

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

Berdasarkan latar belakang dan identifikasi masalah, maka akan dijelaskan tentang analisis masalah yang ditemukan di dalam penelitian ini. Adapun masalahnya yaitu belum diketahuinya akurasi pengenalan tulisan tangan bahasa Sunda dengan metode klasifikasi *Convolutional Neural Network* (CNN). Pada penelitian sebelumnya CNN telah membuktikan ke efektifitasannya dalam hal mendeteksi text aksara Jawa pada gambar scene [9] dan mendapatkan 87,48% akurasi untuk mengenali wajah secara real-time [7]. Dan juga metode CNN berdasarkan penelitian belum pernah dikombinasikan secara lengkap dengan metode *Sauvola Thresholding* dalam kasus pengenalan tulisan tangan aksara Sunda, dalam penelitian ini Metode CNN akan dikombinasikan dengan metode *Sauvola Thresholding* dengan menggabungkan kedua algoritma secara bersamaan kedalam penelitian, kedua metode ini cukup baik pencapaian hasil akurasinya. Kedua metode ini digunakan untuk mengatasi permasalahan mengenai pengenalan tulisan tangan berbentuk huruf cetak. Proses *Thresholding* sangat penting dalam penelitian ini karena dari proses itulah baru bisa dilakukan metode klasifikasi dan mengetahui akurasi dari pengenalan tulisan tangan [14], sehingga perlu dilakukan proses *Sauvola Thresholding* terlebih dahulu.

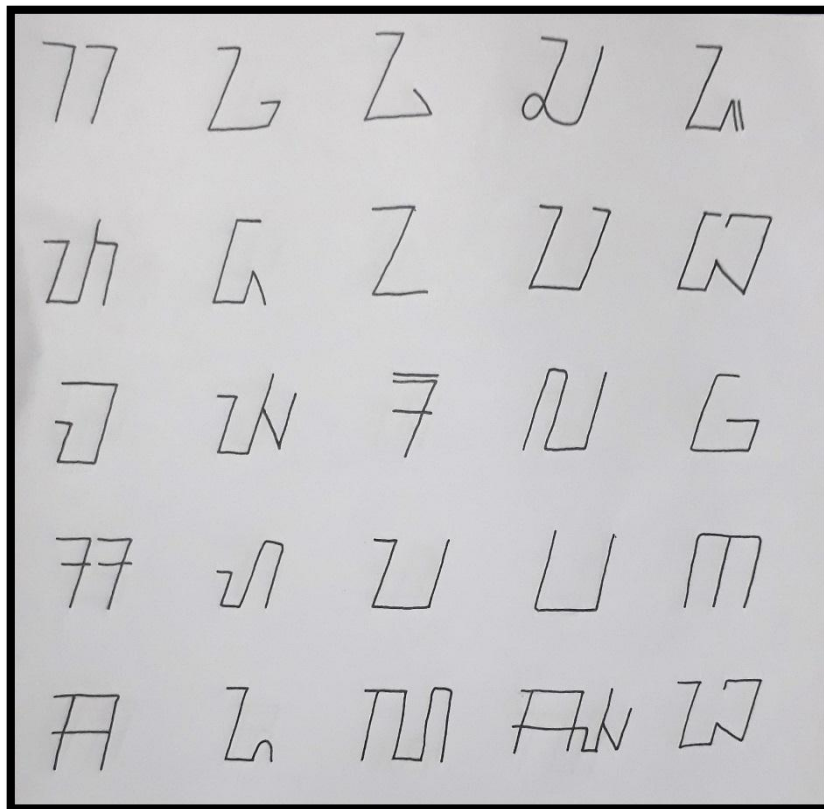
3.2 Analisis Sistem

Dalam analisis sistem akan dijabarkan mengenai analisis yang berkaitan dengan pembuatan aplikasi seperti analisis solusi, analisis proses yang dilakukan, basis data.

3.2.1 Analisis Data Masukan

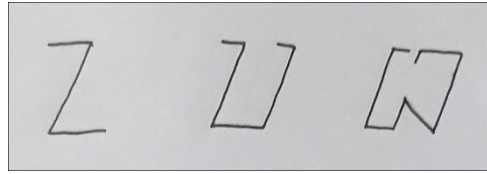
Penyelesaian masalah dilakukan dengan membangun aplikasi pengenalan tulisan tangan dengan metode CNN dan input berupa citra ditulisan tangan aksara sunda. Data masukan diproses dengan pengolahan citra dan diimplementasikan

metode *Convolutional Neural Network* yang menggunakan data latih dari 100 sampel karakter tulisan tangan, sehingga data keluaran atau akurasi diharapkan bernilai tinggi.



Gambar 3. 1 Contoh Citra Tulisan Tangan Data Masukan

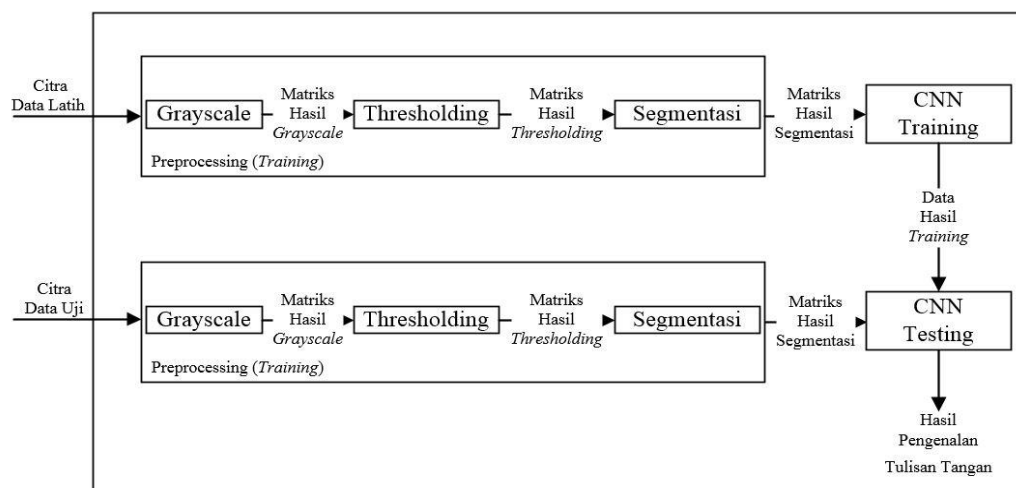
Citra tulisan tangan yang digunakan sebagai data latih merupakan citra tulisan yang tersusun dari karakter ka, ga, nga, ca, ja, nya, ta, da, na, pa, ba, ma, ya, ra, la, wa, sa, ha, fa, va, qa, xa, za, kha dan sya. Data masukan atau data latih ditulis diatas kertas dengan background putih polos. Dalam penelitian ini citra karakter yang digunakan hanya tiga karakter yaitu ca, ja dan nya digunakan untuk mempermudah peneliti dalam memvisualisasikan nilai-nilai yang dihasilkan dari setiap proses.



Gambar 3. 2 Citra masukan yang digunakan

3.2.2 Analisis Proses

Proses yang dilakukan dalam aplikasi ini dibagi menjadi beberapa bagian dimana setiap proses memiliki peranan masing-masing dalam melakukan pengenalan tulisan tangan. Proses yang dilakukan pada gambar 3.3.



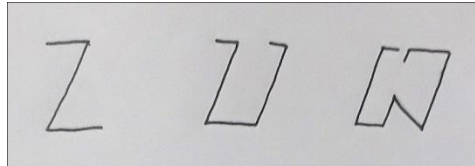
Gambar 3. 3 Gambaran Umum Sistem

Pada blok diagram di Gambar 3.3, proses awal yang dilakukan yaitu memasukkan citra tulisan tangan ke dalam aplikasi. Langkah selanjutnya melalui tahap pengolahan citra meliputi *grayscale*, *thresholding* dan *segmentasi*. Dari tahap pengolahan citra akan didapatkan array nilai desimal dari setiap ciri karakter, *array* desimal kemudian diolah kembali pada tahap klasifikasi dengan metode CNN baik pelatihan maupun pengujian. Hasil dari pengenalan tulisan tangan didapatkan teks digital.

3.2.2.1 Analisis Data Masukan

Data masukan yang digunakan dalam aplikasi ini, yaitu citra tulisan tangan dengan format gambar .jpg. Citra dapat diperoleh melalui scan atau foto dengan resolusi kamera minimal 13 *megapixel*. Citra tulisan tangan yang digunakan ditulis

menggunakan pulpen/spidol hitam 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.



Gambar 3. 4 Contoh Citra Tulisan Tangan

Gambar 3.4 merupakan sebagian gambar data masukan sebagai data latih yang diambil untuk dibahas didalam laporan. Berikut adalah gambar matrik warna *Red* (R), *Green*(G) dan *Blue* (B) dari contoh citra masukan pada Gambar 3.4.

Tabel 3. 1 Matriks RGB (Nilai RGB Citra Masukan)

x/y	1	2	3	4	...	172
1	R=162	R=162	R=161	R=161	...	R=153
	G=160	G=160	G=161	G=161	...	G=151
	B=161	B=161	B=161	B=161	...	B=152
2	R=162	R=162	R=161	R=162	...	R=153
	G=160	G=160	G=161	G=162	...	G=151
	B=161	B=161	B=161	B=162	...	B=152
3	R=162	R=162	R=163	R=163	...	R=153
	G=160	G=160	G=161	G=161	...	G=151
	B=161	B=161	B=162	B=162	...	B=152
4	R=162	R=162	R=163	R=163	...	R=15
	G=160	G=160	G=161	G=161	...	G=151
	B=161	B=161	B=162	B=162	...	B=152
...
59	R=160	R=160	R=161	R=161	...	R=153
	G=160	G=160	G=161	G=161	...	G=153
	B=160	B=160	B=161	B=161	...	B=155

3.2.2.2 Grayscale

Mengubah format warna menjadi *grayscale* berfungsi untuk mengecilkan range warna menjadi 0 sampai dengan 255. Proses ini akan memudahkan ketika ingi melakukan *threshold* citra menjadi citra hitam putih. Gambar 3.5 merupakan blok diagram proses *grayscale*.



Gambar 3. 5 Blok Diagram Proses Grayscale

Adapun langkah-langkah yang dilakukan sebagai berikut.

