BAB 2
TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Pada tinjauan pustaka, telah dirangkum beberapa penelitian terdahulu yang berkaitan dengan topik penelitian ini, diantaranya sebagai berikut :

Azis dan Kurniawan (2006) melakukan penelitian mengenai pengenalan pola sidik jari menggunakan metode *bidirectional associative memory*. Hasil kesimpulan yang diperoleh adalah metode *bidirectional associative memory* dapat digunakan untuk mengenali pola dari sidik jari. Keterbatasan dari penelitian tersebut adalah metode *bidirectional associative memory* tidak dapat mengenali secara tepat pola sidik jari yang memiliki *noise* antara 40%-91%.

Dalam penelitian Pitoyo, Zuraiyah, & Qur'ania (2006) mengenai pengenalan huruf tulisan tangan menggunakan metode *zoning* dengan jumlah zona 8 dan *support vector machine*. Dalam penelitiannya, *tools* yang digunakan untuk membuat aplikasi adalah Matlab R2014a. Hasil dari penelitian ini, akurasi ketepatan dalam mengenali tulisan tangan huruf kecil akan lebih tinggi jika menggunakan metode *zoning* gabungan ICZ dan ZCZ yaitu sebesar 76,92% sedangkan untuk huruf kapital sebesar 88,46%.

Penelitian yang dilakukan (Arifin, 2009) mengenai identifikasi pola sidik jari manusia dengan noise berkisar 0-50% dan 60-91% menggunakan metode *bidirectional associative memory*. Hasil dari penelitian ini adalah metode *bidirectional associative memory* dapat mengenali semua pola sidik jari manusia dengan tepat setelah pola dikenai noise berkisar 0-30%, kecuali 1 pola sidik jari. Keterbatasan dari penelitian ini adalah bahwa *bidirectional associative memory* tidak dapat mengenali beberapa pola sidik jari manusia yang memiliki noise antara

40% - 91%. Berbeda dengan Kesumaesthy, Dayawati, & Wirayuda (2010) yang melakukan penelitian mengenai pengenalan huruf cetak dengan menggunakan 2 metode yaitu *modified direction* dan *bidirectional associative memory*. Hasil dari penelitian ini adalah penggunaan metode *bidirectional associative memory* dalam pengenalan huruf memiliki tingkat akurasi lebih tinggi dengan menggunakan ekstraksi ciri *modified direction* yaitu bernilai 39,64%.

Demikian juga Rosnita, Zarlis, & Efendi (2016) melakukan penelitian menggunakan *bidirectional associative memory* untuk mendeteksi pola tajwid pada citra Al-Qur'an. Hasil penelitian ini menunjukkan bahwa algoritma *bidirectional associative memory* dapat digunakan sebagai salah satu pendekatan untuk pendeteksian pola tajwid pada citra Al-Qur'an dengan tingkat akurasi 84%.

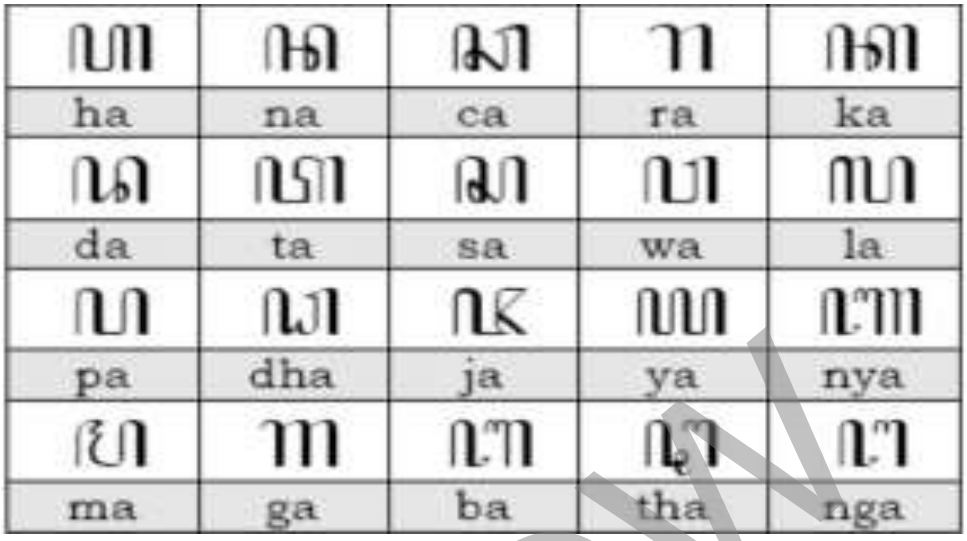
Berdasarkan penelitian-penelitian sebelumnya ekstraksi fitur *zoning* dan jaringan syaraf tiruan *bidirectional associative memory* dapat digunakan untuk mengenali beberapa jenis karakter dan bentuk. Pemilihan metode ekstraksi fitur *zoning* dan jaringan syaraf tiruan *bidirectional associative memory* pada penelitian ini adalah karena mengacu pada keberhasilan pada penelitian-penelitian sebelumnya dalam mengenali karakter dengan tingkat akurasi yang cukup tinggi. Aksara Jawa sebagai obyek penelitian juga dipilih karena belum ada penelitian mengenai pengenalan aksara Jawa menggunakan ekstraksi fitur zoning dan jaringan syaraf tiruan *bidirectional associative memory*.

2.2. Landasan Teori

2.2.1. Aksara Jawa

Aksara Jawa lebih dikenal sebagai Hanacaraka atau Carakan, aksara ini merupakan aksara turunan aksara Brahmi yang digunakan atau pernah digunakan unuk penulisan naskah-naskah berbahasa Jawa, Madura, Sunda, Bali dan Sasak. ((Wahab, 2006)). Aksara Jawa memiliki 20 huruf dasar, 20 huruf pasangan yang berfungsi menutup bunyi vokal, 8 huruf "utama", 8 pasangan huruf utama, 5 aksara

swara (huruf vokal depan), 5 aksara rekan dan 5 pasangannya. Pada aksara Jawa Carakan baku terdapat 20 huruf dasar (aksara nglegena). (Darusuprapta, 2003) Gambar huruf dasar aksara Jawa (Carakan) dapat dilihat pada Gambar 2.1.



Gambar 2.1 Aksara Jawa Carakan (Nglegena)

2.2.2. Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (*artifical neural network*) adalah cabang dari kecerdasan buatan. Jaringan syaraf tiruan merupakan sistem komputasi yang arsitektur dan operasinya diilhami dari pengetahuan tentang sel saraf biologis di dalam otak. Model jaringan syaraf ditunjukkan dengan kemampuannya dalam analisis, prediksi dan asosiasi. Kemampuan yang dimiliki jaringan syaraf tiruan dapat digunakan untuk belajar dan menghasilkan aturan atau operasi dari beberapa contoh atau *input* yang dimasukkan dan membuat prediksi tentang kemungkinan *output* yang akan muncul atau menyimpan karakteristik *input* yang diberikan kepada jaringan syaraf tiruan. (Astuti, 2009)

Proses informasi terjadi pada banyak elemen (*neuron*) , kemudian sinyal dikirim ke *neuron-neuron* melalui penghubung. Setiap penghubung antar *neuron* memiliki bobot yang akan memperkuat atau memperlemah sinyal. Untuk menentukan *output*, biasanya setiap *neuron* menggunakan fungsi aktivasi yang

dikenakan pada jumlah *input* yang akan diterima. Selanjutnya *output* akan dibandingkan dengan suatu batas ambang yang telah ditentukan. (Siang, 2005)

2.2.3. Citra *Grayscale*

Citra grayscale atau citra skala keabuan memberi kemungkinan warna yang lebih banyak dari pada citra biner, karena ada nilai-nilai lain diantara nilai minimum (biasanya = 0) dan nilai maksimum. Banyaknya kemungkinan minimum dan nilai maksimumnya bergantung pada jumlah bit yang digunakan (Gonzalez, 2002).

Citra skala keabuan (grayscale) mempunyai kemungkinan warna antara hitam (minimum) dan putih (maksimum). Contoh untuk skala keabuan (grayscale) 4 bit, maka jumlah kemungkinan nilainya adalah $2^4 = 16$ (memiliki 16 warna), dan nilai maksimumnya adalah $2^4 - 1 = 15$, kemungkinan warna 0 (min) sampai 15 (maks). Sedangkan untuk skala keabuan 8 bit, maka jumlah kemungkinan nilainya adalah $2^8 = 256$ (memiliki 256 warna), dan nilai maksimumnya $2^8 - 1 = 255$, kemungkinan warna minimum 0 sampai maksimum 255. (Balza, 2005)

2.2.4. *Thresholding*

Thresholding adalah pengolahan citra menjadi citra yang memiliki dua nilai tingkat keabuan yaitu hitam dan putih. *Thresholding* memiliki proses pengembangan hasil dari *grayscaling* untuk menghasilkan citra biner (Sutoyo, 2009).

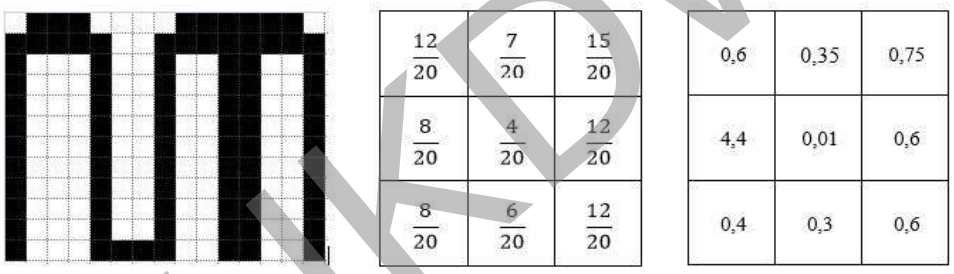
$$g(x,y) = \begin{cases} 1 & jika \ \ f(x,y) \geq T \\ 0 & jika \ \ f(x,y) < T \end{cases} \qquad [2.1]$$

Pada persamaan [2.1] di atas dimana g(x,y) merupakan citra biner hasil *thresholding* dari citra grayscale f(x,y), dan T merupakan *threshold*. Kualitas citra biner akan sangat bergantung pada nilai T. *Thresholding* akan mengubah citra RGB ataupun *grayscale* menjadi citra biner yang bernilai 0 dan 1, sehingga akan memudahkan komputer dalam melakukan ekstraksi terhadap citra.

