari sebuah objek.

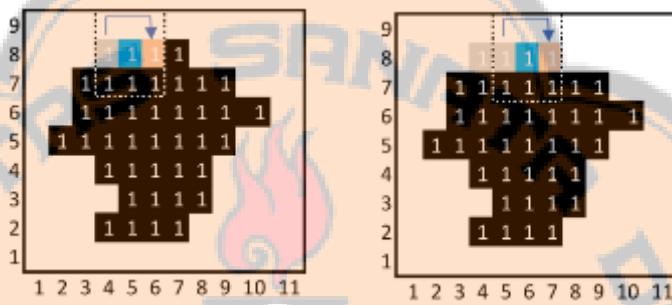


**Gambar 2.4 -** Visualisasi jalan kerja algoritma pelacakan kontur Moore. Dari kiri ke kanan : citra objek, proses pelacakan kontur dimulai dari titik (2,5), kontur moore yang dihasilkan, dan template 8 arah ketetanggan yang digunakan dalam pelacakan kontur (Kadir, 2013)



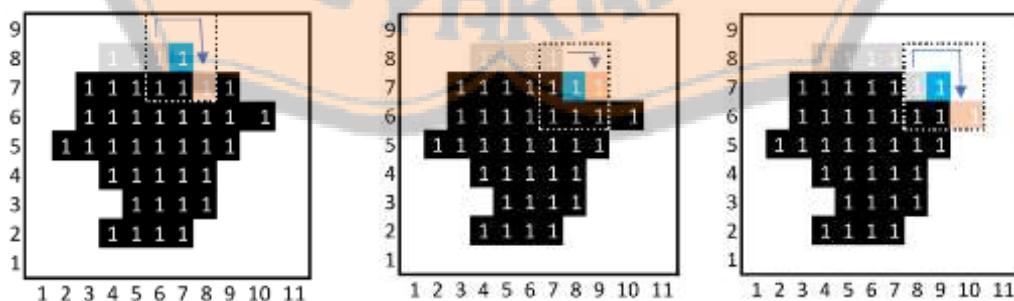
**Gambar 2.5 -** Visualisasi Pelacakan Kontur Moore kiri ke kanan : Citra Asli, Titik awal pelacakan kontur, iterasi ke-1

Sebagai contoh, pada gambar 2.7 (kiri) merupakan sebuah citra yang akan dicari konturnya menggunakan algoritma pelacakan kontur Moore. Yang pertama dilakukan adalah mencari piksel objek tersebut dengan posisi paling kiri dan paling atas, sehingga didapatkan titik (4,8) – biru. Dari titik tersebut, masuk iterasi pertama pelacakan. Dari titik awal (4,8) sisi kiri, lakukan pencarian piksel hitam (1) pertama yang searah jarum jam, didapatkan titik (5,8).



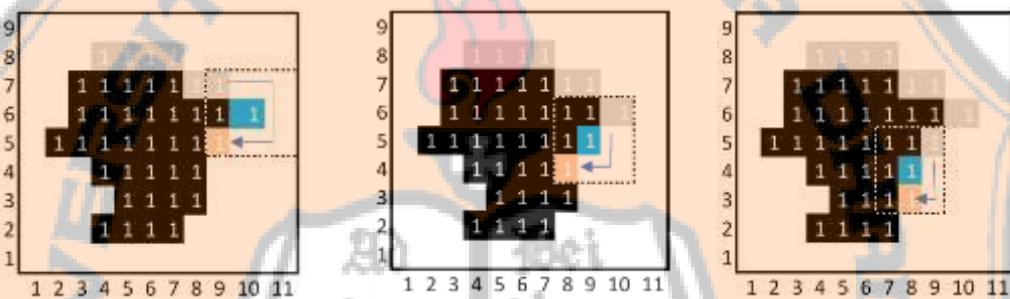
**Gambar 2.6 - Visualisasi Pelacakan Kontur Moore kiri ke kanan : Iterasi ke-2,**  
3

Pada iterasi ke-2, pelacakan dimulai dari (5,8), dengan adanya titik (4,8) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (4,8) hingga mendapatkan piksel hitam (1) pertama yaitu (6,8). Pada iterasi ke-3, pelacakan dimulai dari (6,8), dengan adanya titik (5,8) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (5,8) hingga mendapatkan piksel hitam (1) pertama yaitu (7,8).



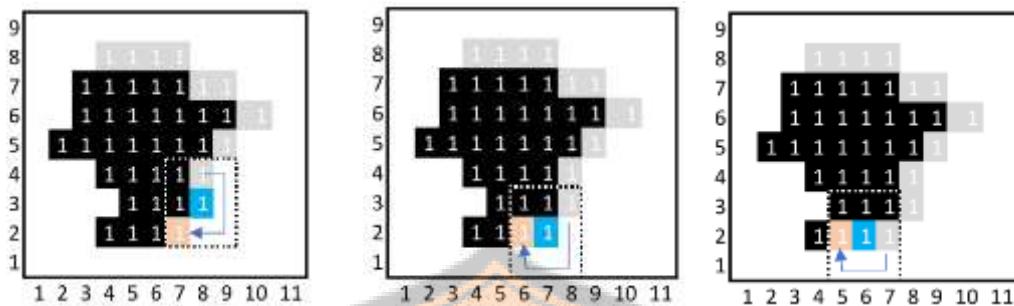
**Gambar 2.7 - Visualisasi Pelacakan Kontur Moore kiri ke kanan : Iterasi ke-4,**  
5, dan 6

Pada iterasi ke-4, pelacakan dimulai dari (7,8), dengan adanya titik (6,8) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (6,8) hingga mendapatkan piksel hitam (1) pertama yaitu (8,7). Pada iterasi ke-3, pelacakan dimulai dari (8,7), dengan adanya titik (6,8) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (6,8) hingga mendapatkan piksel hitam (1) pertama yaitu (9,7). Pada iterasi ke-4, pelacakan dimulai dari (9,7), dengan adanya titik (8,7) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (8,7) hingga mendapatkan piksel hitam (1) pertama yaitu (10,6).



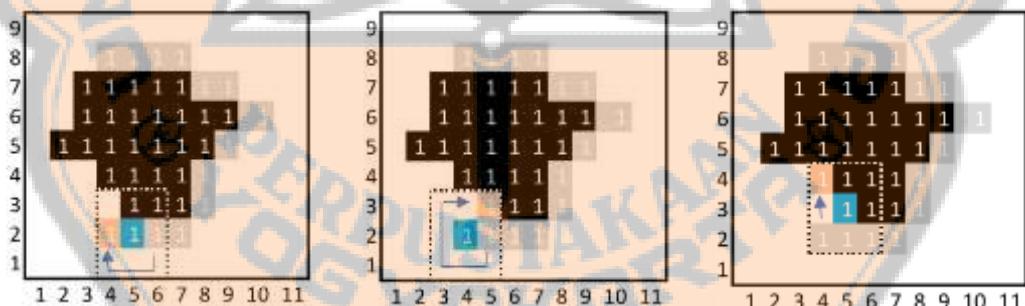
**Gambar 2.8 - Visualisasi Pelacakan Kontur Moore kiri ke kanan : Iterasi ke-7, 8, dan 9**

Pada iterasi ke-7, pelacakan dimulai dari (10,6), dengan adanya titik (9,7) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (9,7) hingga mendapatkan piksel hitam (1) pertama yaitu (9,5). Pada iterasi ke-8, pelacakan dimulai dari (9,5), dengan adanya titik (10,6) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (10,6) hingga mendapatkan piksel hitam (1) pertama yaitu (8,4). Pada iterasi ke-9, pelacakan dimulai dari (8,4), dengan adanya titik (9,5) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (9,5) hingga mendapatkan piksel hitam (1) pertama yaitu (8,3).



**Gambar 2.9** - Visualisasi Pelacakan Kontur Moore **kiri ke kanan** : Iterasi ke-10, 11, dan 12

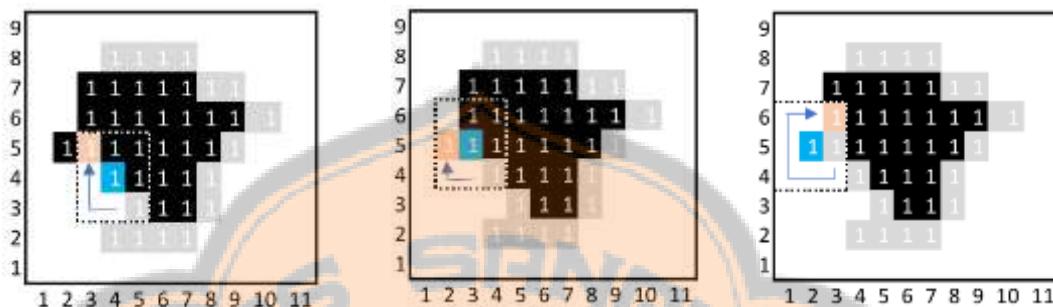
Pada iterasi ke-10, pelacakan dimulai dari (8,3), dengan adanya titik (8,4) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (8,4) hingga mendapatkan piksel hitam (1) pertama yaitu (7,2). Pada iterasi ke-11, pelacakan dimulai dari (7,2), dengan adanya titik (8,3) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (8,3) hingga mendapatkan piksel hitam (1) pertama yaitu (6,2). Pada iterasi ke-12, pelacakan dimulai dari (6,2), dengan adanya titik (7,2) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (7,2) hingga mendapatkan piksel hitam (1) pertama yaitu (5,2).



**Gambar 2.10** - Visualisasi Pelacakan Kontur Moore **kiri ke kanan** : Iterasi ke-13, 14, dan 15

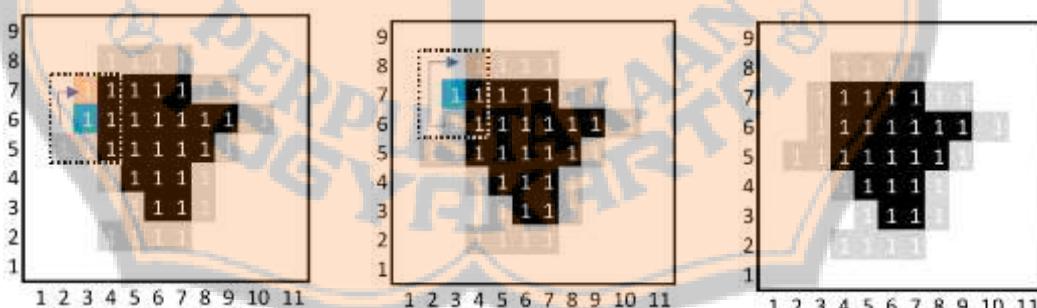
Pada iterasi ke-13, pelacakan dimulai dari (5,2), dengan adanya titik (6,2) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (6,2) hingga mendapatkan piksel hitam (1) pertama yaitu (4,2). Pada iterasi ke-14, pelacakan dimulai dari (4,2), dengan adanya titik (5,2) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (5,2) hingga mendapatkan piksel hitam (1) pertama yaitu (5,3). Pada iterasi ke-15, pelacakan dimulai dari (5,3), dengan adanya titik

(4,2) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (4,2) hingga mendapatkan piksel hitam (1) pertama yaitu (4,4).



**Gambar 2.11 - Visualisasi Pelacakan Kontur Moore kiri ke kanan : Iterasi ke-16, 17, dan 18**

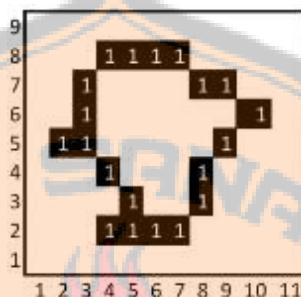
Pada iterasi ke-16, pelacakan dimulai dari (4,4), dengan adanya titik (5,2) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (5,2) hingga mendapatkan piksel hitam (1) pertama yaitu (3,5). Pada iterasi ke-17, pelacakan dimulai dari (3,5), dengan adanya titik (4,4) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (4,4) hingga mendapatkan piksel hitam (1) pertama yaitu (2,5). Pada iterasi ke-18, pelacakan dimulai dari (2,5), dengan adanya titik (4,3,5) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (3,5) hingga mendapatkan piksel hitam (1) pertama yaitu (3,6).



**Gambar 2.12 - Visualisasi Pelacakan Kontur Moore kiri ke kanan : Iterasi ke-19, 20, dan objek dengan kontur internalnya**

Pada iterasi ke-19, pelacakan dimulai dari (3,6), dengan adanya titik (5,2) sebagai titik kontur sebelumnya, maka arah pencarian dimulai dari (5,2) hingga mendapatkan piksel hitam (1) pertama yaitu (3,7). Pada iterasi ke-20, pelacakan dimulai dari (3,7), dengan adanya titik (3,6) sebagai titik kontur sebelumnya, maka

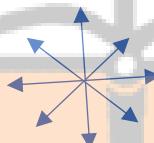
arah pencarian dimulai dari (3,6) hingga mendapatkan piksel hitam (1) pertama yaitu (4,8) yang merupakan titik awal dari iterasi pelacakan kontur ini. Dengan demikian, kontur dapat dinotasikan dalam pasangan koordinat : (4,8), (5,8), (6,8), (7,8), (8,7), (9,7), (10,6), (9,5), (8,4), (8,3), (7,2), (6,2), (5,2), (4,2), (5,3), (4,4), (3,5), (2,5), (3,6), (3,7), (4,8) yang merupakan titik awal.



Gambar 2.13 - Kontur internal objek hasil pelacakan

### 2.5.3. *Chain Code*

Merupakan contoh representasi kontur yang diperkenalkan oleh Freeman pada 1961. dinyatakan dengan menggunakan pendekatan 8-ketetanggan dengan angka sebagai penanda sesuai arah.



3	2	1
4		0
5	6	7

Gambar 2.14 - Arah rantai kode beserta kodennya (Kadir, 2013)

Untuk mempermudah perolehan kode rantai piksel menjadi tetangga, perlu adanya pembuatan indeks. Indeks yang digunakan merupakan nomor yang akan diberikan dalam sebuah 8 ketetanggan yang menunjukkan posisi dalam ketetanggan tersebut.

1	2	3
4	5	6
7	8	9

3	2	1
4		0
5	6	7

Gambar 2.15 - Kiri : Representasi indeks kode rantai, Kanan : kode rantai

Adapun indeks dapat dihitung dengan

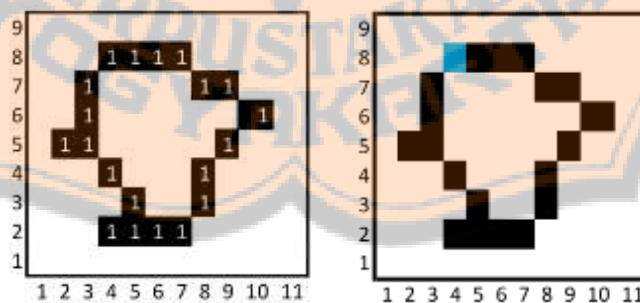
$$\text{indeks} = 3\Delta y + \Delta x + 5 \quad 2.5$$

Dalam hal ini  $\Delta x$  merupakan selisih nilai kolom 2 piksel yang bertetangga dan  $\Delta y$  merupakan selisih nilai baris 2 piksel yang bertetangga. Adapun hubungan dari indeks dan kode rantai 2 piksel yang bertetangga dinyatakan dalam tabel 2.5.

**Tabel 2.4** Indeks dan kode rantai dua piksel bertetangga (Kadir, 2013)

$\Delta x$	$\Delta y$	Kode Rantai	Indeks (2.6)
0	+1	6	8
0	-1	2	2
-1	+1	5	7
-1	-1	3	1
+1	+1	7	9
+1	-1	1	3
-1	0	4	4

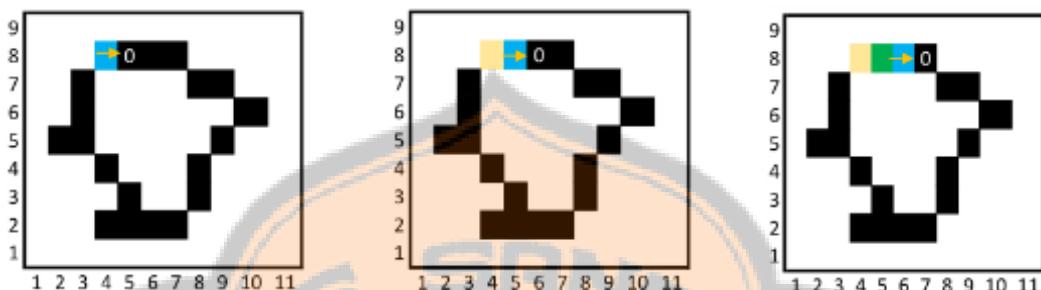
Sebagai contoh, pada sebuah objek yang sudah didapatkan kontur dengan titik (4,8), (5,8), (6,8), (7,8), (8,7), (9,7), (10,6), (9,5), (8,4), (8,3), (7,2), (6,2), (5,2), (4,2), (5,3), (4,4), (3,5), (2,5), (3,6), (3,7), (4,8) (lihat : 2.5.2) akan dilakukan pencarian rantai kode objek tersebut. Maka pertama dilakukan adalah mencari titik awal untuk pencarian rantai kode. Pada contoh ini, pelacakan kode rantai dimulai dari piksel (4,8) yang merupakan titik paling kiri dan paling atas objek.



**Gambar 2.16 - Kiri : Kontur Objek, Kanan : Titik awal penelusuran**

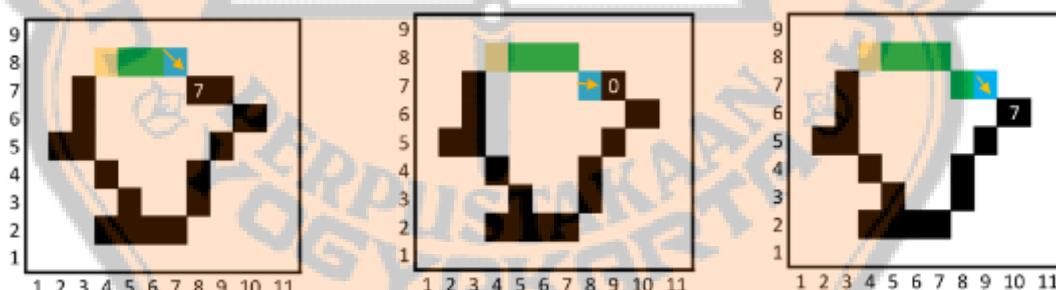
Setelah titik mulai didapatkan, iterasi akan dimulai ke arah kanan dari titik (4,8), yakni (5,8) sehingga didapatkan kode 0, dengan indeks 4. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (5,8), sehingga didapatkan titik (6,8),

dinyatakan dengan kode 0, dengan indeks 4. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (6,8), sehingga didapatkan titik (7,8), dinyatakan dengan kode 0, dengan indeks 4.



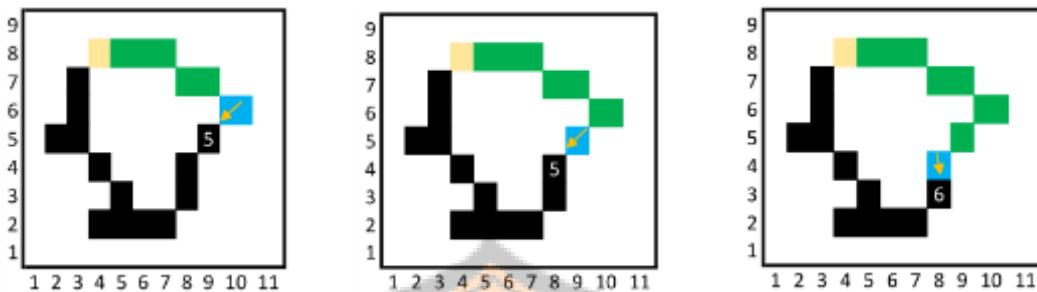
**Gambar 2.17 - Kiri ke Kanan :** Visualisasi pencarian kode rantai iterasi ke-1, 2, dan 3

Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (7,8), sehingga didapatkan titik (8,7), dinyatakan dengan kode 7, dengan indeks 9. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (8,7), sehingga didapatkan titik (9,7), dinyatakan dengan kode 0, dengan indeks 4. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (9,7), sehingga didapatkan titik (10,6), dinyatakan dengan kode 7, dengan indeks 9.



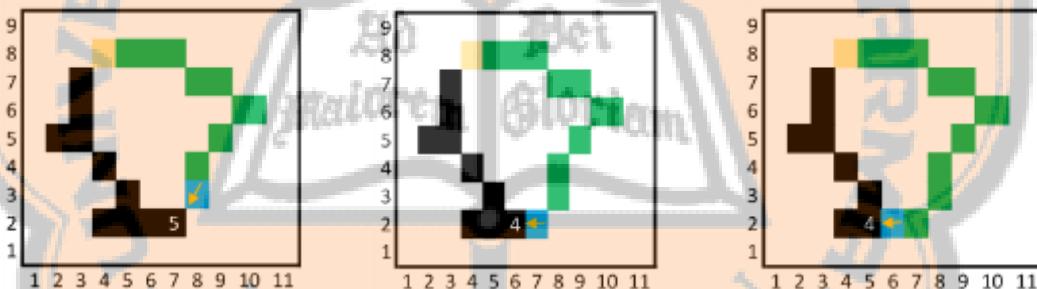
**Gambar 2.18 - Kiri ke Kanan :** Visualisasi pencarian kode rantai iterasi ke-4, 5, dan 6

Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (10,6), sehingga didapatkan titik (9,5), dinyatakan dengan kode 5, dengan indeks 6. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (9,5), sehingga didapatkan titik (8,4), dinyatakan dengan kode 5, dengan indeks 6. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (8,4), sehingga didapatkan titik (8,3), dinyatakan dengan kode 6, dengan indeks 8.



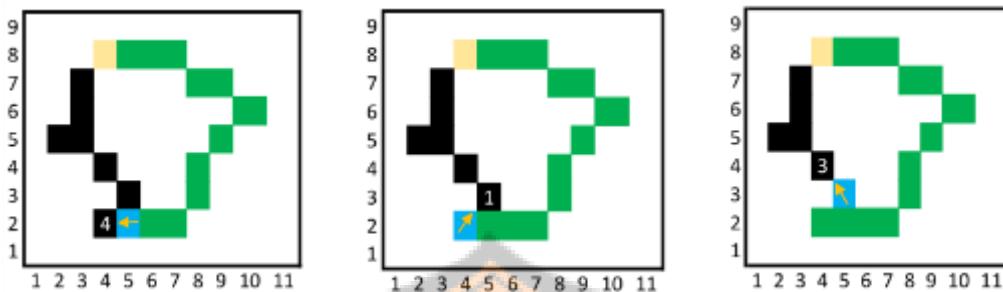
**Gambar 2.19 - Kiri ke Kanan :** Visualisasi pencarian kode rantai iterasi ke-7, 8, dan 9

Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (8,3), sehingga didapatkan titik (7,2), dinyatakan dengan kode 5 dengan indeks 6. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (7,2), sehingga didapatkan titik (6,2), dinyatakan dengan kode 4 dengan indeks 4. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (6,2), sehingga didapatkan titik (5,2), dinyatakan dengan kode 4 dengan indeks 4.



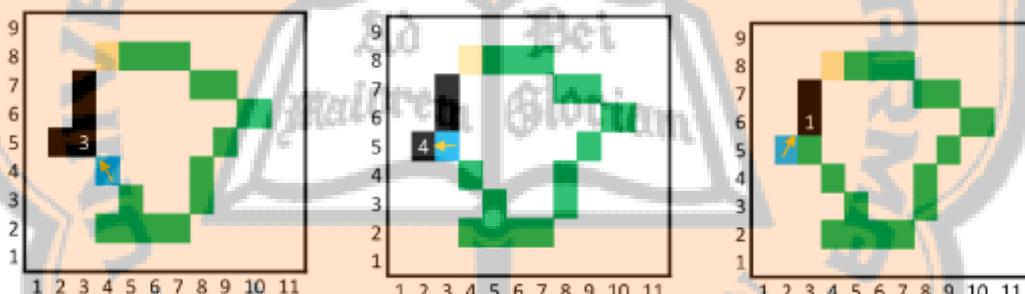
**Gambar 2.20 - Kiri ke Kanan :** Visualisasi pencarian kode rantai iterasi ke-10, 11, dan 12

Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (5,2), sehingga didapatkan titik (4,2), dinyatakan dengan kode 4 dengan indeks 4. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (4,2), sehingga didapatkan titik (5,3), dinyatakan dengan kode 1 dengan indeks 3. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (5,3), sehingga didapatkan titik (4,4), dinyatakan dengan kode 3 dengan indeks 1.



**Gambar 2.21 - Kiri ke Kanan :** Visualisasi pencarian kode rantai iterasi ke-13, 14, dan 15

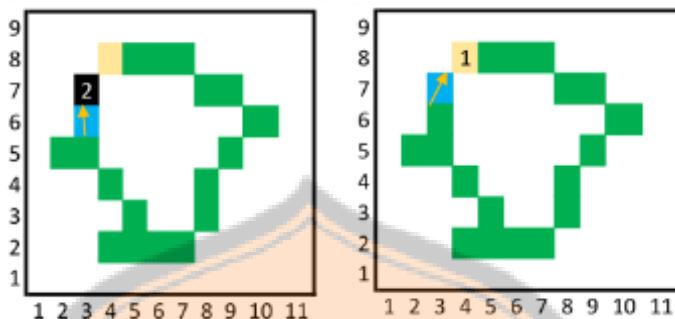
Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (4,4), sehingga didapatkan titik (3,5), dinyatakan dengan kode 3 dengan indeks 1. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (3,5), sehingga didapatkan titik (2,5), dinyatakan dengan kode 4 dengan indeks 4. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (2,5), sehingga didapatkan titik (3,6), dinyatakan dengan kode 1 dengan indeks 3.



**Gambar 2.22 - Kiri ke Kanan :** Visualisasi pencarian kode rantai iterasi ke-16, 17, dan 18

Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (3,6), sehingga didapatkan titik (3,7), dinyatakan dengan kode 2 dengan indeks 2. Iterasi selanjutnya dilakukan dari titik sebelumnya yakni (3,7), sehingga didapatkan titik (4,8) yang merupakan

titik awal penelusuran kode rantai, dinyatakan dengan kode 1 dengan indeks 3.



**Gambar 2.23 - Kiri ke Kanan :** Visualisasi pencarian kode rantai iterasi ke-19 dan 20

**Tabel 2.5 -** Tabel hasil kode rantai dan indeksnya dari kontur objek masukan

No	x	y	Kode	Indeks
1	4	8	start	start
2	5	8	0	6
3	6	8	0	6
4	7	8	0	6
5	8	7	7	9
6	9	7	0	6
7	10	6	7	9
8	9	5	5	7
9	8	4	5	7
10	8	3	6	8
11	7	2	5	7
12	6	2	4	4
13	5	2	4	4
14	4	2	4	4
15	5	3	1	3
16	4	4	3	1
17	3	5	3	1
18	2	5	4	4
19	3	6	1	3
20	3	7	2	2
1	4	8	1	3

Dalam proses penelusuran kode rantai tersebut, dapat dituliskan kode rantai berupa 00070755654441334121 dari titik (4,8), dapat dilihat pada tabel 2.5.

#### 2.5.4. Ciri dari Penghitungan Kontur Internal dan *Chain Code*

##### 2.5.4.1. Luas

Cara sederhana untuk menghitung luas adalah dengan melakukan penghitungan terhadap jumlah piksel on (1) pada objek tersebut. Namun terdapat metode lain dengan pendekatan penghitungan luas yang menggunakan *chain code* seperti yang dilakukan Putra (2010). Dari ketentuan kode rantai yang didapatkan, penghitungan luas dinyatakan dengan

$$\text{Kode 0} \rightarrow \text{Area} = \text{Area} + Y$$

$$\text{Kode 1} \rightarrow \text{Area} = \text{Area} + (Y + 0.5)$$

$$\text{Kode 2} \rightarrow \text{Area} = \text{Area} + 0$$

$$\text{Kode 3} \rightarrow \text{Area} = \text{Area} - (Y + 0.5)$$

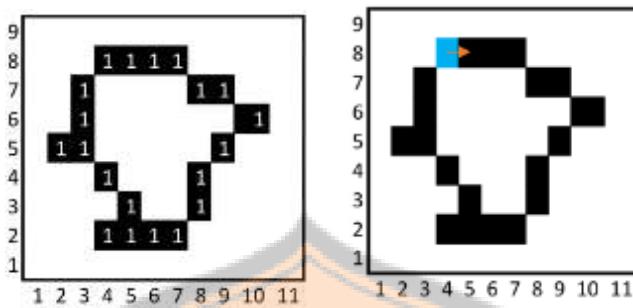
$$\text{Kode 4} \rightarrow \text{Area} = \text{Area} - Y$$

$$\text{Kode 5} \rightarrow \text{Area} = \text{Area} - (Y + 0.5)$$

$$\text{Kode 6} \rightarrow \text{Area} = \text{Area} + 0$$

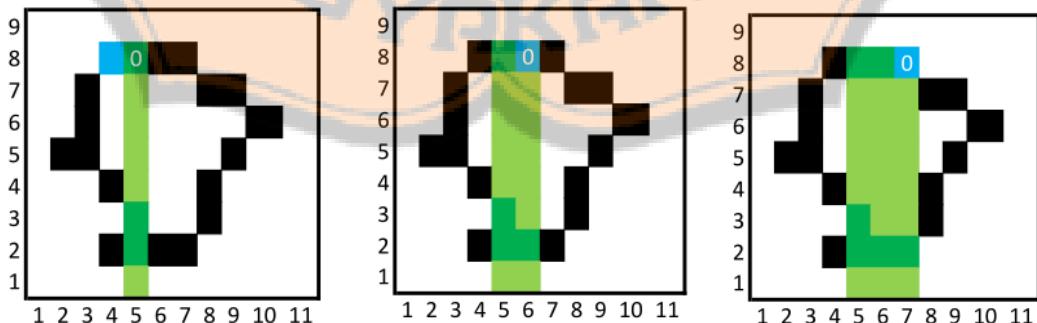
$$\text{Kode 7} \rightarrow \text{Area} = \text{Area} + (Y - 0.5)$$

Secara umum, penghitungan luas didasari dengan koordinat piksel penyusun objek dan arah kode rantainya. Dimana arah kanan dari sebuah piksel dilakukan penambahan berdasarkan ordinatnya (y). Sedangkan pada arah kiri piksel tersebut dilakukan pengurangan berdasarkan ordinatnya. Penambahan dan pengurangan ‘0,5’ dilakukan pada kode rantai yang diagonal yakni 1 3 5 dan 7. Sedangkan pada arah horizontal (kode 0 dan 4), akan dilakukan penambahan dan pengurangan luas dengan ordinatnya (y). Sedangkan pada arah horizontal (kode 2 dan 6), tidak dilakukan penambahan dan pengurangan penghitungan luas karena tidak ada perubahan absisnya (x). Untuk penjelasan dapat dilihat pada contoh berikut.