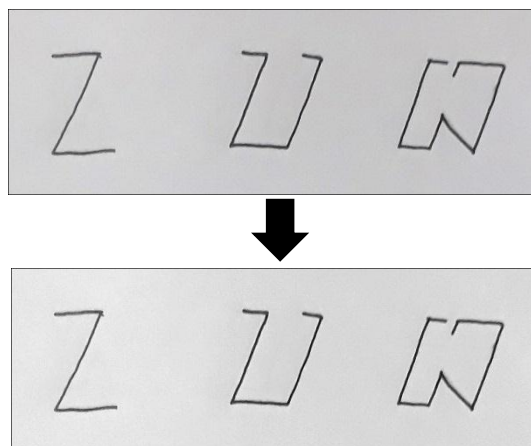
1. Warna citra dikelompokkan berdasarkan nilai *red*, *green* dan *blue*
2. Kemudian menggunakan persamaan 2.1, maka akan didapatkan nilai warna *grayscale* citra.
3. Nilai *grayscale* yang didapat menggunakan nilai RGB pada setiap *pixel*.

Misalkan citra pada *pixel* $RGB_{1,1}$ mempunyai nilai *Red* = 162, *Green* = 160, *Blue* = 161, maka berdasarkan persamaan 2.1 menjadi

$$\begin{aligned}
 I &= (0,2989 * R) + (0,5870 * G) + (0,1141 * B) \\
 &= (0,2989 * 162) + (0,5870 * 160) + (0,1141 * 161) \\
 &= 48.42 + 93.92 + 18.37 \\
 &= 160,37 \\
 &= 160 \text{ (nilai yang digunakan).}
 \end{aligned}$$

karena angka dibelakang koma kurang dari (< 5), maka akan dibulatkan menjadi 160.

Dari perhitungan di atas , maka *pixel* yang tadinya bernilai *Red* = 162, *Green* = 160, *Blue* = 161 diperbaharui menjadi nilai *grayscale* = 160. Gambar 3.5 di bawah ini merupakan hasil dari proses *grayscale*.



Gambar 3. 6 Hasil Proses Grayscale

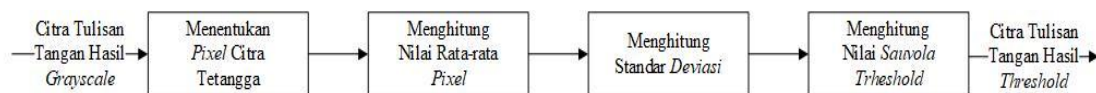
Berikut adalah gambar matrik warna *Red* (R), *Green* (G), dan *Blue* (B) dari hasil prose *grayscale*.

Tabel 3. 2 Matriks G (Nilai Hasil *Grayscale* Citra Masukan)

x/y	1	2	3	4	...	173
1	160	160	161	161	...	151
2	160	160	161	161	...	151
3	160	160	161	161	...	151
4	160	160	161	161	...	151
...
60	159	159	161	161	...	153

3.2.2.3 *Threshold*

Metode *threshold* yang digunakan dalam penelitian ini menggunakan metode *Sauvola Threshold* yang bertujuan untuk membedakan objek dan *Background* dari citra agar lebih mudah dikenali pada tahapan selanjutnya. *Sauvola* masuk ke dalam kategori dari metode *local threshold*, dimana nilai ambang pada setiap *pixel* nya tergantung dari jumlah tetangga yang digunakan. Gambar 3.7 merupakan *flowchart* proses *Sauvola Threshold*.



Gambar 3. 7 Blok Diagram Proses *Sauvola Threshold*

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 hasil yang didapatkan 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* Matriks $G_{1,1}$ dengan nilai *grayscale* 160. Selanjutnya cari *pixel* tetangga mana saja yang sesuai dengan jumlah tetangga yang sudah ditentukan sebelumnya yaitu 21. Kemudian dari hasil pencarian tersebut didapatkan matriks citra seperti tabel 3.3, dimana tetangga $G_{1,1}$ adalah dari $G_{1,1}$ sampai dengan $G_{11,11}$.

Tabel 3. 3 Contoh Matriks G yang akan di *Threshold*

x/y	1	2	3	4	5	6	7	8	9	10	11	...	173
1	160	160	161	161	161	161	161	161	161	161	161	...	151
2	160	160	161	161	161	161	161	161	161	161	161	...	151
3	160	160	161	161	161	161	161	161	161	161	161	...	151
4	160	160	161	161	161	161	161	161	161	161	161	...	151
5	160	160	160	161	161	161	161	161	161	161	161	...	151
6	160	160	160	160	160	160	161	161	161	161	161	...	151
7	159	159	160	160	160	160	160	160	161	161	161	...	151
8	159	159	159	160	160	160	160	160	163	161	161	...	151
9	160	160	160	160	160	160	160	161	161	158	163	...	151
10	160	160	160	159	160	160	160	160	159	161	161	...	152
11	160	160	160	159	159	159	160	160	159	161	159	...	152
...
60	159	159	161	161	160	160	160	161	160	160	160	...	153

Langkah pertama cari nilai rata-rata $m_{1,1}$ dengan persamaan 2.3, sehingga hasilnya seperti berikut.

$$m(x,y) = \frac{\sum_{i=\min}^{i=\max} \sum_{j=\min}^{j=\max} G_{i,j}}{i*j}$$

$$m(1,1) = \frac{160+160+161+\dots+153}{11*11}$$

$$= 160,534$$

menurut aturan pembulatan jika angka dibelakang koma lebih dari 5 maka akan di bulatkan ke atas dari 160,534 menjadi 160 (nilai yang digunakan). Kemudian untuk mencari nilai standar deviasi $s(1,1)$ digunakan persamaan 2.4 sehingga hasilnya sebagai berikut.

$$s(x,y) = \sqrt{\frac{\sum_{i=\min}^{i=\max} \sum_{j=\min}^{j=\max} (G_{i,j} - m(x,y))^2}{(i*j)-1}}$$

$$s(1,1) = \sqrt{\frac{(160-160)^2 + (160-160)^2 + (161-160)^2 + \dots + (153-160)^2}{(11*11)-1}} = 0,812$$

langkah selanjutnya masukkan nilai $m(1,1)$ dan $s(1,1)$ ke dalam persamaan 2.2 untuk mendapatkan nilai ambang $T(1,1)$. Sehingga hasilnya menjadi

$$T(x,y) = m(x,y) * (1 + k * (\frac{s(x,y)}{R} - 1))$$

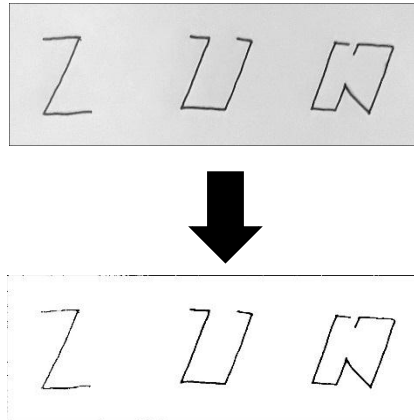
$$\begin{aligned} T(1,1) &= m(1,1) * (1 + 0,3 * (\frac{s(1,1)}{128} - 1)) \\ &= 160 * (1 + 0,3 * (\frac{0,81}{128} - 1)) \\ &= 112,6 \end{aligned}$$

Karena dua angka dibelakang koma bernilai genap dan lebih dari 5 maka dibulatkan keatas menjadi 112,5. Dari perhitungan di atas didapatkan nilai ambang 112,5. Langkah selanjutnya langsung dimasukkan dalam persamaan 2.5 sehingga akan didapatkan nilai *pixel* baru. Dari contoh di atas maka *pixel* yang awalnya dengan nilai 160 akan berubah menjadi 255 karena 160 lebih besar dari 112,5. Nilai 255 didapatkan berdasarkan persamaan 2.5. Berikut adaah persamaan 2.5.

$$f(x,y) = \begin{cases} 0, & \text{img}(x,y) < T(x,y) \\ 255, & \text{img}(x,y) \geq T(x,y) \end{cases}$$

Diketahui nilai $G_{1,1} = 160$ dan nilai $T(1,1) = 112,5$ maka nilai baru untuk piksel $G_{1,1}$ adalah 255 karena nilai $G_{1,1} > T(1,1)$. Berikut adalah gambar citra yang telah di

proses menggunakan *sauvola threshold*. Berikut adalah gambar citra yang sudah diproses dengan menggunakan metode *sauvola threshold*.



Gambar 3. 8 Hasil *Sauvola Threshold*

Kemudian tabel 3.4 berikut adalah matrik warna *Red* (R), *Green* (G) dan *Blue* (B) dari hasil proses *Sauvola Threshold*.

Tabel 3. 4 Matriks H (Citra Masukan Hasil *Threshold*)

x/y	1	2	3	4	...	173
1	255	255	255	255	...	255
2	255	255	255	255	...	255
3	255	255	255	255	...	255
4	255	255	255	255	...	255
...
60	255	255	255	255	...	255

3.2.2.4 Binerisasi

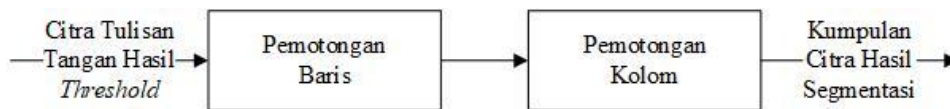
Setelah didapatkan nilai *threshold* dari citra latih, maka langkah selanjutnya adalah merubah nilai *threshold* ke dalam nilai biner dengan merubah nilai 255 menjadi angka 0 dan nilai 0 menjadi angka 1. Sehingga didapatkan matriks nilai binerisasi dari citra *threshold*. Berikut adalah matriks nilai biner yang didapatkan.

Tabel 3. 5 Matriks B (Hasil Binerisasi Citra Masukan)

x/y	1	2	3	4	...	173
1	0	0	0	0	...	0
2	0	0	0	0	...	0
3	0	0	0	0	...	0
4	0	0	0	0	...	0
...
60	0	0	0	0	...	0

3.2.2.5 Segmentasi Citra

Pada proses ini, input yang digunakan yaitu citra hitam putih hasil *sauvola threshold*. Selanjutnya akan dilakukan pemotongan untuk mendapatkan citra huruf tulisan tangan. Berikut adalah blok *diagram* proses segmentasi dalam gambar 3.9.

**Gambar 3. 9 Blok Diagram Proses Segmentasi**

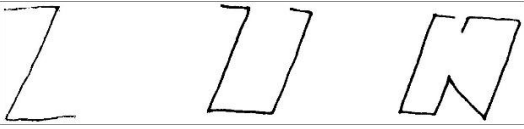
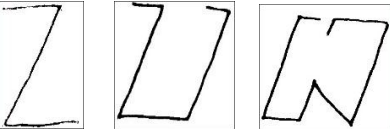
Pemotongan dilakukan untuk setiap baris (*horizontal*) pada citra input terlebih dahulu, kemudian melakukan pemotongan setiap kolom (*vertical*) pada setiap citra hasil pemotongan secara baris.

