BAB 3
ANALISIS PENGENALAN TULISAN TANGAN

3.1 Analisis Masalah

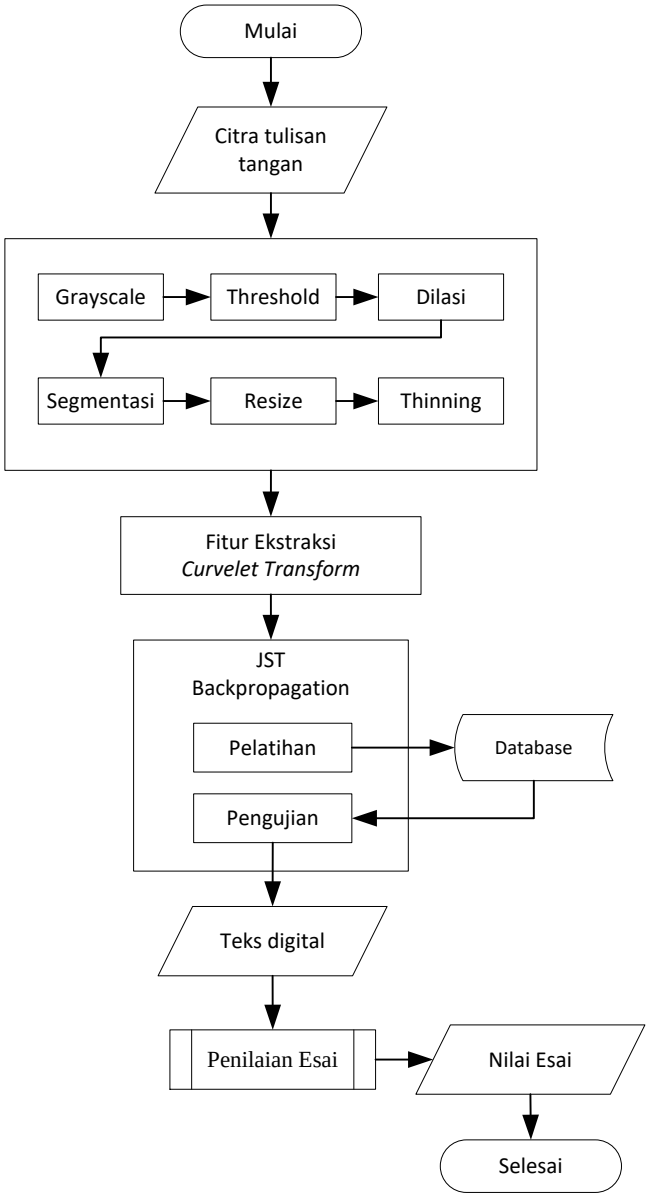
Berdasarkan identifikasi masalah yang sudah disebutkan sebelumnya, maka akan dijelaskan tentang masalah yang ditemukan. Adapun masalah yang ditemukan pada penelitian ini adalah bagaimana cara mengenali tulisan tangan menggunakan metode *Curvelet Transform* dan Jaringan Saraf Tiruan *Backpropagation.* Pada penelitian sebelumnya pengenalan tulisan tangan telah digunakan untuk mengenali huruf alfabet, namun terdapat kendala dalam proses pengenalannya, hal ini disebabkan karena beberapa huruf alfabet memiliki kemiripan bentuk dengan huruf lainnya serta pada penerapannya pengenalan tulisan tangan baru diimplementasikan pada kasus yang secara umum. Pada penelitian ini metode *Curvelet Transform* akan digunakan untuk menghasilkan ciri dari citra tulisan tangan yang selanjutnya akan diklasifikasi menggunakan Jaringan Saraf Tiruan *Backpropagation*. Hasil dari pengenalan tulisan tangan akan digunakan untuk proses penilaian esai. Akurasi pengenalan tulisan tangan yang didapatkan akan mempengaruhi hasil penilaian esai, sehingga perlu diketahui akurasi pengenalannya.

3.2 Analisis Sistem

Dalam analisis sistem ini akan dijabarkan mengenai analisis dan tahapan dalam proses pengenalan tulisan tangan di antaranya analisis data masukan, tahapan *preprocessing* dan tahapan pembangunan model estimasi untuk proses klasifikasi menggunakan Jaringan Saraf Tiruan *Backpropagation.*

3.2.1. Analisis Proses

Proses pengenalan tulisan tangan yang dilakukan dalam penelitian ini dibagi menjadi beberapa bagian di mana setiap proses memiliki peranan masing-masing dalam melakukan pengenalan tulisan tangan. Berikut proses yang dilakukan pada gambar 3.1.



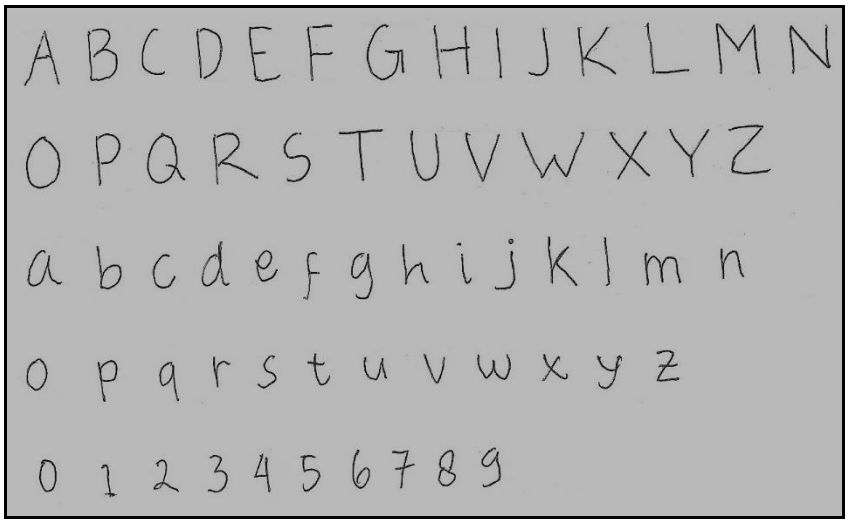
Gambar 3. 1 Arsitektur Umum Sistem

Pada gambar 3.1, proses awal yang dilakukan yaitu memasukkan citra tulisan tangan ke dalam aplikasi. Langkah selanjutnya adalah tahapan *preprocessing* citra yang meliputi *graysacle*, *thresholding*, *thining*, segmentasi dan *resize*. Setelah tahap *preprocessing* selesai maka citra masukan telah siap untuk dilakukan proses ekstraksi ciri, proses ekstraksi ciri berguna untuk mengubah citra ke dalam suatu nilai unik yang merupakan ciri dari citra tersebut.

Nilai ciri yang telah dihasilkan pada tahap ekstraksi kemudian diolah kembali pada tahap klasifikasi dengan metode Jaringan Saraf Tiruan *Backpropagation* baik pelatihan maupun pengujian. Hasil dari pengenalan tulisan tangan didapatkan teks digital yang selanjutnya akan dilakukan proses penilaian esai menggunakan metode Rabin-*karp* untuk mendapatkan nilai ujian.

3.2.2. Data Masukan

Data masukan yang digunakan pada penelitian ini, yaitu citra tulisan tangan dengan format gambar *.jpg*. Citra dapat diperoleh melalui *scan* atau foto dengan resolusi kamera minimal 8 *megapixel*. Citra tulisan tangan yang digunakan menggunakan pulpen dalam kertas HVS dengan *background* putih. Selain itu warna citra juga tidak terlalu gelap agar aplikasi dapat lebih maksimal melakukan pengenalan karakter dari setiap tulisan tangan. Dalam penelitian ini data masukan secara otomatis di *resize* menjadi ukuran 8x1300 *pixel*.



Gambar 3. 2 Citra Input Tulisan Tangan

Gambar 3.2 merupakan contoh citra yang digunakan sebagai data masukan aplikasi. Berikut adalah tabel matriks warna *Red* (R), *Green* (G) dan *Blue* (B) dari contoh citra masukan di atas.

Tabel 3. 1 Matriks Nilai GRB Citra Masukan

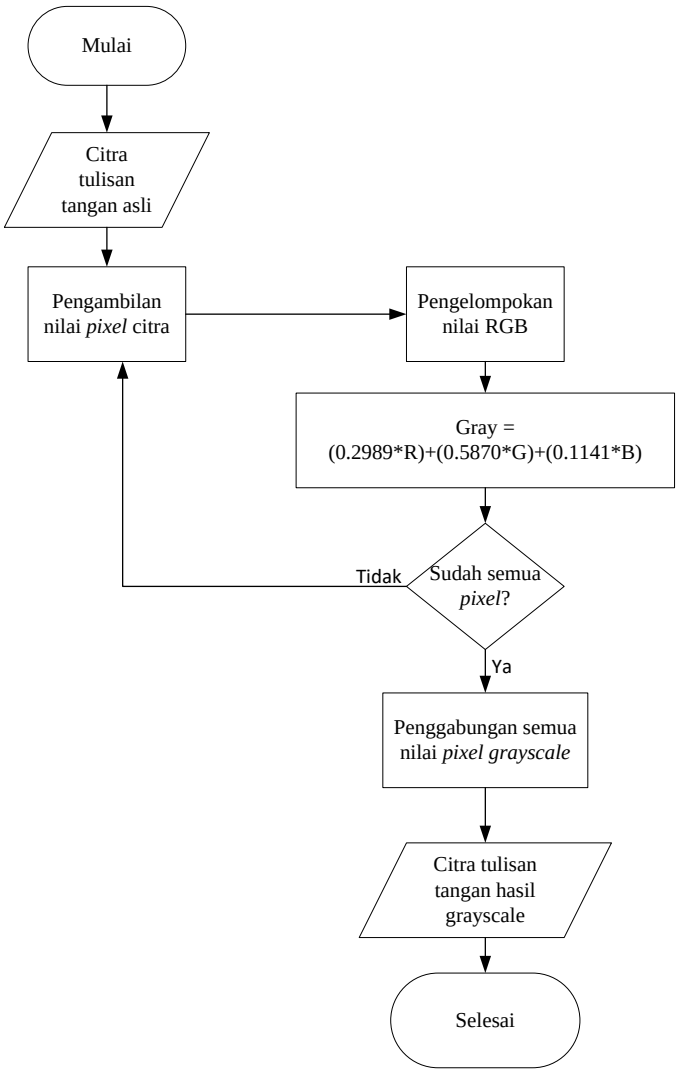
X/Y	0	1	2	3	…	749
0	R=192 G=200 B=204	R=188 G=198 B=198	R=184 G=196 B=197	R=184 G=191 B=194	…	R=172 G=173 B=172
1	R=198 G=202 B=210	R=193 G=199 B=205	R=186 G=197 B=199	R=187 G=193 B=198	…	R=178 G=171 B=174
2	R=192 G=205 B=212	R=193 G=193 B=201	R=186 G=197 B=199	R=181 G=193 B=195	…	R=171 G=175 B=172
3	R=193 G=204 B=212	R=191 G=196 B=206	R=186 G=187 B=189	R=187 G=193 B=208	…	R=172 G=173 B=172
⋮	⋮	⋮	⋮	⋮	⋱	…
1299	R=191 G=212 B=211	R=193 G=199 B=205	R=186 G=191 B=189	R=181 G=193 B=195	…	R=172 G=163 B=169

3.2.3. Preprocessing

Pada proses *preprocessing* data masukan citra akan diolah dalam beberapa tahapan yakni *grayscale*, *Thresholding*, *Thinning*, *Segementasi* dan *Resize*. Proses ini dilakukan agar data masukan citra dapat memenuhi kriteria tertentu sebelum dilakukan proses ekstraksi ciri dan klasifikasi.

1. Grayscale

Mengubah format warna menjadi *grayscale* berfungsi untuk mengecilkan *range* warna dari 0 sampai dengan 255. Proses ini akan memudahkan ketika ingin melakukan *threshold* citra menjadi citra hitam putih. Gambar 3.3 merupakan *flowchart* proses *grayscale.*



Gambar 3. 3 Flowchart Proses Grayscale

Adapun langkah-langkah yang dilakukan sebagai berikut.

1. Warna citra dikelompokan berdasarkan nilai red, green dan blue.
2. Kemudian menggunakan persamaan 2.1, maka akan didapatkan nilai warna grayscale citra.
3. Nilai grayscale yang didapat menggantukan nilai RGB pada setiap pixel.

Misalkan citra pada *pixel* (0,0) mempunyai nilai *Red* =155, *Green* = 135, *Blue* =162, maka berdasarkan persamaan 2.1 menjadi :

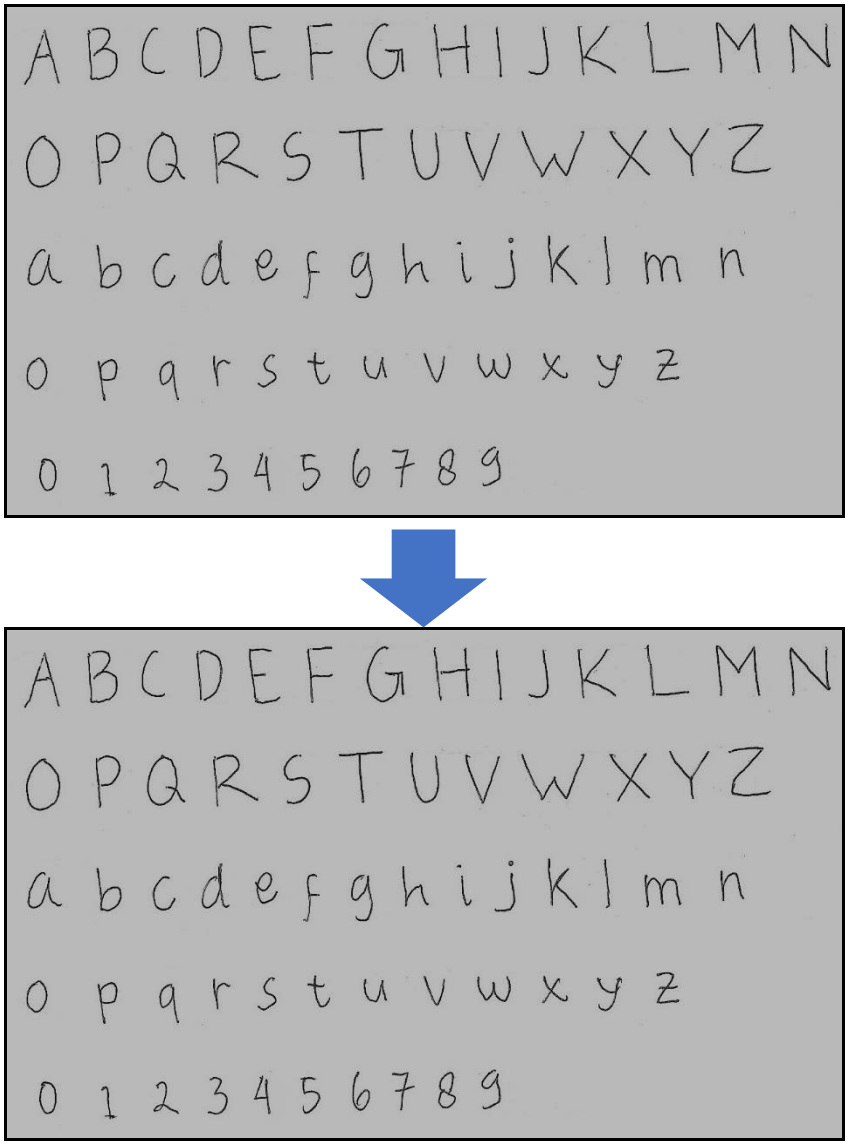
I = (0,2989 * R) + (0,5870 * G) + (0,1141 * B)

= (0,2989 * 155) + (0,5870 * 135) + (0,1141 * 162)

= 46,3295 + 79,245 + 18,4842

= 144

Dari perhitungan di atas, maka *pixel* yang tadinya bernilai *Red* = 155, *Green* = 135, *Blue* = 162 diperbaharui menjadi nilai grayscale = 144. Gambar 3.4 di bawah ini merupakan hasil dari proses *grayscale*.



Gambar 3. 4 Hasil Proses Grayscale

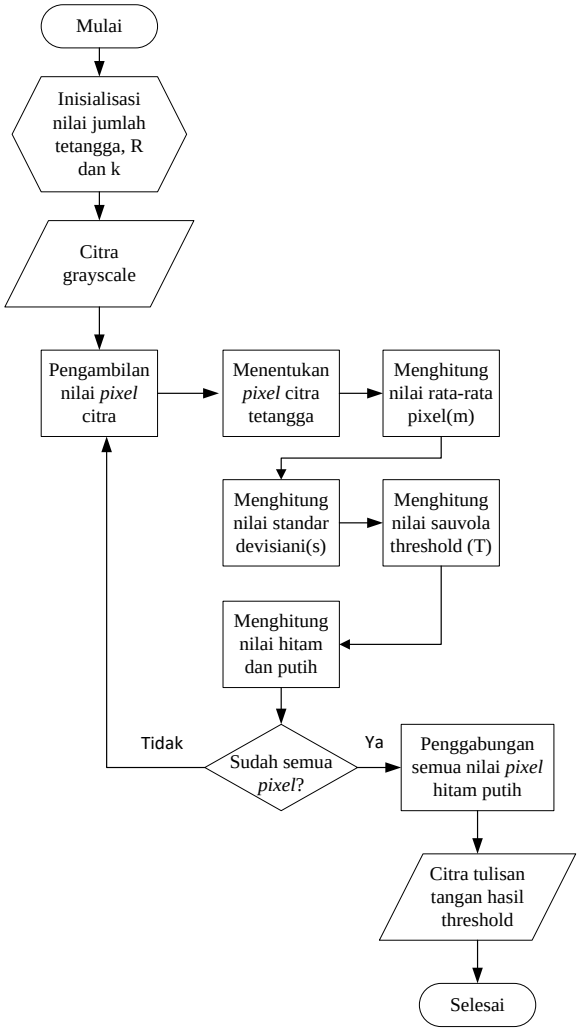
Berikut adalah tabel matriks warna *Red* (R), *Green* (G) dan *Blue*(B) dari hasil *grayscale*.

