IDENTIFIKASI CITRA HURUF ARAB MENGGUNAKAN METODE JARINGAN SYARAF TIRUAN *KOHONEN*

S K R I P S I

Diajukan Kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

O l e h

ELLIA NURANTI KUSUMA
NIM. 10650032

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM
MALANG
2015

HALAMAN PERSETUJUAN

IDENTIFIKASI CITRA HURUF ARAB MENGGUNAKAN METODE JARINGAN SYARAF TIRUAN *KOHONEN*

SKRIPSI

Oleh :

ELLIA NURANTI KUSUMA
NIM: 10650032

Telah Disetujui,

Malang, 10 Maret 2015

Dosen Pembimbing I	Dosen Pembimbing II
A'la Syauqi, M.Kom	**Dr. Ahmad Barizi, M.A**
NIP. 19177120 12008011 007	**NIP. 19731212 199803 1 001**

Mengetahui,

Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP.197404242009011008

HALAMAN PENGESAHAN

IDENTIFIKASI CITRA HURUF ARAB MENGGUNAKAN METODE JARINGAN SYARAF TIRUAN *KOHONEN*

S K R I P S I

Oleh :

Ellia Nuranti Kusuma
NIM. 10650032

Telah Dipertahankan Di Depan Dewan Penguji Skripsi
Dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer Strata Satu (S. Kom)

Tanggal 10 April 2015

Susunan Dewan Penguji :			Tanda Tangan
1. Penguji Utama	:	Dr. Cahyo Crysdian NIP. 19740424 2009011 008	( )
2. Ketua Penguji	:	Dr. Muhammad Faisal, M.T NIP. 19740502 00501 1 007	( )
3. Sekretaris	:	A'la Syauqi, M.Kom NIP. 19177120 12008011 007	( )
4. Anggota Penguji	:	Dr. Ahmad Barizi, M.A NIP. 19731212 199803 1 001	( )

Mengetahui dan Mengesahkan,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

S U R A T P E R N Y A T A A N

Saya yang bertandatangan di bawah ini saya:

Nama	:	Ellia Nuranti Kusuma
NIM	:	10650032
Fakultas/Jurusan	:	Sains dan Teknologi / Teknik Informatika
Judul Penelitian	:	Identifikasi Citra Huruf Arab Menggunakan Metode Jaringan Syaraf Tiruan *Kohonen*

Menyatakan dengan sebenar-benarnya bahwa skripsi yang saya buat tidak terdapat unsur-unsur penjiplakan karya penelitian atau karya ilmiah yang pernah dilakukan atau dibuat oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata hasil penelitian ini terbukti terdapat unsur-unsur jiplakan, maka saya bersedia untuk mempertanggungjawabkan, serta diproses sesuai peraturan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sebenar-benarnya dan tanpa paksaan dari siapapun.

Malang, 10 Maret 2015

Yang Membuat Pernyataan,

Ellia Nuranti Kusuma
NIM: 10650032

MOTTO

واستعينوا بالصبر والصلوة وانها لكبيرة الا ءلى الخاشعين

" Jadikanlah Sabar dan Shalat sebagai penolongmu. Dan sesungguhnya yang demikian itu sungguh berat, kecuali bagi orang-orang yang Khusyu'"

(Qs. Al-Baqarah : 45)

HALAMAN PERSEMBAHAN

Segala Puji bagi Allah SWT,

Kupersembahkan sebuah karya teruntuk orang-orang

yang ada di sekelilingku yang sangat kusayangi dan aku banggakan

Karya ini saya Persembahkan kepada :

Ayah (Sudianto), Ibunda (Nurul Munifah (alm)) & Ibunda Insiyah Yuningsih

Terima Kasih atas Do'a, Dukungan, Bimbingan yang tak pernah berhenti mengiringi di setiap langkahku. Terima kasih atas segala pengorbanan, nasehat dan perjuangan serta limpahan kasih sayang yang telah ayah & ibu curahkan.

Seluruh keluargaku

Adik-ku (Yukha Nurudduja Kusuma), Pakde dan Bude, Kakek & Nenek, Tante & Om, serta keluarga besarku yang telah memberi dukungan dalam segala hal.

Sahabat-sahabatku

Sahabat-sahabat seperjuangan Teknik Informatika UIN Malang angkatan 2010 , semoga Allah SWT senantiasa melimpahkan rahmat dan barokah ilmu yang kelak mengantarkan kita semua untuk menjalani kehidupan mendatang. Amin...

Serta rekan-rekan dan semua pihak yang tidak dapat disebutkan satu persatu, yang telah membantu dari awal kuliah hingga tersusunnya skripsi ini.
Terimakasih...

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Syukur Alhamdulillah, segala puji dan syukur dengan tulus kami persembahkan ke hadirat Allah SWT, karena hanya dengan rahmat, petunjuk dan hidayah-Nya penulis mampu menyelesaikan tugas akhir yang berjudul *IDENTIFIKASI CITRA HURUF ARAB MENGGUNAKAN METODE JARINGAN SYARAF TIRUAN KOHONEN*.

Shalawat serta salam penulis haturkan pada junjungan Nabi Muhammad SAW yang memberikan motivasi bagi umat Islam, khususnya bagi penulis untuk selalu berproses menuju insan yang memiliki intelektual tinggi & berakhlak mulia.

Penyelesaian skripsi ini merupakan suatu pekerjaan rumit & panjang yang wajib penulis selesaikan. Namun berkat ma'unnah Allah SWT dan bantuan dari berbagai pihak baik berupa moril maupun materiil , akhirnya tugas akhir ini dapat terselesaikan dengan baik. Oleh karena itu penulis menyampaikan rasa hormat, ungkapan terima kasih kepada:

1. Prof. Dr. H. Mudjia Rahardjo, M.Si , selaku Rektor UIN Maulana Malik Ibrahim Malang.
2. Dr. drh. Hj. Bayyinatul Muchtaromah, M.Si, selaku Dekan Fakultas Sains dan Teknologi UIN Maulana Malik Ibrahim Malang.
3. Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika UIN Maulana Malik Ibrahim Malang.

4. A'la Syauqi, M.Kom, selaku pembimbing I yang dengan sabar memberikan arahan, saran dan motivasi pada peneliti sehingga skripsi ini dapat terselesaikan dengan baik.
5. Dr. Ahmad Barizi, M.A, selaku Dosen Pembimbing II yang bersedia meluangkan waktu untuk memberikan bimbingan dan arahan dalam menyelesaikan skripsi ini.
6. Seluruh Teman-teman seperjuangan Teknik Informatika angkatan 2010, yang selalu memberikan motivasi untuk penyelesaian skripsi ini.

Sebagaimana pepatah "tiada gading yang tak retak", maka skripsi ini pun masih banyak terdapat kekurangan dan kelemahan. Oleh karena itu peneliti mengaharapkan kritik dan saran untuk perbaikan di masa mendatang.

Penulis berharap semoga skripsi ini bisa memberikan manfaat bagi pembaca. Selain itu peneliti berharap semoga skripsi ini dapat memberikan nilai guna baik bagi peneliti maupun bagi pembaca. *Amin Ya Robbal'Alamin*

Malang, 10 Maret 2015

Peneliti,

Ellia Nuranti Kusuma

DAFTAR ISI

HALAMAN JUDUL ..... **i**
HALAMAN PERSETUJUAN ..... **ii**
HALAMAN PENGESAHAN ..... **iii**
HALAMAN PERNYATAAN ..... **iv**
MOTTO ..... **v**
HALAMAN PERSEMBAHAN ..... **vi**
KATA PENGANTAR ..... **vii**
DAFTAR ISI ..... **viii**
DAFTAR TABEL ..... **x**
DAFTAR GAMBAR ..... **xii**
DAFTAR LAMPIRAN ..... **xiii**
ABSTRAK ..... **xiv**
ABSTRACT ..... **xv**

BAB I PENDAHULUAN

1.1 Latar Belakang 1
1.2 Identifikasi Masalah 5
1.3 Tujuan Penelitian 5
1.4 Batasan Penelitian 6
1.5 Manfaat Penelitian 6
1.6 Sistematika Penulisan 7

BAB II TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu 10
2.2. Bahasa Arab 12
2.2.1 Pentingnya Belajar Bahasa Arab 12
2.2.2 Urgensi Teknologi Dalam Pembelajaran Bahasa Arab 16
2.3 Sistem Pengenalan Biometrika 17
2.4 Pengenalan Pola 18
2.5 Pengenalan Pola Tulisan Tangan 19
2.6 Pengolahan Citra 21
2.7 Algoritma *Kohonen Neural Network* 23
2.8 Ruang Lingkup Aplikasi 26

BAB III ANALISIS DAN PERANCANGAN

3.1 Gambaran Umum Sistem 29
3.2 Alat dan Bahan 31
3.3 Perancangan dan Alur Sistem 32
3.3.1 *Diagram* Penggunaan Aplikasi 33
3.3.2 Alur dan Desain Sistem Aplikasi 34
3.3.3 Alur Sistem (*Image Processing*) 36
3.3.4 Alur Sistem *Downsampling* 36
3.3.5 Alur Sistem *Dot Detection* 38
3.3.6 Alur Sistem *Cropping* dan *Clear Space* 41
3.3.7 Alur Sistem *Feature Extraction* 43
3.4 Alur Sistem Lapisan (Layer) *Network* / Jaringan Syaraf Tiruan 45
3.4.1 Layer 1 : Layer Inputan JST 50
3.4.2 Layer 2: Bobot Koneksi (Penentuan Bobot) 52
3.4.2.1 Proses Training JST Kohonen 52
3.4.3 Layer 3 : Perkalian Tiap Neuron Output 65
3.4.4 Layer 4 : Seleksi Nilai Output Terbesar 67
3.4.5 Layer 5 : Mendapatkan Output Neuron Pemenang 68
3.5 Desain Tampilan Aplikasi 69

BAB IV HASIL DAN PEMBAHASAN

4.1 Alat dan Bahan 73
4.2 Deskripsi Program 73
4.3 Implementasi dan Pembahasan 82
4.3.1 Proses Input Gambar 82
4.3.2 Proses *Downsample* Gambar 83
4.3.3 Mengatur Ukuran dan Posisi 84
4.3.4 Cara Kerja *Downsample* dalam Sistem 88
4.3.5 Penerapan Algoritma JST Kohonen 89
4.3.6 *Training* JST Kohonen 93
4.4 Hasil Uji Coba Aplikasi 97

4.5 Analisa Hasil dan Pembahasan Algoritma 107
4.6 Integrasi dan Kajian Islam 110

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan 114
5.2 Saran 115

DAFTAR PUSTAKA

LAMPIRAN-LAMPIRAN

DAFTAR TABEL

Tabel 1.1 Pola Inputan huruf Arab Alif sampai Ya' yang dibuat menggunakan pemetaan pada grid-grid pixel 7x9 ……76

Tabel 1.2 Klasifikasi nilai matriks basisdata *sample.dat* berdasarkan pola inputan huruf Arab……..80

Tabel 1.3 Uji Coba Data Training Input huruf Arab yang menghasilkan parameter yaitu nilai *error*……...104

DAFTAR GAMBAR

Gambar 2.1 Koordinat Gambar Digital 21
Gambar 2.2 Skema Jaringan Syaraf Tiruan 24
Gambar 3.3 Blog diagram system 32
Gambar 3.3.1 Diagram penggunaan Aplikasi 33
Gambar 3.3.2 Desain Sistem Aplikasi 34
Gambar 3.3.3 Alur Sistem (*Image Processing*) 36
Gambar 3.3.4 Alur Sistem (*Downsampling*) 37
Gambar 3.3.5 Source code proses (*Downsampling)* 38
Gambar 3.3.6 Alur Sistem (*Dot Detection*) 40
Gambar 3.3.7 Source code proses (*Dot detection "Pixel Grabber")* 41
Gambar 3.3.6 Source code proses (*Dot detection)* 41
Gambar 3.3.8 Alur Proses (*Cropping*) 42
Gambar 3.3.8 Source code proses (*Cropping)* 42
Gambar 3.3.9 Source code proses (*Cropping)* 42
Gambar 3.3.10 Alur Proses (*Feature Extraction)* 44
Gambar 3.3.11 Source code (*downsample process)* 44
Gambar 3.4 *Flowchart* Algoritma JST *Kohonen* Pada Sistem 47
Gambar 3.5 Arsitektur jaringan / *network* JST Kohonen 49
Gambar 3.5.1 *Grid pixel* 7x9 sebagai inputan neuron 50
Gambar 3.5.2 Source code untuk menghitung panjang vektor 51
Gambar 3.5.3 Source code normalisasi input 51
Gambar 3.5.4 *Flowchart* algortima JST *Kohonen* pada saat *Training* 53
Gambar 3.5.5 Source code untuk mengacak bobot 55
Gambar 3.5.6 Source code evaluasi error dan melacak neuron pemenang 57
Gambar 3.5.7 Source code penyesuaian & perhitungan tiap bobot koneksi 60
Gambar 3.5.8 Alur proses untuk memenangkan neuron yang belum menang 62
Gambar 3.5.9 Source code perulangan training (memenangkan neuron) 63
Gambar 3.5.10 Source code evaluasi error terkecil dan terbaik 65
Gambar 3.5.11 Source code proses perhitungan output neuron (dot product) 67

Gambar 3.5.12 Source code menghasilkan dan memilih neuron pemenang 69
Gambar 3.6 Desain Halaman *Interface* Aplikasi 70
Gambar 4.0 Halaman *Interface* Aplikasi Identifikasi 77
Gambar 4.0.1 Source code proses *Image processing*.....84
Gambar 4.0.2 Source code input disimpan dalam format gambar *off-screen*.....85
Gambar 4.0.3 Source code inputan gambar disimpan dalam grid *pixel* 7x9.....86
Gambar 4.0.4 Source code metode "*pixelGrabber*" pada grid *pixel* 7x9.....88
Gambar 4.0.5 Source code metode untuk menerima koordinat *y* batas tulisan ...89
Gambar 4.0.6 Source code untuk menghitung garis dan "*cropping*" 89
Gambar 4.0.7 Source code membagi gambar ke area dibawah resolusi 7x9 90
Gambar 4.0.8 Source code proses neuron masukan awal pada jaringan 93
Gambar 4.0.9 Source code winner neuron yang disimpan dalam parameter 93
Gambar 4.0.10 Source code hitung neuron output dengan mengalikan bobot 94
Gambar 4.0.11 Source code mengatur ukuran agar tetap mengenali inputan.....96
Gambar 4.0.12 Source code membangun jaringan syaraf & training set 95
Gambar 4.0.13 Source code menyesuaikan bobot matrik & training jaringan.....97
Gambar 4.0.14 Source code menghitung error dan mengecek error terkecil.....98
Gambar 4.0.15 Source code proses selesai jika error bernilai minimum.....98
Gambar 4.0.16 Source code memilih satu neuron yang benar-benar menang.....98
Gambar 4.1 Proses Memuat Data pada Aplikasi 99
Gambar 4.2 Proses Training Data pada Aplikasi.....100
Gambar 4.3 Aksi Input Tulisan pada Canvas.....102
Gambar 4.4 Proses Pengenalan Tulisan pada Aplikasi.....103
Gambar 4.5 Output Pengenalan Tulisan pada Aplikasi.....104
Gambar 4.6 Output Identifikasi huruf Arab (Fa').....104
Gambar 4.7 Output Identifikasi huruf Arab (Mim).....105
Gambar 4.8 Output Identifikasi huruf ('Ain) dapat dikenali oleh Sistem.....111
Gambar 4.9 Output Identifikasi huruf (Fa') tidak dapat dikenali Sistem.....112
Gambar 4.10 Output Identifikasi huruf (Tsa') tidak dapat dikenali Sistem.....113

DAFTAR LAMPIRAN

Lampiran 1: Ciri Citra Pola Tulisan Tangan berdasarkan grid pixel 7x9

Lampiran 2: Uji Coba data Pengujian Input Tulisan Huruf Arab

ABSTRAK

Nuranti Kusuma, Ellia. 2015. 10650032. ***Identifikasi Citra Huruf Arab Menggunakan Metode Jaringan Syaraf Tiruan Kohonen***. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
Pembimbing: (I) A'la Syauqi, M.Kom (II) Dr. Ahmad Barizi, M.A

Kata Kunci : *Huruf Arab, Image Processing, Jaringan Syaraf Tiruan.*

Penelitian ini mengimplementasikan metode untuk melakukan proses pengenalan dan identifikasi tulisan tangan huruf Arab. Pada pemrosesan awal digunakan metode *Image Processing* untuk memproses inputan citra tulisan huruf Arab yaitu dengan cara mengambil nilai *pixel* inputan citra yang digunakan untuk ekstraksi fitur citra tulisan huruf Arab. Metode yang selanjutnya adalah algoritma jaringan syaraf tiruan *Kohonen* yang digunakan untuk proses pengenalan dan identifikasi citra tulisan huruf Arab. Sampel data citra tulisan dibuat polanya yang dirupakan dengan nilai-nilai matriks oleh sistem sendiri yang akan digunakan sebagai acuan.

Image Processing berfungsi untuk mengekstrak ciri-ciri citra tulisan huruf Arab dengan cara melakukan pengambilan nilai *pixel* aktif inputan citra yang berwarna hitam dan bernilai 1. Selanjutnya sistem juga membuat sample Ciri yang sebelumnya dibuat di dalam sistem untuk dijadikan perbandingan antara inputan data dari user dan data dari sistem. Nilai *pixel* yang aktif (bernilai 1) akan diteruskan ke dalam proses pengenalan citra menggunakan JST *Kohonen* yang akan dianggap sebagai node/neuron JST.

Inputan sistem adalah citra tulisan huruf Arab. Pada awal pemrosesan input tulisan Arab *real-time* dirubah ke dalam citra berformat gambar JPEG. Lalu sistem melakukan pemrosesan gambar dengan mengekstrak gambar. Untuk mengetahui perbandingan hasil data training dan data testing algoritma jaringan syaraf tiruan yang terbaik maka dilakukan dengan cara melakukan uji coba input tulisan saat training dan testing secara berulang-ulang. Sehingga akan dihasilkan sebuah angka tingkat kesalahan (*error*) saat proses training dan testing.

Data-data tersebut menghasilkan hasil huruf untuk pengenalan jaringan syaraf tiruan *Kohonen* yang cukup baik, yakni 85%. Dari hasil uji sistem saat training dan testing terbaik, berpengaruh untuk jumlah *sample* citra tulisan yang banyak, dari hasil penelitian yang dilakukan untuk kerja jaringan syaraf tiruan menurun untuk *sample* jumlah citra tulisan yang sedikit dan kemiripan bentuk tulisan.

مستخلص البحث

إيليا نورانتي كوسوما ، 2015. 10650032. تحديد الهوية لطبيعة الحروف العربية باستخدام منهج الشبكة العصبية كوهونين (kohonen) . شعبة تقنية الإعلامية في كلية العلوم والتكنولوجيا بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانق.

المشرف : أعلى شوقي الماجستير و دكتور أحمد بارزي الماجستير

الكلمة الأساسية : الحروف العربية، منهج الشبكة العصبية كوهونين (kohonen)، معالجة الصور

بحث هذا البحث العلمي في تطبيق المنهج لعملية التعرف والمعرفة وكذلك تحديد الهوية لطبيعة الحروف العربية باستخدام معالجة الصور في أول تطبيقه وهذا لعملية وتنفيذ طبيعة الحروف العربية الداخلة بأخذ نتيجة *pixel* لاقتطف النوع الحروف العربية. والمنهج المستخدم بعده هو خوارزمية الشبكة العصبية كوهونين (kohonen) لمعرفة وتحديد الهوية لطبيعة الحروف العربية. وعينات بيانات لطبيعة الحروف العربية يصنع له طرازا وأسلوبا المشبه بقيمة المصفوفة بالنظام كالإشارة.

ويستخدم معالجة الصورلاستخراج طبيعة الحروف العربية بأخذ نتيجة *pixel* النشاط بلون الأسود بقيمة 1. ثم جعل النظام عينة لتقارن بين البيانات من المستعمِل والبيانات من النظام. و نتيجة *pixel* النشاط بلون الأسود بقيمة 1 يتصل إلى عملية التعرف والمعرفة باستخدام منهج الشبكة العصبية كوهونين (kohonen) المعرَّف بالشبكة العصبية كوهونين (kohonen) أو عصبة JST

وإدخال النظام هو طبيعة الحروف العربية . وفي أول نظم إدخالها يُغير طبيعة الحروف العربية إلى .JPEG . لتكون الشكل صورة لمعرفة المقارنة بين التجربة والنتيجة من تجربة باستخدام منهج الشبكة العصبية كوهونين الأحسن ، ثم يوصل بإدخال الكتابة مرارا للتجربة وهذا لحصول درجة وقيمة ورقما في الخطيئة (*error*) عند عملية التجربة.

وتلك البيانات تحصل على معرفة الشبكة العصبية كوهونين الجيد هي 85 %. ونتيجة التجربة الأحسن يؤثر كثيرا إلى عدد العينات لكثرة الكتابة. وانخفض نتيجة عمل الشبكة العصبية كوهونين لعينة الكتابة القليلة من حيث النوع والعدد والتشابه في الكتابة.

ABSTRACT

Nuranti Kusuma, Ellia. 2015. 10650032. ***Arabic Character Image Identification Using Kohonen Artificial Neural Network Algorithm***. Thesis. Informatics Department of Faculty of Science and Technology. State Islamic University Maulana Malik Ibrahim, Malang.
Advisor : (I) A'la Syauqi, M.Kom (II) Dr. Ahmad Barizi, M.A

Keyword : *Arabic Characters, Image Processing, and Artificial Neural Network.*

This study implement method to perform Arabic character handwritten recognize and identification processing. The first process using a *Image Processing* method to process of Arabic character image input that is a take image input *pixel* value that using to feature extraction process of Arabic character letter Image. The next method using a *Kohonen Artificial Neural Network* algorithm that using to recognize and identification process of Arabic character image. Letter of character image data sample will be make a pattern that represented with matrix values by own system that will be used as reference or orientation.

The Purpose of *Image Processing* is that to extraction Arabic character image feature with do taking active *pixel* value of image input that *pixel* color a black and value is 1. The next, a system too make a feature sample that before make it in the system to become a comparison between data input from user and data from system. Active *Pixel* value (value is a 1) will be continue an image recognition process that using a *Kohonen Artificial Neural Network* that will be named a node / neuron of neural network.

Input of system is a Arabic character image. The first process, input of *real-time* Arabic character letter would be change in JPEG image format. Then system work with do image processing that is a image extraction process. To know that a comparison result of training data and testing data of neural network algorithm best work, so will be do that with a experiment try test character letter input when training and testing according repeatedly. With the result that will be a fault digit rate (*error*) when training and testing process.

The data will be a result to recognition process of *Kohonen* neural network be the best, that is 85%. From the experiment test system when training and testing be the best, influence to many image character letter quantity, from the result of research that do thing for neural network activity will be a decrease for few image character letter quantity and identically character letter shapes.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Bahasa Arab tersusun atas susunan huruf Arab / huruf Hijaiyah yang tentunya kita jumpai pada kitab suci agama islam yaitu berupa Al-Qur'an. Al Qur'an adalah kitab suci bagi pemeluk agama Islam yang wajib dipelajari dan setiap Muslim wajib bisa membaca Alqur'an serta bisa mengenali tiap-tiap huruf Hijaiyah yang menyusunnya. Seperti yang telah disebutkan di Al-Qur'an :

إِنَّآ أَنزَلْنَٰهُ قُرْءَٰنًا عَرَبِيًّا لَّعَلَّكُمْ تَعْقِلُونَ ۝٢

Artinya: "*Sesungguhnya Kami menurunkannya berupa Al Quran dengan berbahasa Arab, agar kamu memahaminya".* (Q.S.Yusuf [12]:2)

Belajar bahasa Arab wajib dilakukan oleh setiap Muslim. Bahasa Arab wajib dipelajari karena bahasa Arab merupakan bahasa Al-Qur'an, bahasa komunikasi dan informasi umat Islam. Allah SWT menurunkan Al-Qur'an kepada Rasulullah SAW dalam bahasa Arab yang nyata agar menjadi mukjizat yang kekal dan menjadi hidayah bagi seluruh manusia di setiap waktu (zaman). Selain itu, Hukum mempelajari bahasa Arab adalah wajib karena tidaklah sempurna mempelajari Al-Qur'an tanpa adanya belajar bahasa Arab yang akan menjadi pedoman dalam mengetahui tafsir dan makna dari ayat-ayat Al-Qur'an. Seyogyanya Seorang Muslim

tidak hanya diharuskan untuk belajar mengenali, membaca dan menulis saja. Unsur terpenting dari wajibnya belajar bahasa Arab adalah mengetahui huruf-huruf yang menyusun tulisan Arab itu sendiri. Huruf Arab mempunyai peranan yang sangat penting dalam menguasai suatu bahasa arab. Karena mengetahui huruf dari suatu bahasa Arab, akan mengantarkan kita ke dunia luar untuk proses sosialisasi yang lebih luas.

Berdasarkan penalaran tersebut dapat ditarik kesimpulan bahwa belajar Huruf Arab adalah sangat perlu dalam penguasaan bahasa Arab. Apalagi jika media yang digunakan dinilai sangat mumpuni dalam perealisasiannya.

Sistem Biometrik pada hakikatnya merupakan sebuah sistem pengenalan pola yang melakukan identifikasi personal dengan melakukan keotentikan dari karakteristik fisiologis (wajah, retina, sidik jari, tulisan tangan). Biometrik mempunyai kemampuan yang lebih baik daripada teknologi konvensional, terutama dalam hal memproses ciri / fitur guna menjadi mudah, selain itu juga mempunyai tingkat keunikan tertentu. Pengembangan teknologi biometrik seperti wajah, sidik jari, iris mata, tulisan tangan sudah banyak ditemukan dengan tujuan tertentu, seperti sistem keamanan, sistem pengenalan, sistem pengabsenan, dll. Teknologi biometrika yang sudah diciptakan dan diterapkan diberbagai aplikasi terkadang pada kenyataannya proses pengenalan tersebut masih mengalami kegagalan. Beberapa kesalahan sering muncul diakibatkan oleh beberapa faktor diantaranya misalnya jarak

tepi citra, perhitungan koordinat, tingkat kemiringan, pengukuran skala, bentuk tulisan huruf, dll.

Identifikasi adalah salah satu tahapan kegiatan dalam proses Interpretasi. Interpretasi merupakan perbuatan mengkaji dengan maksud untuk mengidentifikasi objek dan menilai arti pentingnya objek tersebut (*Este dan Simonett, 1975*). Identifikasi bertujuan untuk membedakan komponen-komponen yang satu dengan yang lainnya. Identifikasi berfungsi untuk membedakan objek-objek antara satu dengan lainnya lalu mencocokkan antara 2 atau lebih objek yang sedang diidentifikasi. Identifikasi pada citra dapat dibagi menjadi 3 ciri utama yang tergambar pada citra berdasarkan fitur yang terekam oleh sensor, yaitu ciri Spektoral, ciri Spatial, dan ciri temporal. Pada penelitian ini menggunakan ciri Spatial yang meliputi bentuk, ukuran, pola, tekstur dan batas jarak. Dalam interpretasi citra, pengenalan pola merupakan bagian penting, karena tanpa pengenalan identitas dan jenis objek, maka objek yang tergambar pada citra tidak mungkin dianalisis (*Sutanto:1999*). Hal ini tentu memunculkan sebuah permasalahan baru yang harus ada solusi penyelesaiannya.

Pada penelitian-penelitian mengenai pengenalan pola tulisan tangan sebelum melakukan proses pencocokan dan identifikasi, terdapat langkah untuk memproses citra *(Image).* Hasil dari pemrosesan Citra ini akan menghasilkan suatu *feature* yang akan digunakan sebagai input untuk algoritma pengenalan jaringan syaraf tiruan (JST) *Kohonen Neural Network*. Jaringan syaraf tiruan *(Artificial Neural Network)*

adalah salah satu sistem pemrosesan informasi yang didesain dengan meniru cara kerja sistem otak manusia dalam menyelesaikan suatu permasalahan dengan melakukan proses pembelajaran dan pelatihan melalui perubahan bobot-bobot. Jaringan syaraf tiruan mampu melakukan pengenalan kegiatan berbasis data-data. data-data tersebut akan dipelajari dan dilatih oleh jaringan saraf tiruan sehingga mempunyai kemampuan untuk memberikan keputusan terhadap data yang belum pernah dipelajari dan dilatih (*Kiki dan Kusumadewi, 2004: 2*).

Sejak ditemukan pertama kali oleh Mc Culloch dan Pitts pada tahun 1948, JST telah berkembang pesat dan telah banyak digunakan pada banyak aplikasi. Perhatian yang besar pada JST disebabkan adanya keunggulan yang dimilikinya seperti kemampuan untuk belajar dan pelatihan, komputasi paralel dan kemampuan untuk memodelkan fungsi linear dan nonlinear (*Dayhoff dan Judith.E, 1990: 264*).

Berdasarkan permasalahan tersebut maka akan dibangun suatu aplikasi berbasis komputer yang bertujuan untuk dapat mengidentifikasi suatu objek yaitu berupa tulisan pada suatu citra. Proses identifikasi ini bekerja untuk dapat mengetahui perbedaan dari proses mendeteksi dan mengenali tulisan huruf Arab yang akan / sedang diidentifikasi dengan tulisan huruf Arab yang sudah teridentifikasi (sudah dikenali dan diketahui). Sehingga tidak akan terjadi ketidakvalidan data objek tulisan pada citra.

Aplikasi ini akan diberi kemampuan untuk mengenal pola tulisan tangan *(Handwrite Recognition)* huruf Arab lalu mengidentifikasi huruf Arab yang akan /

sedang dilatih dengan data citra tulisan di database dengan menerapkan metode *Kohonen Neural Network*. Aplikasi ini akan dijalankan pada *Software Integrated System Netbeans IDE.*

1.2 Identifikasi Masalah

Berdasarkan penjelasan pada latar belakang tersebut, maka identifikasi masalah dari dilakukannya penelitian ini adalah bagaimana mengimplementasikan metode Jaringan syaraf Tiruan *Kohonen* untuk proses pengenalan dan pengidentifikasian citra huruf Arab pada sistem yang akan dibangun.

1.3 Tujuan Penelitian

Dari latar belakang yang telah dibahas sebelumnya, maka tujuan dari dilakukannya penelitian ini adalah pembuatan aplikasi untuk identifikasi arti tulisan huruf Arab secara *online* yang dibuat berbasis komputer dengan menerapkan *Kohonen Neural Network* sebagai metode pengenalan citra tulisan tangan. Dengan dibangunnya aplikasi ini diharapkan dapat menciptakan suatu sistem yang bertujuan untuk dapat mengidentifikasi suatu objek pada citra yang pada akhirnya berfungsi untuk membedakan objek-objek tersebut antara satu dengan lainnya lalu mencocokkan antara 2 objek yang sedang diidentifikasi.

1.4 Batasan Penelitian

Adapun batasan terhadap ruang lingkup permasalahan dalam penelitian ini adalah sebagai berikut :

1. Aplikasi pengenalan pola tulisan ini beracuan pada pengenalan Huruf Hijaiyah yang mana dapat mengenali semua Huruf Hijaiyah mulai dari Alif sampai Ya' dan terbatas pada belum menyertakan komponen vokal Huruf Hijaiyah yaitu *Harakat Fathah* (bervokal a) , *Kasrah* (bervokal i) dan *Dhummah* (bervokal u) serta komponen Harakat lain seperti *Tasdyid, Tanwin, dll*. Dengan kata lain aplikasi ini hanya dapat mengenali Huruf Hijaiyah asli / murni.
2. Aplikasi pengenalan pola tulisan ini dibuat untuk mengenali Huruf Hijaiyah dengan berbasis Tulisan tangan secara *Online / Real Time* , inputnya menggunakan mouse (cursor) yang hasilnya berupa Pengenalan Huruf akan di representasikan pada *Screen* (layar).

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut :

a. Manfaat Keilmuan

Hasil penelitian bermanfaat untuk mengembangkan teori-teori ilmu komputer terutama yang berhubungan dengan *image processing* & *handwrite recognition* yang dituangkan dalam pembuatan sebuah aplikasi identifikasi tulisan huruf

Arab sebagai media dalam proses pengidentifikasian dan proses pengenalan pola citra tulisan tangan Arab.

b. Manfaat Praktis

Dari penelitian akan dihasilkan aplikasi untuk pengenalan dan sebuah sistem identifikasi arti tulisan Huruf Arab berbasis komputer dengan menerapkan metode *Kohonen Neural Network* sebagai metode dalam mengenali pola tulisan. Dengan dibuatnya aplikasi ini diharapkan akan mampu menciptakan sebuah sistem pengidentifikasian objek tulisan Huruf Arab pada citra yang kemudian dapat memberikan output berupa deskripsi tulisan huruf Arab yang diharap kelak akan dapat memberikan kontribusi terhadap perkembangan dunia Teknologi informasi khususnya dalam proses pengidentifikasian *image* dalam bidang pemrograman.

1.6 Sistematika Penulisan

Sistematika penulisan skripsi ini dengan melakukan pembagian bab sebagai berikut :

BAB I PENDAHULUAN

Pada bab ini membahas mengenal latar belakang penelitian, identifikasi masalah, tujuan penelitian, manfaat penelitian, batasan penelitian dan sistematika penyusunan laporan skripsi.