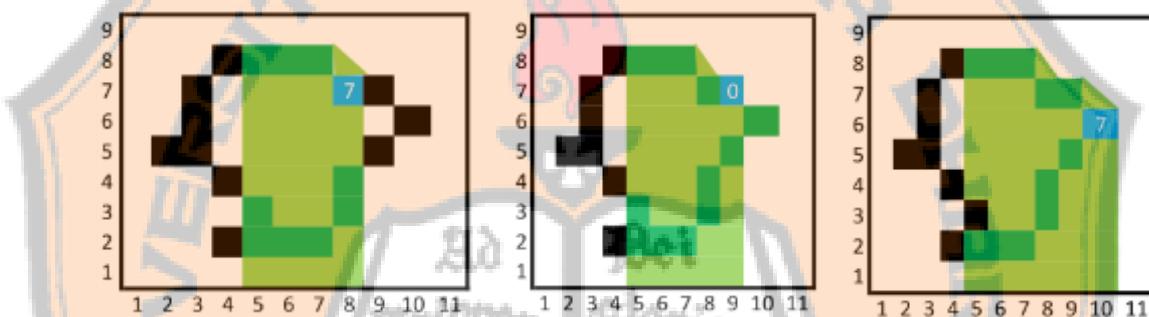
**Gambar 2.24 - Kiri ke Kanan :** Visualisasi pencarian luas dengan kode rantai, Kontur objek (kiri) dan Titik (4,8) sebagai titik awal (kanan)

Sebuah citra objek dengan kontur (4,8), (5,8), (6,8), (7,8), (8,7), (9,7), (10,6), (9,5), (8,4), (8,3), (7,2), (6,2), (5,2), (4,2), (5,3), (4,4), (3,5), (2,5), (3,6), (3,7), (4,8) dengan kode 00070755654441334121 akan dilakukan penghitungan luas dengan pendekatan kode rantai. Yang pertama dilakukan adalah pencarian titik awal penelusuran dari kode rantai yakni (4,8). Iterasi selanjutnya adalah pada kode pertama (0) merupakan representasi kontur dengan koordinat (5,8), sehingga luas yang akan ditambahkan adalah 8 sesuai dengan ordinatnya, sehingga luasnya adalah *8 units*. Pada iterasi selanjutnya adalah kode ke-2 (0) merupakan representasi kontur dengan koordinat (6,8), sehingga luas yang akan ditambahkan adalah 8 sesuai dengan ordinatnya, sehingga luasnya adalah *16 units*. Pada iterasi selanjutnya adalah kode ke-3 (0) merupakan representasi kontur dengan koordinat (7,8), sehingga luas yang akan ditambahkan adalah 8 sesuai dengan ordinatnya, sehingga luasnya adalah *24 units*



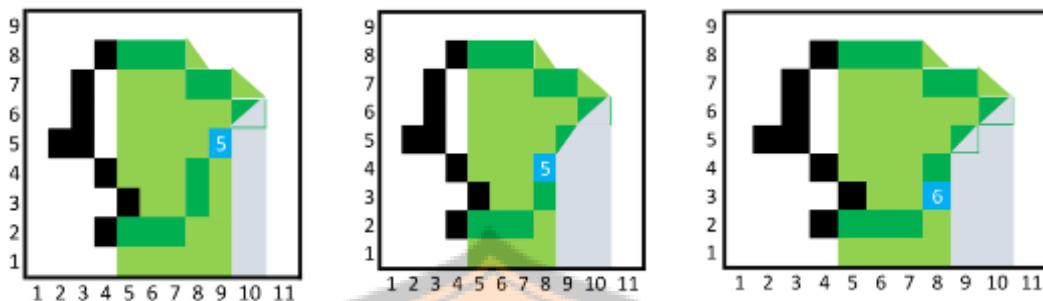
**Gambar 2.25 - Kiri ke Kanan :** Visualisasi pencarian luas dengan kode rantai, Iterasi 1, 2 dan 3

Pada iterasi selanjutnya adalah kode ke-4 (7) merupakan representasi kontur dengan koordinat (8,7), sehingga luas yang akan ditambahkan adalah 7 sesuai dengan ordinatnya dan 0.5 karena arah diagonal, sehingga luasnya adalah  $31.5 \text{ units}$ . Pada iterasi selanjutnya adalah kode ke-5 (0) merupakan representasi kontur dengan koordinat (9,7), sehingga luas yang akan ditambahkan adalah 7 sesuai dengan ordinatnya, sehingga luasnya adalah  $38.5 \text{ units}$ . Pada iterasi selanjutnya adalah kode ke-6 (7) merupakan representasi kontur dengan koordinat (10,6), sehingga luas yang akan ditambahkan adalah 6 sesuai dengan ordinatnya dan 0.5 karena arah diagonal, sehingga luasnya adalah  $45 \text{ units}$



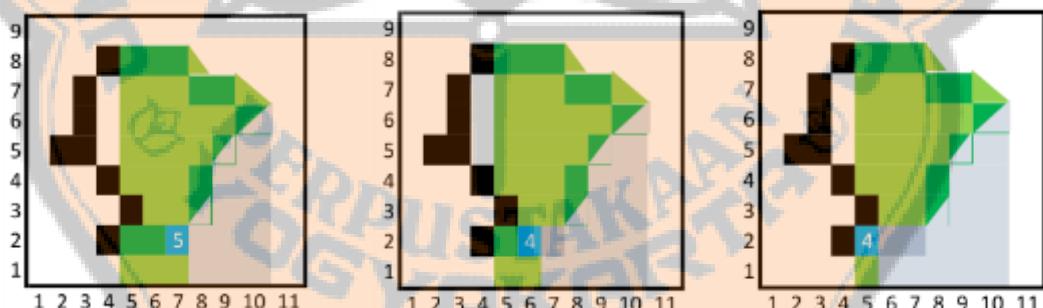
**Gambar 2.26 - Kiri ke Kanan :** Visualisasi pencarian luas dengan kode rantai, Iterasi 4, 5 dan 6

Pada iterasi selanjutnya adalah kode ke-7 (5) merupakan representasi kontur dengan koordinat (9,5), sehingga luas yang akan dikurangi adalah 5 sesuai dengan ordinatnya dan dikurangi 0.5 karena arah diagonal, sehingga luasnya adalah  $39.5 \text{ units}$ . Pada iterasi selanjutnya adalah kode ke-8 (5) merupakan representasi kontur dengan koordinat (8,4), sehingga luas yang akan dikurangi adalah 4 sesuai dengan ordinatnya dan dikurangi 0.5 karena arah diagonal, sehingga luasnya adalah  $35 \text{ units}$ . Pada iterasi selanjutnya adalah kode ke-9 (6) merupakan representasi kontur dengan koordinat (8,3), pada kode ini tidak dilakukan penambahan atau pengurangan luas karena tidak ada perubahan absis dari kontur sebelumnya, sehingga luasnya adalah  $35 \text{ units}$ .



**Gambar 2.27 - Kiri ke Kanan :** Visualisasi pencarian luas dengan kode rantai, Iterasi 7, 8 dan 9

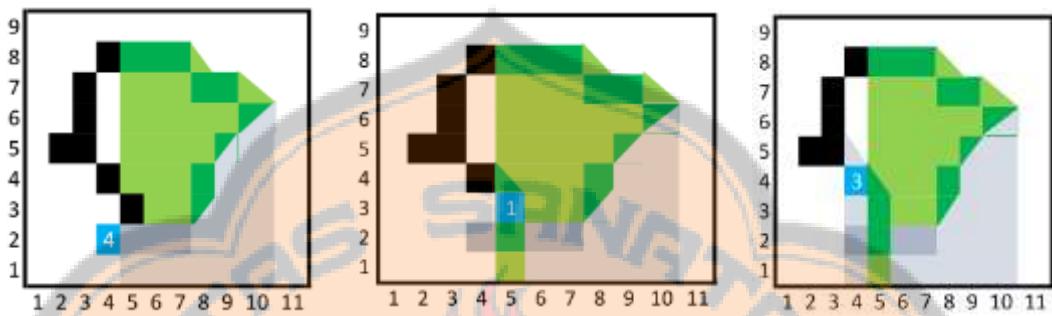
Pada iterasi selanjutnya adalah kode ke-10 (5) merupakan representasi kontur dengan koordinat (7,2), sehingga luas yang akan dikurangi adalah 2 sesuai dengan ordinatnya dan dikurangi 0.5 karena arah diagonal, sehingga luasnya adalah  $32.5 \text{ units}$ . Pada iterasi selanjutnya adalah kode ke-11 (4) merupakan representasi kontur dengan koordinat (6,2), sehingga luas yang akan dikurangi adalah 2 sesuai dengan ordinatnya, sehingga luasnya adalah  $30.5 \text{ units}$ . Pada iterasi selanjutnya adalah kode ke-12 (4) merupakan representasi kontur dengan koordinat (5,2), sehingga luas yang akan dikurangi adalah 2 sesuai dengan ordinatnya, sehingga luasnya adalah  $28.5 \text{ units}$ .



**Gambar 2.28 - Kiri ke Kanan :** Visualisasi pencarian luas dengan kode rantai, Iterasi 10, 11 dan 12

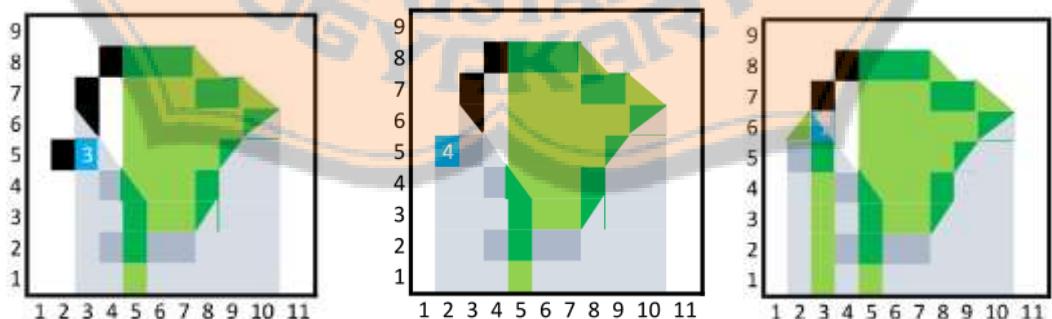
Pada iterasi selanjutnya adalah kode ke-13 (4) merupakan representasi kontur dengan koordinat (4,2), sehingga luas yang akan dikurangi adalah 2 sesuai dengan ordinatnya, sehingga luasnya adalah  $26.5 \text{ units}$ . Pada iterasi selanjutnya adalah kode ke-14 (1) merupakan representasi kontur dengan koordinat (5,3), sehingga luas yang akan ditambah adalah 3 sesuai dengan ordinatnya dan ditambah 0.5 karena

arah diagonal, sehingga luasnya adalah 30 *units*. . Pada iterasi selanjutnya adalah kode ke-15 (3) merupakan representasi kontur dengan koordinat (4,4), sehingga luas yang akan dikurangi adalah 4 sesuai dengan ordinatnya dan dikurangi 0.5 karena arah diagonal, sehingga luasnya adalah 25.5 *units*



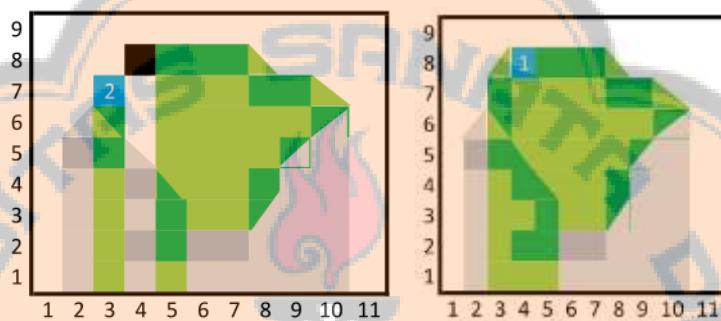
**Gambar 2.29 - Kiri ke Kanan :** Visualisasi pencarian luas dengan kode rantai, Iterasi 13, 14 dan 15

Pada iterasi selanjutnya adalah kode ke-16 (3) merupakan representasi kontur dengan koordinat (3,5), sehingga luas yang akan dikurangi adalah 5 sesuai dengan ordinatnya dan dikurangi 0.5 karena arah diagonal, sehingga luasnya adalah 20 *units*. Pada iterasi selanjutnya adalah kode ke-17 (4) merupakan representasi kontur dengan koordinat (2,5), sehingga luas yang akan dikurangi adalah 5 sesuai dengan ordinatnya, sehingga luasnya adalah 15 *units*. Pada iterasi selanjutnya adalah kode ke-18 (1) merupakan representasi kontur dengan koordinat (3,6), sehingga luas yang akan ditambahkan adalah 6 sesuai dengan ordinatnya dan ditambah 0.5 karena arah diagonal, sehingga luasnya adalah 21.5 *units*



**Gambar 2.30 - Kiri ke Kanan :** Visualisasi pencarian luas dengan kode rantai, Iterasi 16, 17 dan 18

Pada iterasi selanjutnya adalah kode ke-19 (2) merupakan representasi kontur dengan koordinat (3,7), pada kode ini tidak dilakukan penambahan atau pengurangan luas karena tidak ada perubahan absis dari kontur sebelumnya, sehingga luasnya adalah 21.5 *units*. Pada iterasi selanjutnya adalah kode ke-20 (1) merupakan representasi kontur dengan koordinat (4,8) yang merupakan titik awal penelusuran, sehingga luas yang akan ditambahkan adalah 8 sesuai dengan ordinatnya dan ditambah 0.5 karena arah diagonal, sehingga luasnya adalah 30 *units*



**Gambar 2.31 - Kiri ke Kanan :** Visualisasi pencarian luas dengan kode rantai, Iterasi 19 dan 20

**Tabel 2.6 -** Tabel hasil penghitungan luas berdasarkan kode rantai dan koordinat penyusun kontur

No	x	y	Kode	(+/-) Luas
1	4	8	start	0
2	5	8	0	8
3	6	8	0	8
4	7	8	0	8
5	8	7	7	7,5
6	9	7	0	7
7	10	6	7	6,5
8	9	5	5	-5,5
9	8	4	5	-4,5
10	8	3	6	0
11	7	2	5	-2,5
12	6	2	4	-2
13	5	2	4	-2
14	4	2	4	-2
15	5	3	1	3,5
16	4	4	3	-4,5
17	3	5	3	-5,5
18	2	5	4	-5
19	3	6	1	6,5
20	3	7	2	0
1	4	8	1	8,5
<b>Luas</b>				<b>30</b>

Sehingga hasil akhir dari penghitungan menggunakan penelusuran kode rantai didapatkan 30 *units*.

### 2.5.4.2. Perimeter

Perimeter atau keliling menyatakan panjang tepi suatu objek. Untuk melakukan penghitungan perimeter, terdapat 2 cara. Pada cara pertama, digunakan penghitungan berdasarkan jumlah piksel dari deteksi tepi. Cara ini memberikan hasil baik jika tepi objek terhubung dengan 4 ketetanggan, namun tidak dengan 8 ketetanggaan. Hal ini terjadi karena jarak antara dua piksel tidak bersifat konstan. Akan bernilai 1 jika 4 tetangga, dan  $\sqrt{2}$  pada 8 tetangga yang diagonal (Gambar 2.). Sehingga, digunakan pendekatan penghitungan perimeter dari chain code yang didapatkan dengan rumus

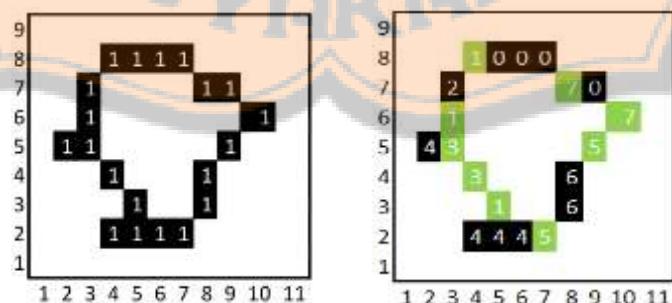
$$\text{Perimeter} = N_e + N_o\sqrt{2} \quad 2.6$$

Dengan  $N_e$  menyatakan jumlah kode genap dan  $N_o$  menyatakan jumlah kode ganjil.



**Gambar 2.32** - Jarak antar piksel pada 8 ketetanggan, bernilai 1 pada piksel horizontal dan vertikal (**kiri**), dan bernilai  $\sqrt{2}$  pada piksel yang diagonal (**kanan**) (Kadir, 2013)

Sebagai contoh sebuah objek dengan kontur (4,8), (5,8), (6,8), (7,8), (8,7), (9,7), (10,6), (9,5), (8,4), (8,3), (7,2), (6,2), (5,2), (4,2), (5,3), (4,4), (3,5), (2,5), (3,6), (3,7), (4,8) dengan kode 00070755654441334121 yang dimulai dari (4,8)



**Gambar 2.33** Visualisasi penghitungan perimeter (keliling) menggunakan kode rantai. Kontur Objek (kiri) dan representasi kode genap-ganjil (kanan)

Untuk setiap kode yang genap yakni 0, 2, 4, 6 yang merupakan kontur dengan arah vertikal dan horizontal akan ditambah 1 sebagai keliling. Pada kode rantai yang ganjil (1, 3, 5, 7) yang merupakan kontur dengan arah diagonal akan ditambahkan  $\sqrt{2}$  sebagai keliling. Hasil dapat dilihat pada tabel berikut.

**Tabel 2.7 - .** Hasil penghitungan perimeter (keliling) menggunakan kode rantai

No	x	y	Kode	+
1	4	8	start	0
2	5	8	0	1
3	6	8	0	1
4	7	8	0	1
5	8	7	7	1,414214
6	9	7	0	1
7	10	6	7	1,414214
8	9	5	5	1,414214
9	8	4	6	1
10	8	3	6	1
11	7	2	5	1,414214
12	6	2	4	1
13	5	2	4	1
14	4	2	4	1
15	5	3	1	1,414214
16	4	4	3	1,414214
17	3	5	3	1,414214
18	2	5	4	1
19	3	6	1	1,414214
20	3	7	2	1
1	4	8	1	1,414214
<b>Perimeter</b>				<b>23,72792</b>

Dari tabel diatas, hasil penghitungan keliling adalah  $11+9\sqrt{2}$  units atau 23,727 units

#### 2.5.4.3. Panjang, Lebar, dan Diameter

Diameter adalah jarak terpanjang antara dua titik dalam tepi objek. Hal tersebut dapat dihitung dengan menggunakan metode “*Brute Force*” seperti yang dilakukan Costa dan Cesar (2001). Yang pertama dilakukan ialah melakukan pencarian jarak antar titik-titik yang menjadi kontur sebuah objek. Dengan menggunakan penghitungan jarak *euclidean*, yang dapat dirumuskan dengan persamaan berikut

$$\text{jarak} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad 2.7$$

Dimana  $(x_1, y_1)$  adalah koordinat kontur-1 dan  $(x_2, y_2)$  adalah koordinat kontur-2. Penghitungan jarak akan diterapkan pada semua titik yang menjadi kontur dari sebuah objek yang merupakan kelilingnya. Setelah didapatkan jarak yang paling maksimum, maka dari kedua titik kontur tersebut ditarik sebuah garis lurus yang menyatakan panjang atau diameter objek tersebut. Setelah itu, lakukan penghitungan gradien dari garis panjang atau diameter tersebut dengan persamaan

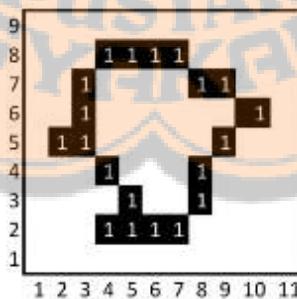
$$\text{GradienGarisPanjang} = \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad 2.8$$

Setelah gradien panjang atau diameter didapatkan, lakukan pencarian gradien garis kedua yang tegak lurus dengan garis tersebut. Gradien yang tegak lurus tersebut dapat dihitung dengan pesamaan

$$\text{GradienGarisLebar} = \frac{-1}{(\text{GradienGarisPanjang})} \quad 2.9$$

Setelah gradien garis lebar didapatkan, selanjutnya dilanjutkan pencarian titik yang memenuhi gradien tersebut, namun dengan jarak yang paling besar juga. Dalam praktik penggunaan algoritma *brute force* ini, perlu adanya toleransi pada garis yang memenuhi gradien garis lebar, dikarenakan akan sangat sulit memenuhi prasyarat gradien yang benar-benar tegak lurus terutama pada citra objek yang berukuran kecil.

Sebagai contoh sebuah objek dengan kontur  $(4,8), (5,8), (6,8), (7,8), (8,7), (9,7), (10,6), (9,5), (8,4), (8,3), (7,2), (6,2), (5,2), (4,2), (5,3), (4,4), (3,5), (2,5), (3,6), (3,7), (4,8)$



**Gambar 2.34 - .** Objek dengan kontur internal yang akan dicari panjang (diameter) dan lebar

Yang pertama dilakukan adalah mencari jarak *euclidean* antar titik penyusun kontur, dapat dilihat pada tabel 2.8.

**Tabel 2.8 - . Pengitungan Jarak Euclidean antar titik kontur objek**

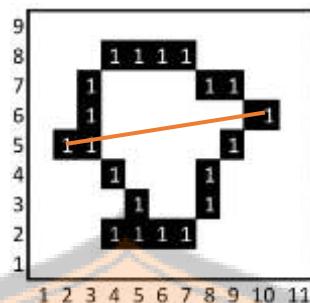
. Jarak			Titik	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	max
			x	4	5	6	7	8	9	10	9	8	8	7	6	5	4	3	4	3	2	3		
			y	8	8	8	8	7	7	6	5	4	3	2	2	2	2	3	4	5	5	6	7	
Titik	x	y																						
1	4	8		0,0	1,0	2,0	3,0	4,1	5,1	6,3	5,8	5,7	6,4	6,7	6,3	6,1	6,0	5,1	4,0	3,2	3,6	2,2	1,4	6,7
2	5	8		1,0	0,0	1,0	2,0	3,2	4,1	5,4	5,0	5,0	5,8	6,3	6,1	6,0	6,1	5,0	4,1	3,6	4,2	2,8	2,2	6,3
3	6	8		2,0	1,0	0,0	1,0	2,2	3,2	4,5	4,2	4,5	5,4	6,1	6,0	6,1	6,3	5,1	4,5	4,2	5,0	3,6	3,2	6,3
4	7	8		3,0	2,0	1,0	0,0	1,4	2,2	3,6	3,6	4,1	5,1	6,0	6,1	6,3	6,7	5,4	5,0	5,0	5,8	4,5	4,1	6,7
5	8	7		4,1	3,2	2,2	1,4	0,0	1,0	2,2	2,2	3,0	4,0	5,1	5,4	5,8	6,4	5,0	5,0	5,4	6,3	5,1	5,0	6,4
6	9	7		5,1	4,1	3,2	2,2	1,0	0,0	1,4	2,0	3,2	4,1	5,4	5,8	6,4	7,1	5,7	5,8	6,3	7,3	6,1	6,0	7,3
7	10	6		6,3	5,4	4,5	3,6	2,2	1,4	0,0	1,4	2,8	3,6	5,0	5,7	6,4	7,2	5,8	6,3	7,1	8,1	7,0	7,1	8,1
8	9	5		5,8	5,0	4,2	3,6	2,2	2,0	1,4	0,0	1,4	2,2	3,6	4,2	5,0	5,8	4,5	5,1	6,0	7,0	6,1	6,3	7,0
9	8	4		5,7	5,0	4,5	4,1	3,0	3,2	2,8	1,4	0,0	1,0	2,2	2,8	3,6	4,5	3,2	4,0	5,1	6,1	5,4	5,8	6,1
10	8	3		6,4	5,8	5,4	5,1	4,0	4,1	3,6	2,2	1,0	0,0	1,4	2,2	3,2	4,1	3,0	4,1	5,4	6,3	5,8	6,4	6,4
11	7	2		6,7	6,3	6,1	6,0	5,1	5,4	5,0	3,6	2,2	1,4	0,0	1,0	2,0	3,0	2,2	3,6	5,0	5,8	5,7	6,4	6,7
12	6	2		6,3	6,1	6,0	6,1	5,4	5,8	5,7	4,2	2,8	2,2	1,0	0,0	1,0	2,0	1,4	2,8	4,2	5,0	5,0	5,8	6,3
13	5	2		6,1	6,0	6,1	6,3	5,8	6,4	6,4	5,0	3,6	3,2	2,0	1,0	0,0	1,0	1,0	2,2	3,6	4,2	4,5	5,4	6,4
14	4	2		6,0	6,1	6,3	6,7	6,4	7,1	7,2	5,8	4,5	4,1	3,0	2,0	1,0	0,0	1,4	2,0	3,2	3,6	4,1	5,1	7,2
15	5	3		5,1	5,0	5,1	5,4	5,0	5,7	5,8	4,5	3,2	3,0	2,2	1,4	1,0	1,4	0,0	1,4	2,8	3,6	3,6	4,5	5,8
16	4	4		4,0	4,1	4,5	5,0	5,0	5,8	6,3	5,1	4,0	4,1	3,6	2,8	2,2	2,0	1,4	0,0	1,4	2,2	2,2	3,2	6,3
17	3	5		3,2	3,6	4,2	5,0	5,4	6,3	7,1	6,0	5,1	5,4	5,0	4,2	3,6	3,2	2,8	1,4	0,0	1,0	1,0	2,0	7,1
18	2	5		3,6	4,2	5,0	5,8	6,3	7,3	8,1	7,0	6,1	6,3	5,8	5,0	4,2	3,6	3,6	2,2	1,0	0,0	1,4	2,2	8,1
19	3	6		2,2	2,8	3,6	4,5	5,1	6,1	7,0	6,1	5,4	5,8	5,7	5,0	4,5	4,1	3,6	2,2	1,0	1,4	0,0	1,0	7,0
20	3	7		1,4	2,2	3,2	4,1	5,0	6,0	7,1	6,3	5,8	6,4	6,4	5,8	5,4	5,1	4,5	3,2	2,0	2,2	1,0	0,0	7,1
max				6,7	6,3	6,3	6,7	6,4	7,3	8,1	7,0	6,1	6,4	6,7	6,3	6,4	7,2	5,8	6,3	7,1	8,1	7,0	7,1	

Dari penghitungan jarak antar piksel penyusun kontur, jarak paling besar didapatkan dari titik (10,6) dan (2,5) dengan jarak 8,1. Dapat ditarik garis lurus dari kedua titik ini, garis inilah yang merupakan garis panjang (diameternya). Setelah itu cari gradien dari garis panjang ini dengan persamaan 2.8, maka didapatkan gradien garis panjang = 1/8. Selanjutnya dilakukan pencarian garis yang tegak lurus dengan persamaan 2.9, sehingga didapatkan gradien lebar adalah -8. Gradien garis antar titik penyusun kontur dapat dilihat pada tabel berikut.

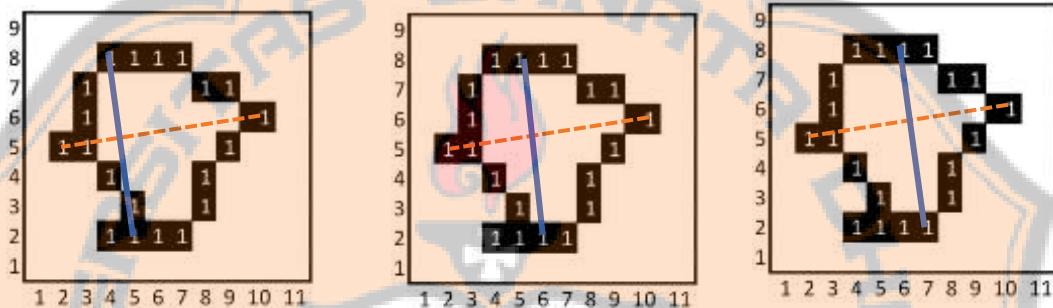
**Tabel 2.9 - . Pengitungan gradien garis yang melewati titik kontur objek**

gradien			Titik	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	MIN
	x	y		4	5	6	7	8	9	10	9	8	8	7	6	5	4	5	4	3	2	3	3	
	x	y		8	8	8	8	7	7	6	5	4	3	2	2	2	2	3	4	5	5	6	7	
1	4	8		INF	0,0	0,0	0,0	-0,3	-0,2	-0,3	-0,6	-1,0	-1,3	-2,0	-3,0	-6,0	INF	-5,0	INF	3,0	1,5	2,0	1,0	-6,0
2	5	8		0,0	INF	0,0	0,0	-0,3	-0,3	-0,4	-0,8	-1,3	-1,7	-3,0	-6,0	INF	6,0	INF	4,0	1,5	1,0	1,0	0,5	-6,0
3	6	8		0,0	0,0	INF	0,0	-0,5	-0,3	-0,5	-1,0	-2,0	-2,5	-6,0	INF	6,0	3,0	5,0	2,0	1,0	0,8	0,7	0,3	-6,0
4	7	8		0,0	0,0	0,0	INF	-1,0	-0,5	-0,7	-1,5	-4,0	-5,0	INF	6,0	3,0	2,0	2,5	1,3	0,8	0,6	0,5	0,3	-5,0
5	8	7		-0,3	-0,3	-0,5	-1,0	INF	0,0	-0,5	-2,0	INF	INF	5,0	2,5	1,7	1,3	1,3	0,8	0,4	0,3	0,2	0,0	-2,0
6	9	7		-0,2	-0,3	-0,3	-0,5	0,0	INF	-1,0	INF	3,0	4,0	2,5	1,7	1,3	1,0	1,0	0,6	0,3	0,3	0,2	0,0	-1,0
7	10	6		-0,3	-0,4	-0,5	-0,7	-0,5	-1,0	INF	1,0	1,0	1,5	1,3	1,0	0,8	0,7	0,6	0,3	0,1	0,1	0,0	-0,1	-1,0
8	9	5		-0,6	-0,8	-1,0	-1,5	-2,0	INF	1,0	INF	1,0	2,0	1,5	1,0	0,8	0,6	0,5	0,2	0,0	0,0	-0,2	-0,3	-2,0
9	8	4		-1,0	-1,3	-2,0	-4,0	INF	3,0	1,0	1,0	INF	INF	2,0	1,0	0,7	0,5	0,3	0,0	-0,2	-0,2	-0,4	-0,6	-4,0
10	8	3		-1,3	-1,7	-2,5	-5,0	INF	4,0	1,5	2,0	INF	INF	1,0	0,5	0,3	0,3	0,0	-0,3	-0,4	-0,3	-0,6	-0,8	-5,0
11	7	2		-2,0	-3,0	-6,0	INF	5,0	2,5	1,3	1,5	2,0	1,0	INF	0,0	0,0	0,0	-0,5	-0,7	-0,8	-0,6	-1,0	-1,3	-6,0
12	6	2		-3,0	-6,0	INF	6,0	2,5	1,7	1,0	1,0	1,0	0,5	0,0	INF	0,0	0,0	-1,0	-1,0	-1,0	-0,8	-1,3	-1,7	-6,0
13	5	2		-6,0	INF	6,0	3,0	1,7	1,3	0,8	0,8	0,7	0,3	0,0	0,0	INF	0,0	INF	-2,0	-1,5	-1,0	-2,0	-2,5	-6,0
14	4	2		INF	6,0	3,0	2,0	1,3	1,0	0,7	0,6	0,5	0,3	0,0	0,0	0,0	INF	1,0	INF	-3,0	-1,5	-4,0	-5,0	-5,0
15	5	3		-5,0	INF	5,0	2,5	1,3	1,0	0,6	0,5	0,3	0,0	-0,5	-1,0	INF	1,0	INF	-1,0	-1,0	-0,7	-1,5	-2,0	-5,0
16	4	4		INF	4,0	2,0	1,3	0,8	0,6	0,3	0,2	0,0	-0,3	-0,7	-1,0	-2,0	INF	-1,0	INF	-1,0	-0,5	-2,0	-3,0	-3,0
17	3	5		3,0	1,5	1,0	0,8	0,4	0,3	0,1	0,0	-0,2	-0,4	-0,8	-1,0	-1,5	-3,0	-1,0	-1,0	INF	0,0	INF	INF	-3,0
18	2	5		1,5	1,0	0,8	0,6	0,3	0,3	0,1	0,0	-0,2	-0,3	-0,6	-0,8	-1,0	-1,5	-0,7	-0,5	0,0	INF	1,0	2,0	-1,5
19	3	6		2,0	1,0	0,7	0,5	0,2	0,2	0,0	-0,2	-0,4	-0,6	-1,0	-1,3	-2,0	-4,0	-1,5	-2,0	INF	1,0	INF	INF	-4,0
20	3	7		1,0	0,5	0,3	0,3	0,0	0,0	-0,1	-0,3	-0,6	-0,8	-1,3	-1,7	-2,5	-5,0	-2,0	-3,0	INF	2,0	INF	INF	-5,0
MIN				-6,0	-6,0	-6,0	-5,0	-2,0	-1,0	-1,0	-2,0	-4,0	-5,0	-6,0	-6,0	-5,0	-5,0	-3,0	-3,0	-1,5	-4,0	-5,0		

Pada contoh ini, tidak ditemukan gradien garis tepat -8, namun paling mendekati adalah -6, yang dimiliki oleh pasangan titik (4,8)-(5,2), (5,8)-(6,2), dan (6,8)-(7,2). Ketiga pasangan titik ini memiliki jarak 6,3.



**Gambar 2.35 - .** Garis Panjang (Diameter) objek dengan gradien  $1/8$ , pada titik  $(10,6)$ ,  $(2,5)$  dengan jarak  $8,1$



**Gambar 2.36 - .** Garis Lebar objek dengan gradien  $-6$ , pada titik  $(4,8)-(5,2)$ ,  $(5,8)-(6,2)$ , dan  $(6,8)-(7,2)$  dengan jarak  $6,3$

Perlu adanya toleransi dalam penggunaan metode ini, dimana pada contoh ini hanya digunakan gradien yang mendekati  $-8$  yakni garis dengan gradien  $-6$ . Dengan demikian, panjang (diameter) dari objek tersebut adalah **8,1** dan lebarnya adalah **6,3**

### 2.5.5. Ciri dari Penghitungan Perimeter, Luas, Panjang dan Lebar

Perimeter, luas, panjang (diameter), dan lebar tidak dapat digunakan secara mandiri sebagai sebuah deskriptor, dikarenakan ciri tersebut sangat berpengaruh terhadap ukuran objek. Agar tidak terpengaruh oleh penyekalaan ukuran, digunakan beberapa fitur turunan dari perimeter, luas, panjang, lebar tersebut

#### 2.5.5.1. Rasio Kebulatan

Kebundaran (*circularity*) bentuk adalah perbandingan antara luas objek dan kuadrat perimeternya. Rasio kebulatan dapat dinyatakan dengan

$$kebulatan(R) = 4\pi \frac{Luas(R)}{Perimeter^2(R)} \quad 2.10$$

Dari persamaan diatas, semakin besar rasio kebulatan, maka objek R mendekati lingkaran sempurna.