Tabel 3. 2 Matriks Nilai GRB Citra Masukan

X/Y	0	1	2	3	…	749
0	R=198 G=198 B=198	R=195 G=195 B=195	R=193 G=193 B=193	R=194 G=194 B=194	…	R=172 G=172 B=172
1	R=198 G=198 B=198	R=194 G=194 B=194	R=192 G=192 B=192	R=186 G=186 B=186	…	R=171 G=171 B=171
2	R=198 G=198 B=198	R=193 G=193 B=193	R=198 G=198 B=198	R=189 G=189 B=189	…	R=174 G=174 B=174
3	R=182 G=182 B=182	R=187 G=187 B=187	R=192 G=192 B=192	R=191 G=191 B=191	…	R=178 G=178 B=178
⋮	⋮	⋮	⋮	⋮	⋱	…
1299	R=192 G=192 B=192	R=194 G=194 B=194	R=196 G=196 B=196	R=193 G=193 B=193	…	R=172 G=172 B=172

2. Thresholding

Metode threshold yang digunakan dalam penelitian ini yaitu menggunakan metode Sauvola Threshold. Sauvola merupakan kategori dari metode local threshold, di mana nilai ambang pada setiap pixel tergantung dari jumlah tentangga yang digunakan. Gambar 3.5 merupakan flowchart proses sauvola threshold.



Gambar 3. 5 Flowchart Proses Thresholding

Dalam penelitian ini, parameter yang digunakan sebagai berikut.

1. Jumlah tetangga = 21 pixel. Nilai 21 cocok digunakan karena jika nilai tetangga terlalu besar, maka waktu yang dibutuhkan menjadi lebih lama, atau jika terlalu kecil maka hasi yang didaptkan tidak maksimal.
2. Konstanta R = 128
3. K = 0,3
4. Nilai m(x,y) didapatkan dari persamaan 2.3
5. Nilai s(x,y) didapatkan dari persamaan standar deviasi 2.4

   Dari citra tulisan tangan yang digunakan, akan dicari nilai ambang pada pixel (0,0) dengan nilai grayscale 144. Selanjutnya cari pixel tetangga

mana saja yang sesuai dengan jumlah tentangga yang sudah ditentukan sebelumnya yaitu 21. Kemudian dari hasi pencarian tersebut didapatkan matriks citra seperti tabel 3.3, di mana tetangga pixel (0,0) adalah dari pixel (0,0) sampai dengan pixel (10,10).

Tabel 3. 3 Contoh Matriks Citra Akan di Threshold

x/y	0	1	2	3	4	5	6	7	8	9	10	11	1
0	144	141	144	144	142	142	143	145	149	146	142	144	141
1	146	143	145	148	142	145	143	146	143	145	147	146	143
2	144	140	148	151	146	140	143	143	144	144	140	144	140
3	147	141	145	144	147	144	144	145	149	145	143	147	141
4	144	138	145	145	146	144	149	142	144	141	144	144	138
5	141	151	145	141	141	142	142	150	143	140	144	141	151
6	146	145	148	141	142	142	144	144	145	145	147	146	145
7	144	143	144	139	138	144	142	144	141	140	142	144	143
8	147	141	139	146	140	150	141	148	141	141	142	147	141
9	144	143	139	142	144	145	148	142	146	146	142	144	143
10	144	149	144	143	147	153	130	144	141	142	140	144	149
11	144	141	144	144	142	142	143	145	149	146	142	144	141
12	146	143	145	148	142	145	143	146	143	145	147	146	143

Langkah pertama cari nilai rata-rata m(0,0) dengan persamaan 2.3, sehingga hasilnya seperti berikut.

$$m(0,0) = \frac{144 + 142 + \cdots + 142 + 142}{11 + 11} = 143{,}79$$

Kemudian untuk mencari nilai standar deivasi s(0,0) digunakan persamaan 2.4 sehingga hasilnya sebagai berikut.

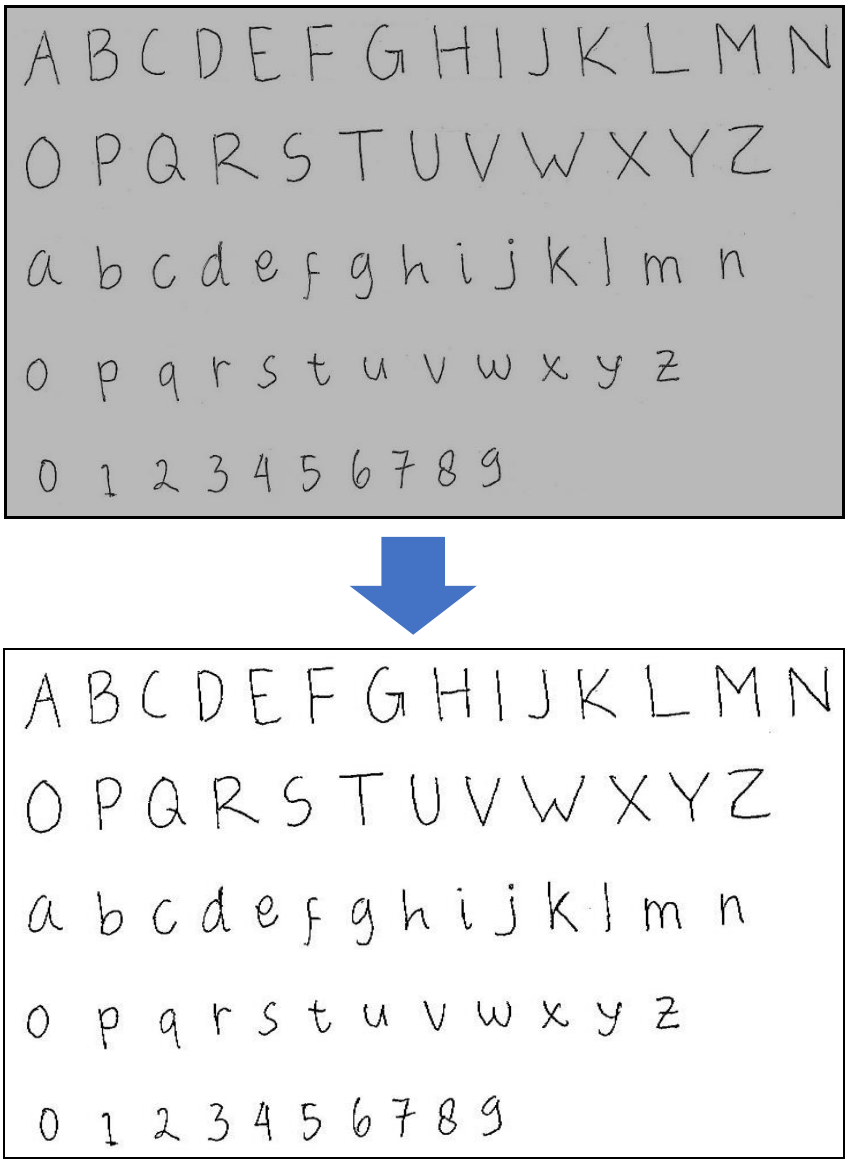
$$m(0,0) = \sqrt{\frac{(1 - 143{,}79)^2 + (1 - 143{,}79)^2 + \cdots + (1 - 143{,}79)^2}{211 + 11}} = 3{,}06$$

Langkah selanjutnya masukan nilai m(0,0) dan s(0,0) ke dalam persamaan 2.2 untuk mendapatkan nilai ambang T(0,0). Sehingga hasilnya menjadi

$$T(0,0) = m(0,0) * (1 + 0{,}3 * \left(\frac{s(0,0)}{128} - 1\right))$$

$$= 143{,}79 * (1 + 0{,}3 * \left(\frac{3{,}06}{128} - 1\right))$$
$$= 101$$

Dari perhitungan di atas didapatkan nilai ambang 101. Langkah selanjutnya langsung dimasukkan dalam persamaan 2.5 sehingga akan didapatkan nilai *pixel* baru. Dari contoh di atas maka *pixel* yang awalnya dengan nilai 144 akan berubah menjadi 255 karena 144 lebih besar dari 101. Berikut adalah gambar citra yang telah di proses menggunakan *sauvola threshold*.



Gambar 3. 6 Hasil Sauvola Threshold

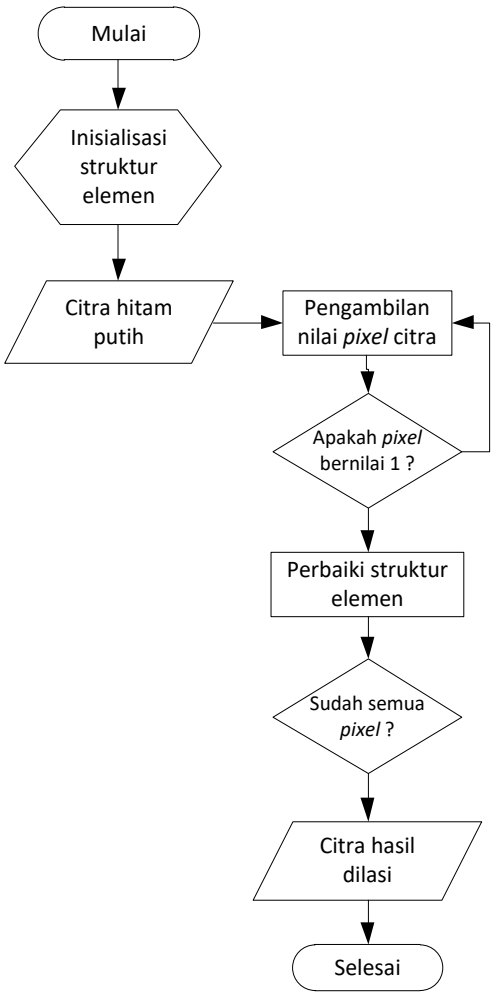
Kemudian tabel berikut adalah matriks warna *Red* (R), *Green* (G) dan *Blue* (B) dari hasil proses *Sauvola Threshold*.

Tabel 3. 4 Matriks Nilai RGB Citra Threshold

X/Y	0	1	2	3	…	749
0	R=255 G=255 B=255	R=255 G=255 B=255	R=255 G=255 B=255	R=255 G=255 B=255	…	R=255 G=255 B=255
1	R=255 G=255 B=255	R=255 G=255 B=255	R=255 G=255 B=255	R=255 G=255 B=255	…	R=255 G=255 B=255
2	R=255 G=255 B=255	R=255 G=255 B=255	R=255 G=255 B=255	R=255 G=255 B=255	…	R=255 G=255 B=255
3	R=255 G=255 B=255	R=255 G=255 B=255	R=255 G=255 B=255	R=255 G=255 B=255	…	R=255 G=255 B=255
⋮	⋮	⋮	⋮	⋮	⋱	…
1299	R=255 G=255 B=255	R=255 G=255 B=255	R=255 G=255 B=255	R=255 G=255 B=255	…	R=255 G=255 B=255

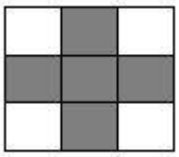
3. Operasi Dilasi

Proses *dilasi* berfungsi untuk memperbaiki pixe pada citra yang pecah dengan cara membandingkan setiap piksel citra input dengan nilai pusat SE. Gambar 3.7 berikut merupakan *flowchart* proses *dilasi*.



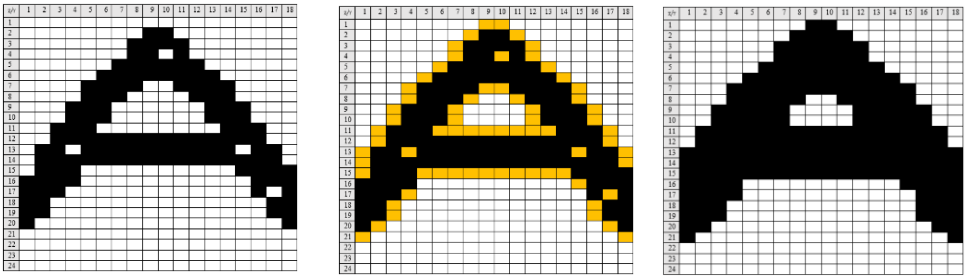
Gambar 3. 7 Flowchar Proses Dilasi

Proses ini dilakukan dengan dengan menggunakan Struktur Elemen (SE) 3x3. Struktur elemen tersebut akan digunakan dalam proses *dilasi* terhadap citra masukan. Citra yang dikenai operasi *dilasi* adalah citra yang sudah melalui proses *threshold*.



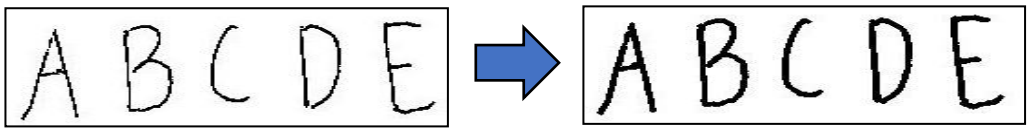
Gambar 3. 8 Struktur Elemen

Proses *dilasi* dilakukan dengan menandai *pixel* objek dengan struktur elemen yang digunakan. Misalkan objek pada *pixel* (9,2) dikenakan operasi dilasi. Sesuai dengan struktur elemen yang digunakan, maka *pixel* (9,1), (8,2), (9,3) dan (10,2) berubah menjadi objek. Namun karena *pixel* (9,3) dan (10,2) merupakan objek, maka yang hanya berubah menjadi objek yaitu *pixel* (9,1) dan (8,2). Begitupun proses selanjutnya sampai semua *pixel* citra ditelusuri.



Gambar 3. 9 Proses Dilasi Citra

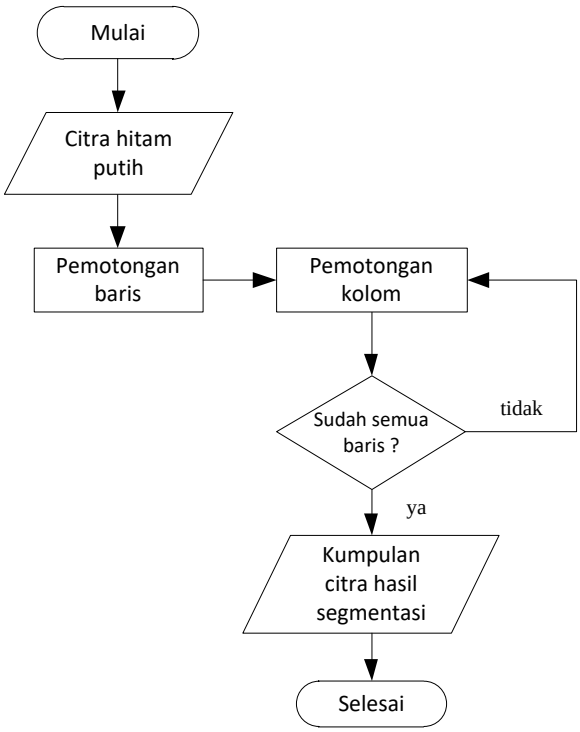
Dari proses *dilasi* yang sudah dilakukan terhadap citra input, Jika diperhatikan terlihat bahwa awalnya citra memiliki bercak putih di bagian tengah. Setelah dilakukan proses *dilasi* terlihat bintik putih tertutup oleh warna *foreground* atau objek.



Gambar 3. 10 Dilasi Citra Inputan

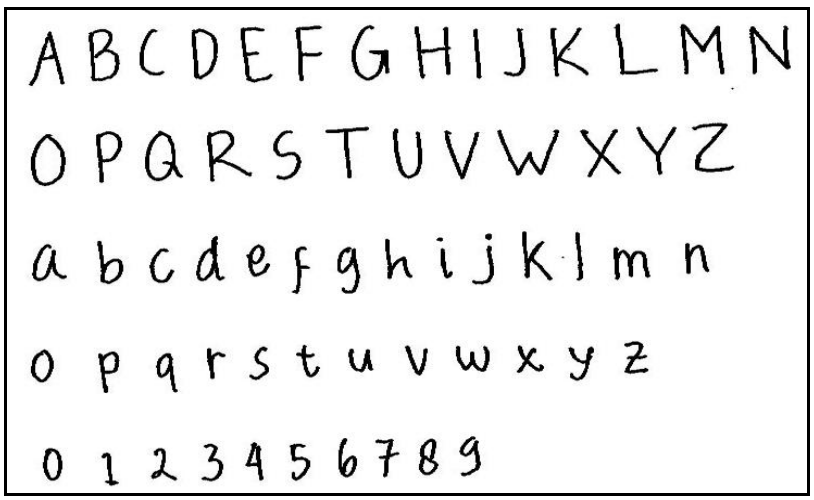
4. Segmentasi

Pada proses ini, input yang digunakan yaitu citra hitam putih hasil *sauvola threshold* dan *dilasi*. Selanjutnya akan dilakukan pemotongan untuk mendapatkan citra huruf tulisan tangan. Berikut adalah *flowchart* proses segmentasi dalam gambar 3.11.