BAB II TINJAUAN PUSTAKA

Pada bab kajian pustaka berisi penelitian serta studi literature terkait judul skripsi yang akan dibangun beserta teori-teori yang menunjang penelitian ini, diantaranya adalah teori tentang pengolahan citra (*Image Processing*), *Feature Extraction, Feature Normalization* dan jaringan syaraf tiruan *Kohonen.*

BAB III ANALISIS DAN PERANCANGAN

Pada bab ini berisi analisa kebutuhan sistem aplikasi meliputi gambaran aplikasi yang akan dibangun penulis, desain *interface*, algoritma program, alur sistem program, serta penerapan algoritma *Kohonen Neural Network* pada aplikasi sistem.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan implementasi dari perancangan dan desain sistem pada bab sebelumnya. Serta analisis keberhasilan algoritma jaringan syaraf tiruan Kohonen untuk mengenali citra tulisan tangan.

BAB V KESIMPULAN DAN SARAN

Pada bab ini akan berisi kesimpulan hasil akhir dari penelitian yang telah dilakukan beserta saran dari penulis untuk penelitian selanjutnya terkait judul penelitian yang telah dibuat.

DAFTAR PUSTAKA

Seluruh bahan rujukan, studi literatur dan sumber referensi dalam proses pembuatan dan penulisan skripsi ini, dicantumkan dalam bab ini. Sumber-sumbernya adalah berupa buku baik secara *hardcover* (print / cetak) maupun buku elektronik (*e-book*) , jurnal-jurnal terkait dan referensi dari internet (*website*).

LAMPIRAN

Berisi data-data kelengkapan atau keterangan tertentu secara terkait yang berfungsi untuk melengkapi uraian laporan yang telah disajikan dalam seluruh laporan penelitian ini, di lampirkan dalam halaman lampiran ini.

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian tentang pengenalan dan pencocokan tulisan tangan telah banyak dilakukan dengan berbagai metode, misalnya penelitian tentang citra yang mengangkat objek berupa huruf lainnya yaitu penelitian tentang *pengenalan Huruf Jepang (Kana) Menggunakan metode Direction Feature Extraction dan Learning Vector Quantization (LVQ)* yang dibuat oleh (Wirayuda dan Wardhani, 2008).

Penelitian tentang pengenalan pola tulisan Arab telah dilakukan menggunakan metode *Hidden Markov Models* (HMMs) (AlKhateeb, Ren, Jiang & Al-Muhtaseb, 2011). Dalam metode ini terdapat 3 tahap proses : tahap proses normalisasi dan segmentasi teks, ekstraksi fitur dari masin-masing kata yang telah tersegmentasi, dan klasifikasi berdasarkan fitur. Untuk keperluan validasi metode ini digunakan database IFN/ENIT yang bersisi 32.492 kata Arab tulisan tangan. Untuk pengenalan pola huruf Arab juga pernah dilakukan berdasarkan fitur *diakritik* (Lutf You, Cheung & Philip Chen, 2014). Diakritik adalah tanda/garis yang ditambah dalam huruf Arab orisinal. Sedangkan untuk segmentasi diakritik digunakan algoritma *flood-fill based* dan *clustering based*. Denga pendekatan ini diperoleh tingkat keakuratan pengenalan tulisan tangan sebesar 98,73% terhadap database yang memuat 10 jenis-jenis huruf Arab.

Penelitian lain tentang pengenalan tulisan Arab adalah sistem Arabic Optical Character Recognition (AOCR). AOCR memiliki 5 tahapan proses yaitu : preprocessing, segmentasi, penipisan/Scalling, ekstraksi fitur dan klasifikasi (Supriana & Nasution, 2003).

Penelitian lainnya yang berkaitan dengan penelitian ini adalah pengenalan pola tulisan tangan teks Arab kursif. Sistem ini men*decompose* dokumen citra menjadi citra teks dan mengekstrak seperangkat fitur statistik sederhana dari celah yang sepanjang garis teksnya. Lalu memasukkan vektor fitur yang dihasilkan dengan metode *Hidden Markov Model Toolkit* (HTK). Metode HTK adalah *toolkit portable* untuk sistem pengenalan suara. Sistem yang diusulkan diterapkan pada data korpus yang mencakup teks Arab lebih dari 600 halaman A4 yang diketik dalam beberapa jenis huruf dari komputer (Khorsheed, 2007). Penelitian juga pernah dilakukan untuk memisahkan komponen-komponen dokumen tulisan tangan Arab sambung (Aouadi, Amiri & Echi, 2003), penelitian mengenai jumlah transisi pada ciri transisi dalam pengenalan pola tulisan tangan aksara Jawa nglegeno dengan Metode *Multiclass Support Vector Machines* (MSVM) (Nugraha & Widhiatmoko, 2012), penelitian tentang segmentasi karakter pada skrip Huruf Bali menggunakan metode *Canny edge Detection* (Dwi Putra & Prapitasari, 2011) dan penelitian tentang Pengembangan Aplikasi text Recognition dengan Klasifikasi Neural Network pada huruf Hijaiyah Gundul (Anif et al, 2013).

Terdapat penelitian pengenalan pola tulisan seperti pengenalan pola tulisan Jawa (Hanacaraka) yang dilakukan proses pembagian data pelatihan agar dapat membandingkan hasil pelatihan pada masing-masing bagian serta

membandingkan nilai keakuratan proses pengujian berdasarkan hasil pelatihan (Nazla Nurmila et al, 2010).

Penelitian lain yaitu Pengenalan tulisan tangan Huruf Alfabet menggunakan metode *Modified Direction Feature (MDF) dan Learning Quantization Feature (LVQ).* Penelitian ini dilakukan dengan mengambil 52 sample Huruf Alfabet, yakni 26 huruf besar dan 26 huruf kecil dan data citra tulisan tangan sebanyak 312 yang ditulis oleh 6 responden. Masing-masing responden menuliskan 26 citra huruf besar dan 26 citra huruf kecil. Uji coba diambil melalui data pelatihan dan data uji coba. Hasil dari uji coba menyatakan semakin banyak data pelatihan yang digunakan, semakin tinggi tingkat akurasi yang diperoleh dan dari segi penurunan nilai akurasi disebabkan oleh beberapa faktor seperti pola dan bentuk huruf yang hampir serupa (Hilyati Safitri et al, 2010).

2.2 Bahasa Arab

2.2.1 Pentingnya Belajar Bahasa Arab

Bahasa Arab merupakan bahasa Al-Quran yang haqiqi sebagai bahasa komunikasi dan informasi umat Islam. Bahasa Arab sangat penting dalam memahami semua pengertian, hukum-hukum dan definisi serta ilmu-ilmu dalam agama Islam. Alasan dikatakan seperti pernyataan diatas karena dominan buku-buku pelajaran tentang Islam baik itu tentang hokum, ajaran ilmu dasar Islam, perilaku dalam Islam yang menggunakan bahasa Arab sebagai isiannya. Jadi, wajib bagi kita umat Islam untuk mempelajari, memahami dan mengerti bahasa Arab dan huruf-huruf Arab yang menyusunnya.

Pernyataan yang sudah pasti bahwasanya bahasa Arab adalah bahasa yang dibuat dan dipilih oleh Allah SWT sebagai satu-satunya bahasa dalam kitab suci Al-Quran. Pernyataan tersebut didukung dan tertera dalam kitab suci Al-Quran yang menyatakan sumber utama pedoman Agama Islam adalah Al-Quran yang ditulis dalam huruf dan bahasa Arab, maka wajib bahasa Arab dijamin keutuhannya hingga akhir zaman, sesuai dengan sabda Allah SWT berfirman dalam Al-Quran :

إِنَّآ أَنزَلْنَٰهُ قُرْءَٰنًا عَرَبِيًّا لَّعَلَّكُمْ تَعْقِلُونَ ۝٣

Artinya: "*Sesungguhnya Kami menjadikan berupa Al Quran dalam bahasa Arab, agar kamu memahaminya".* (Q.S. Az-Zukhruf [43]:3)

Pada ayat tersebut yaitu Q.S.Az-Zukhruf ayat-3 yang menyatakan tentang wajibnya mengerti dan memahami bahasa Arab juga ditemui dalam Q.S.Yusuf ayat-2. Mengapa demikian, karena bahasa Arab memang hukumnya wajib dipelajari agar kita umat manusia dapat memahami tafsir dan makna dalam Al-Qur'an. Dalam ayat tersebut dijelaskan bahwa Allah SWT menurunkan Al-Quran dalam bahasa Arab bukan dalam bahasa *'Ajam* (Bahasa-bahasa yang bukan bahasa Arab) karena yang akan diberi peringatan pertama kali adalah orang-orang Arab agar mereka dapat dengan mudah belajar dan memahami isi dan kandungan yang terkandung dalam kitab suci Al-Quran. Allah SWT tidak menurunkan Al-Quran dalam bahasa *'Ajam* agar tidak ada alasan bagi mereka (orang-orang *'Ajam*) untuk mengatakan bagaimana dapat kita memahami isi dan kandungan makna dalam Al-Quran karena bahasanya bukan bahasa Arab, bahasa kami

(Depag RI, 2011). Sesuai dengan sabda dan Kallam Allah SWT pun berfirman dalam Al-Quran :

"Dan jikalau Kami jadikan Al-Quran itu suatu bacaan dalam bahasa selain Arab, tentulah mereka mengatakan: "Mengapa tidak dijelaskan ayat-ayatnya? Apakah (patut Al-Quran) dalam bahasa asing sedang (rasul adalah orang) Arab? Katakanlah: "Al-Quran itu adalah petunjuk dan penawar bagi orang-orang mukmin. Dan orang-orang yang tidak beriman pada telinga mereka ada sumbatan, sedang Al-Quran itu suatu kegelapan bagi mereka. Mereka itu adalah (seperti) yang dipanggil dari tempat yang jauh " (Qs. Fussilat(41):44)

Ayat tersebut merupakan jawaban dari pertanyaan orang-orang musyrik penduduk Arab yang menyatakan tentang pentingnya belajar bahasa Arab dalam menggali isi dan kandungan makna dalam Al-Quran. Lalu kepada mereka semua disampaikan: "Seandainya Kami menurunkan Al-Quran kepada engkau (Rasul) dengan salah satu bahasa selain bahasa Arab, tentulah orang-orang Makkah yang musyrik akan berkata "Mengapa Al-Quran tidak diturunkan dalam bahasa Arab?". Sehingga kami mudah memahami hukum-hukum, ajaran dan syariat-syariat yang ada dalam Al-Quran". Padahal dulu mereka berkata: "Apakah Al-Quran yang diturunkan itu berbahasa selain bahasa Arab, sedang Rasulullah yang diutus itu berbahasa Arab" (Depag RI, 2011).

Allah SWT menurunkan Al-Quran kepada Rasulullah dalam bahasa Arab yang nyata agar menjadi mukjizat yang kekal dan menjadi hidayah dan sumber hokum serta pedoman utama bagi seluruh umat Islam dalam berperilaku. Al-Quran juga merupakan merupakan mukjizat untuk mengeluarkan manusia dari kegelapan kepada cahaya dari kegelapan "syirik" menuju "tauhid", dari "kebodohan" menuju "pengetahuan" dan dari "kesesatan" menuju "hidayah".

Terdapat pula keutamaan-keutamaan bahasa Arab menurut Al-Quran yang juga dibagi menjadi beberapa point yakni :

1. Sebagai Sumber Informasi dan Sumber Ilmu, karena pada zaman dahulu terdapat sumber-sumber ilmu pengetahuan dan informasi yang ditulis dengan menggunakan bahasa Arab, misalnya yaitu buku-buku dan kitab-kitab, yang ayatnya terdapat dalam firman Allah SWT :

   كِتَٰبٌ فُصِّلَتْ ءَايَٰتُهُۥ قُرْءَانًا عَرَبِيًّا لِّقَوْمٍ يَعْلَمُونَ ﴿٣﴾

   "*Kitab yang dijelaskan ayat-ayatnya satu persatu, ialah Al-Quran yang diturunkan dalam bahasa Arab, untuk kaum yang mengetahui dan memahami kandungannya.*" (Qs.Fussilat(41):3)

2. Berfungsi sebagai faktor pencetus kecerdasan / kepintaran seseorang (intelligensi), karena bahasa Arab itu sendiri adalah bahasa nyata dan satu-satunya bahasa Al-Qur'an yang akan mengeluarkan manusia dari kegelapan "kebodohan" menuju cahaya "pengetahuan" yang ayatnya terdapat dalam firman Allah SWT :

   إِنَّآ أَنزَلْنَٰهُ قُرْءَٰنًا عَرَبِيًّا لَّعَلَّكُمْ تَعْقِلُونَ ﴿٢﴾

Artinya: "*Sesungguhnya Kami menurunkannya berupa Al Quran dengan berbahasa Arab, agar kamu memahaminya".* (Q.S. Yusuf [12]:2)

3. Berfungsi sebagai kecerdasan Spiritual dan keagungan Akhlak, karena bahasa Arab adalah cikal bakal mampunya seseorang dalam memahami dan mengerti ilmu akidah, perilaku, hukum dan syariat dalam Al-Qur'an yang ayatnya terdapat dalam firman Allah SWT :

قُرْءَانًا عَرَبِيًّا غَيْرَ ذِي عِوَجٍ لَّعَلَّهُمْ يَتَّقُونَ ﴿٢٨﴾

"(Ialah) Al-Quran dalam bahasa Arab yang tidak ada kebengkokan (didalamnya) supaya mereka bertakwa." (Qs. Az-Zumar(39):28)

Maka dari itu tidaklah sempurna mempelajari Al-Quran tanpa adanya pengetahuan dalam bahasa Arab yang dalam Al-Quran berisi seluruh seluk beluk tentang syariat, hukum, akidah agama Islam.

2.2.2 Urgensi Teknologi Dalam Pembelajaran Bahasa Arab

Pembelajaran bahasa Arab wajib dipelajari oleh semua umat Islam dimanapun, kapanpun dan siapapun serta dalam media apapun. Media juga sangat berperan penting dalam menunjang pembelajaran dan pemahaman terhadap bahasa Arab. Keefektifan dan keefisienan media juga patut dipertimbangkan dan diselaraskan guna agar seseorang dapat memahami dan belajar bahasa Arab dengan mudah. Salah satu media yang akhir-akhir ini menjadi trend dunia adalah media dari sisi sains dan teknologi.

Teknologi akhir-akhir ini sangat berperan penting bagi manusia dalam beraktifitas. Dengan teknologi, manusia dapat berinteraksi dengan manusia lain

tanpa mengenal batas ruang dan waktu. Dengan teknologi pula semua hal menjadi mudah untuk dilakukan, salah satunya adalah proses belajar-mengajar. Di zaman modern seperti sekarang ini dengan kecanggihan teknologi, manusia dapat belajar berbagai ilmu melalui media teknologi, semisal mengakses informasi melalui internet (*browsing*), proses belajar menggunakan perangkat elektronik dan belajar melalui dunia maya (*e-learning*). Memanfaatkan media teknologi untuk Belajar bahasa Arab adalah sesuatu yang patut untuk dicoba. Tujuannya adalah untuk memudahkan mempelajari dan memahami huruf-huruf Arab dan Bahasa Arab dengan cara efektif dan efisien.

2.3 Sistem Pengenalan Biometrika

Sistem pengenalan biometrika merupakan sistem otentikasi dengan menggunakan biometrika secara otomatis atas identitas seseorang berdasarkan suatu ciri bionetrika dengan mencocokan ciri tersebut dengan ciri biometrika yang telah disimpan dalam database. Secara umum terdapat 2 model sistem pengenalan biometrika, yaitu sistem verifikasi dan sistem identifikasi (Putra, 2009: 24-25).

a. Sistem Verifikasi

   Sistem verifikasi bertujuan untuk menerima atau menolak identitas yang diklaim oleh seseorang. Diperlukan pencocokan "satu ke satu" dari sampel yang diberikan terhadapa acuan (*template*) yang terdaftar atas identitas yang diklaim tersebut. Sistem verifikasi akan menjawab pertanyaan "apakah identitas saya sama dengan identitas yang saya klaim?".

b. Sistem Identifkasi

Sistem identifikasi bertujuaan untuk memecahkan identitas seseorang. Pengguna dapat tidak memberi klaim atau memberi klaim implisit negative untuk identitas terdaftar. Diperlukan pencocokan "satu ke banyak" yaitu pencarian ke seluruh database identitas terdaftar. Sistem identifikasi akan menjawab pertanyaan "identitas siapkan ini?".

Sehingga sistem yang akan dibangun pada tugas akhir ini adalah sistem yang mampu memecahkan identitas seseorang dengan metode pencocokan "satu ke banyak".

Untuk kerja sistem biometrika ditinjukkan oleh 3 hal, yakni :

a. Akurasi (*accuracy*), akurasi berhubungan dengan sejauh mana sistem mapu mengenali secara benar biometrika yang diuji.
b. Kecepatan (*speed*) kecepatan berhubungan dengan lama proses yang dibutuhkan untuk mengenali biometrika yang diuji.
c. Media penyimpan (*storage*), kebutuhan media penyimpanan berhubungan dengan sejauh mana sistem mampu menghemat media penyimpanan untuk menyimpan data-data acuan.

2.4 Pengenalan Pola

Pengenalan Pola juga disebut dengan *Pattern Recognition* adalah sebuah bagian dari suatu disiplin ilmu yaitu *Kecerdasan Buatan* yang mengklasifikasikan objek berdasarkan gambar (*image*) atau parameter-parameter yang telah ditentukan ke dalam kelas-kelas tertentu atau sejumlah kategori yang sudah dispesifikasikan sebelumnya berdasarkan ciri-ciri tertentu (Duda et al, 2001).

Pengenalan pola juga berkaitan dengan hasil pengukuran terhadap suatu objek (Schalkoff, 1991). Secara Umum definisi dari Pengenalan Pola yaitu suatu bagian dari disiplin ilmu Kecerdasan Buatan yang bertumpu pada metode pengklasifikasian objek atau parameter ke dalam kelas-kelas tertentu guna menyelesaikan suatu permasalahan tertentu dengan menggambarkan atau mendeskripsikan suatu fitur yang merupakan ciri utama dari suatu objek (sekumpulan bilangan-bilangan) dalam suatu kelas yang dinilai tepat. Pendeskripsian fitur ini dilakukan berdasarkan pengukuran kuantitatif.

Pengenalan Pola dapat dibedakan menjadi pengenalan *objek konkrit* dan pengenalan *objek abstrak*. Pengenalan *objek konkrit* misalnya Pengenalan pola karakter digital yang akan diterapkan dalam penelitian ini. Dalam sistem Pengenalan Pola ada beberapa pendekatan yang sering dipakai yaitu *statistik*, *sintaksis*, dan *JST* (Jaringan Syaraf Tiruan).

2.5 Pengenalan Pola Tulisan Tangan (*Handwritting Recognition)*

Pengenalan pola Tulisan Tangan adalah salah satu fitur dari Pengenalan Pola. Definisi Pengenalan Pola Tulisan Tangan secara umum yaitu Pengenalan yang mampu mengenali tulisan tangan atau mampu mengenali suatu bentuk karakter pada komputer. Pengenalan pola Tulisan Tangan juga bisa didefinisikan sebagai proses perubahan suatu bahasa yang dihadirkan dalam bentuk ruang melalui tulisan menjadi representasi simbolik (Fuu Lin dan Yu Wei, 2002).

Tahap-tahap dalam Pengenalan Pola Tulisan Tangan terdiri dari beberapa tahap seperti tahapan : *Data Acquistion* (memperoleh data) → *Data*

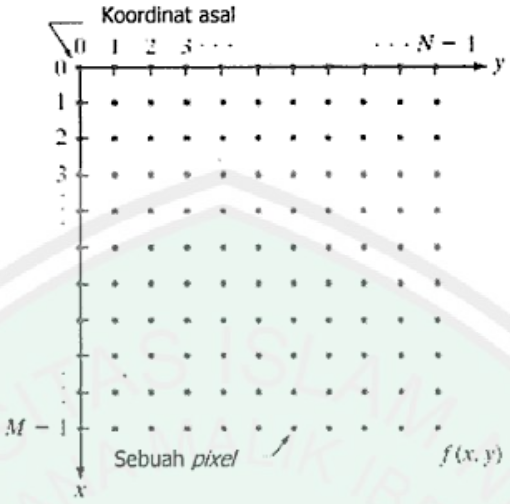
PreProcessing (pemrosesan awal) → *Segmentation* (Segmentasi) → *Feature Extraction* (ekstraksi fitur / ciri) → *Classification* (Pengklasifikasian) → *Handwrite Recognition* (Pengenalan Tulisan Tangan) (Cheriet et al, 2007).

1. P*reprocessing* yaitu proses untuk mengurangi noise (derau) pada citra tulisan dan proses normalisasi ukuran dan bentuk huruf supaya memperoleh akurasi tinggi.
2. *Segmentation* yaitu proses memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertetu. Tujuannya adalah untuk menyederhanakan suatu citra / gambar menjadi sesuatu yang lebih mudah dalam menganalisanya.
3. *Feature Extraction* yaitu proses untuk mendapatkan karakteristik pembeda yang mewakili sifat utama dengan melakukan proses pemisahan dari fitur yang tidak dibutukan untuk klasifikasi dan biasanya terdapat proses reduksi dimensi citra untuk mengurangi kerumitan komputasi pada tahapan klasifikasi juga berupaya untuk memperoleh akurasi yang tinggi.
4. *Classification* adalah proses pengklasifikasian / mengelompokkan fitur ke dalam kelas - kelas yang dinilai tepat / cocok dengan Algoritma klasifikasi tertentu. Hasilnya berupa klasifikasi objek kedalam beberapa kriteria yang telah ditentukan.
5. *Handwrite Recognition* adalah proses akhir dari tahapan sistem yang nantinya akan keluar sebuah output dari klasifikasi yaitu sebuah karakter yang unik dari kelas-kelas dengan nilai akurasi tertentu dari setiap masing-masing karakter.

2.6 Pengolahan Citra

Citra merupakan informasi yang tersimpan dalam bentuk pemetaan bit-bit atau dikenal dengan bitmap. Setiap bit-bit membentuk satu satuan terkecil informasi yang disebut piksel. Satuan dari piksel biasanya dinyatakan dengan posisi *x*, posisi *y*, dan nilai dari piksel. Dalam satu gambar, sepenuhnya terdiri dari piksel-piksel. Pada pemrosesan citra digital sebuah gambar bilangan array 2 dimensi, yang setiap barisnya adalah representasi piksel pada gambar setiap barisnya. Minimal nilai piksel adalah bernilai 0 untuk warna hitam dan maksimum bernilai 1 untuk warna putih.

Menurut (Darma Putra : 2010), gambar digital ialah representasi dari gambar dua dimensi di dalam komputer. Gambar digital merupakan sebuah larik (*array*) yang berisi nilai-nilai riil atau kompleks yang direpresentasikan dengan deretan *bit* tertentu. Suatu gambar dapat didefinisikan sebagai fungsi (*x*,*y*) berukuran *M* baris dan *N* kolom, dengan *x* dan *y* adalah koordinat spasial, dan *f* di titik koordinat (*x*,*y*) sebagai intensitas dari gambar tersebut dengan nilai *x*, *y*, dan *f* berhingga.



Gambar 2.1. Koordinat Gambar Digital
((Sumber: Putra, Darma (2010,p20)

Pengolahan gambar atau pengolahan citra atau yang biasa disebut *Image Processing* adalah merupakan suatu proses yang mengubah dan mengolah sebuah gambar / citra menjadi gambar lain yang memiliki kualitas lebih baik untuk tujuan tertentu (Ardhianto et al, 2011). Menurut (Pratiarso et al, 2009), sesuai dengan perkembangan komputer , pengolahan citra mempunyai beberapa tujuan utama, yaitu sebagai berikut :

a. Memperbaiki kualitas citra, dimana citra yang dihasilkan dapat menampilkan informasi secara lebih jelas dan rinci. Dalam hal ini peran manusia sebagai alat untuk mengolah informasi.

b. Mengekstraksi informasi ciri yang menonjol / dominan pada suatu gambar / citra, dari proses ekstraksi tersebut akan dihasilkan informasi citra dimana manusia mendapat informasi ciri / feature dari citra secara numerik / digital.

Pada pengolahan citra juga diterapakan beberapa proses – proses yang bertujuan untuk perbaikan kualitas citra, misalnya peningkatan kualitas penampakan citra dengan cara perbaikan kontras, perbaikan tepi, *thinning*, *grayscale*, *threshold*, dll. Proses lainnnya yaitu meminimalkan cacat pada citra *(image restoration)* contohnya menghilangkan kesamaran *(debluring).* Proses yang kebanyakan sering digunakan dalam proses pengolahan citra lainnya yaitu pengelompokkan, pencocokan dan mengukur elemen citra yang biasanya digunakan dalam kasus pengenalan pola yaitu *Image Segmentation*. Proses lainnya adalah ekstraksi fitur / ciri pada citra untuk membantu pengidentifikasian objek *(image analysis)* yaitu pendeteksian tepi objek *(edge detection).*

2.7 Kohonen Neural Network

Jaringan syaraf tiruan (JST) adalah representasi buatan dari sistem kerja otak manusia yang mencoba untuk mensimulasikan proses pembelajaran pada otak manusia. Jelasnya, konsep dasar Jaringan Syaraf Tiruan adalah menirukan sel-sel otak manusia dalam proses berpikir. Pada jaringan syaraf tiruan neuron-neuron akan dikumpulkan dalam lapisan-lapisan (layer) yang disebut lapisan neuron (neuron layer). Neuron-neuron pada satu lapisan masukan (*input)* akan mempunyai keluaran berupa lapisan-lapisan keluaran *(output).* Informasi yang diberikan pada jaringan syaraf akan dirambatkan pada *layer input* ke *layer output* (Kusumadewi, 2004: 175).

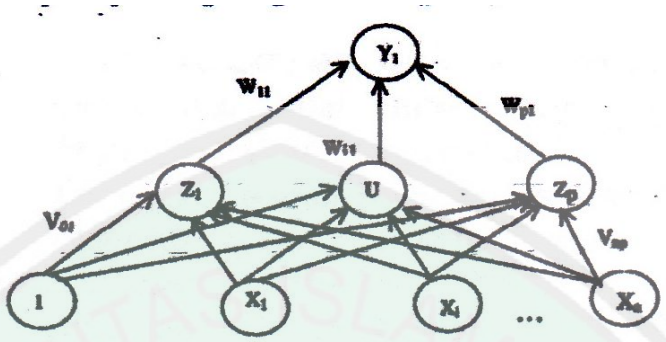
Teknik *Self – organizing map* (SOM) pertama kali dikenalkan oleh *Teuvo Kohonen* dari University of Helsinki pada tahun 1981, merupakan proses

unsupervised learning (pelatihan tak terbimbing) yang mempelajari distribusi himpunan pola – pola tanpa informasi kelas (Kusumadewi, 2003). Teknik ini didasari dengan pemisalan dari bagaimana proses memori otak manusia menyimpan gambar / pola yang pernah dikenali melalui indera penglihatan (mata), lalu mampu mengungkapkan kembali gambar / pola tersebut. Oleh karena itu Model salah satu jenis JST (Jaringan Syafar Tiruan) ini yaitu *Kohonen Neural Network* banyak digunakan pada pengenalan pola objek citra / image. Jaringan Kohonen SOM terdiri dari 2 lapisan (layer), yaitu layer input dan layer output.

Jaringan *Kohonen Neural Network* / Teknik *Self – organizing map* (SOM) ini mengasumsikan sebuah struktur topologi antar unit – unit cluster. Misalkan masukan berupa vektor yang terdiri dari n komponen yang akan dikelompokkan dalam maksimum m buah kelompok (vektor contoh). Keluaran jaringan adalah kelompok yang paling dekat / mirip dengan masukan yang diberikan. Ada beberapa ukuran kedekatan yang dapat dipakai, salah satunya adalah *jarak Euclidean* yang paling minimum. Bobot vektor – vektor contoh berfungsi sebagai penentu kedekatan vektor contoh tersebut dengan masukan yang diberikan (Heaton, 2003). Selama proses pengaturan, vektor contoh yang pada saat itu paling dekat dengan masukan akan muncul sebagai pemenang. Vektor pemenang (dan vektor – vektor sekitarnya) akan dimodifikasi bobotnya.

Secara matematis konsep dasar JST adalah dengan cara menjumlahkan hasil kali dari nilai masukan dengan nilai bobotnya. Gambar 2.2 Memperlihatkan serangkaian X1 …….Xn. setiap masukan akan dikalikan berturut-turut dengan bobot W1…….Wn dengan demikian hasil kali keluaran akan sama dengan :

$$Y = X_1 . W_1 + X_2 . W_2 + \ldots + X_n . W_n \quad \text{(Wijaya, 2004: 48)}$$



Gambar 2.2 Skema Jaringan Syaraf Tiruan

Algoritma dari Jaringan syaraf Tiruan *Kohonen Neural Network / Self – organizing map* (SOM) didefinisikan sebagai berikut :

a. Inisialisasi

- bobot w_{ij} (acak)
- Set radius tetangga / faktor penurunan (R) dan set parameter *learning rate* / laju pemahaman (α)

b. Selama kondisi berhenti masih bernilai salah, lakukan langkah-langkah c - i

c. Untuk setiap vektor masukan x (xi, i = 1,....,n) , lakukan langkah d – f

d. Untuk tiap j (j = 1,.....,m) / (menghitung output pemenang), Hitung :

$$D(j) = \sum_i (w_{ij} - x_i^2)$$

e. Tentukan indeks J sedemikian hingga D(J) bernilai minimum

f. Untuk setiap unit j di sekitar J modifikasi bobot :

$$W_{ij} \text{ (baru)} = w_{ij} \text{ (lama)} + \alpha [\, x_i - w_{ij} \text{ (lama)} \,]$$

g. Modifikasi / perbaiki *learning rate*

$$\alpha (t + 1) = 0{,}5\ \alpha t$$

h. Mereduksi radius dari fungsi tetangga pada waktu tertentu (*epoch*)

i. Uji kondisi berhenti (*error bernilai min atau epoch max terpenuhi*)

2.8 RUANG LINGKUP APLIKASI

Adapun ruang lingkup yang dikerjakan dalam penelitian-penelitian ini adalah pembuatan aplikasi untuk sistem pengenalan dan identifikasi tulisan huruf Arab berbasis komputer dengan menerapkan metode *Kohonen Neural Network* dengan komponen kegiatan sebagai berikut :

a. Pembuatan aplikasi program yang didesain untuk dijalankan di komputer dengan merujuk pada materi **Studi Literatur** :

- Pengolahan citra digital 2 dimensi dan teknik pemrograman dengan bahasa Pemrograman Java.
- Metode Ekstraksi *Feature Extraction* dan *Pixel Grabber*, untuk mendeteksi titik dan *scanning* lokasi keberadaan citra pada citra 2 dimensi.
- Pengenalan pola tulisan tangan yang mengacu pada pengenalan tulisan huruf Arab yang bisa mengenali huruf Alif sampai Ya'.
- Aplikasi identifikasi tulisan berbasis pengenalan pola ini dibuat untuk bisa mengenali huruf Arab dengan berbasis tulisan tangan langsung (*real time*), inputan diperoleh melalui goresan tulisan tangan langsung dari pengguna ke layar komputer, outputnya akan langsung diperoleh

setelah sistem memprosesnya dan hasilnya akan tampil pada layar komputer berbetuk GUI interface.

- Algoritma Jaringan Syaraf Tiruan *Kohonen Neural Network* dan parameter-parameter yang digunakan.
- Komponen yang digunakan dalam aplikasi ini terdiri dari teks dan gambar.
- Output dari sistem adalah identifikasi tulisan huruf Arab Alif sampai Ya'.
- Kajian Islam tentang pengenalan Pola Tulisan Tangan.

b. **Pengumpulan data**, yaitu data masukan berupa beberapa sample data tulisan tangan huruf Arab Alif sampai Ya' yang ditulis langsung pada *panel Drawing* secara *online* untuk keperluan *training* terhadap sistem yang telah dibangun. Kegiatan ini dilakukan dengan cara melakukan *sampling* terhadap beberapa kategori kata citra tulisan tangan huruf Arab, kemudian dilakukan proses *scanning*, selanjutnya akan di lakukan *training* (pelatihan) terhadap sistem.

c. **Analisa data**, dilakukan untuk menganalisa proses *training* yang telah diberikan dengan cara melakukan uji coba langsung sistem dengan cara menuliskan langsung ke layar komputer oleh pengguna. Pengujian tersebut dilakukan secara berulang untuk memperoleh ketepatan sistem terhadap input tulisan tangan yang dilakukan.

d. **Pembuatan Aplikasi**, desain sistem diimplementasikan ke dalam sebuah *source code* dengan bahasa Pemrograman Java. Penulisan *source code* Java

dengan menggunakan editor Netbeans (IDE). Sedangkan untuk penyimpanan data menggunakan sample data yaitu *sample.dat*

e. **Pembuatan laporan**, bertujuan untuk mengarsip / membukukan dan mencatat hasil akhir dan temuan-temuan selama proses penelitian dikerjakan.

BAB III

ANALISA DAN PERANCANGAN

3.1 Gambaran Umum Sistem

Input dari sistem yang akan dibangun adalah citra goresan tulisan tangan huruf Arab secara *online* yang akan dicocokkan dengan matriks citra sample Huruf Arab pada *sample.dat* dan dinamakan *citra sample*. Selanjutnya, sistem akan mengidentifikasi citra goresan tulisan tangan huruf Arab *online* yang ada di dalam *sample.* Jika sistem mampu mengidentifikasi citra *sample,* maka output dari sistem ini adalah identifikasi huruf Arab yang dimaksud. Jika tidak mampu, sistem akan memunculkan identifikasi huruf Arab yang tidak sesuai / tidak cocok dengan inputan huruf Arab yang dimaksud dan ditarik kesimpulan bahwasanya citra *sample* tidak bisa dikenali.