2.2.5. Ekstraksi Ciri *Zoning*

Ekstraksi ciri adalah proses yang dilakukan untuk mendapatkan ciri khusus yang dimiliki oleh sebuah karakter. Proses ini berguna untuk mengenali sebuah karakter pada sistem pengenalan karakter. (Putra, 2010)

Menurut Putra (2012) *Zoning* merupakan salah satu metode ekstraksi ciri pada citra, ekstraksi ciri *zoning* membagi citra menjadi beberapa zona dengan ukuran yang sama. Setiap citra dibagi menjadi N x M zona dan setiap zona dihitung nilai fiturnya sehingga didapatkan fitur karakter dengan panjang N x M. dari setiap zona akan diambil cirinya. Contoh cara pembagian dan pengambilan fitur tiap zona dapat dilihat pada Gambar 2.2. Nilai fitur diambil adalah rasio piksel warna hitam terhadap luas tiap zona. (Mahastama, 2016)

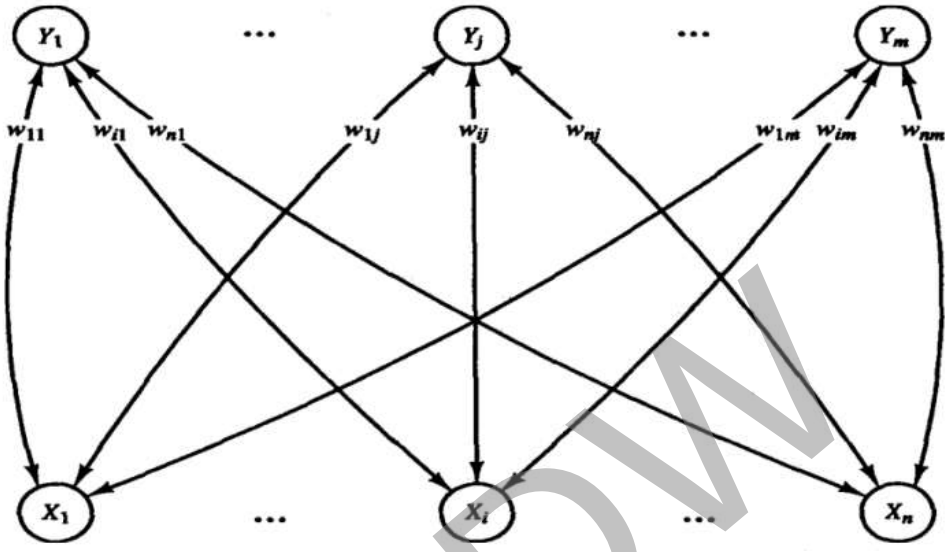


Gambar 2.2 Pembagian dan pengambilan nilai fitur tiap zona

2.2.6. Bidirectional Associative Memory (BAM)

Bidirectional Associative Memory (BAM) adalah model jaringan syaraf tiruan yang memiliki 2 lapisan (*layer*) dan terhubung penuh dari satu lapisan ke lapisan yang lainnya. Pada jaringan ini dimungkinkan adanya hubungan timbal balik antara lapisan *input* dan lapisan *output*. Namun demikian, bobot yang menghubungkan antara satu *neuron* (n) di satu lapisan dengan *neuron* (m) di lapisan lainnya akan sama dengan bobot yang menghubungkan *neuron* (m) ke *neuron* (n). Bisa dikatakan bahwa, matriks bobot yang menghubungkan *neuron-neuron* pada lapisan *output* ke lapisan *input* sama dengan transpose matriks bobot *neuron-neuron* yang menghubungkan lapisan *input* ke lapisan *output*. (Kusumadewi, 2004)

Berikut adalah arsitektur algoritma *Bidirectional Associative Memory* (BAM) memiliki layer X dan layer Y yang saling terhubung secara *bidirectional*. Dimana W merupakan bobot matriks yang dikirim dari layer X ke layer Y dan sebaliknya, dapat dilihat pada Gambar 2.3. (Fausett, 1993)



Gambar 2.3 Arsitektur Jaringan BAM

BAM yang digunakan dalam penelitian adalah BAM Diskret yang memiliki 2 jenis nilai aktivasi yaitu bilangan biner dan bipolar. Pemilihan bilangan bipolar sebagai nilai aktivasi dalam penelitian ini dinilai lebih sesuai dengan metode ekstraksi ciri yang dipakai. Dalam menentukan matriks bobot untuk vektor input bipolar dapat dilihat pada persamaan [2.2] sebagai berikut :

$$w_{ij} = \sum_p (s_i(p) * t_i(p)) \qquad [2.2]$$

Fungsi aktivasi yang digunakan untuk lapisan Y (*output*) dapat dilihat pada persamaan [2.3] berikut :

$$y_j = \begin{cases} 1;\ jika\ y_{in_j} > \theta_j \\ y_j;\ jika\ y_{in_j} = \theta_j \\ -1;\ jika\ y_{in_j} < \theta_j \end{cases} \qquad [2.3]$$

Fungsi aktivasi yang digunakan untuk lapisan X (*input*) dapat dilihat pada persamaan [2.4] berikut :

$$x_i = \begin{cases} 1;\ jika\ x_{in_i} > \theta_i \\ y_j;\ jika\ x_{in_i} = \theta_i \\ -1;\ jika\ x_{in_i} < \theta_i \end{cases} \quad [2.4]$$

Algoritma :

0. Inisialisasi bobot untuk menyimpan p vektor.
   Inisialisasi semua aktivasi sama dengan 0.
1. Set pola *input* x ke lapisan X.
2. Set pola *output* y ke lapisan Y.
3. Jika aktivasi belum konvergen kerjakan langkah 4-6
4. Perbaiki setiap unit aktivasi di lapisan Y
   Hitung :

$$y_{in_j} = \sum_i W_{ij} * x_i \quad [2.5]$$

   Hitung :

$$y_j = f(y\_in_j) \quad [2.6]$$

   Berikan informasi ke lapisan X
5. Perbaiki setiap unit aktivasi di lapisan X.
   Hitung :

$$x_{in_i} = \sum_j W_{ij} * y_j \quad [2.7]$$

   Hitung :

$$x_i = f(x\_in_i) \quad [2.8]$$

   Berikan informasi ke lapisan Y.
6. Jika vektor x dan y telah mencapai keadaan stabil, iterasi berhenti.

Keterangan :

W_{ij} : Matriks bobot hubungan ke-i dan ke-j

x_i : sinyal *input* ke-i

y_j : sinyal *output* ke-j

x_in_i : *input* hasil olahan ke-i

y_in_j : *output* hasil olahan ke-j

θ : nilai ambang

Contoh kasus perhitungan algortima BAM dapat dilihat di bawah ini : (Fausett, 1993, hlm. 144-145)

Diketahui terdapat 2 vektor yang mewakili huruf O dan X dengan elemen-elemennya berupa bilangan bipolar -1 atau 1, sebagai berikut :

Tabel 2.1 Tabel contoh kasus perhitungan algoritma BAM

Huruf	Vektor bipolar	Vektor Target
O	1 1 1 1 -1 1 1 1 1	-1 1
X	1 -1 1 -1 1 -1 1 -1 1	1 1

Pada tabel 2.1 di atas Vektor target merupakan vektor yang akan dicapai dengan membandingkan hasil perhitungan algoritma BAM. Penentuan ukuran vektor bipolar target adalah dengan $2^1 = 2$ karena variasi elemen pada vektor huruf berjumlah 2 yaitu 1 dan -1 dan variasi huruf berjumlah 2 yaitu O dan X, maka jumlah variasi elemen vektor dipangkatkan 1 untuk mencapai jumlah variasi huruf. Elemen-elemen pada vektor target ditentukan sendiri dan dibedakan untuk masing-masing huruf. Pada Inisialisasi bobot dilakukan dengan perkalian antara vektor bipolar dengan target sehingga akan menghasilkan matriks berukuran 2x9 sebagai berikut :

Matriks bobot p=1 untuk huruf O :

$$w_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} * [-1 \quad 1] = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Matriks bobot p=2 untuk huruf X :

$$w_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} * [1 \quad 1] = \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \end{bmatrix}$$

Matriks bobot keseluruhan :

$$W = w_1 + w_2$$

$$W = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 2 & 0 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 0 & 2 \end{bmatrix}$$

Matriks bobot tersebut menghubungkan antara neuron-neuron di lapisan input ke neuron-neuron yang ada di lapisan output.

Matriks bobot diujikan dengan vektor pertama yang mewakili huruf O, maka output yang diperoleh adalah sebagai berikut :

$$y\_in_1 = x_1 * W = [1 \quad 1 \quad 1 \quad 1 \quad -1 \quad 1 \quad 1 \quad 1 \quad 1] * \begin{bmatrix} 0 & 2 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 0 & 0 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 0 & 2 \end{bmatrix} = [-10 \quad 8]$$

Karena (y_in1(1)=–10 < 0, maka y1(1)=-1) dan (y_in1(2)=8 > 0, maka y1(2)=1), maka nilai y1 = [-1 1], sama dengan target yang diharapkan, sedangkan untuk vektor input kedua yang mewakili Huruf X, output yang diperoleh adalah sebagai berikut :

$$y\_in_2 = x_2 * W = [1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1] * \begin{bmatrix} 0 & 2 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 0 & 0 \\ -2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$= [10 \quad 8]$$

Karena (y_in2(1)=10 > 0, maka y2(1)=1) dan (y_in2(2)=8 > 0, maka y2(2)=1), maka nilai y2 = [1 1], sama dengan target yang diharapkan.

Matriks yang menghubungkan neuron-neuron di lapisan output ke neuron-neuron yang ada di lapisan input adalah matriks transpose dari bobot atau W^T.

$$W^T = \begin{bmatrix} 0 & -2 & 0 & -2 & 2 & -2 & 0 & -2 & 0 \\ 2 & 0 & 2 & 0 & 2 & 0 & 2 & 0 & 2 \end{bmatrix}$$

Pengujian dengan arah sebaliknya dimana y digunakan sebagai input dan x menjadi vektor target. Pengujian pada vektor input pertama y1 = [-1 1], maka output yang diperoleh adalah sebagai berikut :

$$x_{in_1} = y_1 * W^T = [-1 \quad 1] * \begin{bmatrix} 0 & -2 & 0 & -2 & 2 & -2 & 0 & -2 & 0 \\ 2 & 0 & 2 & 0 & 2 & 0 & 2 & 0 & 2 \end{bmatrix}$$

$$= [2 \quad 2 \quad 2 \quad 2 \quad -2 \quad 2 \quad 2 \quad 2 \quad 2]$$

Output sama dengan target [1 1 1 1 -1 1 1 1 1], dikenali sebagai huruf O. Pengujian pada vektor input kedua y2 = [1 1], maka output yang diperoleh adalah sebagai berikut :

$$x_{in_2} = y_2 * W^T = [1 \quad 1] * \begin{bmatrix} 0 & -2 & 0 & -2 & 2 & -2 & 0 & -2 & 0 \\ 2 & 0 & 2 & 0 & 2 & 0 & 2 & 0 & 2 \end{bmatrix}$$
$$= [2 \quad -2 \quad 2 \quad -2 \quad 2 \quad -2 \quad 2 \quad -2 \quad 2]$$

Output sama dengan target [1 -1 1 -1 1 -1 1 -1 1], dikenali sebagai huruf X, pengujian BAM terhadap vektor input dari x ke y dan sebaliknya, dari y ke x telah mencapai keadaan stabil, maka iterasi berhenti.

BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1. Analisis Kebutuhan

Pada penelitian ini diperlukan beberapa *software* terkait dengan pengujian aksara Jawa menggunakan ekstraksi fitur *zoning* dan jaringan syaraf tiruan *bidirectional associative memory* untuk klasifikasi pengenalan data uji.

3.1.1 Kebutuhan *Software*

Dalam penelitian ini sistem operasi yang digunakan adalah *Windows 10 Home 64-bit*, selain itu diperlukan berbagai *software* tambahan sebagai berikut :

a. Matlab R2013a untuk membangun sistem
b. Paint untuk mengedit citra
c. Microsoft Office 2016 untuk pembuatan laporan

3.1.2 Kebutuhan *Hardware*

Spesifikasi *hardware* yang digunakan dalam penelitian ini adalah sebagai berikut :

a. *Notebook* Asus X441UV dengan *processor* Intel Core i3-6006U @2.0 GHz, *memory* 4GB DDR4 dan *harddisk* 500GB.
b. Smartphone untuk pengambilan foto citra.

3.1.3 Kebutuhan Data

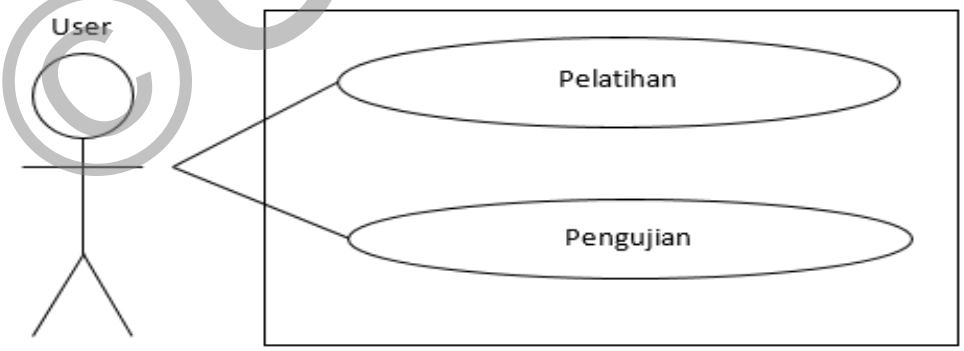
Kebutuhan Data yang akan digunakan dalam penelitian ini adalah sebagai berikut :

a. Karakter aksara Jawa Carakan (Nglegena) yang berjumlah 20 karakter.
b. Data latih terdiri dari 1 citra per karakter, sedangkan data uji masing-masing 5 citra per karakter.
c. Data latih dan data uji merupakan citra tulisan tangan yang difoto dan memiliki format (.jpg) .
d. Ukuran citra untuk data latih dan data uji adalah 100 x 100 piksel.

3.2. Perancangan Sistem

3.2.1 *Usecase* Diagram

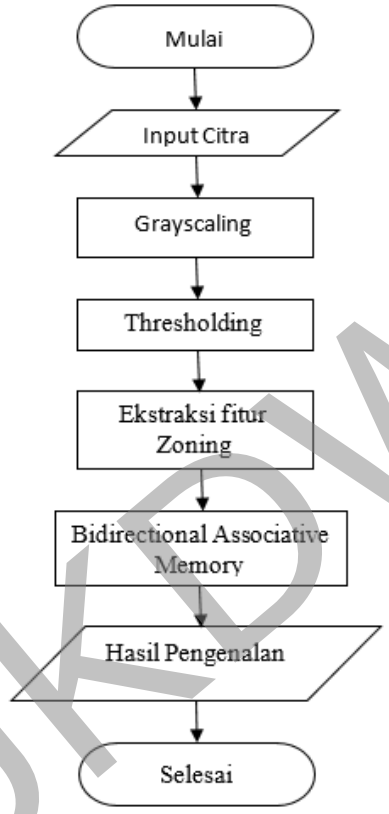
Sistem dibangun dengan satu pengguna yang dapat menggunakan semua fitur yang ada dalam sistem. Pengguna dapat memilih citra yang akan diinputkan ke dalam sistem untuk diklasifikasikan, mengubah citra menjadi citra *grayscale*, mengubah citra menjadi citra biner melalui proses *thresholding*, ekstraksi fitur *zoning*, dan melakukan pengenalan input dengan seluruh citra yang sudah dilatih dan disimpan pada *database* sistem menggunakan *Bidirectional Associative Memory*. Diagram *usecase* dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram usecase sistem

3.2.2 Diagram Alir Sistem

Berikut adalah urutan proses yang akan dilakukan di dalam sistem digambarkan dalam diagram alir sistem.



Gambar 3.2 Flowchart Sistem

Tahapan proses sesuai dengan Flowchart pada gambar 3.1 adalah sebagai berikut :

a. Tahap input citra, pada tahap ini citra pengguna memasukkan citra digital berupa foto satu karakter aksara Jawa dengan format .jpg.
b. Tahap *grayscaling*, pada tahap ini citra yang dimasukkan akan diubah menjadi citra *grayscale* untuk mengurangi proses perhitungan nilai vektor fitur.
c. Tahap *thresholding*, pada tahap ini citra *grayscale* diubah menjadi citra biner untuk memisahkan obyek *foreground* dengan

background, obyek *foreground* memiliki nilai piksel 1 sedangkan *background* bernilai 0.

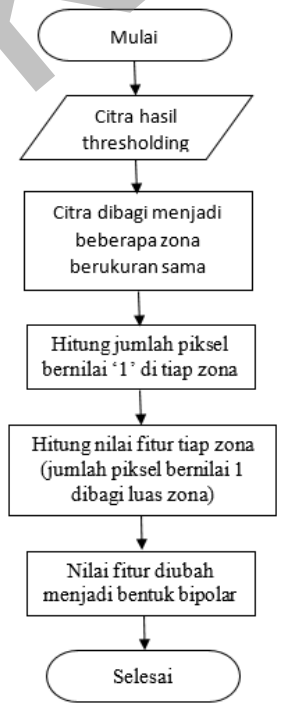
d. Tahap ekstraksi fitur *zoning*, pada tahap ini citra dibagi menjadi beberapa zona yang berukuran sama untuk dihitung nilai piksel yang merupakan obyek *foreground* di tiap zona.

e. Tahap pelatihan BAM, menggunakan bobot awal.

f. Tahap pengenalan aksara Jawa, pada tahap ini dilakukan pengenalan dengan menggunakan bobot akhir, kemudian sistem akan menampilkan hasil pengenalan aksara Jawa dalam tulisan latin.

3.2.3 Zoning

Pada tahapan ekstraksi ciri menggunakan *Zoning* yaitu membagi citra menjadi beberapa zona yang berukuran sama. Berdasarkan gambar 2.2 dari tiap zona yang terdapat piksel yang bernilai 1 (*foreground*) akan dihitung jumlahnya dan dibagi dengan luas zona tersebut.

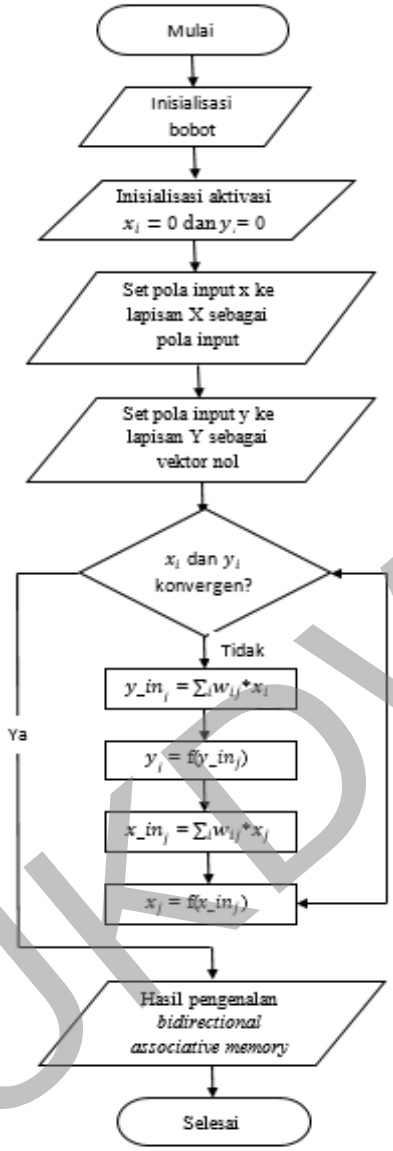


Gambar 3.3 Flowchart ekstraksi fitur Zoning

Setelah membagi jumlah piksel bernilai ‘1’ dengan luas zona maka akan didapatkan nilai fitur dari tiap zona. Nilai tersebut kemudian diubah ke dalam bentuk bipolar dengan membandingkannya dengan nilai *threshold*. bentuk bipolar tersebut akan berjumlah sama dengan jumlah zona yang ditentukan dan disimpan dalam bentuk vektor.

3.2.4 *Bidirectional Associative Memory*

Pada tahapan pengenalan citra menggunakan BAM hasil dari ekstraksi ciri *Zoning* pada proses sebelumnya yang berupa vektor bipolar akan menjadi *input* untuk diproses menggunakan BAM. *Input* citra dari hasil ekstaksi ciri *zoning* akan dibandingkan dengan bobot total dari seluruh citra data latih untuk dicari kecocokannya secara *bidirectional*. Tiap vektor target yang telah ditentukan akan menjadi *output* yang diharapkan, dengan *input* nya berupa vektor hasil *zoning* dari citra data uji akan dibandingkan dengan bobot keseluruhan data latih untuk mencocokannya dengan salah satu *output* berupa vektor target. Dilakukan pencocokan lagi dengan dibalik arah yaitu vektor target sebagai *input* akan dibandingkan dengan bobot untuk memperoleh *output* berupa vektor hasil *zoning*.