Sebagai contoh sebuah citra objek dengan kontur (4,8), (5,8), (6,8), (7,8), (8,7), (9,7), (10,6), (9,5), (8,4), (8,3), (7,2), (6,2), (5,2), (4,2), (5,3), (4,4), (3,5), (2,5), (3,6), (3,7), (4,8) dengan kode 00070755654441334121 akan dilakukan penghitungan rasio kebulatan. Yang pertama dilakukan pencarian luas dan perimeternya. Hal ini dapat dilakukan sesuai dengan subbab sebelumnya, sehingga didapatkan Perimeter = 23.727 units dan Luas = 30 units.

$$\begin{aligned} kebulatan(R) &= 4\pi \frac{30}{(23.727)^2} \\ kebulatan(R) &= 0.6695 \end{aligned}$$

#### 2.5.5.2. Rasio Kerampingan

Kerampingan bentuk adalah perbandingan antara lebar dan panjang. Fitur ini terkadang disebut rasio aspek. Fitur ini digunakan dalam membedakan objek yang ‘gemuk’ dan objek ‘kurus’. Rasio kerampingan dapat dihitung dengan persamaan

$$kerampingan(R) = \frac{Lebar(R)}{Panjang(R)} \quad 2.11$$

Sebagai contoh sebuah citra objek dengan kontur (4,8), (5,8), (6,8), (7,8), (8,7), (9,7), (10,6), (9,5), (8,4), (8,3), (7,2), (6,2), (5,2), (4,2), (5,3), (4,4), (3,5), (2,5), (3,6), (3,7), (4,8) dengan kode 00070755654441334121 akan dilakukan penghitungan rasio kerampingan. Yang pertama dilakukan pencarian panjang dan lebarnya. Hal ini dapat dilakukan sesuai dengan subbab sebelumnya, sehingga didapatkan Panjang = 8.1 units, dan Lebar = 6.3 units

$$kerampingan(R) = \frac{6.3}{8.1}$$

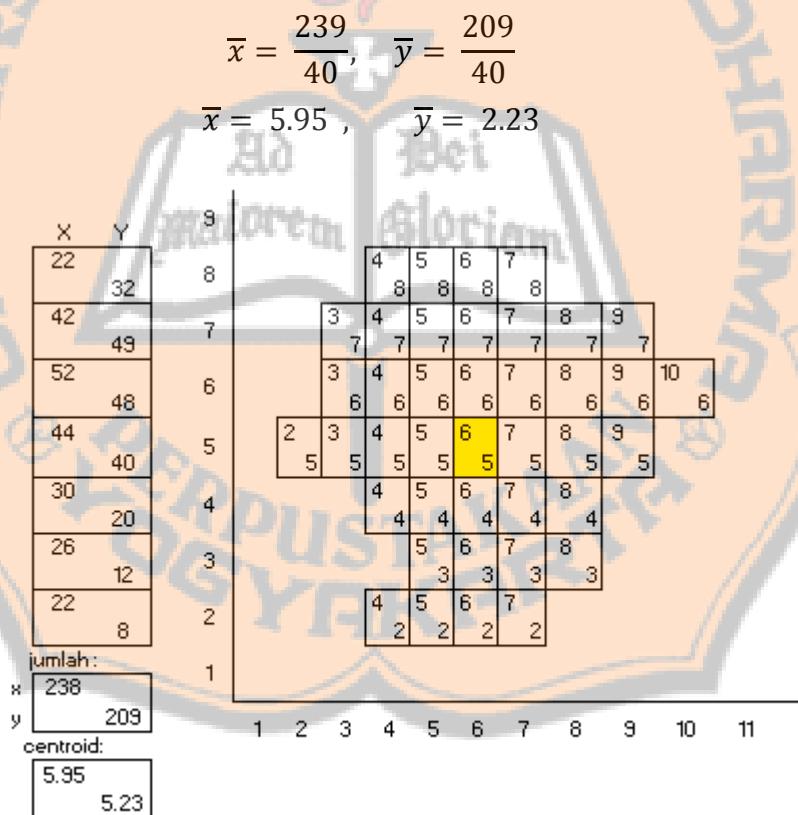
$$kerampingan(R) = 0.754473836 \quad 2.11$$

### 2.5.6. Pusat Massa Objek (Centroid)

Pusat massa atau centroid dapat dicari dengan menggunakan nilai rerata koordinat setiap piksel penyusun objek. Pencarian centroid dapat dinyatakan dengan persamaan sebagai berikut

$$\bar{x} = \frac{\sum x}{luas(R)}, \bar{y} = \frac{\sum y}{luas(R)} \quad 2.12$$

Sebagai contoh sebuah citra objek dengan kontur (4,8), (5,8), (6,8), (7,8), (8,7), (9,7), (10,6), (9,5), (8,4), (8,3), (7,2), (6,2), (5,2), (4,2), (5,3), (4,4), (3,5), (2,5), (3,6), (3,7), (4,8) akan dilakukan pencarian titik pusat massa objeknya. Yang pertama dilakukan adalah melakukan penjumlahan semua nilai x dan y dari objek tersebut. Dari penghitungan, didapatkan jumlah  $x = 239$  dan  $y = 209$  dari luas objek 40 piksel. Sehingga dapat dihitung



**Gambar 2.37 – Visualisasi penghitungan pusat masa objek, dengan hasil (5.95, 5.23)**

## 2.5.7. Ciri dari Penghitungan Centroid, dan Kontur

### 2.5.7.1. Dispersi

Untuk menyatakan suatu ketidakteraturan atau ketidakkompakan bentuk sebuah objek, Nixon dan Aguado (2002) menyarankan penggunaan sebuah fitur dispersi. Penggunaan fitur kekompakan (kebulatan) untuk semua objek dianggap sebagai deskriminasi yang kurang tepat.



**Gambar 2.38 -** Tiga objek dengan kekompakan berbeda. Lingkaran (**kiri**) dan Oval (**tengah**) teratur, dan objek random (**kanan**) tidak teratur (Kadir, 2013)

Berdasarkan definisi Chen (1995) dispersi diukur sebagai perbandingan panjang *chord* utama terhadap area objek. Dapat dinyatakan dalam persamaan

$$I(R) = \pi \max \frac{(\sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2})}{\text{Luas}(R)} \quad 2.13$$

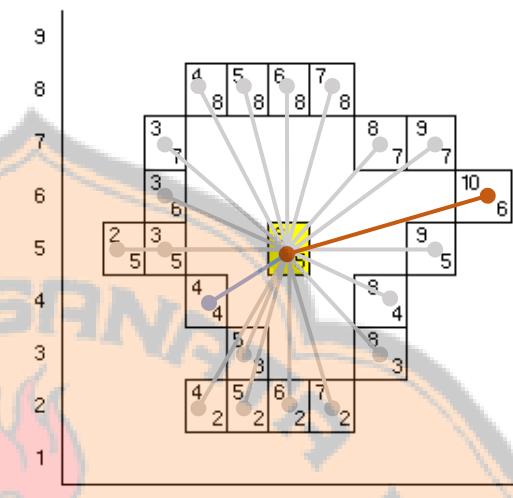
Dimana  $\bar{x}$  dan  $\bar{y}$  merupakan pusat massa area dari R dan Luas(R) menyatakan luas dari objek R. Alternatif lain dalam mencari fitur dispersi dengan menggunakan rasio radius maksimum terhadap radius minimum yang dinyatakan dalam persamaan berikut

$$IR(R) = \frac{\max(\sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2})}{\min(\sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2})} \quad 2.14$$

Sebagai contoh sebuah citra objek dengan kontur (4,8), (5,8), (6,8), (7,8), (8,7), (9,7), (10,6), (9,5), (8,4), (8,3), (7,2), (6,2), (5,2), (4,2), (5,3), (4,4), (3,5), (2,5), (3,6), (3,7), (4,8) akan dilakukan penhitungan fitur dispersinya. Yang pertama dilakukan adalah melakukan pencarian pusat massa objek seperti yang sudah dilakukan pada subbab sebelumnya. Sehingga didapatkan  $\bar{x} = 5.95$ ,  $\bar{y} = 2.23$ . Berikutnya lakukan penghitungan jarak antar piksel penyusun kontur dengan koordinat pusat massa yang sudah didapatkan, dapat dilihat pada tabel berikut.

**Tabel 2.10** – Tabel jarak antara titik penyusun kontur objek dan titik pusat massa

Titik	Jarak		$\bar{x}$	5.95
	x	y	$\bar{y}$	5.225
1	4	8		3.4
2	5	8		2.9
3	6	8		2.8
4	7	8		3.0
5	8	7		2.7
6	9	7		3.5
7	10	6		4.1
8	9	5		3.1
9	8	4		2.4
10	8	3		3.0
11	7	2		3.4
12	6	2		3.2
13	5	2		3.4
14	4	2		3.8
15	5	3		2.4
16	4	4		2.3
17	3	5		3.0
18	2	5		4.0
19	3	6		3.1
20	3	7		3.4
max				4.1
min				2.3



**Gambar 2.39** – Visualisasi penghitungan jarak sentroid dengan kontur, Sentroid-(10,6) merupakan jarak terpanjang dan Sentroid-(4,4) merupakan jarak terpendek

Setelah itu, fitur dipersi dilakukan dengan menggunakan jarak maksimum dan minimum diatas. Untuk dispersi perbandingan *chord* utama dengan luasnya (Chen), maka dapat digunakan luas objek tersebut, seperti pada perhitungan pada subbab sebelumnya

$$I(R) = \pi \frac{4.1}{30}$$

$$I(R) = 0.429133$$

Sedangkan pada dispersi perbandingan rasio maksimum dan minimum dapat dihitung dengan

$$IR(R) = \frac{4.1}{2.3}$$

$$IR(R) = 1.78$$

## 2.6. Jaringan Syaraf Tiruan

Jaringan saraf tiruan adalah sistem pemroses informasi yang dikembangkan dengan model matematika dan memiliki karakteristik mirip dengan jaringan saraf biologi dengan asumsi bahwa (Siang, 2005):

1. Pemrosesan informasi terjadi pada banyak elemen sederhana (*neuron*).
2. Sinyal dikirim diantara *neuron-neuron* melalui penghubung-penghubung.
3. Penghubung antar *neuron* memiliki bobot yang akan memperkuat atau memperlemah sinyal

Untuk menentukan luaran, setiap *neuron* menggunakan fungsi aktivasi yang dikenakan pada jumlah masukan yang diterima. Besarnya keluaran selanjutnya dibandingkan dengan suatu batas ambang.

Sehingga jaringan saraf tiruan ditentukan oleh 3 hal, yaitu Pola hubungan antar *neuron* (arsitektur jaringan), Metode penentuan bobot, dan Fungsi Aktivasi.

### 2.6.1. Arsitektur Jaringan Saraf Tiruan

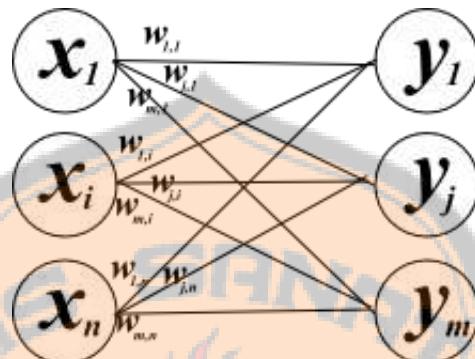
Arsitektur jaringan saraf tiruan merupakan pengaturan spesifik dan hubungan antar *neuron* pada lapisan yang dalam jaringan *neuron*. *Neuron* yang terletak pada lapisan yang sama memiliki sifat dan fungsi aktivasi yang sama.

Neuron dalam jaringan saraf tiruan terdiri dari lapisan masukan, lapisan tersembunyi, dan lapisan keluaran. Lapisan masukan berfungsi menerima sinyal dari luar jaringan. Lapisan tersembunyi (*hidden layer*) berfungsi menerima sinyal dari masukan. Lapisan keluaran berfungsi mengekstrak ciri dari hal yang berkaitan dengan pola atau sinyal yang diterima.

Beberapa arsitektur jaringan saraf tiruan yang sering digunakan:

### 2.6.1.1. Jaringan Layar Tunggal

Dalam arsitektur layar tunggal, sekumpulan *neuron* masukan langsung dihubungkan dengan layer keluaran

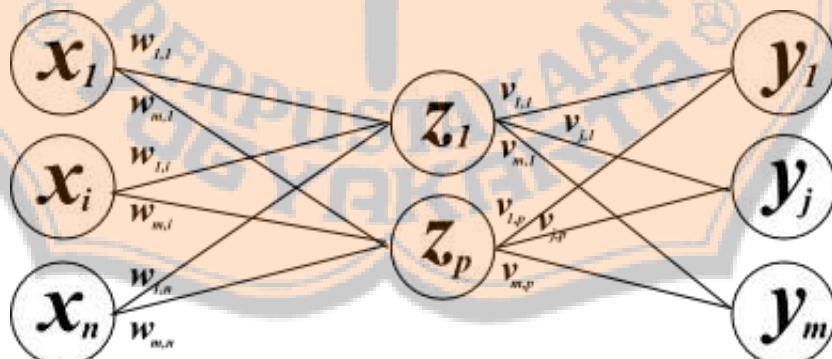


Gambar 2.40 – Jaringan layar tunggal (Siang, 2005)

Gambar 2.40 menunjukkan arsitektur jaringan dengan n masukan dan m keluaran.  $X_1, X_2, \dots, X_n$  adalah *neuron* masukan.  $Y_1, Y_2, \dots, Y_m$  adalah *neuron* keluaran dari jaringan saraf tiruan.  $w_{ij}$  menyatakan bobot hubungan antara unit ke-i dalam masukan dengan unit ke-j dalam keluaran. Bobot bernilai independen. Selama proses pelatihan, bobot akan dimodifikasi untuk meningkatkan keakuratan

### 2.6.1.2. Jaringan Layar Jamak

Dalam arsitektur layar jamak selain terdiri dari masukan dan keluaran, juga terdapat layer tersembunyi.



Gambar 2.41 – Jaringan layar jamak (Siang, 2005)

Gambar 2.41 menunjukkan arsitektur jaringan dengan  $X_1, X_2, \dots, X_n$ , adalah *neuron* masukan,  $Z_1, \dots, Z_p$  adalah *neuron* pada *layer* tersembunyi, dan  $Y_1, Y_2, \dots, Y_m$  adalah *neuron* keluaran.  $w_{np}$  adalah bobot hubungan *layer input* dan *hidden layer* dan  $v_{nm}$  menyatakan bobot hubungan *hidden layer* dan *layer output*.

### 2.6.2. Fungsi Aktivasi

Fungsi aktivasi digunakan untuk menentukan keluaran suatu *neuron*. Dengan argumen bahwa fungsi aktivasi adalah net masukannya, yaitu kombinasi linier masukan dan bobotnya, sehingga dapat dituliskan:

$$net = \sum X_i W_i \quad 2.15$$

$$f(net) = f(\sum X_i W_i)$$

Fungsi aktivasi dalam jaringan saraf tiruan terdiri dari:

#### Fungsi *threshold*

$$f(x) = \begin{cases} 1 & \text{jika } x \geq a \\ 0 & \text{jika } x < a \end{cases} \quad 2.16$$

Jika fungsi threshold bernilai -1 atau 1, dapat disebut threshold bipolar.

#### Fungsi *sigmoid*

$$f(x) = \frac{1}{1 + e^{-x}} \quad 2.17$$

Nilai fungsi sigmoid terletak antara 0 dan 1, dan dapat diturunkan menjadi

$$f(x)' = f(x)(1 - f(x)) \quad 2.18$$

#### Fungsi identitas

$$f(x) = x \quad 2.19$$

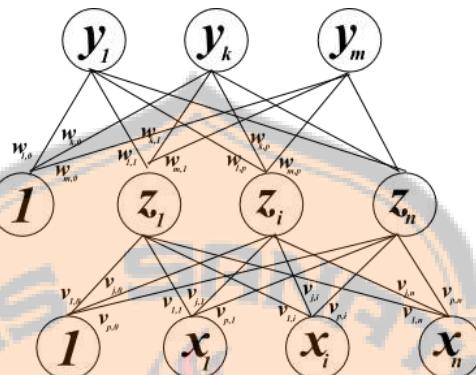
Fungsi ini digunakan jika keluaran jaringan berupa bilangan riil, bukan hanya range 0 dan 1 atau -1 dan 1

### 2.6.3. Jaringan Saraf Tiruan *Backpropagation*

*Backpropagation* merupakan salah satu jaringan layar jamak. *Backpropagation* digunakan untuk melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan, dan disertai dengan kemampuan jaringan untuk memberikan respon yang benar terhadap pola masukan yang serupa dengan pola yang dipakai selama pelatihan

### 2.6.3.1. Arsitektur *Backpropagation*

Arsitektur *Backpropagation* terdiri dari beberapa unit neuron dalam satu atau lebih hidden layer.



Gambar 2.42 – Arsitektur *Backpropagation* (Siang, 2005)

Gambar 2.42 merupakan arsitektur *backpropagation* dengan  $n$  buah masukan dengan bias dan  $m$  buah keluaran.  $v_{ji}$  merupakan bobot dari unit masukan  $X_i$  ke *neuron hidden layer*  $Z_j$  dengan  $V_{j0}$  merupakan bobot yang menghubungkan bias di unit masukan ke *neuron* tersembunyi.  $W_{kj}$  merupakan bobot dari *neuron* tersembunyi dari  $Z_j$  ke *output*  $Y_k$  ( $W_{k0}$ ) merupakan bias di *hidden layer* ke unit keluaran  $Z_k$ .

### 2.6.3.2. Fungsi Aktivasi *Backpropagation*

Fungsi aktivasi pada *backpropagation* tidak hanya menggunakan fungsi aktivasinya, namun juga fungsi turunan dari fungsi tersebut. *Backpropagation* dapat menggunakan fungsi aktivasi *sigmoid biner* (0 s.d 1) maupun *sigmoid bipolar* (-1 s.d 1) beserta turunan fungsinya. Pemilihan fungsi aktivasi bergantung pada kebutuhan nilai keluaran jaringan yang diharapkan. Bila keluaran jaringan terdapat nilai negatif, maka menggunakan *sigmoid bipolar*. Namun jika keluaran yang dicari bernilai positif atau 0, maka digunakan *sigmoid biner*. Selain *sigmoid*, fungsi linear juga sering digunakan dalam *backpropagation*.

2.6.3.2.1. Fungsi *sigmoid* biner

$$f(x) = \frac{1}{1 + e^{-x}} \quad 2.20$$

2.6.3.2.2. Turunan *sigmoid* biner

$$f'(x) = f(x)(1 - f(x)) \quad 2.21$$

2.6.3.2.3. Fungsi *sigmoid* bipolar

$$f(x) = \frac{1}{1 + e^{-x}} - 1 \quad 2.22$$

2.6.3.2.4. Turunan *sigmoid* bipolar

$$f'(x) = \frac{(1 + f(x))(1 - f(x))}{2} \quad 2.23$$

Fungsi *sigmoid* memiliki nilai maksimum 1. Jika pola dengan target lebih dari 1, maka keluaran harus ditransformasikan sehingga memiliki *range* yang sama dengan fungsi *sigmoid* yang digunakan. Alternatif lain adalah penggunaan fungsi *sigmoid* hanya pada *layer hidden* dan masukan. Pada *layer output* digunakan fungsi identitas:

$$f(x) = x \quad 2.24$$

## 2.6.4. Pelatihan Standar Backpropagation

Pelatihan ini terdiri dari 3 fase. Fase maju adalah dimana pola masukan dihitung maju mulai dari layar masukan hingga *layer output* menggunakan fungsi aktivasi yang ditentukan. Fase mundur merupakan kegiatan propagasi mundur dari selisih dari keluaran jaringan dengan target yang diharapkan. Fase ini dimulai dari *layer output*. Fase yang terakhir adalah modifikasi bobot, untuk menurunkan kesalahan yang terjadi.

**Algoritma 2.1** - Pelatihan jaringan saraf tiruan *backpropagation***Langkah 0** : Inisialisasi semua bobot dengan bilangan acak kecil**Langkah 1** : Looping, lakukan langkah 2 s.d 9**Langkah 2** : Untuk setiap pasang data latih, lakukan 3 s.d 8**FASE I : PROPAGASI MAJU****Langkah 3** : Tiap unit masukan menerima sinyal dan meneruskan ke *hidden layer***Langkah 4** : Hitung semua output di *hidden layer*  $z_j$  ( $j = 1, 2, \dots, p$ )

$$z_{net_j} = v_{j,0} + \sum_{i=1}^n (x_i v_{j,i}) \quad 2.25$$

$$z_j = f(z_{net_j}) = \frac{1}{1 + e^{-z_{net_j}}} \quad 2.26$$

**Langkah 5** : Hitung semua keluaran jaringan di unit  $y_k$  ( $k = 1, 2, \dots, m$ )

$$y_{net_k} = w_{k,0} + \sum_{j=1}^p (z_j w_{k,j}) \quad 2.27$$

$$y_k = f(y_{net_k}) = \frac{1}{1 + e^{-y_{net_k}}} \quad 2.28$$

**FASE II : PROPAGASI MUNDUR****Langkah 6** : Hitung faktor  $\delta$  output berdasarkan kesalahan setiap output  $y_k$  ( $k = 1, 2, \dots, m$ )

$$\delta_k = (t_k - y_k)f'(y_{net_k}) = (t_k - y_k)y_k(1 - y_k) \quad 2.29$$

 $\delta$  merupakan nilai kesalahan yang akan dipakai dalam perubahan bobot layer selanjutnya. Hitung perubahan bobot  $w_{kj}$  dengan percepatan  $\alpha$ 

$$\Delta W_{kj} = \alpha \delta_k z_j \quad 2.30$$

**Langkah 7** : Hitung faktor  $\delta$  *hidden layer* berdasarkan kesalahan disetiap layer  $z_j$  ( $j = 1, 2, \dots, p$ )

$$\delta_{net_j} = \sum_{k=1}^m (\delta_k w_{kj}) \quad 2.31$$

Faktor  $\delta$  di hidden layer

$$\delta_j = \delta_{net_j} f'(z_{net_j}) = \delta_{net_j} z_j (1 - z_j) \quad 2.32$$

Hitung perubahan bobot vij

$$\Delta v_{ji} = \alpha \delta_j x_i \quad 2.33$$

### FASE III : PERUBAHAN BOBOT

**Langkah 8 :** Hitung semua perubahan bobotnya

Perubahan bobot garis ke output layer

$$w_{kj}(baru) = w_{kj}(lama) + \Delta W_{kj} \quad 2.34$$

Perubahan bobot garis ke hidden layer

$$v_{ji}(baru) = v_{ji}(lama) + \Delta V_{ji} \quad 2.35$$

**Langkah 9 :**

Pelatihan selesai dilakukan, jaringan dapat dipakai untuk pengenalan pola.

## BAB III

### METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

Pada bab III ini, akan dijelaskan secara umum metode penelitian ini, dengan tujuan melakukan identifikasi bentuk biji kopi menggunakan deskriptor bentuk dasar dan jaringan saraf tiruan propagasi balik pada citra biji kopi. Setelah itu dilanjutkan dengan perancangan sistem sebagai alat uji.

#### 3.1. Metodologi Penelitian

##### 3.1.1. Gambaran Umum

Penelitian ini bertujuan untuk melakukan klasifikasi bentuk biji kopi dengan menggunakan deskriptor bentuk dasar dan jaringan saraf tiruan. Adapun biji kopi akan diklasifikasi menjadi 2 kelas yakni biji kopi utuh dan biji kopi pecah. Data yang digunakan dalam penelitian ini adalah citra biji kopi dari kedua kelas tersebut. Data yang diperoleh dalam penelitian kemudian dilakukan proses simulasi menggunakan alat uji yang dibuat oleh peneliti.

Tahapan yang dilakukan oleh peneliti adalah tahapan studi literatur, tahapan pengambilan data, tahapan perancangan alat uji, tahapan implementasi alat uji dan analisis hasil. Alat uji yang dibuat dirancang akan menggunakan citra biji kopi sebagai masukan, dan keluaran berupa hasil identifikasi dari citra biji kopi tersebut.

##### 3.1.2. Desain Penelitian

###### 3.1.2.1. Studi Literatur

Tahap awal penelitian ini adalah studi literatur. Berikut merupakan beberapa daftar literatur yang digunakan dalam mendukung penelitian ini.

1. Sofi'i, Imam. 2005. *Pemutuan Biji Kopi dengan Pengolahan Citra Digital dan Artificial Neural Network*. Institut Pertanian Bogor:Bogor
2. Madi, Sri Citra Yuliana. 2010. *Pemutuan Biji Kopi dengan Menggunakan Pengolahan Citra (Image Processing)*. Institut Pertanian Bogor:Bogor
3. Faridah, Parikesit, Gea O.F dan Ferdiansjah. 2011. *TELKOMNIKA Vol 9, no*

4. 3 : *Coffee Bean Grade Determination Based on Image Parameter.*  
Direktorat Pendidikan Tinggi : Jakarta
5. Ayitenfsu, Betelihem M. 2014. *Method of Coffee Bean Defect Detection.*  
International Journal of Engineering Research & Technology (IJERT)

### **3.1.2.2. Data Penelitian**

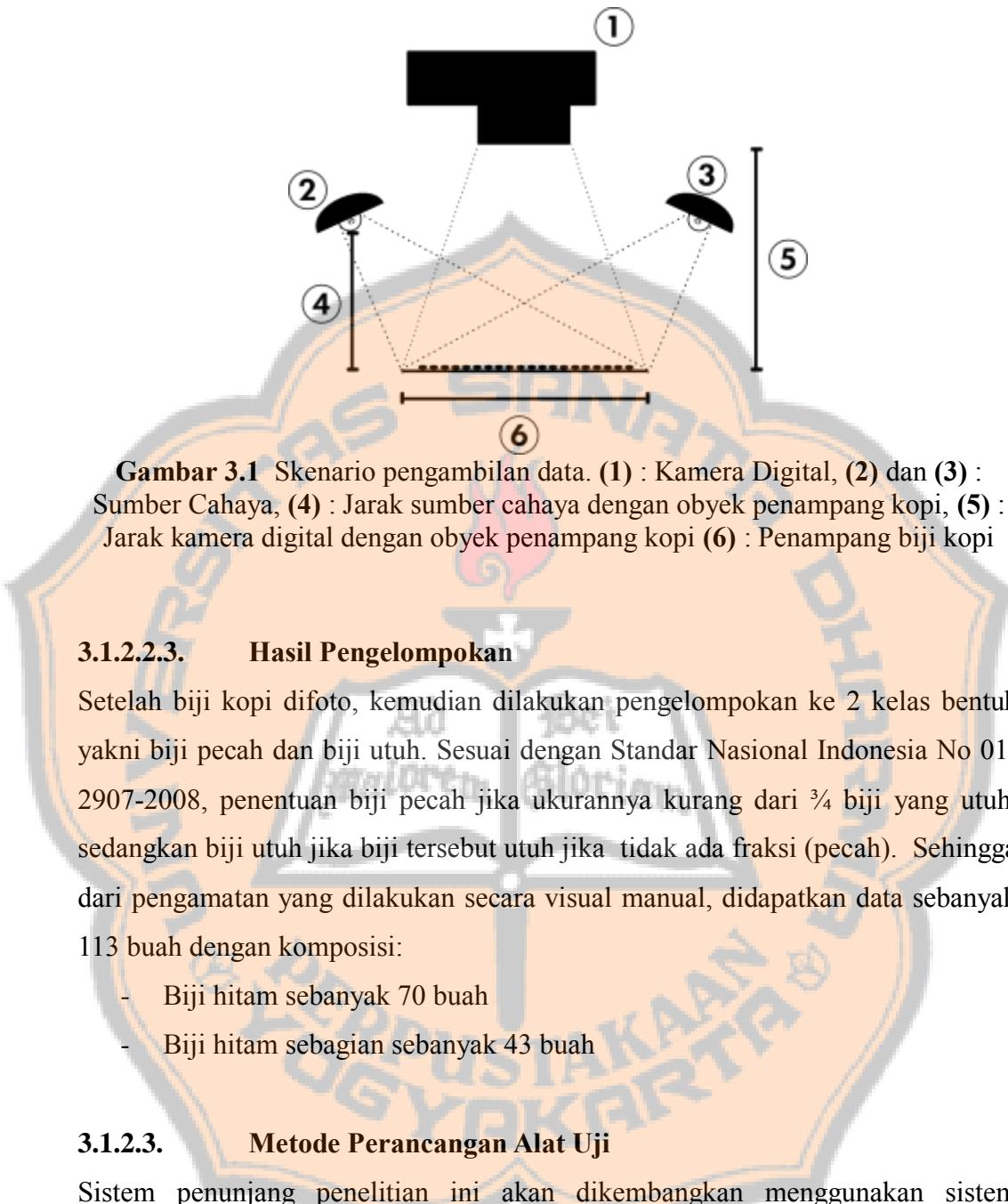
#### **3.1.2.2.1. Kopi**

Biji kopi yang digunakan adalah 1 sampel kopi robusta Menoreh, Kulon Progo seberat 300 gr dan 1 sampel kopi arabica Temanggung dengan berat total 300 gr. Kedua sampel merupakan hasil pengolahan kering. Kedua sampel tersebut kemudian dipisahkan sesuai dengan bagian-bagiannya

#### **3.1.2.2.2. Skenario Pengambilan Data**

Metode pengambilan data citra dari kopi hasil sortasi pada poin (3.1.2.2.1) adalah sebagai berikut:

1. Pengambilan citra sampel biji kopi menggunakan kamera digital Nikon D3300 dengan kecepatan lensa 1/40 dan ISO 100.
2. Penampang latar belakang citra kopi warna putih
3. Biji kopi disebar pada penampang dengan bantuan kasa ukuran 1 x 1 cm
4. Objek biji kopi diambil dengan jarak 45 cm dari lensa kamera.
5. Citra yang digunakan diambil dengan luas penampang 21 x 29.7 cm atau 6000x4000 piksel.
6. Kondisi pencahayaan  $414.5 \pm 2.9$  lux\*
7. Citra disimpan dalam file berekstensi \*.jpeg



### 3.1.2.2.3. Hasil Pengelompokan

Setelah biji kopi difoto, kemudian dilakukan pengelompokan ke 2 kelas bentuk yakni biji pecah dan biji utuh. Sesuai dengan Standar Nasional Indonesia No 01-2907-2008, penentuan biji pecah jika ukurannya kurang dari  $\frac{3}{4}$  biji yang utuh, sedangkan biji utuh jika biji tersebut utuh jika tidak ada fraksi (pecah). Sehingga dari pengamatan yang dilakukan secara visual manual, didapatkan data sebanyak 113 buah dengan komposisi:

- Biji hitam sebanyak 70 buah
- Biji hitam sebagian sebanyak 43 buah

### 3.1.2.3. Metode Perancangan Alat Uji

Sistem penunjang penelitian ini akan dikembangkan menggunakan sistem prototyping. Dimana diawali dengan pengumpulan bahan-bahan penunjang pengembangan sistem, dilanjutkan dengan pembuatan desain. Setelah desain didapatkan, maka dilanjutkan dengan pembuatan prototipe sistem, dan dilakukan evaluasi setelahnya. Setelah evaluasi dilakukan, maka selanjutnya penyempurnaan prototipe dan diakhiri dengan rekayasa perangkat lunak dan implementasi

### 3.1.3. Spesifikasi Perangkat Penelitian

#### 3.1.3.1. Perangkat Keras

Spesifikasi perangkat keras (*hardware*) yang digunakan dalam penelitian ini adalah sebagai berikut

1. *Cropping* citra biji kopi
  1. CPU : Intel Core i5 M 430 2.27 GHz x 4
  2. RAM : 4 GB DDR 3
  3. HDD : 500 GB
  4. *Graphic* : Nvidia GeForce 310M CUDA 512 MB
  5. Monitor : LCD 13.6" 1366 x 768
2. *Preprocessing* citra, ekstraksi ciri, dan perancangan dan penggunaan jaringan saraf tiruan
  1. CPU : Intel Core 2 Duo T5800 2.00Ghz x 2
  2. RAM : 2 GB DDR 3
  3. HDD : 320 GB
  4. *Graphic* : ATI Mobility Radeon HD 3400
  5. Monitor : LCD 13.6" 1366 x 768