Sistem identifikasi huruf Arab terdiri atas modul-modul berikut:

a. Modul pengambilan sample, dibutuhkan untuk memuat file tulisan tangan Huruf Arab bernama *"sample.dat"*. Hasil dari modul ini adalah daftar sample tulisan tangan huruf Arab Alif sampai Ya' yang tampil dan telah dilatih *(training).* File pelatihan ini berisi sample huruf Arab Alif sampai Ya' dengan berbagai versi input tulisan.
b. Modul Pelatihan, Proses pelatihan dapat berlangsung dalam beberapa detik hingga beberapa menit tergantung kecepatan performa komputer. Dalam modul pelatihan akan digunakan data training sejumlah 280 data input tulisan huruf Arab. Dalam sistem aplikasi juga akan digunakan data testing sejumlah

140 data input tulisan tangan huruf Arab yang berbeda (tidak termasuk dalam data training). Selain data training dan data testing, data lain yang menjadi parameter input dalam pengolahan data menggunakan JST *Kohonen* adalah data target output. Jumlah data target output sama dengan jumlah data sample pola huruf Arab.

c. Modul Masukan Input Tulisan, yang dibutuhkan sistem untuk memproses citra huruf Arab, sehingga menghasilkan fitur-fitur dan menyimpanya ke dalam database.

d. Modul Proses, digunakan untuk melihat detail proses yang dilakukan pada proses pencocokan. Input gambar akan tampil dalam sebuah grid yang disebut *downsample*. *Downsample* ini berupa kotak kecil dengan lebar pixel tujuh dan tinggi pixel sembilan (7x9) yang berfungsi untuk menampilkan input gambar tulisan huruf Arab yang dipetakan. Downsample ini juga berfungsi ganda, pertama gambar dengan resolusi rendah membutuhkan neuron input yang kurang memenuhi batasan ukuran, akan memenuhi ukuran secara penuh. Kedua, dengan *downsample* input gambar akan bernilai konstan, baik gambar tersebut terlalu besar atau terlalu kecil, gambar akan tetap dikenali, atau dengan kata lain ukuran pola akan disesuaikan.

e. Modul pencocokan dan Pengenalan, digunakan untuk proses mengenali tulisan. Pada modul ini menggunakan jaringan syaraf tiruan jenis *Kohonen* untuk mencocokkan.

f. Modul setting, untuk menentukan parameter-parameter yang digunakan pada algoritma *Kohonen* .

3.2 Alat dan Bahan

Kebutuhan alat dan bahan untuk melakukan penelitian ini adalah sebagai berikut:

a. Kebutuhan Hardware

Sebuah computer PC/laptop untuk melakukan perancangan dan pembangunan sistem dengan spesifikasi sebagai berikut:

- Prosesor Intel(R) Core(TM) i3
- Memory minimal 1 GB
- Primary Hardisk Total 43 GB (Free harddisk Minimal 4 GB)

b. Kebutuhan Software

Selain kebutuhan hardware penulis juga membutuhkan kebutuhan software untuk melakukan perancangan dan pembuatan sistem. Adapun software tersebut adalah:

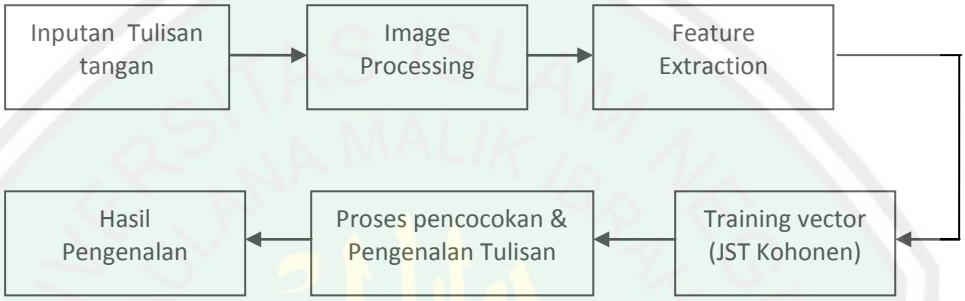
- Netbeans untuk melakukan penulisan *source code* program
- *Sample.dat* untuk melakukan penyimpanan data ciri tulisan tangan
- Microsoft Office untuk membuat dokumentasi dan laporan hasil penelitian

c. Kebutuhan Data

Data tulisan tangan penulis masukan berupa beberapa sample data tulisan tangan huruf Arab Alif sampai Ya' yang ditulis langsung pada *panel Drawing* secara *online* untuk keperluan *training* terhadap sistem yang telah dibangun.

3.3. Perancangan dan Alur Sistem

Perancangan dan alur sistem akan memberikan gambaran yang lebih detail dari aplikasi yang akan dibangun. Pada sub bab ini akan dijelaskan rancangan sistem yang akan dibangun yang ditunjukkan pada Gambar 3.3.

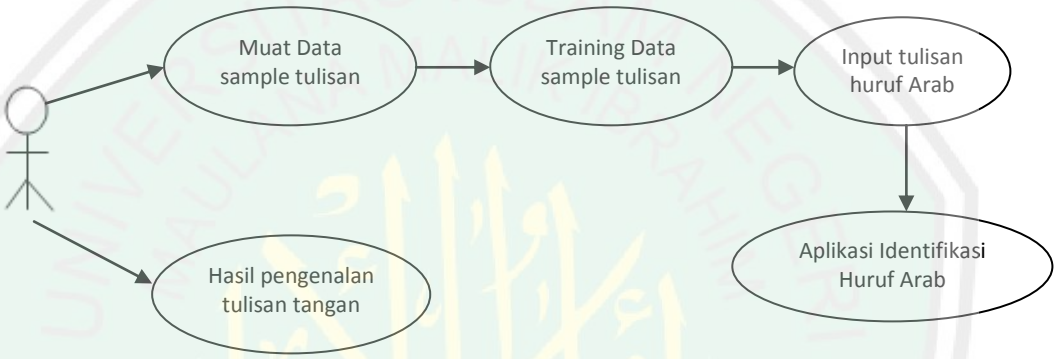


Gambar 3.3. Blok Diagram sistem

Pada gambar blok diagram 3.3. dijelaskan bahwa sistem dimulai dengan menginputkan citra tulisan tangan huruf Arab *online* yang akan di-*convert* ke dalam format berekstensi .JPEG melalui fase pemrosesan gambar (*Image Processing*). Kemudian citra tulisan tangan huruf Arab dirubah ke dalam format biner hitam–putih yaitu bernilai 1 dan 0, dan kemudian masuk ke dalam fase deteksi titik (*dot detection*) dan pencarian ruang dalam citra baik itu terdapat celah spasi atau batas citra tulisan *(bounds*) yang diproses menggunakan metode *Pixel Grabber*. Metode *Pixel Grabber* adalah salah satu Algoritma Pencarian untuk menghasilkan matriks *pixel*. Gambar dari karakter akan menghasilkan batas. Selanjutnya adalah mengkombinasikan fitur dari normalisasi koordinat (x , y) dan representasi biner dari gambar yang disajikan ke input jaringan. Citra tulisan tangan huruf Arab kemudian di *crop* dengan ukuran lebar citra 7 dan tinggi citra 9. Hasil dari Ekstraksi Fitur adalah ciri tulisan tangan huruf Arab yang kemudian

disimpan dalam database untuk digunakan sebagai inputan pada jaringan syaraf tiruan *Kohonen*. Pada saat yang bersamaan sistem juga melakukan inisialisasi target pada jaringan syaraf tiruan *Kohonen*.

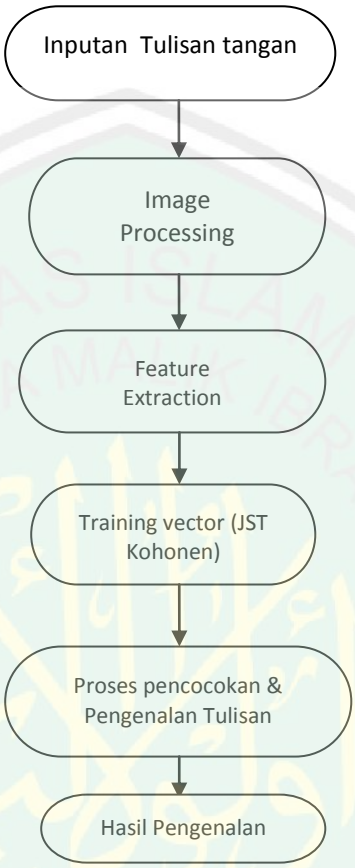
3.3.1 *Diagram* Penggunaan Aplikasi Identifikasi



Gambar 3.3.1. Diagram Penggunaan Aplikasi Identifikasi

Gambaran umum penggunaan aplikasi Identifikasi dijelaskan pada Gambar 3.3.1. diawali dengan *user* melakukan aksi muat data sample tulisan melalui sebuah aksi tombol. Setelah itu *user* juga diharuskan melakukan aksi *Training* Data sample Tulisan melalui sebuah aksi tombol. Setelah itu menginputkan tulisan huruf Arab melalui sebuah *Panel Canvas Drawing* kepada sistem. Input tulisan Huruf Arab yang diinput harus merujuk dari sample Data dari Pembuat Sample data tulisan tangan (*owner*). Kemudian sistem memproses inputan tulisan tangan *online* tersebut dan mengembalikan *output* berupa hasil pengenalan huruf Arab jawaban sesuai dengan inputan yang telah diinputkan oleh *user*.

3.3.2 Alur & Desain Sistem Aplikasi Identifikasi



Gambar 3.3.2. Desain sistem Aplikasi Identifikasi

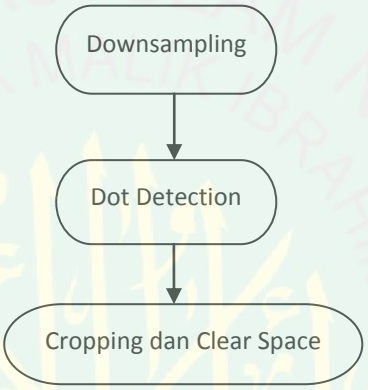
Pada desain sistem Gambar 3.3.2. dijelaskan bahwa sistem dimulai dengan menginputkan citra tulisan tangan huruf Arab *online* pada layar *screen*, kemudian inputan tulisan akan di-*convert* ke dalam format berekstensi .JPEG melalui fase pemrosesan gambar (*Image Processing*). Kemudian citra tulisan tangan huruf Arab dirubah ke dalam format biner hitam–putih yaitu bernilai 1 dan 0, dan kemudian masuk ke dalam fase Ekstraksi Fitur yaitu deteksi titik (*dot detection*) dan pencarian ruang dalam citra baik itu terdapat celah spasi atau batas citra tulisan *(bounds*) yang diproses menggunakan metode *Pixel Grabber*. Metode

Pixel Grabber adalah salah satu Algoritma Pencarian untuk menghasilkan matriks *pixel*. Gambar dari karakter akan menghasilkan batas. Selanjutnya adalah mengkombinasikan fitur dari normalisasi koordinat (x , y) dan representasi biner dari gambar yang disajikan ke input jaringan. Hasil dari Ekstraksi Fitur adalah ciri tulisan tangan huruf Arab yang kemudian disimpan dalam database untuk digunakan sebagai inputan pada jaringan syaraf tiruan *Kohonen*. Pada saat yang bersamaan sistem juga melakukan inisialisasi target pada jaringan syaraf tiruan *Kohonen*.

Pada fase ini akan terdapat fase *training* fitur vektor menggunakan metode *Kohonen Neural network*. Terdapat beberapa langkah yang terlibat dalam proses *Training* / pelatihan ini. Secara keseluruhan proses untuk pelatihan jaringan syaraf *Kohonen* melibatkan langkah-langkah yang melalui beberapa *epochs* sampai kepada tingkat kesalahan *(error)* bernilai minimal yang dapat diterima oleh jaringan saraf *Kohonen*. *Epochs* adalah suatu jangka waktu, dapat diartikan satu set putaran vektor-vektor pelatihan. Beberapa *epoch* diperlukan untuk pelatihan sebuah JST *Kohonen* sehingga kesalahan mendekati nol. Tingkat kesalahan (*error*) ini akan dihitung oleh jaringan saraf *Kohonen* dan dari proses tersebut akan dilakukan proses kalkulasi bobot matriks untuk setiap *epochs*. Untuk setiap pelatihan akan menetapkan satu neuron yang akan dikategorikan sebagai neuron "pemenang". Neuron pemenang ini akan memiliki bobot yang disesuaikan, sehingga akan bereaksi lebih kuat terhadap input waktu berikutnya. Sebagai neuron yang dianggap "menang" ini akan memiliki suatu pola yang berbeda dan akan dapat mengenali pola tertentu yang semakin lama semakin meningkat. Dari

bobot suatu neuron ini, maka proses pencocokan dan pengenalan karakter tulisan tangan huruf Arab berjalan. Akhirnya, akan menghasilkan suatu *output* hasil karakter huruf Arab yang dimaksud.

3.3.3 Alur Sistem (*Image Processing*)



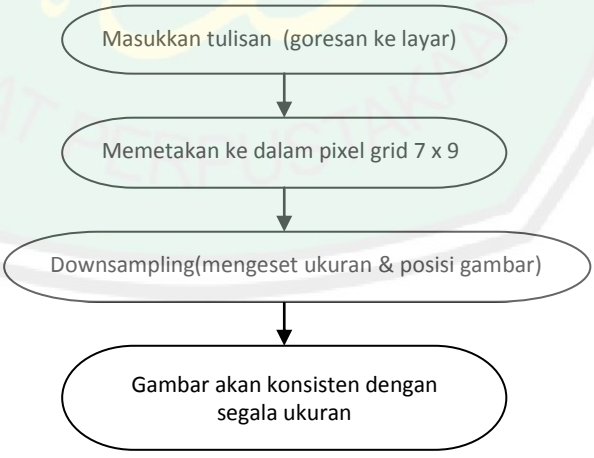
Gambar 3.3.3. Alur sistem (*Image Processing*)

Fase *Image Processing* yang ditunjukkan pada Gambar 3.3.3. adalah fase pengolahan suatu inputan tulisan tangan yang diinputkan oleh user. Hasil dari fase ini adalah suatu gambar 2 dimensi (*image*) yang prosesnya adalah meng-*convert* inputan tulisan tangan *online* menjadi gambar berformat .jpeg melalui pemrosesam gambar (*Image Processing*). Fase ini terdiri dari *Downsampling, Dot Detection, Cropping* dan *Clear Space.*

3.3.4 Alur sistem *Downsampling*

Downsampling adalah tahap untuk pemrosesan dimana setiap input tulisan di-*entry*-kan ke dalam panel area *screen* layar pada sistem. Setiap kali tulisan yang

diambil baik untuk proses pelatihan atau proses pengenalan, harus melalui fase *downsampling*. Fungsi dari *downsampling* itu sendiri adalah memetakan gambar ke dalam *pixel* grid yang telah ditentukan oleh sistem untuk proses identifikasi. Dalam hal ini sistem grid yang disediakan adalah dengan lebar *pixel* 7 dan tinggi *pixel* 9. Pemetaan gambar ini untuk menetralkan resolusi gambar yang terlalu tinggi dan sulit dipahami oleh jaringan syaraf. Keuntungan dari penggunaan tahap *downsampling* adalah gambar dengan resolusi rendah yang kekurangan neuron dan membutuhkan *input* neuron, akan terpenuhi oleh neuron secara otomatis dan keuntungan lainnya adalah dengan *downsampling* ukuran gambar input akan bernilai konstan. Gambar yang berukuran besar atau kecil akan tetap bisa dilatih dan dikenali. Dengan kata lain, ukuran pola gambar akan dinetralkan. Proses *downsampling* ditunjukkan pada Gambar 3.3.4. serta penerapannya dalam source code dapat dilihat pada Gambar 3.3.5.



Gambar 3.3.4. Alur sistem (*Downsampling)*

```
if ( entryImage==null )
 initImage();
g.drawImage(entryImage,0,0,this);
g.setColor(Color.black);
g.drawRect(0,0,getWidth(),getHeight());
g.setColor(Color.red);
g.drawRect(downSampleLeft,downSampleTop,downSampleRight-
            downSampleLeft,downSampleBottom-downSampleTop);        }
 ratioX =(double)(downSampleRight-downSampleLeft) / (double)data.getWidth();
 ratioY = (double)(downSampleBottom-downSampleTop)/ (double)data.getHeight();
 for ( int y=0;y<data.getHeight();y++ ) {
 for ( int x=0;x<data.getWidth();x++ ) {
 if ( downSampleQuadrant(x,y) )
 data.setData(x,y,true);
 else
 data.setData(x,y,false);
```

Gambar 3.3.5. Source code proses (*Downsampling)*

Semua gambar input tulisan yang masuk akan melalui proses *Downsampling.* Hal ini bertujuan untuk mencegah jaringan syaraf dari kebingungan dengan ukuran dan posisi gambar. Area input dan proses untuk input gambar yang cukup besar akan menampung tulisan untuk ukuran tulisan yang berbeda-beda. Dengan *downsampling* gambar akan konsisten baik gambar yang diinputkan berukuran besar atau kecil, gambar akan tetap konsisten.

3.3.5 Alur sistem *Dot Detection*

Semua gambar yang masuk akan di *downsample* sebelum diproses lebih lanjut. Area input dan proses untuk input gambar yang cukup besar akan menampung tulisan untuk ukuran tulisan yang berbeda-beda. Dengan *downsampling* gambar akan konsisten baik gambar yang diinputkan berukuran besar atau kecil, gambar akan tetap konsisten. Tahap selanjutnya adalah mendeteksi titik. Ketika *user* menggambar sebuah gambar input tulisan, hal pertama yang dilakukan adalah program menarik kotak (grid) di sekitar batas

tulisan dari *user*. Hal ini menyebabkan sistem untuk menghilangkan semua ruang putih (*pixel nonaktif*) di sekitar tulisan dari *user*. Dalam hal ini sistem membuat parameter-parameter batas tulisan dai arah atas, bawah, kanan dan kiri yaitu *downsampleTop, downsampleBottom, downsampleRight* dan *downsampleLeft* serta mengatur batas lebar (*width*) dan tinggi (*height*) tulisan. Dan membuat rasio dari koordinat (x ,y). Setelah gambar di-*downsample*, maka akan ditampilkan dalam grid *pixel* 7x9.

Untuk memangkas suatu input gambar yang telah di-*downsample*, maka sistem harus mengambil pola bit gambar. Hal ini dengan cara mendeteksi keberadaan titik-titik *pixel* aktif (yang berwarna hitam dan bernilai 1) dan pencarian ruang dalam citra baik itu terdapat celah spasi atau batas citra tulisan (*bounds*). Hal ini dilakukan dengan menggunakan metode algoritma "*pixel grabber*". Algoritma "*Pixel Grabber*". Metode "*Pixel Grabber"* adalah salah satu Algoritma Pencarian untuk menghasilkan matriks *pixel*. Gambar dari karakter akan menghasilkan batas. Metode ini juga bertugas untuk melihat apakah ada *pixel* dalam pendeteksian yang dilakukan garis *scan*. Selanjutnya adalah mengkombinasikan fitur dari normalisasi koordinat (x , y) dan representasi biner dari gambar yang disajikan ke input jaringan.

Algoritma Metode "*Pixel grabber"* digambarkan dalam alur Algoritma :

i. Cari batas teks seluruh gambar dengan pemindaian dari atas ke bawah untuk batas atas Y1, kiri ke kanan untuk batas kiri X1, kanan ke kiri untuk batas

kanan X2 dan bawah ke atas untuk batas terendah Y2 dari atas, kiri, kanan dan bawah sisi area gambar.

ii. *Scanning* citra biner dari Y1 terhadap Y2

iii. Jika ada *pixel* hitam, kemudian *scan* dari X1 ke X2 untuk baris tertentu untuk mendeteksi *pixel* putih.

iv. Jika tidak ada *pixel* putih yang ditemukan, maka terdapat celah baris,

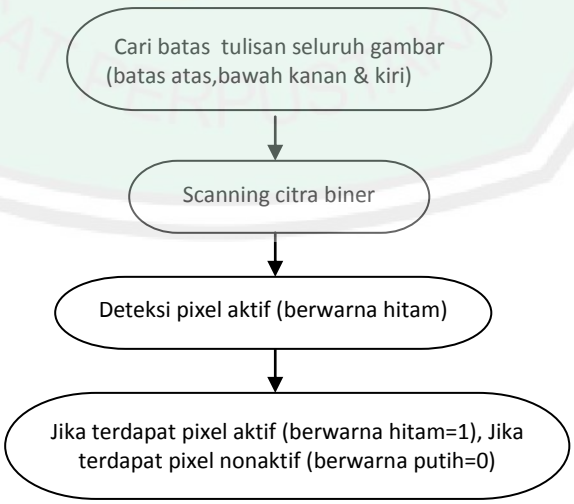
v. Ulangi langkah ii - iv untuk seluruh gambar untuk menemukan jumlah celah baris.

vi. Jika *pixel* hitam ditemukan, maka karakter *pixel* dianggap sebagai '1'.

vii. Jika *pixel* putih ditemukan, maka karakter *pixel* dianggap sebagai '0'.

viii. Ulangi langkah vi – vii untuk mendapatkan representasi biner dari gambar.

Alur Proses *Dot Detection* ditunjukkan pada Gambar 3.3.6. serta penerapannya dalam source code dapat dilihat pada Gambar 3.3.7.



Gambar 3.3.6. Alur proses (*Dot detection "Pixel Grabber")*

```
int w = entryImage.getWidth(this);
for ( int i=0;i<w;i++ ) {
if ( pixelMap[(y*w)+i] !=-1 )
return false;      }
return true;
int w = entryImage.getWidth(this);
int h = entryImage.getHeight(this);
for ( int i=0;i<h;i++ ) {
if ( pixelMap[(i*w)+x] !=-1 )
return false;         }
return true;
  PixelGrabber grabber=new PixelGrabber(entryImage,0,0,w,h,true);
  try {
      grabber.grabPixels();
      pixelMap = (int[])grabber.getPixels();
      findBounds(w,h);
```

Gambar 3.3.7. Source code proses (*Dot detection "Pixel Grabber")*

Metode "*Pixel Grabber"* adalah salah satu Algoritma Pencarian untuk menghasilkan matriks *pixel*. Gambar dari karakter akan menghasilkan batas. Metode ini juga bertugas untuk melihat apakah ada *pixel* dalam pendeteksian yang dilakukan garis *scan*. Jika terdapat pixel aktif (berwarna hitam) maka akan bernilai 1, sebaliknya jika terdapat pixel nonaktif (berwarna putih) makan akan bernilai 0 dan dianggap terdapat celah baris (*bounds*).

3.3.6 Alur sistem *Cropping* dan *Clear Space*

Tahap selanjutnya pada Pemrosesan gambar adalah *Cropping*. *Cropping* adalah proses pemotongan citra gambar sesuai ukuran yang sudah ditentukan. Pemotongan dilakukan dengan memberi batasan jarak pemotongan empat baris imajiner dari atas, kiri, sisi bawah dan kanan gambar. Garis-garis ini akan segera berhenti memotong segera setelah garis-garis ini melewati *pixel* yang sebenarnya. Dengan demikian, empat baris akan hilang dan beralih ke tepi luar gambar. Pada

fase ini akan dilakukan *cropping* otomatis dari batas garis sebelah atas, sebelah kiri, sebelah bawah dan sebelah kanan gambar. Alurnya dapat dilihat pada Gambar 3.3.8. serta penerapannya dalam source code dapat dilihat pada Gambar 3.3.9.

Gambar 3.3.8. Alur proses (*Cropping)*

```
protected boolean hapusGarisHor(int y)    {
      int w = entryImage.getWidth(this);
      for (int i=0; i<w; i++) {
      if (pixelMap[(y*w)+i] !=-1)
      return false;        }
      return true;        }
     for ( int y=0; y<h; y++) {
      if ( !hapusGarisHor(y) ) {
          downSampleTop=y;
          break;      }
      for ( int x=0; x<w;x++ ) {
      if ( !hapusGarisVer(x) ) {
      downSampleLeft = x;
      break;            }
```

Gambar 3.3.9. Source code proses (*Cropping)*

Proses *Cropping* ini sekaligus akan menghitung garis atas dan bawah yang akan dilakukan proses pemotongan. Terdapat parameter untuk metode *cropping* pemotongan garis secara horizontal dan pemotongan garis secara vertikal. Metode

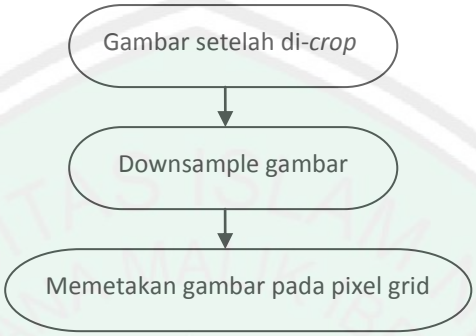
garis horizontal menerima koordinat (y) yang menentukan garis horizontal untuk di-*scan*. Dan metode garis vertikal menerima koordinat (x) yang menentukan garis vertikal untuk di-*scan*. Proses ini juga disertai deteksi ruang-ruang kosong yang bernilai *empty* (pixel putih) dan mengecek apakah ada nilai-nilai dalam setiap *pixel*. Selanjutnya jika ditemui pixel putih akan dinilai 0 atau -1 sehingga akan diabaikan dan dilakukan proses penghapusan celah kosong (*Clear space).*

Sistem akan menghitung garis atas dan bawah untuk dilakukan proses pemotongan. Untuk menghitung garis atas dan memotong baris kotak, program akan dimulai dari atas (0) dan terus ke bagian bawah gambar. Begitu garis pertama ditemukan maka dia tidak dihapus, dan program akan menetapkan ini sebagai bagian atas potongan baris kotak. Proses yang sama, dilakukan secara terbalik yang diterapkan untuk menentukan bagian bawah gambar. Sedangkan proses untuk menentukan kiri dan kanan dilakukan dengan cara yang sama.

3.3.7 Alur sistem *Feature Extraction*

Dalam proses ini akan dilakukan proses *Feature Extraction* yang dilakukan dengan proses *Downsample*. Setelah proses pemotongan dilakukan, maka gambar harus dimunculkan dalam kotak *Downsample* untuk diambil fitur cirinya. Dalam hal ini melibatkan pengambilan gambar resolusi yang lebih besar untuk resolusi *pixel* 7x9. Untuk itu gambar akan dibagi ke dalam grid yang dibuat untuk wadah input gambar yang mempunyai resolusi tinggi. Gambar akan dibagi / dipetakan ke dalam grid *pixel* di bawah resolusi lebar 7 dan tinggi 9. Jadi, jika ada *pixel* di suatu daerah sudah penuh *(full)* , maka *pixel* yang sesuai pada downsample 7x9

juga akan mengisinya. Alurnya ditunjukkan pada Gambar 3.3.10. serta penerapannya dalam source code dapat dilihat pada Gambar 3.3.11.



Gambar 3.3.10. Alur proses *Feature Extraction* (*downsample process)*

```
protected boolean Wilayah_downSample (int x,int y)      {
    int w = entryImage.getWidth(this);
    int startX = (int)(downSampleLeft+(x*rasioX));
    int startY = (int)(downSampleTop+(y*rasioY));
    int endX = (int)(startX + rasioX);
    int endY = (int)(startY + rasioY);
        for (int yy=startY;yy<=endY;yy++) {
        for (int xx=startX;xx<=endX;xx++) {
            int loc = xx+(yy*w);
            if (pixelMap[loc]!= -1 )          }
```

Gambar 3.3.11. Source code (*downsample process)*

Dalam sistem akan dibuat *method* untuk menampung wilayah tempat *downsample* gambar. Fungsi dari parameter ini adalah menerima jumlah wilayah yang harus dihitung. Pertama, koordinat awal dan akhir *x* dan *y* harus dihitung. Untuk menghitung koordinat *x* pertama untuk wilayah tertentu, parameter "*downsampleLeft*" digunakan. Parameter "*downsampleLeft*" adalah bagian sisi sebelah kiri pemotongan. Maka koordinat *x* akan dikalikan dengan rasioX. RasioX merupakan rasio berapa banyak pixel daerah masing-masing. Maka setelah itu kita dapat menentukan dimana untuk menempatkan awal posisi *x* dan posisi *y*.

Sebuah *method* untuk menampung wilayah tempat *downsample* gambar akan menghasilkan sample gambar yang akan disimpan dalam sebuah *class* untuk Sample Data. *Class* ini adalah sebuah *class* yang berfungsi untuk kelas menampung sample data gambar yang berisi boolean grid pixel 7x9. *Class* ini juga merupakan struktur yang membentuk masukan untuk proses *Training* dan proses *Recognize*.

3.4 Alur Sistem *Kohonen Neural Network*

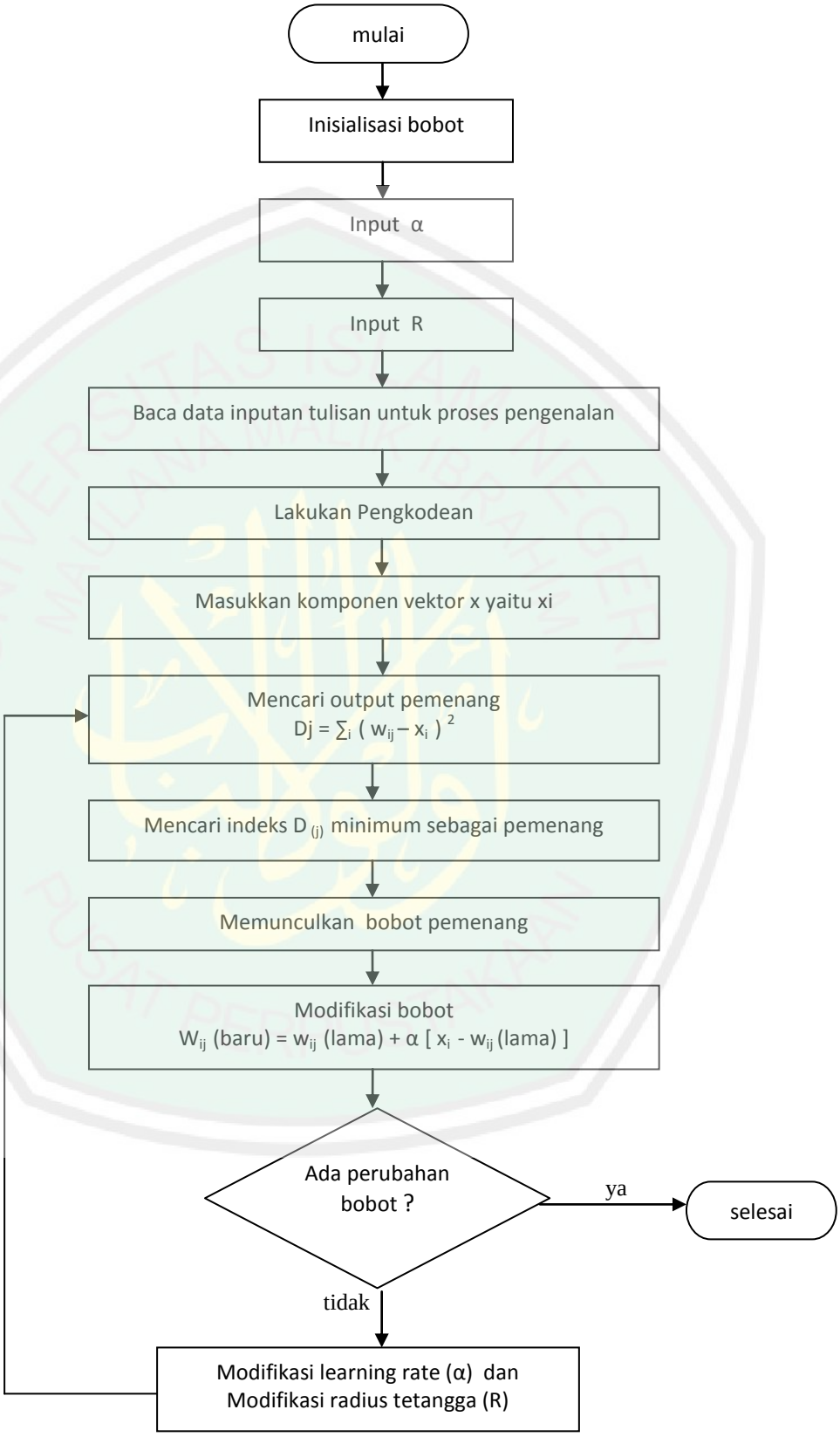
Ada berbagai jenis jaringan syaraf. Jaringan syaraf yang akan diterapkan dalam sistem yang akan dibangun adalah jaringan syaraf tiruan Kohonen. Pola karakter downsample yang diambil oleh user akan diumpankan dan diteruskan ke inputan neuron. Terdapat satu masukan neuron untuk setiap *pixel* dalam gambar *downsample*. Karena gambar *downsample* adalah yang berdimensi 7x9, jadi ada 63 masukan neuron.

Output neuron jaringan syaraf bekerja dengan cara berkomunikasi dengan *user*. Atau dengan kata lain berpikir seperti yang user pikirkan. Jumlah *neuron output* selalu sesuai dengan sample tulisan yang disediakan. Terdapat 63 huruf yang disediakan dalam sample, dan akan ada 63 output neuron.