Gambar 3. 11 Flowchart Porses Segmentasi

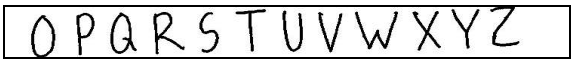
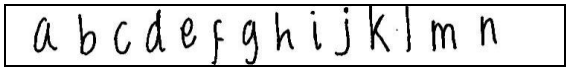
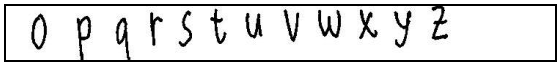
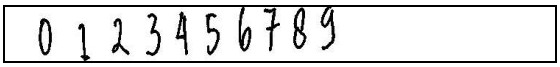
Pemotongan dilakukan untuk setiap baris (*vertical*) pada citra input terlebih dahulu, kemudian melakukan pemotongan setiap kolom (*horizontal*) pada setiap citra hasil pemotongan secara baris. Gambar 3.12 merupakan citra tulisan tangan yang sudah melalui proses *threshold* dan *dilasi* kemudian akan dilakukan proses segmentasi.



Gambar 3. 12 Citra untuk di segmentasi

Pemotongan untuk setiap baris (*vertical*) dilakukan dengan menelusuri *pixel* citra dimulai dari *pixel* baris ke-0. Penelusuran terus dilakukan sampai menemukan *pixel* objek, kemudian tandai sebagai label awal pemotongan. Selanjutnya lakukan penelusuran kembali sampai dalam satu baris *pixel* citra tidak ditemukan *pixel* objek, kemudian tandai sebagai label akhir pemotongan. Label awal dan label akhir ini yang digunakan sebagai acuan untuk memotong citra setiap baris (*vertical*). Lakukan hal yang sama untuk pemotongan baris selanjutnya. Untuk pemotogan setiap kolom (*horizontal*) sama seperti pemotongan baris, hanya saja penelusuran citra dimulai dari *pixel* kolom ke-0. Berikut adalah tabel hasil segmentasi citra tulisan tangan.

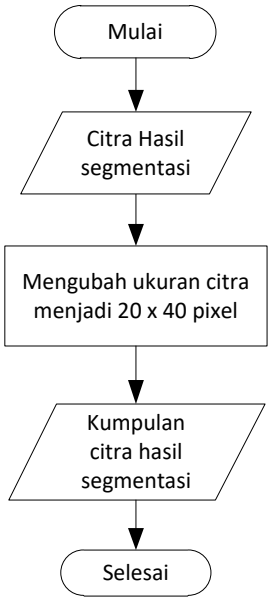
Tabel 3. 5 Segmentasi Citra

No.	Pemotongan Baris	Pemotongan Kolom
1		
2		
3		
4		
5		

Dari tabel 3.5 proses pemotongan dilakukan terhadap baris terlebih dahulu, sehingga didapatkan 5 citra. Setelah itu, setiap citra hasil pemotongan baris dilakukan pemotongan kolom sehingga didapatkan karakter-karakter yang sudah tersegmentasi. Pada proses segmentasi fitur spasi tidak digunakan sehingga proses segmentasi hanya dapat membaca piksel citra yang berwarna hitam. Dalam contoh ini, terdapat 62 citra karakter yang berhasil di segmentasi.

5. Resize

Resize adalah proses mengubah ukuran suatu citra menjadi lebih besar atau kecil dari ukuran citra awal dengan ukuran yang telah ditetapkan sebelumnya. Dalam penelitian ini proses *resize* menggunakan *class Bitmap* yang terdapat pada aplikasi Visual Studio 2015 dengan bahasa pemrograman C#. Berikut adalah *flowchart* proses *resize* pada gambar 3.13.



Gambar 3. 13 Flowchart Porses Resize

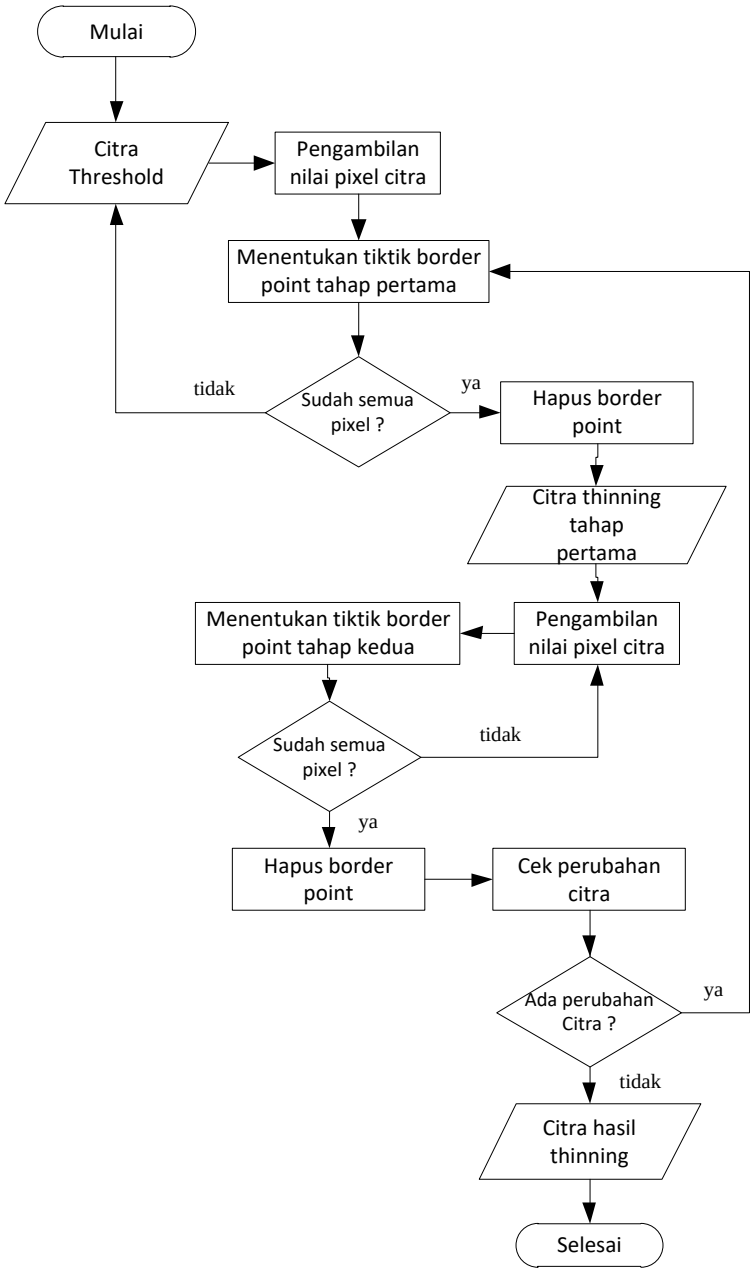
Proses ini dilakukan agar input yang di proses pada saat metode klasifikasi seragam atau konsisten. Dalam penelitian ini karakter-karakter yang sudah tersegmen pada tahap segmentasi di-*resize* menjadi ukuran 20 x 40 *pixel*, ukuran tersebut digunakan berdasarkan penilitian yang telah dilakukan sebelumnya agar ukuran setiap segementasi bernilai sama [5]. Gambar 3.14 merupakan beberapa contoh hasil dari proses *resize* dalam *pixel*.



Gambar 3. 14 Resize citra inputan

6. Thinning

Metode *thinning* yang digunakan dalam penelitian ini yaitu menggunakan metode *Zhang-Suen.* *Zhang-Suen* terdiri dari beberapa penelusuran, di mana setiap penelusurannya terdiri dari 2 langkah dasar yang diaplikasikan terhadap titik yang piksel nya bernilai 1. Gambar 3.15 merupakan *flowchart* proses *thinning Zhang-Suen*.



Gambar 3. 15 Flowchart Proses Thinning

Setiap iterasi dari metode ini terdiri dari dua sub-iterasi berurutan yang dilakukan terhadap *contour points* dari wilayah citra. *Contour point* adalah setiap pixel dengan nilai 1 dan memiliki setidaknya satu *8-neighbor* yang memiliki nilai 0. Misalkan diambil P1 sebagi contour point pada gambar 3.16.

P9	P2	P3			
P8	P1	P4			
P7	P6	P5			

Gambar 3. 16 Contoh Data Thinning

Langkah pertama yang dilakukan adalah menandai contour point p untuk dihapus jika semua kondisi pertama terpenuhi:

Tabel 3. 6 Kondisi Thinning Tahap Pertama

Kondisi	Hasil	Status
2 <= N(P1) <=6	2 <= 4 <= 6	Terpenuhi
S(P1) = 1	1 = 1	Terpenuhi
P2 * P4 * P6 = 0	0 * 1 *1 = 0	Terpenuhi
P4 * P6 * P8 = 0	1 * 1 * 0 = 0	Terpenuhi

Lakukan langkah pertama untuk setiap contour point yang terdapat pada setiap pixel citra. Kemudian hapus setiap countour point yang telah di tandai dengan merubah nilainya menjadi 0. Gambar 3.17 Merupakan pixel citra yang telah di proses pada tahap pertama.

	0			0	
				0	
	0	0	0	0	

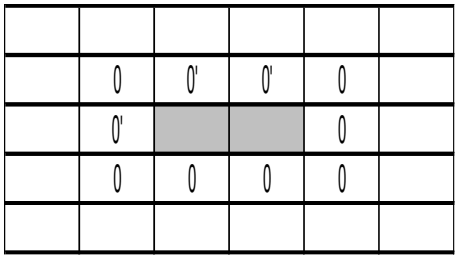
Gambar 3. 17 Penghapusan Contour Point

Langka kedua adalah menandai kembali contour point pada citra yang telah di proses pada tahap pertama. Kemudian hapus countour point yang telah ditandai dengan merubah nilai nya menjadi 0 apabila kondisi kedua terpenuhi.

Tabel 3. 7 Kondisi Thinning Tahap Pertama

Kondisi	Hasil	Status
2 <= N(P1) <=6	2 <= 4 <= 6	Terpenuhi
S(P1) = 1	1 = 1	Terpenuhi
P2 * P4 * P8 = 0	0 * 1 *1 = 0	Terpenuhi
P2 * P6 * P8 = 0	1 * 1 * 0 = 0	Terpenuhi

Langkah pertama dan kedua terus dilakukan hingga tidak terdapat lagi contour point yang harus dihapus. Dapat dilihat pada gambar 3.18 Yang merupakan hasil dari proses sebelumnya tidak ditemukan lagi contour point yang dapat ditandai.



Gambar 3. 18 Hasil Akhir Thinning

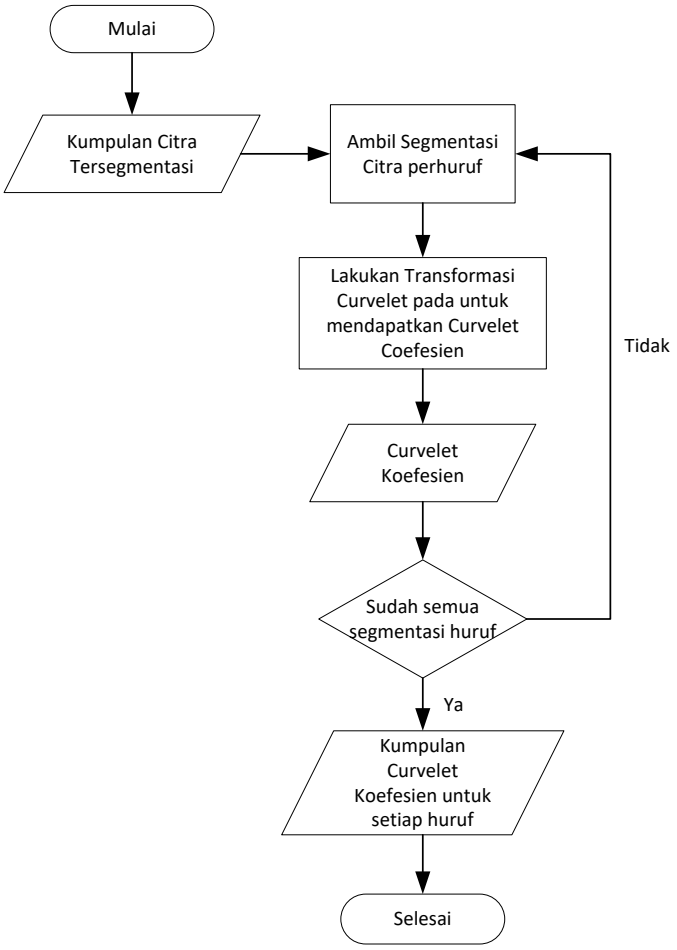
Proses *thinning* dilakukan terhadap setiap citra yang telah di segmentasi sebelumnya, terlihat bahwa awalnya citra huruf cukup tebal. Setelah dilakukan proses *thinning* terlihat citra huruf menipis dengan ketebalan 1 *pixel*.



Gambar 3. 19 Hasil Thinning Citra Inputan

3.2.4. Fitur Ekstraksi dengan Curvelet

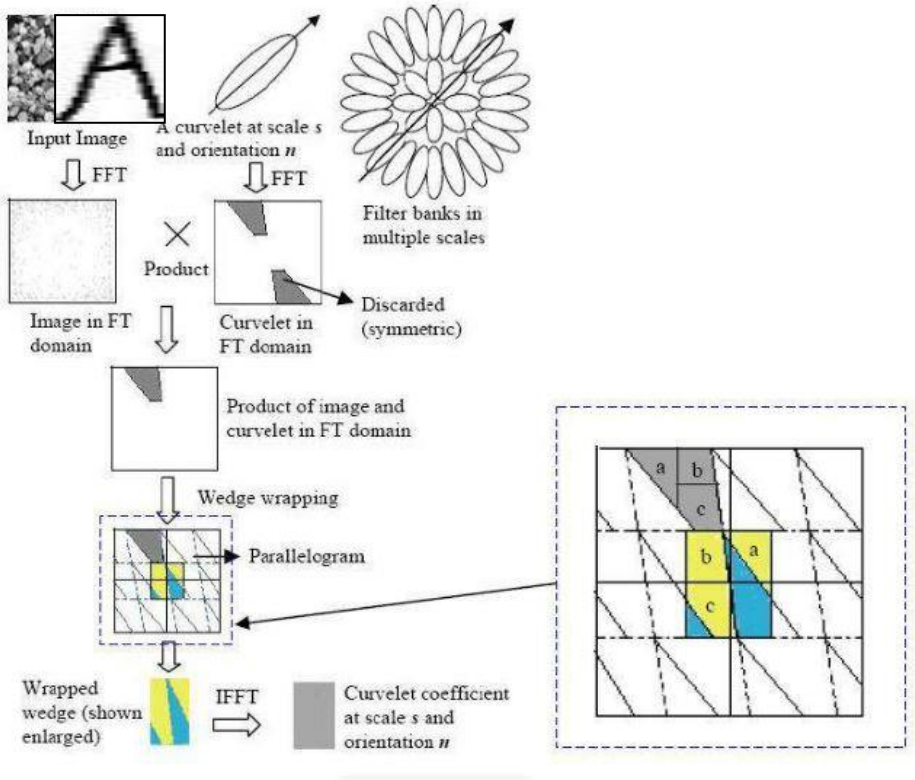
Pada tahapan ini akan dilakukan Ekstraksi ciri terhadapa data citra yang telah disegmentasi pada tahapan sebelumnya. Ciri dari setiap citra yang telah disegementasi didapatkan dengan cara melakukan transformasi citra menjadi suatu nilai diskrit yang mengandung ciri unik dari citra tersebut. Berikut adalah *flowchart* proses *resize* pada gambar 3.20.



Gambar 3. 20 Flowchart Transformasi Curvelet

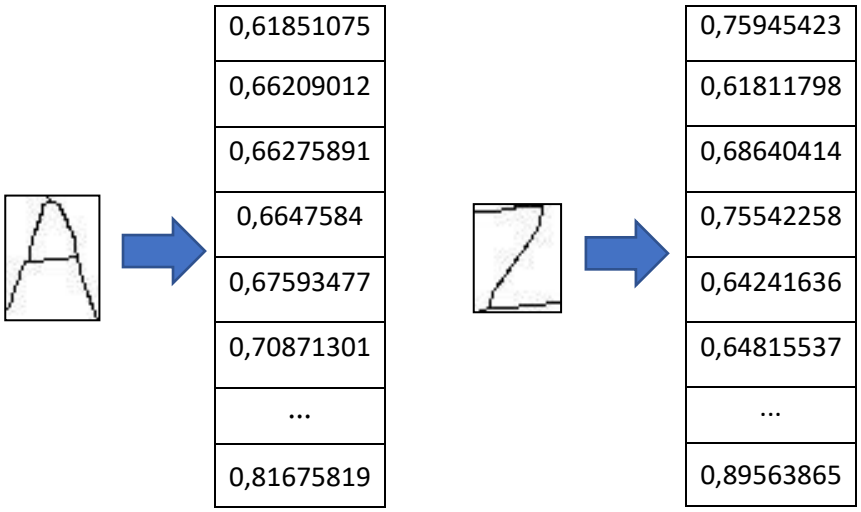
Untuk mendapatkan ciri dari citra yang dicari maka dilakukan transformasi Curvelet digital pada 2D-Cartesian grid $f[m,n], < 0 <= m < M, 0 < n < N$ untuk setiap citra yang telah di segmentasi.

$$C^2(a,b,\theta) = \sum_{\substack{0<m<M \\ 0<n<N}} \mathrm{f[m,n]}\varphi^n ab\theta[m,n]$$



Gambar 3. 21 Transformasi Curvelet Via USFFT

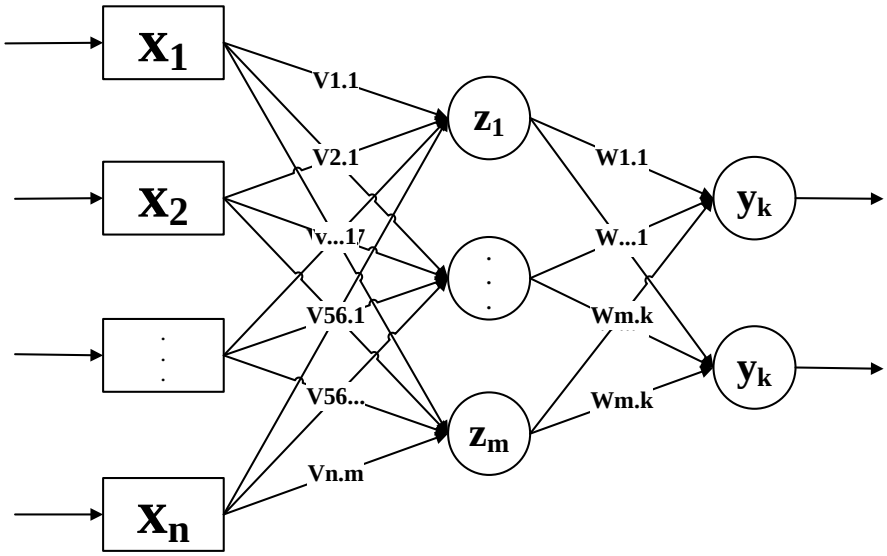
Setelah dilakukan Fast Discrete Curvelet Transform maka akan dihasilkan matriks koefesien Curvelet sebanyak 91 nilai ciri yang mewakili setiap segementasi citra. Pada penilitian ini nilai koefesien curvelet dihitung menggunakan tools curvelab yang disediakan oleh www.curvelet.org. Matriks koefesien tersebut akan dijadikan vektor fitur yang akan digunakan pada tahapan klasifikasi selnjutnya. Berikut adalah koefesien yang dihasilkan dari citra segmentasi A dan Z pada Gambar 3.22.