Pemotongan untuk setiap baris (*horizontal*) dilakukan dengan menelusuri *pixel* citra dari *pixel* baris ke-1. Penelusuran terus dilakukan sampai dalam suatu baris ditemukan *pixel* objek, kemudian baris itu ditandai sebagai label awal pemotongan. Selanjutnya lakukan penelusuran kembali sampai dalam satu baris *pixel* citra tidak ditemukan *pixel* objek, kemudian tandai baris sebelum itu sebagai label akhir pemotongan. Label awal dan label akhir ini yang digunakan sebagai acuan untuk memotong citra setiap baris (*horizontal*). Lakukan hal yang sama untuk pemotongan baris selanjutnya.

Dari matriks biner citra masukan pada Tabel 3.5 dilakukan penelusuran per baris dan pertama ditemukan *pixel* objek pada baris ke-14. Tandai baris ke-14, setelah itu telusuri lagi per baris sehingga ditemukan baris yang tidak memiliki *pixel* objek, yaitu pada baris ke-48. Maka tandai baris sebelumnya yaitu baris ke-47. Berikut adalah matriks hasil pemotongan baris.

Pada proses segmentasi fitur spasi tidak digunakan sehingga proses segmentasi hanya dapat membaca *pixel* citra yang berwarna hitam. Proses segmentasi citra masukan dapat dilihat pada Gambar 3.7.

Tabel 3. 8 Hasil Proses Segmentasi

No	Pemotongan Baris	Pemotongan Kolom
1		

Berikut adalah hasil dari matriks citra masukan yang sudah disegmentasi. Pada Tabel 3.9, 3.10 dan 3.11 adalah matriks NA, PA, dan BA hasil segmentasi yang telah dibinerisasi.

Tabel 3. 9 Matriks Citra Huruf NA Hasil Segmentasi

x/y	1	2	3	4	...	21
1	0	0	0	0	...	0
2	0	0	0	1	...	0
3	1	1	1	1	...	0
4	0	0	0	0	...	0
...
34	1	1	1	1	...	0

Tabel 3. 10 Matriks Citra Huruf PA Hasil Segmentasi

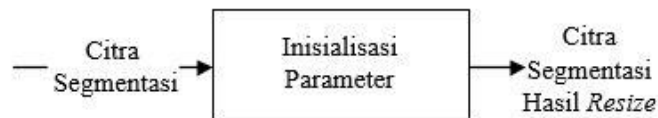
x/y	1	2	3	4	...	30
1	0	0	0	0	...	0
2	0	0	0	0	...	0
3	0	0	0	0	...	0
4	0	0	0	0	...	1
...
34	0	0	0	0	...	0

Tabel 3. 11 Matriks Citra Huruf BA Hasil Segmentasi

x/y	1	2	3	4	...	35
1	0	0	0	0	...	0
2	0	0	0	0	...	0
3	0	0	0	0	...	0
4	0	0	0	0	...	0
...
34	0	0	0	0	...	0

1.2.2.6 Resize

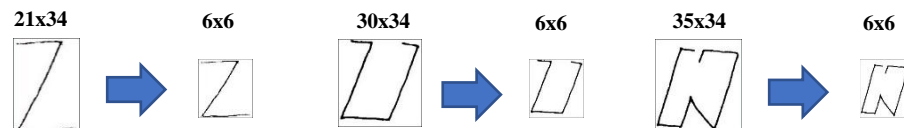
Resize adalah proses mengubah ukuran suatu citra menjadi lebih besar atau kecil dari ukuran citra awal dengan ukuran yang telah ditetapkan sebelumnya. Berikut adalah alur proses *resize* pada gambar 3.10.



Gambar 3. 10 Blok Diagram Resize

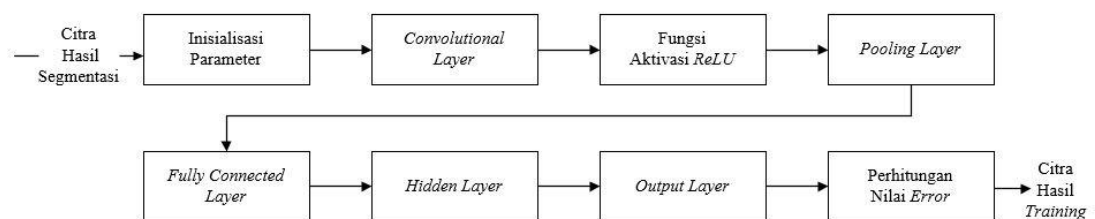
Proses ini dilakukan agar input yang di proses pada saat metode ekstraksi ciri seragam atau konsisten. Dalam penelitian ini karakter-karakter yang sudah tersegmen pada tahap segmentasi di-*resize* menjadi ukuran 6×6 *pixel*, ukuran tersebut digunakan berdasarkan penelitian yang telah dilakukan sebelumnya agar ukuran setiap segmentasi bernilai sama. segmentasi bernilai sama. Gambar 3.11 merupakan beberapa *pixel*.

Gambar 3. 11 Resize Citra Masukan



3.2.3 CNN Training

CNN *Training* terdiri dari beberapa tahap, yaitu tahap inisialisasi, tahap *feedforward*, tahap *backpropagation*, dan tahap *update* bobot. Masukan data berupa citra hitam putih dan keluaran adalah klasifikasi huruf, berikut merupakan alur dari CNN training seperti pada Gambar 3.12.



Gambar 3. 12 Blok Diagram Cnn Training

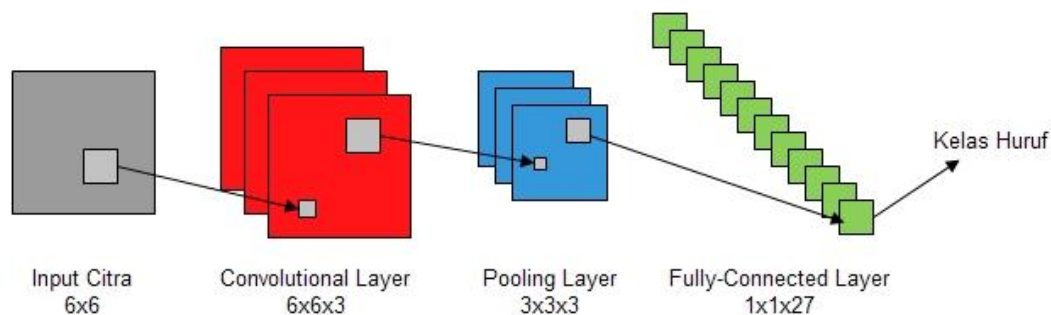
Data masukan yang digunakan dalam analisis ini adalah data citra hitam putih huruf NA pada tabel Tabel 3.12 yang telah di-*resize* sehingga menjadi berukuran 6*6 pixel. Berikut adalah matriks citranya.

Tabel 3. 12 Matriks Citra Huruf Masukan NA

x/y	1	2	3	4	5	6
1	1	1	1	1	1	1
2	0	0	0	0	1	0
3	0	0	0	1	0	0
4	0	0	1	0	0	0
5	0	1	0	0	0	0
6	1	1	1	1	1	1

Dalam analisis ini akan diperiksa apakah citra masukan merupakan huruf NA, PA, atau BA. CNN yang digunakan memiliki 1 *convolutional layer* dengan 3 buah filter, 1 *pooling layer*, dan 1 *fully-connected layer*.

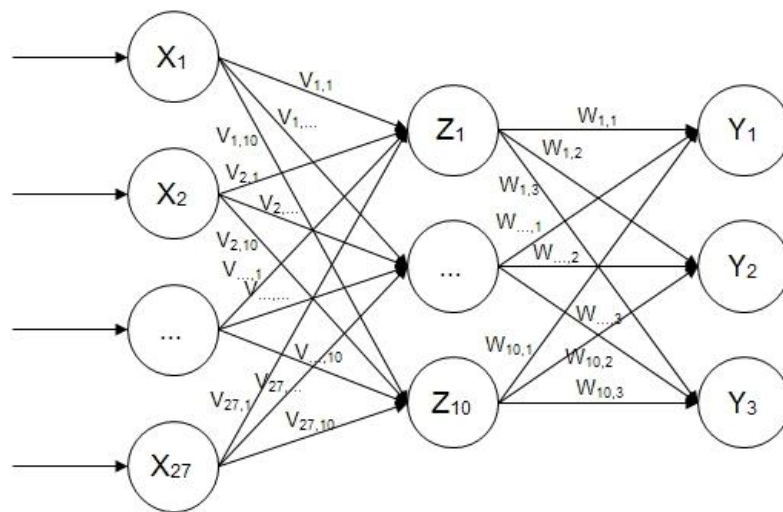
Dalam *convolutional layer* input citra A berukuran 6*6 pixel akan diberikan operasi konvolusi dengan 3 buah filter F berukuran 3*3 pixel yang akan menghasilkan *feature map* FM berukuran 6*6 pixel sebanyak 3 buah. Setelah itu dalam *pooling layer*, *feature map* FM akan diberikan operasi *max-pooling* dengan ukuran filter 2*2 pixel, yang akan menghasilkan *feature map* P dengan ukuran 3*3 pixel sebanyak 3 buah. Setelah itu ketiga *feature map* P akan dipisah menjadi matriks berukuran 1*1 sebanyak 27 buah, dengan kata lain direntangkan menjadi sebuah vektor X dengan jumlah baris 27 dan kolom 1. Vektor ini akan dimasukkan ke dalam *fully-connected layer* untuk memprediksi kelas huruf dari citra yang diinputkan. Berikut adalah arsitektur CNN yang digunakan dalam analisis ini.



Gambar 3. 13 Arsitektur CNN Analisis

Fully-connected layer adalah sebuah *neural network* yang terdiri dari 1 *input layer* dengan 27 *node*, 1 *hidden layer* dengan 10 *node*, dan 1 *output layer* dengan 3 *node*.

Vektor X yang dihasilkan dari *layer* sebelumnya akan menjadi *input layer*. Tiap *node input layer* X akan dikalikan dengan *weight* V dan hasilnya akan menjadi *node hidden layer* Z . Tiap *node hidden layer* Z akan dikalikan dengan *weight* W dan hasilnya akan menjadi hasil prediksi kelas huruf Y berbentuk matriks *one-hot*. Berikut adalah arsitektur *fully-connected layer* yang digunakan dalam analisis ini.