Selain input dan output neuron, ada pula koneksi hubungan antara neuron-neuron individu. Hubungan ini tidak semua sama. Setiap koneksi terdapat layer-layer neuron. Tiap-tiap layer ini mempunyai bobot-bobot. Bobot-bobot inilah yang nantinya menjadi faktor penentu yang akan menentukan apakah suatu

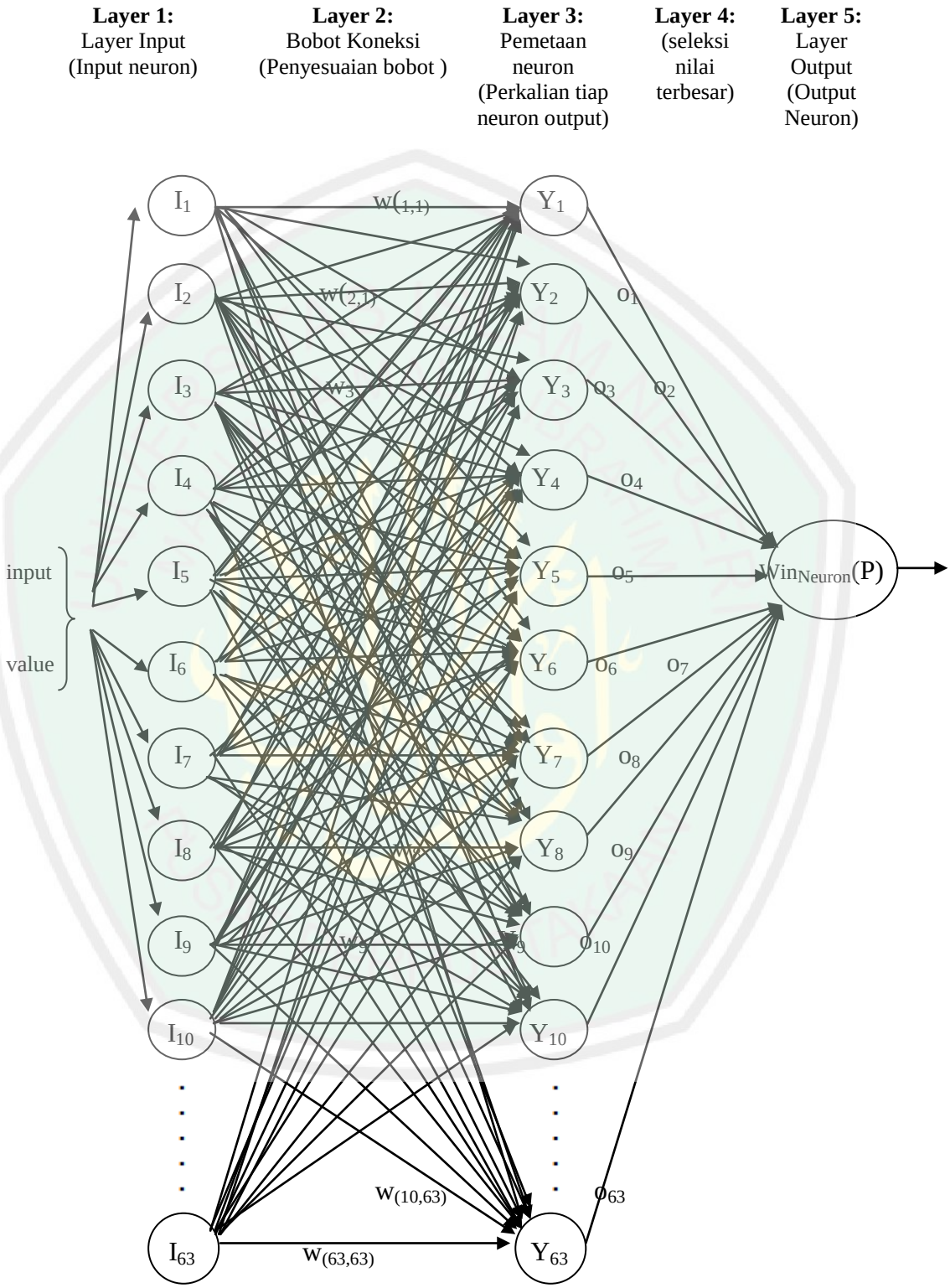
jaringan akan menampilkan pola masukan yang diberikan. Untuk menetukan jumlah koneksi, maka user harus mengalikan jumlah input neuron dengan jumlah output neuron. Sebuah jaringan syaraf dengan 63 output neuron dan 63 masukan neuron akan memiliki total 3969 bobot koneksi. Proses *Training* ini didedikasikan untuk menemukan nilai-nilai yang benar untuk bobot-bobot tersebut.

Jaringan syaraf Kohonen terdiri dari lapisan input dan lapisan output. Jaringan syaraf Kohonen diteruskan / diberikan ke jaringan syaraf menggunakan input neuron. Input neuron ini masing-masing diberi *floating point* yang membentuk pola masukan ke dalam jaringan. Sebuah jaringan syaraf Kohonen mensyaratkan bahwa masukan-masukan ini menjadi tidak normal di kisaran antara nilai -1 dan 1. Penyajian pola masukan ke jaringan akan menimbulkan reaksi dari output neuron. Output dari jaringan syaraf Kohonen hanya terdiri dari neuron output yang benar-benar menghasilkan nilai. Nilai ini bisa bernilai benar atau salah. Ketika pola diinputkan ke jaringan Kohonen, satu output neuron tunggal dipilih sebagai output neuron. Oleh karena itu, output dari jaringan Kohonen biasanya berupa indeks neuron (neuron ke- yang dieliminasi). Flowchart untuk *Kohonen Neural Network* pada sistem, dapat dilihat pada Gambar 3.4.



Gambar 3.4. Flowchart *JST Kohonen* pada sistem

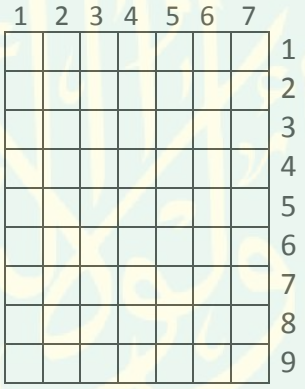
Network / jaringan syaraf adalah kelas dasar untuk kelas Jaringan Syaraf Tiruan *Kohonen* yang pada akhirnya akan memberikan kontribusi dalam proses selanjutnya dalam proses pengenalan menggunakan JST Kohonen. Pada sistem yang akan dibangun alur sistem JST Kohonen terdiri dari layer input, bobot koneksi (nilai bobot yang menghubungkan antara layer input dan layer output) dan layer output. Dalam JST Kohonen, hanya satu dari neuron output yang benar-benar menghasilkan nilai. Selain itu, nilai tunggal ini bisa bernilai benar atau salah. Ketika pola disajikan ke jaringan saraf Kohonen, satu output neuron tunggal dipilih sebagai neuron output. Oleh karena itu, output dari jaringan saraf Kohonen biasanya berupa indeks neuron. Dalam sistem yang dibangun, arsitektur dari peristiwa seluruh proses JST Kohonen ditunjukkan pada Gambar 3.5.



Gambar 3.5. Arsitektur jaringan / network JST Kohonen

3.4.1 Layer1 : layer input

Layer input merupakan layer awal dalam JST Kohonen yang merupakan hasil dari proses downsampling input citra tulisan huruf Arab pada grid *pixel* 7x9. Neuron input dihubungkan dengan neuron output dengan koneksi bobot, yang mana bobot ini selalu diperbaiki pada proses iterasi pelatihan jaringan. Prinsip kerja JST Kohonen adalah pengurangan neuron-neuron tetangga (*neighboor*), sehingga pada akhirnya hanya ada satu neuron output yang terpilih (neuron pemenang). Grid *pixel* 7x9 ditunjukkan pada Gambar 3.5.1.



Gambar 3.5.1 Grid pixel 7x9 sebagai inputan neuron

Gambar 3.5.1 menunjukkan inputan downsample input citra tulisan huruf Arab pada grid *pixel* 7x9. *Pixel* 7x9 menghasilkan 63 input neuron yang berupa nilai 0 atau 1 yang direpresentasikan dengan vektor input. Pada langkah selanjutnya akan dilakukan modifikasi input dengan melakukan normalisasi input. Input pada jaringan syaraf harus antara nilai -1 dan 1 dan harus menggunakan jangkauan. Jika salah satu atau neuron input hanya bernilai antara 0 dan 1, maka kerja JST akan sulit. Untuk menormalkan inputan maka dilakukan perhitungan

untuk panjang vektor. Menghitung panjang vektor dengan cara menjumlahkan kuadrat vektor input :

$$L = i_1^2 + i_2^2 + i_3^2 + \ldots + i_{63}^2$$

Keterangan : L = Panjang vektor
i = Vektor input

Dalam source code penerapannya digambarkan seperti pada Gambar 3.5.2.

```
static double panjang_Vektor( double v[] )       {
    double rtn = 0.0;
    for (int i=0; i<v.length; i++)
    rtn += v[i] * v[i];
    return rtn;
```

Gambar 3.5.2. Source code Proses menghitung panjang vector

Dari hasil perhitungan panjang vektor, akan menghasilkan nilai. Tapi jika nilai terlalu kecil, maka panjang vektor akan meneruskan ke proses selanjutnya dengan nilai yang kecil. Dalam JST Kohonen, panjang vektor identik dengan nilai yang besar. Jadi akan diberikan parameter faktor normalisasi. Faktor normalisasi adalah kebalikan dari akar kuadrat dari panjang vektor yang dirumuskan :

Faktor normalisasi → (N) = $1\sqrt{L}$

Keterangan : L = Panjang vektor

Dalam source code sistem penerapannya seperti pada Gambar 3.5.3.

```
void normalisasi_Input(final double input[] ,double fak_normalisasi[] ,double synth[])    {
    double panjang, d ;
    panjang = panjang_Vektor(input) ;
if (panjang < 1.E-30)
    panjang= 1.E-30;
    fak_normalisasi[0] = 1.0 / Math.sqrt(panjang);
    synth[0] = 0.0;       }
```

Gambar 3.5.3. Source code Proses normalisasi input

Pada tahap ini selesai pada layer 1 (input layer) dan hasil normalisasi input neuron ini akan digunakan pada proses selanjutnya yaitu perhitungan untuk penyesuaian bobot.

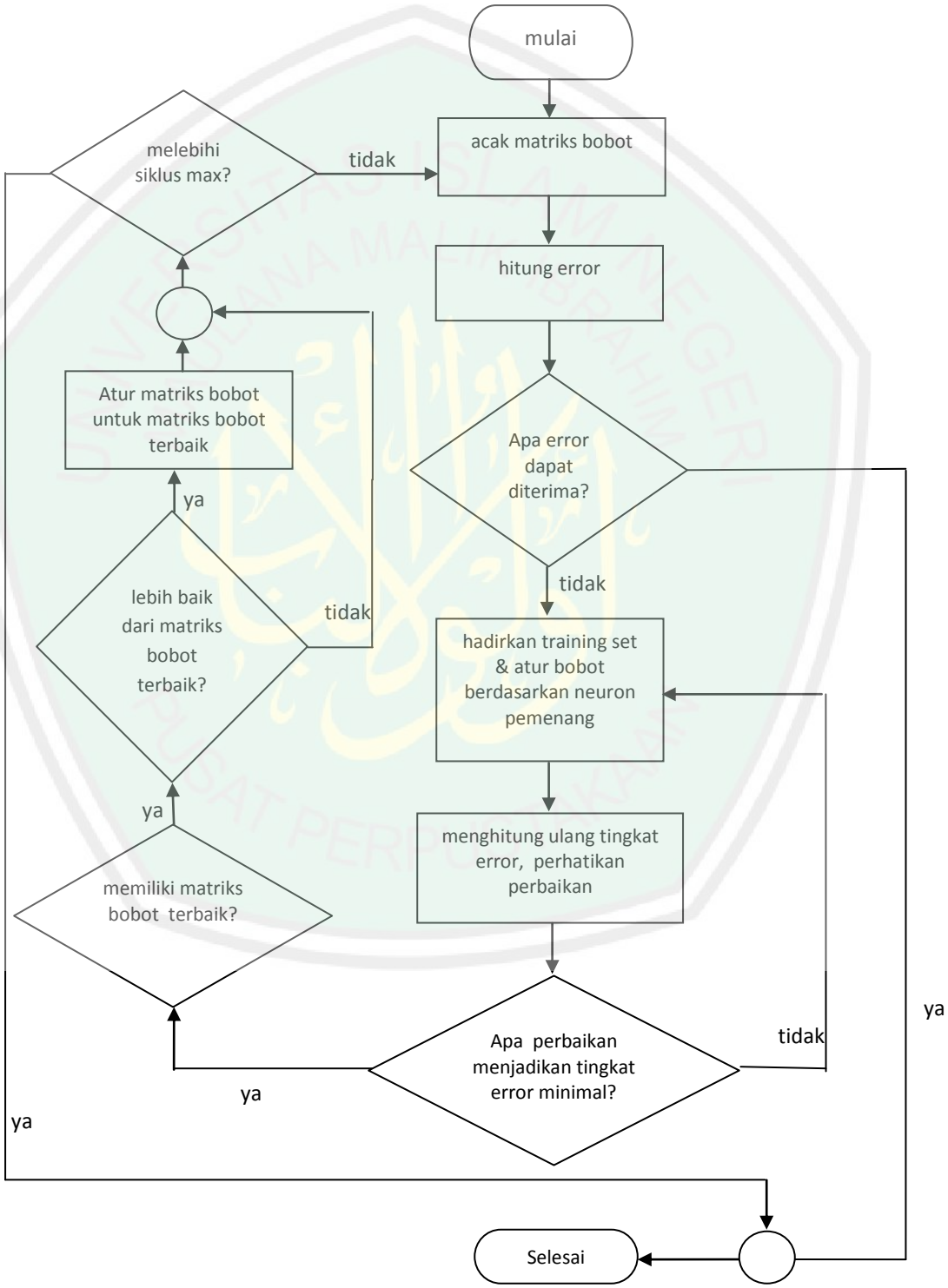
3.4.2 Layer2 : bobot koneksi (Menentukan bobot)

Pada layer 2 (bobot koneksi) adalah proses pelatihan JST Kohonen.pada proses ini adalah proses pemilihan bobot neuron yang akan benar-benar mengenali pola inputan. Dalam JST akan melakukan pelatihan dengan terus-menerus mengevaluasi dan mengoptimalkan bobot neuron. Terdapat beberapa proses dalam pelatihan JST. Secara keseluruhan, proses untuk pelatihan JST Kohonen melibatkan beberapa *epochs* sampai tingkat error JST dibawah target error minimal yang dapat diterima. Pada bagian ini akan dijelaskan bagaimana menghitung tingkat error untuk JST dan dijelaskan juga proses penyesuaian bobot untuk setiap *epochs*. Dalam fase ini juga akan dilakukan untuk penentuan kapan akan dihadirkan *epochs* yang diperlukan untuk melatih jaringan.

3.4.2.1 Proses Training JST Kohonen

Secara keseluruhan, proses untuk *training* JST Kohonen melibatkan langkah-langkah yang harus melalui epoch-epoch sampai kesalahan *(error)* JST Kohonen terletak ditingkat bawah atau yang dapat diterima. Untuk setiap *training*, sistem menetapkan satu neuron akan menjadi “pemenang”. Neuron pemenang ini akan memiliki bobot yang disesuaikan sehingga akan berinteraksi lebih kuat terhadap input *epochs* berikutnya. Sebagai neuron yang dianggap pemenang ini,

kemampuan neuron ini dalam mengenali pola inputan akan meningkat. Proses *Training* JST Kohonen dapat dilihat pada Gambar 3.5.4.



Gambar 3.5.4. Flowchart fase Training JST Kohonen

Pelatihan JST Kohonen dilakukan dengan mengulangi *epoch-epoch* sampai salah satu dari dua neuron bernilai tinggi. Jika kesalahan dihitung dan diketahui kesalahan *(error)* JST Kohonen terletak ditingkat bawah atau yang dapat diterima, proses *Training* selesai. Jika tingkat kesalahan diubah dengan hanya jumlah yang sangat kecil, siklus *Training* secara individu ini akan dibatalkan tanpa adanya epoch-epoch tambahan. Jika diketahui bahwa siklus ini harus dibatalkan, bobot akan dilakukan proses "*inisialisasi nilai acak*" dan siklus *training* baru dimulai. Bagian terpenting dalam siklus *training* ini adalah *epoch-epoch* individu. Pengubahan *epoch-epoch* terdiri dari beberapa parameter-parameter yang nantinya akan mempengaruhi penyesuaian bobot untuk setiap *epochs*.

Dalam flowchart Gambar 3.5.4. hal pertama yang dilakukan adalah bobot awal yang harus ditentukan. Awalnya bobot matriks dipilih dengan memilih nilai acak. Dalam memilih bobot secara acak, tentunya adalah pilihan yang kurang tepat untuk bobot neuron, tetapi akan memberikan nilai awal untuk pengoptimalan bobot neuron.

Setelah bobot neuron awal dipilih secara acak maka proses pelatihan baru dimulai. Nilai-nilai acak kemudian dilatih untuk menghasilkan hasil yang lebih baik. Pada awal pelatihan, bobot juga akan diinisialisasi dengan nilai-nilai acak. Dalam sistem dirumuskan :

$$A_b = \frac{3.464101615}{2 \times (\text{acak angka})}$$

Keterangan : 3.464101615 (Standart SQRT)

Dalam source code penerapannya seperti pada Gambar 3.5.5.

```
void acak_bobot (double bobot[][])        {
double r;
int temp = (int)(3.464101615/(2.*Math.random()));  //SQRT(12)=3.464…
for (int y=0;  y<bobot.length;  y++) {
for (int x=0;  x<bobot[0].length;  x++)    {
      r = (double)acak.nextInt(Integer.MAX_VALUE) + (double)acak.nextInt
      (Integer.MAX_VALUE) - (double)acak.nextInt(Integer.MAX_VALUE) -
      (double)acak.nextInt(Integer.MAX_VALUE);
      bobot[y][x] = temp * r;       }
```

Gambar 3.5.5. Source code proses mengacak bobot

Dalam proses diatas, terlihat sistem akan melakukan perulangan (*looping*) melalui seluruh bobot matriks dan akan memberikan angka secara acak. Disini juga akan terlihat bahwa 2 angka acak yang digunakan untuk setiap perhitungan. Hal ini dimaksudkan untuk lebih mengacak bobot melebihi jumlah yang dihasilkan secara acak tunggal.

Langkah selanjutnya adalah mengevaluasi bobot neuron untuk menentukan tingkat kesalahan (*error*) pada saat ini (training saat ini). Kesalahan (*error*) ini ditentukan oleh seberapa baik input pelatihan (data training huruf yang dilatih) untuk output neuron. Jika *error* dibawah 10 % proses training selesai untuk tiap data training. Jika tingkat *error* masih diatas 10% (belum bisa diterima), sistem akan menghadirkan masing-masing unsur pelatihan untuk jaringan. Saat masing-masing data pelatihan disajikan, sistem akan melacak output neuron pemenang. Dalam sistem, untuk melacak masing-masing output neuron pemenang dirumuskan :

$$D_j = \sum_i (w_{ij} - i_i^2)$$

Keterangan : D_j = output neuron ke-j
w_{ij} = bobot koneksi input ke-i ke output ke-j
i_i = input neuron ke-i

Ilustrasi untuk melacak masing-masing output neuron pemenang dari perumusan diatas dapat dirumuskan :

$$D_{ke\text{-}1} = (w_{1,1} - i_1^2) + (w_{2,1} - i_1^2) + (w_{3,1} - i_1^2) + (w_{4,1} - i_1^2) + (w_{5,1} - i_1^2) + (w_{6,1} - i_1^2) + (w_{7,1} - i_1^2) + (w_{8,1} - i_1^2) + (w_{9,1} - i_1^2) + (w_{10,1} - i_1^2) + \ldots\ldots + (w_{63,1} - i_1^2)$$

$$D_{ke\text{-}2} = (w_{1,2} - i_2^2) + (w_{2,2} - i_2^2) + (w_{3,2} - i_2^2) + (w_{4,2} - i_2^2) + (w_{5,2} - i_2^2) + (w_{6,2} - i_2^2) + (w_{7,2} - i_2^2) + (w_{8,2} - i_2^2) + (w_{9,2} - i_2^2) + (w_{10,2} - i_2^2) + \ldots\ldots + (w_{63,2} - i_2^2)$$

$$D_{ke\text{-}3} = (w_{1,3} - i_3^2) + (w_{2,3} - i_3^2) + (w_{3,3} - i_3^2) + (w_{4,3} - i_3^2) + (w_{5,3} - i_3^2) + (w_{6,3} - i_3^2) + (w_{7,3} - i_3^2) + (w_{8,3} - i_3^2) + (w_{9,3} - i_3^2) + (w_{10,3} - i_3^2) + \ldots\ldots + (w_{63,3} - i_3^2)$$

$$D_{ke\text{-}4} = (w_{1,4} - i_4^2) + (w_{2,4} - i_4^2) + (w_{3,4} - i_4^2) + (w_{4,4} - i_4^2) + (w_{5,4} - i_4^2) + (w_{6,4} - i_4^2) + (w_{7,4} - i_4^2) + (w_{8,4} - i_4^2) + (w_{9,4} - i_4^2) + (w_{10,4} - i_4^2) + \ldots\ldots + (w_{63,4} - i_4^2)$$

$$D_{ke\text{-}5} = (w_{1,5} - i_5^2) + (w_{2,5} - i_5^2) + (w_{3,5} - i_5^2) + (w_{4,5} - i_5^2) + (w_{5,5} - i_5^2) + (w_{6,5} - i_5^2) + (w_{7,5} - i_5^2) + (w_{8,5} - i_5^2) + (w_{9,5} - i_5^2) + (w_{10,5} - i_5^2) + \ldots\ldots + (w_{63,5} - i_5^2)$$

$$D_{ke\text{-}6} = (w_{1,6} - i_6^2) + (w_{2,6} - i_6^2) + (w_{3,6} - i_6^2) + (w_{4,6} - i_6^2) + (w_{5,6} - i_6^2) + (w_{6,6} - i_6^2) + (w_{7,6} - i_6^2) + (w_{8,6} - i_6^2) + (w_{9,6} - i_6^2) + (w_{10,6} - i_6^2) + \ldots\ldots + (w_{63,6} - i_6^2)$$

$$D_{ke\text{-}7} = (w_{1,7} - i_7^2) + (w_{2,7} - i_7^2) + (w_{3,7} - i_7^2) + (w_{4,7} - i_7^2) + (w_{5,7} - i_7^2) + (w_{6,7} - i_7^2) + (w_{7,7} - i_7^2) + (w_{8,7} - i_7^2) + (w_{9,7} - i_7^2) + (w_{10,7} - i_7^2) + \ldots\ldots + (w_{63,7} - i_7^2)$$

.
.
.
.
.

$$D_{ke\text{-}63} = (w_{1,63} - i_{63}^2) + (w_{2,63} - i_{63}^2) + (w_{3,63} - i_{63}^2) + (w_{4,63} - i_{63}^2) + (w_{5,63} - i_{63}^2) + (w_{6,63} - i_{63}^2) + (w_{7,63} - i_{63}^2) + (w_{8,63} - i_{63}^2) + (w_{9,63} - i_{63}^2) + (w_{10,63} - i_{63}^2) + \ldots\ldots + (w_{63,63} - i_{63}^2)$$

Dalam source code penerapannya ditunjukkan pada Gambar 3.5.6.

```
void evaluasi_Kesalahan(double rate , int metod_Belajar, int won[],
      double lbih_besar[] ,double tepat[][] ,double kerja[]) throws RuntimeException     {
  int best, size, tset ;
  double dptr[], fak_Norm[] = new double[1];
  double synth[]=new double[1], cptr[], wptr[], length, diff;
      for (int y=0; y<tepat.length; y++) {
      for (int x=0; x<tepat[0].length; x++) {
      tepat[y][x]=0;    }}
      for (int i=0; i<won.length; i++)
      won[i]=0;
      lbih_besar[0] = 0.0;
  for (tset=0; tset<train.dapat_jumlah_TrainingSet(); tset++) {
  dptr = train.dapat_kumpulan_Input(tset);
  best = pemenang (dptr,fak_Norm ,synth);
  won[best]++;
  wptr = bobotKeluaran[best];
  cptr = tepat[best];
  length = 0.0;
  for (int i=0; i<jumlah_inputNeuron; i++) {
  diff = dptr[i] * fak_Norm[0] - wptr[i] ;
  length += diff * diff ;
     if (metod_Belajar!=0)
     cptr[i] += diff ;
     else
     kerja[i] = rate * dptr[i] * fak_Norm[0] + wptr[i] ;     }
    diff = synth[0] - wptr[jumlah_inputNeuron];
     length += diff * diff ;
     if (metod_Belajar!=0)
     cptr[jumlah_inputNeuron] += diff;
     else
     kerja[jumlah_inputNeuron] = rate * synth[0] + wptr[jumlah_inputNeuron];
     if (length > lbih_besar[0])
     lbih_besar[0] = length;
     if (metod_Belajar==0) {
     normalisasi_bobot(kerja);
     for (int i=0; i<=jumlah_inputNeuron; i++)
     cptr[i] += kerja[i] - wptr[i];      }}
     lbih_besar[0] = Math.sqrt(lbih_besar[0]);   }
```

Gambar 3.5.6. Source code evaluasi error dan melacak neuron pemenang (D_j)

Pada proses diatas, akan ditemukan hasil untuk tiap-tiap neuron output pemenang D_j. Dalam proses ini masing-masing output neuron akan diproses untuk mencari output neuron pemenang yang kemudian akan disesuaikan sehingga akan bekerja lebih baik pada pelatihan untuk *epochs* berikutnya. Masing-masing neuron akan mengkonsolidasikan neuron pemenang. Dan akan digunakan untuk

membantu neuron yang belum pernah menang agar menjadi menang (telah belajar/training dengan benar). Neuron ini akan dipindahkan untuk menghilangkan beberapa bobot dari output neuron yang sudah menang. Setelah itu koreksi error dihitung, dan bobot harus disesuaikan dengan koreksi error ini. Dalam sistem, untuk mencari masing-masing bobot koneksi dirumuskan :

$$w_{ij}\ (\text{baru}) = w_{ij}\ (\text{lama}) + \alpha\ [\ i_i - w_{ij}\ (\text{lama})\ ]$$

Keterangan : w_{ij} = bobot koneksi input ke-i ke output ke-j
i_i = input neuron ke-i
α = learning rate (0,5)

Ilustrasi untuk mencari masing-masing bobot koneksi yang berjumlah 63 input dikali 63 output yaitu 3969 bobot koneksi. Dari perumusan diatas dapat dirumuskan :

xi_1 (neuron ke-1) →

$$w_{1,1}\ (\text{baru}) = w_{1,1}\ (\text{lama}) + \alpha\ [\ i_1 - w_{1,1}\ (\text{lama})\ ]$$
$$w_{1,2}\ (\text{baru}) = w_{1,2}\ (\text{lama}) + \alpha\ [\ i_1 - w_{1,2}\ (\text{lama})\ ]$$
$$w_{1,3}\ (\text{baru}) = w_{1,3}\ (\text{lama}) + \alpha\ [\ i_1 - w_{1,3}\ (\text{lama})\ ]$$
$$w_{1,4}\ (\text{baru}) = w_{1,4}\ (\text{lama}) + \alpha\ [\ i_1 - w_{1,4}\ (\text{lama})\ ]$$
$$w_{1,5}\ (\text{baru}) = w_{1,5}\ (\text{lama}) + \alpha\ [\ i_1 - w_{1,5}\ (\text{lama})\ ]$$
$$\vdots$$
$$w_{1,63}\ (\text{baru}) = w_{1,63}\ (\text{lama}) + \alpha\ [\ i_1 - w_{1,63}\ (\text{lama})]$$

xi_2 (neuron ke-2) →

$$w_{2,1}\ (\text{baru}) = w_{2,1}\ (\text{lama}) + \alpha\ [\ i_2 - w_{2,1}\ (\text{lama})\ ]$$
$$w_{2,2}\ (\text{baru}) = w_{2,2}\ (\text{lama}) + \alpha\ [\ i_2 - w_{2,2}\ (\text{lama})\ ]$$
$$w_{2,3}\ (\text{baru}) = w_{2,3}\ (\text{lama}) + \alpha\ [\ i_2 - w_{2,3}\ (\text{lama})\ ]$$
$$\vdots$$
$$w_{2,63}\ (\text{baru}) = w_{2,63}\ (\text{lama}) + \alpha\ [\ i_2 - w_{2,63}\ (\text{lama})]$$

xi_3 → $w_{3,1}$ (baru) = $w_{3,1}$ (lama) + α [i_3 – $w_{3,1}$ (lama)]
(neuron ke-3) $w_{3,2}$ (baru) = $w_{3,2}$ (lama) + α [i_3 – $w_{3,2}$ (lama)]
$w_{3,3}$ (baru) = $w_{3,3}$ (lama) + α [i_3 – $w_{3,3}$ (lama)]
.
.
.
.
.
$w_{3,63}$ (baru) = $w_{3,63}$ (lama) + α [i_3 – $w_{3,63}$ (lama)]

xi_4 → $w_{4,1}$ (baru) = $w_{4,1}$ (lama) + α [i_4 – $w_{4,1}$ (lama)]
(neuron ke-4) $w_{4,2}$ (baru) = $w_{4,2}$ (lama) + α [i_4 – $w_{4,2}$ (lama)]
$w_{4,3}$ (baru) = $w_{4,3}$ (lama) + α [i_4 – $w_{4,3}$ (lama)]
.
.
.
.
.
$w_{4,63}$ (baru) = $w_{4,63}$ (lama) + α [i_4 – $w_{4,63}$ (lama)]

xi_5 → $w_{5,1}$ (baru) = $w_{5,1}$ (lama) + α [i_5 – $w_{5,1}$ (lama)]
(neuron ke-5) $w_{5,2}$ (baru) = $w_{5,2}$ (lama) + α [i_5 – $w_{5,2}$ (lama)]
$w_{5,3}$ (baru) = $w_{5,3}$ (lama) + α [i_5 – $w_{5,3}$ (lama)]
.
.
.
.
.
$w_{5,63}$ (baru) = $w_{5,63}$ (lama) + α [i_5 – $w_{5,63}$ (lama)]

.
.
.
.
.

xi_{63} → $w_{63,1}$ (baru) = $w_{63,1}$ (lama) + α [i_{63} – $w_{63,1}$ (lama)]
(neuron ke-63) $w_{63,2}$ (baru) = $w_{63,2}$ (lama) + α [i_{63} – $w_{63,2}$ (lama)]
$w_{63,3}$ (baru) = $w_{63,3}$ (lama) + α [i_{63} – $w_{63,3}$ (lama)]
.
.
.
.
.
$w_{63,63}$ (baru) = $w_{63,63}$ (lama) + α [i_{63} – $w_{63,63}$ (lama)]

Dalam source code penerapannya ditunjukkan dalam Gambar 3.5.7.

```
void atur_bobot (double rate,int metode_belajar,int won[],
            double bigcorr[],double correc[][])   {
 double corr, cptr[], wptr[], length, f ;
 bigcorr[0] = 0.0 ;
 for (int i=0; i< jumlah_outputNeuron; i++) {
 if (won[i]==0)
 continue;
 wptr = bobotKeluaran[i];
 cptr = correc[i];
 f = 1.0/(double)won[i];
 if (metode_belajar!=0)
 f*= rate;
 length = 0.0;
 for (int j=0; j<=jumlah_inputNeuron; j++) {
 corr = f * cptr[j];
 wptr[j] += corr;
 length += corr * corr;     }
 if (length > bigcorr[0])
 bigcorr[0] = length;    }
 // skala koreksi
 bigcorr[0] = Math.sqrt(bigcorr[0]) / rate;  }
```

Gambar 3.5.7. Source code penyesuaian & perhitungan tiap neuron bobot koneksi

Pada proses diatas, koreksi tingkat kesalahan dilakukan oleh learning rate. Training terjadi dengan menyesuaikan neuron pemenang. Kadangkala akan ada neuron yang gagal menang karena proses training/belajar tidak berkerja dengan benar. Maka untuk mengatasi masalah ini, sistem akan melakukan perbaikan dengan memodifikasi learning rate. Pada sistem, learning rate ditentukan dengan nilai 0,5. Maka akan dilakukan modifikasi learning rate dirumuskan :

$$\alpha\ (t + 1) = 0{,}5\ \alpha t$$

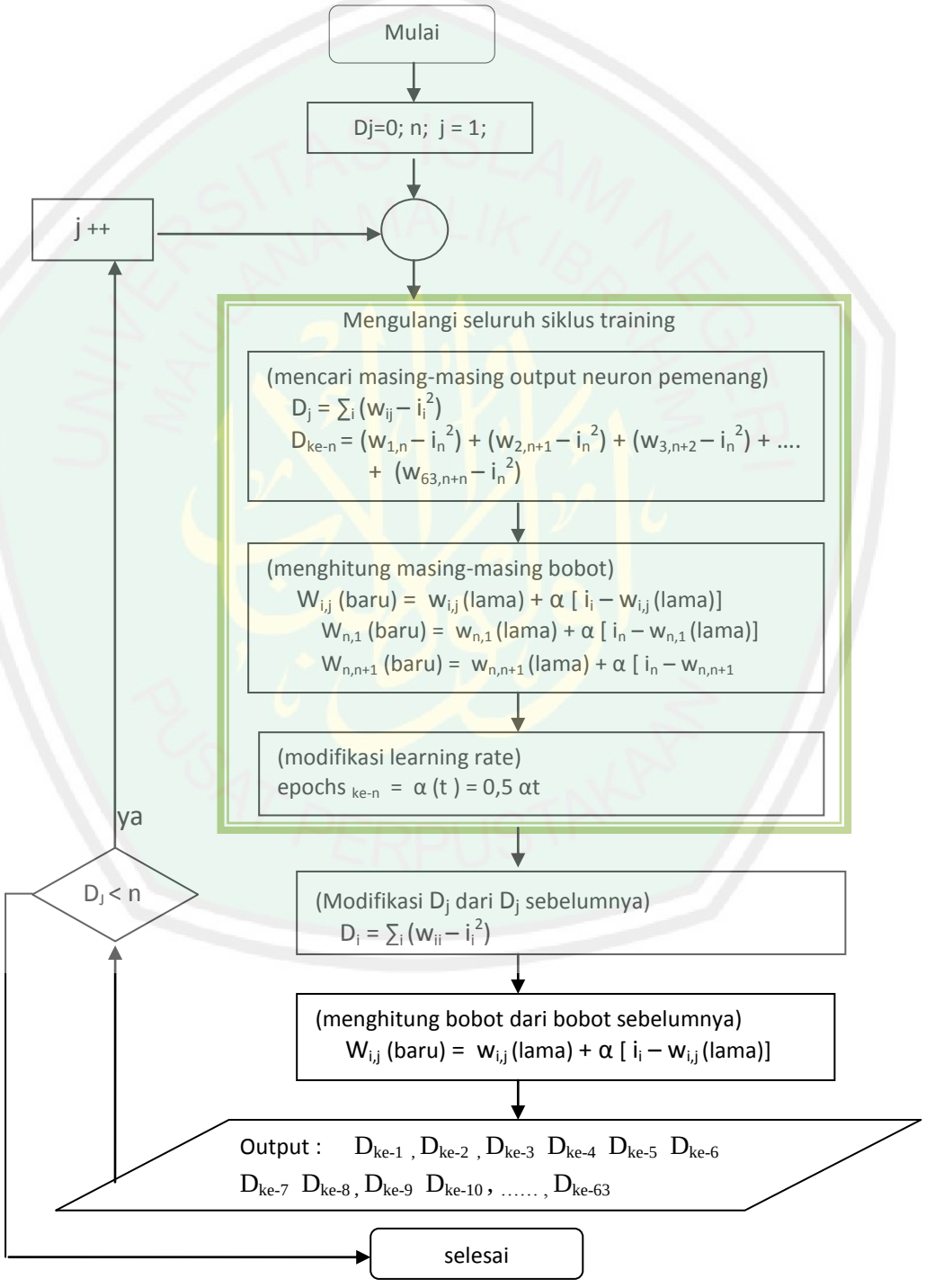
Keterangan : t = epochs
α = learning rate (0,5)

Dalam perumusan diatas terdapat parameter *epochs*. *Epochs* disini diartikan sebagai proses dalam training untuk setiap siklus pelatihan inputan data training. Untuk setiap training data inputan akan menghasilkan error dan proses pengaturan bobot hingga error bernilai minimum dibawah 10% dan bobot neuron dapat ditentukan sebagai neuron pemenang. Proses tersebut untuk satu kali epochs. Kadangkala akan ada neuron yang gagal menang karena proses training/belajar tidak berkerja dengan benar. Maka untuk mengatasi masalah ini, sistem akan melakukan perbaikan dengan memodifikasi learning rate dan mengulang perhitungan dimana proses ini masuk kedalam epochs berikutnya (*epochs ke-2*). Jika ada neuron yang belum berhasil menang, maka sistem pelatihan akan diulang kembali dengan mengulang proses perhitungan error lagi sampai error minimal, ini masuk kedalam epochs berikutnya (*epochs ke-3*), jadi akan ada beberapa epochs dalam satu kali training 1 data inputan dan masing-masing epochs dalam tiap proses training data ini tidaklah sama, tergantung seberapa baik jaringan memproses pola input tulisan. Ilustrasi untuk memodifikasi learning rate dalam proses training untuk tiap *epochs* inputan 1 data training dapat dirumuskan :

$$epochs_{ke\text{-}1} = \alpha\,(t) = 0{,}5\,\alpha t$$
$$epochs_{ke\text{-}2} = \alpha\,(t+1) = 0{,}5\,\alpha t$$
$$epochs_{ke\text{-}3} = \alpha\,(t+2) = 0{,}5\,\alpha t$$
$$epochs_{ke\text{-}4} = \alpha\,(t+3) = 0{,}5\,\alpha t$$
$$\vdots$$
$$epochs_{ke\text{-}n} = \alpha\,(t+n) = 0{,}5\,\alpha t$$

Dari perhitungan diatas akan dihasilkan proses untuk perhitungan penyesuaian neuron pemenang. Dan neuron yang belum menang akan training

lagi agar neuron yang belum menang (belum belajar dengan baik) bisa menang. Pada sistem akan dilakukan proses agar neuron bisa menjadi menang. Ilustrasi proses agar neuron bisa menjadi menang dapat ditunjukkan Gambar 3.5.8.