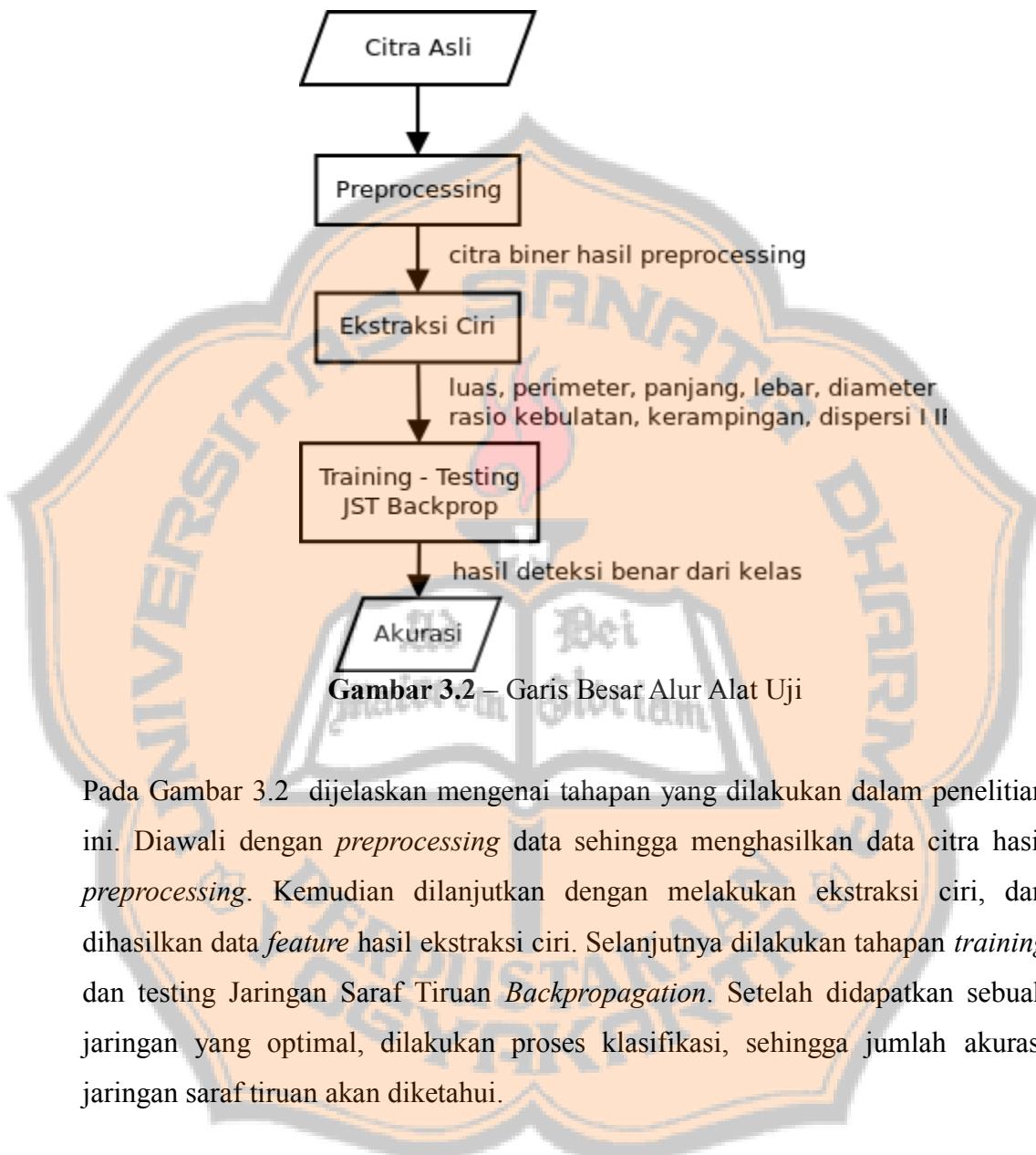
#### 3.1.3.2. Perangkat Lunak

Spesifikasi perangkat lunak (*software*) yang digunakan dalam penelitian ini adalah sebagai berikut

1. *Operating System* :
  1. Ubuntu 16.04 (Xenial Xerus) 64 bit
  2. Windows 7 Ultimate 32 bit
2. Software Pendukung :
  1. GIMP 2.8.16 (*cropping* citra)
  2. Matlab R2012b (jaringan saraf tiruan)

### 3.2. Perancangan Sistem Alat Uji

#### 3.2.1. Alur Penggunaan Alat Uji

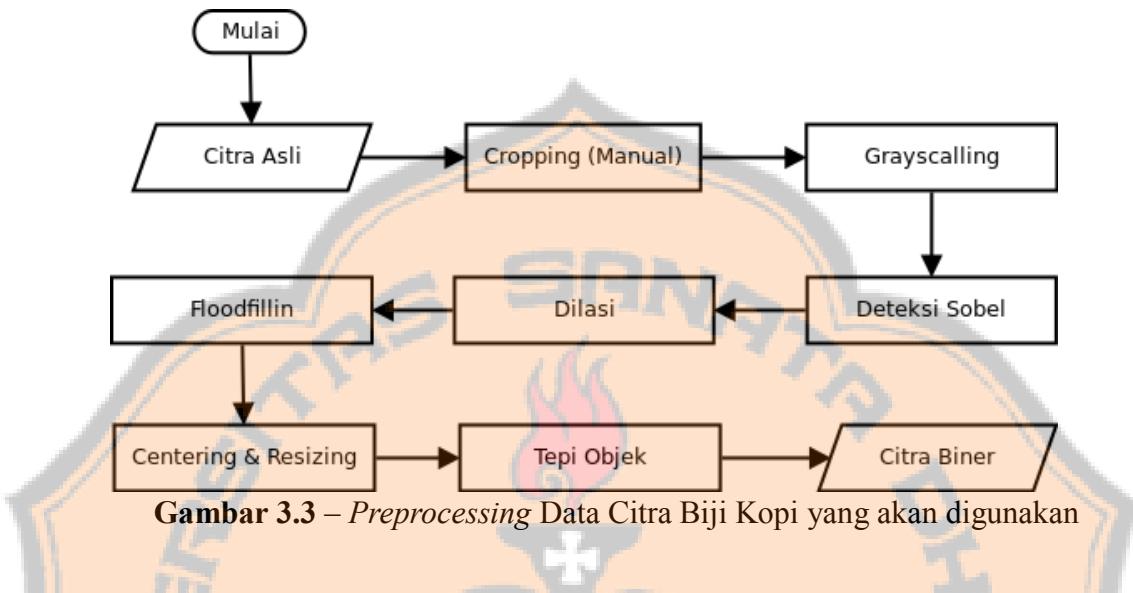


Pada Gambar 3.2 dijelaskan mengenai tahapan yang dilakukan dalam penelitian ini. Diawali dengan *preprocessing* data sehingga menghasilkan data citra hasil *preprocessing*. Kemudian dilanjutkan dengan melakukan ekstraksi ciri, dan dihasilkan data *feature* hasil ekstraksi ciri. Selanjutnya dilakukan tahapan *training* dan *testing* Jaringan Saraf Tiruan *Backpropagation*. Setelah didapatkan sebuah jaringan yang optimal, dilakukan proses klasifikasi, sehingga jumlah akurasi jaringan saraf tiruan akan diketahui.

##### 3.2.1.1. Preprocessing Data

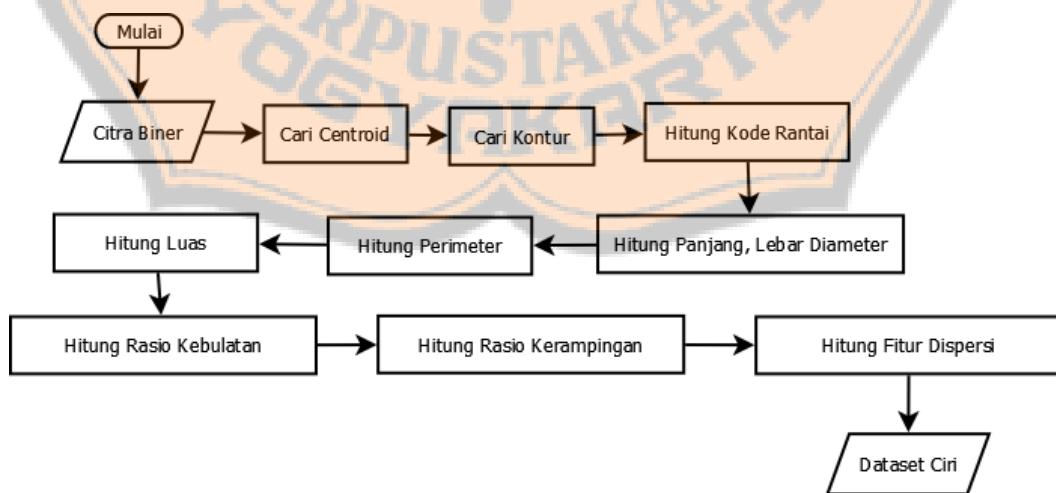
Dalam penelitian ini, *preprocessing* yang dilakukan dimulai dengan *cropping* data citra hasil kamera secara manual. Data yang dihasilkan dari proses *cropping* berupa citra dengan ukuran 256x256 piksel. Kemudian data dimasukkan dalam sistem *preprocessing*, dimana tiap file citra akan mengalami proses *grayscale*, deteksi tepi sobel, dilasi, *flood-fill*, *centering-resizing*, dan operasi tepi objek. Hasil

keluaran dari *preprocessing* data ini berupa citra biner yang akan digunakan dalam proses selanjutnya yakni ekstraksi ciri. Penjelasan perancangan *preprocessing* data citra biji kopi dapat dilihat di subbab 3.2.3



### 3.2.1.2. Ekstraksi Ciri

Pada proses ekstraksi ciri dari citra biji kopi hasil *preprocessing*, yang dilakukan adalah binerisasi, dilanjutkan dengan penggunaan beberapa algoritma untuk penghitungan ciri deskriptor dasar yang akan digunakan. Diawali dengan penentuan pusat massa, kontur dan kode rantai, dan dari ketiga variabel yang didapatkan, dapat dihitung deskriptor turunan yang akan digunakan Penjelasan perancangan ekstraksi ciri citra biji kopi dapat dilihat di subbab 3.2.4

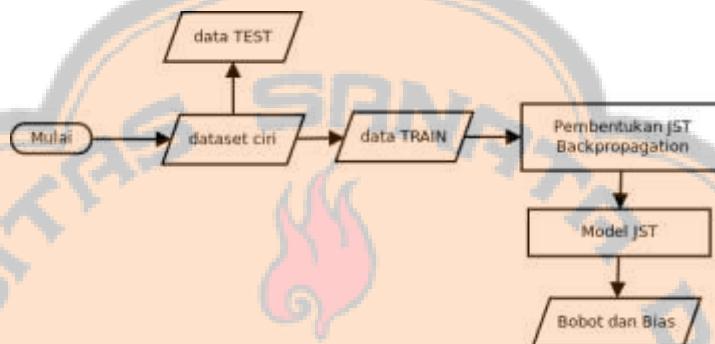


**Gambar 3.4 – Proses Ekstraksi ciri citra biji kopi yang dilakukan**

### 3.2.1.3. Proses Klasifikasi dengan Jaringan Saraf Tiruan

#### 3.2.1.3.1. Proses *Training*

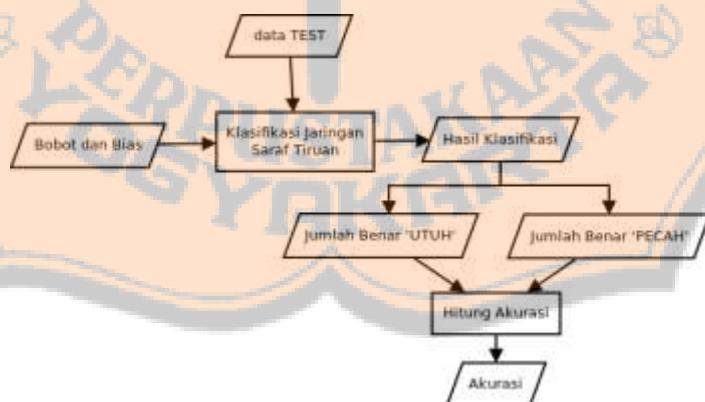
Pada proses *training* jaringan saraf tiruan, data hasil ekstraksi ciri kemudian dibagi untuk *training* dan *testing*. Diawali dengan pembentukan jaringan, hingga model jaringan saraf tiruannya, beserta bobot dan biasnya.



**Gambar 3.5 – Proses pelatihan jaringan saraf tiruan untuk identifikasi**

#### 3.2.1.3.2. Proses *Testing*

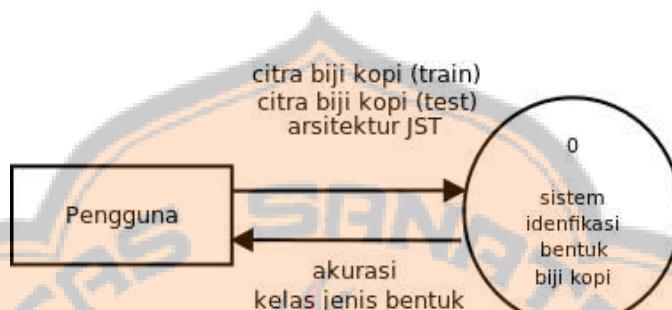
Pada proses *testing* jaringan saraf tiruan, data hasil ekstraksi ciri yang diperuntukkan untuk *testing* dan bias dan bobot hasil *training* jaringan saraf tiruan menjadi masukan. Diawali dengan pembentukan jaringan, hingga model jaringan saraf tiruannya, beserta bobot dan biasnya.



**Gambar 3.6 – Proses Uji klasifikasi dari jaringan saraf tiruan hasil pelatihan**

### 3.2.2. Diagram Konteks

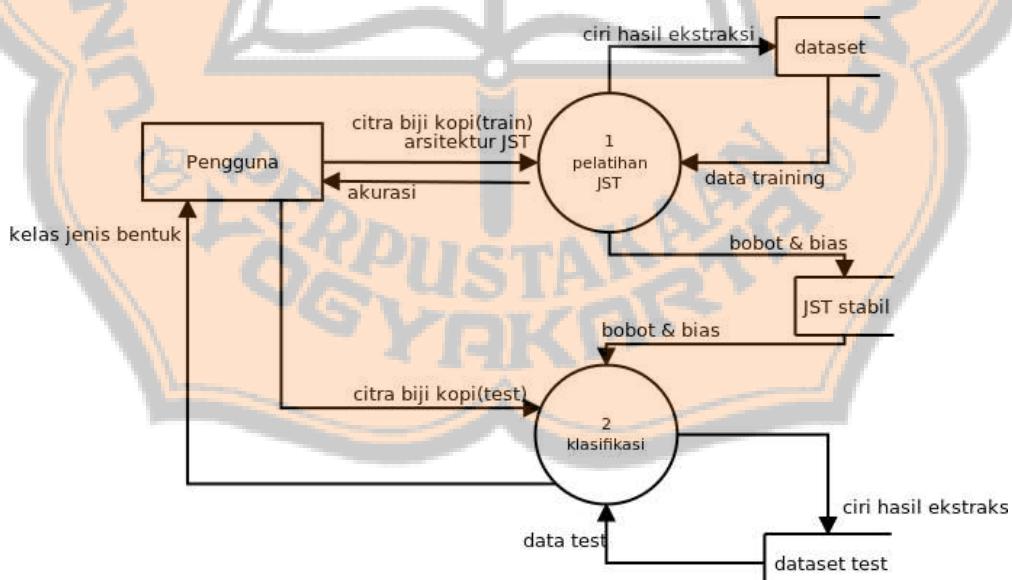
Sistem ini akan menggunakan data masukan berupa citra gambar sampel biji kopi. Dan sistem ini akan menghasilkan akurasi dari proses identifikasi dan kelas jenis bentuk dari citra biji kopi masukan



**Gambar 3.7 – Diagram Konteks Sistem**

#### 3.2.2.1. DFD Level 1 Sisi Pengguna

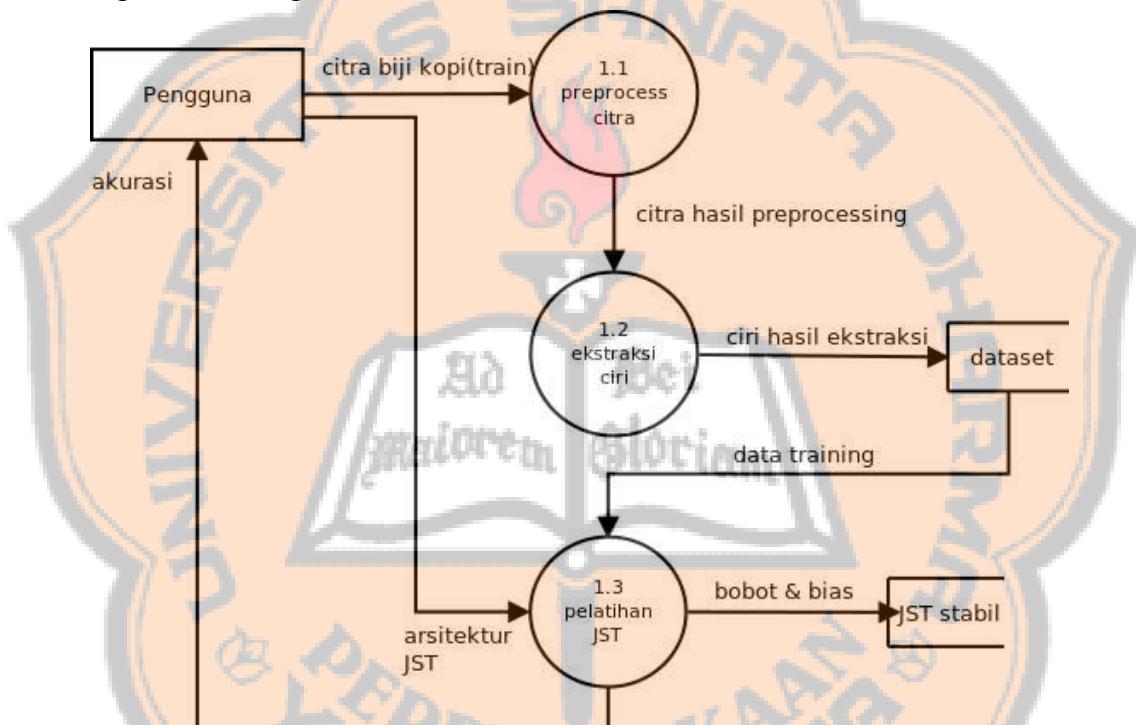
Secara garis besar, sistem yang akan dikembangkan melakukan 2 buah fungsi utama, yakni proses pelatihan jaringan saraf tiruan dan klasifikasi terhadap citra biji kopi dari jaringan yang sudah disusun.



**Gambar 3.8 – Diagram Alur Data, Level 1 Sistem**

### 3.2.2.2. DFD Level 2 Sisi Pengguna Proses Pelatihan JST

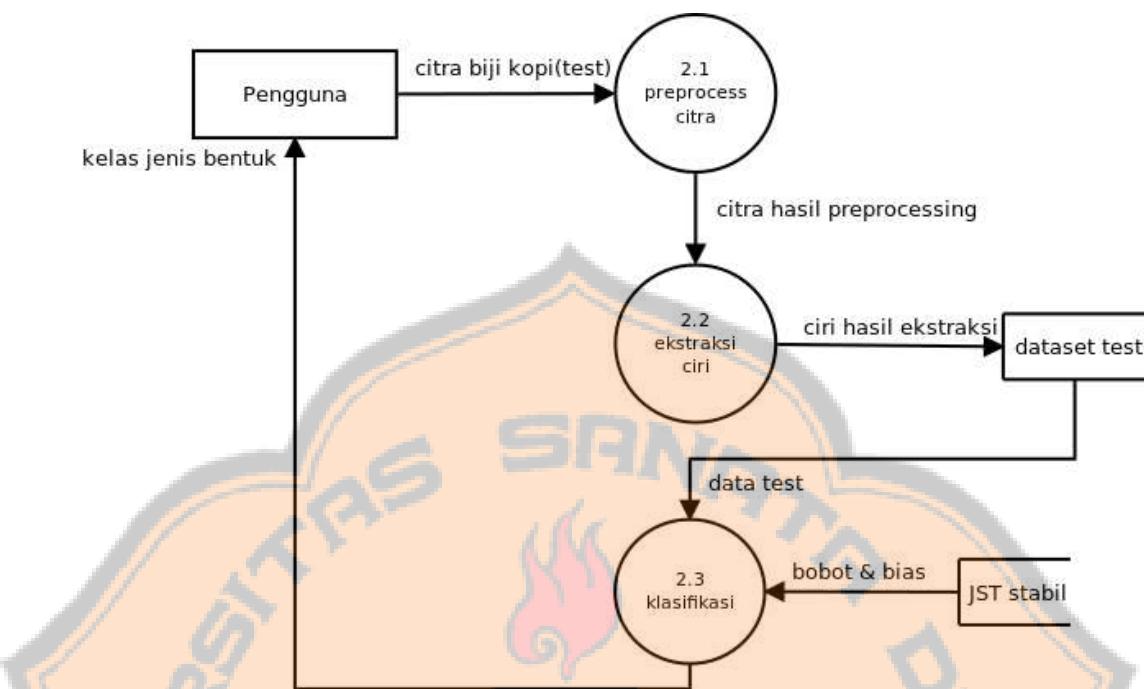
Pada proses pelatihan jaringan saraf tiruan, proses yang dilakukan adalah *prerpocessing* data citra masukan, ekstraksi ciri dan kemudian data dimasukkan kedalam sebuah dataset. Kemudian dilakukan proses pelatihan dengan masukan berupa spesifikasi arsitektur jaringan yang akan digunakan. Dari proses ini dilakukan pelatihan dengan data dibagi menjadi beberapa *fold*, sebagai data uji, tes, dan validasi. Di akhir pelatihan dilakukan penyimpanan bobot dan bias yang akan digunakan sebagai klasifikasi.



Gambar 3.9 – Diagram alur data, Level 2 Proses Pelatihan sistem

### 3.2.2.3. DFD Level 2 Sisi Pengguna Proses Klasifikasi JST

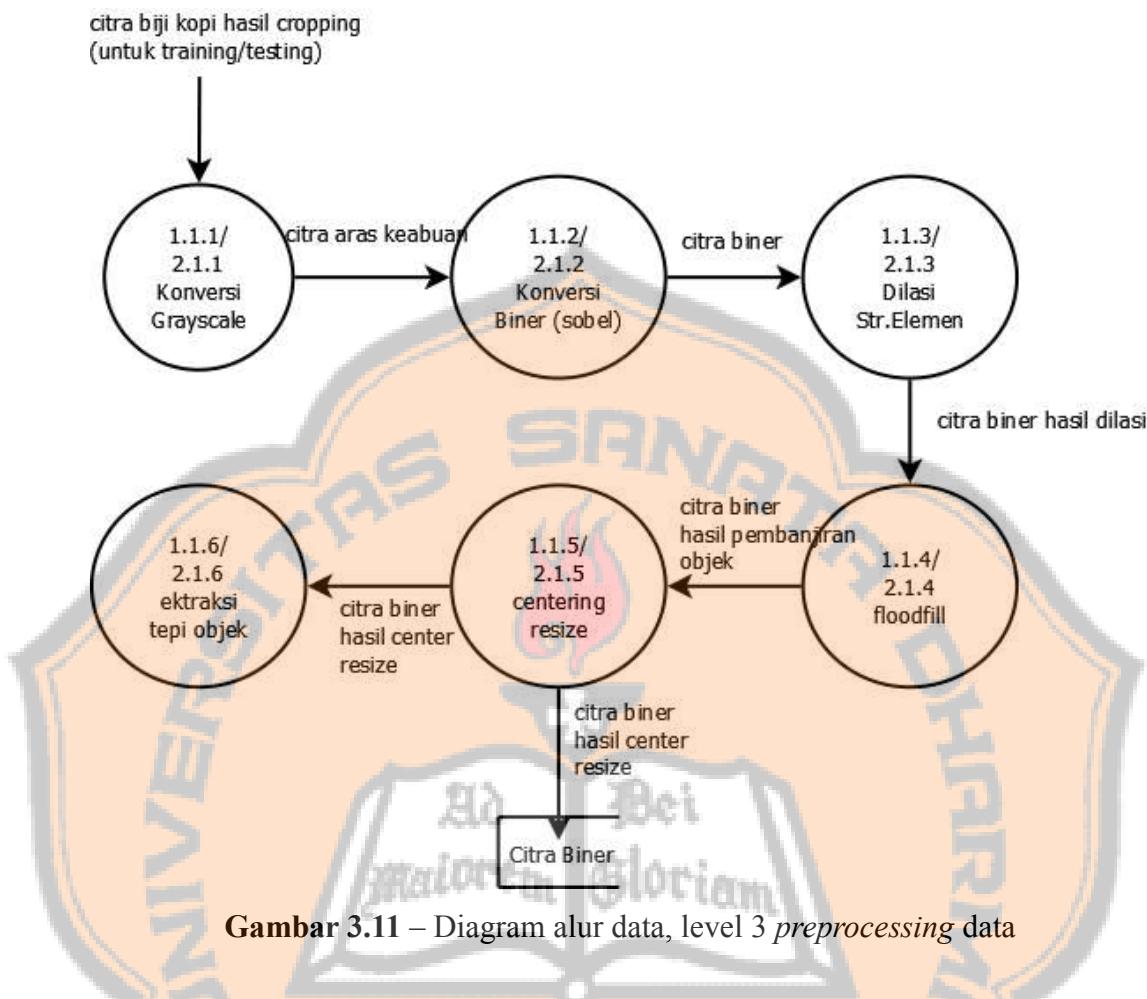
Proses klasifikasi dilakukan setelah proses pelatihan jaringan saraf tiruan selesai dilakukan. Proses yang dilakukan hampir sama, yakni diawali dengan *preprocessing* citra masukan (yang digunakan dalam uji tunggal), dilanjutkan dengan ekstraksi ciri terhadap citra tersebut. Langkah terakhir yang dilakukan adalah melakukan klasifikasi dengan menggunakan bobot bias yang didapatkan dari proses pelatihan, sehingga keluaran berupa kelas jenis bentuk citra biji kopi tersebut.



Gambar 3.10 – Diagram alur data, Level 2 proses klasifikasi

#### 3.2.2.4. DFD Level 3 Sisi Pengguna *Preprocessing Citra* (1.1, 2.1)

Dalam proses *preprocessing* citra biji kopi, baik dalam proses pelatihan maupun pengujian, proses yang dilakukan sama. Diawali dengan masukan berupa citra hasil *cropping* manual, citra tersebut akan dilakukan pengubahan ke aras keabuan, dilanjutkan dengan deteksi sobel, kemudian dilasi dengan *structure element*, *image filling*, dan diakhiri dengan *centering* dan *resizing* bidang latar citra biji kopi. Citra hasil proses inilah yang akan mengalami tahap selanjutnya, ekstraksi ciri. Sebagai visualisasi, dari citra ini juga dilakukan proses ekstraksi tepi, sebagai tampilan perimeternya saja.

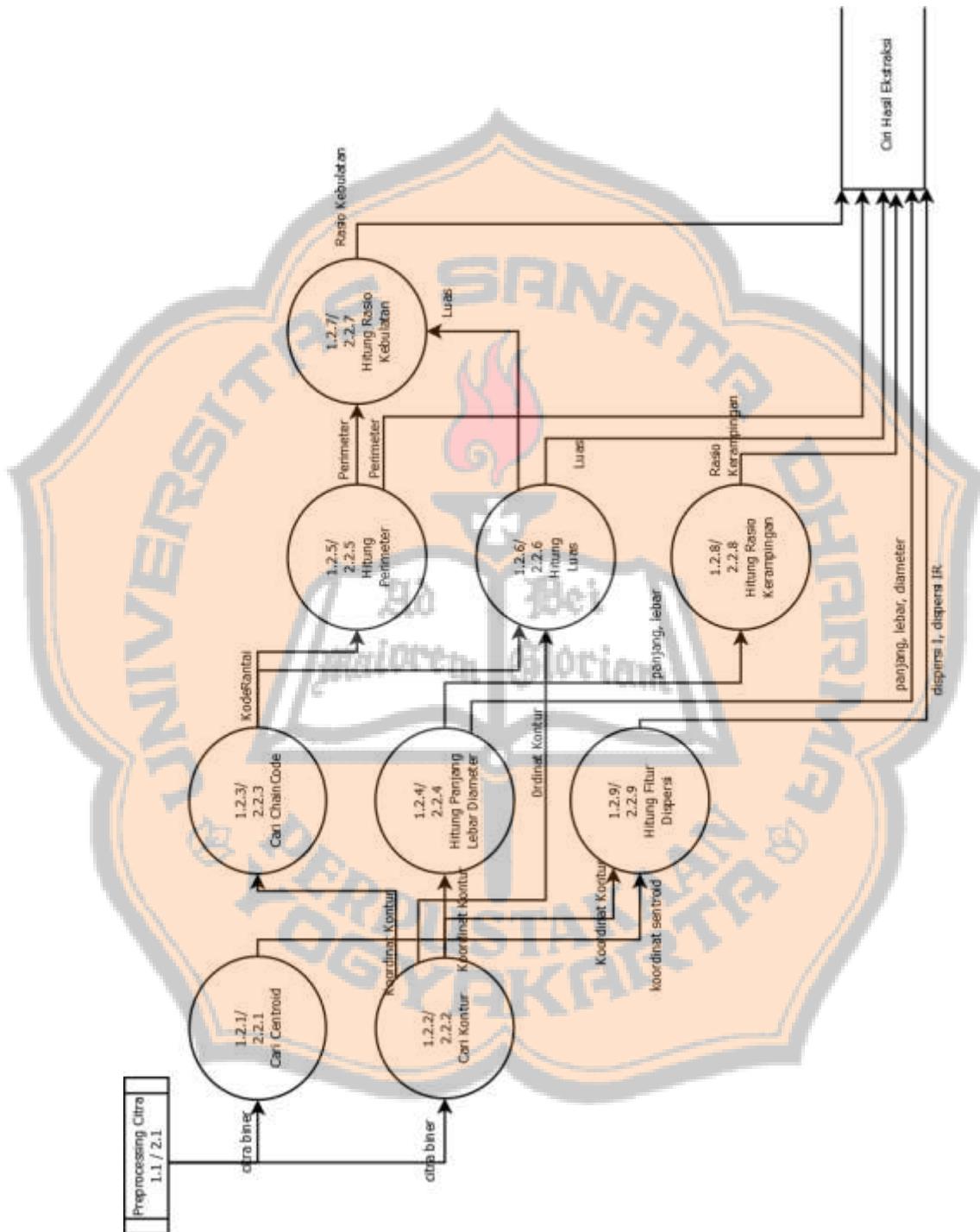


Gambar 3.11 – Diagram alur data, level 3 *preprocessing* data

### 3.2.2.5. DFD Level 3 Sisi Pengguna Proses Ekstraksi Ciri (1.2, 2.2)

Proses Ekstraksi ciri, baik pada proses pelatihan maupun pengujian jaringan saraf tiruan mengalami proses ekstraksi yang sama. Masukan berupa citra biner hasil *preprocessing* sebelumnya. Yang pertama dilakukan adalah penghitungan koordinat sentroid (1.2.1/2.2.1) dan koordinat penyusun kontur objek (1.2.2/2.2.2). Untuk penghitungan kode rantai (1.2.3/2.2.3), yang menjadi masukan adalah koordinat kontur. Pada penghitungan panjang, lebar, diameter (1.2.4/2.2.4) yang menjadi masukan adalah koordinat kontur. Pada penghitungan fitur dispersi (1.2.9/2.2.9), yang menjadi masukan adalah koordinat sentroid dan koordinat kontur. Pada penghitungan perimeter (1.2.5/2.2.5), yang menjadi masukan adalah kode rantai. Pada penghitungan luas (1.2.6/2.2.6), yang menjadi masukan adalah kode rantai dan ordinat kontur. Pada penghitungan rasio kerampingan (1.2.8/2.2.8), yang menjadi masukan adalah panjang (diameter) dan lebar. Pada penghitungan rasio

kebulatan (1.2.7/2.2.7), yang menjadi masukan adalah perimeter dan luas. Semua hasil ekstraksi ciri ini kemudian dimasukkan dalam sebuah *dataset*.



Gambar 3.12 – Diagram alur data, level 3 ekstraksi ciri

### 3.2.3. Rancangan Implementasi *Preprocessing Data*

Dalam penelitian ini, preprocessing data yang digunakan adalah sebagai berikut.

#### 3.2.3.1. *Cropping (Manual)*

*Cropping* merupakan proses yang digunakan untuk memotong citra hasil *capture* dari kamera digital yang digunakan. Proses ini bertujuan membuat data citra tepat berbentuk persegi, sesuai dengan bentuk penampang tempat obyek biji kopi diletakkan. Setiap citra biji kopi berukuran 256x256 piksel, dan latar belakang biji kopi diberikan warna putih, menggunakan perangkat lunak GIMP (GNU *Image Manipulation Program*) 2.8.16. Proses dilakukan dengan melakukan seleksi terhadap persatuan biji kopi, kemudian dilanjutkan dengan memindahkan biji kopi hasil seleksi kedalam worksheet baru berukuran 256 x 256 piksel. Pembersihan latar objek kopi dilakukan dengan menggunakan *Fuzzy Selection Tools* dan *Brush Tools* untuk menghapus objek selain kopi (pasir, dll).