Gambar 3. 22 Hasil Ekstraksi Ciri Curvelet

3.2.5. Model Estimasi

Pada tahapan ini model estimasi akan dibangun menggunakan Jaringan Saraf Tiruan yang melibatkan tahapan pelatihan dan pengujian menggunakan algoritma backpropagation. Untuk jumlah neuron pada arsitektur backpropagation yang digunakan pada penelitian ini dapat dilihat pada Gambar 3.23



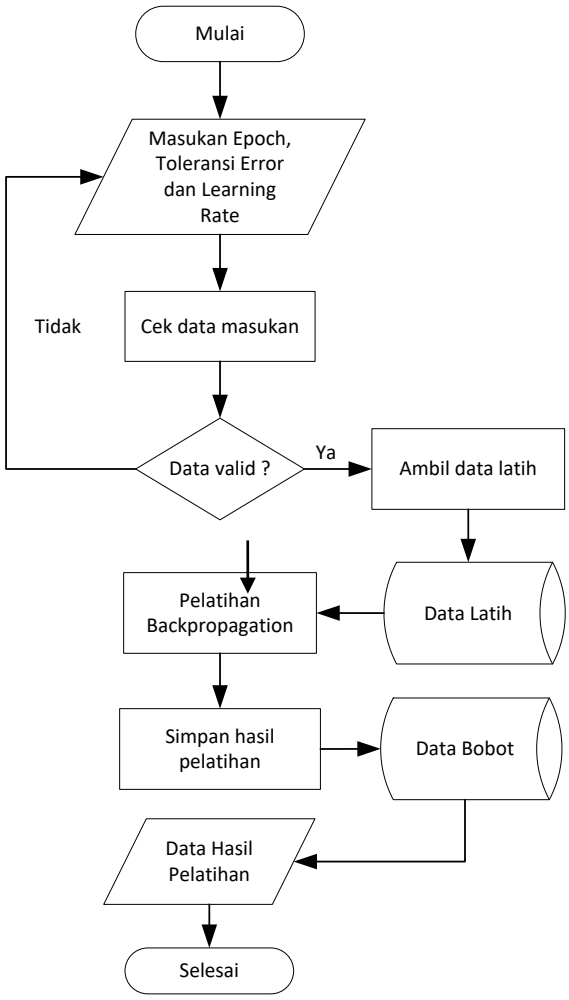
Gambar 3. 23 Jaringan Model Estimasi

Pada gambar 3.23 terdapat *neuron input layer* (X) dengan jumlah 91 yang merupakan hasil dari proses ekstraksi ciri per segmen, untuk penentuan *neuron*

pada *hidden layer* (Z) yaitu 250 *neuron*, ini didasarkan pada penelitian jumlah terhadap *neuron hidden layer*, neuron pada *output layer* (Y) berjumlah 62 *neuron* yang merupakan kelas dari huruf [A-Z], [a-z] dan [0-9]. Diantara *neuron input, hidden dan output* terdapat bobot penghubung yaitu bobot antara *input*-hidden layer (V) dan hidden-output (W) serta bobot biasnya (V_0) dan (W_0).

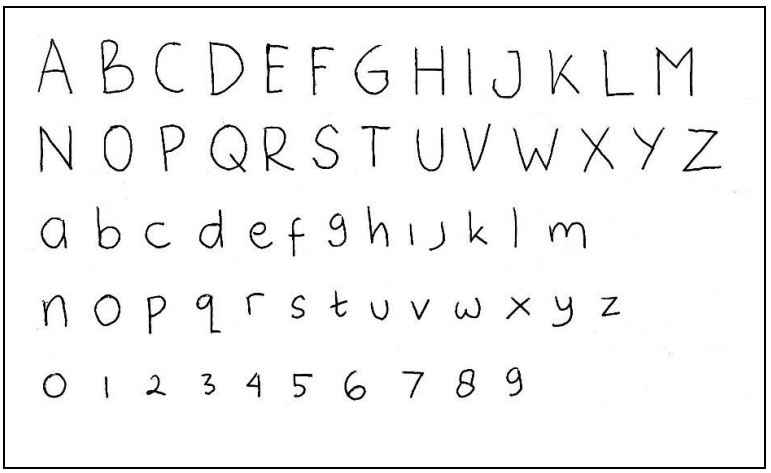
1. Pelatihan *Backpropagation*

Tahap pelatihan dengan menggunakan algoritma *backpropagation* menggunakan proses *feedforward* ,*backward* dan *update* bobot. Bobot yang dihasilkan pada tahap pelatihan akan digunakan untuk proses pengujian. Berikut merupakan flowchart pelatihan *backpropagation* pada Gambar 3.24.



Gambar 3. 24 Flowchart Pelatihan Backpropagation

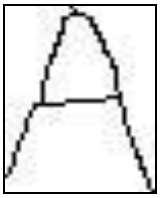
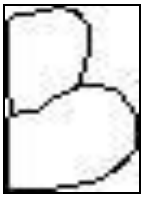
Jumlah data latih yang akan digunakan pada proses pelatihan adalah 50 data latih dimana pada setiap data latih terdiri dari huruf A-Z, a-z dan angka 0-9 yang dapat dilihat pada Gambar 3.25.

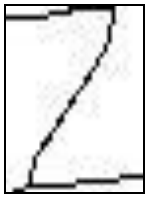


Gambar 3. 25 Contoh Data Latih

Koefesien dari citra latih yang telah didapatkan melalui tahapan *preprocesing* dan ekstraksi ciri akan dijadikan sebagai nilai input sebanyak 91 nilai koefesien curvelet dan kelas yang digunakan berjumlah 62 yaitu kelas huruf A-Z, a-z dan angka 0-9 yang dapat dilihat sebagai matriks pada table 3.8 Dan tabel 3.9.

Tabel 3. 8 Data Latih

Huruf	No	Koefesien
	1	0,87883370
	2	0,96103865
	. ….	
	91	0,82007649
	1	0,83285584
	2	0,61426638
	….	

	91	0,79133522
....		
	1	0,94655826
	2	0,90759856
	....	
	91	0,91657994

Tabel 3. 9 Data Target

Huruf	Target	Biner
A	1	10000000000000000000000000000000000000000000000000000000000000000
B	2	01000000000000000000000000000000000000000000000000000000000000000
C	3	00100000000000000000000000000000000000000000000000000000000000000
D	4	00010000000000000000000000000000000000000000000000000000000000000
E	5	00001000000000000000000000000000000000000000000000000000000000000
F	6	00000100000000000000000000000000000000000000000000000000000000000
G	7	00000010000000000000000000000000000000000000000000000000000000000
H	8	00000001000000000000000000000000000000000000000000000000000000000
I	9	00000000100000000000000000000000000000000000000000000000000000000
J	10	00000000010000000000000000000000000000000000000000000000000000000
K	11	00000000001000000000000000000000000000000000000000000000000000000
L	12	00000000000100000000000000000000000000000000000000000000000000000
M	13	00000000000010000000000000000000000000000000000000000000000000000
N	14	00000000000001000000000000000000000000000000000000000000000000000
O	15	00000000000000100000000000000000000000000000000000000000000000000
P	16	00000000000000010000000000000000000000000000000000000000000000000
Q	17	00000000000000001000000000000000000000000000000000000000000000000
R	18	00000000000000000100000000000000000000000000000000000000000000000

S	19	00000000000000000010000000000000000000000000000000000000000000000000
T	20	00000000000000000001000000000000000000000000000000000000000000000000
U	21	00000000000000000000100000000000000000000000000000000000000000000000
V	22	00000000000000000000010000000000000000000000000000000000000000000000
W	23	00000000000000000000001000000000000000000000000000000000000000000000
X	24	00000000000000000000000100000000000000000000000000000000000000000000
Y	25	00000000000000000000000010000000000000000000000000000000000000000000
Z	26	00000000000000000000000001000000000000000000000000000000000000000000
a	27	00000000000000000000000000100000000000000000000000000000000000000000
b	28	00000000000000000000000000010000000000000000000000000000000000000000
c	29	00000000000000000000000000001000000000000000000000000000000000000000
d	30	00000000000000000000000000000100000000000000000000000000000000000000
e	31	00000000000000000000000000000010000000000000000000000000000000000000
f	32	00000000000000000000000000000001000000000000000000000000000000000000
g	33	00000000000000000000000000000000100000000000000000000000000000000000
h	34	00000000000000000000000000000000010000000000000000000000000000000000
i	35	00000000000000000000000000000000001000000000000000000000000000000000
j	36	00000000000000000000000000000000000100000000000000000000000000000000
k	37	00000000000000000000000000000000000010000000000000000000000000000000
l	38	00000000000000000000000000000000000001000000000000000000000000000000
m	39	00000000000000000000000000000000000000100000000000000000000000000000
n	40	00000000000000000000000000000000000000010000000000000000000000000000
o	41	00000000000000000000000000000000000000001000000000000000000000000000
p	42	00000000000000000000000000000000000000000100000000000000000000000000
q	43	00000000000000000000000000000000000000000010000000000000000000000000
r	44	00000000000000000000000000000000000000000001000000000000000000000000
s	45	00000000000000000000000000000000000000000000100000000000000000000000
t	46	00000000000000000000000000000000000000000000010000000000000000000000
u	47	00000000000000000000000000000000000000000000001000000000000000000000
v	48	00000000000000000000000000000000000000000000000100000000000000000000
w	49	00000000000000000000000000000000000000000000000010000000000000000000
x	50	00000000000000000000000000000000000000000000000001000000000000000000
y	51	00000000000000000000000000000000000000000000000000100000000000000000

z	52	00000000000000000000000000000000000000000000000000010000000000
1	53	00000000000000000000000000000000000000000000000001000000000
2	54	00000000000000000000000000000000000000000000000100000000
3	55	00000000000000000000000000000000000000000000010000000
4	56	00000000000000000000000000000000000000000001000000
5	57	00000000000000000000000000000000000000000000100000
6	58	00000000000000000000000000000000000000000000010000
7	59	00000000000000000000000000000000000000000000001000
8	60	00000000000000000000000000000000000000000000000100
9	61	00000000000000000000000000000000000000000000000010
0	62	00000000000000000000000000000000000000000000000001

a. Inisialisasi

Parameter yang digunakan untuk pelatihan yaitu jumlah maksimum *epoch*, *error minimum*, *momentum* dan laju pembelajaran (*learning rate*). Selanjutnya proses pemberian nilai pada bobot-bobot dengan rentang nilai yang sangat kecil antara 0.001 sampai 0.0001 secara acak yang dapat dilihat sebagai matriks pada Gambar 3.26 dan Gambar 3.27.

	Z_1	Z_2	Z_3	...	Z_{250}
X_1	0.000753	0.000835	0.000791	...	0.000271
X_2	0.000156	0.000199	0.000759	...	0.000803
⋮	⋮	⋮	⋮	⋮	⋮
X_{91}	0.000939	0.000140	0.000591	...	0.00030
Bias	0.000652	0.000846	0.000151	...	0.00083

Gambar 3. 26 Bobot Input Awal

	Y_1	Y_2	Y_3	...	Y_{62}
Z_1	0.000907	0.000842	0.000149	...	0.000339
Z_2	0.000791	0.000347	0.000525	...	0.000385
⋮	⋮	⋮	⋮	⋮	⋮
Z_{250}	0.000307	0.000779	0.000512	...	0.000823
Bias	0.000541	0.000914	0.000532	...	0.000515

Gambar 3. 27 Bobot Hidden Awal

b. Proses *feedforward*

Tabel 3. 10 Proses Feedforward

Proses Feedforward
Langkah 1: Setiap *unit input* (X_i, i = 1, 2, 3, …, 91) mengirimkan sinyal *input* yang berada pada *hidden layer*.
Langkah 2: Hitung keluaran di *hidden layer* menggunakan persamaan (2.7). Z_in_1 = 0.000652 + (0, 87883370 x 0. 000753) + (0, 96103865 x 0.000156) + … + (0,5956 x 0. 000939) = 0.04240 Z_in_2 = 0.000846 + (0, 87883370 x 0. 000835) + (0, 96103865 x 0. 000199) + … + (0,5956 x 0.000591) = 0.04264 . . . Z_in_{250} = 0.00083 + (0, 87883370 x 0.000271) + (0, 96103865 x 0.000803) + … + (0,5956 x 0.00030) = 0.04855
Langkah 3:

Aktifkan nilai keluaran di *hidden layer* menggunakan persamaan (2.8).

$$Z_1 = \frac{1}{1 + e^{-0.04240}} = 0.0423$$

$$Z_2 = \frac{1}{1 + e^{-0.04264}} = 0.0426$$

$$\vdots$$

$$Z_{250} = \frac{1}{1 + e^{-0.04855}} = 0.0505$$

Langkah 4:

Hitung keluaran di *output layer* menggunakan persamaan (2.9).

Y_in_1 = 0.4992+ (0.0423x -0.2895) + (0.0426x -0.1059) + … + (0.0505 x -0.4218) = 0.00662

Y_in_2 = 0.4600 + (0.0423x -0.0677) + (0.0426 x -0.0946) + … + (0.0505x -0.4507) = 0.00694

$$\vdots$$

Y_in_{62} = 0.4600 + (0.0423x 0.000842) + (0.0426x 0.000842) + … + (0.0505x -0.000339 = 0.00687

Langkah 5:

Aktifkan hasil di *output layer* menggunakan Persamaan (2.10).

$$Y_1 = \frac{1}{1 + e^{-0.00662}} = 0.6442$$

$$Y_2 = \frac{1}{1 + e^{-0.00694}} = 0.3174$$

$$\vdots$$

$$Y_{62} = \frac{1}{1 + e^{-0.00687}} = 0.3174$$

c. Proses *Backward*

Pada tahap *backward* merupakan proses menghitung informasi kesalahan dan koreksi bobot yang akan digunakan pada tahap *update* bobot. Proses lengkap *backward* dijelaskan sebagai berikut.

Tabel 3. 11 Proses Backward

Proses *Backward*
Langkah 1: Menghitung informasi kesalahan di *output layer* menggunakan Persamaan (2.11). δ_1 = (1-0.6442) x (0.6442) x ... x (1-0.6442) = 0.0816 δ_2 = (0-0.3174) x (0.3174) x ... x (1-0.3174) = -0.0688 . . . δ_2 = (0-0.2174) x (0.2174) x... x (1-0.2154) = 0.0518
Langkah 2: Menghitung koreksi bobot dari *hidden layer* ke *output layer* (dengan *learning rate* 0.2) menggunakan Persamaan (2.12). $\Delta W_{1.1}$ = 0.2 x 0.2857 x ... x 0.0816 = 0.0047 $\Delta W_{1.2}$ = 0.2 x 0.9851 x ... x -0.0688 = −0.0136 $\Delta W_{1.b2}$ = 0.2 x 0.2174x ... x -0. 2154= −0.0136 . . . $\Delta W_{250.62}$ = 0.2 x 0.0012 x ... x -0.0688 = −0.001
Langkah 3: Menghitung koreksi bobot bias dari *hidden layer* ke *output layer* (dengan *learning rate* 0.2) menggunakan Persamaan (2.13). $\Delta W_{0.1}$ = 0.2 x 0.0816 = 0.0163 . . . $\Delta W_{0.62}$ = 0.2 x -0.0688 = −0.0138

Langkah 4:

Menghitung nilai delta *input* untuk informasi kesalahan di *hidden layer* menggunakan Persamaan (2.14).

$$\delta_in_1 = (0.0816 \text{ x } -0.2895) + (-0.0688 \text{ x } -0.0677) = -0.0190$$

$$\delta_in_2 = (0.0816 \text{ x } -0.1059) + (-0.0688 \text{ x } -0.0946) = -0.0021$$

.
.
.

$$\delta_in_{250} = (0.0816 \text{ x } -0.4218) + (-0.0688 \text{ x } -0.4507) = -0.0034$$

Langkah 5:

Menghitung informasi kesalahan di *hidden layer* menggunakan Persamaan (2.16).

$$\delta_1 = -0.0190 \text{ x } -0.9161 \; x \; (1-(-0.9161)) = 0.0333$$

$$\delta_2 = -0.0021 \text{ x } 4.1936 \; x \; (1-4.1936) = 0.0281$$

.
.
.

$$\delta_{250} = -0.0034 \text{ x } 6.7523 \; x \; (1-6.7523) = 0.1320$$

Langkah 6:

Menghitung koreksi bobot dari *input layer* ke *hidden layer* menggunakan Persamaan (2.17).

$$\Delta V_{1.1} = 0.2 \; x \; 0.0333 \text{ x} -4.5174 = -0.0300$$

$$\Delta V_{1.2} = 0.2 \; x \; 0.0281 \text{ x } -20.5236 = -0.1153$$

.
.
.

$$\Delta V_{91.250} = 0.2 \text{ x } 0.1320 \text{ x } 0.8254 = 0.0217$$

Langkah 7:

Menghitung koreksi bobot bias dari *input layer* ke *hidden layer* menggunakan Persamaan (2.18).

$$\Delta V_{0.1} = 0.2 \; x \; 0.0333 = 0.0067$$

$$\Delta V_{0.2} = 0.2 \; x \; 0.0281 = 0.0056$$

.
.
.