Gambar 3.5.8. Alur proses perulangan untuk memenangkan neuron yang belum menang

Proses diatas dilakukan untuk menangani neuron output yang yang gagal menang agar menjadi menang (belajar dengan baik). Pertama sistem mengulangi seluruh siklus pelatihan dan menemukan pola pelatihan yang paling efektif (belajar dengan benar) karena mewakili sebagai siklus pelatihan yang paling baik untuk output neuron. Lalu sistem melakukan modifikasi neuron output untuk lebih mengklasifikasi kumpulan pelatihan pada modifikasi sebelumnya. Dilakukan dengan cara mengulangi tiap neuron output yang gagal menang dan menyeleksi yang mana yang memiliki aktifasi tertinggi untuk pola pada siklus training. Akhirnya, sistem akan modifikasi neuron bobot sehingga akan mengklasifikasi pola yang lebih baik pada epochs berikutnya.

Dalam source code penerapannya ditunjukkan pada Gambar 3.5.9.

```
void Dibuat_menang(int won[])throws RuntimeException    {
int i, tset, best, size, which=0;
double dptr[], fak_Norm[]=new double[1];
double synth[] = new double[1],dist,optr[];
size =jumlah_inputNeuron+ 1;
dist = 1.E30;
for (tset=0; tset<train.dapat_jumlah_TrainingSet(); tset++) {
dptr = train.dapat_kumpulan_Input(tset);
best = pemenang(dptr ,fak_Norm, synth);
if (output[best] < dist) {
dist = output[best];
which = tset;             }}
dptr = train.dapat_kumpulan_Input(which);
best = pemenang(dptr,fak_Norm,synth);
dist = -1.e30;
 i = jumlah_outputNeuron;
while ((i--)>0) {
 if (won[i]!=0)
 continue;
 if (output[i] > dist) {
 dist = output[i];
 which = i;       } }
optr = bobotKeluaran[which];
System.arraycopy(dptr,0,optr,0,dptr.length);
optr[jumlah_inputNeuron] = synth[0]/fak_Norm[0];
normalisasi_bobot(optr);  }
```

Gambar 3.5.9. Source code untuk membuat neuron menang (mengulangi pelatihan)

Setelah itu proses selesai sampai semua neuron output bisa belajar dengan baik (menang) dalam artian ini "menang" adalah tingkat error sudah dibawah 10% dan sudah dapat diterima oleh sistem. Dalam sistem, rumusan untuk perhitungannya dirumuskan :

$$Ter = Er \times 100\%$$

Keterangan : Ter = total error
Er = error terbaik (minimum)

Ilustrasi untuk menghitung total error output neuron dapat dirumuskan :

$Ter\ D_{ke-1} = Er \times 100\%$
$Ter\ D_{ke-2} = Er \times 100\%$
$Ter\ D_{ke-3} = Er \times 100\%$
$Ter\ D_{ke-4} = Er \times 100\%$
.
.
.
.
.
$Ter\ D_{ke-63} = Er \times 100\%$

Proses ini adalah proses akhir pada layer bobot koneksi. Pertama, error ditentukan, sistem akan melihat apakah sejauh ini error dikategorikan sudah berada dibawah error terbaik (dibawah 10%). Jika error sudah berada dibawah error terbaik, maka error akan dipilih sebagai error terbaik, dan bobot neuron dipilih. Jumlah neuron pemenang dihitung. Hal ini memungkinkan sistem untuk menentukan apakah sudah tidak ada lagi output neuron yang muncul. Jika error sudah berada dibawah error terbaik (dibawah 10%) dan sudah tidak ada lagi output neuron yang muncul, maka proses training selesai dan bobot ditentukan. Dalam source code penerapannya ditunjukkan pada Gambar 3.5.10.

```
totalError = bigerr[0];
if (totalError < best_err) {
best_err = totalError;
Salinan_bobot(bestnet,this);      }
termenang = 0;
for (i=0; i<won.length;i++)
if (won[i]!=0)
termenang++;
if (bigerr[0] < error_selesai)
break;
if ((termenang < jumlah_outputNeuron) &&
(termenang < train.dapat_jumlah_TrainingSet())) {
Dibuat_menang(won);
continue ;     }
    if ((((trial%100)!=0) || (trial==10)) && !jaringan.berhenti)
    return;
    if (jaringan.berhenti) {
    alur_pelatihan = null;
```

Gambar 3.5.10. Source code evaluasi error terkecil dan error terbaik

3.4.3 Layer3 : Perkalian tiap Neuron Output

Proses pada layer2 (bobot koneksi / penyesuaian bobot) telah selesai untuk proses terpenting dalam penentuan output JST Kohonen. Setelah itu dilakukan perhitungan untuk perkalian antara input neuron dan bobot yang telah didapat dari proses pada layer2 (bobot koneksi / penyesuaian bobot). Perhitungan tersebut dilakukan untuk masing-masing output neuron. Melalui rumus tersebut sistem memeriksa perhitungan untuk masing-masing output neuron. Dalam sistem, rumusan untuk perhitungannya dirumuskan :

$$Y_j = \sum_y (i_i \times w_{ij})$$

Keterangan : Y_j = output neuron ke-j
i_i = input neuron ke-i
w_{ij} = bobot neuron dari i ke-j

Ilustrasi untuk mencari masing-masing output neuron dari perumusan diatas dapat dirumuskan :

$$Y_{ke\text{-}1} = (i_1 \times w_{1,1}) + (i_1 \times w_{2,1}) + (i_1 \times w_{3,1}) + (i_1 \times w_{4,1}) + (i_1 \times w_{5,1}) + (i_1 \times w_{6,1}) + (i_1 \times w_{7,1}) + (i_1 \times w_{8,1}) + (i_1 \times w_{9,1}) + (i_1 \times w_{10,1}) + \ldots\ldots + (i_1 \times w_{63,1})$$

$$Y_{ke\text{-}2} = (i_2 \times w_{1,2}) + (i_2 \times w_{2,2}) + (i_2 \times w_{3,2}) + (i_2 \times w_{4,2}) + (i_2 \times w_{5,2}) + (i_2 \times w_{6,2}) + (i_2 \times w_{7,2}) + (i_2 \times w_{8,2}) + (i_2 \times w_{9,2}) + (i_2 \times w_{10,2}) + \ldots\ldots + (i_2 \times w_{63,2})$$

$$Y_{ke\text{-}3} = (i_3 \times w_{1,3}) + (i_3 \times w_{2,3}) + (i_3 \times w_{3,3}) + (i_3 \times w_{4,3}) + (i_3 \times w_{5,3}) + (i_3 \times w_{6,3}) + (i_3 \times w_{7,3}) + (i_3 \times w_{8,3}) + (i_3 \times w_{9,3}) + (i_3 \times w_{10,3}) + \ldots\ldots + (i_3 \times w_{63,3})$$

$$Y_{ke\text{-}4} = (i_4 \times w_{1,4}) + (i_4 \times w_{2,4}) + (i_4 \times w_{3,4}) + (i_4 \times w_{4,4}) + (i_4 \times w_{5,4}) + (i_4 \times w_{6,4}) + (i_4 \times w_{7,4}) + (i_4 \times w_{8,4}) + (i_4 \times w_{9,4}) + (i_4 \times w_{10,4}) + \ldots\ldots + (i_4 \times w_{63,4})$$

$$Y_{ke\text{-}5} = (i_5 \times w_{1,5}) + (i_5 \times w_{2,5}) + (i_5 \times w_{3,5}) + (i_5 \times w_{4,5}) + (i_5 \times w_{5,5}) + (i_5 \times w_{6,5}) + (i_5 \times w_{7,5}) + (i_5 \times w_{8,5}) + (i_5 \times w_{9,5}) + (i_5 \times w_{10,5}) + \ldots\ldots + (i_5 \times w_{63,5})$$

$$Y_{ke\text{-}6} = (i_6 \times w_{1,6}) + (i_6 \times w_{2,6}) + (i_6 \times w_{3,6}) + (i_6 \times w_{4,6}) + (i_6 \times w_{5,6}) + (i_6 \times w_{6,6}) + (i_6 \times w_{7,6}) + (i_6 \times w_{8,6}) + (i_6 \times w_{9,6}) + (i_6 \times w_{10,6}) + \ldots\ldots + (i_6 \times w_{63,6})$$

$$Y_{ke\text{-}7} = (i_7 \times w_{1,7}) + (i_7 \times w_{2,7}) + (i_7 \times w_{3,7}) + (i_7 \times w_{4,7}) + (i_7 \times w_{5,7}) + (i_7 \times w_{6,7}) + (i_7 \times w_{7,7}) + (i_7 \times w_{8,7}) + (i_7 \times w_{9,7}) + (i_7 \times w_{10,7}) + \ldots\ldots + (i_7 \times w_{63,7})$$

$$Y_{ke\text{-}8} = (i_8 \times w_{1,8}) + (i_8 \times w_{2,8}) + (i_8 \times w_{3,8}) + (i_8 \times w_{4,8}) + (i_8 \times w_{5,8}) + (i_8 \times w_{6,8}) + (i_8 \times w_{7,8}) + (i_8 \times w_{8,8}) + (i_8 \times w_{9,8}) + (i_8 \times w_{10,8}) + \ldots\ldots + (i_8 \times w_{63,8})$$

$$Y_{ke\text{-}9} = (i_9 \times w_{1,9}) + (i_9 \times w_{2,9}) + (i_9 \times w_{3,9}) + (i_9 \times w_{4,9}) + (i_9 \times w_{5,9}) + (i_9 \times w_{6,9}) + (i_9 \times w_{7,9}) + (i_9 \times w_{8,9}) + (i_9 \times w_{9,9}) + (i_9 \times w_{10,9}) + \ldots\ldots + (i_9 \times w_{63,9})$$

$$Y_{ke\text{-}10} = (i_{10} \times w_{1,10}) + (i_{10} \times w_{2,10}) + (i_{10} \times w_{3,10}) + (i_{10} \times w_{4,10}) + (i_{10} \times w_{5,10}) + (i_{10} \times w_{6,10}) + (i_{10} \times w_{7,10}) + (i_{10} \times w_{8,10}) + (i_{10} \times w_{9,10}) + (i_{10} \times w_{10,10}) + \ldots\ldots + (i_{10} \times w_{63,10})$$

.
.
.
.
.

$$Y_{ke\text{-}63} = (i_{63} \times w_{1,\,63}) + (i_{63} \times w_{2,\,63}) + (i_{63} \times w_{3,\,63}) + (i_{63} \times w_{4,\,63}) + (i_{63} \times w_{5,\,63}) + (i_{63} \times w_{6,\,63}) + (i_{63} \times w_{7,\,63}) + (i_{63} \times w_{8,\,63}) + (i_{63} \times w_{9,\,63}) + (i_{63} \times w_{10,\,63}) + \ldots\ldots + (i_{63} \times w_{63,\,63})$$

Penerapannya dalam source code digambarkan pada Gambar 3.5.11.

```
double perkalian_titik(double vektor1[],double vektor2[])     {
        int k,m,v;
        double rtn;
        rtn = 0.0;
        k = vektor1.length / 4;
        m = vektor1.length % 4;
        v = 0;
  while ( (k--)>0 ) {
        rtn += vektor1[v] * vektor2[v];
        rtn += vektor1[v+1] * vektor2[v+1];
        rtn += vektor1[v+2] * vektor2[v+2];
        rtn += vektor1[v+3] * vektor2[v+3];
        v+=4;      }
  while ((m--)>0 ) {
       rtn += vektor1[v] * vektor2[v];
       v++;      }
       return rtn;
```

Gambar 3.5.11. Source code Proses menghitung "perkalian titik"

Dari proses diatas, telah dijelaskan alur untuk mendapatkan output dari jaringan syaraf. Dari penjelasan diatas bobot antara node input dan node output akan menentukan output. Proses "perkalian titik" mengembalikan jumlah produk setiap elemen dalam vektor.

3.4.4 Layer4 : Seleksi Nilai Output Terbesar

Merupakan tahap akhir dari seluruh alur proses perhitungan JST Kohonen. pada layer ini akan memiliki 63 neuron output yang tidak ditargetkan. Nilai tersebut akan diputuskan yang mana sebagai neuron pemenang dalam metode JST Kohonen, yaitu hanya satu neuron pemenang. Pada layer ini hasil pengenalan pola citra ada pada bobot-bobot yang terdapat pada output neuron pemenang karena bobot pada neuron pemenang paling mendekati dengan pola yang dilatih pada jaringan. Pada output layer akan menghasilkan pemetaan dari perhitungan antara

input neuron dan bobot. Yang hasilnya juga akan sama dengan input neuron yaitu 63 output. Setelah itu akan dipilih neuron pemenang yaitu nilai dengan output terbesar adalah yang keluar keluar dengan neuron pemenang. Dalam sistem, rumusan untuk perhitungannya dirumuskan :

$$O_{terpilih} = y_{terbesar} > (f)y$$

Keterangan :
$O_{terpilih}$ = nilai output neuron terpilih
$y_{terbesar}$ = hasil output neuron terbesar
$(f)y$ = seluruh output neuron ($y_1, y_2, y_3, y_4, \ldots, y_{63}$)

Pada proses ini layer4 selesai. Yang mana akan diseleksi hasil output neuron terbesar yang akan keluar sebagai output neuron pemenang (P) yang akan diproses dan dikeluarkan pada layer5 (layer output neuron pemenang).

3.4.5 Layer5 : Output neuron pemenang

Layer output neuron pemenang merupakan layer akhir alur sistem JST Kohonen yang akan menghasilkan nilai output neuron yang terbesar yang akan menjadi neuron pemenang. Pada proses ini output yang keluar hanya satu neuron yang benar-benar menang. Maksutnya, output yang dihasilkan adalah berupa indeks neuron. Misalnya, jika pengguna akan mengidentifikasi suatu huruf Arab ke dalam aplikasi yaitu huruf ت pada inputan jaringan grid pixel 7x9, dan di dalam JST menemukan neuron yang menang adalah indeks ke-6, maka sistem mengidentifikasi bahwa neuron indeks ke-6 adalah neuron yang telah belajar mengenali pola huruf Arab ت yang juga dianggap sebagai neuron pemenang. Dalam sistem, rumusan untuk perhitungannya dirumuskan :

$$(P) = O_{terpilih}$$

Keterangan : P = neuron pemenang
$O_{terpilih}$ = nilai output neuron terpilih

Ilustrasi untuk mencari masing-masing output neuron dari perumusan diatas dapat dirumuskan :

$$P = O_{terpilih} > (f)y$$

Keterangan : $O_{terpilih}$ = nilai output neuron terpilih
P = neuron pemenang
$(f)y$ = seluruh output neuron (y_1, y_2, y_3, y_4, …. , y_{63})

Penerapannya dalam source code digambarkan pada Gambar 3.5.12.

```
palingBesar = -1.E30;
for (i=0; i< jumlah_outputNeuron; i++) {
optr = bobotKeluaran[i];
output[i] = perkalian_titik (inputan, optr) *fak_norm[0] + synth[0] *
 optr[jumlah_inputNeuron];
output[i] = 0.5 * (output[i] + 1.0);
if(output[i] > palingBesar) {
 palingBesar = output[i];
 menang= i;       }
```

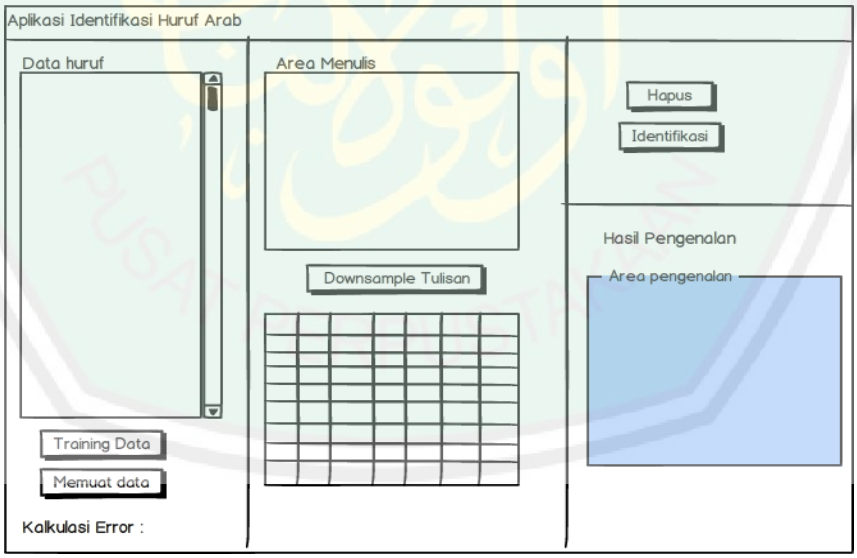
Gambar 3.5.12. Proses memilih dan menghasilkan neuron pemenang

3.5 Desain Tampilan Aplikasi

Dalam setiap aplikasi yang memerlukan *user* / pengguna dalam mengoperasikan suatu aplikasi, pastinya terdapat tampilan halaman tertentu yang digunakan *user* / pengguna dalam menjalankan suatu aplikasi tersebut. Tampilan tersebut biasanya disebut dengan halaman *interface.* Tampilan halaman *interface*

yang akan digunakan dalam aplikasi identifikasi yang akan dibangun ini, dibuat dengan menggunakan *tools-tools* bawaan aplikasi yang dibuat secara rapi yang bertujuan untuk mempermudah user dalam penggunaannya.

Dalam pembuatan aplikasi identifikasi ini akan dibuat hanya dengan menggunakan satu halaman *interface* saja. Halaman ini nantinya akan memuat semua proses awal hingga akhir dalam proses identifikasi. Dalam halaman *interface* ini terdapat beberapa komponen-komponen untuk beberapa proses, yaitu mulai dari proses *downsample* tulisan tangan oleh pengguna, proses *training* menggunakan JST Kohonen sampai proses untuk pengenalan dan pengidentifikasian tulisan tangan yang diinputkan pengguna dalam sistem. Tampilan sistem aplikasi dapat disajikan pada Gambar 3.6.



Gambar 3.6. Desain Halaman Aplikasi Identifikasi Huruf Arab

Terlihat pada tampilan halaman *interface* aplikasi beberapa komponen-komponen yang mendasari proses penggunaan aplikasi tersebut. Terdapat *textarea scroolPane* untuk menampung data-data huruf Arab yang akan diidentifikasi.

Textarea ini berisi data-data huruf Alif sampai Ya' yang berjumlah 28 huruf. Dalam sistem yang akan dibangun akan dibuat 56 data tulisan huruf Arab yang dari masing-masing huruf terdiri dari 2 sample data huruf.

Dalam aplikasi juga terdapat beberapa tombol-tombol yang dirancang menggunakan *button-button* aksi. Terdapat *button* "*Training Data*" yang berfungsi untuk melakukan proses *training* sample data huruf Arab Alif sampai Ya' yang berjumlah 56 sample huruf. Terdapat juga tombol "*Memuat Data*" yang berfungsi untuk me-*load* (memuat) data sample tulisan yang akan dimuat dari basisdata *sample.dat* yang berisi data-data huruf Arab yang akan ditampilkan sebagai *output* dari sistem. Pada sistem juga akan ditampilkan perhitungan kalkulasi *error* yaitu perhitungan tingkat kesalahan / *error* yang paling minimum untuk setiap proses *training* sample huruf Arab dalam basisdata. Kalkulasi error akan menampilkan nilai perhitungan *error* / kesalahan ketika pengguna menggunakan tombol "*memuat data*" lalu melakukan proses training menggunakan tombol "*training data*" dan seketika itu sistem akan menampilkan nilai perhitungan *error* berdasarkan dari kedekatan pola matriks data sample dan data input dari pengguna.

Terdapat sebuah *panel textarea* untuk menuliskan inputan huruf yang ingin diidentifikasi oleh pengguna. Lalu sebuah *panel area* yang berisi grid-grid kotak yang menampilkan sebuah *downsample* (pemetaan pola tulisan dari pengguna) yang pengguna inputkan pada area menulis huruf. Secara otomatis *downsample* menampilkan pemetaan pola inputan huruf dari pengguna yang akan tampil pada panel *downsample* berupa tampilan *pixel-pixel* aktif dan *pixel-pixel* nonaktif yang

direpresentasikan dengan sebuah warna hitam dan putih pada grid-grid kotak panel. Jika teridentifikasi terdapat inputan yang membentuk pola tulisan huruf Arab maka akan diwakili dengan *pixel* warna hitam dan bernilai 1, jika tidak teridentifikasi tedapat inputan yang membentuk pola tulisan huruf Arab / matriks kosong, maka akan diwakili dengan *pixel* warna putih dan bernilai 0.

Terdapat juga *button* tombol "*hapus*" untuk menghapus input tulisan dalam area menulis huruf dari pengguna serta tombol "*identifikasi*" untuk proses pengecekan dan pengenalan input huruf dari pengguna sekaligus proses identifikasi yang akan menampilkan *output* huruf yang dimaksud tersebut. Dan terakhir terdapat *panel* yang akan menampilkan identifikasi *output* huruf yang diinputkan untuk diidentifikasi oleh sistem.

BAB IV

HASIL DAN PEMBAHASAN

Bab ini membahas mengenai hasil uji coba sistem dan implementasi algoritma *Kohonen Neural Network* pada aplikasi yang telah dirancang dan dibangun. Tujuan dilakukan uji coba yaitu untuk mengetahui apakah sistem telah berhasil berjalan sesuai dengan target di awal dan apakah sistem juga telah berhasil dalam memenuhi tujuan utama penelitian sesuai dengan tujuan yang telah dibahas pada bab sebelumnya. Uji coba pada aplikasi ini juga menekankan dalam proses ketepatan pengenalan aplikasi yang mengacu pada kalkulasi tingkat *error* (kesalahan) dalam proses pengenalan tulisan.

4.1 Alat dan Bahan yang digunakan

Kebutuhan alat dan bahan untuk melakukan penelitian ini adalah sebagai berikut:

1. Kebutuhan Perangkat Keras (*Hardware*)

Sebuah PC / Laptop untuk melakukan perancangan dan pembangunan sistem dengan spesifikasi :

- Processor Intel (R) Core (TM) i3
- RAM 2 GB dan Memory minimal 1 GB
- Primary Harddisk Total 43 GB (Free harddisk Minimal 4 GB)

2. Kebutuhan Perangkat Lunak (*Software*)

- Sistem Operasi Windows 7 Ultimate 32-bit
- *Netbeans 7.2* sebagai IDE untuk melakukan penulisan *source code* program
- *Sample.dat* untuk melakukan penyimpanan data ciri tulisan tangan
- *Microsoft Office* untuk membuat dokumentasi dan laporan hasil penelitian.

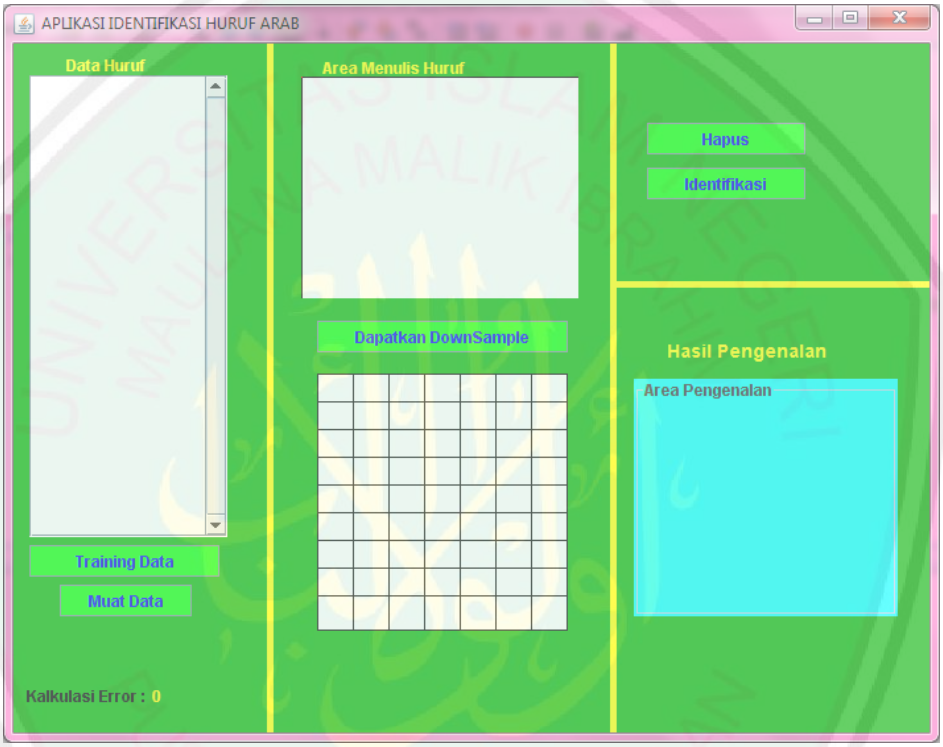
3. Kebutuhan data / Sumber data

Data tulisan tangan penulis masukkan berupa beberapa sample data tulisan tangan huruf Arab Alif sampai Ya' yang ditulis langsung pada *panel drawing* secara *online* untuk keperluan *training* terhadap sistem yang telah dibangun.

4.2 Deskripsi Program

Perancangan *desain interface* pada bab III diimplementasikan dengan menggunakan bahasa pemrogaman java pada editor *Netbeans 7.2* sehingga menghasilkan sebuah halaman *interface* aplikasi. Kriteria uji coba dari aplikasi identifikasi huruf Arab ini yaitu merujuk pada proses pengenalan tulisan tangan secara *online* dari pengguna menggunakan metode Jaringan syaraf Tiruan Kohonen. Sistem akan bekerja dalam mengenali ketepatan pola tulisan tangan huruf Arab dan akan melakukan proses identifikasi Huruf Arab yang ditulis oleh pengguna. Outputnya adalah hasil pengenalan Tulisan Huruf Arab yang digores

pada *panel* gambar dan akan memunculkan karakter Huruf Arab yang dimaksud. Huruf Arab yang digunakan dalam basisdata sistem adalah huruf Alif sampai Ya'. Tampilan halaman *interface*nya bisa dilihat pada Gambar 4.0.