Gambar 3. 14 Arsitektur *Fully-Connected Layer* CNN Analisis

Keluaran yang dihasilkan adalah vektor *one-hot* kelas huruf, yang berisi probabilitas citra masukan adalah sebuah kelas huruf dan bukan kelas huruf yang lain. [15] Karena kelas huruf yang digunakan 3 buah (NA, PA, BA), maka vektor *one-hot* yang digunakan memiliki 3 komponen. Jumlah *node output layer* Y pada *fully-connected layer* CNN juga berjumlah 3 buah, sehingga kita dapat memetakan nilai Y pada vektor *one-hot* kelas huruf, seperti pada tabel berikut.

Tabel 3. 13 Contoh Vektor One-Hot Kelas Huruf

Kelas Huruf	Output Node
NA	Y_1
PA	Y_2
BA	Y_3

Maka nilai probabilitas citra masukan adalah huruf NA terdapat pada nilai Y_1 , nilai probabilitas citra masukan adalah huruf PA terdapat pada nilai Y_2 , dan nilai probabilitas citra masukan adalah huruf BA terdapat pada nilai Y_3 .

3.2.3.1 Inisialisasi CNN

CNN *Training* terdiri dari beberapa tahap, yaitu tahap inisialisasi bobot, tahap *feedforward*, tahap *backpropagation*, dan tahap *update* bobot. Masukan data berupa citra hitam putih dan keluaran adalah klasifikasi kelas huruf. Inisialisasi CNN meliputi inisialisasi parameter pelatihan dan inisialisasi bobot.

1. Inisialisasi Parameter Pelatihan

Parameter pelatihan adalah parameter-parameter yang menentukan kinerja *training* CNN. Parameter-parameter yang diinisialisasi adalah jumlah maksimum *epoch* (iterasi pelatihan), *learning rate* (laju pembelajaran), dan minimum *error*. [25]

2. Inisialisasi Bobot

Dalam analisis ini kelas bobot-bobot yang akan diinisialisasi adalah nilai-nilai awal bobot dan bias filter *convolutional layer* F, bobot dan bias *hidden layer* V, dan bobot dan bias *output layer fully-connected layer* W. Semua bobot diisi dengan nilai awal rentang nilai antara -0,5 dan 0,5 secara acak, sedangkan semua bias diisi dengan nilai awal 0.

Filter *convolutional layer* F berjumlah 3 buah dengan masing-masing berukuran 3*3 pixel, maka jumlah bobot yang harus diisi adalah 27 buah. Berikut adalah matriks-matriks inisialisasi filternya.

Tabel 3. 14 Inisialisasi Matriks-Matriks Filter F

Filter F[1]				Filter F[2]				Filter F[3]			
x/y	1	2	3	x/y	1	2	3	x/y	1	2	3
1	0,42	0,49	0,01	1	-0,36	-0,19	-0,29	1	-0,24	0,28	-0,46
2	0,48	-0,33	0,18	2	0,29	0,44	-0,42	2	0,15	0,09	-0,45
3	0,37	0,13	-0,34	3	0,26	-0,23	-0,44	3	-0,5	0,22	0,5

Sedangkan bias-bias filter *convolutional layer* diisi dengan nilai awal 0.

1. Bias F[1] ($bF[1]$) = 0
2. Bias F[2] ($bF[2]$) = 0

3. Bias $F[3]$ ($bF[3]$) = 0

Matriks-matriks filter F akan digunakan pada proses konvolusi citra masukan pada *convolutional layer*.

Matriks inisialisasi nilai awal bobot-bobot dan bias *hidden layer* V dan *output layer* W dari *fully-connected layer* dapat dilihat pada tabel-tabel berikut. Terdapat 27 *node input layer* X dan 10 *node hidden layer* Z , sehingga jumlah bobot yang harus diisi untuk matriks bobot *hidden layer* V adalah 270 dan jumlah bias adalah 10. Berikut adalah matriks V . $V_{0,y}$ menandakan nilai bias (semua diberi nilai 0) dan $V_{x,y}$ menandakan nilai bobot.

Tabel 3. 15 Inisialisasi Matriks Bobot *Hidden Layer* V

x/y	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0,36	0,46	-0,43	-0,35	0,06	0,25	-0,16	-0,41	0,47	-0,19
2	-0,17	0,47	-0,48	0,31	0,37	-0,19	-0,33	0,24	-0,24	-0,36
3	-0,14	0,13	-0,49	-0,1	0,34	-0,46	0,26	-0,11	0,42	-0,46
4	0,25	-0,05	-0,22	-0,43	-0,12	0	-0,14	0,23	0,25	0,41
5	-0,21	0,04	0,01	-0,43	-0,17	-0,34	-0,17	-0,19	0,07	0,35
6	0,32	-0,19	-0,47	-0,36	0,42	-0,38	0,19	-0,24	-0,19	0,28
7	0,04	0,16	0,23	-0,21	-0,27	-0,32	-0,13	0,3	0,42	0,05
8	-0,44	-0,23	0,02	0,48	0,04	0,22	0,45	-0,38	0,2	-0,05
9	0,49	-0,08	-0,16	-0,14	0,47	0,19	0,5	0,29	-0,12	-0,34
10	0,22	-0,17	-0,14	0,46	0,4	-0,44	-0,07	-0,25	-0,14	0,05
11	-0,33	0,18	0,41	0,13	-0,23	-0,13	-0,18	0,16	0,46	0,15
12	-0,1	-0,04	-0,41	-0,06	0,2	-0,48	0,07	-0,13	0,4	-0,48
13	0,31	-0,37	-0,23	-0,48	-0,32	0,09	0,47	-0,02	-0,1	-0,05
14	-0,09	0,18	0,45	0,22	0,31	0,07	-0,37	-0,36	0,31	-0,13
15	0,09	0,37	0,16	-0,01	0,39	-0,28	0,21	-0,35	0,29	-0,31
16	-0,44	0,26	-0,19	-0,35	-0,43	0,22	-0,5	-0,22	-0,01	0,3
17	0,11	0,4	-0,04	-0,33	-0,11	0,19	0,45	-0,5	0,45	-0,33
18	-0,14	0,25	-0,42	-0,11	-0,26	-0,4	-0,09	0,44	0,34	-0,16
19	-0,13	-0,04	0	-0,13	0,19	0,39	-0,12	0,21	-0,01	-0,47
20	-0,15	-0,5	0,27	-0,43	-0,09	-0,09	0,31	-0,01	-0,27	-0,2
21	-0,24	0,05	0,01	0,14	-0,08	-0,45	0	-0,38	-0,41	0,41
22	0,06	-0,46	-0,19	0,42	-0,03	0,16	0,49	0,24	0,35	-0,06

x/y	1	2	3	4	5	6	7	8	9	10
23	-0,23	-0,26	-0,17	-0,12	0,23	0,33	0,11	-0,03	0,13	-0,07
24	-0,19	0,17	-0,46	0,43	-0,32	0,16	-0,26	0,5	0,26	-0,27
25	-0,3	0,38	-0,18	-0,29	0,49	0,23	-0,34	0,42	0,18	-0,19
26	0,08	0,26	0,1	-0,44	0,44	0,13	0,44	-0,47	0,03	-0,33
27	0,27	-0,09	-0,21	-0,23	0,1	-0,15	-0,44	0,36	-0,45	0,15

Matriks V akan digunakan sebagai bobot dari *input layer* ke *hidden layer* pada *fully-connected layer*.

Terdapat 10 *node hidden layer* Z dan 3 *node output layer* Y, sehingga jumlah bobot yang harus diisi untuk matriks W adalah 30 dan jumlah bias adalah 3. Berikut adalah matriks W. $W_{0,y}$ menandakan nilai bias (semua diberi nilai 0) dan $V_{x,y}$ menandakan nilai bobot.

Tabel 3. 16 Matriks Inisialisasi Hidden Layer W

x/y	1	2	3
0	0	0	0
1	-0,13	0,35	0,3
2	0,42	0,28	-0,13
3	0,31	0,24	0,33
4	0,39	-0,06	-0,36
5	-0,09	-0,12	-0,25
6	0,34	-0,35	0,47
7	-0,31	-0,21	0,36
8	-0,43	-0,26	-0,49
9	0,13	-0,28	0,5
10	0,24	-0,09	-0,08

Matriks V akan digunakan sebagai bobot dari *hidden layer* ke *output layer* pada *fully-connected layer*.

3.2.3.2 Feedforward

Pada tahap *feedforward* akan dilakukan proses CNN dari awal masukan melewati *convolutional*, *pooling*, dan *fully-connected layer* hingga dihasilkan sebuah vektor *one-hot* klasifikasi kelas huruf.

1. *Convolutional Layer*

Pada *convolutional layer* akan dilakukan konvolusi antara matriks citra masukan dengan matriks-matriks filter F. Filter-filter ini akan digeser ke seluruh permukaan gambar sehingga menghasilkan keluaran matriks *feature map* FM. Terdapat beberapa *hyperparameter* dalam *convolutional layer*, yaitu jumlah filter, *stride*, dan *zero-padding*. [25]

- a. Jumlah filter F berjumlah sebanyak 3 buah dengan ukuran 3*3 pixel.
- b. *Stride* adalah seberapa jauh jarak pergeseran filter setiap perkalian. Jika *stride* adalah 2, maka filter akan bergeser 2 pixel ke kanan lalu ke bawah. Dalam analisis ini *stride* yang digunakan adalah 1.
- c. *Zero-padding* adalah jumlah lapisan pixel berisi nilai 0 yang ditambahkan ke setiap sisi matriks masukan. Dalam analisis ini *zero-padding* yang digunakan adalah 1.

Dengan *hyperparameter* di atas maka akan dihasilkan *feature map* FM dengan ukuran yang didapat dari rumus berikut [25]:

$$\begin{aligned}
 \text{feature size} &= \text{ukuran feature map} \\
 \text{input size (ukuran matriks masukan)} &= 6 \\
 \text{filter size (ukuran filter)} &= 3 \\
 \text{pad (zero padding)} &= 1 \\
 \text{str (stride)} &= 1 \\
 \text{feature size} &= \frac{\text{input size} - \text{filter size} + 2\text{pad}}{\text{str}} + 1 \\
 &= \frac{6 - 3 + 2 \cdot 1}{1} + 1 \\
 &= 6
 \end{aligned}$$

Maka *feature map* FM yang dihasilkan berukuran 6*6 pixel.

Langkah-langkah yang dilakukan pada *convolutional layer* adalah sebagai berikut. Pertama-tama di setiap sisi matriks citra masukan akan ditambah pixel berisi nilai 0 sesuai dengan nilai *zero-padding*, sebagai berikut.