Gambar 3.13 – Contoh hasil foto biji kopi pada penampang



Gambar 3.14 - Citra biji kopi hasil *cropping*, dengan ukuran 256 x 256 piksel

### 3.2.3.2. *Grayscaleing*

Pada tahap ini, citra biji kopi yang berwarna akan diubah menjadi citra keabuan. Dengan persamaan (2.3), layer *Red*, *Green*, dan *Blue* dari citra akan dilakukan penghitungan, sehingga didapatkan citra hasil. Penelitian ini menggunakan *toolbox image processing* MATLAB r2012b dengan fungsi ‘*rgb2gray*’

<b>Algoritma 3.1 – Proses <i>Grayscaleing</i> ‘<i>rgb2gray</i>’ (MATLAB documentation)</b>
--

**Masukan :**  $f(m,n)$  citra masukan berukuran  $mxn$  piksel

**Keluaran :**  $g(m,n)$  citra baru dengan aras keabuan (*grayscale*)

1. Perulangan 1 s.d m, 1 s.d n

2 .  $g(m,n) = 0.2989 * R(f(m,n)) + 0.5870 * G(f(m,n)) + 0.1140 * B(f(m,n))$

### 3.2.3.3. Deteksi Tepi Sobel

Pada tahap ini, akan dilakukan pendektsian tepi dari gambar yang sudah dilakukan perubahan ke skala keabuan sebelumnya. Dimulai dengan penghitungan *threshold*, kemudian dengan penggunaan *fudgefactor* yang dikalikan dengan *threshold*, yang menjadi parameter masukan dalam proses deteksi tepi sobel. Citra yang dihasilkan merupakan citra biner, dengan tepi dan kontur dalam biji yang berwarna putih. Penelitian ini menggunakan *toolbox image processing* MATLAB r2012b dengan fungsi ‘*edge*’

<b>Algoritma 3.2 – Proses deteksi biner Sobel dengan toolbox image MATLAB</b>
---

**Masukan :**  $f(m,n)$ , citra grayscale masukan berukuran  $mxn$  piksel, *fudgefactor*

**Keluaran :**  $g(m,n)$  citra baru biner hasil deteksi Sobel

1. Tentukan nilai *threshold*

2.  $g = \text{hasil konversi } f \text{ dengan toolbox dengan parameter } threshold \times fudgefactor \text{ dengan perintah } 'edge'$

### 3.2.3.4. Dilasi Structure Element

Digunakan untuk mempertegas tepi yang berhasil dideteksi menggunakan operator sobel sebelumnya. Dilakukan dengan mendilasi 2 (dua) buah kernel (*structure element*). Penelitian ini menggunakan *toolbox image processing* MATLAB r2012b dengan fungsi ‘*imdilate*’

### 3.2.3.4.1. Dilasi dengan Se90

Dilasi tahap pertama dilakukan dengan sebuah garis vertikal berukuran 3x3 piksel, dan didilasikan dengan piksel hasil deteksi tepi.

### 3.2.3.4.2. Dilasi dengan Se0

Dilasi tahap pertama dilakukan dengan sebuah garis horizontal berukuran 3x3 piksel, dan didilasikan dengan piksel hasil deteksi tepi yang sudah dilakukan dilasi dengan ‘Se90’ sebelumnya.

<b>Algoritma 3.3 – Proses dilasi dengan toolbox image MATLAB</b>
--

**Masukan :**  $f(m,n)$ , citra biner hasil deteksi tepi sobel

**Keluaran :**  $g(m,n)$  citra biner hasil dilasi *structure element*

1. StructureElement90  $\leftarrow$  kernel 3x3 garis vertical
2. StructureElement0  $\leftarrow$  kernel 3x3 garis horizontal
3.  $g = \text{hasil dilasi } f \text{ dengan StructureElemen90 dan StuctureElemen0 dengan perintah 'imdilate'}$

### 3.2.3.5. *Image Filling*

Merupakan proses yang digunakan untuk mengisi lubang dari citra hasil operasi dilasi sebelumnya. Menggunakan operasi *flood-fill*, dari piksel latar dari citra biner masukannya. Penelitian ini menggunakan *toolbox image processing* MATLAB r2012b dengan fungsi ‘imfill’

<b>Algoritma 3.4 – Proses dilasi imfill dengan toolbox image MATLAB</b>
---

**Masukan :**  $f(m,n)$ , citra biner hasil dilasi

**Keluaran :**  $g(m,n)$  citra biner hasil pembanjiran

1.  $g = \text{hasil proses dilasi dan pembanjiran dengan perintah 'imfill'}$

### 3.2.3.6. *Image Center & Resize*

Merupakan proses yang dilakukan untuk mencari *centroid* dari obyek biji kopi yang didapatkan dari hasil proses sebelumnya, kemudian melakukan translasi objek kopi tersebut dengan sebuah bidang baru, sehingga objek kopi benar-benar berada ditengah sesuai *centroidnya*, dan background ditambahkan panjang atau lebarnya. Algoritma untuk proses ini dibuat oleh Frederik Kratzert (2015) dengan menggunakan beberapa kombinasi proses penambahan bidang citra dan translasi

**Algoritma 3.5 – Proses *image center & resize***

**Masukan :**  $f(m,n)$ , citra biner hasil dilasi

**Keluaran :**  $g(m,n)$  citra biner hasil pemindahan

1. Lakukan penambahan baris atau kolom untuk membuat bidang citra menjadi tepat persegi, gunakan perintah ‘padarray’
2. Hitung *centroid* bidang citra tersebut dengan perintah ‘regionprops’
3. Lakukan penambahan baris atau kolom pada bidang citra sesuai dengan titik *centroid* yang didapatkan dengan perintah ‘padarray’
4.  $g$  = merupakan hasil translasi  $f$  dengan citra bidang hasil penambahan dengan perintah ‘circshift’

**3.2.3.7. Tepi Objek**

Merupakan proses yang dilakukan untuk mengekstraksi tepi objek dari citra yang sudah di *Center* dan *Resize* sebelumnya. Hasil keluaran merepresentasikan kontur dari biji kopi.

**Algoritma 3.6 – Proses tepi biner**

**Masukan :**  $f(m,n)$ , citra biner hasil *centering-resizing*

**Keluaran :**  $g(m,n)$  citra biner tepi

1. Lakukan indeks dari 8 ketetanggaan pada setiap piksel
2. Jika jumlah piksel dalam 8 tetangga = 8,  $g(m,n) = 0$
3. Jika tidak  $g(m,n) = f(m,n)$

**3.2.4. Rancangan Implementasi Ekstraksi Ciri : *Basic Region Descriptor***

Setelah data biji kopi mengalami *preprocessing*, selanjutnya dilakukan pencarian ciri dari citra biji kopi yang sudah berupa biner. Dalam proses ekstraksi ini, dilakukan pencarian kontur dan rantai kode terlebih dahulu. Selanjutnya dilakukan penghitungan ciri-ciri turunan dari kontur dan kode yang didapatkan.

**3.2.4.1. Pencarian Kontur**

Metode untuk pelacakan kontur yang digunakan adalah pelacakan kontur internal dengan algoritma Moore. Masukan dari tahap pertama ini merupakan citra biner hasil *preprocessing*, dan keluarannya merupakan sebuah matriks ( $2 \times n$ ) yang berisi pasangan koordinat kontur.

**Algoritma 3.7 – Memperoleh kontur internal Moore**

**Masukan :**  $f(m,n)$  citra masukan berukuran  $mxn$  piksel

**Keluaran :**  $kontur(s)$  larik yang berisi piksel kontur sebanyak s

1. dapatkan piksel terkiri teratas yang bernilai 1, selanjutnya posisi piksel dicatat pada  $b0$  dan posisi untuk memperoleh piksel berikutnya dicatat pada  $c0$ , yang mula-mula diisi 4 (arah barat)
2. periksa tetangga  $b0$  searah jarum jam dimulai dari  $c0$ . Piksel pertama yang bernilai 1 dicatat pada  $b1$ . Adapun posisi yang mendahului  $b1$  dicatat pada  $c1$
3.  $kontur(1) \leftarrow b0; kontur(2) \leftarrow b1; jum \leftarrow 2$
4.  $b \leftarrow b1; c \leftarrow c1$
5. perulangan:

Cari piksel pada 8 tetangga yang pertama kali bernilai 1 dengan pencarian dimulai dari arah  $c$  dengan menggunakan pola arah jarum jam.

Catat posisi piksel tersebut ke  $b$

Catat posisi yang mendahului piksel tersebut ke  $c$

Tambahkan  $b$  sebagai kontur dengan :

$jum \leftarrow jum + 1; kontur(jum) \leftarrow b;$

Jika  $b = b0$

keluar perulangan;

**3.2.4.2. Penentuan Rantai Kode (*Chain code*)**

Metode untuk penentuan rantai kode adalah algoritma Freeman. Dimana masukannya berupa kontur biji kopi, dan keluarannya berupa kode, serta koordinat titik awal rantai kode tersebut dimulai.

**Algoritma 3.8 – Proses pencarian kode rantai**

**Masukan :**  $Kontur(s)$ , kontur penyusun objek

**Keluaran :**  $KodeRantai(s)$

1. Perulangan selama jumlah  $Kontur(s)$   
Hitung  $selisihY, selisihX$   
 $Indeks = 3 \times selisihY + selisihX + 5$   
 $KodeRantai(s) =$  kode pada indeks (Gambar 2.15)

**3.2.4.3. Ciri 1 : Luas**

Metode untuk penghitungan luas menggunakan pendekatan penghitungan luas dengan kode rantai pada algoritma Putra. Masukan berupa kontur dan kode rantai, sedangkan keluarannya berupa luas dalam satuan *units*.

**Algoritma 3.9 – Proses pencarian Luas****Masukan :** Kontur( $s$ ), KodeRantai( $s$ )**Keluaran :** Luas

1. Perulangan selama jumlah KodeRantai( $s$ )
2.  $Ordinat = \text{Piksel}Y$  pada Kontur ke  $s$
3. Jika kode
  - [0] Luas = Luas +  $Ordinat$
  - [1] Luas = Luas +  $Ordinat + 0.5$
  - [3] Luas = Luas -  $Ordinat + 0.5$
  - [4] Luas = Luas -  $Ordinat$
  - [5] Luas = Luas -  $Ordinat + 0.5$
  - [2][6] Luas = Luas
  - [7] Luas = Luas +  $Ordinat - 0.5$

**3.2.4.4. Ciri 2 : Perimeter**

Metode untuk penghitungan keliling menggunakan pendekatan penghitungan dari kode rantai. Masukan berupa kode rantai, keluaran berupa perimeter dalam satuan units.

**Algoritma 3.10 – Proses pencarian perimeter****Masukan :** KodeRantai( $s$ )**Keluaran :** Perimeter

1. Perulangan selama jumlah KodeRantai( $s$ )
2. Jika kode
  - [Genap] jumlahGenap+=1
  - [Ganjil] jumlahGanjil+=1
3.  $\text{Perimeter} = \text{jumlahGenap} + \text{jumlahGanjil} * \sqrt{2}$

**3.2.4.5. Ciri 3, 4, 5 : Panjang, Lebar, Diameter**

Pada penghitungan panjang, lebar dan diameter, yang menjadi masukan adalah kontur citra biji kopi. Keluaran berupa panjang (beserta koordinat 2 titik terjauh sebagai panjang objek, serta gradiennya), lebar (beserta koordinat 2 titik terjauh dari gradien yang tegak lurus dari garis panjang), dan diameter dari objek biji kopi.

**Algoritma 3.11 – Estimasi panjang (diameter), lebar bentuk**

**Masukan :** Kontur(s) → Kontur internal bentuk

**Keluaran :** panjang, diameter, lebar

1.  $S \leftarrow$  jumlah elemen Kontur
2.  $jarak\_maks \leftarrow 0$
3. Perulangan  $p = 1$  s.d  $c-1$
4. Perulangan  $q = p+1$  s.d  $c$ 
  - Jika  $|Kontur(p) - Kontur(q)| > jarak\_maks$
  - $piksel1 \leftarrow p$
  - $piksel2 \leftarrow q$
  - end
  - berhenti perulangan
5.  $diameter \leftarrow jarak\_maks$
6.  $panjang = diameter$
7. Hitung gradien piksel1 dan piksel2 (GradienGarisPanjang)
8.  $GradienGarisLebar = -1/GradienGarisPanjang$
9. Perulangan  $p = 1$  s.d  $c-1$
10. Perulangan  $q = p+1$  s.d  $c$ 
  - Jika  $|Kontur(p) - Kontur(q)| > jarak\_maks$
  - Jika gradien  $Kontur(p)$  dan  $Kontur(q)$  dalam toleransi GradienGarisLebar
  - $piksel3 \leftarrow p$
  - $piksel4 \leftarrow q$
  - end
  - berhenti perulangan
11.  $Lebar = jarak\_maks$

**3.2.4.6. Ciri 6 : Rasio Kebulatan**

Dalam penghitungan rasio kebulatan, yang menjadi masukan adalah area luas dan perimeter objek kopi.

**Algoritma 3.12 – Proses pencarian Rasio Kebulatan**

**Masukan :** Luas, Perimeter

**Keluaran :** RasioKebulatan

1.  $RasioKebulatan = 4\pi \times Luas / Perimeter^2$

**3.2.4.7. Ciri 7 : Rasio Kerampingan**

Dalam penghitungan rasio kerampingan, yang menjadi masukan adalah panjang dan lebar objek kopi.

**Algoritma 3.13 – Proses pencarian Rasio Kerampingan****Masukan :** Panjang, Lebar**Keluaran :** RasioKerampingan

1. RasioKebulatan = Lebar/Panjang

**3.2.4.8. Penghitungan *Centroid***

Dalam pencarian pusat massa objek kopi, digunakan nilai rerata koordinat setiap piksel. Keluarannya berupa koordinat pusat massa objek kopi

**Algoritma 3.14 – Estimasi penghitungan sentroid****Masukan :**  $f(m,n) \rightarrow$  citra masukan berukuran  $m \times n$ **Keluaran :** pusat\_x, pusat\_y

1.  $pusat\_x \leftarrow 0, pusat\_y \leftarrow 0$
2.  $luas \leftarrow 0$
3. perulangan  $q = 1$  s.d  $m$
4. perulangan  $p = 1$  s.d  $n$
5. jika  $f(q,p) = 1$   
 $luas = luas + 1$   
 $pusat\_x \leftarrow pusat\_x + p$   
 $pusat\_y \leftarrow pusat\_y + q$   
end  
keluar perulangan
6.  $pusat\_x \leftarrow pusat\_x / luas$
7.  $pusat\_y \leftarrow pusat\_y / luas$

### 3.2.4.9. Ciri 8 dan 9 : Dispersi I, Dispersi IR

Dalam pencarian fitur dispersi, yang menjadi masukan berupa kontur objek kopi, serta koordinat pusat massanya. Keluaran berupa I dan IR

<b>Algoritma 3.15 – Estimasi penghitungan Dispersi I dan IR</b>
---

**Masukan :** *pusat\_x, pusat\_y, Kontur(s), Luas*

**Keluaran :** *dispersiI, dispersiIR*

1. Perulangan 1 s.d *s*
2. Hitung jarak antara titik *Kontur(s)* dengan (*pusat\_x, pusat\_y*)
3. Cari *jarak\_maksimum* dari langkah 2
4. Cari *jarak\_minimum* dari langkah 2
5.  $dispersiI = \pi \times jarak\_maksimum / Luas$
6.  $dispersiIR = jarak\_terbesar / jarak\_minimum$

Semua ciri yang dihasilkan disimpan dalam sebuah *Structure Array*, dan kemudian dilakukan pembuatan dataset berdasarkan kelas bentuk yang kemudian akan digunakan pada proses identifikasi.

### 3.2.5. Perancangan Jaringan Saraf Tiruan

#### 3.2.5.1. Lapisan Tersembunyi

Pada penelitian ini menggunakan sistem Jaringan Saraf Tiruan *Backpropagation* 1 Hidden Layer dan 2 Hidden Layer

#### 3.2.5.2. Neuron Tersembunyi

*Hidden neuron* yang digunakan dalam penelitian adalah 5, 10, 15, 20, 25 buah *neuron* pada *hidden layer* ke-1, dan yang optimal akan menjadi *layer* ke 1, dan pada *layer* ke-2 akan dicoba kembali *neuron* dengan jumlah 5, 10, 15, 20, dan 25

#### 3.2.5.3. Fungsi Aktivasi

Fungsi Aktivasi yang digunakan dalam sistem jaringan saraf tiruan *backpropagation* ini adalah *softmax*

#### 3.2.5.4. Fungsi Training

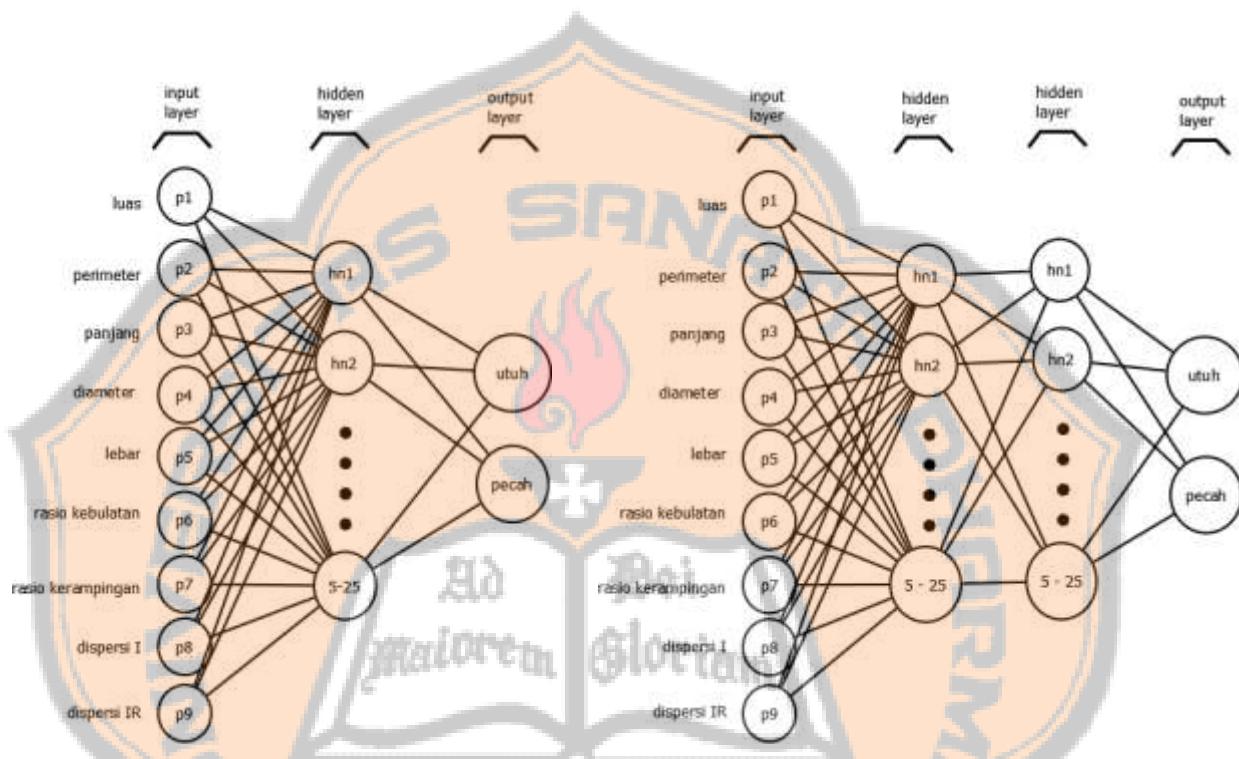
Sebagai parameter pengujian, akan digunakan *training*, yakni ‘trainlm’, ‘trainbfg’, ‘trainrp’, ‘trainscg’, ‘traincgb’, ‘traincfg’, ‘traincgp’, ‘trainoss’, dan ‘traingdx’.

#### 3.2.5.5. Target

Penentuan kelas untuk setiap biji kopi adalah sebagai berikut

**Tabel 3.1 – Target Jaringan Saraf Tiruan *Backpropagation* untuk kelas**

No.	Jenis Kelas	Target
1.	Biji Utuh	[ 1 0 ]
2.	Biji Pecah	[ 0 1 ]

**Gambar 3.15** Arsitektur Jaringan Saraf Tiruan yang akan digunakan dalam penelitian ini

### 3.2.5.6. *Epoch* (iterasi)

Dalam penelitian digunakan batas iterasi 100 untuk jaringan saraf tiruan backpropagation dengan 1 dan 2 hidden layer,

### 3.2.6. Pemilihan Data *Training-Testing-Validation*

Dalam penelitian ini menggunakan *k-fold validation* yakni 3 dan 5 *fold* dengan pembagian sebagai berikut

#### 3.2.6.1. *3 Fold Cross Validation*

Pada penggunaan 3 *fold*, maka, data pada masing-masing kelas (pecah dan utuh) akan dibagi menjadi 3 bagian. Kemudian jaringan saraf tiruan yang

disusun akan dilatih sebanyak 3 kali, dengan pemilihan data *train*, *validation*, dan *test* seperti tabel 3.2 dibawah.

**Tabel 3.2 – 3 Fold Cross Validation**

Percobaan	Train	Validation	Test
1	3	1	2
2	1	2	3
3	2	3	1

### 3.2.6.2. 5 Fold Cross Validation

Pada penggunaan 5 fold, maka, data pada masing-masing kelas (pecah dan utuh) akan dibagi menjadi 5 bagian. Kemudian jaringan saraf tiruan yang disusun akan dilatih sebanyak 5 kali, dengan pemilihan data *train*, *validation*, dan *test* seperti tabel 3.3 dibawah.

**Tabel 3.3 – 5 Fold Cross Validation**

Percobaan	Train	Validation	Test
1	3,4,5	1	2
2	1,4,5	2	3
3	1,2,5	3	4
4	1,2,3	4	5
5	2,3,4	5	1

Semua pembagian data menggunakan sistem indeks, dimana dari dataset yang digunakan dihitung jumlah data tiap bagian dengan penghitungan modulo, sehingga didapatkan indeks dari tiap bagian. Dalam proses training menggunakan *toolbox neural network* MATLAB, pembagian data menggunakan fungsi *divideind*, dimana data akan dibagi berdasarkan indeksnya, menjadi data latih, data uji, dan data validasi.

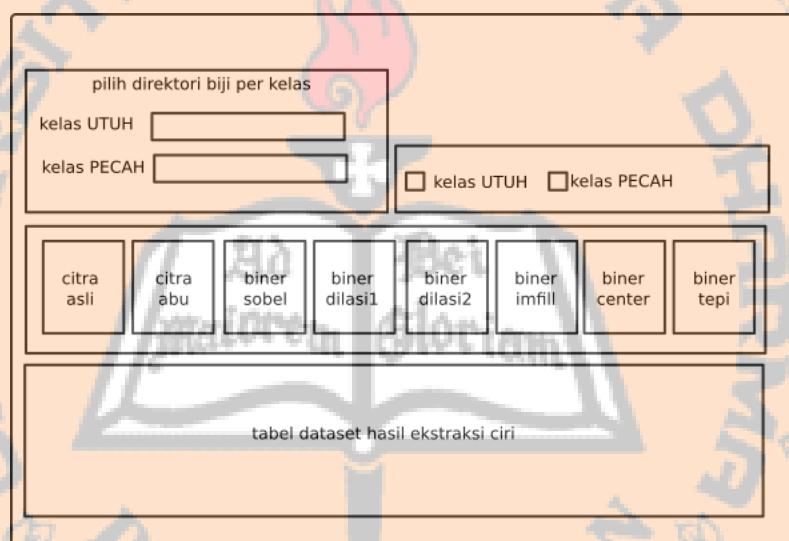
### 3.2.7. Model Desain

#### 3.2.7.1. *User interface Sistem*

Berikut merupakan rancangan antarmuka sistem uji yang akan digunakan dalam penelitian ini. Tampilan disusun sedemikian rupa sehingga dapat mempermudah pengguna dalam melakukan pengujian.

##### 3.2.7.1.1. Halaman *Preprocessing* (Pembuatan Dataset)

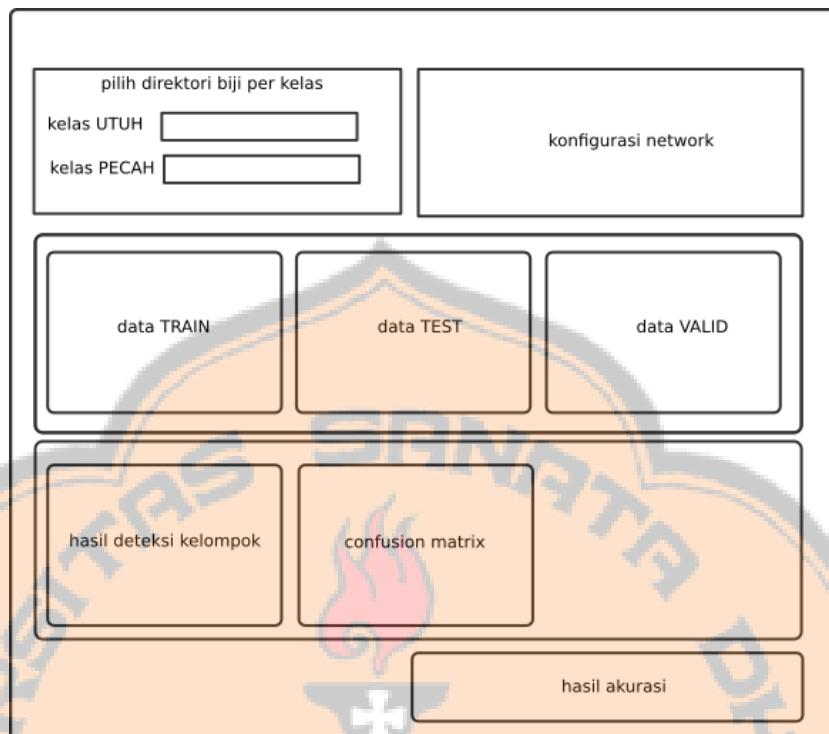
Antar muka ini akan digunakan sebagai tampilan *preprocessing* data, dan tampilan data hasil ekstraksi ciri. Masukan dalam antarmuka ini berupa direktori dari folder yang berisi kumpulan citra dari masing-masing kelas biji utuh dan biji pecah



Gambar 3.16 – Home Screen *Preprocessing*

##### 3.2.7.1.2. Halaman Test Group

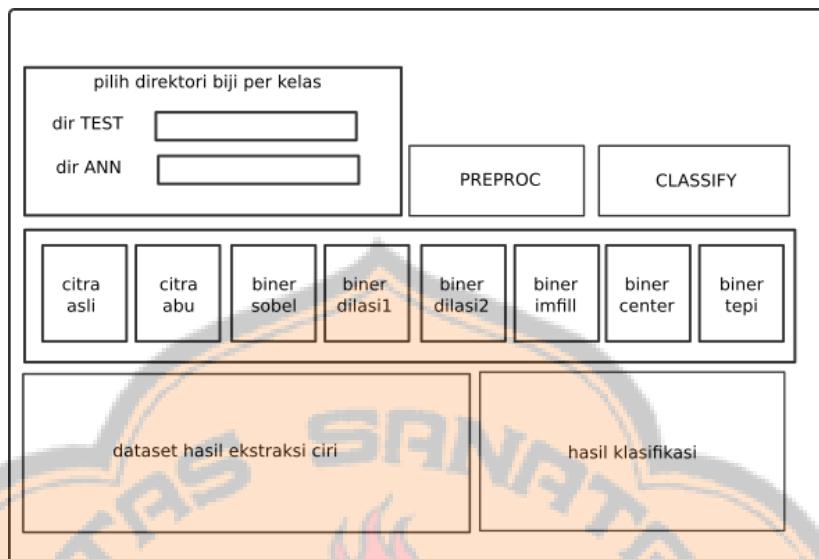
Antar muka ini digunakan dalam penyusunan dan perancangan dari jaringan saraf tiruan yang akan digunakan dalam proses klasifikasi. Antar muka ini akan menampilkan *dataset* yang akan digunakan sebagai *training*, *testing*, dan *validation*. Selain itu, ditampilkan juga hasil klasifikasi kelompok data, serta *confusion matrix* nya. Masukan antar muka ini berupa direktori *dataset* biji utuh dan *dataset* biji pecah



Gambar 3.17 – Halaman Set Data Training

### 3.2.7.2. Halaman *Test* Tunggal

Antar muka ini digunakan untuk melakukan pengujian dari jaringan saraf tiruan yang sudah dibuat sebelumnya. Uji yang dilakukan merupakan data tunggal. Masukan berupa direktori citra yang akan diuji, dan direktori file jaringan saraf tiruan yang sudah disimpan. Pada antar muka ini ditampilkan proses *preprocessing*, ekstraksi ciri, dan hasil klasifikasi dari citra biji kopi tersebut.



Gambar 3.18 – Halaman Hasil

### 3.2.8. Analisis Hasil

Penelitian ini menggunakan 3 dan 5 *fold* dalam pembagian data, sehingga percobaan akan dilakukan k (3, dan 5 sesuai percobaan) kali dalam proses *training* dan *testing* nya. Hasil percobaan ini akan berupa *confusion matrix*. Yang menunjukkan data dikenali sebagai kelompoknya.

Tabel 3.4 – Tabel Confusion Matrix untuk akurasi klasifikasi jenis cacat

	1	2
1	■■■■■	■■■■■
2	■■■■■	■■■■■

Dari *confusion matrix* tersebut dapat dihitung besar akurasinya. Penghitungan akurasi dilakukan untuk melihat seberapa optimal jaringan saraf tiruan *backpropagation* dalam mengenali jenis cacat dan klasifikasinya.

$$\text{Akurasi} = \frac{\Sigma \text{databenar}}{\Sigma \text{seluruhdata}} \times 100\% \quad (3.1)$$

## BAB IV

### IMPLEMENTASI SISTEM DAN HASIL

Bab ini membahas uraian dari implementasi sistem yang sudah dibuat, yang berupa hasil penelitian dalam melakukan pengujian terhadap pemilihan data *training – testing – validasi* dengan *k-foldnya* dan jumlah iterasi *training* yang digunakan

#### **4.1. Implementasi *Coding* Sistem**

##### **4.1.1. *Preprocessing***

Pada tahap ini, masukan dalam system berupa direktori file nama dari citra biji kopi yang akan di *preprocess*. Diawali dengan membaca file citra tersebut, kemudian dilanjutkan dengan proses yang sudah dibahas dalam perancangan sebelumnya. Keluaran berupa variabel ‘prep’, yakni citra biner.

###### **Listing Program 4.1 – *Preprocessing* Data**

```
function [a, b, c, d, e, f, prep, edgeprep] = preprocess (I)
    s1 = strel('line', 3, 90);
    s2 = strel('line', 3, 0);
    fudgefactor = .5;
    a = imread(I);
    b = rgb2gray (a);
    [~, threshold] = edge(b, 'sobel');
    c = edge(b, 'sobel', fudgefactor * threshold );
    d = imdilate (c, s1);
    e = imdilate (d, s2);
    f = imfill (e, 'holes');
    f = imfill (f, 'holes');
    prep = centerobject(f);
    edgeprep = tepibiner(prep);
end
```

##### **4.1.1.1. *Resize – Centering***

Tahap ini merupakan bagian dari preprocessing. Fungsi yang dibuat ini digunakan untuk meletakkan objek kopi kedalam sebuah bidang baru, dengan ukuran objek tetap, namun ada perubahan pada bidang latar. Dimana proses ini dilakukan pencarian sentroid, kemudian dilanjutkan dengan translasi objek kopi dalam citra.