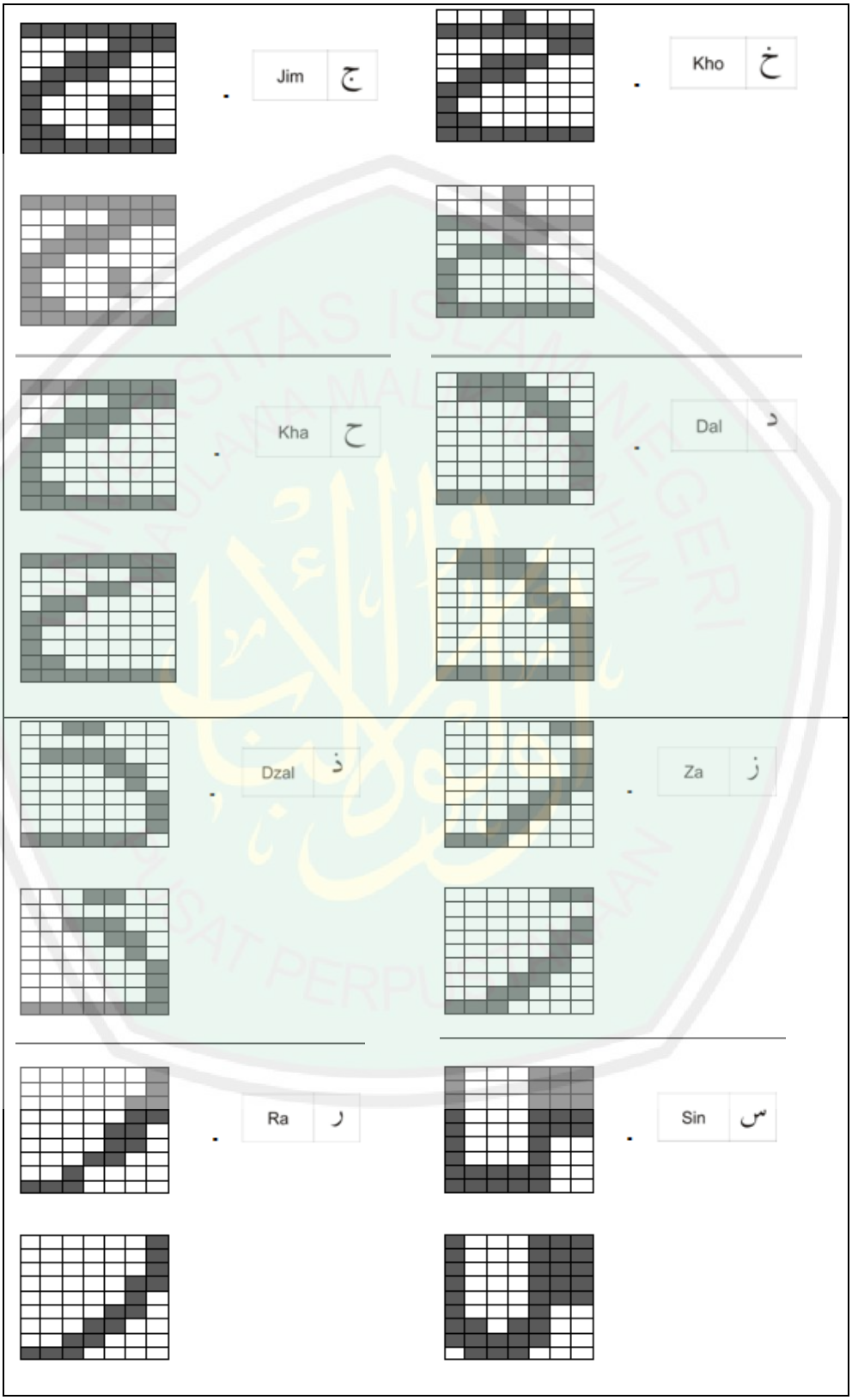


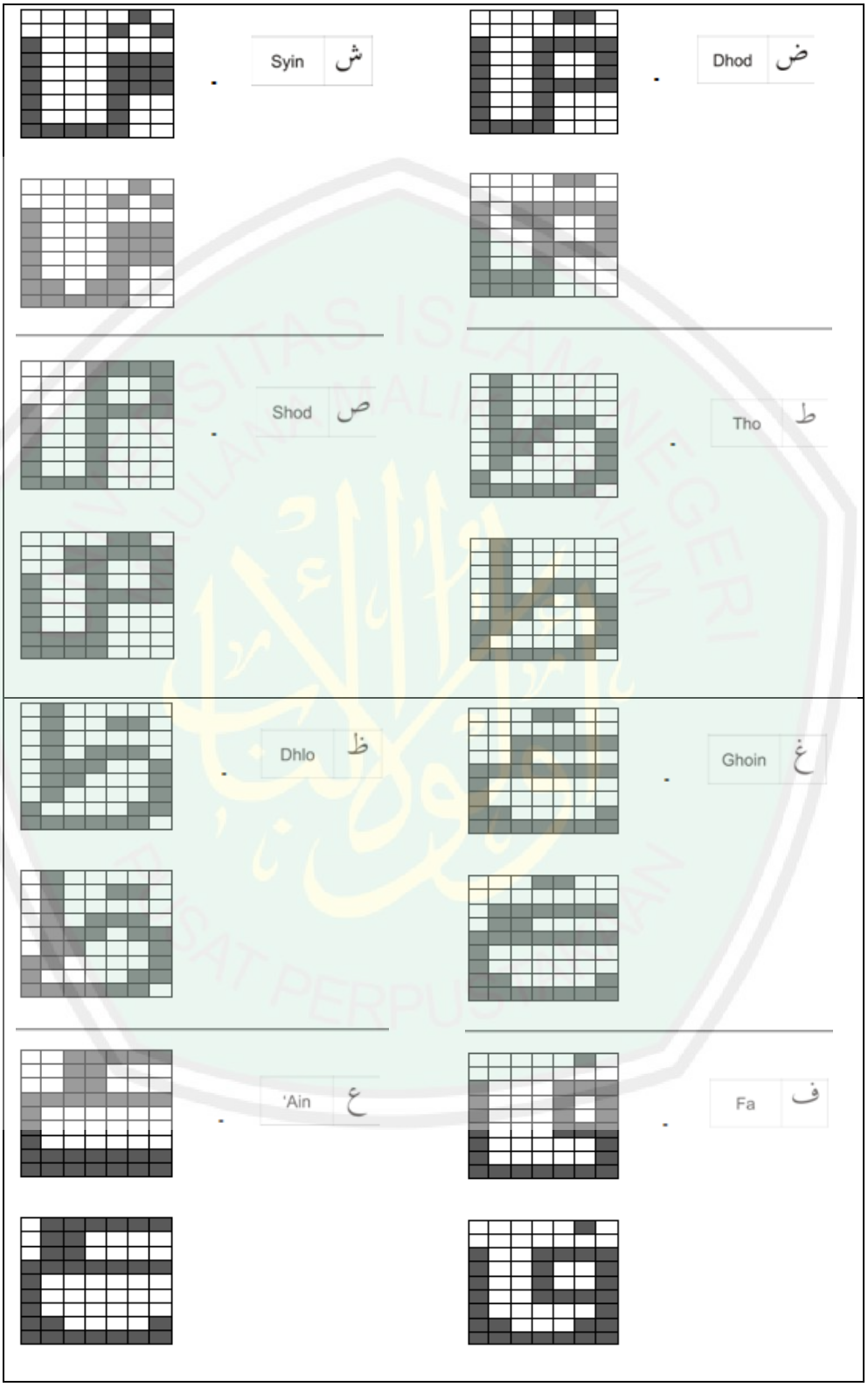
Gambar 4.0. Halaman *interface* Aplikasi Identifikasi Huruf Arab

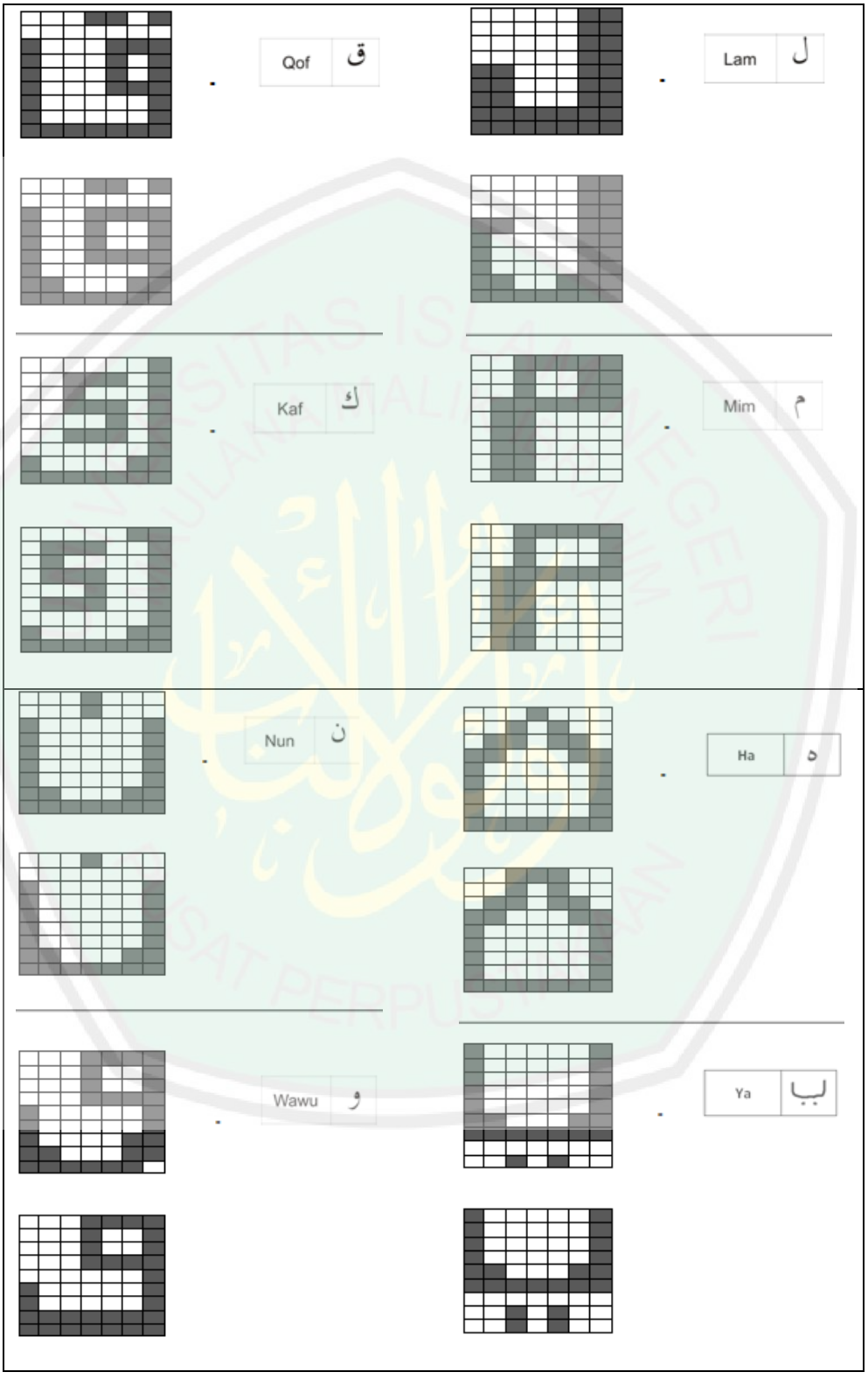
Pada sistem yang akan dibangun, mengarah pada data sample yang dibuat pada basisdata yang telah dimuat pada basisdata *sample.dat*. Terdapat data pokok yang harus dibuat ke database, yakni ciri pola huruf Arab Alif sampai Ya' yang dibuat dalam nilai matriks 1 untuk *pixel* aktif dan 0 untuk *pixel* nonaktif. Pada basisdata *sample.dat* terdapat 56 data sample huruf Arab. Masing-masing huruf dibuat 2 sample pola yang terdiri dari nilai matriks, misalnya huruf Alif dibuat 2 pola nilai matriks. Jumlah huruf Arab Alif sampai Ya' berjumlah 28 huruf, jadi

pada sistem yang dibangun akan dibuat 56 pola huruf Arab yang berisi nilai-nilai matriks 1 dan 0.

Tabel 1.1 Menunjukkan pola inputan huruf Arab Alif sampai Ya' yang dibuat menggunakan pemetaan pada grid-grid *pixel* 7x9







Tabel 1.2 Menunjukkan klasifikasi nilai matriks basisdata *sample.dat* berdasarkan pola inputan huruf Arab

0001000000110000010000001000000100000010000001000000110000001000	ا
0001100000100000010000001000000100000010000001000000100000011000	ا
1000001100000110000011000001100000110000011111111100000000001000	ب
1000001100000110000011000001100000111111110000000001000000001000	ب
0010100000000010000011000001100000110000011000001100000111111111	ت
0010100001010000000001000001100000110000011000001100000111111111	ت
0001000000000010101011000001100000110000011000001100000111111111	ث
0001000001010010000011000001100000110000011000001100000111111111	ث
1111111000011100111000111000110000010001101000110110000011111111	ج
1111111000011100111000111000110000010001001000100110000011111111	ج
1111111000011100111000111000110000010000001000000110000011111111	ح
1111111000001100011000110000110000010000001000000110000011111111	ح
0001000111111100000110011100011100011000001000000110000011111111	خ
0001000000100011111111000110001110001000000100000010000001111111	خ
0111000011110000001100000010000000100000010000001000000111111110	د
0111000011110000001000000110000001100000010000001000000111111111	د
0011000000000001111000000110000000100000001000000100000011111110	ذ
0001100000000000111000000110000000100000001000000100000011111111	ذ
0000001000000100000011000001100000110000011000011000010000111000	ر
0000001000000100000001000001100000100000110000110000110001110000	ر
0000011000000000000010000001000001100001100001100001100011100000	ز
0000011000000000000010000011000001000001100001100001100011100000	ز
1000111100011110001111000111100011110001001000100111110011111100	س
1000111100011110001111000111100011110001001101100111110001110000	س
0000010000010110000000100011110001111000111100010010001001111100	ش
0000010000010110000000100011110001111000111100010011011001111100	ش
0001111000100100010011001111110010001001000100100010010001111000	ص
0000110001111100110011001001100111111001000100100011110001111000	ص
0000110000000010011111001001100100110011111001000100100011110000	ض
0000110000000010111111011001100100110011111001000111100011110000	ض
0100000010000001000000101110011100101100010100001100001111111110	ط
0100000010000001000000101110011101101100011100001100001111111110	ط
0100000010011001000000101110011100101100010100001100001111111110	ظ
0100000010011001000000101110011101101100011100001100001111111110	ظ

00111110011000001100011111111100000010000001000000111111111111111	ع
01111110110000011000011111111100000010000001000000110000111111111	ع
00011000000000001111100100001111111100000001000000110000111111111	غ
00011000000000011111101100001111111100000001000000110000111111111	غ
00000100000000100011110001011000101100011110000011000001111111111	ف
00000100000000100111110010011001001100111110000011100011111111111	ف
00011010000000100011110001011000101100011110000011000001111111111	ق
00011010000000100111110010011001001100111110000011100011111111111	ق
00000010011101001000100111010000101001110100000011000011111111111	ك
00000110111001010000101110010001001011100100000011000011111111111	ك
00000110000011000001100000111100011110001111000111111111111111111	ل
00000110000011000001111000111000011100001110000111100111111111111	ل
00111110010011001001101111110110000011000001100000110000011000000	م
00111110010001001000101111110110000011000001100000110000011000000	م
00010000001000100000110000011000001100000110000011100011111111111	ن
00010000000000100000110000011000001100000110000011100011111111111	ن
00011110001001000100100011111000001100000110000011110001111111110	و
00011110001001000100100011111000000110000011000001111111111111111	و
00010000010100011011011000111000001100000110000011000001111111111	ه
00111000010100011011011000111000001100000110000011000001111111111	ه
10000011000001100000110000011000001110001111111110000000001010000	ي
10000011000001100000110000011100011111111100000000010100001010000	ي

Pada basisdata *sample.dat* berisi kumpulan data-data masukan semua huruf arab Alif sampai Ya' yang dibuat pola inputannya dahulu untuk proses pelatihan, pengenalan dan identifikasi. Jadi, sistem tidak langsung bisa mengenali input tulisan yang diinputkan oleh pengguna. Data-data pada basisdata pertama harus dilakukan *training* dahulu agar nantinya bisa mengenali input tulisan dari pengguna. Intinya, jika pengguna ingin memakai sistem ini, maka pengguna diwajibkan untuk merujuk pada pola inputan yang telah ada dalam basisdata. Jika pengguna ingin memakai sistem aplikasi ini dan pengguna tidak merujuk pada pola inputan yang telah ada, maka proses pengenalan dan identifikasi akan

berjalan sulit atau bahkan hasil yang keluar akan tidak bisa mengenali input tulisan dari pengguna.

4.3 Implementasi dan Pembahasan

4.3.1 Proses Input Gambar

Proses input gambar dimulai dengan menginputkan tulisan tangan ke dalam system. Tulisan kemudian di-*capture* agar menjadi sebuah gambar / citra yang kemudian dirubah ke dalam citra biner. Dan diimplementasikan dengan *source code* proses *Image processing* yang ditunjukkan pada Gambar 4.0.1.

```
        entryImage = createImage(getWidth(),getHeight());
        entryGraphics = entryImage.getGraphics();
        entryGraphics.setColor(Color.white);
        entryGraphics.fillRect(0,0,getWidth(),getHeight());  }
    public void paint(Graphics g)      {
        if (entryImage==null)
        initImage();
        g.drawImage(entryImage,0,0,this);
        g.setColor(Color.black);
        g.drawRect(0,0,getWidth(),getHeight());
        g.setColor(Color.red);
        g.drawRect(downSampleKiri,downSampleAtas,
        downSampleKanan-downSampleKiri,
        downSampleBawah-downSampleAtas);}
```

Gambar 4.0.1. *Source code* proses *Image processing*

Gambar yang diinputkan akan ditangani oleh proses mouse. Dalam sistem akan dibuat aksi proses *mousemotionevent.* Dalam sistem akan dibuat menulis menggunakan inputan langsung menggunakan mouse pada panel gambar. Maka akan ada proses untuk mengambil titik-titik yang menghubungkan dari satu titik ke titik yang lain. Dalam sistem yang dibangun tidak cukup hanya menangani titik-titik, nantinya titik-titik ini akan menjadi suatu garis yang terdiri dari banyak titik. Dalam prosesnya, program ini memiliki waktu untuk menerima semua nilai

untuk memproses inputan mouse. Dengan menggambar titik-titik yang akan membentuk suatu garis, maka akan dibuat agar tulisan dapat diproses dengan menggunakan *source code* Gambar 4.0.2.

```
protected void processMouseMotionEvent(MouseEvent e)      {
      if (e.getID()!=MouseEvent.MOUSE_DRAGGED)
      return;
      entryGraphics.setColor(Color.black);
      entryGraphics.drawLine(lastX,lastY,e.getX(),e.getY());
      getGraphics().drawImage(entryImage,0,0,this);
      lastX = e.getX();
      lastY = e.getY();
```

Gambar 4.0.2. Source code Input disimpan dalam format gambar *off-screen*

Pada *source code* diatas, adalah proses agar tulisan apapun yang pengguna inputkan akan disimpan ke dalam format gambar *off-screen*. Gambar *off-screen* adalah gambar yang diciptakan dari proses goresan tulisan tangan dari pengguna pada *panel drawing* yang terdiri dari kumpulan titik-titik yang membentuk suatu garis melalui proses *mousemotionEvent*.

4.3.2 Proses Downsample Gambar

Proses downsample adalah proses dimana input tulisan yang pengguna goreskan pada panel *drawing* akan diambil dan ditampilkan sistem pada grid-grid / kotak-kotak kecil (*pixel*) yang telah dibuat untuk menampilkan tulisan dari pengguna. Pada kotak downsample, gambar akan diambil dan dipetakan ke dalam kotak grid yang mempunyai resolusi tertentu. Pada sistem yang dibangun, kotak downsample dibuat dengan resolusi *pixel* dengan lebar *pixel* 7 dan tinggi *pixel* 9.

Pada proses ini terdapat sub-proses untuk menyimpan gambar downsample yang disimpan dalam data sample. Data sample disini berupa *array* yang berupa bilangan bulat untuk menyimpan gambar downsample yang prosesnya dapat

diimplementasikan pada *source code* yang berfungsi untuk menyimpan gambar *downsample* yang ditunjukkan pada Gambar 4.0.3.

```
      boolean grid[][];
      char tulisan;
public SampleData_Inputan(char tulisan,int lebar,int tinggi)  {
      grid = new boolean[lebar][tinggi];
      this.tulisan = tulisan;  }
 void kumpulanData(int x,int y,boolean n)   {
    grid[x][y]=n;  }
 public boolean dapatData(int x,int y)    {
    return grid[x][y]; }
 public int dapatTinggi()    {
    return grid[0].length;  }
 public int dapatLebar()      {
    return grid.length;  }
 public char dapatTulisan()   {
    return tulisan;  }
 void kumpulanTulisan(char tulisan)     {
    this.tulisan = tulisan; }
  public int compareTo(Object o)     {
    SampleData_Inputan objek = (SampleData_Inputan)o;
    if (this.getLetter()>objek.getLetter()
```

Gambar 4.0.3. Source code inputan gambar disimpan dalam grid *pixel* 7x9

Pada proses diatas inputan gambar akan disimpan dalam downsample grid *pixel* lebar 7 dan tinggi 9. Dalam proses diatas juga terdapat proses untuk mengatur dan mendapatkan data yang terkait dengan downsample jaringan.

4.3.3 Mengatur Ukuran dan Posisi

Semua inputan tulisan yang telah diproses dalam pemrosesan citra diatas akan mengalami proses downsample sebelum dilakukan proses pengenalan dan identifikasi. Proses downsample bertujuan untuk mencegah jaringan syaraf dari kerumitan dan kesulitan dengan masalah ukuran dan posisi tulisan dalam *panel drawing* dan agar tidak lagi menjadi rumit dan sulit dalam proses pengenalan dan identifikasi nantinya.

Pada area menulis dalam *panel drawing*, dibuat dengan ukuran yang cukup lebar agar pengguna bisa leluasa menggambar pada layar. Ketika pengguna menggoreskan sebuah karakter huruf arab yang merujuk pada *data sample* dalam database, pastinya bisa terletak dimana saja dan tidak sama dengan pola *data sample* pada database. Oleh karena itu, dalam sistem akan dibuat proses *downsample* karakter masing-masing huruf arab yang bertujuan untuk memudahkan proses pengenalan. Dengan proses *downsample*, gambar dalam panel akan berukuran konsisten, baik itu berukuran besar atau kecil sistem masih akan bisa tetap mengenalinya.

Ketika pengguna menggambar sebuah karakter, hal pertama yang dilakukan adalah sistem mengambil kotak di sekitar batas tulisan. Hal ini memungkinkan sistem untuk menghilangkan semua ruang putih (yang tidak terdapat goresan) di sekitar tulisan. Setelah itu, dilakukan proses pemotongan / pemangkasan gambar yang termasuk juga di dalam proses *downsample*. Proses pemotongan / pemangkasan gambar ini dilakukan dengan mengambil pola bit gambar. Dalam proses ini akan digunakan metode "*PixelGrabber*". Metode "*Pixel Grabber"* adalah salah satu Algoritma Pencarian untuk menghasilkan matriks *pixel*.

Algoritma Metode "*Pixel grabber"* digambarkan dalam alur Algoritma :

i. Cari batas teks seluruh gambar dengan pemindaian dari atas ke bawah untuk batas atas Y1, kiri ke kanan untuk batas kiri X1, kanan ke kiri untuk batas kanan X2 dan bawah ke atas untuk batas terendah Y2 dari atas, kiri, kanan dan bawah sisi area gambar.
ii. *Scanning* citra biner dari Y1 terhadap Y2
iii. Jika ada *pixel* hitam, kemudian *scan* dari X1 ke X2 untuk baris tertentu untuk mendeteksi *pixel* putih.

iv. Jika tidak ada *pixel* putih yang ditemukan, maka terdapat celah baris,
v. Ulangi langkah ii - iv untuk seluruh gambar untuk menemukan jumlah celah baris.
vi. Jika *pixel* hitam ditemukan, maka karakter *pixel* dianggap sebagai '1'.
vii. Jika *pixel* putih ditemukan, maka karakter *pixel* dianggap sebagai '0'.
viii. Ulangi langkah vi – vii untuk mendapatkan representasi biner dari gambar.

Pada tahap ini metode "*pixelGrabber*" akan diterapkan pada grid *pixel* dengan lebar *pixel* 7 dan tinggi *pixel* 9. Penerapan metode ini jika diterapkan pada *source code* metode "*pixelGrabber*" pada Gambar 4.0.4.

```
void downSample()    {
     int lebar = entryImage.getWidth(this);
     int tinggi = entryImage.getHeight(this);
     PixelGrabber grabber = new PixelGrabber(entryImage,0,0,lebar, tinggi,true);
try  {
      grabber.grabPixels();
      pemetaanPixel = (int[])grabber.getPixels();
      cariBatas(lebar,tinggi);
      SampleData_Inputan data = sample.getData();
      rasioX =(double)(downSampleKanan-downSampleKiri) /(double)data.getWidth();
      rasioY =(double)(downSampleBawah-downSampleAtas) /(double)data.getHeight();
```

Gambar 4.0.4. Source code metode "*pixelGrabber*" pada grid *pixel* 7x9

Pada kode diatas, terdapat variabel "*pemetaanPixel"*. Variabel "*pemetaanPixel*" ini berfungsi untuk menyimpan *array* dari data yang berbentuk *integer* yang *array* data ini mengandung pola bit gambar. Setelah itu adalah tahap memotong gambar (*cropping*) dan menghapus spasi antar goresan pixel gambar. *Cropping* dilakukan dengan menghilangkan empat baris / kotak *pixel* dari sisi atas, kiri, bawah dan kanan dari gambar. Akan terdapat garis-garis yang menangkap / mengklaim area yang telah dipotong yang mewakili dari gambar. Garis-garis ini akan segera mengambil area tulisan setelah garis-garis ini menemukan dan men-*scanning* pixel yang terdapat goresan. Dengan cara ini, empat baris dari sisi atas, kiri, bawah dan kanan gambar akan dibuang

(dihilangkan) ke tepi luar gambar. Penerapannya dapat dilihat pada *source code* pada Gambar 4.0.5.

```
    //Untuk garis horisontal
protected boolean GarisHorison(int y)   {
    int lebar = entryImage.getWidth(this);
    for(int i=0; i<lebar; i++) {
    if (pemetaanPixel[(y*lebar)+i]!=-1)    }
    //untuk garis vertikal
protected boolean GarisVertikal(int x)   {
    int lebar = entryImage.getWidth(this);
    int tinggi = entryImage.getHeight(this);
    for(int i=0; i<tinggi; i++) {
    if(pemetaanPixel[(i*lebar)+x]!=-1)    }
```

Gambar 4.0.5. Source code untuk menerima koordinat *y* garis batas tulisan

Pada *source code* diatas untuk garis Horisontal, terdapat metode untuk menerima koordinat *y* yang berfungsi untuk mengecek dan memeriksa garis horizontal. Kemudian dibuat perulangan (*looping*) melalui setiap koordinat *x* pada baris yang dicek dan diperiksa , pengecekan itu dilakukan untuk melihat nilai-nilai tiap *pixel* apakah ada atau tidak (terdapat *pixel* (bernilai 1) atau tidak terdapat *pixel* (bernilai 0)). Jika tidak terdapat *pixel*, maka *pixel* akan dihilangkan. Setelah itu akan dilakukan perhitungan empat sisi untuk mencari ruang kosong / celah / spasi dalam gambar, yang ditunjukkan dalam *source code* Gambar 4.0.6.

```
    // sisi kiri
for (int x=0; x<lebar; x++) {
    if (!GarisVertikal(x)) {
    downSampleKiri = x;
    break;   }
```

Gambar 4.0.6. Source code untuk menghitung garis dan melakukan "cropping"

Dari *source code* diatas dapat dijelaskan bahwa sistem akan melakukan proses perhitungan garis sebelah kiri dan melakukan proses pemotongan tepi luar gambar sebelah kiri. Untuk menghitung garis kiri dan melakukan pemotongan *grid pixel*, sistem akan memulainya pada angka 0 dan terus ke bagian kiri gambar.

Begitu garis pertama ditemukan, maka sistem akan menetapkan garis pertama ini sebagai potongan bagian kiri *grid pixel*. Cara yang sama dilakukan sistem untuk menangani sisi atas, bawah dan kanan.

4.3.4 Cara Kerja Downsample dalam Sistem

Dari langkah sebelumnya, diketahui telah dilakukan proses pemotongan yang melibatkan proses downsample. Hal tersebut melibatkan pengambilan gambar dari resolusi yang lebih besar untuk resolusi 7x9. Untuk melihat bagaimana mengurangi domain area gambar ke resolusi 7 x 9, maka system harus berupaya untuk agar gambar dengan resolusi tinggi dapat ditampung dalam grid pixel. Dalam hal ini system akan membagi gambar ke area daerah dibawah resolusi 7x9. Jika ada pixel di suatu daerah penuh, maka pixel yang sesuai pada downsample gambar 7x9 juga akan mengisinya. Prosesnya dapat diterapkan pada *source code* Gambar 4.0.7.

```
int w = entryImage.getWidth(this);
int startX = (int)(downSampleKiri+(x*rasioX));
int startY = (int)(downSampleAtas+(y*rasioY));
int endX = (int)(startX + rasioX);
int endY = (int)(startY + rasioY);
for (int yy=startY; yy<=endY; yy++) {
for (int xx=startX; xx<=endX; xx++) {
int area = xx+(yy*w);
if (pemetaanPixel[area]!= -1 )
return true;
```

Gambar 4.0.7. Source code membagi gambar ke area daerah dibawah resolusi 7x9

Pada proses yang diterapkan pada *source code* diatas, system menerima jumlah area wilayah yang harus dihitung. Pertama, koordinat awal dan akhir x dan y harus dihitung. Untuk menghitung koordinat x pertama untuk suatu area wilayah tertentu, pertama *downsampleKiri* digunakan, ini adalah sisi kiri grid

kotak yang akan dilakukan pemotongan. Maka koordinat x akan dikalikan dengan *rasioX* , yang merupakan parameter untuk mewakili berapa banyak jumlah pixel masing-masing area. Hal ini memungkinkan untuk menentukan dimana untuk menempatkan *startX*. Posisi awal *startX* dihitung dengan cara yang sama. Selanjutnya program ini melakukan proses pengulangan *(looping*) melalui setiap koordinat x dan y di daerah tertentu. Jika bahkan satu pixel harus ditentukan untuk diisi, maka system akan membenarkan dengan method *return true*; yang mewakili bahwa area ini harus dipenuhi. Pada proses ini dihasilkan sample gambar yang akan disimpan dalam system yang diwakili dalam suatu *class*. *Class* ini akan berisi array nilai Boolean 7x9.

4.3.5 Penerapan Algoritma Jaringan Syaraf Tiruan Kohonen

Pada system akan menerapkan algoritma JST Kohonen yaitu suatu metode yang mempunyai input, ouput dan sebuah nilai bobot. Pada system yang dibangun pola karakter *downsample* yang diambil oleh pengguna diteruskan ke sebuah input neuron. Neuron disini dimaksudkan sebuah jaringan awal / masukan awal untuk masuk dalam perhitungan metode JST kohonen yang dirupakan dengan sebuah nilai. Terdapat 1 input neuron untuk setiap pixel dalam gambar downsample. Karena gambar downsample adalah dibuat dengan resolusi pixel 7x9 yang dirupakan dengan sebuah grid, yang berarti akan ada 63 inputan neuron dalam input awal ke jaringan Kohonen.

Dalam metode JST Kohonen juga akan ada sebuah output neuron. Output neuron disini dimaksudkan dengan sebuah neuron keluaran dari sebuah jaringan Kohonen yang telah diproses dalam metode dari sebuah inputan awal neuron yang

juga diwakilkan oleh sebuah nilai. Neuron Output jaringan syaraf berpikir seperti apa yang pengguna gambar pada area gambar. Jumlah neuron output selalu sesuai dengan jumlah sample tulisan yang dibuat. Yaitu dalam system sejumlah 56 neuron output tulisan huruf arab.

Selain sebuah neuron input dan output, akan ada juga koneksi / relasi antara neuron per individu. Koneksi ini tidak selalu sama, yang mana setiap koneksi ini mempunyai tugas masing-masing yang pada akhirnya akan menentukan apakah suatu jaringan akan menampilkan untuk suatu pola masukan yang dimaksud dan diberikan oleh pengguna. Untuk menentukan jumlah koneksi, system bekerja dengan cara mengalikan jumlah neuron input dengan jumlah neuron output. Sebuah jaringan syaraf dengan 56 neuron output dan 63 neuron input akan memiliki total 3528 bobot koneksi. Proses *training* ini dibuat untuk menemukan nilai-nilai yang benar untuk bobot-bobot tersebut. Proses pengenalan (*recognize*) dimulai ketika pengguna menggambar sebuah karakter huruf arab alif sampai ya' yang tentunya harus merujuk pada pola yang ada pada *sample.dat* pada system yang telah dibuat sebelumnya guna memudahkan proses pengenalan dan kemudian memprosesnya dengan melakukan proses pengenalan dan identifikasi. Pertama tulisan downsample untuk area grid pixel 7x9. Downsample gambar ini harus disalin (*copy*) dari array 2 dimensi untuk array ganda yang akan diteruskan ke neuron input (neuron masukan awal pada jaringan) yang diterapkan pada *source code* Gambar 4.0.8.

```
 Masukan.downSample();
double inputan[] = new double[7*9];
int index=0;
SampleData_Inputan datas = sample.getData();
   for (int y=0; y<datas.getHeight(); y++) {
   for (int x=0; x<datas.getWidth(); x++) {
   inputan[index++] = datas.getData(x,y)?.5:-.5;     }
```

Gambar 4.0.8. Source code proses menerima neuron masukan awal pada jaringan

Dari proses diatas, sistem melakukan konversi (*perubahan struktur data*) dari input berupa goresan gambar dari pengguna menjadi sebuah nilai matriks pixel yang berupa nilai Boolean. Sebuah neuron (unit pemroses informasi) pada JST membutuhkan suatu *floating point* (bilangan yang dapat digunakan untuk merepresentasikan sebuah nilai yang sangat besar atau sangat kecil) dan dalam system akan diberikan nilai 5 untuk pixel hitam dan nilai 5 untuk pixel putih. Array 7x9 yang akan bernilai 63 selanjutnya akan diteruskan ke output neuron. Hal ini dilakukan dengan melewatkan inputan array 7x9 ke parameter "*pemenang*" pada metode JST Kohonen. Ini akan mengembalikan yang mana dari 63 neuron akan memungkinkan menang dan akan disimpan dalam sebuah parameter integer yang diterapkan dalam *source code* Gambar 4.0.9.

```
 int terbaik = jaringan.pmenang(inputan,fak_normalisasi,sinth);
char peta[] = pemetaan_neuron();
JOptionPane.showMessageDialog(this,"Proses Identifikasi!"     ,"Memproses
                Tulisan",JOptionPane.INFORMATION_MESSAGE);
Message.setText("Dikenali :   "  + peta[terbaik]);
Message.setFont(new Font("", Font.BOLD, 28));
Message.setBounds(4,30,190,50);
char peta1[] =pemetaan_neuron();
hapus_aksiPerform(null);     }
```

Gambar 4.0.9. Source code winner neuron yang disimpan dalam sebuah parameter

Pada tahap ini ternyata sebuah neuron pemenang tidak terlalu membantu dalam system, karena tidak menunjukkan tulisan mana yang benar-benar dikenali.

Untuk mengatasi masalah ini, maka system akan melapisi / memproses neuron hingga sebuah input tulisan akan dikenali, dengan cara setiap gambar karakter huruf arab yang akan dilatih akan dimasukkan ke dalam jaringan dan neuron pemenang haruslah ditentukan terlebih dahulu. Misalnya jika pengguna menulis huruf ت ke dalam jaringan JST dan neuron pemenang adalah berada pada neuron ke-10, maka neuron ke-10 adalah neuron yang telah belajar untuk mengenali pola ت . Hal ini dilakukan dengan membuat suatu method yang berfungsi untuk mengembalikan array karakter dan indeks setiap elemen array sesuai dengan jumlah neuron yang mengenali karakter.