$\Delta V_{0.250} = 0.2 \text{ x } 0.1320 = 0.0264$

d. Tahap *Update* Bobot

Pada tahap ini, bobot yang lama yaitu bobot hasil inisialisasi akan digantikan dengan bobot yang baru. Bobot baru merupakan bobot hasil penjumlahan dari bobot lama dengan koreksi bobotnya. Berikut merupakan tahap proses dari update bobot.

Pada tahap ini, bobot lama beserta biasnya di update dengan persamaan.

Tabel 3. 12 Proses update bobot

Proses *Update* Bobot
Langkah 1: Perbaiki bobot W $W_{1.1}(baru) = -0.2895 + 0.0047$ = -0.2848 $W_{1.2}(baru) = -0.0677 - 0.0136 =$ -0.0813 . . . $W_{250.62}(baru) = -0.4507 - 0.001 =$ -0.4517
Langkah 2: Perbaiki bobot bias W0 $W_{01}(baru) = 0.4992 + 0.0163 = 0.5000$ $W_{02}(baru) = 0.4600 - 0.0138 = 0.4462$ $W_{62}(baru) = 0.3510 - 0.0218 = 0.3292$
Langkah 3: Perbaiki bobot V $V_{1.1}(baru) = -0.4589 - 0.0300$ = -0.4889 $V_{1.2}(baru) = 0.3503 - 0.1153 = 0.2350$ . . . $V_{91.250}(baru) = -0.2758 + 0.0217$ = -0.2541
Langkah 4:

Perbaiki bobot bias V0

$V_{01}(baru) = -0.4118 + 0.0067 =$ -0.4051

$V_{02}(baru) = -0.3554 + 0.0056 =$ -0.3498

.
.
.

$V_{0250}(baru) = -0.2654 - 0.0264 =$ -0.2918

Proses pelatihan akan berhenti bila *Mean Square Error* (MSE) sudah lebih kecil dengan *error* yang sudah ditetapkan. Agar dapat mengendalikan pembelajaran, maka iterasi akan dihentikan jika sudah besar dari *epoch* yang sudah ditetapkan pada tahap inisialisasi. Berikut merupakan bobot dari hasil pelatihan dengan *error 0.001*, *epoch* 1000 dan *learning rate* 0.05. nilai learning rate di pilih sebesar 0.05 agar nilai bobot pada proses pelatihan tidak terlalu cepat berubah-ubah sehingga dapat meminimumkan kemungkinan error yang besar.

	Z_1	Z_2	Z_3	…	Z_{250}
X_1	-0,009424	-0,512446	0,01630	...	-0,840636
X_2	0,016696	-1,144063	0,03512	...	-0,617479
⋮	⋮	⋮	⋮	⋮	⋮
X_{91}	0,042974	-0,713769	0,050920	...	0,023757
Bias	0,032409	0,046697	0,046697	...	-0,166388

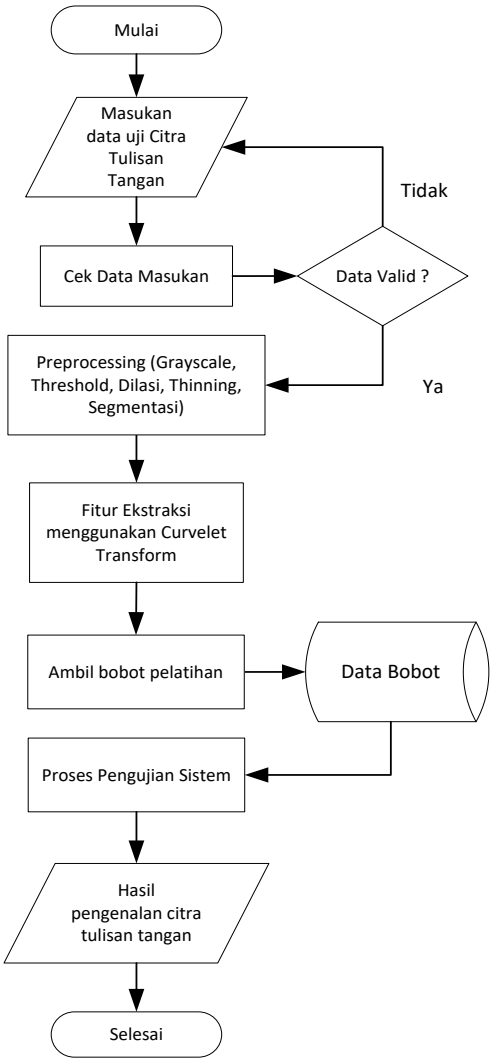
Gambar 3. 28 Bobot Input Akhir

	Y_1	Y_2	Y_3	…	Y_{62}
Z_1	0,049856	0,039656	0,056905	...	0,4296884
Z_2	0,210675	-0,041552	0,227606	...	0,6812988
⋮	⋮	⋮	⋮	⋮	⋮
Z_{250}	0,133895	-0,833125	0,167254	...	-0,268443
Bias	0,090951	0,380739	0,343380	...	0,327407

Gambar 3. 29 Bobot HIdden Ahkhir

2. Pengujian *Backpropagation*

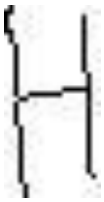
Tahap pengujian dengan menggunakan algoritma *backpropagation* hanya menggunakan proses *feedforward*. Bobot yang dihasilkan pada tahap pelatihan sebelumnya akan digunakan untuk proses pengujian. Berikut merupakan flowchart pengujian *backpropagation* pada Gambar 3.30.



Gambar 3. 30 Flowchart Pengujian Backpropagation

Sebelum masuk dalam tahap pengujian *Backpropagation*, citra inputan akan dilakukan proses *preprocessing* dan ekstraksi ciri pada citra tulisan tangan seperti pada tahap sebelumnya hingga menghasilkan nilai koefisien *curvelet.* Berikut adalah contoh citra Huruf ‘H’ yang telah didapatkan nilai curveletnya.

Tabel 3. 13 Tabel Koefesien

Huruf : I	No	Koefisien
	1	0,9693134
	2	0,9032678
	...	
	91	0,8397714

Nilai koefisien *curvelet* yang didapat selanjutkan akan digunakan untuk proses pengujian. Berikut merupakan proses *feedforward* pada pengujian *backpropagation*.

Tabel 3. 14 Tabel Proses FeedForward

Proses *Feedforward*
Langkah 1: Setiap *unit input* (X_i, i = 1, 2, 3, …, 91) mengirimkan sinyal *input* yang berada pada *hidden layer*.
Langkah 2: Hitung keluaran di *hidden layer* menggunakan persamaan (2.7). Z_in_1 = 0,032409 + (0,9693134 x -0,009424) + (0,9032678 x 0,016696) + …+ (0,8397714 x 0,042974) = 2,082493 Z_in_2 = 0,046697 + (0,9693134 x -0,512446) + (0,9032678 x -1,144063) + … + (0,8397714 x -0,713769) = -0,343642 . . . Z_in_{250} = -0,166388 + (0,9693134 x -0,840636) + (0,9032678 x -0,617479) + … + (0,8397714 x 0,023757) = -0,246887
Langkah 3: Aktifkan nilai keluaran di *hidden layer* menggunakan persamaan (2.8).

$Z_1 = \frac{1}{1+e^{-2.082495}} = 0.7971$

$Z_2 = \frac{1}{1+e^{0.343642}} = 0.5871$

.
.
.

$Z_{250} = \frac{1}{1+e^{0.246887}} = 0.4282$

Langkah 4:

Hitung keluaran di *output layer* menggunakan persamaan (2.9).

Y_in_1 = 0,090951 + (0.2857 x 0,049856) + (0.9851 x 0,210675) + …+ (0.0012 x -0,133895) = 0.1575

Y_in_2 = 0,380739 + (0.2857 x 0,039656) + (0.9851 x -0,041552) + …+ (0.0012 x -0,833125) = -0.8288

.
.
.

Y_in_{62} = 0,327407+ (0.2857 x 0,4296884) + (0.9851 x 0,6812988) + …+ (0.0012 x -0,268443) = -0.3217

Langkah 5:

Aktifkan hasil di *output layer* menggunakan Persamaan (2.10).

$Y_1 = \frac{1}{1+e^{-0.1575}} = 0.01198$

$Y_2 = \frac{1}{1+e^{-0,8288}} = 0.01141$

.
.
.

$Y_{62} = \frac{1}{1+e^{-0,8288}} = 0.065667$

Nilai y hasil perhitungan pada langkah pengujian menjadi hasil pengenalan. Kemudian ditentukan batas ambang (*threshold*) dari nilai hasil. Dari fungsi sigmoid biner diambil *threshold* = 0.5, artinya jika nilai y memiliki angka lebih besar dari 0.5, maka nilai tersebut dibulatkan ke atas, dan sebaliknya.

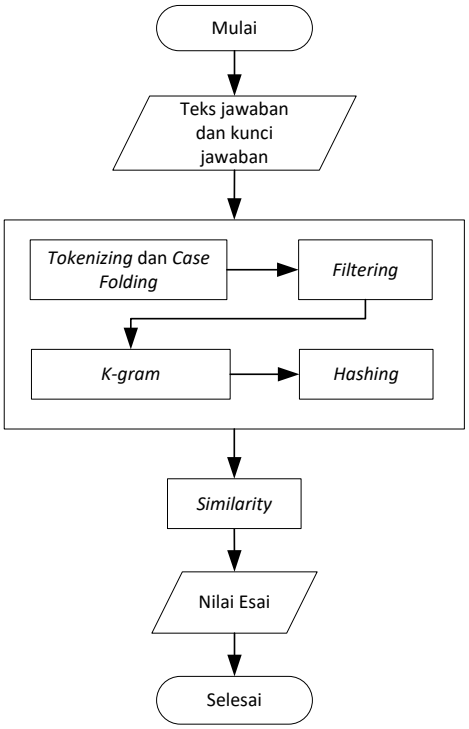
$$Y = f(x) \begin{cases} 1 & \text{jika} \quad x > 0.5 \\ 0 & \text{jika} \quad x < 0.5 \end{cases}$$

Pada perhitungan di atas didapatkan nilai y = {0.01198, 0.01141, 0.0168, … ,0.52134, … , 0.065667} maka hasil y ={0,0,0,0,0, 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0}. , artinya data uji terasuk kelas “H”, yang memiliki nilai biner serupa.

Berdasarkan nilai biner yang di dapatkan dari nilai y maka huruf yang dihasilkan pada pengenalan adalah huruf “H”.

3.2.6. Penilaian Esai

Pada tahapan ini proses penilaian esai hanya menentukan tingkat kemiripan antara kunci jawaban dengan jawaban. Berikut adalah *flowchart* proses penilaian esai dalam gambar 3.31.



Gambar 3. 31 flowchart Penilaian Esai

Metode yang digunakan yaitu Rabin-Karp yang terdiri dari K-gram, *hashing* dan *similarity* dengan proses awal *tokenizing* dan *case folding* kemudian *filtering*. Misalkan jawaban dan kunci jawaban yang didapatkan pada Tabel 3.15.

Tabel 3. 15 Jawaban dan Kunci Jawaban

Jawaban	Kunci Jawaban
Harddisk merupakan perangkat keras yang digunakan untuk menyimpan data	Hard Disk adalah perangkat keras komputer yang bekerja secara sistematis dan digunakan sebagai media penyimpanan data

1. Tokenizing dan Case Folding

Dari jawaban maupun kunci jawaban akan dilakukan proses tokenizing dan case folding dengan memisahkan teks menjadi per kata, kemudian mengubah setiap huruf besar pada teks menjadi huruf kecil. Berikut adalah hasil dari proses tokenizing dan case folding.

Tabel 3. 16 Tokenizing dan Case Folding

Jawaban	Kunci Jawaban
harddisk merupakan perangkat keras yang digunakan untuk menyimpan data	hard disk adalah perangkat keras komputer yang bekerja secara sistematis dan digunakan sebagai media penyimpanan data

2. Filtering

Setelah melalui proses tokenizing dan case folding, selanjutnya dilakukan proses filtering untuk menghilangkan kata yang termasuk ke dalam stoplist / stopword. Kata ‘adalah’,’merupakan’,’yang’,’untuk’ dihilangkan sehingga hasilnya sebagai berikut.

Tabel 3. 17 Filtering

Jawaban	Kunci Jawaban
harddisk perangkat keras digunakan menyimpan data	hard disk perangkat keras komputer bekerja sistematis digunakan media penyimpanan data

3. K-gram

Pada proses ini, teks yang sudah melalui proses tokenizing dan case folding dan filtering dilakukan pemecahan sebanyak k huruf secara kontinyu. Dalam penelitian ini menggunakan k=4 maka hasil K-gramnya sebagai berikut :

Tabel 3. 18 K-gram

Jawaban	Kunci Jawaban
hard, ardd, rddi, ddis, disk, iskp, skpe, kper, pera, eran, rang, angk, ngka, gkat, katk, atke, pker, kera, reas, rasb, asdi, sdig, digu, igun, guna, unak, naka, akan, kann, anme, nmen, meny, enyi, nyim, yimp, impa, mpan, pand, anda, ndat, data	hard, ardd, rddi, ddis, disk, iskp, skpe, kper, pera, eran, rang, angk, ngka, gkat, katk, atke, tker, kera, eras, rask, asko, skom, komp, ompu, mput, pute, uter, terb, erbe, rbek, beke, eker, kerj, erja, rjas, jasi, asis, sist, iste, stem, tema, emat, mati, atis, tisd, isdi, sdig, digu, igun, guna, unak, naka, akan, kanm, anme, nmed, medi, edia, diap, iape, apen, peny, enyi, nyim, yimp, impa, mpan, pana, anan, nand, Anda, ndat, data

4. Hashing

hashing digunakan untuk mengubah sebuah string menjadi suatu nilai yang unik dengan panjang tertentu. Dalam penelitian ini, diketahui nilai yang digunakan sebagai berikut :

K-gram = hard
C = nilai *ascii* setiap karakter K-gram (h=104, a=97, r=114, d=100)
a = 11
k = Sesuai banyak karakter k-gram yaitu 4

maka dari persamaan 2.3 didapatkan hasil sebagai berikut :

$$H = C_1*a_{(k-1)} + C_2*a_{(k-2)} + C_3*a_{(k-3)} + C_4*a_{(k-4)}$$
$$= 104*11^3 + 97*11^2 + 114*11^1 + 100*11^0$$
$$= 138424 + 11737 + 1254 + 100$$
$$= 151515$$

Dari hasil perhitungan di atas, didapatkan nilai *hash* untuk *string* ‘hard’ adalah 167492. Berikut nilai *hash* dari jawaban dan kunci jawaban.

Tabel 3. 19 Hashing

Jawaban	Kunci Jawaban
151515, 144101, 165039, 146470, 147177, 154959, 167345, 157194, 162644, 149402, 164784, 143657, 160147, 151223, 155537, 144421, 168568, 155989, 149407, 164836, 144227, 166423, 147055, 153615, 152557, 170211, 159421, 143231, 155473, 143717, 160820, 158631, 149177, 162315, 175067, 154273, 159808, 162119, 143614, 159693, 146210	151515, 144101, 165039, 146470, 147177, 154959, 167345, 157194, 162644, 149402, 164784, 143657, 160147, 151223, 155537, 144421, 168568, 155989, 149407, 164843, 144310, 167342, 157159, 162279, 160034, 164606, 170988, 167969, 149404, 164810, 143937, 148603, 155998, 149488, 165742, 154193, 144292, 167151, 155047, 168321, 167913, 148803, 158197, 144413, 168466, 154875, 166423, 147055, 153615, 152557, 170211, 159421, 143231, 155473, 143717, 160810, 158505, 147783, 146984, 152825, 143880, 162624, 149177, 162315, 175067, 154273, 159808, 162116, 143594, 159457, 143614, 159693, 146210

Hasil *hash* yang didapatkan selanjutnya akan dicari nilai *fingerprint* (nilai unik) dari jawaban dan kunci jawaban dan menghilangkan nilai *hash* yang serupa. Pada jawaban dan kunci jawaban tidak ditemukan nilai hash unik, sehingga tidak ada perubahan nilai hash pada jawaban dan kunci jawaban.