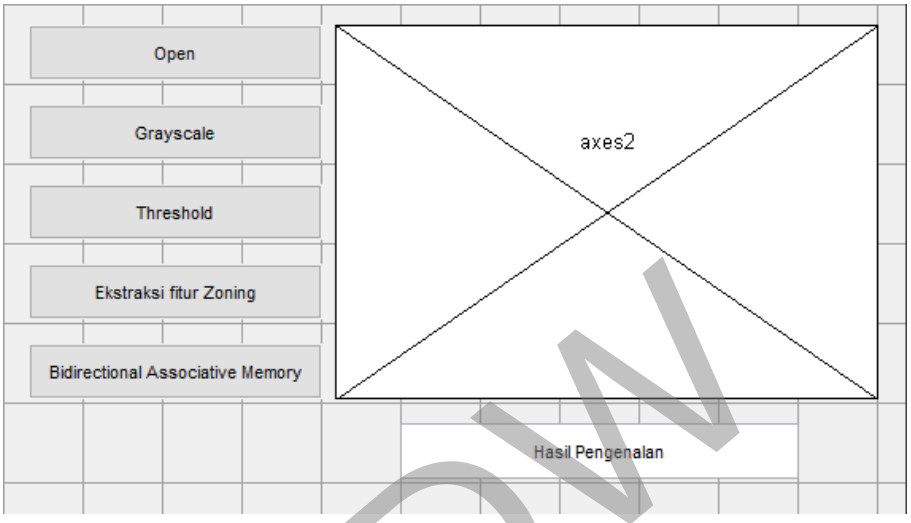
Gambar 3.4 Flowchart Bidirectional Associative Memory

Berdasarkan Gambar 3.4 urutan proses *Bidirectional Associative Memory* adalah sebagai berikut :

0. Inisialisasi bobot
1. Inisialisasi fungsi aktivasi $x_i = 0$ dan $y_i = 0$
2. Masukkan pola *input* x ke lapisan X
3. Masukkan pola *input* y ke lapisan Y
4. Selama x dan y belum konvergen lakukan langkah 6
5. Jika x dan y konvergen maka proses pengenalan berhasil

3.3. Rancangan Antarmuka

Rancangan antarmuka sistem yang akan dibangun adalah sebagai berikut :



Gambar 3.5 Rancangan desain antarmuka sistem

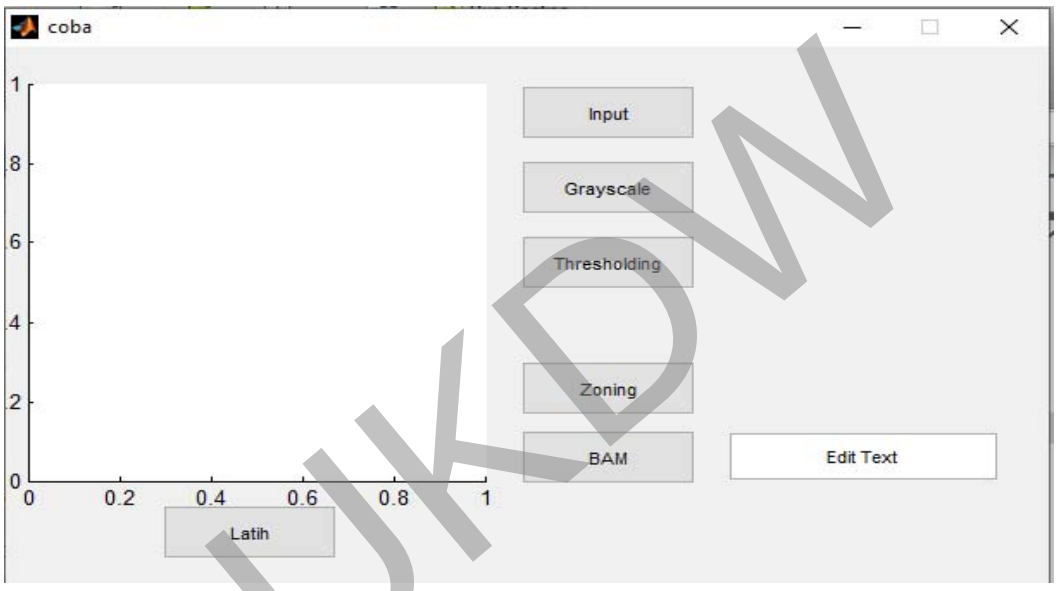
Rancangan Gambar 3.5 terdiri dari berbagai komponen sebagai berikut :

a. Tombol *Open* berfungsi untuk menampilkan gambar citra yang akan diproses.
b. Tombol *Grayscale* berfungsi memproses citra yg dimasukkan dan diubah menjadi citra *grayscale*.
c. Tombol *Threshold* berfungsi memproses citra menjadi citra biner.
d. Tombol Ekstraksi fitur *Zoning* berfungsi menerapkan ekstraksi fitur *zoning*.
e. Tombol *Bidirectional Associative Memory* berfungsi menerapkan jaringan syaraf tiruan *Bidirectional Associative Memory*.
f. *Picture Box* berfungsi menampilkan citra.
g. *Text Box* Hasil Pengenalan berfungsi menampilkan hasil pengenalan citra.

BAB 4
IMPLEMENTASI DAN ANALISIS SISTEM

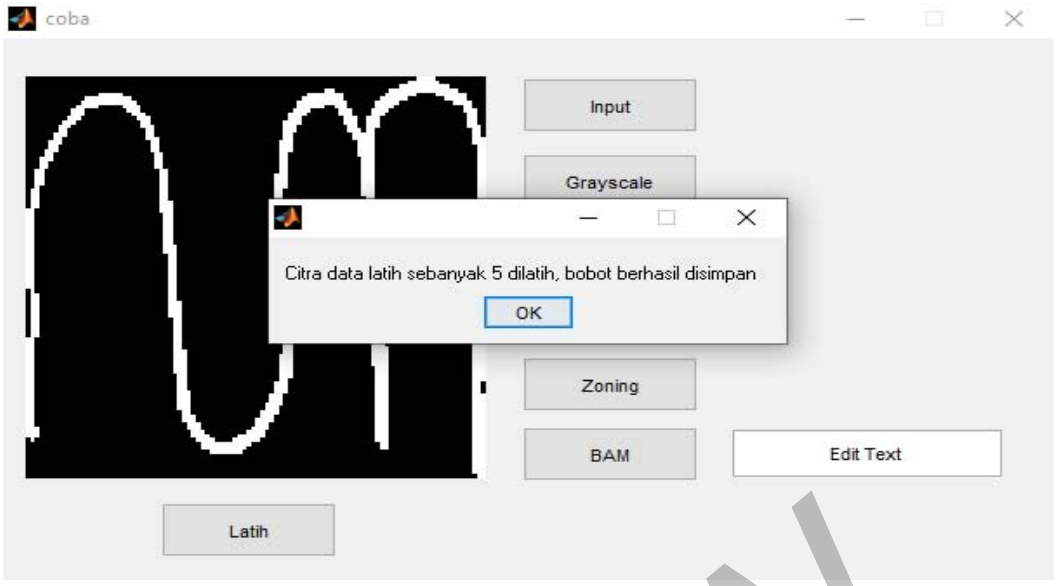
4.1. Implementasi Sistem

Pada Gambar 4.1 merupakan tampilan antarmuka awal sistem dengan *Picture Box* masih kosong karena belum dilakukan proses *Input* citra.



Gambar 4.1 Tampilan antarmuka awal sistem

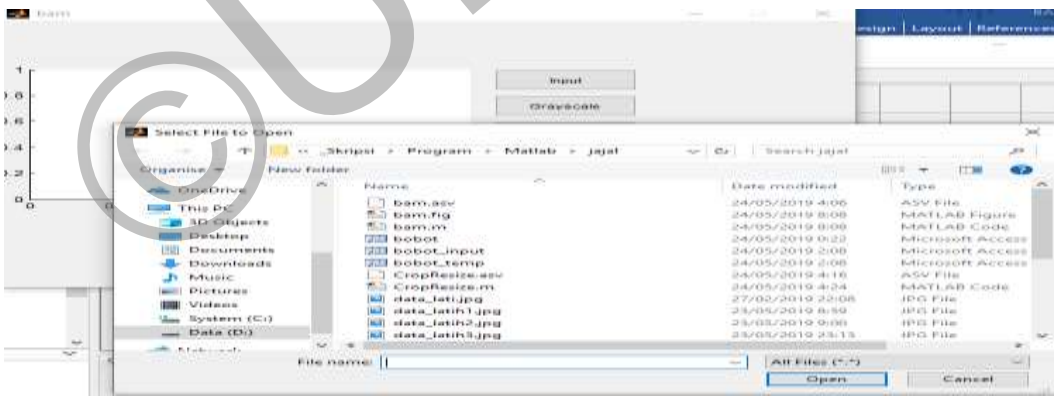
Tampilan antarmuka sistem terdiri dari 1 halaman utama sistem yang memiliki 6 tombol. Tombol *Input* yang berfungsi untuk memasukkan *file* citra yang akan diuji. Sebelum melakukan penginputan citra untuk proses pengujian, dilakukan proses pelatihan data menggunakan tombol latih. Data akan dilatih menggunakan BAM dan disimpan ke dalam *database* sistem.



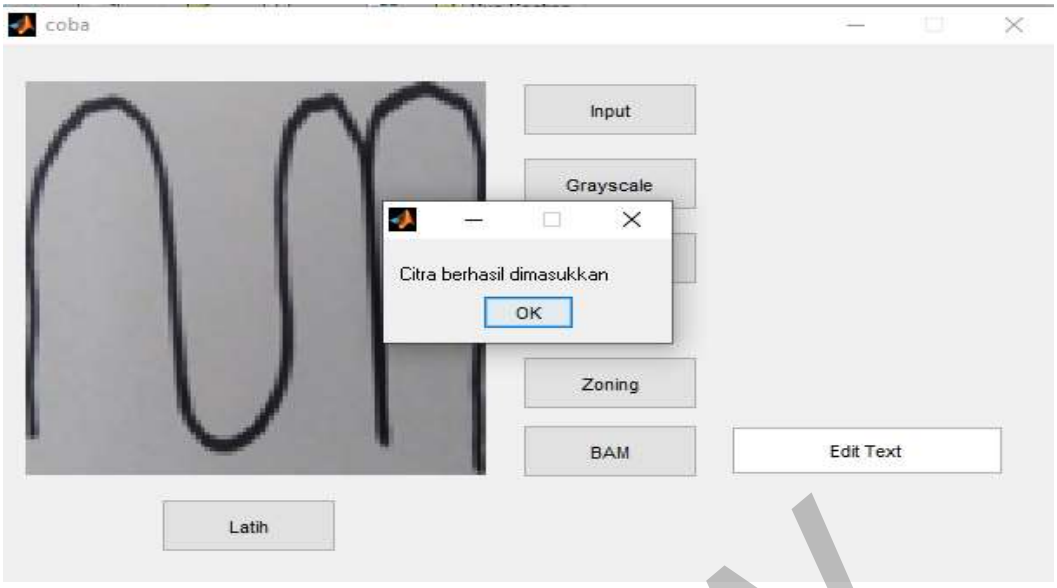
Gambar 4.2 Tampilan antarmuka setelah melakukan pelatihan data

Gambar 4.2 menunjukkan proses pelatihan data berhasil dilakukan, pada proses ini sistem akan melatih seluruh data latih yang ada di dalam *folder* yang sama. Proses pengujian dapat dilakukan setelah proses pelatihan data dilakukan.

Pada Gambar 4.3 adalah tampilan setelah pengguna menekan tombol *Input* adalah proses memilih citra yang akan dimasukkan ke dalam sistem.