Proses dalam system dominan dilakukan oleh jaringan syaraf di dalam metode pemenang. Dalam metode pemenang tidak dilakukan proses penormalan masukan dan proses perhitungan nilai output dari setiap neuron output. Bagaimanapun keluaran neuron yang memiliki nilai output terbesar adalah dianggap sebagai pemenang. Akan dibuat sebuah variable yang disetting ke jumlah yang sangat kecil untuk menunjukkan bahwa belum ada pemenang, yang diterapkan dalam *source code* Gambar 4.0.10.

```
 int i, menangg=0;
double nilai_terbesar, optimal[];
normalisasi_Input(input, fak_Normalisasi, sinth);
nilai_terbesar = -1.E30;
for (i=0; i<juml_outputNeuron; i++) {
optimal = bobots_output[i];
output[i] = perkalianTitik(input,optimal) * fak_Normalisasi[0]+ sinth[0] *
            optimal[juml_inputNeuron];
output[i] = 0.5 * (output[i] + 1);
if (output[i] > nilai_terbesar) {
nilai_terbesar = output[i] ;
menangg = i;    }
```

Gambar 4.0.10. Source code menghitung neuron output dengan mengalikan bobot

Bobot setiap neuron output dihitung dengan cara mengambil nilai dari proses perkalian titik-titik dari masing-masing perkalian bobot neuron ke input neuron. Perkalian titik-titik dihitung dengan mengalikan masing-masing dari nilai input-input neuron terhadap bobot antara neuron input dan neuron output. Bobot ini ditentukan selama proses training. Output disimpan dan jika output terbesar sejauh ini telah ditemukan, maka output ini sudah dinyatakan sebagai neuron pemenang.

4.3.6 Training Jaringan Syaraf Tiruan Kohonen

Proses *Training* dalam system yang akan dibangun mewakili pada proses belajar adalah sebuah proses pemilihan bobot matriks neuron yang benar-benar yang akan bisa mengenali pola inputan pada system. Sebuah proses belajar pada JST Kohonen akan melakukan proses belaja dengan terus-menerus mengecek , mengevaluasi dan mengoptimalkan bobot matriks. Untuk melakukan hal ini, bobot matriks awal harus ditentukan. Bobot matriks awal dibuat dengan memilih bobot matriks secara acak. Bagi system, tentu ini merupakan aturan yang kurang cocok untuk membuat bobot matriks secara acak, tetapi akan memberikan titik awal untuk proses pelatihan dan pencocokan yang akan dilakukan nantinya.

Setelah bobot matriks awal ditentukan maka proses *training* dapat dimulai. Pertama, bobot matriks dievaluasi untuk menentukan tingkat kesalahan. Kesalahan *(error)* ini ditentukan oleh seberapa baik pelatihan pemetaan pola inputan (goresan tulisan pada *canvas* yang dilakukan oleh pengguna) untuk suatu output neuron. Ketika tingkat *error* rendah, misalnya di bawah 10%, maka proses *training* selesai.

Proses *training* dimulai ketika pengguna menggunakan tombol "*training*" pada program. Ini menjadi awal pelatihan dengan melakukan proses perhitungan jumlah neuron input dan neuron output. Langkah pertama yaitu jumlah neuron input pada jaringan ditentukan dari ukuran gambar downsample. Karena pada system ditentukan resolusi gambar downsample adalah 7x9 pixel, maka jumlah neuron inputan akan berjumlah 63. Jumlah neuron output akan menyesuaikan seberapa banyak karakter huruf yang telah dibuat dan disimpan pada sample.dat. yang prosesnya diterapkan pada *source code* Gambar 4.0.11.

```
int Neuron_masukan   = Desain_Tampilan.tinggiDOWNSAMPLE*
                       Desain_Tampilan.lebarDOWNSAMPLE;
int Neuron_keluaran  = daftar_tulisan.size();
```

Gambar 4.0.11. Source code mengatur ukuran JST agar tetap mengenali inputan

Pada proses diatas, adalah proses dimana sistem telah menentukan ukuran jaringan syaraf yang bertujuan untuk agar sistem tetap mengenali input tulisan sesuai dengan sample yang ada pada sistem. Misalnya jika pada sistem dibuat 2 atau 3 sample per tulisan huruf arab, maka harus dipastikan bahwa jumlah output neuron tetap 56.

Proses selanjutnya adalah membangun kumpulan training untuk memproses data yang ditraining dan juga membangun jaringan syaraf tempat untuk menampung proses data input yang akan diproses pada suatu jaringan. Kumpulan data training dibangun untuk menampung jumlah yang sesuai dan cocok dengan sample.

Pada system akan disediakan sebanyak 56 tulisan huruf arab / hijaiyah yang akan dirujuk yang prosesnya dapat diterapkan pada *source code* Gambar 4.0.12.

```
Proses_TrainingJST set = new Proses_TrainingJST (Neuron_masukan,Neuron_keluaran);
kumpulan.juml_setTrainingSet(daftar_tulisan.size());
for(int t=0; t<daftar_tulisan.size(); t++)  {
int indeks=0;
SampleData_Inputan datas =  (SampleData_Inputan)daftar_tulisan.getElementAt(t);
   for (int y=0; y<datas.getHeight(); y++) {
   for (int x=0; x<datas.getWidth(); x++) {
   kumpulan.setInput(t,indeks++,datas.getData(x,y)?.5:-.5);    }
```

Gambar 4.0.12. Source code membangun jaringan syaraf & kumpulan training saat bekerja

Pada proses diatas tampak jaringan syaraf telah dibangun dan kumpulan training telah berjalan. Dengan kumpulan training yang telah ditetapkan dan dibuat, maka proses belajar bisa diberlakukan. Dalam hal ini, system akan menyesuaikan bobot matriks sampai jaringan terlatih yang diterapkan pada *source code* Gambar 4.0.13.

```
jaringan=new JST_Kohonen(Neuron_masukan,Neuron_keluaran,this);
jaringan.setTrainingSet(kumpulan);
jaringan.learn();      )
```

Gambar 4.0.13. Source code menyesuaikan bobot matriks sampai jaringan terlatih

Pada system juga akan ditampilkan proses untuk melihat dan mengevaluasi seberapa baik bobot-bobot saat berkerja. Hal ini ditentukan dengan melihat seberapa baik data training menyebar di seluruh output neuron. Jika banyak output neuron yang mengenali pola pelatihan yang sama, maka kumpulan bobot matriks tidak akan bekerja dengan baik. Berdasarkan hal tersebut, maka tingkat error harus dihitung, yang didasarkan pada bagaimana agar kumpulan pelatihan menyebar dengan baik di seluruh neuron output.

Setelah error dihitung dan ditentukan, system harus melihat apakah error samapai saat ini berada dibawah error terbaik (error kecil). Jika error kecil, maka error ini dikategorikan ke error terbaik (error terkecil) dan bobot neuron akan dipilih dan diproses. Yang prosesnya diterapkan pada *source code* Gambar 4.0.14.

```
evaluasi_error(rate,Metod_blajar,won,terbesar,koreksi,kerja);
totalError = terbesar[0] ;
if (totalError < error_terbaik) {
error_terbaik = totalError;
salin_bobot(jaringan_terbaik,this);   }
```

Gambar 4.0.14. Source code menghitung error dan mengecek error terkecil

Jumlah neuron pemenang kemudian dihitung yang bertujuan untuk system akan menentukan apakah tidak ada neuron output yang aktif (sesuai dan cocok dengan sample). Selain itu, jika error berada dibawah 10% (error kecil), maka proses training selesai. Prosesnya diterapkan pada *source code* Gambar 4.0.15.

```
pemenang = 0;
for(i=0; i<won.length; i++)
if(won[i]!=0)
pemenang++;
if (terbesar[0] < quitError)
break;
```

Gambar 4.0.15. Source code **e**rror bernilai minimum & maka proses training selesai

Jika tidak ada jumlah perhitungan jumlah output neuron yang menang, maka satu neuron dipilih untuk menjadi neuron pemenang yang ditunjukkan pada *source code* Gambar 4.0.16.

```
if ((pemenang < juml_outputNeuron)&&(pemenang < latih.dapatJuml_TrainingSet())) {
pemenang_utama(won);
continue ;  }
```

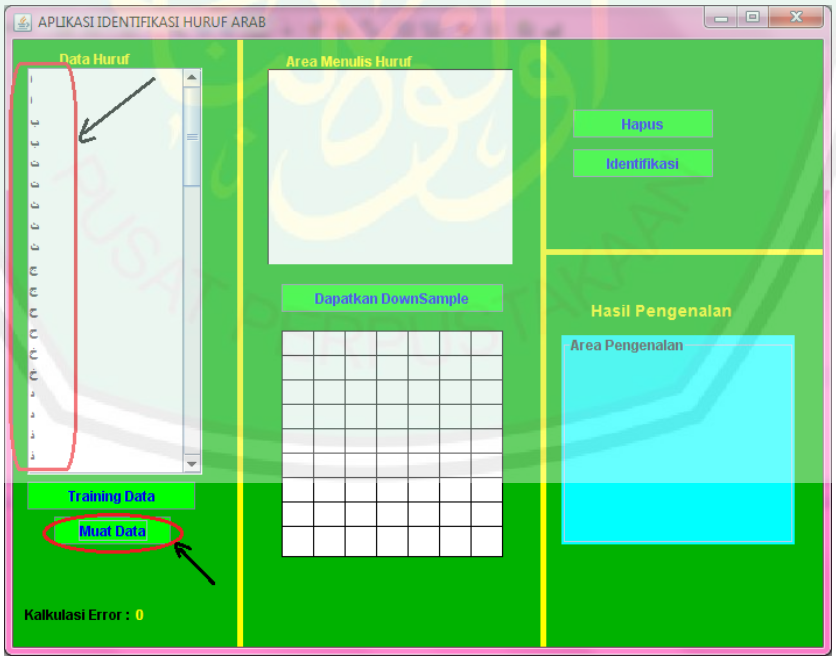
Gambar 4.0.16. Source code memilih satu neuron yang benar-benar menang

Bobot matriks awal telah dievaluasi dan ditentukan, hal itu disesuaikan berdasarkan errornya. Penyesuaian tersebut sedikit dilakukan berdasarkan koreksi yang dihitung saat kesalahan / error ditentukan. Yang proses selanjutnya adalah menghitung error dan menyesuaikan bobot matriks diteruskan sampai kesalahan bernilai kecil.

4.4 Hasil Uji Coba Aplikasi

Pada sistem akan dilakukan uji coba aplikasi dengan memasukkan / menginputkan data real. Yaitu dengan menginputkan 56 pola data tulisan huruf arab. Inputan goresan huruf tulisan arab akan di-*capture* dan diubah menjadi format ekstensi .jpeg dan akan di *downsample* ke pixel dengan resolusi lebar 7 dan tinggi 9 yang direpresentasikan dengan suatu grid kotak persegi downsample tulisan.

Pada sistem yang dibangun, pertama pengguna harus mengklik tombol "***Muat Data***" pada aplikasi ini. Hal ini bertujuan untuk memuat data-data tulisan huruf Arab pada basisdata "*sample.dat*" pada sistem yang ditunjukkan pada Gambar 4.1.

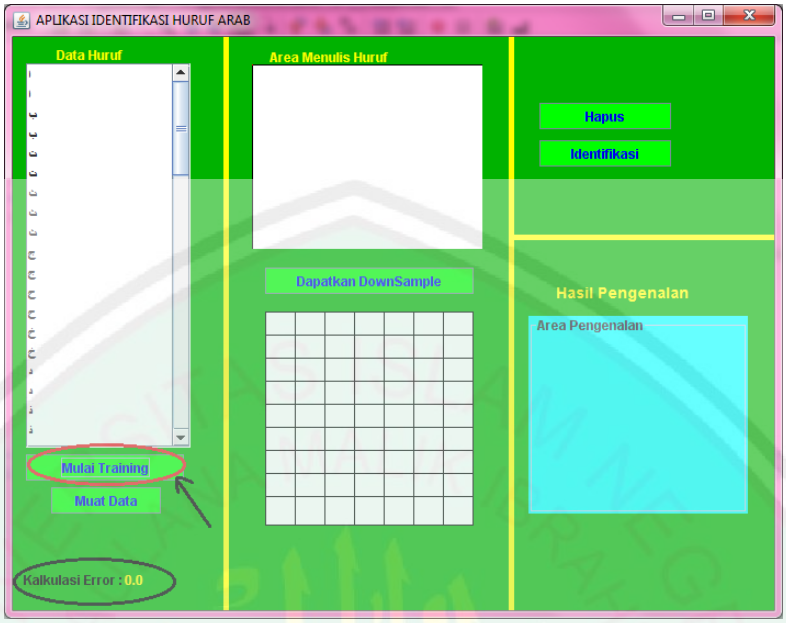


Gambar 4.1. Proses memuat data pada Aplikasi Identifikasi

Pada gambar terlihat sistem akan memuat data-data tulisan yang diambil dari basisdata *sample.dat* pada sistem. Pada Panel *scroolPane* akan muncul

karakter huruf-huruf Arab yang sebelumnya telah dibuat pada *sample.dat* dan memuatnya dalam Panel *scroolPane* sistem dalam bentuk list daftar huruf arab Alif sampai Ya'. Disini semua data input tulisan huruf Arab yang termuat pada aplikasi telah terlatih secara otomatis pada sistem. Dalam aplikasi ini data-data file pelatihan yang digunakan berisi *entry* huruf Arab yang terdiri dari 28 huruf Arab yang masing-masing huruf dibuat 2 pola inputan huruf, jadi file yang digunakan pada sistem berisi entry huruf arab berjumlah 56 pola huruf.

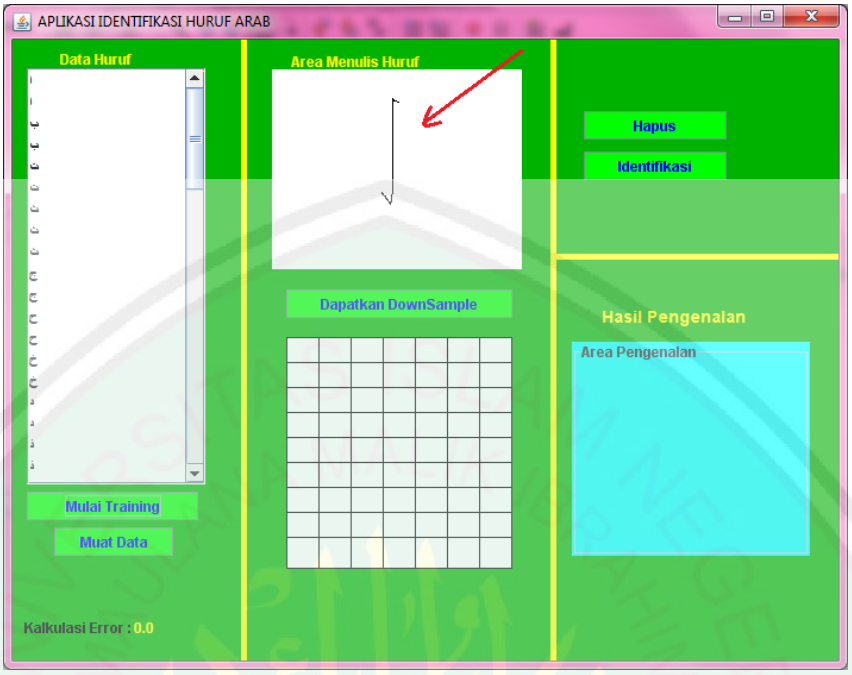
Setelah data-data set sample dimuat pada aplikasi yang dibangun, maka tahap selanjutnya adalah proses *Training* / pelatihan data-data set pola input sample menggunakan metode JST Kohonen. Proses ini dilakukan dengan mengklik tombol "***Training Data***" , maka sistem akan memulai proses pelatihan. Proses pelatihan dapat berlangsung dimana saja dari beberapa detik hingga beberapa menit tergantung pada performa kecepatan komputer. Setelah pelatihan selesai, maka akan ditampilkan bahwa proses pelatihan selesai yang tampilannya bisa dilihat pada Gambar 4.2.



Gambar 4.2. Proses training data pada Aplikasi Identifikasi

Sekarang sistem telah memiliki data-data set pelatihan yang telah dimuat dan telah dilatih oleh JST Kohonen. Jadi sistem telah mampu untuk melakukan proses pengenalan dan identifikasi karakter huruf arab.

Halaman *interface* aplikasi dibuat dengan mudah dalam penggunaannya. Pertama pengguna bisa menulis karakter huruf arab yang ingin dilakukan proses identifikasi pada *panel drawing* aplikasi yang bisa ditunjukkan pada Gambar 4.3.



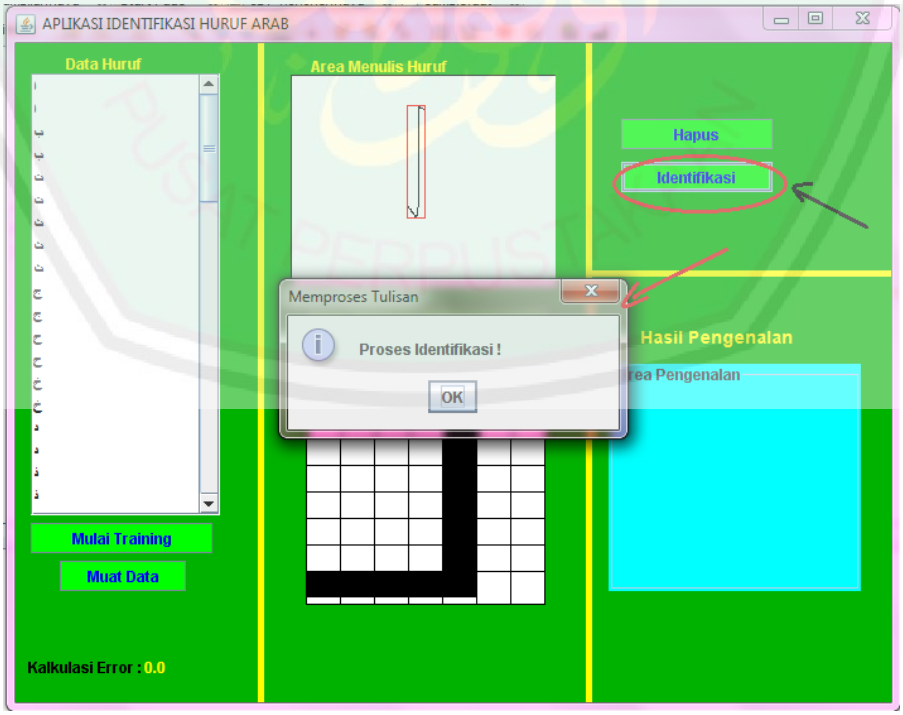
Gambar 4.3. Input tulisan huruf pada Aplikasi Identifikasi

Pada gambar terdapat *panel drawing* yang digunakan sebagai media untuk menginputkan tulisan. Terdapat juga dua buah tombol "***hapus***" dan "***identifikasi***". Tombol "*hapus*" berfungsi untuk menghapus / membersihkan tulisan di area *panel drawing* sedangkan tombol "*identifikasi*" untuk melakukan proses pengenalan karakter huruf arab.

Ketika pengguna telah selesai menuliskan huruf pada *panel drawing*, setelah itu pengguna mengklik tombol "*identifikasi*" untuk melakukan proses pengenalan. Ketika pengguna mengklik tombol "*identifikasi*", maka akan tampil *information message* yang menyatakan bahwa proses identifikasi tengah berjalan. Tidak hanya itu, secara otomatis sistem akan menampilkan *downsample* input tulisan dari pengguna dalam grid kotak *downsample* yang berupa pemetaan gambar huruf ke dalam kotak-kotak kecil dengan lebar *pixel* 7 dan tinggi 9. Disini juga akan ditampilkan proses *Image Procesing* yaitu *cropping*. *Cropping* dalam sistem ini

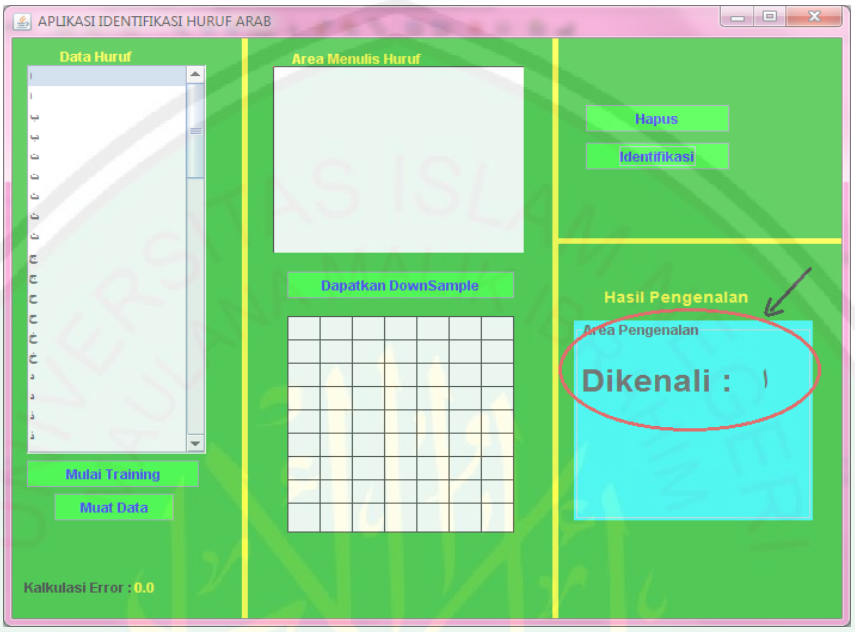
yaitu sistem hanya mengambil gambar di sekitar tulisan saja (hanya yang terdapat tulisan). Tujuan lain dari proses *cropping* ini yaitu untuk memotong setiap ruang kosong (putih bernilai 0) yang tidak terdapat tulisan dan juga mengatur posisi tulisan. Posisi tulisan yang dimaksud adalah pengguna bebas menulis di area mana saja dalam *panel drawing*, entah itu di sebelah atas, bawah, tengah, kanan atau kiri dan program ini akan masih tetap bisa mengenali tulisan dari pengguna.

Setelah itu jika ingin mengidentifikasi suatu huruf arab, maka pengguna harus mengklik tombol "*identifikasi*". Menulis huruf di *panel drawing* lalu mengklik tombol "*identifikasi*" , maka sistem akan melakukan proses pengenalan tulisan dan menampilkan *downsample* tulisan pengguna dan kemudian akan menghasilkan output karakter huruf menggunakan JST Kohonen. Prosesnya dapat dilihat pada Gambar 4.4.

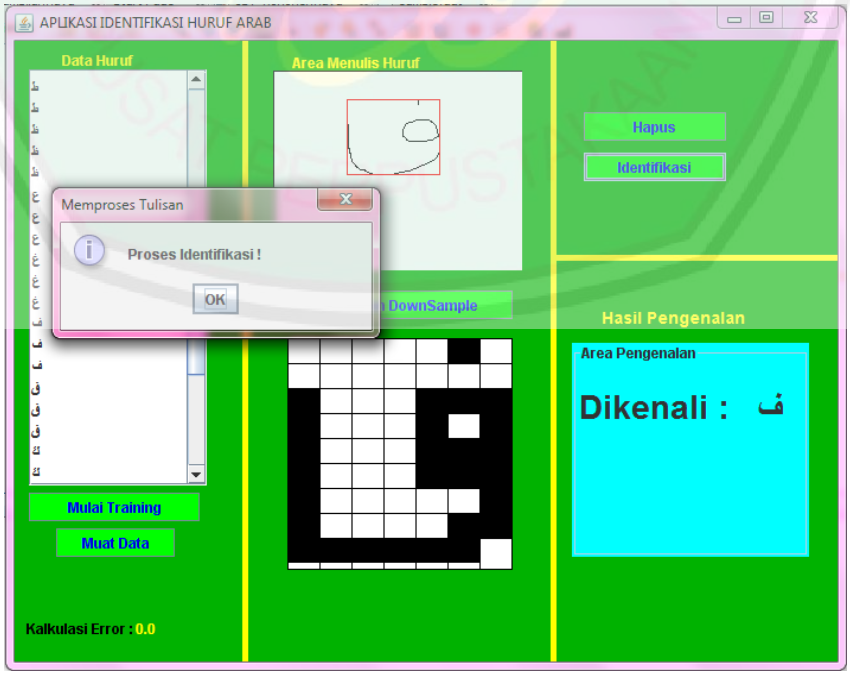


Gambar 4.4. Proses pengenalan tulisan pada Aplikasi

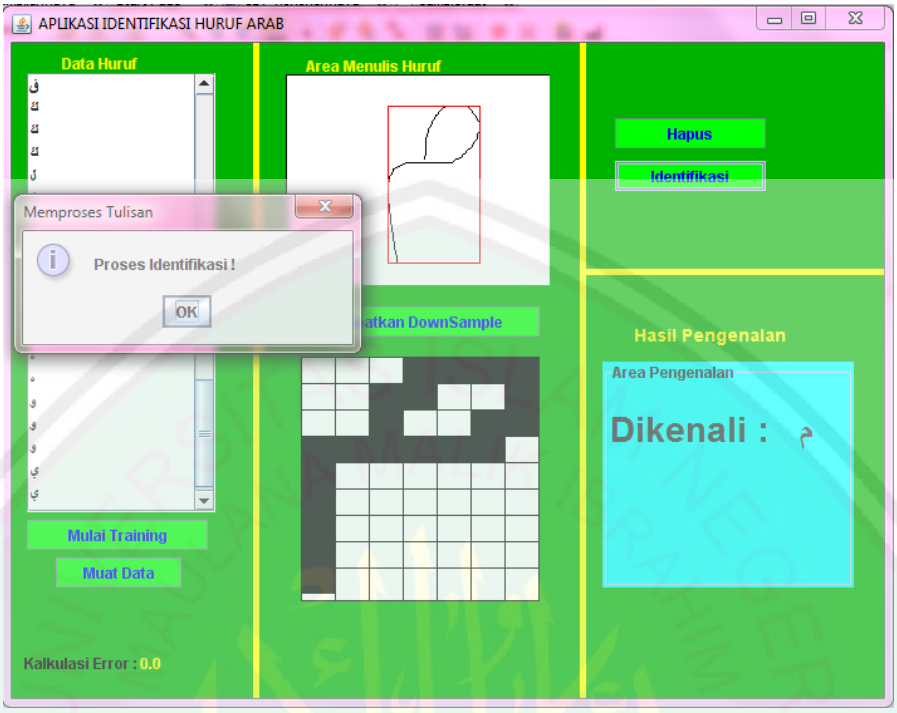
Setelah itu akan muncul ouput dari proses pengenalan dan identifikasi diatas pada *panel* Area pengenalan seperti yang bisa ditunjukkan pada Gambar 4.5.



Gambar 4.5. Output dari sistem pengenalan tulisan aplikasi



Gambar 4.6. Output identifikasi huruf Fa'

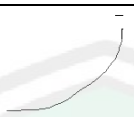
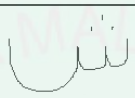
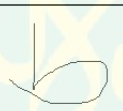


Gambar 4.7. Output identifikasi huruf Mim

Data *training* yang dibuat dalam sistem berjumlah 280 data pelatihan *(training)* inputan huruf Arab. Untuk kalkulasi dalam perhitungan error yang dapat diterima oleh sistem serta error yang sudah memenuhi kriteria error minimum untuk proses pengenalan dan identifikasi yaitu nilai error harus dibawah 10 %. Jika error sudah dibawah nilai 10 %, maka proses pengenalan tulisan tangan dapat dilakukan. Dan jika error masih bernilai diatas 10%, maka sistem akan mengulangi ke tahap *training* JST dalam sistem untuk memproses data agar error bernilai minimum (dibawah 10%).

Tabel 1.3. Tabel uji coba data *training* input huruf yang menunjukkan nilai *error*. Untuk uji coba data training input huruf yang menunjukkan nilai *error* inputan tulisan huruf arab data selengkapnya dilampirkan pada halaman lampiran.

No	Inputan Huruf Arab	Uji coba input huruf arab	Tingkat Kesalahan (*error*) saat training
1	ا		0,093433
2	ب		0,087543
3	ث		0,089328
4	ث		0,089459
5	ج		0,089666
6	ح		0,090332
7	خ		0,090999
8	د		0,086339
9	ذ		0,085907

10	ر		0,088553
11	ز		0,086551
12	س		0,085449
13	ش		0,086621
14	ص		0,094311
15	ض		0,090202
16	ط		0,095118
17	ظ		0,094559
18	ع		0,098885
19	غ		0,095330
20	ف		0,089443

21	ق		0,095158
22	ك		0,083663
23	ل		0,088251
24	م		0,086638
25	ن		0,088510
26	و		0,085390
27	ه		0,088371
28	ي		0,087490

Untuk pengujian data uji / *testing* input huruf yang menunjukkan tingkat *error* serta ketepatan inputan tulisan huruf arab dilampirkan pada halaman lampiran.

Menentukan prosentase keberhasilan data testing ketepatan pengenalan :

$$= \frac{\text{Jumlah huruf Arab yang dikenali}}{\text{Jumlah data uji coba}} \times 100\ \%$$

Pada uji coba data testing akan dibuat data uji sebanyak 140 uji coba. Masing-masing 5 uji coba dari tiap inputan 1 huruf Arab. Dan dari uji coba data testing diperoleh ketepatan pengenalan huruf Arab sebanyak 119 uji coba input huruf Arab dapat dikenali dan sisanya tidak dapat dikenali. Berdasarkan hasil uji coba testing input huruf Arab akan dihitung prosentase keberhasilan pengenalan

Menentukan prosentase keberhasilan data testing ketepatan pengenalan :

$$= \frac{\text{Jumlah huruf Arab yang dikenali}}{\text{Jumlah data uji coba}} \times 100\ \%$$

$$= \frac{119}{140} \times 100\ \%$$

$$= 85\ \%$$

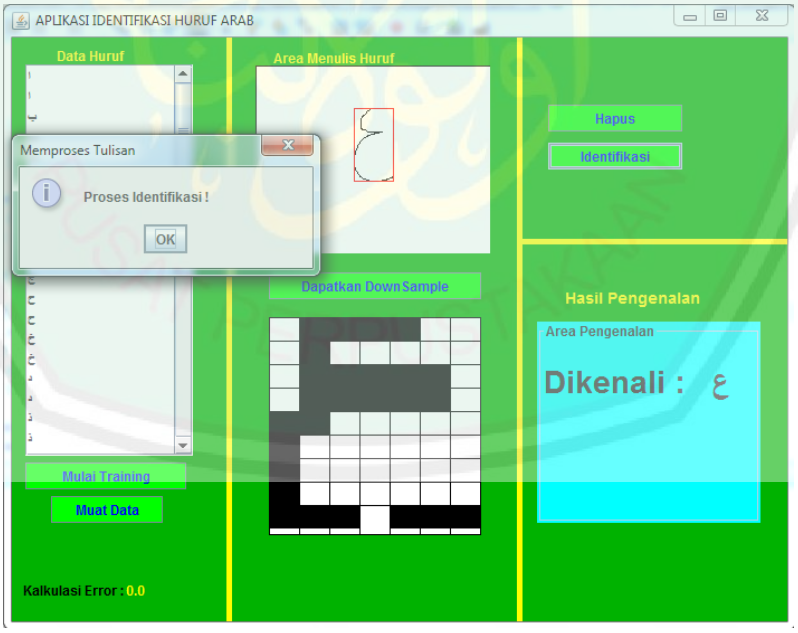
4.5 Analisa Hasil dan Pembahasan

Berdasarkan uji coba yang telah dilakukan, diketahui bahwa aplikasi identifikasi huruf Arab yang telah dibangun mampu mengenali input tulisan tangan untuk setiap input tulisan tangan *real-time* yang dituliskan oleh pengguna. Ketepatan dalam pengenalan aplikasi identifikasi huruf Arab dengan menerapkan metode JST *Kohonen* pada keberhasilan pemrosesan pengenalan mencapai 85%.

Hasil analisa terhadap data uji coba diketahui bahwa ketepatan pengenalan tulisan tangan dipengaruhi oleh tingkat kemiripan inputan dari pengguna dengan data rujukan *sample* yang telah dibuat oleh sistem yaitu dalam bentuk pola inputan huruf Arab Alif sampai Ya' yang dibuat menggunakan pemetaan pada grid-grid *pixel* 7x9. Jika inputan huruf yang diinputkan pengguna merujuk dari

sample dan pengguna melakukan pemrosesan tulisan semirip mungkin dengan *sample*, maka sistem akan dengan tepat mengenali inputan tulisan tangan *real-time* dari pengguna. Dan jika pengguna akan melakukan pemrosesan identifikasi tulisan huruf Arab dan tidak merujuk pada sample yang telah dibuat oleh sistem, maka kemungkinan besar sistem tidak akan bisa mengenali input tulisan yang dimaksud dari pengguna.

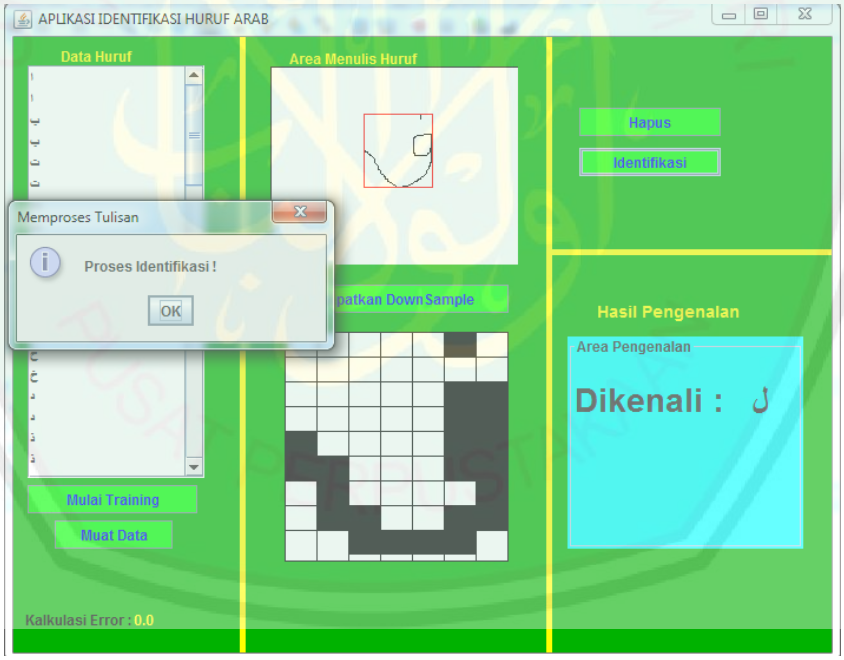
Gambar 4.8. merupakan contoh hasil pengenalan pola huruf ‘Ain. Pengguna menuliskan bentuk huruf ‘Ain yang merujuk pada *sample* dan pengguna juga menuliskannya semirip mungkin dengan *sample* yang telah dibuat oleh sistem. Jadi, sistem akan dengan tepat mengenali huruf Arab ‘Ain yang diinputkan yang dimaksud oleh pengguna.