Tabel 3. 17 Matriks A (Matriks Citra Masukan Dengan Zero-Padding)

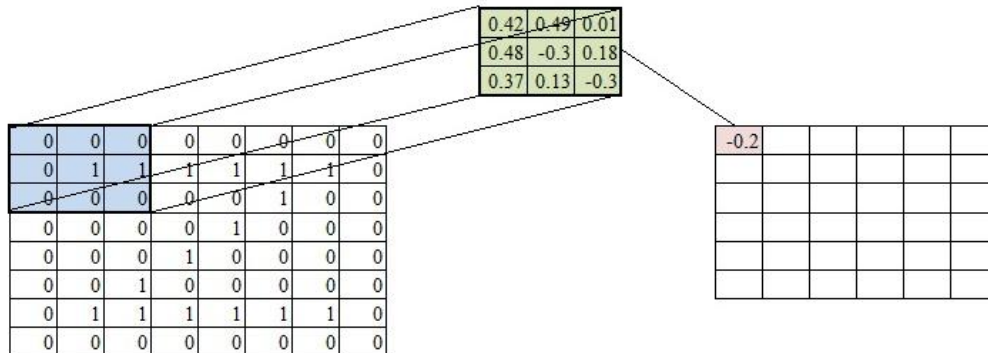
x/y	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	1	1	1	1	1	1	0
3	0	0	0	0	0	1	0	0
4	0	0	0	0	1	0	0	0
5	0	0	0	1	0	0	0	0
6	0	0	1	0	0	0	0	0
7	0	1	1	1	1	1	1	0
8	0	0	0	0	0	0	0	0

Setelah itu lakukan operasi konvolusi antara citra masukan dengan filter F, dimulai dari pojok kiri atas, lalu bergeser ke kanan lalu ke bawah. Tambah setiap hasil dengan bias filter bF. Lakukan untuk setiap filter F. Filter-filter F yang digunakan dapat dilihat pada Tabel 3.17.

Sebagai contoh, akan dilakukan operasi konvolusi [25] antara citra masukan A dengan *zero-padding* dengan filter F[1]. Posisikan filter F[1] di atas citra masukan A. Lalu kalikan tiap pixel A dengan pixel filter F[1] yang sesuai. Setelah itu jumlahkan semua hasilnya, dan tambah dengan bias bF[1] untuk menjadi pixel pertama pada *feature map* FM[1]. Berikut adalah perhitungannya.

$$\begin{aligned}
 FM[1]_{1,1} &= (A_{1,1} * F[1]_{1,1} + A_{1,2} * F[1]_{1,2} + A_{1,3} * F[1]_{1,3} + A_{2,1} * F[1]_{2,1} \\
 &\quad + A_{2,2} * F[1]_{2,2} + A_{2,3} * F[1]_{2,3} + A_{3,1} * F[1]_{3,1} + A_{3,2} \\
 &\quad * F[1]_{3,2} + A_{3,3} * F[1]_{3,3}) + bF[1] \\
 &= (0 * 0.42 + 0 * 0.49 + 0 * 0.01 + 0 * 0.48 + 1 * -0.3 + 1 * \\
 &\quad 0.18 + 0 * 0.37 + 0 * 0.13 + 0 * -0.3) + 0 \\
 &= -0.2
 \end{aligned}$$

Berikut adalah visualiasi operasi konvolusi filter F[1] dengan matriks masukan A untuk pixel FM[1]_{1,1}.

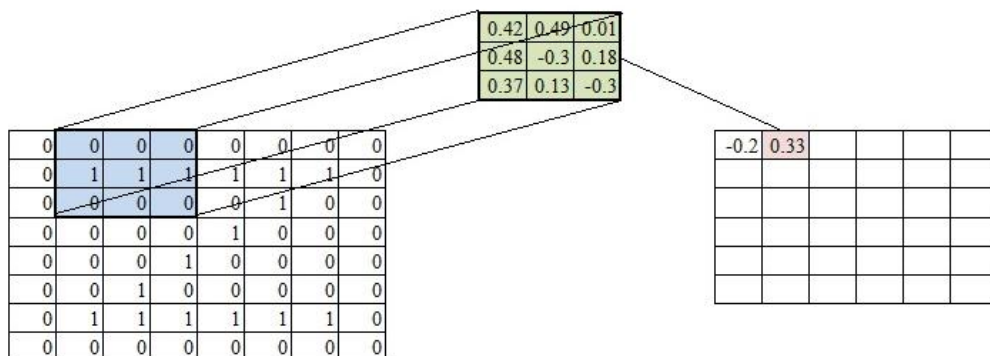


Gambar 3. 15 Contoh 1 Operasi Dot Citra Input dan Filter F[1]

Setelah itu geser filter ke kanan sebanyak 1 pixel sesuai *stride*, dan lakukan kembali operasi konvolusi.

$$\begin{aligned}
 FM[1]_{1,2} &= (A_{1,2} * F[1] + A_{1,3} * F[1]_{1,2} + A_{1,4} * F[1]_{1,3} + A_{2,2} * F[1]_{2,1} \\
 &\quad + A_{2,3} * F[1]_{2,2} + A_{2,4} * F[1]_{2,3} + A_{3,2} * F[1]_{3,1} + A_{3,3} \\
 &\quad * F[1]_{3,2} + A_{3,4} * F[1]_{3,3}) + bF[1] \\
 &= (0 * 0.42 + 0 * 0.49 + 0 * 0.01 + 1 * 0.48 + 1 * -0.3 + 1 * \\
 &\quad 0.18 + 0 * 0.37 + 0 * 0.13 + 0 * -0.3) + 0 \\
 &= 0.33
 \end{aligned}$$

Berikut adalah visualisasi operasi konvolusi filter F[1] dengan matriks masukan A untuk pixel FM[1]_{1,2}.



Gambar 3. 16 Contoh 2 Operasi Dot Citra Input dan Filter F[1]

Lakukan pergeseran hingga ke seluruh bagian citra, hingga didapat hasil *feature map* sebagai berikut.

Tabel 3. 18 Feature Map FM[1]

x/y	1	2	3	4	5	6
1	-0.2	0.33	0.33	0	0.46	0.52
2	0.5	0.92	0.58	1.23	0.96	1.39
3	0	-0.3	0.31	0.05	0.97	0.42
4	-0.3	0.31	0.05	0.97	0.42	0
5	0	-0.2	1.13	0.58	0.16	0.5
6	-0.1	0.82	0.75	0.33	0.33	0.15

Ulangi untuk setiap filter, sehingga menghasilkan 3 *feature map* FM. Hasil *feature map* FM[2] untuk filter F[2] dan *feature map* FM[3] untuk F[3] dapat dilihat pada tabel 3.19 dan 3.20.

Tabel 3. 19 Feature Map FM[2]

x/y	1	2	3	4	5	6
1	0.02	0.31	0.31	-0.1	0.08	0.99
2	-0.5	-0.8	-1.3	-1.5	-0.1	-0.3
3	0	-0.4	-0.7	0.41	0.1	-0.4
4	-0.4	-0.7	0.41	0.1	-0.4	0
5	-1.1	-0.3	-0.3	-0.8	-0.4	0.03
6	-0.3	0.12	-0.1	0.31	0.31	0.73

Tabel 3. 20 Feature Map FM[3]

x/y	1	2	3	4	5	6
1	-0.36	-0.21	-0.21	0.29	0.01	-0.26
2	-0.18	-0.42	0.08	-0.65	-0.83	0.19
3	0	0.5	-0.23	-0.87	0.43	-0.24
4	0.5	-0.23	-0.87	0.43	-0.24	0
5	0.27	-0.15	0.65	-0.02	0.22	-0.28
6	-0.82	0.07	-0.45	-0.21	-0.21	0.24

Lalu jalankan fungsi aktivasi pada tiap *feature map* FM. Setiap pixel *feature map* FM akan dimasukkan ke dalam fungsi aktivasi ReLU, dimana setiap pixel yang memiliki nilai < 0 akan dijadikan 0, dengan rumus $ReLU(x) = \max(0, x)$. *Feature map* FM yang telah dimasukkan ke dalam fungsi aktivasi ReLU akan menghasilkan *feature map* R. Ulangi untuk tiap *feature map* FM, sehingga akan menghasilkan tiga *feature map* R. Berikut adalah *feature map* R[1], R[2], dan R[3], dengan font tebal menandakan bahwa pixel tersebut awalnya adalah < 0 pada *feature map* FM.

Tabel 3. 21 Feature Map ReLU R[1]

x/y	1	2	3	4	5	6
1	0	0.33	0.33	0	0.46	0.52
2	0.5	0.92	0.58	1.23	0.96	1.39
3	0	0	0.31	0.05	0.97	0.42
4	0	0.31	0.05	0.97	0.42	0
5	0	0	1.13	0.58	0.16	0.5
6	0	0.82	0.75	0.33	0.33	0.15

Tabel 3. 22 Feature Map ReLU R[2]

x/y	1	2	3	4	5	6
1	0.02	0.31	0.31	0	0.08	0.99
2	0	0	0	0	0	0
3	0	0	0	0.41	0.1	0
4	0	0	0.41	0.1	0	0
5	0	0	0	0	0	0.03
6	0	0.12	0	0.31	0.31	0.73

Tabel 3. 23 Feature Map ReLU R[3]

x/y	1	2	3	4	5	6
1	0	0	0	0.29	0.01	0
2	0	0	0.08	0	0	0.19
3	0	0.5	0	0	0.43	0
4	0.5	0	0	0.43	0	0
5	0.27	0	0.65	0	0.22	0
6	0	0.07	0	0	0	0.24

Feature map R[1], R[2] dan R[3] akan digunakan pada tahap selanjutnya yaitu *pooling layer*.

2. Pooling Layer

Pooling layer bertujuan untuk mengecilkan ukuran *feature map* R. Dalam penelitian ini jenis *pooling* yang digunakan adalah *max pooling*, yaitu memilih nilai maksimum dalam suatu jendela. Pemilihan ini akan diulangi dengan menggeser jendela ke seluruh permukaan citra sehingga menghasilkan keluaran matriks *feature map* P yang berisi nilai-nilai maksimum yang terpilih. Lakukan untuk ketiga *feature map* R, hingga menghasilkan *feature map* P[1], P[2], dan P[3].