Gambar 4.3 Tampilan setelah menekan tombol Input

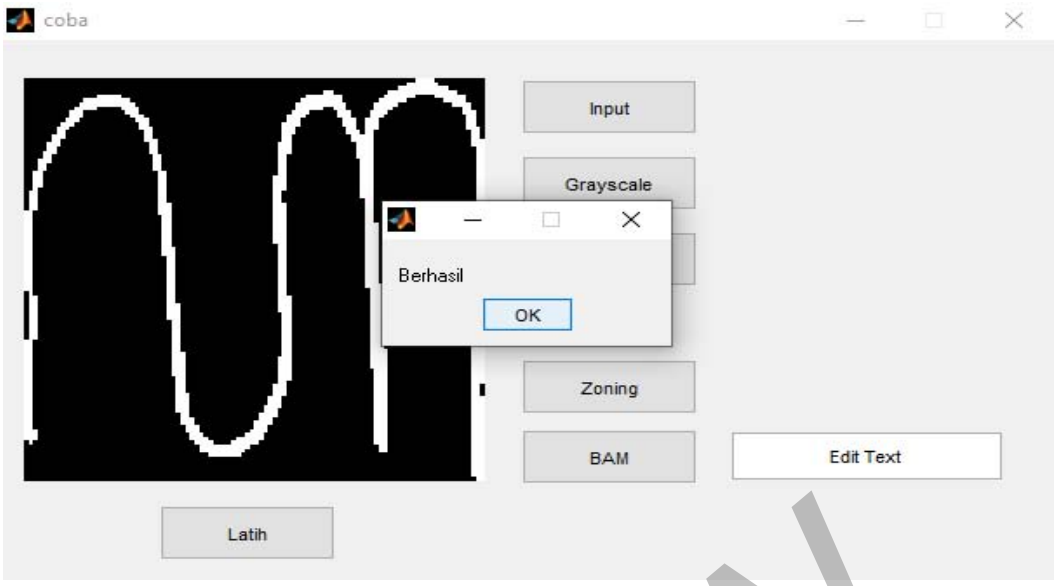


Gambar 4.4 Tampilan setelah dimasukkan citra

Setelah memilih citra yang akan dimasukkan, citra yang sudah dipilih akan ditampilkan di *Picture Box* dan akan muncul *Dialog Box* yang menandakan citra berhasil dimasukkan ke dalam sistem seperti yang ada di Gambar 4.4.

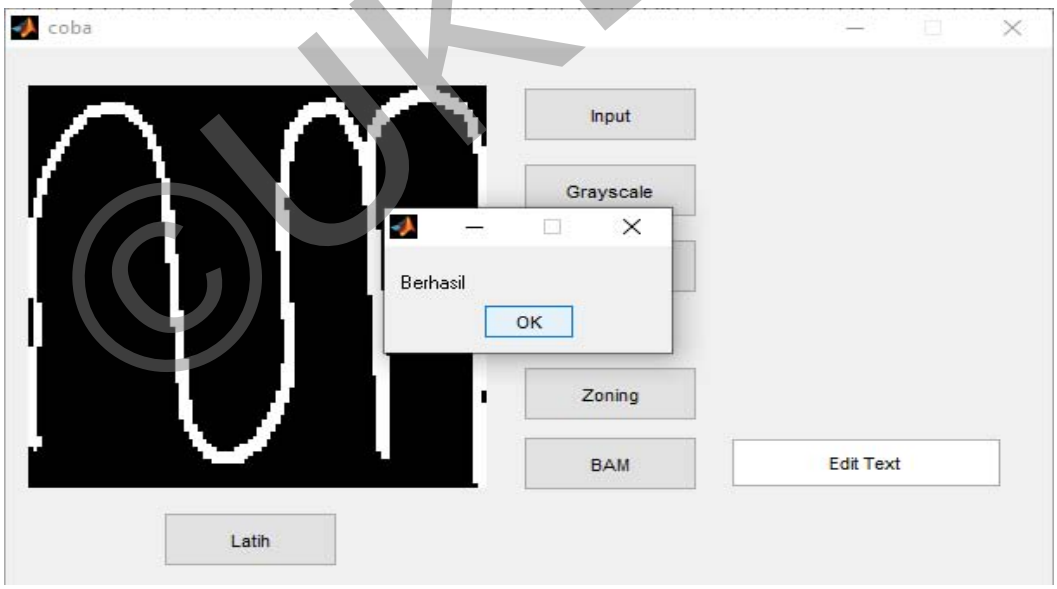
Tombol *grayscale* berfungsi mengubah citra yang dimasukkan menjadi citra *grayscale*. Citra yang sudah dimasukkan ke dalam sistem akan diubah menjadi citra grayscale setelah Pengguna menekan tombol *Grayscale*. Citra hasil *Grayscale* akan ditampilkan dalam *Picture Box* menggantikan citra yang sebelumnya. Jika citra yang dimasukkan merupakan citra *Grayscale* maka citra yang ada dalam *Picture Box* tidak berubah dan muncul *Dialog Box* yang menandakan proses berhasil dilakukan, proses bisa dilanjutkan ke *Thresholding*.

Pada Tombol *Thresholding* berfungsi memproses citra yang telah dijadikan *grayscale* sebelumnya dan mengubahnya menjadi citra biner. Pada Gambar 4.5 citra grayscale diubah menjadi citra biner setelah Pengguna menekan tombol *Thresholding*, hasil dari proses tersebut ditampilkan di dalam *Picture Box* dan muncul *Dialog Box* yang menandakan proses berjalan dengan baik.



Gambar 4.5 Tampilan setelah melakukan proses Thresholding

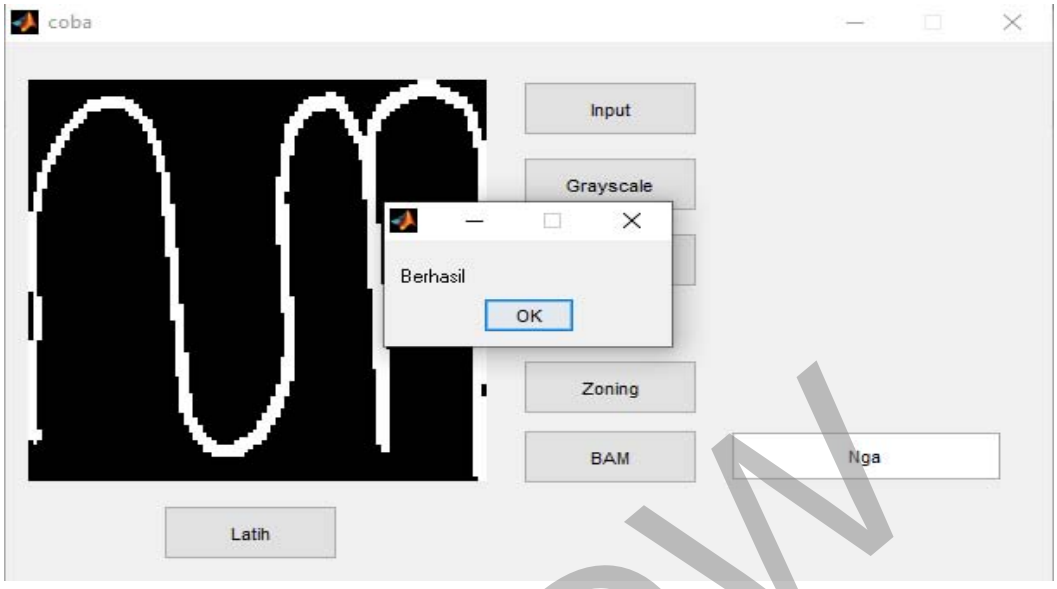
Tombol *Zoning* berfungsi mengambil ciri dari citra yang telah diubah menjadi citra biner, *Dialog Box* muncul ketika tombol Zoning ditekan seperti pada Gambar 4.6.



Gambar 4.6 Tampilan antarmuka setelah menekan tombol Zoning

Pengenalan menggunakan BAM dilakukan dengan menekan tombol BAM. Tombol berfungsi untuk mengenali citra yang dimasukkan dengan menerapkan

jaringan syaraf tiruan BAM. Sistem kemudian menampilkan hasilnya pada kotak *Edit Text* seperti pada Gambar 4.7.



Gambar 4.7 Tampilan antarmuka hasil pengenalan citra menggunakan BAM

4.2. Analisis Sistem

Pada Tabel 4.1 menunjukkan daftar data latih dan vektor target untuk *output* dari setiap karakter aksara Jawa yang diperoleh melalui rumus 2^5 >= jumlah karakter aksara Jawa. Karena jumlah karakter aksara Jawa *Nglegena* berjumlah 20 maka dipilih 2^5=32 karena lebih dari jumlah karakter aksara Jawa. Data latih yang dipakai adalah 20 citra aksara Jawa *Nglegena* sedangkan untuk data uji berjumlah 5 untuk tiap aksara totalnya berjumlah 100 data uji.

Tabel 4. 1
Data Target

No	Aksara Jawa	Target	Output
1	Ha	1 1 1 1 1	Ha
2	Na	-1 1 1 1 1	Na
3	Ca	1 -1 1 1 1	Ca
4	Ra	1 1 -1 1 1	Ra
5	Ka	1 1 1 -1 1	Ka
6	Da	1 1 1 1 -1	Da
7	Ta	-1 -1 1 1 1	Ta
8	Sa	1 -1 -1 1 1	Sa
9	Wa	1 1 -1 -1 1	Wa
10	La	1 1 1 -1 -1	La
11	Pa	1 1 -1 1 -1	Pa
12	Dha	1 -1 1 -1 1	Dha
13	Ja	-1 1 -1 1 1	Ja
14	Ya	-1 -1 -1 1 1	Ya
15	Nya	1 -1 -1 -1 1	Nya
16	Ma	1 1 -1 -1 -1	Ma
17	Ga	1 -1 1 -1 -1	Ga
18	Ba	-1 1 -1 1 -1	Ba
19	Tha	-1 -1 1 -1 1	Tha
20	Nga	-1 -1 -1 -1 -1	Nga

4.2.1 Analisis Hasil Ekstraksi Ciri Zoning

Nilai dari ekstraksi fitur berupa vektor bipolar bernilai 1 atau -1 akan dipakai dalam proses pengenalan BAM yang berfungsi sebagai parameter input pada pengenalan BAM. Hasil pengujian untuk pelatihan data latih zoning dari sebagian dari citra data latih yang berjumlah 5 citra, didapatkan 5 vektor bipolar dari masing-masing data latih dapat dilihat pada Tabel 4.2.