Gambar 4.8. Output identifikasi huruf ‘Ain dapat dikenali oleh Sistem.

Jika pengguna yang akan melakukan pemrosesan identifikasi tulisan huruf Arab dan tidak merujuk pada sample yang telah dibuat oleh sistem, maka

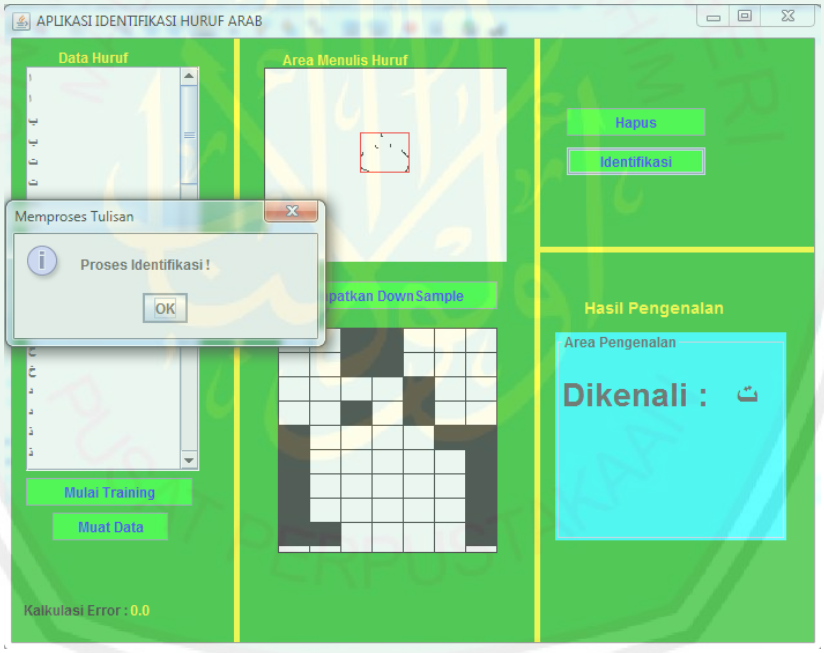
kemungkinan besar sistem tidak akan bisa mengenali input tulisan yang dimaksud dari pengguna. Ada beberapa faktor yang mempengaruhi kegagalan tersebut. Pertama karena pengguna tidak merujuk pada sample, artinya pengguna melakukan identifikasi dengan menuliskan huruf secara sembarangan atau sesuai kehendak dan tidak merujuk pada sample yang telah dibuat oleh system. Jadi, system tidak akan mengenali input tulisan dari pengguna, seperti yang ditunjukkan pada Gambar 4.9. yaitu inputan huruf Arab Fa' oleh pengguna, dan hasil output tidak sesuai dengan inputan yaitu output menghasilkan huruf Lam.



Gambar 4.9. Output identifikasi huruf Fa' tidak dapat dikenali oleh Sistem karena tidak merujuk pada sample system (menulis secara sembarangan)

Faktor kedua yang mendasari kegagalan proses pengenalan tulisan selain tidak merujuk pada *sample* yang telah dibuat oleh sistem adalah faktor kemiripan antara bentuk huruf Arab satu dengan bentuk huruf Arab lainnya, walaupun

pengguna tersebut sudah merujuk pada sample yang telah dibuat oleh sistem. Misalnya contoh yang dilihatkan pada Gambar 4.10. yaitu inputan huruf Arab Tsa' oleh pengguna tidak sesuai dengan hasil output system yaitu hasil output adalah huruf Arab Ta'. Hal tersebut karena terdapat kemiripan antara pola huruf Tsa' dengan huruf Ta' yaitu berbentuk garis lengkung yang menghadap keatas dan pada sample yang telah dibuat oleh system, pola juga dibuat membentuk garis lengkung yang menghadap keatas, perbedaannya hanya terdapat pada titik yaitu titik pada huruf Tsa' berjumlah 3 buah dan pada huruf Ta' berjumlah 2 buah.



Gambar 4.10. Output identifikasi huruf Tsa' tidak dapat dikenali oleh Sistem karena terdapat kemiripan antara bentuk huruf Tsa' dengan huruf Ta'

4.6 Integrasi dan Kajian Islam

Sumber tertinggi ajaran agama Islam adalah Al-Qur'an yang juga merupakan kitab suci agama Islam. Dalam Al-qur'an berisi segala petunjuk tentang ilmu, hukum, akhlak, perilaku yang diterapkan pada kebiasaan manusia

sehari-harinya. Jadi, wajib hukumnya bagi manusia yang berakal untuk mengerti dan memahami isi dan makna dari kalimat-kalimat dalam Al-qur'an.

Kita sebagai manusia yang berakal wajib hukumnya untuk bisa membaca dan menulis serta belajar. Belajar adalah proses dimana seseorang ingin tahu akan suatu hal baru dan menirunya. Dalam konteks belajar yang baik dan benar pasti ada suatu pedoman yang berfungsi sebagai tuntunan seseorang dalam proses belajar, yang tidak lain adalah Al-qur-an. Wahyu pertama yang turun kepada junjungan kita Rasullullah SAW adalah ajuran untuk membaca melalui sebuah Kallam. Kallamullah Firman Allah SWT yang memuat hal tersebut terdapat dalam surat Al-Alaq ayat : 4-5

الَّذِي عَلَّمَ بِالْقَلَمِ ۞ عَلَّمَ الْإِنْسَانَ مَا لَمْ يَعْلَمْ ۞

Artinya : "Yang mengajarkan (manusia) dengan perantara Kallam. Dia mengajarkan kepada manusia apa yang tidak atau belum diketahuinya." (Qs.Al-Alaq): 4-5).

Ayat diatas mempunyai makna bahwa manusia melakukan proses belajar-mengajar dan mempelajari suatu hal dengan melalui suatu perantara yaitu Kallam. Kallam disini bisa berupa ucapan, kata, kalimat, teks tertulis, teks tidak tertulis, dll. Dan tujuan utama dari proses belajar-mengajar adalah untuk mengetahui suatu hal baru yang tidak atau belum diketahui oleh manusia tersebut agar menjadi tahu dan bisa menirunya sesuia dengan ajaran Al-Qur-an. Tidak hanya itu, ayat tersebut juga bermakna suatu penyampaian informasi yang berisi pesan-pesan pendidikan kepada manusia yang bersaranakan media baik itu media tulis, baca atau media teknologi.

Dalam proses belajar-mengajar yang bersumber dari Al-Qur-an, tentu saja bahasa dan tulisan yang digunakan adalah bukan menggunakan huruf alphabet, tapi menggunakan bahasa arab yang menggunakan tulisan huruf Hijaiyah. Huruf Arab atau huruf Hijaiyah berjumlah 28 huruf yang diawali dengan huruf Alif dan diakhiri dengan huruf Ya'. Dalam Al-qur-an semua isi tulisan menggunakan huruf arab tersebut, jadi wajib bagi kita umat Islam untuk mengerti dan memahami masing-masing komponen tulisan huruf Arab yang menyusun kata dan kalimat arab dalam Al-Qur-an. Semua itu wajib dipelajari bagi umat Islam karena dalam Al-qur-an berisi tentang ajaran perilaku, hukum, dan tata cara berkehidupan di dunia, sesuai dengan firman Allah SWT dalam surat Yusuf ayat : 2

إِنَّآ أَنزَلْنَٰهُ قُرْءَٰنًا عَرَبِيًّا لَّعَلَّكُمْ تَعْقِلُونَ ۝٢

Artinya: "Sesungguhnya Kami menurunkannya berupa Al Quran dengan berbahasa Arab, agar kamu memahaminya". (Q.S. Yusuf [12]:2)

Ayat diatas mempunyai makna bahwa penting dan wajibnya bagi umat Islam dalam belajar dan mempelajari huruf Arab agar bisa mengerti bahasa Arab yang baik dan benar, baik itu dalam proses pembacaan, pelafalan dan penghafalan. Bahasa Arab juga wajib dipelajari oleh manusia karena bahasa Arab adalah bahasa yang paling jelas, fasih, luas maknanya serta lebih mengena dan cocok untuk jiwa manusia.

Allah SWT memberikan informasi tentang ilmu pengetahuan yang bersifat umum di dalam Al-qur'an bukan ilmu pengetahuan yang bersifat teknis. Di ayat yang lain Allah SWT memerintahkan umat islam menjadi insan ulul albab. Insan

ulul albab secara umum dan jelas dijelaskan oleh firman Allah SWT dalam surat Ali Imran ayat : 190-191.

ٱلَّذِينَ يَذۡكُرُونَ ٱللَّهَ قِيَٰمٗا وَقُعُودٗا وَعَلَىٰ جُنُوبِهِمۡ وَيَتَفَكَّرُونَ فِي خَلۡقِ ٱلسَّمَٰوَٰتِ وَٱلۡأَرۡضِ رَبَّنَا مَا خَلَقۡتَ هَٰذَا بَٰطِلٗا سُبۡحَٰنَكَ فَقِنَا عَذَابَ ٱلنَّارِ ١٩١

Artinya: (yaitu) orang-orang yang mengingat Allah sambil berdiri atau duduk atau dalam keadan berbaring dan mereka memikirkan tentang penciptaan langit dan bumi (seraya berkata): "Ya Tuhan Kami, Tiadalah Engkau menciptakan ini dengan sia-sia, Maha suci Engkau, Maka peliharalah Kami dari siksa neraka. (Qs.Al-Imran):190-191).

Insan ulul albab adalah insan yang berakal dan senatiasa berpikir apa-apa yang diciptakan oleh Allah SWT. Karena seorang insan ulul albab tahu benar bahwasanya tidak ada sesuatu yang diciptakan Allah SWT secara sia-sia. Insan ulul albab menjadikan Al-qur'an sebagai sumber inspirasi untuk mengeksplorasi Al-qur'an.

Selain sebagai sumber inspirasi, diturunkanya Al-Qur'an juga bertujuan untuk memberi gambaran dan petunjuk dan sebagai pembeda antara yang haq dan batil.

BAB V

KESIMPULAN DAN SARAN

5.1 KESIMPULAN

Dari hasil penelitian dan implementasi uji coba aplikasi yang dilakukan oleh peneliti dapat disimpulkan beberapa hal-hal terkait penelitian yang telah dilakukan yaitu telah dibuat aplikasi identifikasi tulisan huruf arab menggunakan jaringan syaraf tiruan *Kohonen* dengan pemrosesan awal yaitu ekstraksi fitur dan downsample yang dapat memberikan hasil pengenalan karakter tulisan. Dalam metode Jaringan Syaraf Tiruan *Kohonen* terdapat proses training yang merupakan proses penting dalam alur sistem yang dibangun. Pada proses training, sistem akan terus melatih data inputan yang diolah sampai nilai *error* (tingkat kesalahan) minimum. Artinya, metode akan menghasilkan hasil akhir nilai output karena nilai *error* akan dihitung sampai *error* bernilai minimum (rendah/kecil). Jadi, sistem akan memberikan hasil pengenalan tulisan huruf arab.

Hasil uji coba sistem dalam aplikasi identifikasi tulisan arab ini akan menghasilkan hasil akhir berupa huruf arab. Pada sistem, penggunaan harus mengacu dan merujuk pada sample pola huruf arab yang telah dibuat oleh sistem yang disimpan dalam sample pola. Dan pada uji coba huruf arab yang dicoba dapat menghasilkan prosentase keberhasilan pengenalan yaitu 85 %.

5.2 SARAN

Beberapa saran untuk penelitian dan pengembangan aplikasi lebih lanjut agar lebih menarik adalah :

a. Dapat mengenali tulisan arab dengan menggunakan harakat seperti fatkhah, kasroh, dhummah, dan lain sebagainya. Dan juga bisa mengenali bukan hanya satu huruf saja, tetapi bisa lebih dari satu huruf yang dengan kata lain bisa mengenali sebuah kata bahasa arab.
b. Pengembangan lebih lanjut aplikasi ini bisa dengan menggunakan jenis transformasi citra yang berbeda-beda, karena dalam aplikasi ini masih menggunakan ektraksi fitur secara umum, bisa menggunakan semisal transformasi wavelet atau jenis lainya.
c. Pada system yang dibangun, hanya pembuat pola (pemilik) yang bisa melakukan pengenalan dan identifikasi, karena system ini dirancang dengan membuat pola huruf yang disimpan dalam data sample. Jika pengguna lain ingin bisa menggunakan system ini, maka pengguna lain harus membuat pola sample sendiri sesuai dengan karakter yang ingin dikenali dan diidentifikasi. Jika pengguna lain menggunakan aplikasi ini dengan menggunakan data sample pemilik, maka akan dipastikan hasil akhir akan buruk dalam proses pengenalan. Maka dari itu pengembangan lebih lanjut aplikasi ini bisa mengenali tulisan dari banyak pengguna tentunya dengan hasil yang baik.

DAFTAR PUSTAKA

Alqur'an dan Terjemahanya. 1971. Jakarta: Yayasan Penyelenggara Penterjemah/Pentafsir Al-Qur'an. Depag RI, 2011.

AlKhateeb, J. H., Ren, J., Jiang, J., & Al-Muhtaseb, H., 2011. *Offline handwritten Arabic cursive text recognition using Hidden Markov Models and re-ranking*. Pattern Recognition Letters, 32(8), 1081–1088. doi:10.1016/j.patrec.2011.02.006

Anif, M., Juanita, S. & Afriyani, I.D., 2013. *Pengembangan Aplikasi Text Recognition dengan Klasifikasi Neural Network Pada Huruf Hijaiyah Gundul*. BIT, 10(1), pp.59-67.

Aouadi, N., Amiri, S., & Echi, a. K. 2013. *Segmentation of Connected Components in Arabic Handwritten Documents*. Procedia Technology, 10, 738–746. doi:10.1016/j.protcy.2013.12.417

Ardhianto, E., Munawaroh, S. & Prihandono, A. 2011. Pengolahan citra Digital untuk identifikasi Ciri sidik jari berbasis Minutiae. Jurnal Teknologi Informatika DINAMIK.

Cheriet, Mohamed, Kharma Nannaf, Liu, Cheng-Lin and Ching Y. Suen. (2007). *Character Recognition System A Guide for Student And Practioners*, John Willey and Sons, Inc, New Jersey.

Darma Putra. (2010). *Pengolahan Citra Digital*, Andi, Yogyakarta.

Duda, R.O., Hart, P.E. & Stork, D.G., 2001. *Pattern Classification. 2nd ed*. New York: Wiley.

Dwi Putra, I.K.A. & Prapitasari, L.P.A., 2011. *Segmentasi Karakter Pada Skrip Bahasa Bali Menggunakan Metode Canny Edge Detection*. In Konferensi Nasional Sistem dan Informatika. Bali, 2011.

Heaton, J., "*Introduction to Neural Network with Java"*, page2.html, 2003

Hilyati Safitri, Fitri Damayanti, Kurniawan Eka Permana, *Pengenalan Tulisan Tangan Huruf Alfabet menggunakan metode Modified Direction Feature (MDF) dan Learning Quantization Feature (LVQ).* Jurnal Masyarakat Informatika Universitas Trunojoyo, 2010.

Khorsheed, M. S. (2007). *Offline recognition of omnifont Arabic text using the HMM ToolKit (HTK).* Pattern Recognition Letters, 28(12), 1563–1571. doi:10.1016/j.patrec.2007.03.014

Kusumadewi, Sri. "*Membangun Jaringan Syaraf Tiruan Menggunakan MATLAB & EXCEL LINK*", Graha Ilmu, Yogyakarta, 2004

Mohammed Lutf, Xinge You, YM Cheung, and C.L. Philip Chen, "*Arabic Font Recognition based on Diacritics Features*, " Pattern Recognition, Vol. 47, No. 2, pp. 672-684, 2014, Elsevier

Nugraha, A.W.W. & Widhiatmoko, 2012. *Jumlah Transisi pada Ciri Transisi dalam Pengenalan Pola Tulisan Tangan Aksara Jawa Nglegeno dengan Multiclass Support Vector Machines*. Dinamika Rekayasa, 8(1), pp.18-24.

Nazla Nurmila, Aris, Sugiharto, Eko Adi Sarwoko, "*Algoritma Backpropagation Neural Network untuk Pengenalan pola karakter Huruf Jawa",* Jurnal Masyarakat Informatika, F. MIPA UNDIP, Volume 1, Nomor 1, ISSN 2086-4930.

P. Nugraha dan A. B. Mutiara, *"Metode Ekstraksi Data untuk Pengenalan Huruf dan angka Tulisan Tangan dengan Menggunakan Jaringan Syaraf Buatan Propagasi Balik"*,

http://www.gunadarma.ac.id, 2004. (diakses pada tgl 22 Februari 2014 jam 15.56 WIB).

Prabowo, Anindito, "*Perbandingan antara Metode Kohonen Neural Network dengan Metode Learning Vector Quantization pada Pengenalan Pola Tandatangan",* TugasAkhir- Perpustakaan F. MIPA UNDIP, Semarang, 2007.

Puspitaningrum, D. 2006. Pengantar Jaringan Syaraf Tiruan. Yogyakarta: CV. Andi Offset.

R. Plamondon and S. N. Srihari, "*On-line and off- line handwritten character recognition: A comprehensive survey*,"IEEE. Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 63-84, 2000.

Schalkoff, R.J., 1991. *Pattern Recognition: Statistical, Structural and Neural Approaches*. New York: Wiley.

Sheng Fuu Lin, Yu Wei Chang And Chien-Kun Su, "*A Study on Chinese Carbon-Signature-Recognition*", http://www.iis.sinica.edu.tw/JISE/2002/200414_06.html,2002. Diakses tanggal 10 Oktober 2014.

Siang, Jong Jek, *"Jaringan Syaraf Tiruan dan Pemrogramannya menggunakan Matlab"*, Andi, Yogyakarta, 2005.

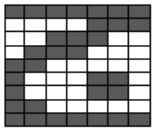
Supriana, I., & Nasution, A. 2013. *Arabic Character Recognition System Development. Procedia Technology*, 11(Iceei), 334–341. doi:10.1016/j.protcy.2013.12.199

Wirayuda, T.A.B., Wardhani, M.L.D.K., Adiwijaya. *Pengenalan Huruf Jepang (Kana) menggunakan Direction Feature Extraction dan Learning Vector Machine (LVQ).* (TA) – Bandung: ITT. 2008

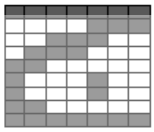
Lampiran 1

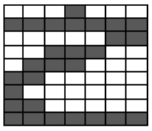
Ciri citra pola tulisan tangan berdasarkan grid pixel 7x9

Lampiran 1. Menunjukkan rujukan sample pola inputan huruf Arab Alif sampai Ya' yang dibuat menggunakan pemetaan pada grid-grid *pixel* 7x9

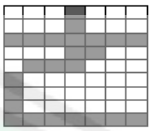


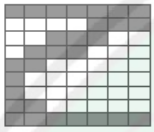
. Jim ج



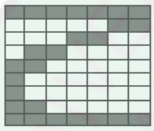


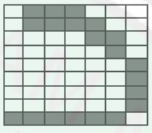
. Kho خ



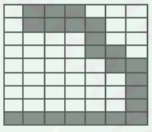


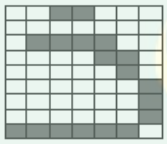
. Kha ح



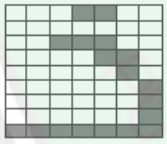


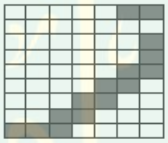
. Dal د



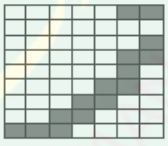


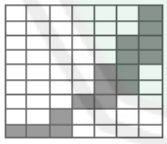
. Dzal ذ



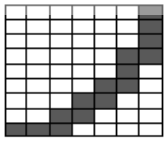


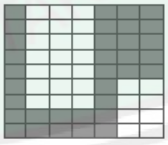
. Za ز



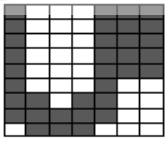


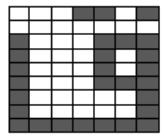
. Ra ر



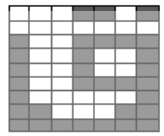


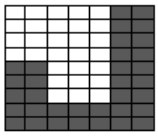
. Sin س



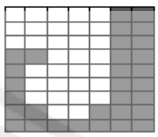


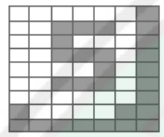
Qof ق



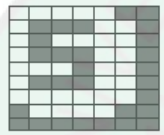


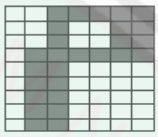
Lam ل



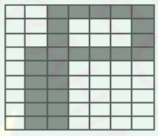


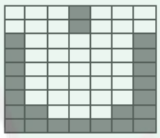
Kaf ك



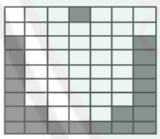


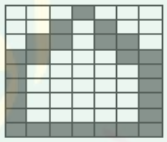
Mim م



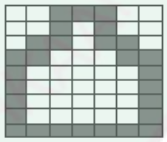


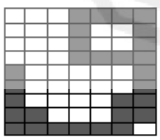
Nun ن



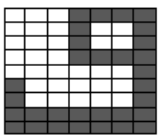


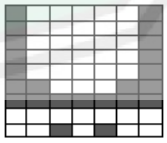
Ha ه



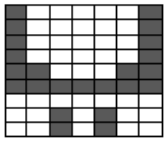


Wawu و





Ya بب

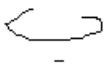


Lampiran 2

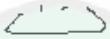
UJI COBA DATA INPUT TULISAN HURUF ARAB

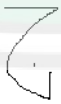
Lampiran 2. Uji coba data training input huruf Arab yang menunjukkan nilai *error* pada proses pelatihan (*training*) JST Kohonen inputan tulisan huruf arab. Data *training* yang dibuat dalam sistem berjumlah 280 data inputan huruf Arab.

No	Inputan Huruf Arab	Uji coba input huruf arab	Tingkat Kesalahan (*error*) saat training
1			0,093433
2			0,092847
3			0,093344
4			0,091211
5			0,093321
6			0,092433

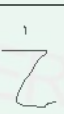
7			0,092144
8			0,091452
9			0,093441
10			0,090768
11	ب		0,087543
12			0,090045
13			0,090002
14			0,090227
15			0,091211
16			0,091887

17			0,091009
18			0,092113
19			0,081221
20			0,090231
21	ت		0,089328
22			0,085449
23			0,090991
24			0,093218
25			0,093219
26			0,089776

27			0,086781
28			0,088679
29			0,091338
30			0,096525
31	ث		0,089459
32			0,087449
33			0,090113
34			0,093662
35			0,091774
36			0,099833

37			0,083221
38			0,084477
39			0,083568
40			0,086119
41	ج		0,089666
42			0,091771
43			0,089221
44			0.091223
45			0,090881
46			0,081882

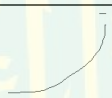
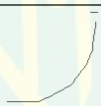
47			0,087228
48			0,093338
49			0,083982
50			0,081239
51	ح		0,090332
52			0,080907
53			0,087332
54			0,084489
55			0,081314
56			0,093424

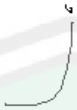
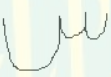
57			0,089776
58			0,085982
59			0,099432
60			0,087433
61	خ		0,090999
62			0,081009
63			0,090990
64			0,084329
65			0,083249
66			0,082117

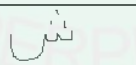
67			0,082342
68			0,088376
69			0,094239
70			0,087631
71	ا		0,086339
72			0,083776
73			0,083324
74			0,087397
75			0,074029
76			0,080832

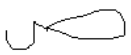
77			0,091930
78			0,093465
79			0,081849
80			0,083239
81	نا		0,085907
82			0,090992
83			0,092347
84			0,089732
85			0,094889
86			0,089439

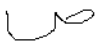
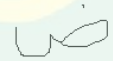
87			0,087337
88			0,081328
89			0,085003
90			0,084327
91			0,088553
92			0,086887
93			0,091343
94			0,083111
95			0,082229
96			0,089404

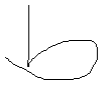
97			0,085522
98			0,080433
99			0,087388
100			0,087002
101	ز		0,086551
102			0,082221
103			0,088712
104			0,083113
105			0,090393
106			0,093300
107			0,087033

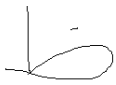
108			0,079723
109			0,072118
110			0,089665
111	س		0,085449
112			0,084667
113			0,083114
114			0,084288
115			0,085656
116			0,086711
117			0,086236

118			0,0862517
119			0,081311
120			0,085213
121	ش		0,086621
122			0,090443
123			0,080883
124			0,084242
125			0,089732
126			00,87231
127			0,088324
128			0,089931

129			0,088762
130			0,085266
131	ص		0,094311
132			0,088002
133			0,087331
134			0,079883
135			0,087422
136			0,085552
137			0,089765
138			0,090139
139			0,071388

140			0,095884
141	ظ		0,090202
142			0,094112
143			0,098233
144			0,072388
145			0,089722
146			0,090891
147			0,091388
148			0,083422
149			0,094239
150			0,083299

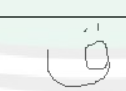
151	ط		0,095118
152			0,085220
153			0,091344
154			0,083442
155			0,096556
156			0,095434
157			0,096343
158			0,094299
159			0,097143
160			0,083490

161	ظ		0,094559
162			0,080371
163			0,090133
164			0,094244
165			0,089242
166			0,088299
167			0,081939
168			0,093887
169			0,082493
170			0,089243

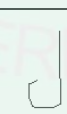
171	ع		0,098885
172			0,098821
173			0,083773
174			0,082890
175			0,095832
176			0,089284
177			0,089822
178			0,095424
179			0,082335
180			0,082565

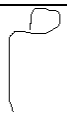
181	غ		0,095330
182			0,094772
183			0,087232
184			0,090234
185			0,082998
186			0,082348
187			0,090138
188			0,082138
189			0,088211
190			0,089227

191	ف		0,089443
192			0,096223
193			0,083422
194			0,082477
195			0,097132
196			0,089243
197			0,081324
198			0,094248
199			0,095143
200			0,093288

201	ق		0,095158
202			0,093351
203			0,082133
204			0,087138
205			0,091319
206			0,087390
207			0,082377
208			0,083556
209			0,082776
210			0,081877

211	ك		0,083663
212			0,083875
213			0,083211
214			0,088331
215			0,083771
216			0,085665
217			0,086352
218			0,085144
219			0,085242
220			0,090153

221	ل		0,088251
222			0,087442
223			0,083472
224			0,080682
225			0,090653
226			0,089261
227			0,083627
228			0,090527
229			0,084445
230			0,086153

231	م		0,086638
232			0,087404
233			0,082617
234			0,082678
235			0,091388
236			0,090637
237			0,086289
238			0,087266
239			0,085662

240			0,085246
241	ن		0,088510
242			0,087508
243			0,090278
244			0,089276
245			0,089268
246			0,086278
247			0,087257
248			0,087263

249			0,087259
250			0,089299
251	و		0,085390
252			0,084944
253			0,090747
254			0,093544
255			0,089234
256			0,090443
257			0,090678
258			0,087635
259			0,083678

260			0,087368
261	ه		0,088371
262			0,087973
263			0,087837
264			0,088727
265			0,088222
266			0,087899
267			0,087337
268			0,083898
269			0,087376
270			0,087688

271	ي		0,087490
272			0,086820
273			0,082737
274			0,086356
275			0,089093
276			0,087463
277			0,089766
278			0,086491
279			0,083337
280			0,085766

Lampiran 2. Tabel uji coba testing input huruf yang menunjukkan tingkat *error* serta ketepatan pengenalan inputan tulisan huruf arab. Data pengujian (*testing)* yang dibuat dalam sistem berjumlah 140 data inputan huruf Arab.

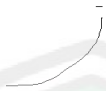
No	Inputan Huruf Arab	Uji coba input huruf arab	Tingkat Kesalahan (*error*)	Ketepatan (Tepat/ Tidak)
1	ا		0,093433	Tepat
2			0,092847	Tepat
3			0,093344	Tepat
4			0,091211	Tepat
5			0,093321	Tepat
6	ب		0,087543	Tepat
7			0,090045	Tidak Tepat (Dibaca : ي)
8			0,090002	Tepat

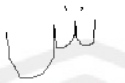
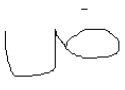
9			0,090227	Tepat
10			0,090231	Tepat
11	ت		0,089328	Tepat
12			0,085449	Tepat
13			0,088679	Tepat
14			0,091338	Tepat
15			0,096525	Tidak Tepat (dibaca : ث)
16	ث		0,089459	Tepat
17			0,099833	Tidak Tepat (Dibaca : ت)
18			0,083221	Tepat

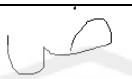
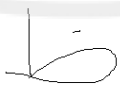
19			0,083568	Tepat
20			0,086119	Tepat
21	ج		0,089666	Tepat
22			0,091771	Tidak Tepat (Dibaca : ح)
23			0,089221	Tepat
24			0,093338	Tidak Tepat (Dibaca : ع)
25			0,083982	Tepat
26	ح		0,090332	Tepat
27			0,080907	Tepat
28			0,089776	Tepat
29			0,085982	Tepat

30			0.087433	Tepat
31	خ		0,090999	Tidak Tepat (Dibaca : ح)
32			0,081009	Tepat
33			0,084329	Tepat
34			0,083249	Tepat
35			0,082117	Tepat
36	ر		0,086339	Tepat
37			0,083776	Tepat
38			0,091930	Tidak Tepat (Dibaca : ل)
39			0,081849	Tepat

40			0,083239	Tepat
41	ذ		0,085907	Tepat
42			0,090992	Tidak Tepat (Dibaca : د)
43			0,092347	Tidak Tepat (Dibaca : ل)
44			0,087337	Tepat
45			0,084327	Tepat
46	ر		0,088553	Tepat
47			0,086887	Tepat
48			0,091343	Tepat
49			0,080433	Tidak Tepat (Dibaca : د)

50			0,087388	Tepat
51	ز		0,086551	Tepat
52			0,082221	Tidak Tepat (Dibaca : ذ)
53			0,088712	Tepat
54			0,072118	Tidak Tepat (Dibaca : ر)
55			0,089665	Tepat
56	س		0,085449	Tepat
57			0,086236	Tepat
58			0,0862517	Tepat
59			0,081311	Tepat
60			0,085213	Tepat

61	ش		0,086621	Tepat
62			0,090443	Tidak Tepat (Dibaca : س)
63			0,080883	Tepat
64			0,084242	Tepat
65			0,089732	Tepat
66	ص		0,094311	Tepat
67			0,088002	Tepat
68			0,087331	Tepat
69			0,079883	Tepat
70			0,095884	Tidak Tepat (Dibaca : ض)
71	ظ		0,090202	Tidak Tepat (Dibaca: ص)

72			0,094112	Tepat
73			0,072388	Tidak Tepat (Dibaca: ص)
74			0,089722	Tepat
75			0,090891	Tepat
76	ط		0,095118	Tepat
77			0,085220	Tepat
78			0,094299	Tepat
79			0,097143	Tepat
80			0,083490	Tepat
81	ظ		0,094559	Tepat
82			0,080371	Tidak Tepat (Dibaca : ط)

83			0,090133	Tepat
84			0,094244	Tepat
85			0,089242	Tepat
86	ع		0,098885	Tepat
87			0,098821	Tepat
88			0,083773	Tepat
89			0,082890	Tepat
90			0,082565	Tepat
91			0,082348	Tepat
92			0,090138	Tepat

93			0,082138	Tidak Tepat (Dibaca : ع)
94			0,088211	Tepat
95			0,089227	Tepat
96	ف		0,089443	Tepat
97			0,096223	Tepat
98			0.083422	Tepat
99			0,082477	Tepat
100			0,097132	Tidak Tepat (Dibaca : و)
101	ق		0,095158	Tepat
102			0,093351	Tidak Tepat (Dibaca : ف)

103			0,082133	Tepat
104			0,087138	Tepat
105			0,091319	Tepat
106	ك		0,083663	Tepat
107			0,083875	Tepat
108			0,083211	Tepat
109			0,088331	Tepat
110			0,083771	Tepat
111	ل		0,088251	Tepat
112			0,087442	Tepat

113			0,083472	Tepat
114			0,080682	Tepat
115			0,086153	Tepat
116	م		0,086638	Tepat
117			0,087404	Tepat
118			0,082617	Tepat
119			0,082678	Tepat
120			0,091388	Tepat
121	ن		0,088510	Tepat
122			0,087508	Tepat

123			0,090278	Tepat
124			0,089276	Tepat
125			0,089268	Tepat
126	و		0,085390	Tepat
127			0,084944	Tepat
128			0,090747	Tepat
129			0,093544	Tidak Tepat Dibaca : ف)
130			0,089234	Tepat
131	ه		0,088371	Tepat
132			0,087973	Tepat

133			0,087837	Tepat
134			0,088727	Tepat
135			0,088222	Tepat
136	ي		0,087490	Tepat
137			0,086820	Tepat
138			0,082737	Tepat
139			0,086356	Tepat
140			0,089093	Tepat