Terdapat beberapa *hyperparameter* dalam *pooling layer*, yaitu ukuran jendela dan *stride*. Dalam *pooling layer* ini ukuran jendela yang digunakan adalah 2*2 pixel dengan *stride* adalah 2. Dengan *hyperparameter* di atas maka akan dihasilkan *feature map* P dengan ukuran yang didapat dari rumus berikut: [25]

feature size = ukuran *feature map*

input size = ukuran matriks masukan = 6

window size = ukuran jendela = 2

str = *stride* = 2

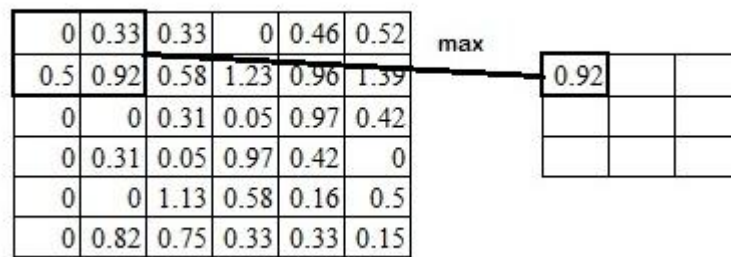
$$\begin{aligned} \text{feature size} &= \frac{\text{input size} - \text{window size}}{\text{str}} + 1 \\ &= \frac{6-2}{2} + 1 \\ &= 3 \end{aligned}$$

Maka *feature map* P yang dihasilkan berukuran 3*3 pixel.

Sebagai contoh, akan dilakukan operasi *max pooling* pada *feature map* R[1] yang akan menghasilkan *feature map* P[1]. Letakkan jendela pada pojok kiri atas *feature map* R[1]. Berarti *pixel* yang diperiksa adalah R[1]_{1,1}, R[1]_{1,2}, R[1]_{2,1}, R[1]_{2,2}. Berikut adalah perhitungan operasi *max pooling* untuk P[1]_{1,1}.

$$\begin{aligned} P[1]_{1,1} &= \max(R[1]_{1,1}, R[1]_{1,2}, R[1]_{2,1}, R[1]_{2,2}) \\ &= \max(0, 0.33, 0.5, 0.92) \\ &= 0.92 \end{aligned}$$

Dari keempat *pixel* tersebut, yang memiliki nilai tertinggi adalah R[1]_{2,2} = 0,92. Sehingga nilai *pixel* P[1]_{1,1} adalah 0,92. Berikut adalah visualisasi operasi *max pooling* pada *feature map* R[1].

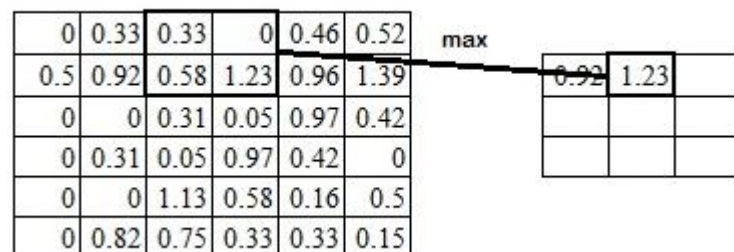


Gambar 3. 17 Contoh 1 Operasi Max Pooling P[1]

Setelah itu geser filter ke kanan sebanyak 2 pixel sesuai *stride*, dan lakukan kembali operasi *max pooling*.

$$\begin{aligned}
 P[1]_{1,2} &= \max(R[1]_{1,3}, R[1]_{1,4}, R[1]_{2,3}, R[1]_{2,4}) \\
 &= \max(0.33, 0, 0.58, 1.23) \\
 &= 1.23
 \end{aligned}$$

Dari $R[1]_{1,3}, R[1]_{1,4}, R[1]_{2,3}, R[1]_{2,4}$, yang memiliki nilai tertinggi adalah $R[1]_{2,4} = 1.23$. Sehingga nilai pixel $P[1]_{1,2}$ adalah 1,23.



Gambar 3. 18 Contoh 2 Operasi Max Pooling P[1]

Lakukan pergeseran hingga ke seluruh bagian citra, hingga didapat hasil *feature map* P[1] sebagai berikut.

Tabel 3. 24 Feature Map P[1]

x/y	1	2	3
1	0.92	1.23	1.39
2	0.31	0.97	0.97
3	0.82	1.13	0.5

Ulangi untuk setiap *feature map*. Hasil *feature map* P[2] untuk *feature map* R[2] dan hasil *feature map* P[3] untuk *feature map* R[3] dapat dilihat pada tabel 3.25 dan 3.26.

Tabel 3. 25 Feature Map P[2]

x/y	1	2	3
1	0.31	0.31	0.99
2	0	0.41	0.1
3	0.12	0.31	0.73

Tabel 3. 26 Feature Map P[3]

x/y	1	2	3
1	0	0.29	0.19
2	0.5	0.43	0.43
3	0.27	0.65	0.24

Feature map P[1], P[2] dan P[3] akan digunakan pada tahap selanjutnya yaitu *fully-connected layer*.

3. Fully-Connected Layer

Dalam CNN *fully-connected layer* adalah sebuah *neural network* yang memiliki *input layer*, *hidden layer* dan *output layer*.

3.2.3.2.1 Input Layer

Input layer X adalah penggabungan matriks *feature map* P[1], P[2] dan P[3] yang didapat dari *pooling layer* dan direntangkan menjadi sebuah vektor sepanjang jumlah pixel keseluruhan ketiga *feature map*. Total pixel *feature map* P[1], P[2], P[3] adalah 27, sehingga *input layer* X memiliki 27 *node*. Berikut adalah *input layer fully-connected layer*.

Tabel 3. 27 Input Layer X

x_i	Nilai	x_i	Nilai	x_i	Nilai
x_1	0,92	x_{10}	0,31	x_{19}	0
x_2	1,23	x_{11}	0,31	x_{20}	0.29
x_3	1,39	x_{12}	0,99	x_{21}	0.19
x_4	0,31	x_{13}	0	x_{22}	0.5
x_5	0,97	x_{14}	0,41	x_{23}	0.43
x_6	0,97	x_{15}	0,1	x_{24}	0.43
x_7	0,12	x_{16}	0,12	x_{25}	0.27
x_8	0,31	x_{17}	0,31	x_{26}	0.65
x_9	0,73	x_{18}	0,73	x_{27}	0.24

Nilai-nilai *input layer* X akan digunakan pada perhitungan pada *hidden layer* Z.

3.2.3.2.2 Hidden Layer

Setiap *node* di *input layer* X akan mengirimkan sinyal input kepada setiap *hidden layer* Z. Setiap *node* X_i akan dikalikan dengan bobot $V_{j,i}$ lalu ditambahkan dengan bias $V_{0,i}$ untuk menghasilkan nilai masukan z_in_i . Nilai-nilai bobot V dapat dilihat pada Tabel 3.28. Contoh penghitungan masukan *hidden layer* z_in adalah sebagai berikut.

$$z_in_i = \sum_{j=1}^n X_j * V_{j,i} + V_{0,i}$$

z_in_i = masukan untuk *node hidden layer* Z ke-i dengan jumlah node n

X_j = *node* X ke-j

$V_{j,i}$ = *weight* V untuk *node* X_j dan *node* Z_i

$V_{0,i}$ = bias V untuk *node* z_in_i

$$\begin{aligned} z_in_1 &= \sum_{j=1}^{10} X_j * V_{j,1} + V_{0,1} \\ &= (X_{1,1} * V_{1,1} + X_{1,2} * V_{2,1} + X_{1,3} * V_{3,1} + \dots + X_{1,27} * V_{27,1}) + V_{0,1} \\ &= (0.92 * 0.36 + 1.23 * -0.2 + 1.39 * -0.1 + \dots + 0.24 * 0.27) + 0 \\ &= -0.158 \end{aligned}$$

Hidden layer Z memiliki 10 *node*, sehingga akan dihasilkan 10 nilai z_in . Hasil penghitungan z_in yang lainnya dapat dilihat pada Tabel 3.28.

Tabel 3. 28 Matriks z_in

z_in_i	Nilai
z_in_1	-0.5935
z_in_2	1.1101
z_in_3	-2.8335
z_in_4	-0.8653
z_in_5	1.7356
z_in_6	-1.8285
z_in_7	0.6701
z_in_8	-0.6772
z_in_9	2.3769
z_in_{10}	-1.771

Setelah itu hitung nilai keluaran Z dengan mengaktifkan nilai masukan z_{in} menggunakan fungsi aktivasi ReLU, dengan rumus $ReLU(x) = \max(0, x)$.

$$Z_i = \max(0, z_{in_i})$$

Z_i = keluaran untuk *node hidden layer* ke- i

z_{in_i} = masukan untuk *node hidden layer* ke- i

$$Z_1 = \max(0, z_{in_1})$$

$$= \max(0, -0.158)$$

$$= 0$$

Hasil penghitungan Z yang lainnya dapat dilihat pada Tabel 3.29.

Tabel 3. 29 Matriks Z

Z_i	Nilai
Z_1	0
Z_2	1.1101
Z_3	0
Z_4	0
Z_5	1.7356
Z_6	0
Z_7	0.6701
Z_8	0
Z_9	2.3769
Z_{10}	0

Nilai-nilai matriks *hidden layer* Z akan digunakan pada perhitungan *output layer* Y .

3.2.3.2.3 Output Layer

Setiap *node* di *output layer* Z_j akan mengirimkan sinyal input kepada setiap *output layer* Y_i . Setiap *node* Z akan dikalikan dengan bobot $W_{j,i}$ lalu ditambahkan dengan bias $W_{0,i}$. Nilai-nilai bobot W dapat dilihat pada Tabel 3.11. Penghitungan masukan *output layer* y_{in} adalah sebagai berikut.

$$y_{in_i} = \sum_{j=1}^m Z_j * W_{j,i} + W_{0,i}$$

y_{in_i} = masukan untuk *node hidden layer* Z ke- i dengan jumlah *node* m

Z_j = *node* Z ke- j

$W_{j,i}$ = *weight* W untuk *node* Z_j dan *node* Y_i

$W_{0,i}$ = bias W untuk *node* y_{in_i}

$$y_{in_i} = \sum_{j=1}^m Z_j * W_{j,i} + W_{0,i}$$

$$\begin{aligned}
y_{in_1} &= \sum_{j=1}^3 Z_{1,j} * W_{j,1} + W_{0,1} \\
&= (Z_{1,1} * W_{1,1} + Z_{1,2} * W_{2,1} + Z_{1,3} * W_{3,1} + \dots + Z_{1,10} * W_{10,1}) + W_{0,1} \\
&= (0 * -0.1 + 1.1101 * 0.42 + 0 * 0.31 + 0 * 0.39 + \dots + 0 * 0.24) + 0 \\
&= 1.50878
\end{aligned}$$

Output layer Y memiliki 10 *node*, sehingga akan dihasilkan 3 nilai y_{in} . Hasil penghitungan y_{in} yang lainnya dapat dilihat pada Tabel 3.30.