**Listing Program 4.2 – Centering – Resizing**

```

function cm = centerobject(bw)
if ~islogical(bw)
    error('Input BW must be of type ''logical'''');
end
cc = bwconncomp(bw,8);
if cc.NumObjects > 1
    error('There is more than one object in the binary image');
end
%adds coloums/rows of zeros to create a square image
sz = size(bw);
if sz(1) < sz(2)
    temp = (sz(2) - sz(1))*0.5;
    if mod(temp,1) > 0
        bw = padarray(bw,[0 1], 'post');
    end
    bw = padarray(bw,[round(temp) 0]);
elseif sz(2) < sz(1)
    temp = (sz(1) - sz(2))*0.5;
    if mod(temp,1) > 0
        bw = padarray(bw,[1 0], 'post');
    end
    bw = padarray(bw,[0 round(temp)]);
end
%gets translation factors in x and y direction
state = regionprops(bw,'Centroid');
sz = size(bw);
delta_y = round(sz(1)/2-state.Centroid(2));
delta_x = round(sz(2)/2-state.Centroid(1));
%extend image, so that translation fit in any case in the boundaries
delta_max = max(abs(delta_y),abs(delta_x));
bw = padarray(bw,[delta_max+10, delta_max+10]);
%execute translation
cm = circshift(bw,[delta_y,delta_x]);
end

```

**4.1.2. Ekstraksi Ciri**

Pada listing berikut, akan dilakukan pemanggilan fungsi-fungsi individual yang melakukan ekstraksi ciri dari citra biner hasil preprocessing sebelumnya. Masukan berupa citra biner hasil *preprocessing*, dan citra biner ini akan mengalami pemrosesan bertahap dalam pengambilan ciri. Ciri awal yang dicari sesuai dengan perancangan adalah kontur dan sentroid. Dari kedua ciri ini, penghitungan ciri-ciri berikutnya dapat dilakukan. Setelah semua ciri didapatkan, akan dilakukan pembentukan *dataset*, dimana terdiri dari nama file, 2 ciri awak, dan 9 ciri utama yang kemudian disimpan pada sebuah file structure

### **Listing Program 4.3 – Ekstraksi Ciri**

```

function [a, b, c, d, e, f, prep, edgeprep, strBijidanCiri] =
functionekstraksi(direktori, kelas)
    [a, b, c, d, e, f, prep, edgeprep] = preprocess(direktori);
    kontur = inbound_tracing(prep);
    [kode_rantai, xawal, yawal] = chain_code(kontur);
    [panjang, lebar] = peroleh_lebar(kontur);
    [diameter] = peroleh_diameter(kontur);
    [d1,d2] = dispersi(kontur, prep);
    hasilLuas2 = luas2(kontur, kode_rantai, prep);
    hasilperim2 = perim2(kode_rantai);
    rasioKebulatan = peroleh_kebulatan (hasilperim2,hasilLuas2);
    [ratioKerampingan] = peroleh_kerampingan(lebar, panjang);
    field1 = 'namaFile';
    field2 = 'konturMoore';
    field3 = 'kodeRantai';
    field4 = 'luas';
    field5 = 'perimeter';
    field6 = 'panjang';
    field7 = 'lebar';
    field8 = 'diameter';
    field9 = 'ratioKebulatan';
    field10 = 'ratioKerampingan';
    field11 = 'dispersi1';
    field12 = 'dispersi2';
    field13 = 'kelas';
    value1 = direktori;
    value2 = kontur;
    value3 = kode_rantai;
    value4 = hasilLuas2;
    value5 = hasilperim2;
    value6 = panjang;
    value7 = lebar;
    value8 = diameter;
    value9 = rasioKebulatan;
    value10 = ratioKerampingan;
    value11 = d1;
    value12 = d2;
    value13 = kelas;
    strBijidanCiri =
struct(field1,value1,field2,value2,field3,value3,field4,value4,field5,value5,fie
ld6,value6,field7,value7,field8,value8,field9,value9,field10,value10,field11,va
ue11,field12,value12,field13,value13);

```

#### **4.1.2.1. Cari Kontur**

Dalam proses pencarian kontur internal, digunakan algoritma pelacakan kontur internal Moore. Pada listing berikut, yang menjadi masukan adalah citra biner hasil *preprocessing* sebelumnya keluaran berupa matriks ukuran mx2, yang merupakan pasangan koordinat kontur-kontur penyusun objek.

**Listing Program 4.4 – Kontur Internal**

```

function [Kontur] = inbound_tracing(BW)
[jum_baris, jum_kolom] = size(BW);
selesai = false;
for p=1:jum_baris
    for q=1:jum_kolom
        if BW(p,q) == 1
            b0.y = p;
            b0.x = q;
            selesai = true;
            break;
        end
    end
    if selesai
        break;
    end
end
c0 = 4;
for p = 1:8
    [dy, dx] = delta_piksel(c0);
    if BW(b0.y + dy, b0.x + dx) == 1
        b1.y = b0.y+dy;
        b1.x = b0.x+dx;
        c1 = sebelum(c0);
        break;
    else
        c0 = berikut(c0);
    end
end
Kontur = [];
Kontur(1,1) = b0.y;
Kontur(1,2) = b0.x;
Kontur(2,1) = b1.y;
Kontur(2,2) = b1.x;
n=2;
b=b1;
c=c1;
while true
    for p=1:8
        [dy,dx] = delta_piksel(c);
        if BW(b.y+dy, b.x+dx) ==1
            b.y = b.y+dy;
            b.x = b.x+dx;
            c = sebelum(c);
            n=n+1;
            Kontur(n,1) = b.y;
            Kontur(n,2) = b.x;
            break;
        else
            c = berikut(c);
        end
    end
    if (b.y == b0.y) && (b.x == b0.x)
        break;
    end
end

```

**4.1.2.2. Hitung *Centroid***

Pada tahap ini, akan dilakukan pencarian titik pusat dari citra masukan. Masukan tahap ini berupa citra biner, sedangkan keluarannya berupa pasangan koordinat pusat x dan pusat y. Implementasi dapat dilihat pada listing program berikut

**Listing Program 4.5 – Cari Sentroid**

```

function [pusatx, pusaty] = centroid(BW)
    [tinggi, lebar] = size(BW);
    pusatx = 0;
    pusaty = 0;
    luas = 0;
    for q=1:tinggi
        for p=1:lebar
            if BW(q,p)==1
                luas = luas+1;
                pusatx = pusatx+p;
                pusaty
                pusaty = pusaty+q;
                pusaty
            end
        end
    end
    pusatx = pusatx/luas;
    pusaty = pusaty/luas;

```

**4.1.2.3. Hitung Kode Rantai**

Pada tahap ini akan dilakukan penghitungan kode rantai. Yang menjadi masukan adalah kontur hasil pelacakan pada proses sebelumnya. Pada tahap ini dilakukan pencarian selisih x dan y antar kontur, sehingga dapat dilakukan pencarian indeks. Dari indeks ini dapat ditentukan kode rantainya. Keluaran tahap berupa teks berisi kode rantai dan koordinat awal x dan y untuk penelusuran ulang.

**Listing Program 4.6 – Hitung Kode Rantai**

```

function [kode_rantai, xawal, yawal] = chain_code(U)
    Kode = ['3', '2', '1', '4', '0', '0', '5', '6', '7'];
    xawal = U(1,2);
    yawal = U(1,1);
    kode_rantai = '';
    for p=2:length(U)
        deltax = U(p,1) - U(p-1,1);
        deltay = U(p,2) - U(p-1,2);
        indeks = 3*deltay+deltax+5;
        kode_rantai = strcat(kode_rantai, Kode(indeks));
    end

```

**4.1.2.4. Hitung Panjang, Lebar, Diameter**

Dalam tahap ini akan dilakukan pencarian panjang (diameter), dan lebar objek kopi. Sebagai masukan dalam tahap ini adalah kontur hasil penelusuran sebelumnya. Dan sebagai keluaran berupa panjang dan lebar, dimana diameter merupakan titik terpanjang objek, dan diasumsikan sama dengan panjang. Implementasi algoritma brute force (Nixon, Aguado) dapat dilihat pada listing program berikut.

**Listing Program 4.7 – Pencarian Panjang (Diameter), Lebar**

```

function [panjang, lebar] = peroleh_lebar(U)
TOLERANSI_JARAK = 3;
U(length(U), :) = [];
n = length(U);
jarak_maks = 0;
piksell1 = 0;
piksell2 = 0;
for p=1:n-1
    for q=p+1:n
        jarak = sqrt((U(p,1)-U(q,1))^2 + ((U(p,2)-U(q,2))^2));
        if jarak>jarak_maks
            jarak_maks = jarak;
            piksell1 = p;
            piksell2 = q;
        end
    end
y1 = U(piksell1,1);
x1 = U(piksell1,2);
y2 = U(piksell2,1);
x2 = U(piksell2,2);
panjang = jarak_maks;
maks = 0;
posx3 = -1;
posx4 = -1;
posy3 = -1;
posy4 = -1;
if (not((abs(x1-x2)<TOLERANSI_JARAK) || ((abs(y1-y2)<TOLERANSI_JARAK))))
    grad1 = (y1-y2)/(x1-x2);
    grad2 = -1/grad1;
    for p=1:n-1
        for q=p+1:n
            x3 = U(p, 2); y3 = U(p,1);
            x4 = U(q, 2); y4 = U(q,1);
            pembagi = (x4-x3);
            if pembagi == 0
                continue;
            end
            grad3 = (y4-y3)/(x4-x3);
            if abs(grad3-grad2)<0.1*abs(grad2)
                jarak = sqrt((x3-x4)^2+(y3-y4)^2);
                if jarak > maks
                    maks = jarak;
                    posx3 = x3;
                    posx4 = x4;
                    posy3 = y3;
                    posy4 = y4;
                end
            end
        end
    end
elseif (abs(y1-y2)<TOLERANSI_JARAK)
    grad1 = 0;
    grad2 = inf;
    for p=1:n-1
        for q=p+1:n
            x3 = U(p, 2); y3 = U(p,1);
            x4 = U(q, 2); y4 = U(q,1);
            deltax = abs(x4-x3);
            if (deltax > abs(y1-y2))
                continue;
            end
            jarak = sqrt((x3-x4)^2+(y3-y4)^2);
            if jarak > maks
                maks = jarak;
                posx3 = x3;
                posx4 = x4;
                posy3 = y3;
                posy4 = y4;
            end
        end
    end
end

```

```

        end
    end
else
    grad1 = inf; grad2 = 0;
    for p=1:n-1
        for q=p+1:n
            x3 = U(p, 2); y3 = U(p,1);
            x4 = U(q, 2); y4 = U(q,1);
            deltax = abs(x4-x3);
            if (deltay > abs(x1-x2))
                continue;
            end
            jarak = sqrt((x4-x3)^2+(y4-y3)^2);
            if jarak > maks
                maks = jarak;
                posx3 = x3;
                posx4 = x4;
                posy3 = y3;
                posy4 = y4;
            end
        end
    end
    x3 = posx3;
    y3 = posy3;
    x4 = posx4;
    y4 = posy4;
    lebar = maks;

```

#### 4.1.2.5. Hitung Fitur Dispersi

Pada penghitungan fitur dispersi Chen dan dispersi maksimum minimum, yang menjadi masukan adalah kontur dan koordinat pusat x dan pusat y. Diawali dengan mencari jarak antar kontur dengan koordinat pusat. Kemudian dilakukan penghitungan dispersi I dan IR

##### **Listing Program 4.8 – Penghitungan Dispersi**

```

function [d1,d2] = dispersi(U, px, py)
rerata = 0;
terkecil = 9999999;
terbesar = 0;
jum_piksel = length(U);
for j=1:jum_piksel
    panjang = sqrt((U(j,1)-py)^2+(U(j,2)-px)^2);
    rerata = rerata+panjang;
    if panjang > terbesar
        terbesar = panjang;
    end
    if panjang < terkecil
        terkecil = panjang;
    end
end
a = luas(BW);
d1 = pi*terbesar/a;
d2 = terbesar / terkecil;

```

#### 4.1.2.6. Hitung Perimeter

Pada penghitungan perimeter, digunakan kode rantai sebagai masukan. Dimana kode masukan akan dibedakan antara kode genap dan ganjil. Setelah itu, dilakukan penghitungan perimeterya.

##### **Listing Program 4.9 – Pencarian Perimeter**

```
function hasilperim2 = perim2(kode_rantai)
    jum_genap = 0;
    jum_ganjil = 0;
    for p=1:length(kode_rantai)
        kode = kode_rantai(p);
        if (kode=='0') || (kode=='2') || (kode=='4') || (kode=='6') || (kode=='8')
            jum_genap = jum_genap+1;
        else
            jum_ganjil = jum_ganjil+1;
        end
    end
    hasilperim2 = jum_ganjil+jum_genap*sqrt(2);
```

#### 4.1.2.7. Hitung Luas

Pada penghitungan luas, digunakan pendekatan dengan penghitungan kode rantai. Sebagai masukan berupa kontur objek yang diambil ordinatnya, kode rantai. Penentuan penambahan atau pengurangan luas didasarkan kode masukan.

##### **Listing Program 4.10 – Pencarian Luas**

```
function hasilLuas2 = luas2(U,kode_rantai,BW)
    [tinggi, lebar] = size(BW);
    hasilLuas2 = 0;
    for p=1:length(kode_rantai)
        kode = kode_rantai(p);
        y = tinggi + 1 - U(p);
        switch kode
            case '0'
                hasilLuas2 = hasilLuas2+y;
            case '1'
                hasilLuas2 = hasilLuas2+y+0.5;
            case '2'
                hasilLuas2 = hasilLuas2;
            case '3'
                hasilLuas2 = hasilLuas2-y-0.5;
            case '4'
                hasilLuas2 = hasilLuas2-y;
            case '5'
                hasilLuas2 = hasilLuas2-y+0.5;
            case '6'
                hasilLuas2 = hasilLuas2;
            case '7'
                hasilLuas2 = hasilLuas2+y-0.5;
        end
    end
```

#### 4.1.2.8. Hitung Rasio Kerampingan

Pada tahap ini dilakukan penghitungan rasio kerampingan. Dalam fungsi ini digunakan lebar dan panjang objek kopi sebagai masukan dan rasio kerampingan dihitung dari perbandingan lebar dan panjang objek tersebut.

<b>Listing Program 4.11 – Hitung Rasio Kerampingan</b>
--

<pre>function [rasioKerampingan] = peroleh_kerampingan(lebar, panjang)     rasioKerampingan = lebar/panjang;</pre>
--

#### 4.1.2.9. Hitung Rasio Kebulatan

Pada tahap ini dilakukan penghitungan rasio kebulatan. Dalam fungsi ini digunakan perimeter dan luas objek kopi sebagai masukan dan rasio kebulatan dihitung dari perbandingan perimeter kuadrat dan luas objek tersebut.

<b>Listing Program 4.12 – Hitung Rasio Kebulatan</b>
--

<pre>function rasioKebulatan = peroleh_kebulatan(p,a)     rasioKebulatan = 4*pi*a/(p^2);</pre>
--

### 4.1.3. Training dan Identifikasi

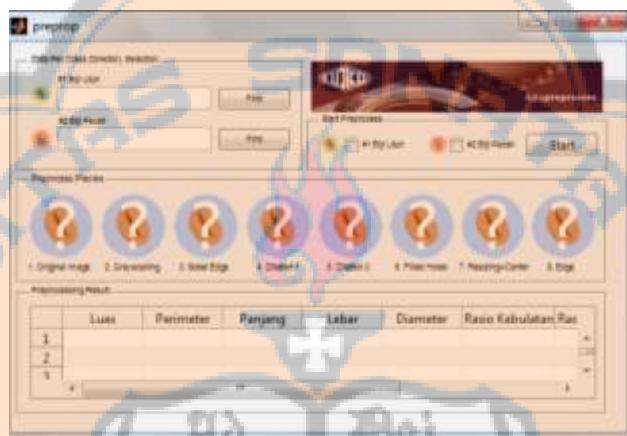
Dalam proses pelatihan jaringan saraf tiruan, sebagai masukan berupa *dataset* hasil ekstraksi ciri sebelumnya, dan pengaturan arsitektur yang akan digunakan dalam melakukan perancangan jaringan saraf tiruan. Proses meliputi deklarasi jaringan saraf tiruan yang akan digunakan (perintah ‘patternnet’), dengan jumlah *hidden layer* dan *hidden neuron* pada setiap *layer*nya, jumlah iterasi, dan pemilihan data *training-testing-validation*. Dalam proses ini jaringan saraf tiruan akan dilatih sebanyak ‘k’ masukan, dimana bobot dan bias akan disimpan dan diubah selama proses pelatihan, tetapi pemilihan data akan mengikuti pembagian seperti yang sudah dirancang pada 3.2.6. Setelah jaringan saraf tiruan terbentuk, akan dilakukan pengujian (*test*) dari kelompok data yang digunakan sebagai data uji kelompok. Nilai akurasi akan dihitung dari jumlah data benar dan jumlah data keseluruhan. Implementasi dapat dilihat pada listing program berikut.

### **Listing Program 4.13 – Proses Training dan Identifikasi**

#### **4.2. Desain Tampilan Sistem**

#### **4.2.1. Tampilan jendela pembuatan dataset (*Preprocessing* dan Ekstraksi Ciri)**

Dalam penelitian ini, yang pertama dilakukan adalah pembuatan *dataset* yang akan digunakan dalam proses *training*. Dalam tampilan jendela pertama ini, akan melakukan *preprocessing* data dan ekstraksi ciri.



**Gambar 4.1** – Tampilan awal untuk *preprocessing* dan ekstraksi ciri

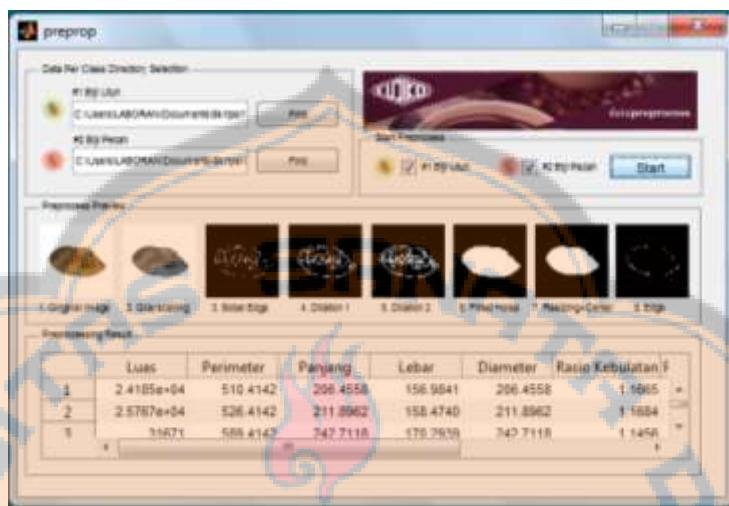
Yang pertama-tama yang dilakukan adalah memilih file citra biji kopi yang sudah dilakukan secara manual terlebih dahulu. Dalam tahap ini, pengguna akan memasukkan direktori folder yang berisi citra dari kelas biji utuh dan biji pecah.



**Gambar 4.2**– Pemilihan data untuk *preprocessing*

Setelah itu, pengguna akan mencentang data kelas mana saja yang akan di *preprocess* dan diekstraksi cirinya. Pada tahap ini file tiap kelas akan disimpan

dalam sebuah file .mat yang berupa struktur bernama ‘strBijidanCiriUtuh.mat’ dan ‘strBijidanCiriPecah.mat’ yang akan digunakan dalam proses selanjutnya dalam penyusunan arsitektur jaringan saraf tiruan.



Gambar 4.3 – Tampilan *preview* dan *dataset* yang dihasilkan.

#### 4.2.2. Tampilan jendela penyusunan arsitektur Jaringan Saraf Tiruan (*Training*)

Pada tampilan jendela ini, akan dilakukan penyusunan arsitektur jaringan saraf tiruan optimal yang akan digunakan.



Gambar 4.4 – Tampilan Jendela penyusunan jaringan saraf tiruan

Pada jendela ini, pengguna akan memasukkan *file* struktur dari tiap-tiap kelas (utuh dan pecah). *File* ini merupakan file *dataset* struktur yang dibuat dari proses yang dilakukan dari tampilan jendela sebelumnya (*preprocess*). File ‘strBijidanCiriUtuh.mat’ sebagai *file* struktur untuk kelas biji utuh, dan ‘strBijidanCiriPecah.mat’ sebagai *file* struktur untuk kelas biji pecah. .



**Gambar 4.5 –**file ‘strBijidanCiriUtuh.mat’ dan ‘strBijidanCiriPecah.mat’

Setelah itu, pengguna akan melakukan pengaturan terhadap arsitektur jaringan saraf tiruan yang akan digunakan. Pengaturan yang dapat dilakukan pengguna meliputi pemilihan fungsi *training*, jumlah *hidden layer*, jumlah *hidden neuron* pada *layer 1* dan *layer 2*, tipe penggunaan data mentah dan normalisasi, pemilihan k-value untuk pemilihan data *training-test-validation*, dan jumlah iterasi (*epoch*) proses *training*.



**Gambar 4.6 –**Tampilan perancangan Jaringan Saraf Tiruan

Setelah proses *training* dilakukan, jendela tampilan akan menampilkan data yang menjadi data *train-test-validation*, hasil simulasi dari jaringan saraf tiruan yang dibentuk, serta *confusion matrix*nya. Sehingga ditampilkan juga hasil jumlah klasifikasi yang benar, jumlah klasifikasi yang salah, serta akurasinya.



**Gambar 4.7 – Grafik peforma dan status pelatihan**

Setelah itu, sistem akan melakukan penyimpanan dari jaringan saraf tiruan yang telah dilatih, sesuai dengan parameter-parameter sesuai dengan konfigurasi arsitekturnya, dengan format .mat. Selain itu, sistem juga akan menyimpan beberapa diagram analisa yakni diagram peforma dari jaringan saraf tiruan yang telah dilatih, dan status *training* dalam tiap iterasinya.

#### 4.2.3. Tampilan jendela klasifikasi (*Test* tunggal menggunakan JST terbentuk)

Setelah jaringan saraf tiruan terbentuk, dilakukan pengujian simulasi terhadap data tunggal, dimana data tersebut tidak termasuk dalam kelompok data *train-test-validation*. Pengguna diminta untuk memasukkan file citra biji kopi yang akan diidentifikasi, serta file jaringan saraf tiruan yang telah dibuat sebelumnya.



**Gambar 4.8 – Tampilan jendela identifikasi tunggal**

Dalam jendela tampilan ini akan dilakukan *preprocessing* data serta ekstraksi ciri. Sebelum diidentifikasi, terdapat opsi untuk identifikasi data dari ciri yang didapatkan secara mentah, atau melalui proses normalisasi terlebih dahulu.



**Gambar 4.9** – Tampilan jendela identifikasi tunggal dan hasilnya

Hasil akhir berupa jenis kelas yang didapatkan, disertai dengan gambar, yakni citra tersebut masuk dalam kelas biji utuh atau pecah.

### 4.3. Contoh Hasil *Preprocessing* Data dan Ekstraksi Ciri

#### 4.3.1. *Preprocessing*

Berikut merupakan contoh dari hasil *preprocessing* sebuah citra biji kopi yang dimasukkan kedalam sistem. File citra yang digunakan adalah ‘pecah3.jpg’ yang berada didalam direktori sistem.

##### 4.3.1.1. *Grayscale*

Pada tahap ini, citra asli biji kopi masukan akan diubah kedalam citra dengan aras keabuan.

imageASLI                    imageGRAYSCALE



**Gambar 4.10** – Citra Asli (kiri), Citra Grayscale (kanan)

#### 4.3.1.2. Tepi Sobel

Tahap ini merupakan perubahan citra aras keabuan ke citra biner, dengan menggunakan deteksi tepi operator sobel



**Gambar 4.11 – Citra Grayscale (kiri), Citra Biner (kanan)**

#### 4.3.1.3. Dilasi

Tahap ini merupakan penegasan citra dari tepi yang dihasilkan dari operasi sebelumnya yakni deteksi tepi sobel. Pada tahap ini operasi dilasi dilakukan 2 kali, yakni dengan *structure element* garis vertical dan *structure element* garis horizontal



**Gambar 4.12 – Citra Biner (kiri), Dilasi Vertikal (tengah), Dilasi Horizontal (kanan)**

#### 4.3.1.4. *Image Filling*

Tahap ini digunakan untuk mengambil bentuk luar kontur dari citra biji kopi masukan, dimana operasi ini menggunakan prinsip pemanjangan piksel untuk menutup lubang yang terdapat di dalam area objek.



**Gambar 4.13 – Citra Biner (kiri), Hasil mengisi lubang objek (kanan)**

#### 4.3.1.5. *Centering & Resizing*

Tahap ini dilakukan untuk memindahkan citra objek biji kopi kedalam sebuah bidang baru, dimana objek akan di tempatkan tepat ditengah setelah sentroidnya diketahui. Citra biner inilah yang selanjutnya akan dilakukan proses ekstraksi ciri.



**Gambar 4.14** – Citra Biner (kiri), Hasil *centering* dan *resizing* (kanan)

#### 4.3.1.6. Kontur Tepi

Tahap ini dilakukan untuk menampilkan kontur bentuk dari citra biji kopi biner yang mengalami beberapa proses sebelumnya.



**Gambar 4.15** – Citra Biner (kiri), Piksel Kontur (kanan)

### 4.3.2. Ekstraksi Ciri

#### 4.3.2.1. Kontur

Berisi vektor mx2 yang menunjukkan pasangan koordinat penyusun kontur citra objek biji kopi. Pada contoh citra ‘pecah3.jpg’ didapatkan 347x2 (347 piksel)

**Tabel 4.1** – 15 koordinat kontur pertama ‘pecah3.jpg’

x	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	...
y	99	99	99	99	99	99	99	99	99	99	99	99	99	100	100	...

#### **4.3.2.2. Kode Rantai**

Berisi karakter yang menunjukkan arah koordinat penyusun kontur citra objek biji kopi. Pada contoh citra ‘pecah3.jpg’ 347 kode

**Tabel 4.2 –** kode rantai citra ‘pecah.3.jpg’

### 4.3.2.3. Luas

Berisi angka yang menunjukkan luas dari objek biji kopi. Pada contoh citra ‘pecah3.jpg’ 9841 unit

### 4.3.2.4. Perimeter

Berisi angka yang menunjukkan keliling dari objek biji kopi. Pada contoh citra ‘pecah3.jpg’ 347,4142 unit

#### **4.3.2.5. Panjang, Lebar, Diameter**

Berisi angka yang menunjukkan panjang, lebar, dan diameter objek biji kopi. Pada contoh citra ‘pecah3.jpg’ 153,9415 units panjang (diameter) dan 88,0057 units lebar.

#### **4.3.2.6. Rasio Kebulatan**

Berisi angka yang menunjukkan rasio kebulatan objek biji kopi. Pada contoh citra ‘pecah3.jpg’ 1,0246

#### **4.3.2.7. Rasio Kerampingan**

Berisi angka yang menunjukkan rasio kerampingan objek biji kopi. Pada contoh citra ‘pecah3.jpg’ 0.5717

#### **4.3.2.8. Dispersi I dan IR**

Berisi angka yang menunjukkan dispersi chen dan disperse maksimum-minimum objek biji kopi. Pada contoh citra ‘pecah3.jpg’ 0.0269 dispersi I, dan 2.1178 dispersi IR.

#### **4.4. Uji Coba dan Analisa Penggunaan Alat Uji sesuai dengan parameter-parameter untuk mencari jaringan optimal**

Hasil penelitian ini berupa nilai akurasi dari pengenalan jenis bentuk biji kopi., dengan pelatihan sejumlah 113 data citra. Akan dilakukan pengujian dengan beberapa parameter dengan tujuan untuk mencari jaringan saraf tiruan yang paling optimal dalam melakukan pengenalan bentuk biji kopi. Adapun parameter yang diubah adalah nilai, jenis penggunaan data, *k-fold*, fungsi *training*, dan jumlah *neuron* pada *hidden layer* tertentu. Dalam pemilihan jumlah akan diberikan nilai N dimana  $5 \leq N \leq 25$  dan nilai N akan bertambah 5 setiap percobaanya. Dalam pemilihan *k-fold*, dibatasi menjadi 3 dan 5 fold saja.

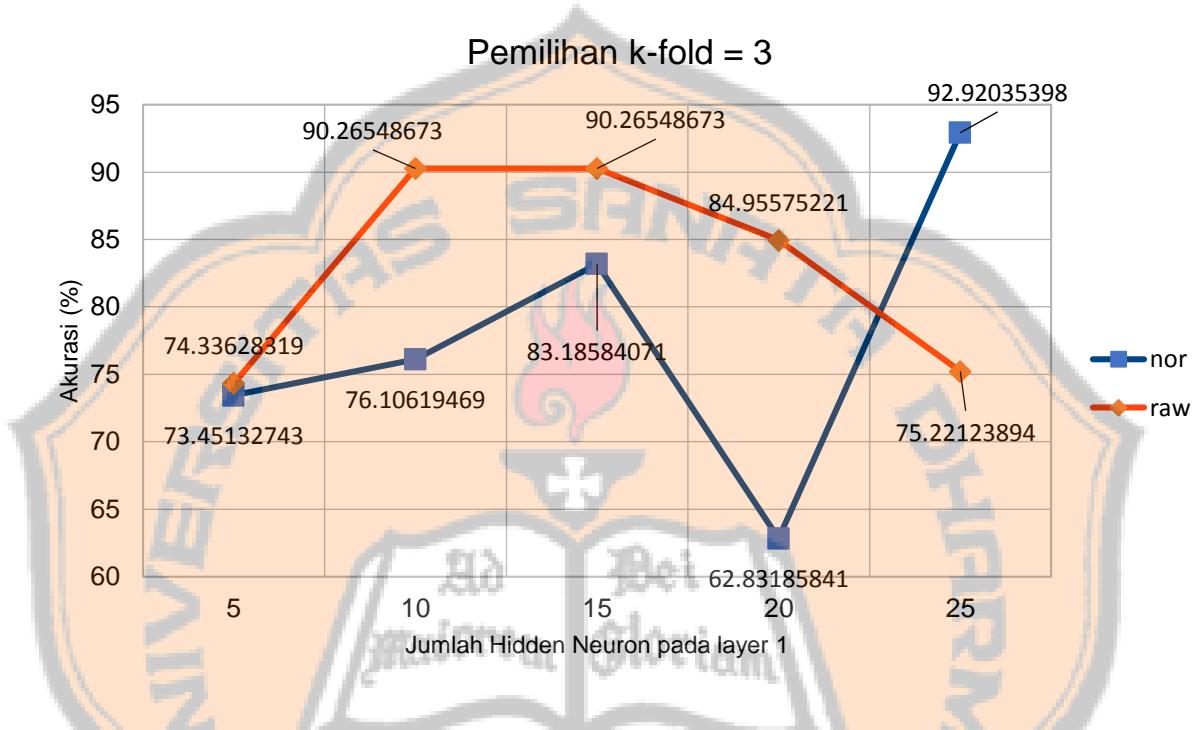
##### **4.4.1. Pengaturan arsitektur awal**

Dalam tahap awal, ditentukan arsitektur default dengan konfigurasi awal sebagai berikut

- Fungsi transfer *hidden layer* 1 adalah tansig
- Fungsi aktivasi softmax
- Fungsi *training* traingdx
- Batas iterasi training sebesar 100

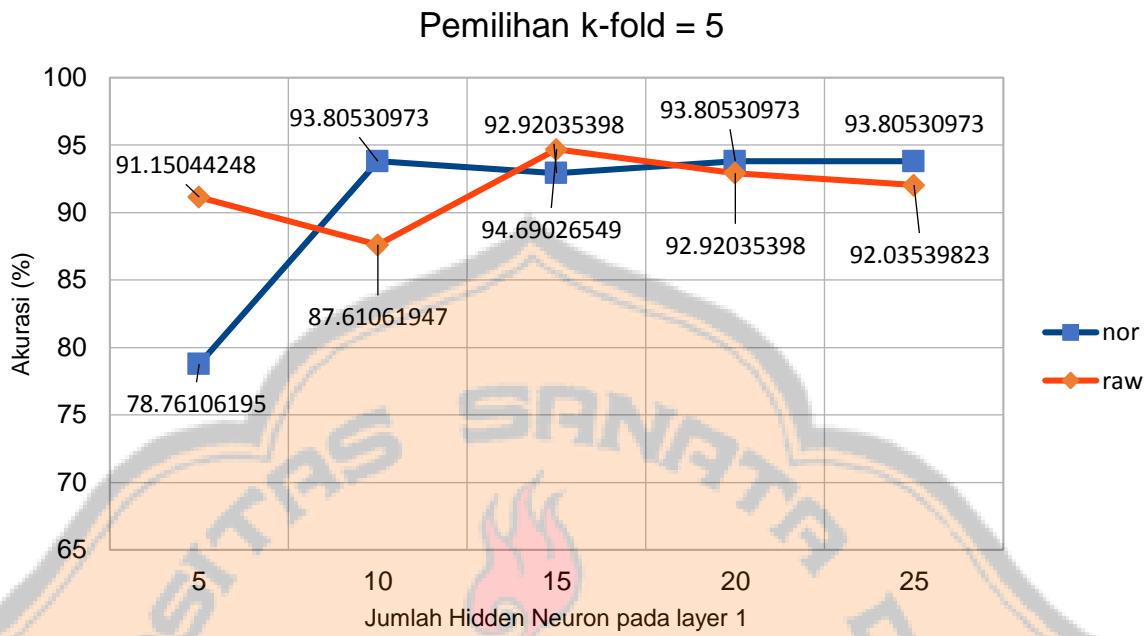
#### 4.4.2. Pencarian jaringan optimal : Jumlah *Neuron* pada *Hidden Layer* Pertama

##### 4.4.2.1. Pemilihan jenis penggunaan data, K-fold dan jumlah hidden neuron



**Gambar 4.16 – Uji 1 :** Hasil akurasi identifikasi dengan arsitektur: jumlah *hidden neuron* pada *hidden layer* ke-1 dengan *k value* = 3 pada data yang dinormalisasi dan data mentah

Dengan menggunakan pengaturan dari arsitektur awal, pada tahap ini akan digunakan beberapa kombinasi dari arsitektur jaringan saraf tiruan dengan perubahan jumlah *hidden neuron* pada *hidden layer* pertama dan *k-fold* data yang digunakan. Sebagai perbandingan digunakan *dataset* yang mengalami proses normalisasi dengan min-max dan dataset ciri mentah yang langsung dimasukkan dalam proses training jaringan saraf tiruan. Pada hasil uji pertama, baik penggunaan data mentah (*raw*) maupun dataset yang dinormalisasi (*nor*) memiliki tingkat akurasi yang bervariasi. Seperti pada gambar 4.16, saat digunakan 10, 15 dan 25 dan *k-value* nya 3, akurasinya mencapai 90.2% pada data mentah, 90.2% pada data mentah, dan 92.9% pada *dataset* yang mengalami normalisasi.