5. Similarity

Pada tahapan ini dilakukan proses *similarity* untuk mendapatkan tingkat kemiripan antara jawaban dengan kunci jawaban. Tingkat kemiripan ini merupakan nilai yang didapatkan dari hasil penilaian esai.

Tabel 3. 20 Similarity

Jawaban	Kunci Jawaban
151515, 144101, 165039, 146470, 147177, 154959, 167345, 157194, 162644, 149402, 164784, 143657, 160147, 151223, 155537, 144421, 168568, 155989, 149407, 164836, 144227, 166423, 147055, 153615, 152557, 170211, 159421, 143231, 155473, 143717, 160820, 158631, 149177, 162315, 175067, 154273, 159808, 162119, 143614, 159693, 146210	151515, 144101, 165039, 146470, 147177, 154959, 167345, 157194, 162644, 149402, 164784, 143657, 160147, 151223, 155537, 144421, 168568, 155989, 149407, 164843, 144310, 167342, 157159, 162279, 160034, 164606, 170988, 167969, 149404, 164810, 143937, 148603, 155998, 149488, 165742, 154193, 144292, 167151, 155047, 168321, 167913, 148803, 158197, 144413, 168466, 154875, 166423, 147055, 153615, 152557, 170211, 159421, 143231, 155473, 143717, 160810, 158505, 147783, 146984, 152825, 143880, 162624, 149177, 162315, 175067, 154273, 159808, 162116, 143594, 159457, 143614, 159693, 146210

Dari table 3. Dapat dilihat bahwa :

a) Hash jawaban (A) : 41 hash
b) Hash kunci jawaban (B) : 73 hash
c) Hash serupa (C) : 36 hash

Kemudian dengan menggunakan persamaan 2. , akan dihitung nilai similaritas antar jawaban dan kunci jawaban.

$$S = \frac{2 * C}{A + B}$$

$$= \frac{2 * 36}{41 + 73}$$

$$= 0{,}6315 * 100\% = 63{,}15\%$$

Tingkat kemiripan antara jawaban hasil pengenalan tulisan tangan dengan kunci jawaban adalah 63,15%. Maka nilai yang didapatkan dari hasil jawaban terhadap kunci jawaban adalah 63,15.

3.3 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional yang dilakukan dalam penelitian ini yaitu analisis kebutuhan perangkat keras, kebutuhan perangkat lunak dan kebutuhan pengguna.

3.4.1. Analisis Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan merupakan perangkat keras yang mampu mendukung perangkat lunak yang digunakan dalam membuat serta mengoperasikan aplikasi. Berikut adalah spesifikasi perangkat keras minimum yang direkomendasikan :

Tabel 3. 21 Analisis Kebutuhan Perangkat Keras

No	Komponen	Spesifikasi
1	*Processor*	Intel Pentium 4 1.6 GHz
2	RAM	1 GB
3	VGA	256 MB
4	*Harddisk*	250 GB

3.4.2. Analisis Kebutuhan Perangkat Lunak

Analisis perangkat lunak dilakukan agar dapat mengoptimalkan pembangunan aplikasi serta menjalankan aplikasi. Spesifikasi perangkat lunak minimum yang direkomendasikan dalam pembuatan aplikasi ini adalah sebagai berikut.

Tabel 3. 22 Analasis Kebutuhan Perangkat Lunak

No	Perangkat Lunak	Keterangan
1	Sistem Operasi	Microsoft Windows 7 Ultimate
2	Tools Pemrograman	Visual Studio 2015

3	Penyimpanan Basis data	XAMPP v.3.2.1
		Mysql Conector 5.1

3.4.3. Analisis Kebutuhan Pengguna

Agar aplikasi dapat dijalankan, maka dibutuhkan pengguna dengan kemampuan yang dapat menjalankan aplikasi yang dibangun. Kemampuan pengguna yang dibutuhkan untuk menjalankan aplikasi yaitu :

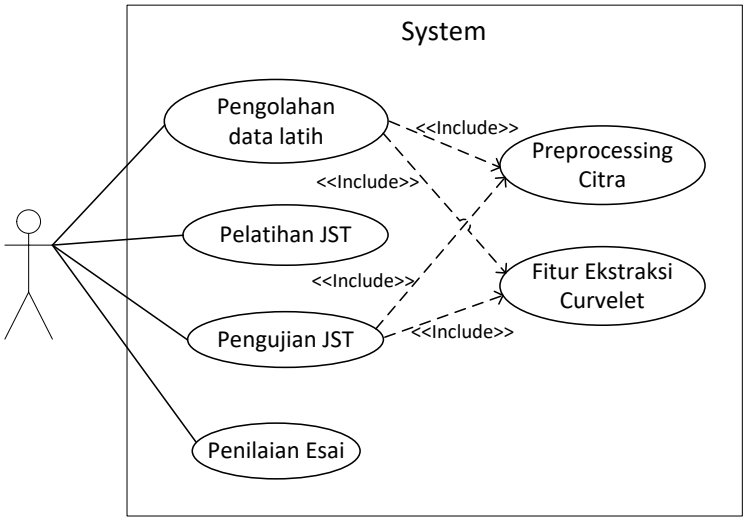
1. Dapat mengoperasikan komputer
2. Mengerti tentang penggunaan Xampp

3.4 Analisis Kebutuhan Fungsional

Selain melakukan analisis terhadap kebutuhan non fungsional, juga dilakukan analisis terhadap kebutuhan fungsionalnya meliputi *use case* diagram, *use case scenario*, *activity* diagram, s*equence* diagram dan *class* diagram.

3.5.1. Use Case Diagram

Use case diagram merupakan permodelan yang menunjukkan interaksi antara sistem dengan pengguna (aktor). Dalam *use case* yang dibuat menggunakan lima *use case*, di mana ada *use case* yang berelasi dengan *use case* lainnya. Relasi yang digunakan yaitu *include*. Berikut adalah *use case* diagram untuk aplikasi yang dibangun pada Gambar 3.32.



Gambar 3. 32 Use case Sistem

1. Definisi Aktor

Aktor merupakan pihak yang mengakses usecase. Table 3.24 Merupakan deskripsi dari aktor.

Tabel 3. 23 Definisi Aktor

No.	Aktor	Deskripsi
1	Pengguna	Pengguna aplikasi

2. Definisi *Use case*

Use case menggabarkan apa saja yang dapat dilakukan oleh sistem. Tabel 3.25 merupakan deskripsi dari setiap use case.

Tabel 3. 24 Definisi Use Case

No.	*Use Case*	Deskripsi
1	Preprocessing citra	Merupakan proses pengolahan citra
2	Fitur ekstraksi curvelet	Merupakan proses pengambilan ciri citra
3	Pengolahan data latih	Merupakan proses pengolahan data latih
4	Pelatihan JST	Merupakan proses pelatihan data latih
5	Pengujian JST	Merupakan proses pengujian data uji
6	Penilaian Esai	Merupakan proses pengecekan nilai esai

3. *Use case Scenario*

Use case scenario menjelaskan skenario dari setiap proses yang digambarkan pada diagram *use case*. Berikut ini skenario *use case* dari diagram *use case* yang terdapat pada Gambar 3.33.

Tabel 3. 25 Use case Scenario *Preprocessing*

Use Case Name	*Preprocessing* Citra
Related Requirments	Ekstraksi Ciri
Goal In Context	Memasukkan dan mengolah citra agar dapat dilakukan ekstraksi ciri.
Precondition	Aktor sudah memilih *file* citra
Successfull End Condition	Sistem berhasil melakukan *preprocessing*.

Failed End Condition	Sistem tidak berhasil melakukan proses *preprocessing*.	
Actors	Pengguna	
Trigger	*File* citra yang akan di proses telah dipilih dan siap untuk diproses.	
Main Flow	**Step**	***Action***
	1	Pengguna memilih input citra
	2	Sistem menampilkan pilihan citra
	3	Pengguna memilih citra tulisan tangan
	4	Sistem menampilkan citra yang dipilih pengguna
	5	Pengguna memilih *Grayscale*
	6	Sistem menampilkan hasil *grayscale*
	7	Pengguna memilih *Threshold*
	8	Sistem menampilkan hasil *Threshold*
	9	Pengguna memilih Dilasi
	10	Sistem menampilkan hasil dilasi
	11	Pengguna memilih *Thinning*
	12	Sistem menampilkan hasil *thinnig*
	13	Pengguna memilih Segmentasi
	14	Sistem menampilkan segmentasi
Extension	**Step**	***Branching Action***
	-	-

Tabel 3. 26 Use case Scenario Ekstraksi Ciri

Use Case Name	Ekstraksi Ciri Curvelet
Related Requirments	*Preprocessing*, Pengolahan data latih, Pengujian
Goal In Context	Mendapatkan ciri dari citra yang akan digunakan untuk proses klasifikasi.
Precondition	Citra telah di *preprocessing* terlebih dahulu.
Successfull End Condition	Sistem berhasil melakukan proses ekstraksi ciri
Failed End Condition	Sistem tidak berhasil melakukan proses ekstraksi

	ciri	
Actors	Pengguna	
Trigger	*File* citra telah di *processing* dan sudah siap untuk di lakukan ekstraksi ciri	
Main Flow	**Step**	***Action***
	1	Pengguna memilih Ekstrasi Ciri
	2	Sistem melakukan proses ekstraksi ciri
Extension	**Step**	***Branching Action***
	-	-

Tabel 3. 27 Use case Scenario Pengolahan Data Training

Use Case Name	Pengolahan data latih	
Related Requirments	Ekstraksi Ciri	
Goal In Context	Melakukan penambahan dan penghapusan data latih	
Precondition	Data latih sudah tersedia	
Successfull End Condition	Sistem berhasil memperbarui data latih	
Failed End Condition	Sistem tidak berhasil memperbarui data latih	
Actors	Pengguna	
Trigger	Ciri citra telah didapatkan dan sudah siap untuk disimpan	
Main Flow	**Step**	***Action***
	1	Pengguna memilih input citra
	2	Sistem menampilkan pilihan citra
	3	Pengguna memilih citra karakter tulisan tangan
	4	Sistem menampilkan citra yang dipilih pengguna
	5	Pengguna memilih simpan data.
	6	Sistem menampilkan data latih baru
Extension	**Step**	***Branching Action***
	-	-

Alternative Flow	**Step**	***Action***
	1	Pengguna memilih data kelas yang akan dihapus kemudian memilih hapus
	2	Sistem melakukan proses hapus data latih kemudian menampilkan pembaharuan data latih
Extension	**Step**	***Branching Action***
	-	-

Tabel 3. 28 Use case Scenario Pelatihan JST

Use Case Name	Pelatihan JST	
Related Requirments	Pengolahan data training, Pengujian	
Goal In Context	Menghitung dan memperbaharui data bobot kelas	
Precondition	Data latih sudah tersedia	
Successfull End Condition	Sistem berhasil memperbarui bobot	
Failed End Condition	Sistem tidak berhasil memperbarui bobot	
Actors	Pengguna	
Trigger	Data pelatihan telah diambil dan siap untuk dilatih	
Main Flow	**Step**	***Action***
	1	Pengguna memasukkan inisialisasi (*Learning rate*, *MaxEpoch*, *Error* Minimum dan Jumlah *Hidden*)
	2	Pengguna memilih pelatihan
	3	Sistem melakukan perhitungan untuk memperbaharui data bobot, kemudian menampilkan pesan jika pelatihan sudah selesai
Extension	**Step**	***Branching Action***
	-	-

Tabel 3. 29 Use case Scenario Pengujian JST

Use Case Name	Pengujian JST	
Related Requirments	Ekstraksi Ciri, Pelatihan	
Goal In Context	Menghitung dan menampilkan hasil konversi dari citra tulisan tangan menjadi teks digital	
Precondition	Citra uji tulisan tangan sudah di masukan	
Successfull End Condition	Sistem berhasil melakukan pengenalan tulisan tangan	
Failed End Condition	Sistem tidak berhasil melakukan pengenalan tulisan tangan	
Actors	Pengguna	
Trigger	Ciri citra telah didapatkan dan siap untuk di klasifikasi	
Main Flow	**Step**	***Action***
	1	Pengguna memilih *input* citra
	2	Sistem menampilkan pilihan citra
	3	Pengguna memilih citra tulisan tangan
	4	Sistem menampilkan citra yang dipilih pengguna
	5	Pengguna melakukan tahapan *preprocessing*
	6	Pengguna melakukan tahapan ekstraksi ciri
	7	Pengguna memilih pengujian
	8	Sistem menampilkan hasil pengujian berupa teks digital
Extension	**Step**	***Branching Action***
	-	-

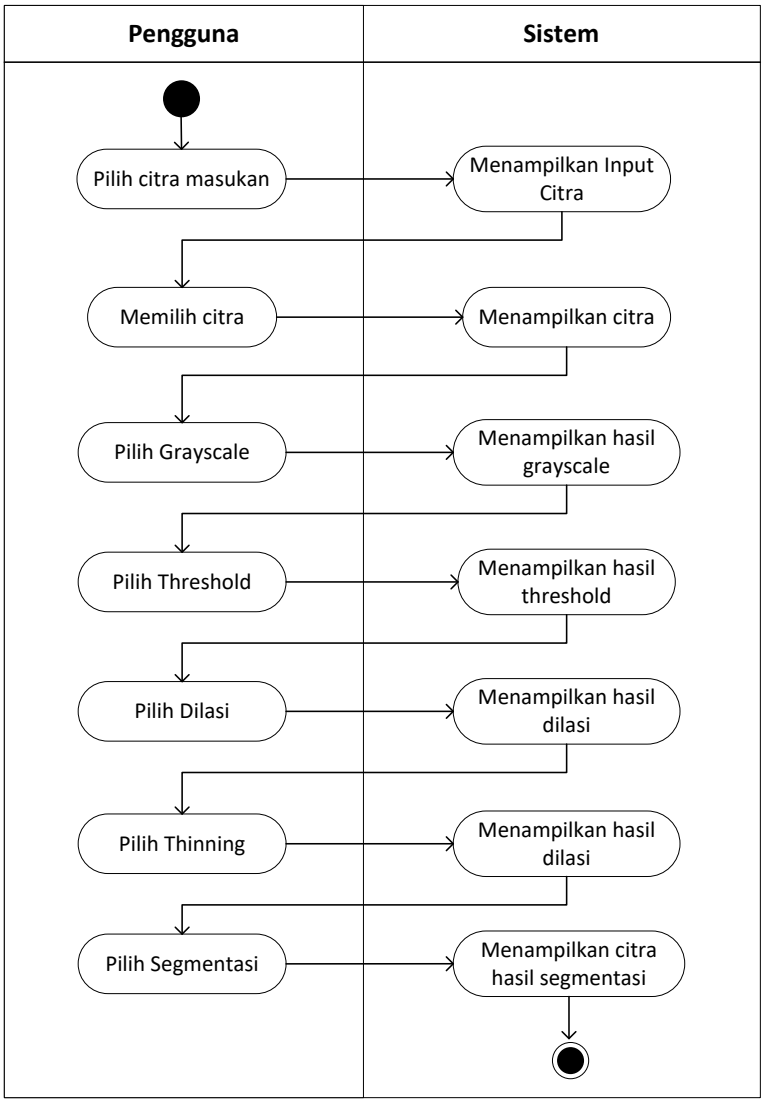
Tabel 3. 30 Use case Scenarion Penilaian Esai

Use Case Name	Penilaian Esai
Related Requirments	Pengujian

Goal In Context	Melakukan evaluasi jawaban ujian	
Precondition	Teks jawaban ujian sudah ada dan nilai ujian belum ditampilkan	
Successfull End Condition	Sistem berhasil menghitung *score* esai	
Failed End Condition	Sistem tidak berhasil menghitung *score* esai	
Actors	Pengguna	
Trigger	Jawaban dan kunci jawaban telah siap untuk dilakukan proses penilaian esai	
Main Flow	**Step**	***Action***
	1	Pengguna memilih tombol cek jawaban
	2	Sistem melakukan proses perhitungan dan menampilkan hasil ujian berdasarkan jawaban yang ada.
Extension	**Step**	***Branching Action***
	-	-

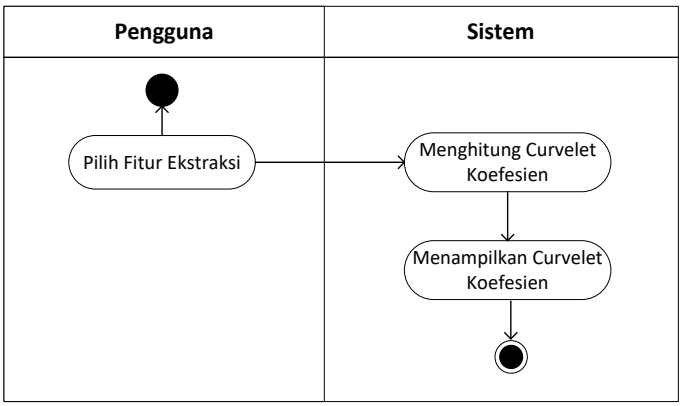
3.5.2. Activity Diagram

Activity diagram memodelkan bagaimana aliran use case bekerja. Berikut adalah activity diagram dari use case sebelumnya.