Tabel 4. 2

Hasil ekstraksi ciri Zoning dari data latih

Data latih	Vektor Zoning
data_latih_1	-1 -1 -1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 1 1 1 -1 -1 -1 1 -1 -1
data_latih_2	1 1 -1 1 -1 1 1 -1 1 1 1 1 1 1 1 1 -1 1 1 1 1 -1 1 1
data_latih_3	1 1 1 1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 1 1 1
data_latih_4	1 1 1 1 -1 -1 1 -1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 -1 1 -1 -1 1
data_latih_5	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
data_latih_6	1 1 -1 -1 -1 1 1 -1 1 1 1 1 -1 1 1 1 1 1 1 1 1 1 -1 1 1
data_latih_7	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
data_latih_8	1 1 -1 1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 1 1 1 1 1 1 -1 1
data_latih_9	1 1 1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 -1 1 1 1
data_latih_10	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1
data_latih_11	1 1 -1 1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 -1 1 1 -1 1 1 1 1
data_latih_12	1 1 -1 1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 1 1 1 1 1 1 1 1
data_latih_13	1 1 1 1 1 1 1 1 1 1 1 1 1 -1 1 1 1 1 1 1 1 1 1 1 1
data_latih_14	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
data_latih_15	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
data_latih_16	1 1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
data_latih_17	1 1 1 1 1 1 1 1 1 1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 -1 1
data_latih_18	1 1 1 1 1 1 1 -1 1 1 1 1 -1 1 1 1 1 1 1 1 1 1 1 1 1
data_latih_19	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
data_latih_20	1 1 1 1 1 1 1 1 1 1 1 1 1 -1 1 1 1 1 1 1 1 1 1 1 1

Dari hasil ekstraksi ciri *zoning* didapatkan vektor bipolar berukuran 25x1 yang akan menjadi *input* pada proses pengenalan BAM.

4.2.2 Analisis Hasil Pengenalan BAM

Penelitian ini dilakukan menggunakan 1 jenis data latih yang akan diujikan dengan 5 jenis sampel data uji aksara Jawa *Carakan*.

Tabel 4. 3

Tabel hasil pengujian data uji 1

Input	Hasil	Benar / Salah
ha1.jpg	nga	Salah
na1.jpg	nga	Salah
ca1.jpg	nga	Salah
ra1.jpg	ha	Salah
ka1.jpg	nga	Salah
da1.jpg	nga	Salah
ta1.jpg	nga	Salah
sa1.jpg	nga	Salah
wa1.jpg	nga	Salah
la1.jpg	nga	Salah
pa1.jpg	nga	Salah
dha1.jpg	nga	Salah
ja1.jpg	nga	Salah
ya1.jpg	nga	Salah
nya1.jpg	nga	Salah
ma1.jpg	nga	Salah
ga1.jpg	nga	Salah
ba1.jpg	nga	Salah
tha1.jpg	nga	Salah
nga1.jpg	nga	Benar

Pada pengujian data uji pertama yang ditunjukkan pada Tabel 4.3 citra ca1.jpg dikenali sebagai aksara Ha, sedangkan selain itu dikenali sebagai aksara Nga. Citra pada data uji 1 yang dikenali dengan benar berjumlah 1, sehingga presentase keberhasilannya adalah 1/20*100% = 5%.

Tabel 4.4

Tabel hasil pengujian data uji 2

Input	Hasil	Benar / Salah
ha2.jpg	nga	Salah
na2.jpg	nga	Salah
ca2.jpg	nga	Salah
ra2.jpg	nga	Salah
ka2.jpg	nga	Salah
da2.jpg	nga	Salah
ta2.jpg	nga	Salah
sa2.jpg	nga	Salah
wa2.jpg	nga	Salah
la2.jpg	nga	Salah
pa2.jpg	nga	Salah
dha2.jpg	nga	Salah
ja2.jpg	nga	Salah
ya2.jpg	nga	Salah
nya2.jpg	nga	Salah
ma2.jpg	nga	Salah
ga2.jpg	nga	Salah
ba2.jpg	nga	Salah
tha2.jpg	nga	Salah
nga2.jpg	nga	Benar

Berbeda dengan pengujian pertama yang ada citra lain yang dikenali selain aksara Nga, pada pengujian data uji kedua yang ditunjukkan pada Tabel 4.4, semua citra pada data uji 2 dikenali sebagai aksara Nga. jumlah citra yang dapat dikenali dengan benar pada data uji berjumlah 1. Presentase keberhasilan pada pengujian data uji 2 adalah 1/20*100% = 5%.

Tabel 4.5

Tabel hasil pengujian data uji 3

Input	Hasil	Benar / Salah
ha3.jpg	nga	Salah
na3.jpg	nga	Salah
ca3.jpg	nga	Salah
ra3.jpg	nga	Salah
ka3.jpg	nga	Salah
da3.jpg	nga	Salah
ta3.jpg	nga	Salah
sa3.jpg	nga	Salah
wa3.jpg	nga	Salah
la3.jpg	nga	Salah
pa3.jpg	nga	Salah
dha3.jpg	nga	Salah
ja3.jpg	nga	Salah
ya3.jpg	nga	Salah
nya3.jpg	nga	Salah
ma3.jpg	nga	Salah
ga3.jpg	nga	Salah
ba3.jpg	nga	Salah
tha3.jpg	nga	Salah
nga3.jpg	nga	Benar

Pengujian data uji ketiga yang ditunjukkan pada Tabel 4.5, semua citra pada data uji 3 dikenali sebagai aksara Nga, sehingga citra yang dikenali benar pada data uji 3 berjumlah 1. Presentase keberhasilan pada pengujian data uji 3 adalah 5%.

Tabel 4.6

Tabel hasil pengujian data uji 4

Input	**Hasil**	**Benar / Salah**
ha4.jpg	nga	Salah
na4.jpg	nga	Salah
ca4.jpg	nga	Salah
ra4.jpg	nga	Salah
ka4.jpg	nga	Salah
da4.jpg	nga	Salah
ta4.jpg	nga	Salah
sa4.jpg	nga	Salah
wa4.jpg	nga	Salah
la4.jpg	nga	Salah
pa4.jpg	nga	Salah
dha4.jpg	nga	Salah
ja4.jpg	nga	Salah
ya4.jpg	nga	Salah
nya4.jpg	nga	Salah
ma4.jpg	nga	Salah
ga4.jpg	nga	Salah
ba4.jpg	nga	Salah
tha4.jpg	nga	Salah
nga4.jpg	nga	Benar

Pada Pengujian data uji keempat yang ditunjukkan pada Tabel 4.6, semua citra pada data uji 4 dikenali sebagai aksara Nga, sehingga citra yang dikenali benar

pada data uji 4 berjumlah 1. Presentase keberhasilan pada pengujian data uji 4 adalah 5%.

Tabel 4.7

Tabel hasil pengujian data uji 5

Input	Hasil	Benar / Salah
ha5.jpg	nga	Salah
na5.jpg	nga	Salah
ca5.jpg	nga	Salah
ra5.jpg	nga	Salah
ka5.jpg	nga	Salah
da5.jpg	nga	Salah
ta5.jpg	nga	Salah
sa5.jpg	nga	Salah
wa5.jpg	nga	Salah
la5.jpg	nga	Salah
pa5.jpg	nga	Salah
dha5.jpg	nga	Salah
ja5.jpg	nga	Salah
ya5.jpg	nga	Salah
nya5.jpg	nga	Salah
ma5.jpg	nga	Salah
ga5.jpg	nga	Salah
ba5.jpg	nga	Salah
tha5.jpg	nga	Salah
nga5.jpg	nga	Benar

Pengujian data uji kelima yang ditunjukkan pada Tabel 4.7, semua citra pada data uji 5 dikenali sebagai aksara Nga, sehingga citra yang dikenali benar pada

data uji 5 berjumlah 1. Presentase keberhasilan pada pengujian data uji 5 adalah 5%.

Setelah dilakukan pengenalan aksara jawa menggunakan BAM untuk seluruh dataset uji dapat disimpulkan bahwa secara keseluruhan sistem hanya bisa mengenali aksara Nga dengan akurasi dari setiap pengujian dataset uji adalah 5%. Pada pengujian dataset uji kedua sistem dapat mengenali aksara selain Nga, yaitu pada *file* ca1.jpg dikenali sebagai aksara Ha.

Tabel 4. 8

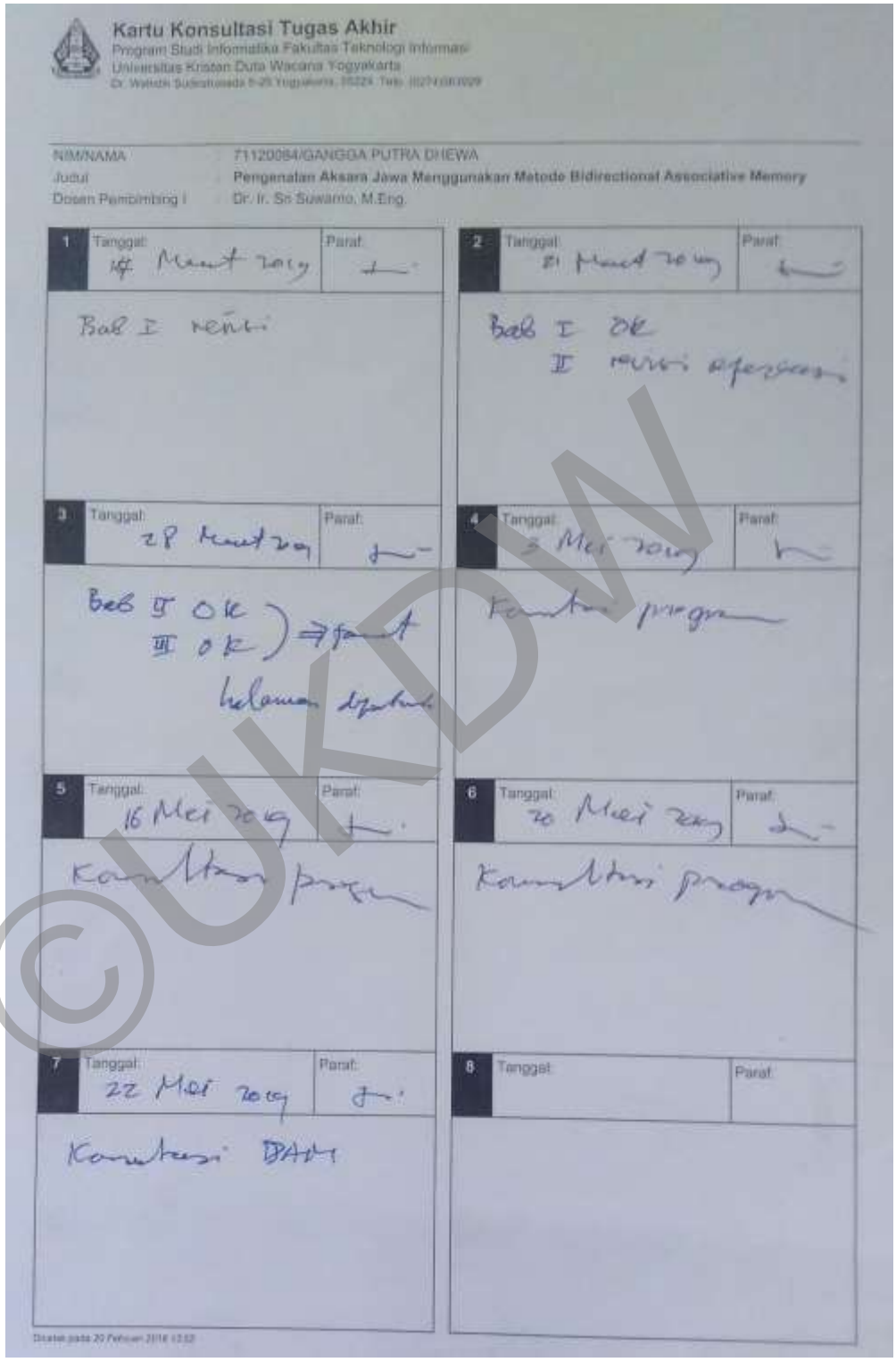
Tabel Hasil pengujian secara keseluruhan