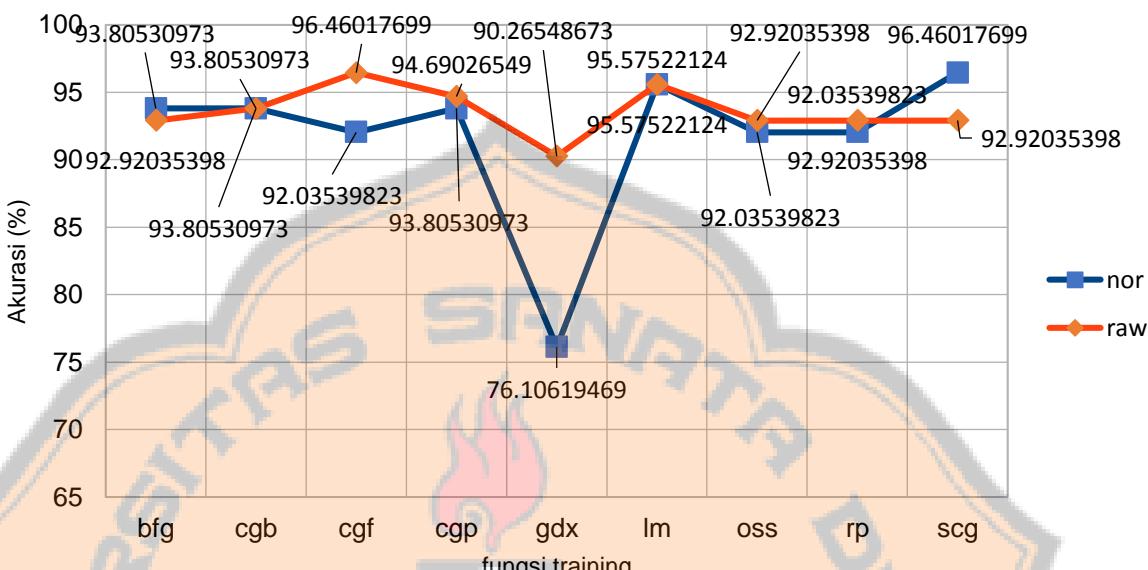


**Gambar 4.17 – Uji 2 :** Hasil akurasi identifikasi dengan arsitektur: jumlah *hidden neuron* pada *hidden layer* ke-1 dengan *k value* = 5 pada data yang dinormalisasi dan data mentah

Sedangkan jika digunakan *k-value* 5 untuk pemilihan data *train-test-validation*, hasil akurasi tertinggi diraih saat arsitektur menggunakan *hidden neuron* 20 dengan 94.7% pada data hasil normalisasi, dan 94.7% saat menggunakan *hidden neuron* 15 pada data mentah (Gambar 4.17). Sehingga dalam proses selanjutnya dipilih arsitektur dengan jumlah 10, 15, 20, dan 25 untuk diuji kembali menggunakan pemilihan *k-value* dan beberapa fungsi *training* yang berbeda untuk mendapatkan jaringan yang lebih optimal.

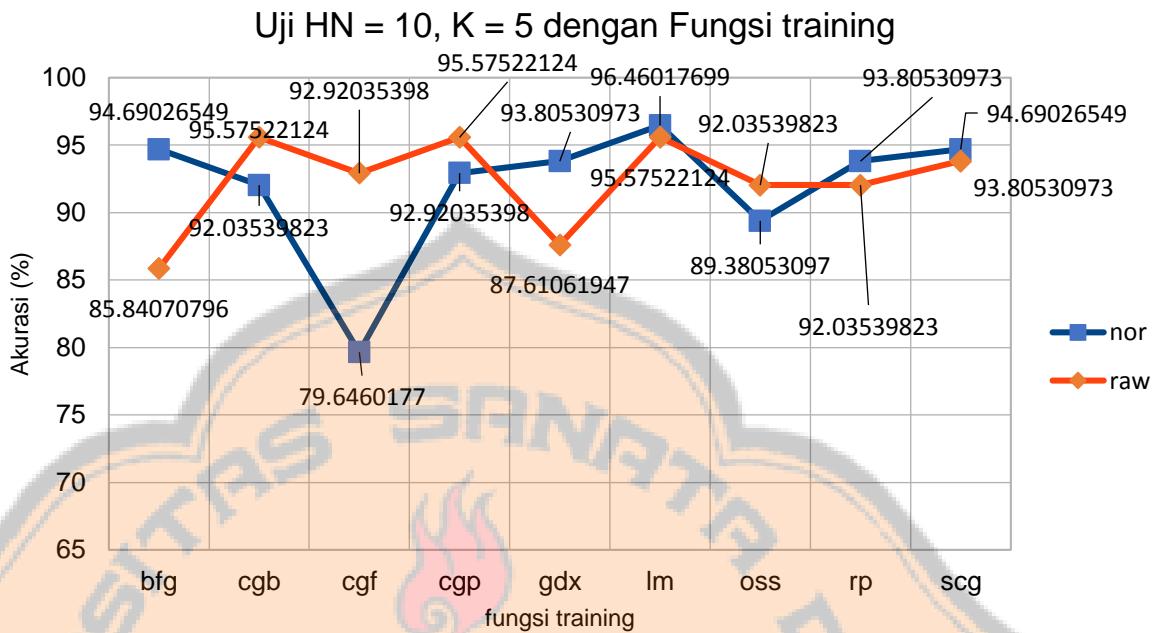
#### 4.4.2.2. Pemilihan fungsi *training*, *k-fold*, dan **10 15 20 25 hidden neuron**

Uji  $HN = 10$ ,  $K = 3$  dengan Fungsi *training*



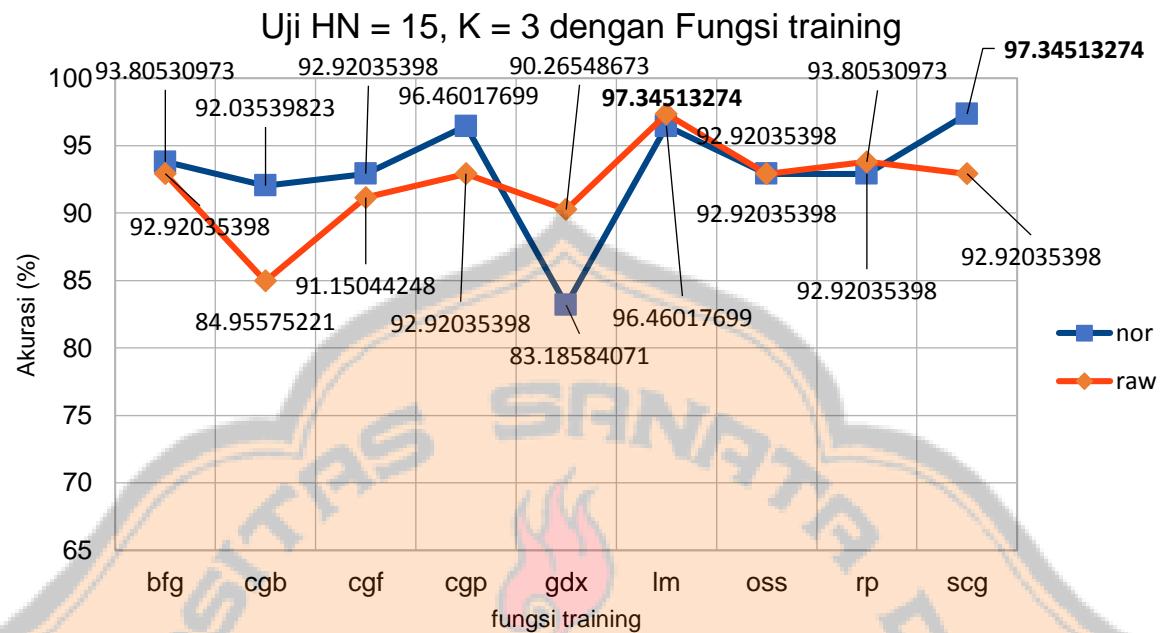
**Gambar 4.18** – Uji 3 : Hasil akurasi identifikasi dengan arsitektur: 10 *Hidden Neuron*,  $K=3$ , dan fungsi *training* pada data hasil normalisasi dan data mentah

Pada gambar 4.18, ditampilkan hasil akurasi identifikasi dengan menggunakan *hidden neuron* 10 dan *k-fold* 3. Dapat dilihat hasil tertinggi diraih oleh fungsi traincfg (*Fletcher-Reeves updates*) pada data mentah sebesar 96.5%, sedangkan pada penggunaan data hasil normalisasi, hasil tertinggi diraih pada fungsi trainscg (*Scaled conjugate gradient backpropagation*) dengan tingkat akurasi 96.5%



**Gambar 4.19 – Uji 4 : Hasil akurasi identifikasi dengan arsitektur: 10 Hidden Neuron, K=5, dan fungsi training pada data hasil normalisasi dan mentah**

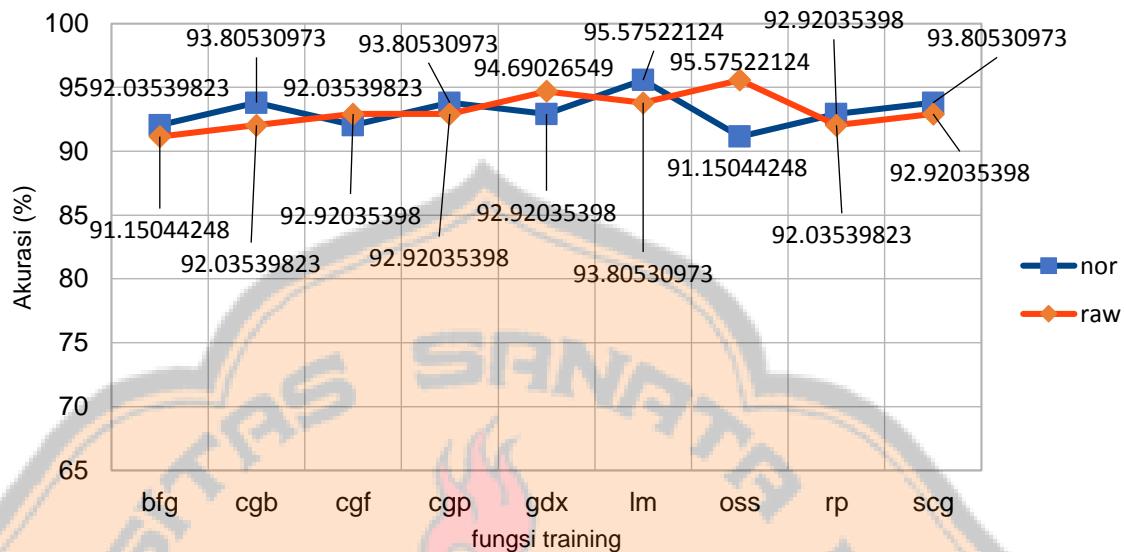
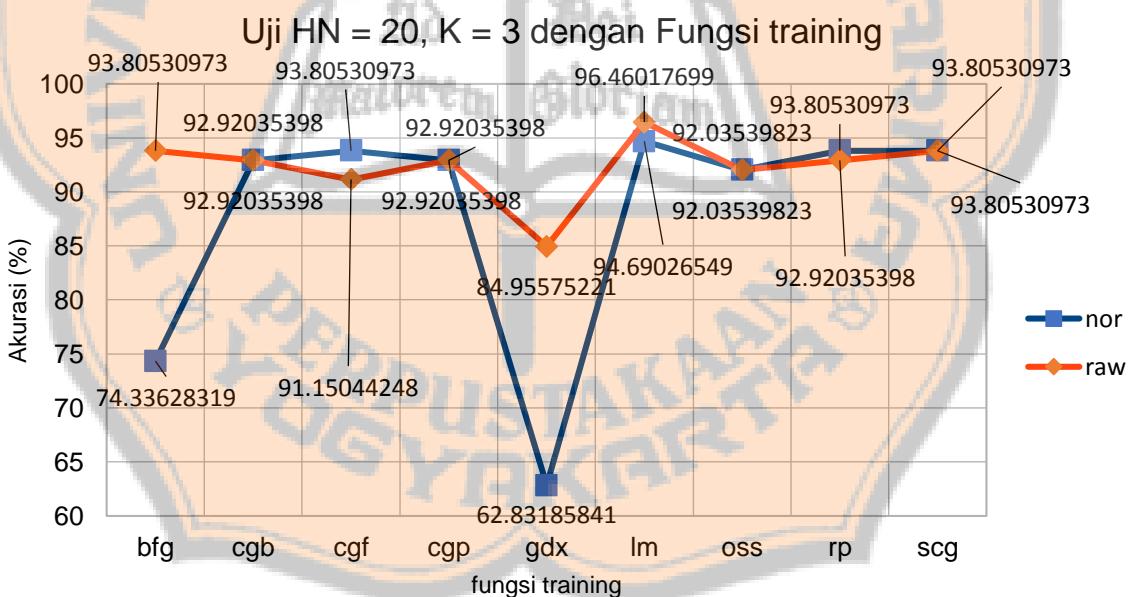
Pada gambar 4.19, ditampilkan hasil akurasi identifikasi dengan menggunakan *hidden neuron* 10 dan *k-fold* 5. Dapat dilihat hasil tertinggi diraih oleh fungsi *trainlm* (*Lavenberg-Marquardt backpropagation*) dengan akurasi sebesar 96.46% dimana data yang digunakan adalah dataset hasil normalisasi.



**Gambar 4.20 – Uji 5 : Hasil akurasi identifikasi dengan arsitektur: 15 Hidden Neuron, K=3, dan fungsi training pada data hasil normalisasi dan data mentah**

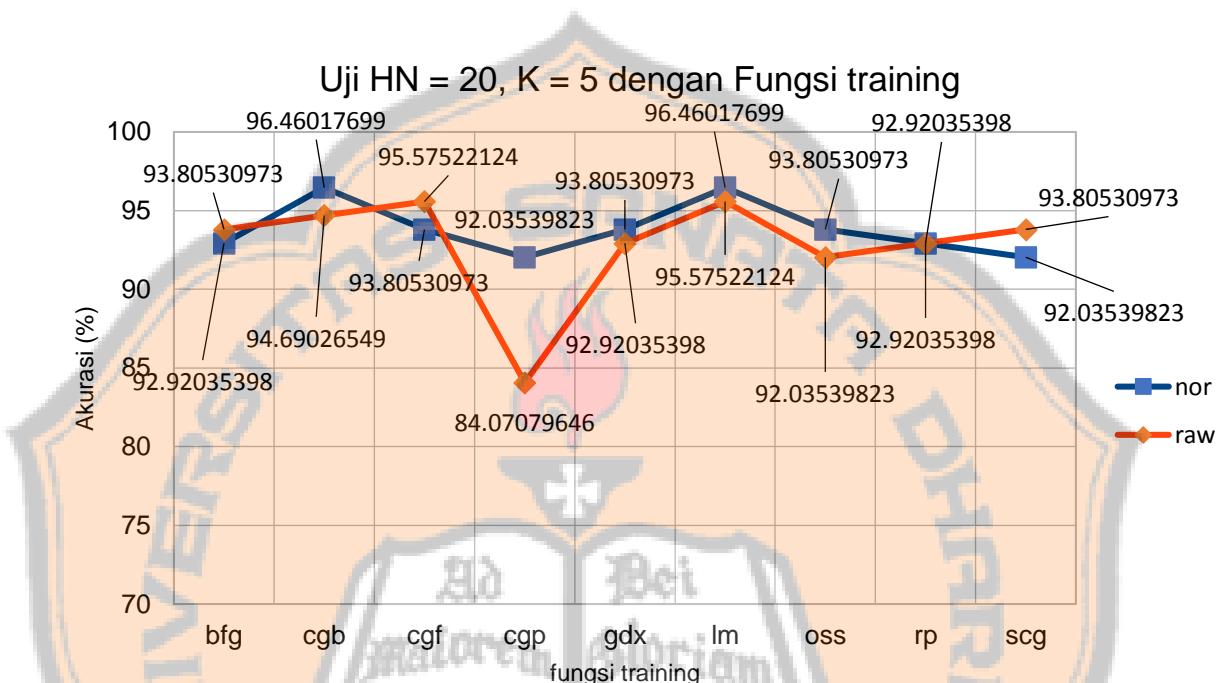
Pada gambar 4.20, ditampilkan hasil akurasi identifikasi dengan menggunakan *hidden neuron* 15 dan *k-fold* 3. Dapat dilihat hasil tertinggi diraih oleh fungsi trainlm (*Lavenberg-Marquardt backpropagation*) dengan akurasi sebesar 97.34% dimana data yang digunakan adalah *dataset mentah*. Sedangkan pada data normalisasi, hasil tertinggi terletak pada fungsi trainscg (*Scaled conjugate gradient backpropagation*) sebesar 97.34%.

Sedangkan saat digunakan nilai *k-foldnya* 5, hasil identifikasi terbaik didapatkan saat menggunakan fungsi trainlm (*Lavenberg-Marquardt backpropagation*) dengan hasil akurasi sebesar 95.57% pada data hasil normalisasi. Hal ini dapat dilihat pada gambar 4.21.

Uji  $HN = 15, K = 5$  dengan Fungsi trainingGambar 4.21 – Uji 6 : Hasil akurasi identifikasi dengan arsitektur: 15 Hidden Neuron, K=5, dan fungsi *training* pada data hasil normalisasi dan data mentahGambar 4.22 – Uji 7 : Hasil akurasi identifikasi dengan arsitektur: 20 Hidden Neuron, K=3, dan fungsi *training* pada data hasil normalisasi dan data mentah

Pada gambar 4.22, ditampilkan hasil akurasi identifikasi dengan menggunakan *hidden neuron* 20 dan *k-fold* 3. Dapat dilihat hasil tertinggi diraih oleh fungsi *trainlm* (*Lavenberg-Marquardt backpropagation*) dengan akurasi sebesar 96.46% dimana data yang digunakan adalah *dataset* mentah.

Sedangkan saat digunakan nilai  $k$ -*fold*nya 5, hasil identifikasi terbaik didapatkan saat menggunakan fungsi trainlm (*Lavenberg-Marquardt backpropagation*) dengan hasil akurasi sebesar 96.46% pada data hasil normalisasi. Hal ini dapat dilihat pada gambar 4.23.

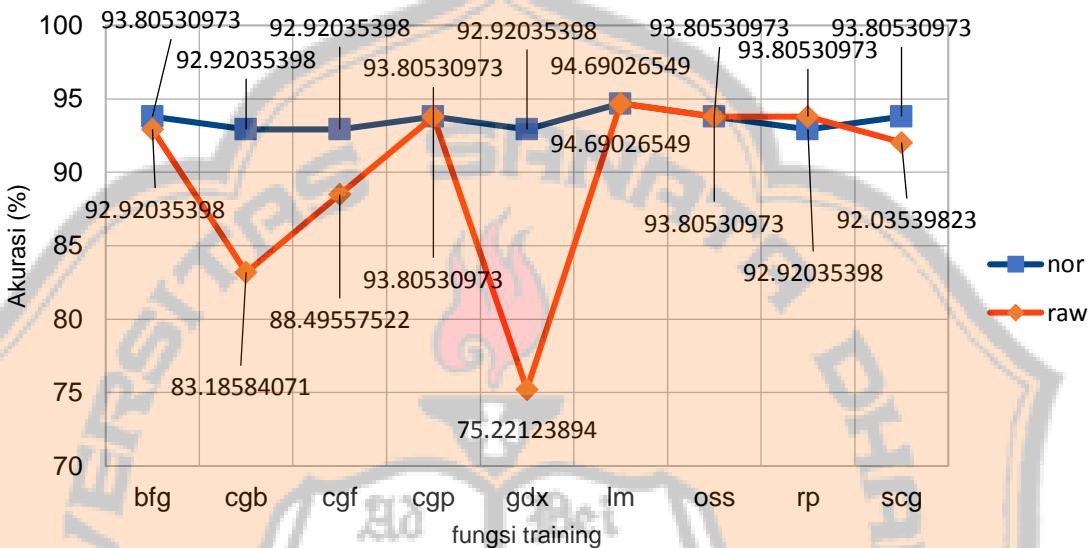


**Gambar 4.23 – Uji 8 :** Hasil akurasi identifikasi dengan arsitektur: 20 *Hidden Neuron*, K=5, dan fungsi *training* pada data hasil normalisasi dan data mentah

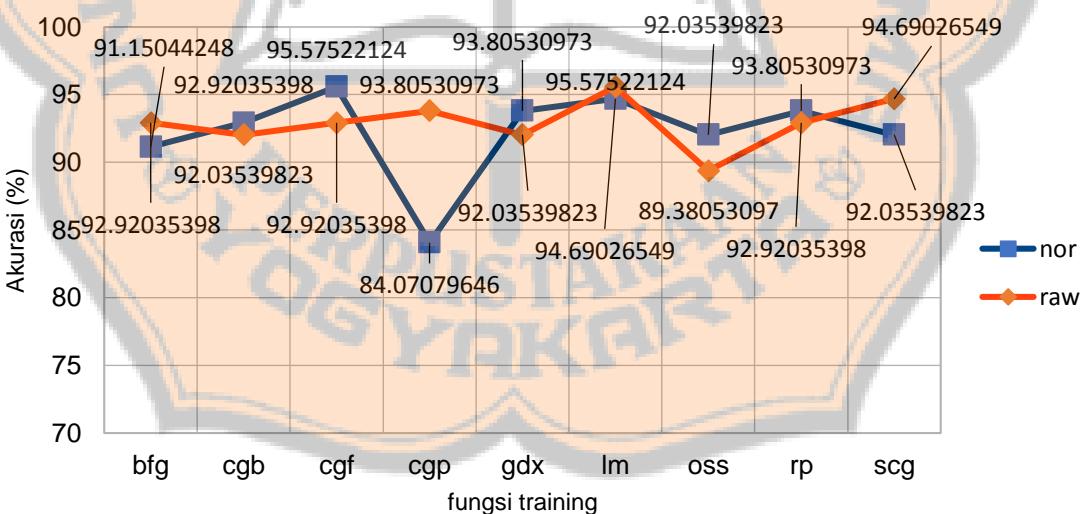
Pada gambar 4.24 , ditampilkan hasil akurasi identifikasi dengan menggunakan *hidden neuron* 25 dan  $k$ -*fold* 3. Dapat dilihat hasil tertinggi diraih oleh fungsi trainlm (*Lavenberg-Marquardt backpropagation*) dengan akurasi sebesar 96.46% dimana data yang digunakan adalah *dataset* mentah dan *dataset* hasil normalisasi.

Sedangkan saat digunakan nilai  $k$ -foldnya 5, hasil identifikasi terbaik didapatkan saat menggunakan fungsi traincfg (*Fletcher-Reeves updates*) dengan hasil akurasi sebesar 95.57% pada data hasil normalisasi dan 95.57% pada data mentah. Hal ini dapat dilihat pada gambar 4.24.

**Uji  $HN = 25, K = 3$  dengan Fungsi training**



**Uji  $HN = 25, K = 5$  dengan Fungsi training**

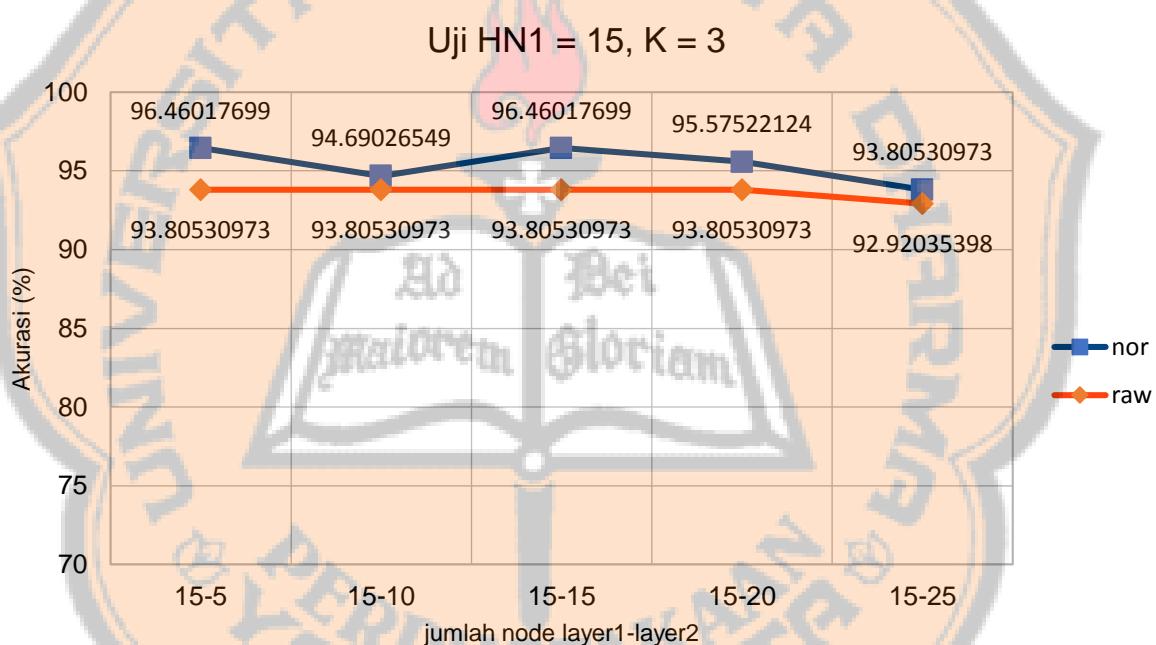


**Gambar 4.24 – Uji 9 dan 10:** Hasil akurasi identifikasi dengan arsitektur: 25 *Hidden Neuron*,  $K=3$  , dan fungsi *training* pada data hasil normalisasi dan data mentah

Dari beberapa percobaan sebelumnya, akurasi tertinggi dicapai pada arsitektur yang menggunakan 15 *hidden neuron* (97.34%). Dan fungsi *training* yang paling banyak memberikan hasil maksimum adalah fungsi *trainlm* (*Lavenberg-Marquardt backpropagation*). Sehingga konfigurasi ini akan digunakan pada tahap selanjutnya, yakni mencari *jumlah neuron* pada *hidden layer* ke-2 dengan tujuan pencarian jaringan saraf tiruan yang paling optimal dalam melakukan identifikasi.

#### **4.4.3. Pencarian jaringan optimal : Jumlah Neuron pada Hidden Layer kedua**

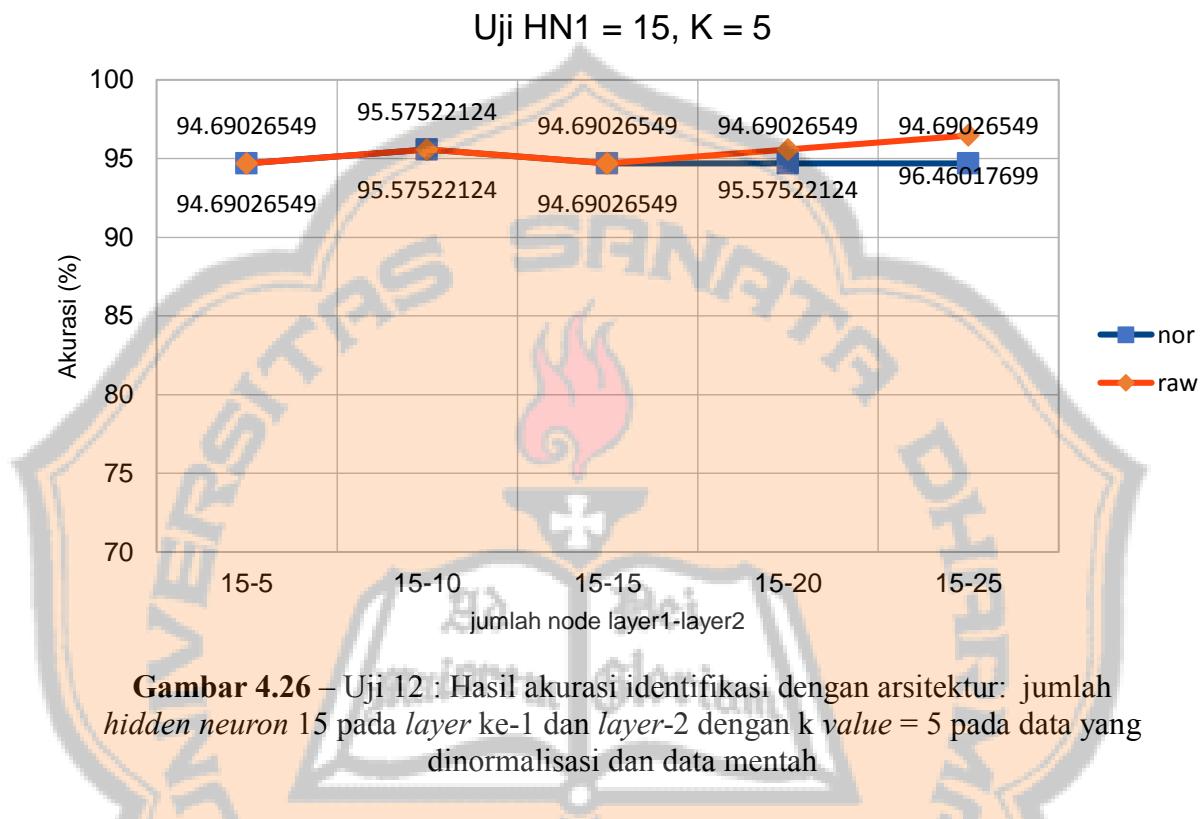
##### **4.4.3.1. Pemilihan jenis penggunaan data, K-fold, dan jumlah hidden neuron**



**Gambar 4.25 – Uji 11 : Hasil akurasi identifikasi dengan arsitektur: jumlah hidden neuron 15 pada layer ke-1 dan layer-2 dengan k value = 3 pada data yang dinormalisasi dan data mentah**

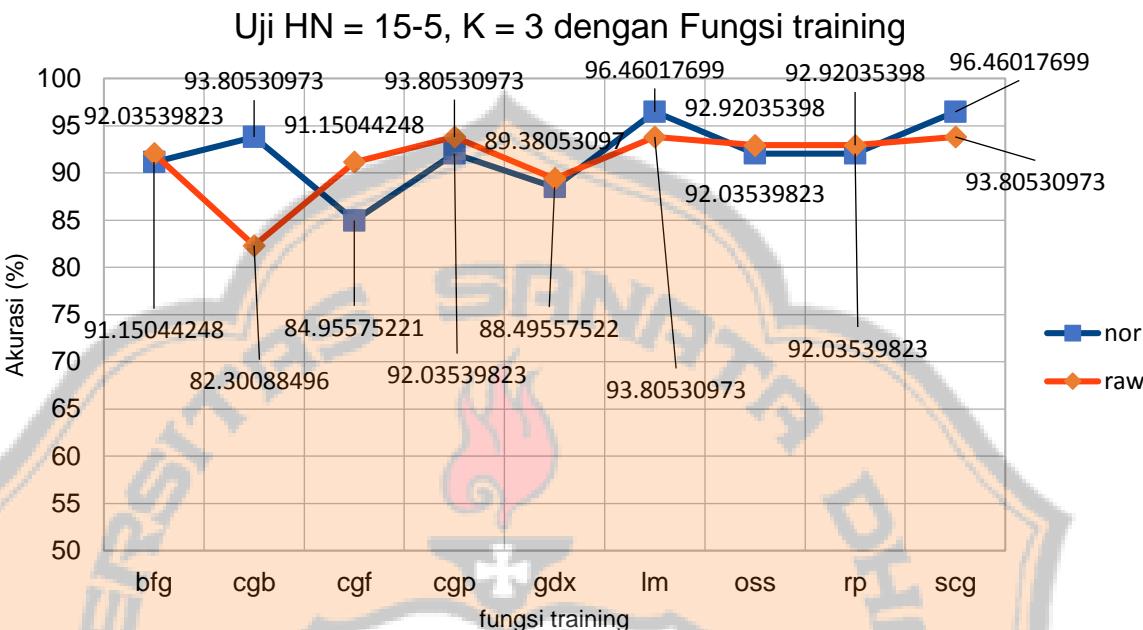
Penentuan jumlah *neuron* pada *layer* kedua akan dilakukan seperti sebelumnya, dimana didapatkan jumlah *neuron* pada *layer* pertama adalah 15, dan fungsi *training* *trainlm* (*Lavenberg-Marquardt backpropagation*). Dari gambar 4.25, hasil akurasi terbesar didapatkan pada arsitektur dengan jumlah *neuron* kedua 5 dan 15 dengan nilai 96.46% dan kedua akurasi tersebut didapatkan dari penggunaan data hasil normalisasi.

Di sisi lain, jika menggunakan *hidden neuron* pertama 15 dan *k-fold* nya 5, hasil akurasi terbesar didapatkan saat menggunakan 25 *hidden neuron* pada *layer* kedua, dengan nilai 96.46% dengan menggunakan data yang tidak dinormalisasi,



Dari dua percobaan sebelumnya, akan digunakan *hidden neuron* 5, 15, dan 25 dalam pengujian selanjutnya, dikarenakan memiliki nilai akurasi yang sama yakni 96.46%. Maka selanjutnya akan dilakukan pengujian kembali dengan *hidden neuron* layer pertama 15, kemudian disertai dengan pemilihan nilai *k-fold* dan fungsi *training* untuk mencari jaringan yang paling optimal dalam melakukan identifikasi.