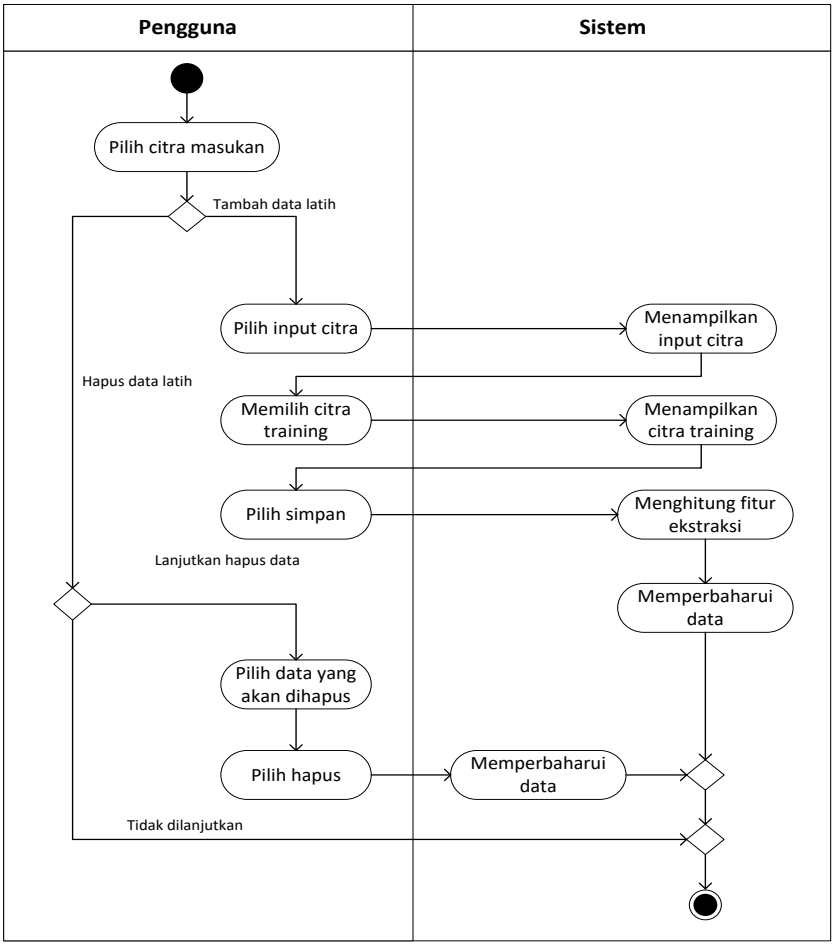
Gambar 3. 33 Activity diagram *Preprocessing*

Gambar 3.33 merupakan aktivitas-aktivitas yang dilakukan pada proses *preprocessing*. Aktivitas dimulai dari memasukkan citra tulisan tangan hingga citra tersebut sudah tersegmentasi



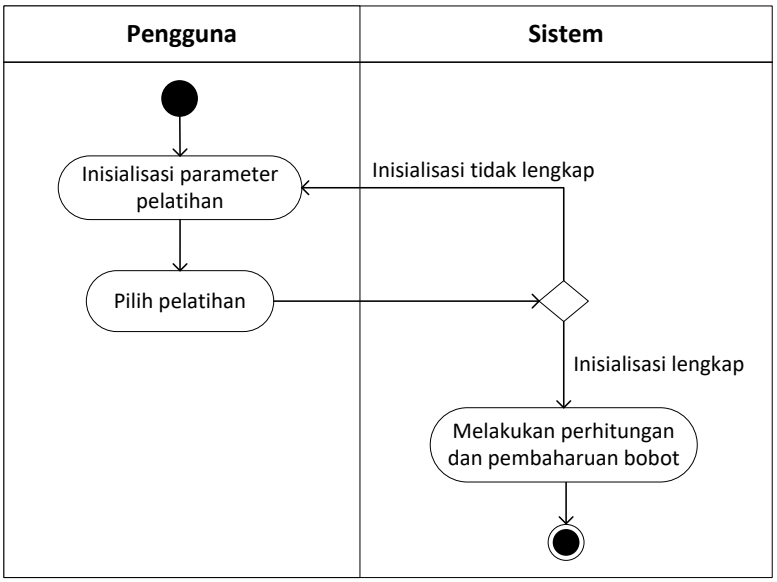
Gambar 3. 334 Activity diagram Fitur Ekstraksi

Gambar 3.34 merupakan aktivitas-aktivitas yang dilakukan pada proses fitur ekstraksi. Aktivitas dimulai dari perhitungan curvelet transform pada setiap citra yang di segmentasi dan menghasilkan koefesien curvelet.



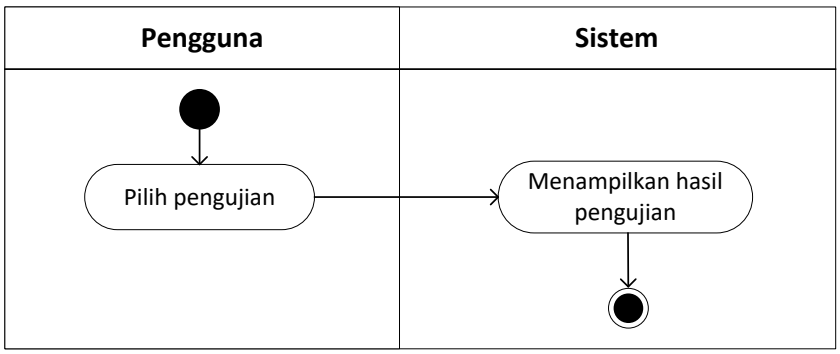
Gambar 3. 345 Activity diagram Pengolahan Data Latih

Gambar 3.35 merupakan aktivitas-aktivitas yang dilakukan pada proses pengolahan data pelatihan. Aktivitas yang dilakukan terdiri dari penambahan data latih dan penghapusan data latih.



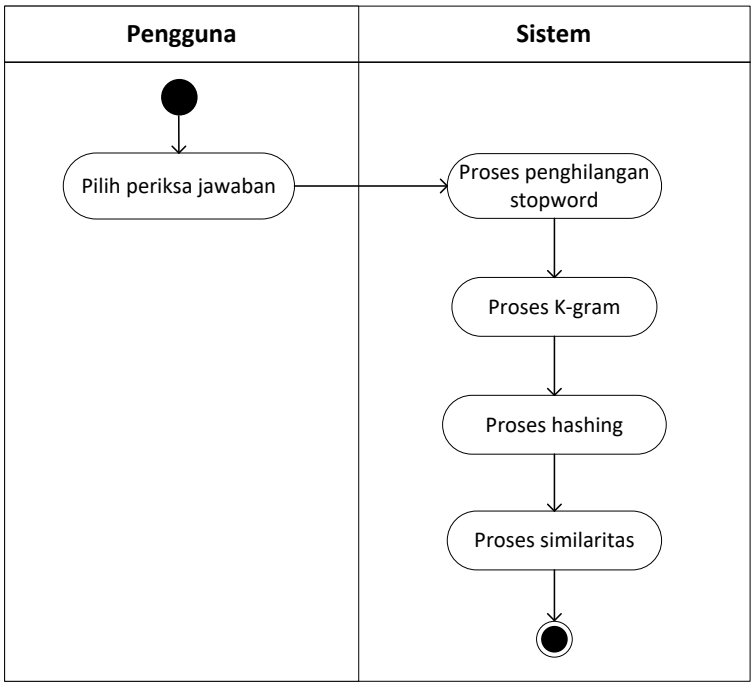
Gambar 3. 356 Activity diagram Pelatihan JST

Gambar 3.36 merupakan aktivitas-aktivitas yang dilakukan pada proses pelatihan JST *Backpropagation*. Pelatihan diawali dengan memasukkan inisialisasi parameter yang digunakan, kemudian pilih pelatihan.



Gambar 3. 367 Activity diagram Pengujian JST

Gambar 3.37 merupakan aktivitas-aktivitas yang dilakukan pada proses pengujian menggunakan JST *Backpropagation*. Setelah pengguna memilih pengujian, maka sistem akan menampilkan teks digital hasil pengujian tersebut.

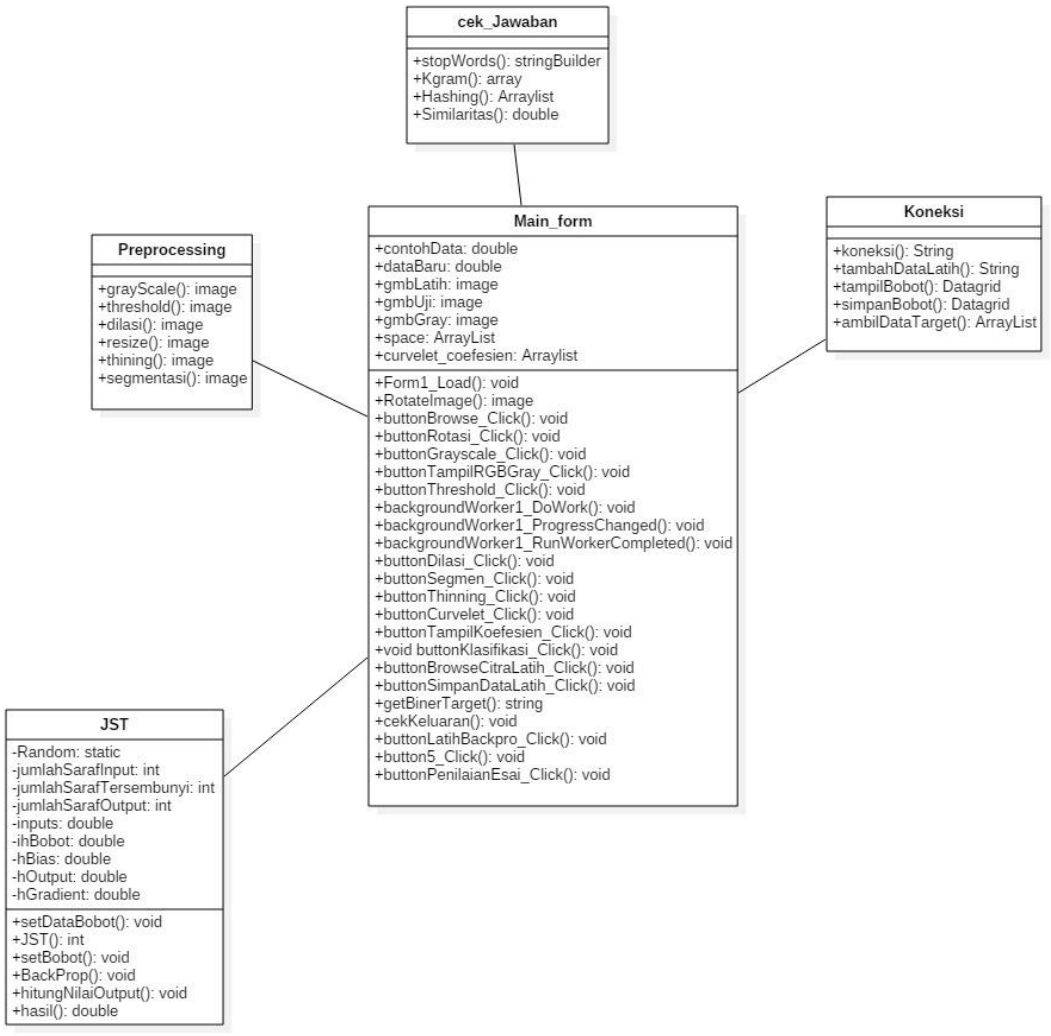


Gambar 3. 378 Activity diagram Penilaian Esai

Gambar 3.38 merupakan aktivitas-aktivitas yang dilakukan pada proses cek jawaban ujian. Setelah pengguna memilih cek jawaban, maka sistem akan melakukan proses perhitungan sampai mendapatkan nilai similaritas.

3.5.3. Class Diagram

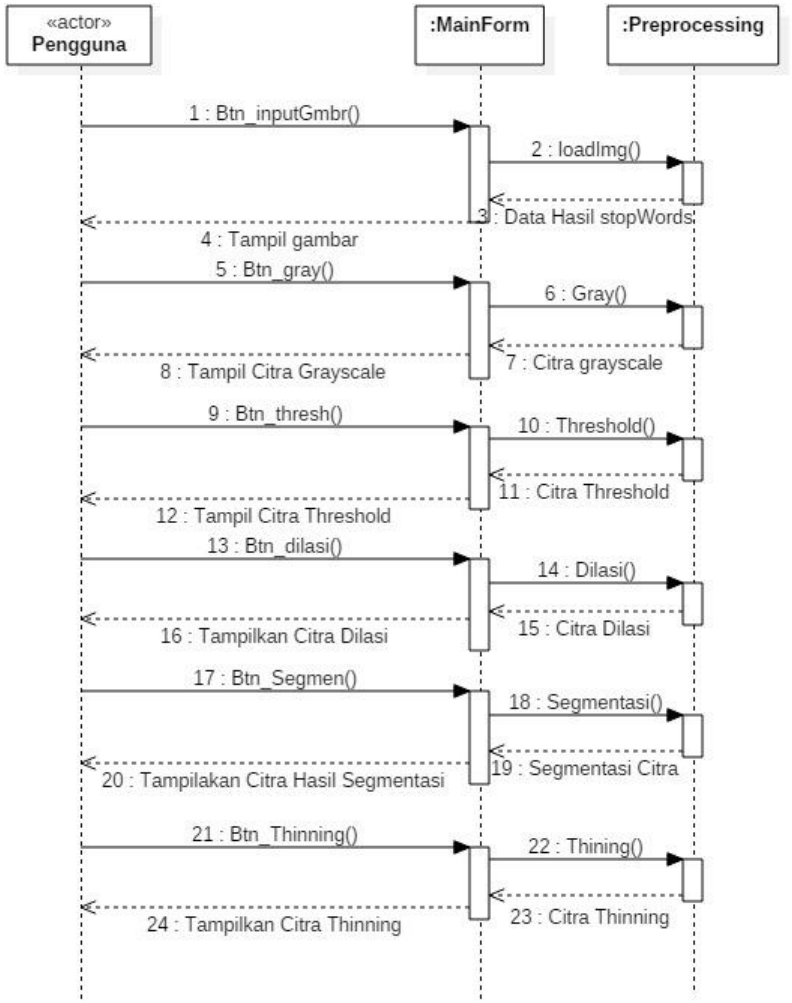
Class diagram menggambarkan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling sering ditemui dalam pemodelan sistem berorientasi objek. Berikut adalah *class diagram* dari sistem dapat dilihat pada gambar 3.39.



Gambar 3. 39 Class Diagram Sistem

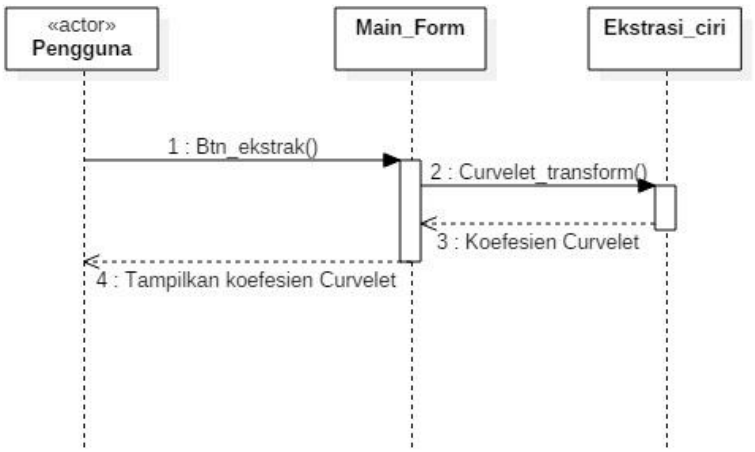
3.5.4. Sequence Diagram

Sequence diagram memungkinkan melihat aliran pesan dari objek ke objek dalam urutan waktu.



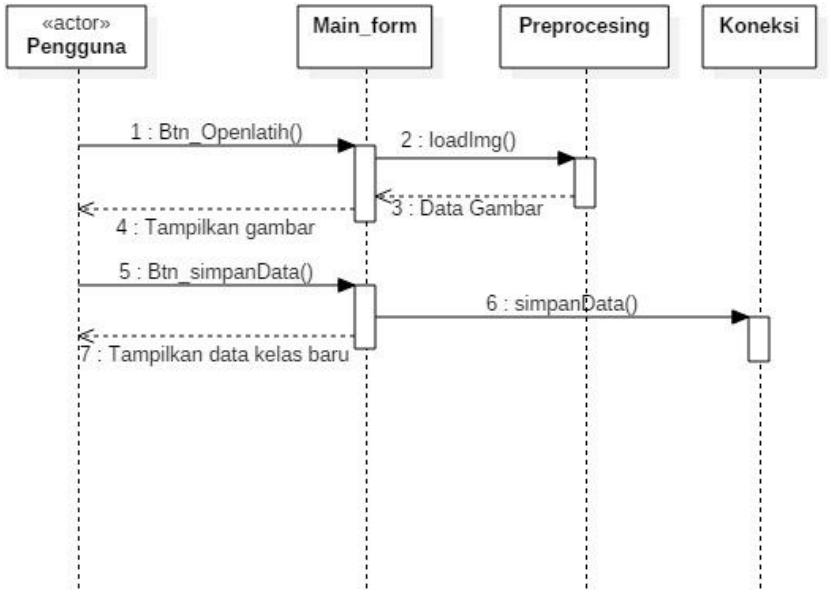
Gambar 3. 380 Sequence Diagram *Preprocessing*

Gambar 3.40 merupakan *sequence diagram* pada proses *preprocessing*. Dalam proses ini, kelas-kelas yang digunakan adalah kelas Main_form dan Preprocessing.