Dataset uji	Jumlah Pengujian	Jumlah Benar	Jumlah Salah	Akurasi
1	20	1	19	5%
2	20	1	19	5%
3	20	1	19	5%
4	20	1	19	5%
5	20	1	19	5%
Total	100	5	95	5%

Pada tabel 4.8 hasil pengujian keseluruhan terhadap setiap dataset uji jumlah citra dikenali benar berjumlah 5 dari total 100 citra data uji, sisanya sejumlah 95 citra data uji dikenali salah. Presentase keberhasilan keseluruhan pengujian adalah 5/100*100% = 5%.

LAMPIRAN

SCAN KARTU KONSULTASI SKRIPSI



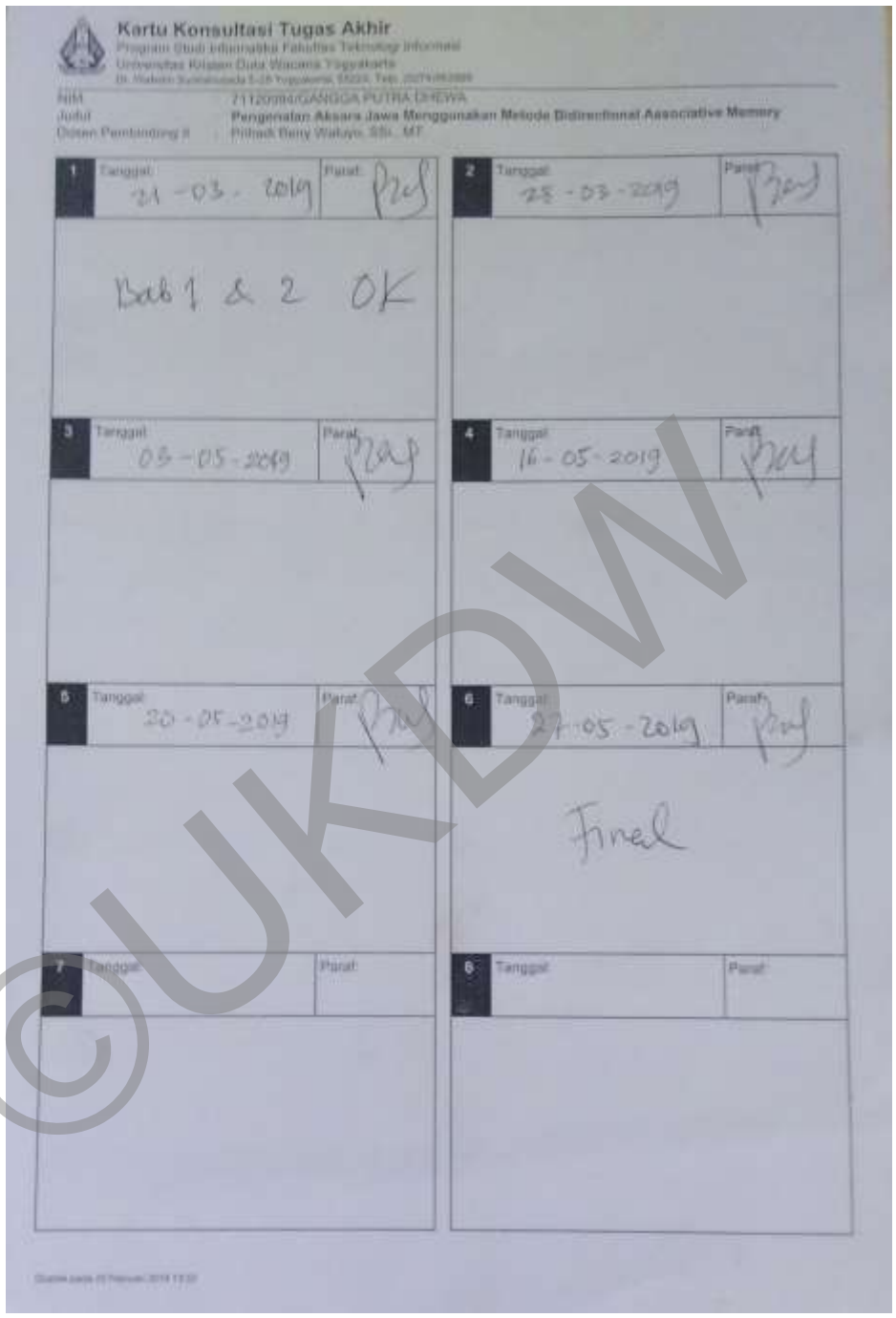
Kartu Konsultasi Tugas Akhir
Program Studi Informatika Fakultas Teknologi Informasi
Universitas Kristen Duta Wacana Yogyakarta
Dr. Wahidin Sudirohusodo 5-25 Yogyakarta, 55224. Telp. (0274) 563929

NIM/NAMA : 71120084/GANGGA PUTRA DHEWA
Judul : **Pengenalan Aksara Jawa Menggunakan Metode Bidirectional Associative Memory**
Dosen Pembimbing I : Dr. Ir. Ss Suwarno, M.Eng.

1	Tanggal: 14 Maret 2019	Paraf:	Bab I revisi
2	Tanggal: 21 Maret 2019	Paraf:	Bab I OK II revisi aplikasi
3	Tanggal: 28 Maret 2019	Paraf:	Bab II OK III OK) ⇒ lanjut halaman depan
4	Tanggal: 2 Mei 2019	Paraf:	Konsultasi program
5	Tanggal: 16 Mei 2019	Paraf:	Konsultasi program
6	Tanggal: 20 Mei 2019	Paraf:	Konsultasi program
7	Tanggal: 22 Mei 2019	Paraf:	Konsultasi BAB I
8	Tanggal:	Paraf:	

Dicetak pada 20 Februari 2018 13:02

SCAN KARTU KONSULTASI SKRIPSI



Kartu Konsultasi Tugas Akhir

Program Studi Informatika Fakultas Teknologi Informasi
Universitas Kristen Duta Wacana Yogyakarta
Jl. Raden Soesilo No 5-25 Yogyakarta, 55224, Telp. (0274) 563929

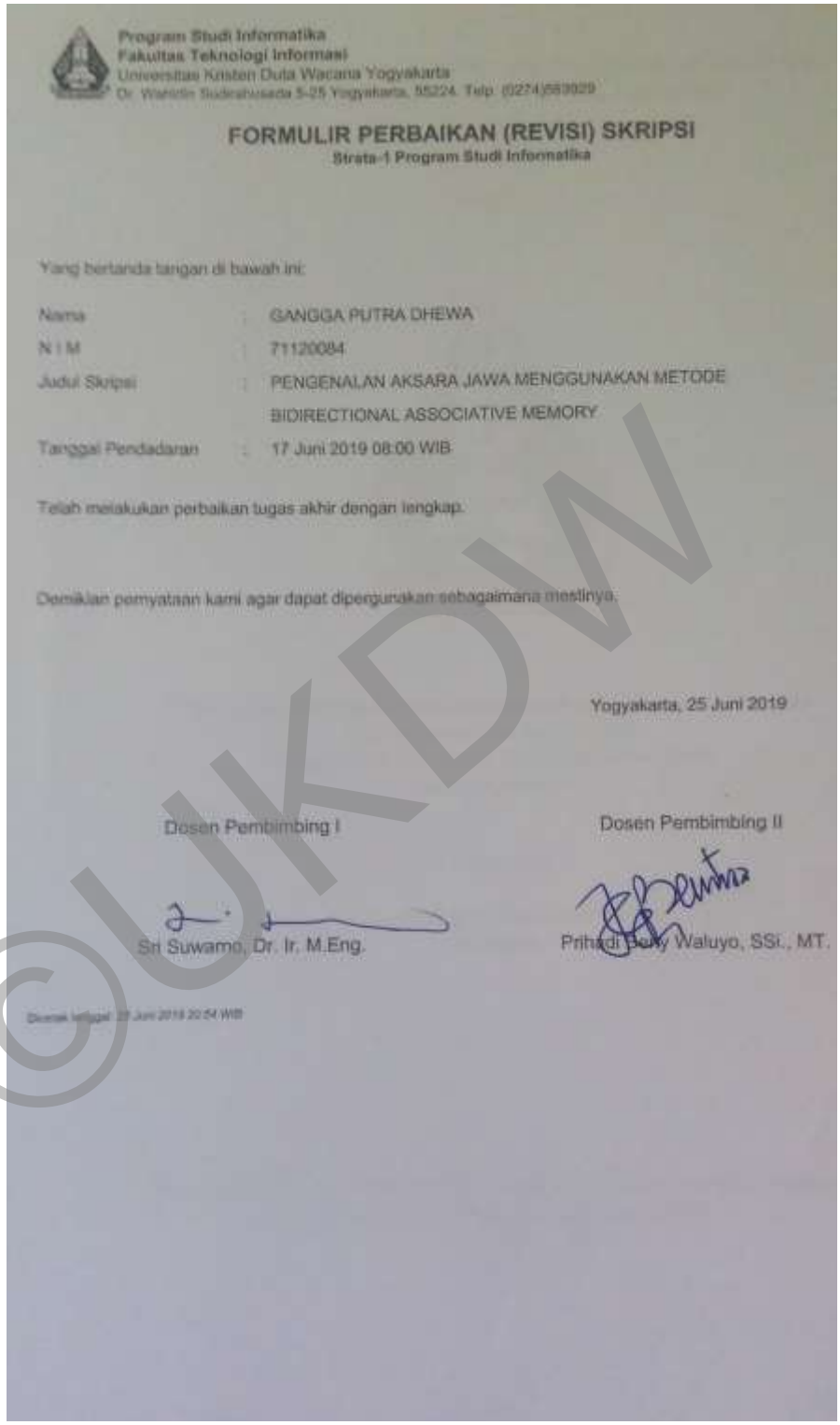
NIM : 71120094/GANGGA PUTRA DHEWA
Judul : **Pengenalan Aksara Jawa Menggunakan Metode Bidirectional Associative Memory**
Dosen Pembimbing II : Ridack Beny Waluyo, SSi., MT

	Tanggal	Paraf
1	21-03-2019	
2	25-03-2019	
3	03-05-2019	
4	16-05-2019	
5	20-05-2019	
6	27-05-2019	
7		
8		

Bab 1 & 2 OK

Final

SCAN FORMULIR REVISI SKRIPSI



Program Studi Informatika
Fakultas Teknologi Informasi
Universitas Kristen Duta Wacana Yogyakarta
Dr. Wahidin Sudirohusodo 5-25 Yogyakarta, 55224. Telp. (0274)563929

FORMULIR PERBAIKAN (REVISI) SKRIPSI
Strata-1 Program Studi Informatika

Yang bertanda tangan di bawah ini:

Nama	:	GANGGA PUTRA DHEWA
N I M	:	71120084
Judul Skripsi	:	PENGENALAN AKSARA JAWA MENGGUNAKAN METODE BIDIRECTIONAL ASSOCIATIVE MEMORY
Tanggal Pendadaran	:	17 Juni 2019 08:00 WIB

Telah melakukan perbaikan tugas akhir dengan lengkap.