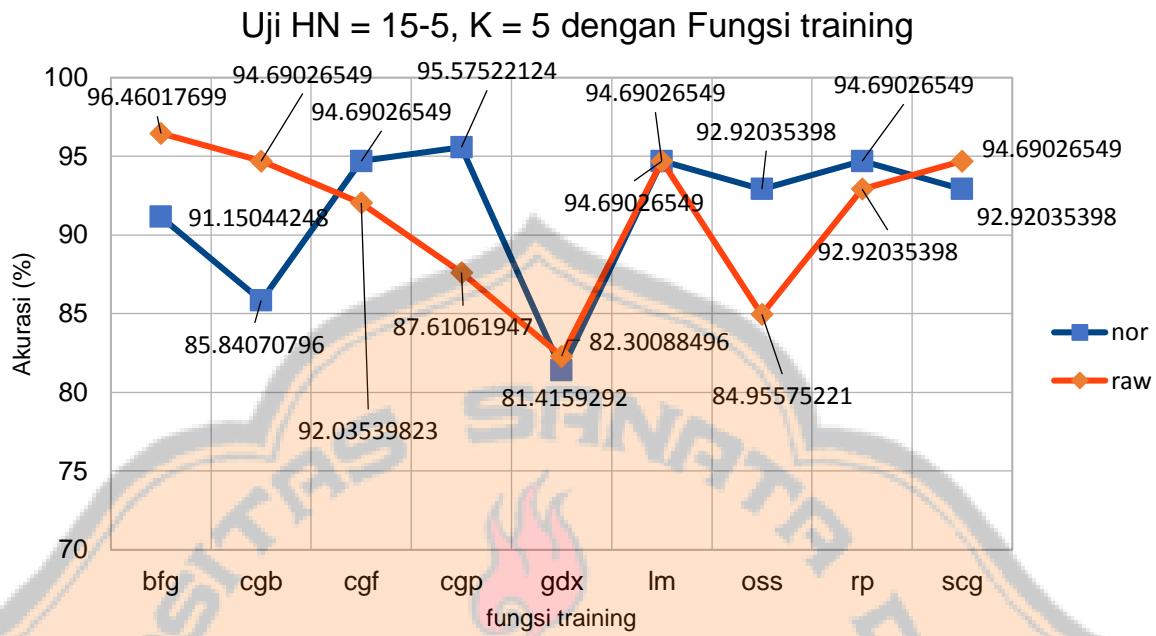
#### 4.4.3.2. Pemilihan fungsi *training* dengan *k-fold* pada *hidden neuron* 5, 15 dan 25



**Gambar 4.27 – Uji 13 : Hasil akurasi identifikasi dengan arsitektur: 15 *neuron* pada layer ke-1 dan 5 *neuron* pada layer-2 dengan *k value* =3 pada data yang dinormalisasi dan data mentah**

Pada gambar 4.27 ditampilkan hasil akurasi identifikasi yang menggunakan 15 *neuron* pada *hidden layer* 1, 5 *neuron* pada *hidden layer* 2, *k fold* bernilai 3 dan fungsi *training*nya. Dapat dilihat akurasi terbesar terletak pada trainlm (*Lavenberg-Marquardt backpropagation*) yakni 96.46% dan trainscg (*Scaled conjugate gradient backpropagation*) yang mencapai 96.46%. Kedua fungsi *training* ini mencapai akurasi terbesar pada penggunaan data hasil normalisasi.

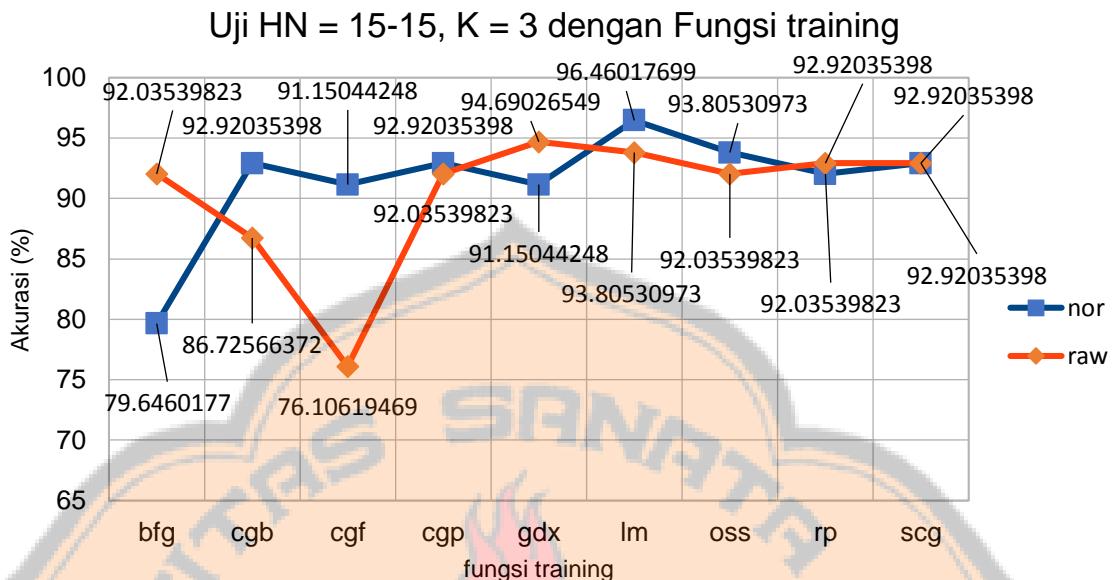
Sedangkan pada gambar 4.28 ditampilkan hasil akurasi identifikasi yang menggunakan 15 *neuron* pada *hidden layer* 1, 5 *neuron* pada *hidden layer* 2, *k fold* bernilai 5 dan fungsi *training*nya. Dapat dilihat akurasi terbesar terletak pada trainbfg (*BFGS quasi-Newton Backpropagation*) dimana data yang digunakan merupakan data mentah yang tidak dinormalisasi.



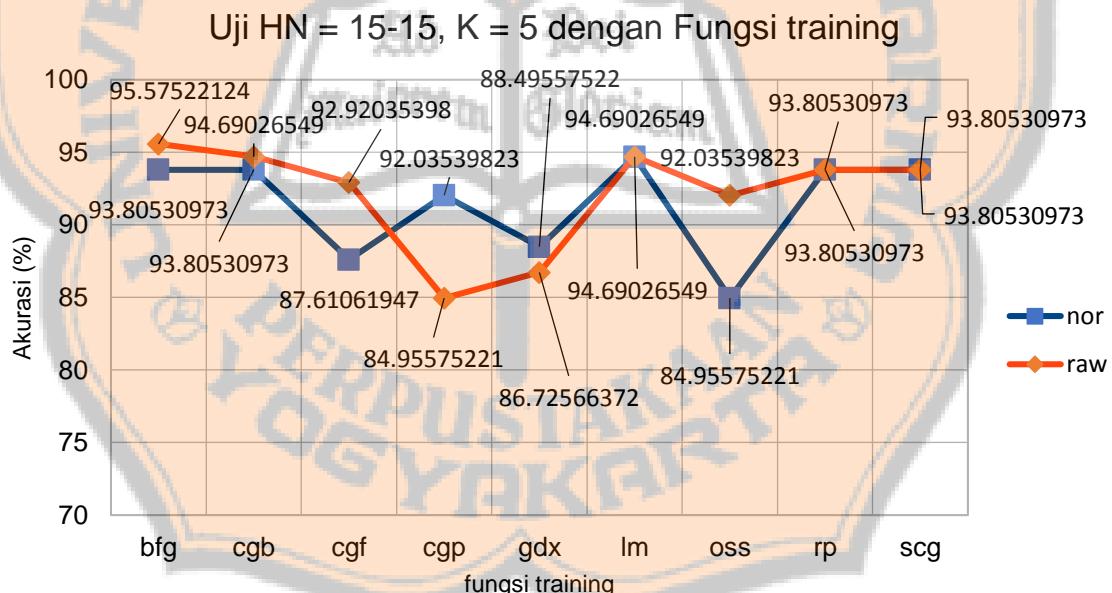
**Gambar 4.28 – Uji 14 : Hasil akurasi identifikasi dengan arsitektur: 15 neuron pada layer ke-1 dan 5 neuron pada layer-2 dengan k value =5 pada data yang dinormalisasi dan data mentah**

Pada gambar 4.29 ditampilkan hasil akurasi identifikasi yang menggunakan 15 neuron pada hidden layer 1, 15 neuron pada hidden layer 2, k fold bernilai 3 dan fungsi trainingnya. Dapat dilihat akurasi terbesar terletak pada trainlm (*Lavenberg-Marquardt backpropagation*) yakni 96.46% dengan menggunakan data hasil normalisasi.

Sedangkan pada gambar 4.30 ditampilkan hasil akurasi identifikasi yang menggunakan 15 neuron pada hidden layer 1, 15 neuron pada hidden layer 2, k fold bernilai 5 dan fungsi trainingnya. Dapat dilihat akurasi terbesar terletak pada trainbfg (*BFGS quasi-Newton Backpropagation*) dimana data yang digunakan merupakan data mentah yang tidak dinormalisasi.



**Gambar 4.29** – Uji 15: Hasil akurasi identifikasi dengan arsitektur: 15 *neuron* pada *layer* ke-1 dan 15 *neuron* pada *layer*-2 dengan *k value* = 3 pada data yang dinormalisasi dan data mentah

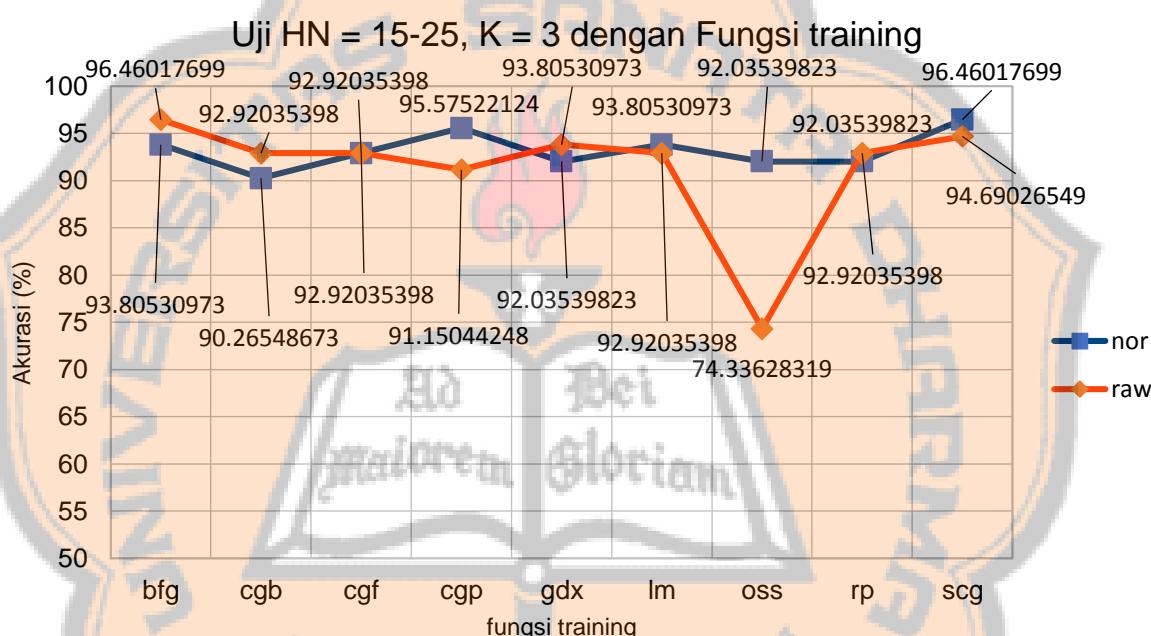


**Gambar 4.30** – Uji 16: Hasil akurasi identifikasi dengan arsitektur: 15 *neuron* pada *layer* ke-1 dan 15 *neuron* pada *layer*-2 dengan *k value* = 5 pada data yang dinormalisasi dan data mentah

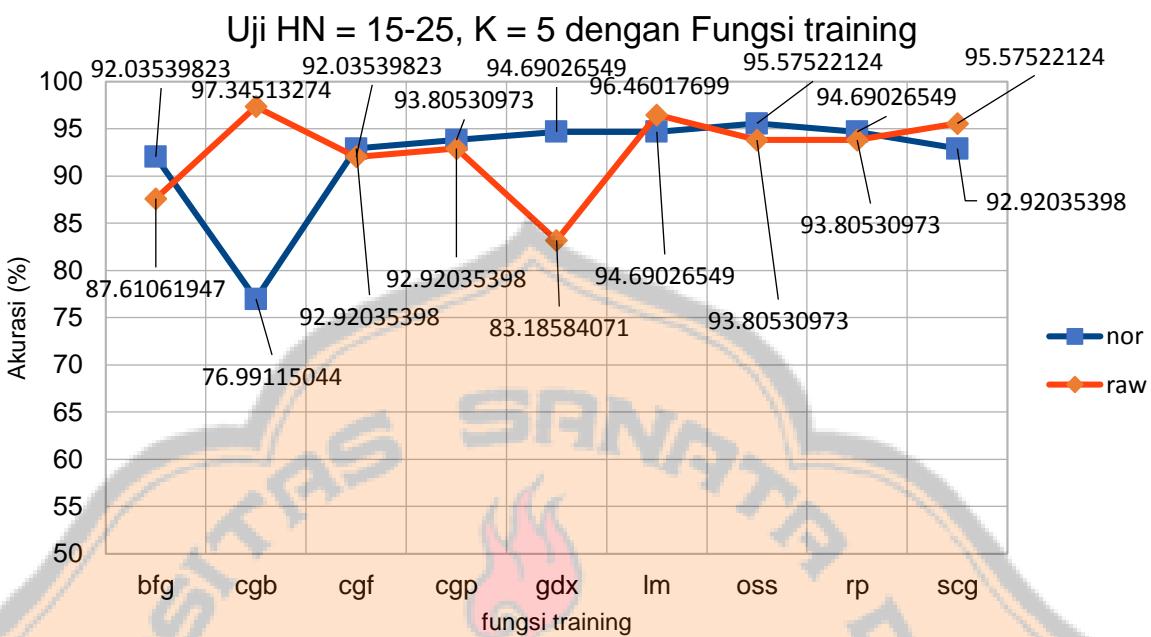
Pada gambar 4.31 ditampilkan hasil akurasi identifikasi yang menggunakan 15 *neuron* pada *hidden layer 1*, 25 *neuron* pada *hidden layer 2*, *k fold* bernilai 3 dan fungsi *trainingnya*. Dapat dilihat akurasi terbesar terletak pada *trainscg* (*Scalable*

*Conjugate gradient backpropagation) yakni 96.46% dengan menggunakan data hasil normalisasi.*

Sedangkan pada gambar 4.32 ditampilkan hasil akurasi identifikasi yang menggunakan 15 *neuron* pada *hidden layer 1*, 25 *neuron* pada *hidden layer 2*, *k fold* bernilai 5 dan fungsi *trainingnya*. Dapat dilihat akurasi terbesar terletak pada traincgb (*Powell-Beale restarts*) dimana data yang digunakan merupakan data mentah yang tidak dinormalisasi.



**Gambar 4.31 – Uji 17 :** Hasil akurasi identifikasi dengan arsitektur: 15 *neuron* pada *layer ke-1* dan 25 *neuron* pada *layer-2* dengan *k value* = 3 pada data yang dinormalisasi dan data mentah



**Gambar 4.32 – Uji 18 : Hasil akurasi identifikasi dengan arsitektur: 15 neuron pada layer ke-1 dan 25 neuron pada layer-2 dengan k value = 5 pada data yang dinormalisasi dan data mentah**

#### 4.5. Uji Coba data Tunggal

Setelah dilakukan pelatihan pada arsitektur jaringan saraf tiruan, selanjutnya akan dilakukan pengujian terhadap data tunggal yang tidak termasuk dalam proses pelatihan atau pengujian data kelompok. Data uji untuk masing-masing jenis kelas berjumlah 3 buah file citra, dimana setiap citra akan dilakukan uji identifikasi dengan jaringan saraf tiruan yang sudah dibuat sebelumnya. Adapun hasilnya dapat dilihat pada tabel berikut ini

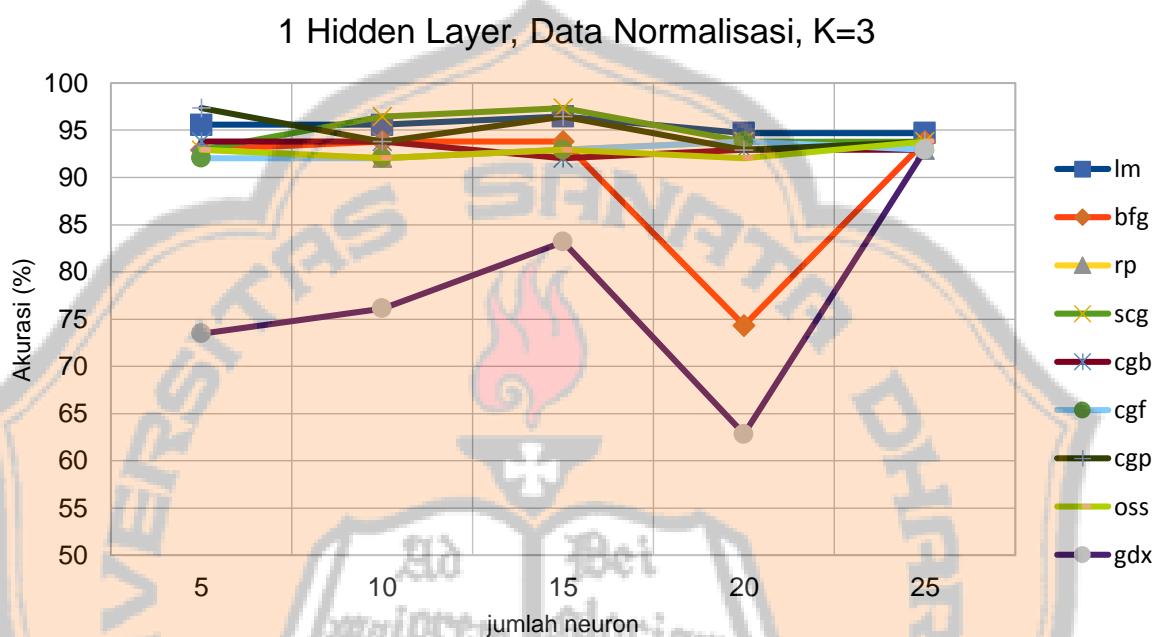
**Tabel 4.3 – Hasil Uji Data Tunggal**

No	File Citra Biji Kopi	Hasil
1	3 (tiga) citra biji kopi utuh	Benar semua
2	3 (tiga) citra biji kopi pecah	Benar semua

#### 4.6. Analisa Hasil

Dari beberapa hasil percobaan yang telah dilakukan sebelumnya, hasil akurasi yang dicapai dari beberapa arsitektur jaringan saraf tiruan cukup bervariasi.

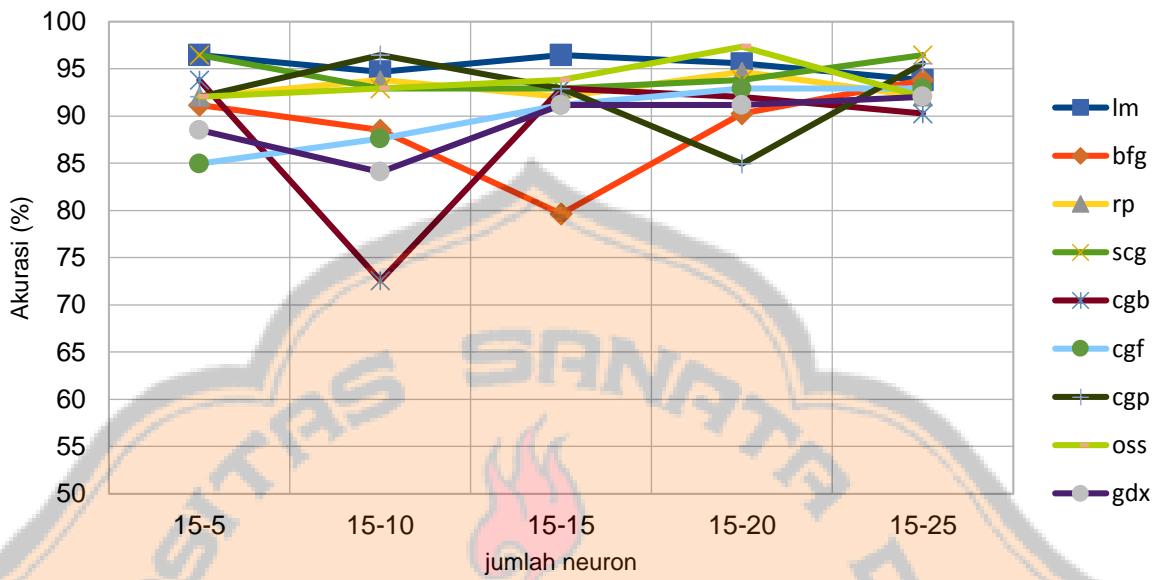
Pada saat penggunaan 1 *hidden layer*, akurasi tertinggi dicapai saat penggunaan fungsi *training* traincgp dengan 5 *hidden neuron* nya, dan trainscg dengan 15 *hidden neuron*nya pada penggunaan data normalisasi yakni 97.35% (3 misklasifikasi dari 113 data).



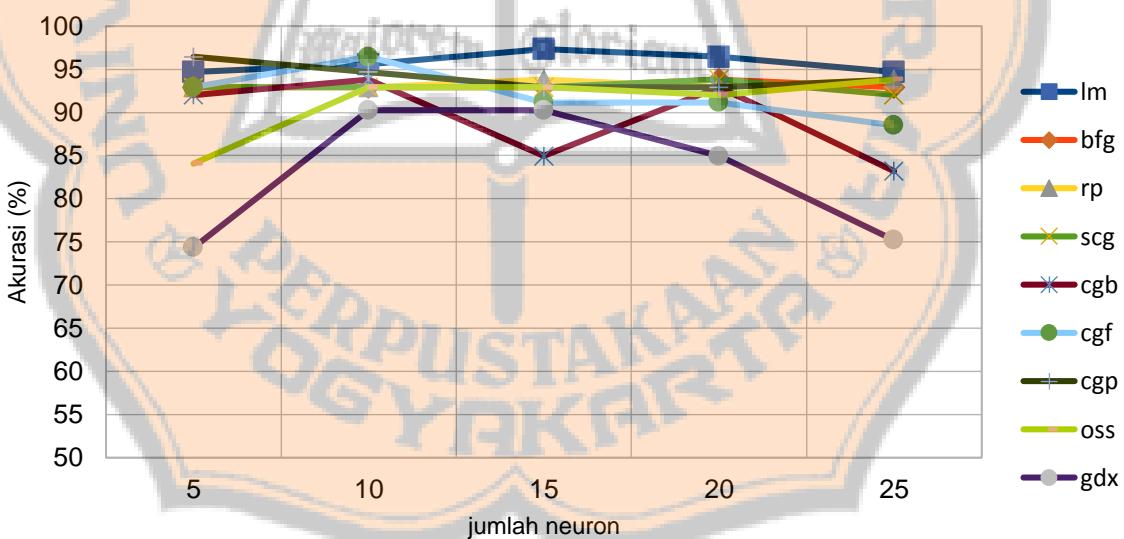
Gambar 4.33 – Akurasi yang dicapai tiap fungsi *training* pada pemilihan jumlah *neuron* pada *layer* pertama dan penggunaan data hasil normalisasi ( $k=3$ )

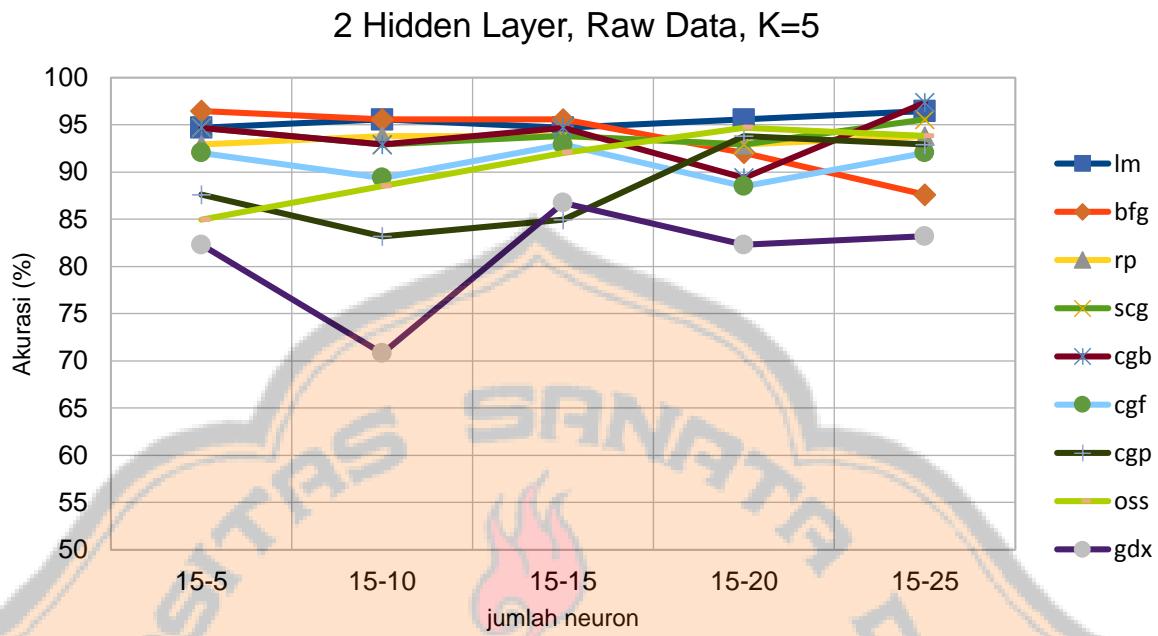
Saat masih digunakan data normalisasi, pada arsitektur 2 *hidden layer*, dimana 15 *neuron* pada *layer* pertama dan hasil akurasi tertinggi dicapai saat penggunaan trainoss dengan 20 *neuron* pada *layer* kedua. Hasil akurasi yang dicapai sebesar 97.35% (3 misklasifikasi dari 113 data).

2 Hidden Layer, Data Normalisasi, K=3

Gambar 4.34 – Akurasi yang dicapai tiap fungsi *training* pada pemilihan jumlah neuron pada *layer* kedua dan penggunaan data hasil normalisasi (k=3)

1 Hidden Layer, Raw Data, K=3

Gambar 4.35 – Akurasi yang dicapai tiap fungsi *training* pada pemilihan jumlah neuron pada *layer* pertama dan penggunaan data hasil mentah (k=3)



**Gambar 4.36** – Akurasi yang dicapai tiap fungsi *training* pada pemilihan jumlah *neuron* pada *layer* kedua dan penggunaan data hasil mentah (*k*=5)

Sedangkan saat penggunaan data mentah tanpa normalisasi terlebih dahulu, pada arsitektur 1 *hidden layer*, akurasi tertinggi dicapai saat menggunakan trainlm dan 15 *hidden neuron* dengan hasil akurasi sebesar 97.35% (3 misklasifikasi dari 113 data). Dan saat menggunakan 2 *hidden layer*, dimana *layer* pertama 15 neuron, hasil akurasi tertinggi dicapai saat menggunakan fungsi traincgb dan 25 *hidden neuron* pada *layer* kedua, dengan hasil akurasi sebesar 97.35% (3 misklasifikasi dari 113 data), dimana pada proses ini digunakan 5 *k-fold* dalam proses *train-test-validation*.

Dalam penelitian ini, jaringan saraf tiruan yang diatih sudah dapat melakukan identifikasi bentuk biji utuh dan biji pecah, penggunaan 3 *fold* data sudah memberikan hasil yang baik dalam proses test hasil *training*. Hal ini dapat dilihat dari akurasi yang cukup tinggi

## BAB V

### PENUTUP

#### 5.1. Kesimpulan

Dalam hasil penelitian identifikasi bentuk biji kopi menggunakan deskriptor bentuk dasar dan jaringan saraf tiruan dapat disimpulkan sebagai berikut

1. Proses ekstraksi ciri, dengan menggunakan deskriptor bentuk dasar (*basic region descriptor*) dapat menghasilkan sebuah *dataset* ciri yang terdiri dari 9 komponen yakni panjang, lebar, diameter, perimeter, luas, rasio kerampingan, rasio kebulatan, dispersi I dan dispersi IR
2. Proses identifikasi menggunakan jaringan saraf tiruan :
  1. Dalam pemilihan arsitektur, terdapat 5 (buah) arsitektur yang paling optimal dalam melakukan identifikasi yakni:
    1. *1 Hidden Layer, 5 Hidden Neuron, K=3*, data normalisasi, dan fungsi *training traincgp* dengan 97.35%.
    2. *1 Hidden Layer, 15 Hidden Neuron, K=3*, data normalisasi, dan fungsi *traning trainscg* dengan 97.35%
    3. *2 Hidden Layer, 15 Hidden Neuron -1, 20 Hidden Neuron -2, K=3*, data normalisasi, dan fungsi *training trainoss* dengan 97.35%
    4. *1 Hidden Layer, 15 Hidden Neuron, K=3*, data mentah, dan fungsi *training trainlm* dengan 97.35%
    5. *2 Hidden Layer, 15 Hidden Neuron -1, 25 Hidden Neuron-2, K=5*, data normalisasi, dan fungsi *training traincgb* dengan 97.35%
  2. Dalam pemilihan *fold*, nilai *3-fold* dalam penentuan pemilihan data *trainin-testing-validation* memiliki hasil peforma yang lebih tinggi dari *5-fold*
  3. Jaringan saraf tiruan yang terbentuk sudah mampu melakukan identifikasi terhadap bentuk biji kopi pecah dan biji kopi utuh

## 5.2. Saran

Identifikasi bentuk biji kopi menggunakan deskriptor bentuk dasar dan jaringan saraf tiruan, ada beberapa saran yang penulis berikan seperti

1. Beberapa ciri bisa ditambahkan seperti intensitas warna atau cahaya sehingga identifikasi bisa dilakukan ke ranah yang lebih luas seperti identifikasi jenis cacat biji kopi sesuai SNI
2. Dengan menggunakan ekstraksi ciri tersebut, identifikasi dapat dilakukan dengan metode selain jaringan saraf tiruan seperti metode *Naive Bayes*, *K-Nearest Neighbors*, atau metode lainnya.

## DAFTAR PUSTAKA

- Armansyah, M. 2010. *Mempelajari Minuman Formulasi dari Kombinasi Bubuk Kakao dengan Jahe Instan*. Teknonoli Pertanian Universitas Hasanuddin : Makassar
- Ayitenfsu, Betelihem Mesfin. 2014. *International Journal of Engineering Research & Technology Vol. 3 Issue 2 : Method of Coffee Bean Defect Detection*. IJERT : India
- Badan Standardisasi Nasional (BSN). 2008. *SNI 01-2907-2008 Tentang Biji Kopi*
- Faridah, Parikesit, Gea O.F dan Ferdiansjah. 2011. *TELKOMNIKA Vol 9, no 3 : Coffee Bean Grade Determination Based on Image Parameter*. Direktorat Pendidikan Tinggi : Jakarta
- Freeman, Herbert. 1961. *IRE Transaction on Electronic Computers : On the Encoding of Arbitrary Geometric Configurations*. IEEE : United States
- Gonzalez, R.C. dan Woods. 2002. *Digital Image Processing*. Prentice Hall : USA
- International Coffee Organization. 2002. *ICC Resolution no 407/02*. International Coffee Council : London, England.
- Kadir, Abdul dan Susanto, Adhi. 2013. *Teori dan Aplikasi Pengolahan Citra*. Penerbit Andi : Yogyakarta
- Kratzert, Frederick. 2015. Matlab File Exchange : centerobject(bw),  
<http://www.mathworks.com/matlabcentral/fileexchange/52560-centerobject-bw->

Madi, Sri Citra Yuliana. 2010. *Pemutuan Biji Kopi dengan Menggunakan Pengolahan Citra (Image Processing)*. Institut Pertanian Bogor:Bogor

Parker, J. R. 1997. *Algorithms for Image Processing and Computer Vision*. Springer-Verlay Telos : USA

Putra, Darma. 2010. *Pengolahan Citra Digital*. Penerbit Andi:Yogyakarta

Siang, Jong Jek. 2005. *Jaringan Saraf Tiruan dan Pemrogramannya Menggunakan Matlab*. Penerbit Andi : Yogyakarta.

Soff'i, Imam. 2005. *Pemutuan Biji Kopi dengan Pengolahan Citra Digital dan Artificial Neural Network*. Institut Pertanian Bogor:Bogor

Widiarti, Anastasia Rita. 2013. *Teori dan Aplikasi Pengolahan Citra Digital : Transliterasi Otomatis Citra Dokumen Teks Aksara Jawa*. Lintang Pustaka Utama : Yogyakarta