Tabel 3. 30 Matriks y_{in}

y_{in_i}	Nilai
y_{in_1}	1.50878
y_{in_2}	0.49475
y_{in_3}	2.3431

Setelah itu aktifkan nilai keluaran Y dengan menggunakan fungsi aktivasi

softmax, dengan rumus $softmax(x)_i = \frac{e^{x_i}}{\sum_{i=1}^m e^M}$.

$$Y_i = \frac{e^{y_{in_i}}}{\sum_{i=1}^m e^M}$$

Y_i = keluaran untuk *node output layer* ke- i

y_{in_i} = masukan untuk *node output layer* ke- i

M = semua masukan untuk *node output layer*, berjumlah m buah

$$Y_1 = \frac{e^{y_{in_1}}}{\sum_{i=1}^m e^M} = \frac{e^{1.50878}}{e^{1.50878} + e^{0.49475} + e^{2.3431}} = 0.347115605$$

Hasil penghitungan Y yang lainnya dapat dilihat pada Tabel 3.31.

Tabel 3. 31 Matriks Y

Y_i	Nilai
Y_1	0.347115605
Y_2	0.113824397
Y_3	0.539059999

Dari hasil keluaran Y dapat dipetakan klasifikasi kelas huruf sebagai berikut.

Tabel 3. 32 Matriks Y

Huruf	Tingkat Keyakinan
$Y_1 = \text{NA}$	35%
$Y_2 = \text{PA}$	11%
$Y_3 = \text{BA}$	54%

3.2.3.3 Backpropagation

Pada tahap *backpropagation* [14] akan dilakukan pelatihan pada kelas huruf NA. Pada tahap ini akan dilakukan proses penyesuaian tiap *weight* dan *bias* berdasarkan *error* yang didapat pada tahap *feedforward*. Pertama akan dihitung gradien dari *loss function* terhadap semua parameter (*weight* dan *bias*) yang ada dengan mencari turunan parsial (*partial derivative*) dari fungsi tersebut. Setelah itu *update* semua parameter dengan menggunakan *Stochastic Gradient Descent (SGD)* [13].

1. Penghitungan *loss function*

Loss function adalah sebuah fungsi yang mengukur seberapa bagus performa dari *neural network* dalam melakukan prediksi terhadap target. *Loss function* akan menghitung *loss*, yaitu selisih dari nilai *output* dengan nilai target yang diharapkan. Untuk masalah klasifikasi, *loss function* yang biasa digunakan adalah *cross-entropy loss function* :

$$L = - \sum_i^m t_i \log(Y_i)$$

dimana L adalah *loss*, t adalah vektor *one-hot* target yang diharapkan, dan Y adalah nilai matriks output pada tahap *feedforward* dengan jumlah kelas m buah. Nilai matriks Y diambil dari Tabel 3.33, Sedangkan karena kelas huruf citra yang diharapkan adalah NA, nilai vektor t adalah sebagai berikut.

Tabel 3. 33 Vektor t

t_i	Nilai
$t_1 (\text{NA})$	1
$t_2 (\text{PA})$	0
$t_3 (\text{BA})$	0

Vektor t di atas memaksimalkan probabilitas kelas huruf NA (1 atau 100%) dan meminimalkan probabilitas kelas huruf PA dan BA (0 atau 0%).

Berikut perhitungan *cross-entropy loss function*.

$$L = -\sum_i^m t_i \log(Y_i)$$

$$L = -(t_1 \log(Y_1) + t_2 \log(Y_2) + t_3 \log(Y_3))$$

$$L = -(1 * \log(0.347115605) + 1 * \log(0.113824397) + 1 * \log(0.539059999))$$

$$L = 1.058097399$$

2. Penghitungan gradien kesalahan terhadap parameter bobot W_{ji}

Setelah itu hitung gradien kesalahan terhadap parameter bobot W_{ji} , dengan menggunakan rumus *chain rule* [25].

$$\frac{\partial L}{\partial W_{ji}} = \frac{\partial L}{\partial Y_i} \frac{\partial Y_i}{\partial y_{in_i}} \frac{\partial y_{in_i}}{\partial W_{ji}}$$

$$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1 - Y_i)}$$

$$\frac{\partial Y_i}{\partial y_{in_i}} = Y_i(1 - Y_i)$$

$$\frac{\partial y_{in_i}}{\partial W_{ji}} = Z_j$$

$$\frac{\partial L}{\partial W_{ji}} = Y_i(1 - Y_i)Z_j$$

Perhitungan gradien kesalahan terhadap parameter bobot W_{11} adalah sebagai berikut.

$$\frac{\partial L}{\partial W_{1,1}} = \frac{\partial L}{\partial Y_1} \frac{\partial Y_1}{\partial y_{in_1}} \frac{\partial y_{in_1}}{\partial W_{1,1}}$$

$$\frac{\partial L}{\partial W_{1,1}} = Y_1(1 - Y_1)Z_1$$

$$\frac{\partial L}{\partial W_{1,1}} = 0.347115605(1 - 0.347115605) * 0$$

$$\frac{\partial L}{\partial W_{1,1}} = 0$$

Hasil penghitungan Y yang lainnya dapat dilihat pada Tabel 3.34.

Tabel 3. 34 Matriks Gradien $\frac{\partial L}{\partial w_{ji}}$

j/i	1	2	3
1	0	0	0
2	-0.72476697	0.126356463	0.598410505
3	0	0	0
4	0	0	0
5	-1.13314616	0.197553623	0.935592534
6	0	0	0
7	-0.43749783	0.076273728	0.361224105
8	0	0	0
9	-1.55184092	0.270549208	1.281291711
10	0	0	0

3. Penghitungan gradien kesalahan terhadap parameter bobot V_{kj}

Setelah itu hitung gradien kesalahan terhadap parameter bobot V_{kj} , dengan menggunakan rumus *chain rule*.

$$\frac{\partial L}{\partial V_{kj}} = \sum_i^m \frac{\partial L}{\partial Y_i} \frac{\partial Y_i}{\partial y_{in_i}} \frac{\partial y_{in_i}}{\partial Z_j} \frac{\partial Z_j}{\partial z_{in_j}} \frac{\partial z_{in_j}}{\partial V_{kj}}$$

$$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1 - Y_i)}$$

$$\frac{\partial Y_i}{\partial y_{in_i}} = Y_i(1 - Y_i)$$

$$\frac{\partial y_{in_i}}{\partial Z_j} = W_{ji}$$

$$\frac{\partial Z_j}{\partial z_{in_j}} = Z_j(1 - Z_j)$$

$$\frac{\partial z_{in_j}}{\partial V_{kj}} = X_k$$

$$\frac{\partial L}{\partial V_{kj}} = \sum_i^m (Y_i - t_i)(W_{ji})(Z_j(1 - Z_j))(X_k)$$

Dengan m adalah jumlah *node output layer* Y.

Perhitungan gradien kesalahan terhadap parameter bobot V_{11} adalah sebagai berikut.

$$\frac{\partial L}{\partial V_{1,1}} = \sum_{i=1}^3 (Y_i - t_i)(W_{1,i})(Z_1(1 - Z_1))(X_1)$$

$$\frac{\partial L}{\partial V_{1,1}} = \sum_{i=1}^3 (Y_i - t_i)(W_{1,i})(0(1 - 0))(0,92)$$

$$\frac{\partial L}{\partial V_{1,1}} = \sum_{i=1}^3 (Y_i - t_i)(W_{1,i})(0)$$

$$\frac{\partial L}{\partial V_{1,1}} = (Y_1 - t_1)(W_{1,1})(0) + (Y_2 - t_2)(W_{1,2})(0) + (Y_3 - t_3)(W_{1,3})(0)$$

$$\frac{\partial L}{\partial V_{1,1}} = (0.347115605 - 1)(-0,13)(0) + (0.113824397 - 0)(0,35)(0)$$

$$+ (0.539059999 - 0)(0,3)(0)$$

$$\frac{\partial L}{\partial V_{1,1}} = 0$$

Hasil penghitungan Y yang lainnya dapat dilihat pada Tabel 3.35.

Tabel 3. 35 Matriks Gradien $\frac{\partial L}{\partial v_{kj}}$

k/j	1	2	3	4	5	6	7	8	9	10
1	0	0.035129 654	0	0	0.105317 103	0	0.075770 02	0	-0.4600 23029	0
2	0	0.046966 82	0	0	0.140804 388	0	0.101301 223	0	-0.61 5030789	0
3	0	0.053076 325	0	0	0.159120 406	0	0.114478 618	0	-0.69 5034795	0
4	0	0.011837 166	0	0	0.035487 285	0	0.025531 203	0	-0.155 00776	0
5	0	0.037038 874	0	0	0.111040 859	0	0.079887 956	0	-0.48 5024281	0
6	0	0.037038 874	0	0	0.111040 859	0	0.079887 956	0	-0.48 5024281	0
7	0	0.031311 213	0	0	0.093869 592	0	0.067534 149	0	-0.410 020526	0
8	0	0.043148 379	0	0	0.129356 877	0	0.093065 351	0	-0.56 5028286	0
9	0	0.019092 203	0	0	0.057237 556	0	0.041179 359	0	-0.25 0012516	0
10	0	0.011837 166	0	0	0.035487 285	0	0.025531 203	0	-0.155 00776	0
11	0	0.011837 166	0	0	0.035487 285	0	0.025531 203	0	-0.15 500776	0
12	0	0.037802 563	0	0	0.113330 361	0	0.081535 131	0	-0.49 5024782	0

k/j	1	2	3	4	5	6	7	8	9	10
13	0	0	0	0	0	0	0	0	0	0
14	0	0.015655 607	0	0	0.046934 796	0	0.033767 074	0	-0.20 5010263	0
15	0	0.003818 441	0	0	0.011447 511	0	0.008235 872	0	-0.050 002503	0
16	0	0.004582 129	0	0	0.013737 013	0	0.009883 046	0	-0.06 0003004	0
17	0	0.011837 166	0	0	0.035487 285	0	0.025531 203	0	-0.15 500776	0
18	0	0.027874 617	0	0	0.083566 832	0	0.060121 864	0	-0.365 018273	0
19	0	0	0	0	0	0	0	0	0	0
20	0	0.011073 478	0	0	0.033197 783	0	0.023884 028	0	-0.14 5007259	0
21	0	0.007255 037	0	0	0.021750 271	0	0.015648 156	0	-0.09 5004756	0
22	0	0.019092 203	0	0	0.057237 556	0	0.041179 359	0	-0.250 012516	0
23	0	0.016419 295	0	0	0.049224 298	0	0.035414 249	0	-0.21 5010764	0
24	0	0.016419 295	0	0	0.049224 298	0	0.035414 249	0	-0.21501 0764	0
25	0	0.010309 79	0	0	0.030908 28	0	0.022236 854	0	-0.13 5006759	0
26	0	0.024819 864	0	0	0.074408 823	0	0.053533 167	0	-0.32 5016271	0
27	0	0.009164 258	0	0	0.027474 027	0	0.019766 092	0	-0.12 0006008	0

4. Penghitungan gradien kesalahan terhadap parameter filter F

Setelah itu hitung gradien kesalahan terhadap parameter filter F. Sesuai aturan *chain rule*, sebelumnya kita harus mencari gradien untuk *layer-layer* sebelumnya terlebih dahulu. Lalu hitung gradien masing-masing filter dengan gradien yang didapat sebelumnya.