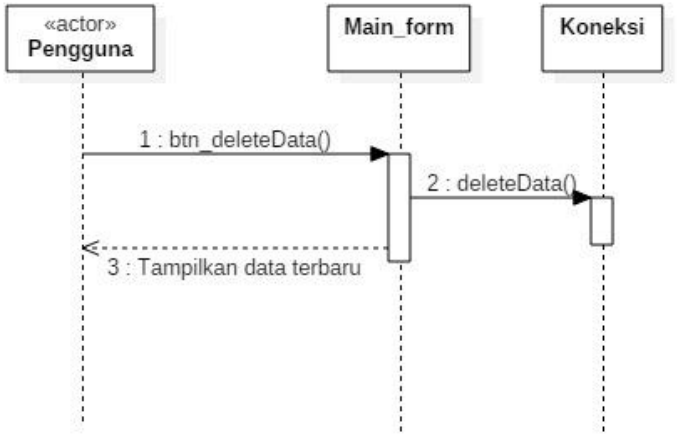
Gambar 3. 391 Sequence Diagram Fitur Ekstraksi

Gambar 3.41 merupakan *sequence diagram* pada proses ekstraksi ciri. Dalam proses ini, kelas-kelas yang digunakan adalah kelas Main_form dan Ekstraksi_ciri.



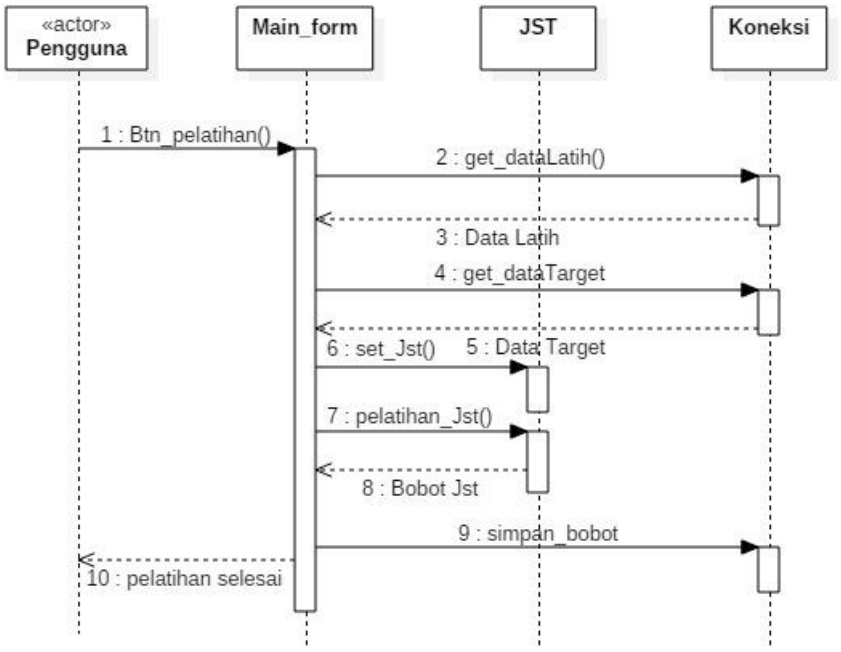
Gambar 3. 402 Sequence Diagram Pengolahan Data Latih

Gambar 3.42 merupakan *sequence diagram* pada proses pengolahan data latih skenario utama, di mana proses yang dilakukan adalah menambahkan data latih. Dalam proses ini, kelas-kelas yang digunakan adalah kelas Main_form, Preprocessing dan Koneksi.



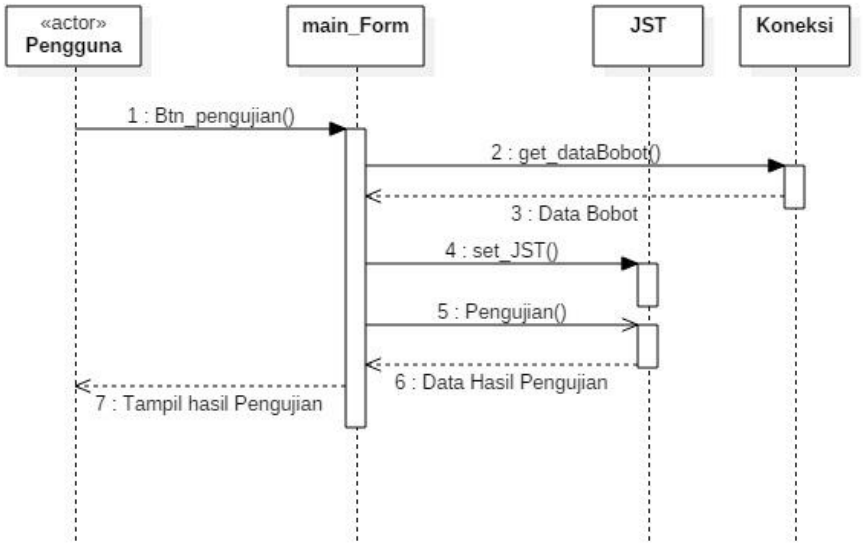
Gambar 3. 413 Sequence Diagram Olah Data Skenario Alternatif

Gambar 3.43 merupakan *sequence diagram* pada proses pengolahan data latih skenario alternatif 1, di mana proses yang dilakukan adalah menghapus data latih. Dalam proses ini, kelas-kelas yang digunakan adalah kelas Main_form dan Koneksi.



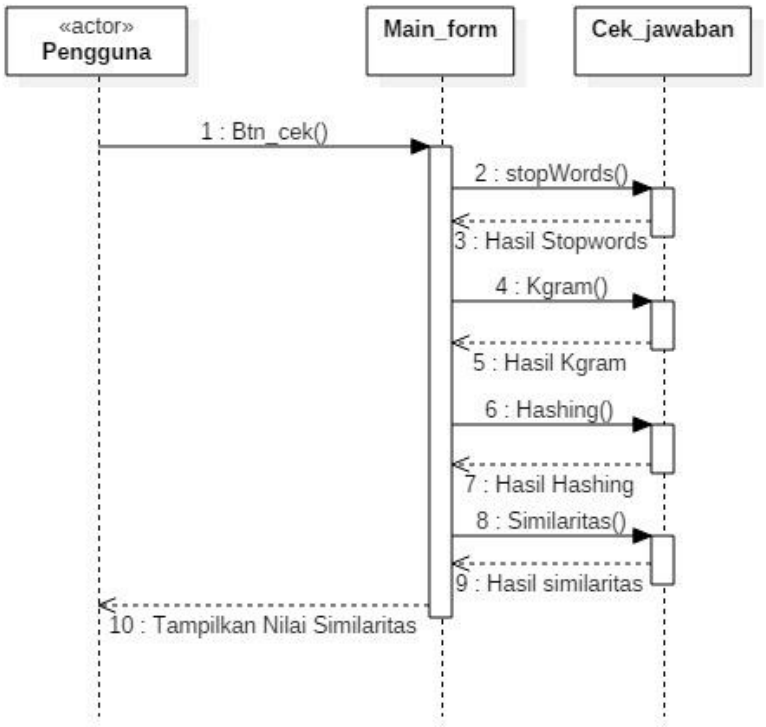
Gambar 3. 424 Sequence Diagram Pelatihan JST

Gambar 3.44 merupakan *sequence diagram* pada proses pelatihan JST. Dalam proses ini, kelas-kelas yang digunakan adalah kelas Main_form, JSTdan Koneksi.



Gambar 3. 435 Sequence Diagram Pengujian JST

Gambar 3.45 merupakan *sequence diagram* pada proses pengujian JST. Dalam proses ini, kelas-kelas yang digunakan adalah kelas Main_form, JST dan Koneksi.



Gambar 3. 446 Sequence Diagram Penilaian Esai

Gambar 3.46 merupakan *sequence diagram* pada proses cek jawaban ujian. Dalam proses ini, kelas-kelas yang digunakan adalah kelas Main_form dan Cek_jawaban.

3.5 Perancangan Sistems

Dalam Perancangan sistem ini, dibuat beberapa perancangan seperti perancangan *database*, antar muka, pesan (*Alert*) dan jaringan semantik.

3.6.1. Perancangan Basis Data

Struktur tabel menggambarkan detail basis data pada tabel yang berisi field, tipe data, dan keterangan lainnya.

1. Tabel data latih

   Tabel data latih merupakan tabel yang akan digunakan untuk menyimpan data yang akan digunakan pada proses pelatihan.

Tabel 3. 31 Struktur Tabel Data_latih

No.	Nama Field	Tipe Data	Ukuran	Keterangan
1	Id_latih	Int	11	*PRIMARY KEY, NOT NULL*
2	Nama_latih	Varchar	12	*NOT NULL*
3	Koefesien_curvelet	Text		*NOT NULL*
4	Target_kelas	Text		*NOT NULL*

2. Tabel bobot v

   Tabel bobot v merupakan tabel yang akan digunakan untuk menyimpan data bobot input – hidden yang digunakan untuk proses pengujian.

Tabel 3. 32 Struktur Tabel Bobot_v

No	Nama Field	Tipe Data	Ukuran	Keterangan
1	Id_bobot	Int	11	*PRIMARY KEY, NOT NULL*
2	Kode_bobot	varchar	11	*NOT NULL*
3	Id_log	Int	11	*NOT NULL*
4	Nilai_bobot	Text		*NOT NULL*

3. Tabel bobot w

   Tabel bobot w merupakan tabel yang akan digunakan untuk menyimpan data bobot hidden - output yang digunakan untuk proses pengujian.

Tabel 3. 33 Struktur Tabel Bobot_w

No	Nama Field	Tipe Data	Ukuran	Keterangan
1	Id_bobot	Int	11	*PRIMARY KEY, NOT NULL*
2	Kode_bobot	varchar	11	*NOT NULL*
3	Id_log	Int	11	*NOT NULL*
4	Nilai_bobot	Text		*NOT NULL*

4. Tabel log pelatihan

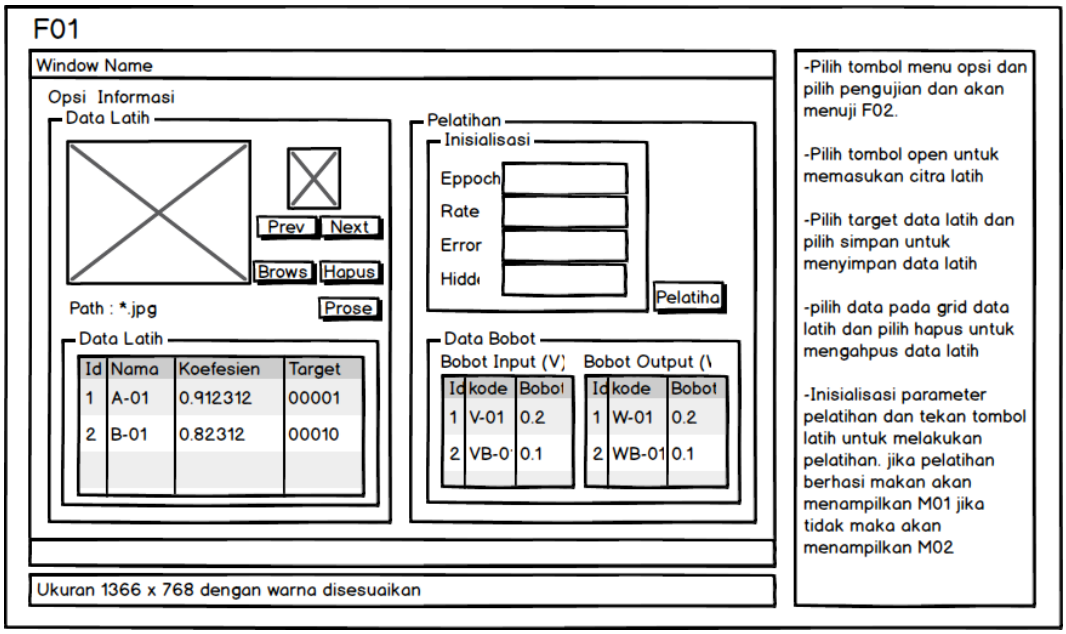
   Tabel log pelatihan merupakan tabel yang akan digunakan untuk menyimpan data parameter hasil pelatihan.

Tabel 3. 34 Struktur Table Log_pelatihan

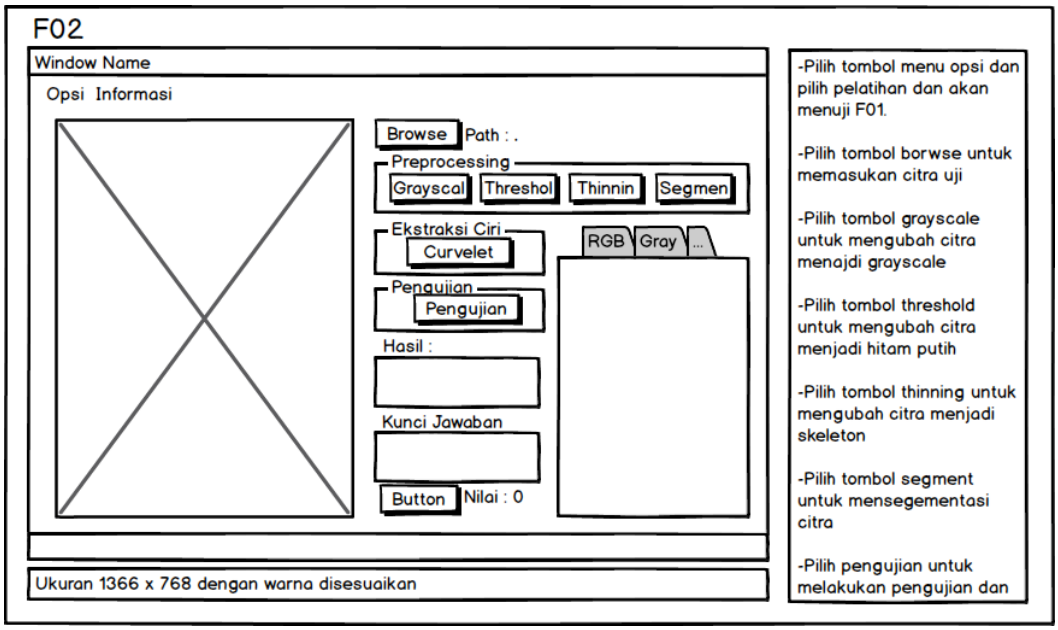
No	Nama Field	Tipe Data	Ukuran	Keterangan
1	Id_log	Int	11	*PRIMARY KEY, NOT NULL*
2	tgl_pelatihan	Datetime		*NOT NULL*
3	Eppoch	int	11	*NOT NULL*
4	Learning	Text		*NOT NULL*
5	Error	Text		*NOT NULL*

3.6.2. Perancangan Antarmuka

Perancangan antarmuka merupakan perancangan *interface* yang dilakukan dari aplikasi yang dibuat. Dalam penelitian ini menggunakan dua antarmuka yang bisa dilihat pada Gambar 3.47 dan Gambar 3.48.



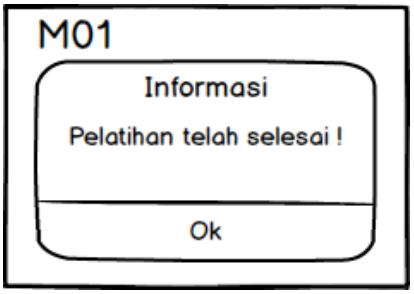
Gambar 3. 47 Tampilan Antar Muka Pelatihan



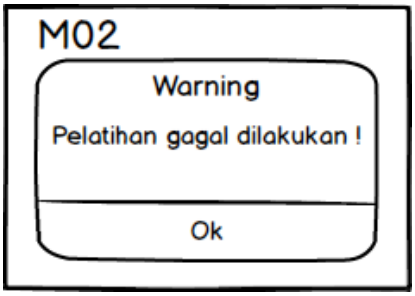
Gambar 3. 48 Tampilan Antar Muka Pengujian

3.6.3. Perancangan Pesan

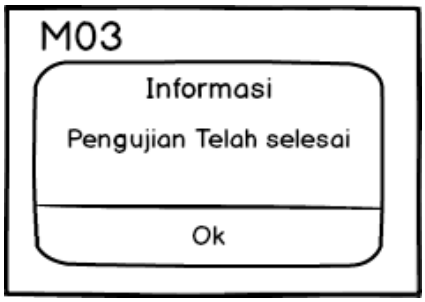
Pesan berguna untuk membuat sistem lebih interaktif, karena dapat menyampaikan informasi apakah suatu proses berjalan sukses atau sebaliknya. Berikut adalah beberapa Perancangan Message Box dari sistem yang dibuat.



Gambar 3. 49 Pesan Pelatihan Selesai



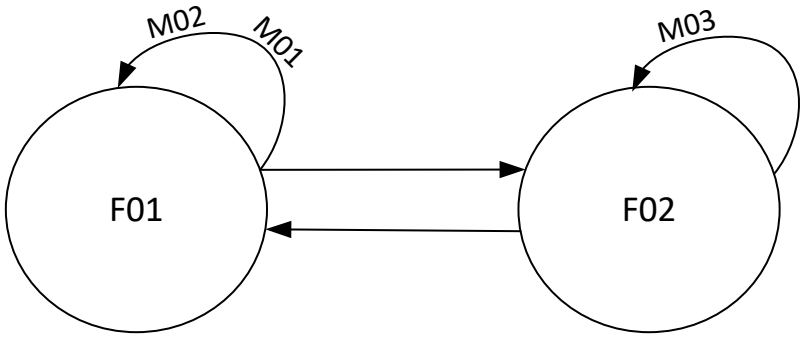
Gambar 3. 45 Pesan Pelatihan Gagal



Gambar 3. 46 Pesan Pengujian Selesai

3.6.4. Jaringan Semantik

Jaringan Semantik dapat memudahkan untuk melihat bagaimana alur atau keterhubungan antara satu antarmuka dengan antarmuka yang lainnya atau antar muka dengan pesan. Berikut adalah Jaringan Semantik dari sistem yang dibuat pada Gambar 3.52.



Gambar 3. 472 Jaringan Semantik Sistem