Demikian pernyataan kami agar dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 25 Juni 2019

Dosen Pembimbing I

Sri Suwarno, Dr. Ir. M.Eng.

Dosen Pembimbing II

Prihadi Soby Waluyo, SSi., MT.

Disetujui tanggal 27 Juni 2019 20:54 WIB

LISTING PROGRAM

bam.m

```
function varargout = bam(varargin)
% BAM MATLAB code for bam.fig
%      BAM, by itself, creates a new BAM or raises the existing
%      singleton*.
%
%      H = BAM returns the handle to a new BAM or the handle to
%      the existing singleton*.
%
%      BAM('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in BAM.M with the given input
arguments.
%
%      BAM('Property','Value',...) creates a new BAM or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before bam_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to bam_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help bam

% Last Modified by GUIDE v2.5 16-Jun-2019 20:47:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @bam_OpeningFcn, ...
                   'gui_OutputFcn',  @bam_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before bam is made visible.
function bam_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to bam (see VARARGIN)

% Choose default command line output for bam
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes bam wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = bam_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in latih.
function latih_Callback(hObject, eventdata, handles)
% hObject    handle to latih (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
bam_latih();


% --- Executes on button press in input.
function input_Callback(hObject, eventdata, handles)
% hObject    handle to input (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename,fileformat] = uigetfile({'*.*'});

handles.citra = imread([fileformat,filename]);
handles.citra = imresize(handles.citra,[100 100]);

axes(handles.axes1);
```

```
imshow(handles.citra);
guidata(hObject, handles);
msgbox('Citra berhasil dimasukkan');
set(handles.hasil,'String','');



% --- Executes on button press in grayscale.
function grayscale_Callback(hObject, eventdata, handles)
% hObject    handle to grayscale (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
citra = handles.citra;
grayscale = rgb2gray(citra);
axes(handles.axes1);
imshow(grayscale);
handles.citra2 = grayscale;
guidata(hObject, handles);

msgbox('Berhasil');



% --- Executes on button press in threshold.
function threshold_Callback(hObject, eventdata, handles)
% hObject    handle to threshold (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
citra = handles.citra2;
thres = graythresh(citra);
citra = im2bw(citra,thres);
citra = imcomplement(citra);

axes(handles.axes1);
imshow(citra);
handles.citra3 = citra;
guidata(hObject, handles);
msgbox('Berhasil');



% --- Executes on button press in zoning.
function zoning_Callback(hObject, eventdata, handles)
% hObject    handle to zoning (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
global zoning

citra = handles.citra3;
```

```
% thres = graythresh(citra);
% citra = im2bw(citra,thres);

zoning = zoning25(citra);
%handles.citra4 = zoning;
msgbox('Berhasil');


% --- Executes on button press in bam.
function bam_Callback(hObject, eventdata, handles)
% hObject    handle to bam (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
global zoning

aksara = bam_uji(zoning);
hasil = aksara;
set(handles.hasil, 'String',hasil);
msgbox('Berhasil');


function hasil_Callback(hObject, eventdata, handles)
% hObject    handle to hasil (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of hasil as text
%        str2double(get(hObject,'String')) returns contents of
hasil as a double
%hasil = 'Ha';
set(handles.bam, 'String', hasil);



% --- Executes during object creation, after setting all
properties.
function hasil_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hasil (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

bam_latih.m

```
function bam_latih()
    nama = {
        'Ha';'Na';'Ca';'Ra';'Ka';
        'Da';'Ta';'Sa';'Wa';'La';
        'Pa';'Dha';'Ja';'Ya';'Nya';
        'Ma';'Ga';'Ba';'Tha';'Nga';
    };
    target = {
        '1 1 1 1 1';%1
        '-1 1 1 1 -1';%2
        '1 -1 1 1 1';%3
        '1 1 -1 1 1';%4
        '1 1 1 -1 1';%5
        '1 1 1 1 -1';%6
        '-1 -1 1 1 1';%7
        '1 -1 -1 1 1';%8
        '1 1 -1 -1 1';%9
        '1 1 1 -1 -1';%10
        '1 1 -1 1 -1';%11
        '1 -1 1 -1 1';%12
        '-1 1 -1 1 1';%13
        '-1 -1 -1 1 1';%14
        '1 -1 -1 -1 1';%15
        '1 1 -1 -1 -1';%16
        '1 -1 1 -1 -1';%17
        '-1 1 -1 1 -1';%18
        '-1 -1 1 -1 1';%19
        '-1 -1 -1 -1 -1';%20
    };

    file = dir(['latih', '/*.jpg']);
    countFile = length(file);
    for i=1:countFile
        citra = imread(['latih/data_latih_',num2str(i), '.jpg']);
        %imshow(citra);
        citra = rgb2gray(citra); %mengubah rgb ke grayscale
        thres = graythresh(citra);
        citra = im2bw(citra, thres);
        citra = imcomplement(citra);
        citra = imresize(citra,[100 100]);
%         pause(1);
%         imshow(citra);

        zoning{i,1} = zoning25(citra);
        matrix = zoning{i,1};
        [bz, kz] = size(matrix);
        x = reshape(matrix', bz*kz, 1);
        %disp(x);

        for j=1:length(target)
            w_temp{j,1} = x * str2num(target{j,1});
            huruf(j) = str2double(target(j));
```

```
            %disp(nama(j));
            %disp(target(j));
        end

        for m=1:length(w_temp)
            if (m == 1)
                bobot{i,1} = w_temp{m,1}
            else
                bobot{i,1} = bobot{i,1} + w_temp{m,1}
            end
        end
        disp(['Selesai latih data ke-',num2str(i)]);
    end
    bobot = sum(cat(3, bobot{:}), 3);


    msgbox(['Citra data latih sebanyak ', num2str(i),' dilatih,
bobot berhasil disimpan']);

    save('zoning.mat','zoning');
    save('bobot.mat', 'bobot');
    save('bobot_temp.mat', 'w_temp');
    save('nama.mat', 'nama');
    save('target.mat', 'target');

end
```

zoning25.m

```
function zoning = zoning25( citra )
zona_index = 1;
total_zone = 25;
zona_width = 20;
zona_row = 5;
luas = 400;%20x20

for x=1:total_zone
    pixel_putih=0;
    for i=1:zona_width
        for j=1:zona_width
            if zona_index <= 5
                %1,1 - 20,100
                if citra(i,(j+((zona_index-1)*zona_width))) == 1
                    pixel_putih = pixel_putih+1;
                end
            elseif zona_index > 5 && zona_index <=10
                %21,1 - 21,100, zona_index 6-10
                if citra(i+zona_width,(j+(mod((zona_index-
1),5)*zona_width))) == 1
                    pixel_putih = pixel_putih+1;
                end
            elseif zona_index > 10 && zona_index <= 15
```

```
                %41,1 - 41,100, zona_index 10-15
                if citra(i+(zona_width*2),(j+(mod((zona_index-
1),5)*zona_width))) == 1
                    pixel_putih = pixel_putih+1;
                end
            elseif zona_index > 15 && zona_index <= 20
                %61,1 - 61,100, zona_index 16-20
                if citra(i+(zona_width*3),(j+(mod((zona_index-
1),5)*zona_width))) == 1
                    pixel_putih = pixel_putih+1;
                end
            elseif zona_index > 20 && zona_index <= 25
                %81,1 - 81,100, zona_index 21-25
                if citra(i+(zona_width*4),(j+(mod((zona_index-
1),5)*zona_width))) == 1
                    pixel_putih = pixel_putih+1;
                end
            end
        end
    end

    hasil(zona_index)=pixel_putih/luas;
    if hasil(zona_index)<=0.001
        fitur(zona_index)=-1;
    else
        fitur(zona_index)=1;
    end

    zona_index = zona_index + 1;
%     disp(fitur);
    zoning = fitur;
    save('fitur.mat');
    save('hasil.mat');
end
```

bam_uji

```
function [ aksara ] = bam_klasifikasi(zoning)
    if exist('bobot.mat','file')
        load('bobot.mat');
        load('nama.mat');
        load('target.mat');
    else
        msgbox('Hitung bobot terlebih dahulu');
        return;
    end

    disp('hasil zoning');
    disp(zoning);

    %test x ke y
```

```
for b=1:numel(bobot)
    y_in = zoning * bobot;

    disp('y_in here');
    disp(y_in);

    theta = (min(y_in) + max(y_in))/2;

    disp(theta);

    for i=1:numel(y_in)
        if y_in(i) < theta
            y_in(i) = -1;
        elseif y_in(i) > theta
            y_in(i) = 1;
        end
    end

    for i=1:numel(y_in)
        if y_in(i) < theta
            y_in(i) = -1;
        elseif y_in(i) > theta
            y_in(i) = 1;
        end
    end

    disp('target here');
    disp(y_in);

    for k=1:numel(target)
        if isequal(y_in,str2num(target{k}))
            hasil = target{k};
            index = k;
            break
        else
            index = 0;
        end
    end

    x_in = y_in * bobot';
    theta=(min(x_in) + max(x_in))/2;

    for i=1:numel(x_in)
        if x_in(i) < theta
            x_in(i) = -1;
        elseif x_in(i) > theta
            x_in(i) = 1;
        end
    end

    y = y_in;
    if isequal(y,y_in)
        break;
```

```
    end
    y = y_in;
    zoning = x_in;
end

disp(index);

if index == 0
    aksara = 'Tidak dikenali';
else
    aksara = nama{index,1};
end
```