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial X} \frac{\partial X}{\partial P} \frac{\partial P}{\partial F}$$

Pertama hitung nilai $\frac{\partial L}{\partial X}$. Hitung gradien kesalahan terhadap parameter bobot

X_k , dengan menggunakan rumus *chain rule*.

$$\begin{aligned}
\frac{\partial L}{\partial X_k} &= \sum_j^n \sum_i^m \frac{\partial L}{\partial Y_i} \frac{\partial Y_i}{\partial y_{in_i}} \frac{\partial y_{in_i}}{\partial Z_j} \frac{\partial Z_j}{\partial z_{in_j}} \frac{\partial z_{in_j}}{\partial X_k} \\
\frac{\partial L}{\partial Y_i} &= \frac{Y_i - t_i}{Y_i(1 - Y_i)} \\
\frac{\partial Y_i}{\partial y_{in_i}} &= Y_i(1 - Y_i) \\
\frac{\partial y_{in_i}}{\partial Z_j} &= W_{ji} \\
\frac{\partial Z_j}{\partial z_{in_j}} &= Z_j(1 - Z_j) \\
\frac{\partial z_{in_j}}{\partial X_k} &= V_{kj} \\
\frac{\partial L}{\partial X_k} &= \sum_j^n \sum_i^m (Y_i - t_i)(W_{ji})(Z_j(1 - Z_j))(V_{kj})
\end{aligned}$$

Dengan m adalah jumlah *node output layer* Y dan n adalah jumlah *node hidden layer* Z.

Perhitungan gradien kesalahan terhadap parameter bobot X_1 adalah sebagai berikut.

$$\begin{aligned}
\frac{\partial L}{\partial X_1} &= \sum_1^{10} \sum_1^3 (Y_i - t_i)(W_{ji})(Z_j(1 - Z_j))(V_{1,j}) \\
\frac{\partial L}{\partial X_1} &= \sum_1^{10} \sum_1^3 (Y_i - t_i)(W_{j,i})(Z_j(1 - Z_j))(V_{1,j}) \\
\frac{\partial L}{\partial X_1} &= \sum_1^3 (Y_i - t_i)(W_{1,i})(Z_1(1 - Z_1))(V_{1,1}) + \dots \\
&\quad + \sum_1^3 (Y_i - t_i)(W_{10,i})(Z_{10}(1 - Z_{10}))(V_{1,10})
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L}{\partial X_1} &= (Y_1 - t_1)(W_{1,1})(Z_1(1 - Z_1))(V_{1,1}) \\
&\quad + (Y_2 - t_2)(W_{1,2})(Z_1(1 - Z_1))(V_{1,1}) \\
&\quad + (Y_3 - t_3)(W_{1,3})(Z_1(1 - Z_1))(V_{1,1}) + \dots \\
&\quad + (Y_1 - t_1)(W_{10,1})(Z_{10}(1 - Z_{10}))(V_{1,10}) \\
&\quad + (Y_2 - t_2)(W_{10,2})(Z_{10}(1 - Z_{10}))(V_{1,10}) \\
&\quad + (Y_3 - t_3)(W_{10,3})(Z_{10}(1 - Z_{10}))(V_{1,10}) \\
\frac{\partial L}{\partial X_1} &= (0,454639354 - 1)(-0,13)(0(1 - 0))(0,36) \\
&\quad + (0,186776455 - 0)(0,35)(0(1 - 0))(0,36) \\
&\quad + (0,358584191 - 0)(0,3)(0(1 - 0))(0,36) + \dots \\
&\quad + (0,454639354 - 1)(0,24)(0(1 - 0))(-0,19) \\
&\quad + (0,186776455 - 0)(-0,09)(0(1 - 0))(-0,19) \\
&\quad + (0,358584191 - 0)(-0,08)(0(1 - 0))(-0,19) \\
\frac{\partial L}{\partial X_1} &= -0.223755826
\end{aligned}$$

Hasil penghitungan X_k yang lainnya dapat dilihat pada Tabel 3.36.

Tabel 3. 36 Matriks Gradien $\frac{\partial L}{\partial X_k}$

k	Nilai	k	Nilai	k	Nilai
1	-0.223755826	10	0.10353709	19	0.015340099
2	0.153130093	11	-0.264292167	20	0.131142998
3	-0.144711736	12	-0.172877256	21	0.197761474
4	-0.152182712	13	0.037950834	22	-0.15565207
5	-0.066936127	14	-0.143120008	23	-0.039542465
6	0.151477422	15	-0.068938404	24	-0.181560462
7	-0.245515922	16	-0.075475461	25	-0.04740359
8	-0.067146992	17	-0.185268341	26	0.08153408
9	0.151930913	18	-0.197638223	27	0.196784343

Setelah itu kita hitung hitung nilai $\frac{\partial X}{\partial P}$ dengan menggunakan nilai yang didapat dari perhitungan $\frac{\partial L}{\partial X}$. Nilai ini didapat dari mencari gradien pada layer *pooling*

layer. Pertama kita kembalikan lagi bentuk vektor $\frac{\partial L}{\partial x}$ menjadi 3 gradien

feature map $\frac{\partial L}{\partial x[1]}$, $\frac{\partial L}{\partial x[2]}$, dan $\frac{\partial L}{\partial x[3]}$.

Tabel 3. 37 Matriks Gradien Feature Map $\frac{\partial L}{\partial x[1]}$

x/y	1	2	3
1	-0.223755826	0.153130093	-0.144711736
2	-0.152182712	-0.066936127	0.151477422
3	-0.245515922	-0.067146992	0.151930913

Tabel 3. 38 Matriks Gradien Feature Map $\frac{\partial L}{\partial x[2]}$

x/y	1	2	3
1	0.10353709	-0.264292167	-0.172877256
2	0.037950834	-0.143120008	-0.068938404
3	-0.075475461	-0.185268341	-0.197638223

Tabel 3. 39 Matriks Gradien Feature Map $\frac{\partial L}{\partial x[3]}$

x/y	1	2	3
1	0.015340099	0.131142998	0.197761474
2	-0.15565207	-0.039542465	-0.181560462
3	-0.04740359	0.08153408	0.196784343

Untuk operasi *max-pooling feature map* P, gradien P melihat kepada nilai-nilai maksimum *feature map* FM[1] sesuai dengan hasil *max-pooling* pada tahap *feedforward*. Gradien P untuk pixel-pixel dengan nilai maksimum adalah $\frac{\partial L}{\partial x}$, sedangkan gradien untuk pixel-pixel lainnya adalah 0. Berikut posisi nilai-nilai maksimum *feature map* FM untuk FM[1], FM[2], dan FM[3].

Tabel 3. 40 Posisi Pixel-Pixel Nilai Maksimum Pada Feature Map FM[1]

x/y	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	1	0	1	0	1
3	0	0	0	0	1	0
4	0	1	0	1	0	0
5	0	0	1	0	0	1
6	0	1	0	0	0	0

Tabel 3. 41 Posisi Pixel-Pixel Nilai Maksimum Pada Feature Map FM[2]

x/y	1	2	3	4	5	6
1	0	1	1	0	0	1
2	0	0	0	0	0	0
3	0	0	0	1	1	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	1	0	1	0	1

Tabel 3. 42 Posisi Pixel-Pixel Nilai Maksimum Pada Feature Map FM[3]

x/y	1	2	3	4	5	6
1	0	0	0	1	0	0
2	0	0	0	0	0	1
3	0	0	0	0	1	0
4	0	0	0	1	0	0
5	0	0	1	0	0	0
6	0	1	0	0	0	1

Substitusikan nilai $\frac{\partial L}{\partial x}$ untuk setiap nilai 1 pada matriks, sehingga didapat

matriks gradien *feature map* $\frac{\partial L}{\partial p}$ seperti berikut.

Tabel 3. 43 Matriks Gradien Feature Map $\frac{\partial X[1]}{\partial P[1]}$

x/y	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	-0.005243628	0	0.058170787	0	0.012471279
3	0	0	0	0	0.038177646	0
4	0	-0.030587371	0	-0.019275521	0	0
5	0	0	-0.009814486	0	0	0.051048769
6	0	-0.041315732	0	0	0	0

Tabel 3. 44 Matriks Gradien Feature Map $\frac{\partial X[2]}{\partial P[2]}$

x/y	1	2	3	4	5	6
1	0	0.028008 804	-0.041 067025	0	0	-0.010 23867
2	0	0	0	0	0	0
3	-0.024 642229	0	0	0.00365 1795	0.03529 7694	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	-0.030 771478	0	-0.00 425878	0	-0.02 997477

Tabel 3. 45 Matriks Gradien Feature Map $\frac{\partial X[3]}{\partial P[3]}$

x/y	1	2	3	4	5	6
1	0.009678 772	0	0	-0.007 664188	0	0
2	0	0	0	0	0	0.02208 5118
3	-0.035 114983	0	0	0	-0.037 880989	0
4	0	0	0	-0.001 475703	0	0
5	0	0	0.0553 8236	0	0	0
6	0	0.0361 40866	0	0	0	0.01964 4106

Setelah itu kita hitung hitung nilai $\frac{\partial P}{\partial F}$ dengan menggunakan nilai yang didapat dari perhitungan $\frac{\partial X}{\partial P}$. Nilai ini didapat dari mencari gradien pada layer *convolution layer*. Untuk menghitungnya, lakukan operasi konvolusi seperti pada tahap *feedforward*, dengan masukannya adalah matriks citra masukan (yang sudah diberi *zero-padding*) A, dan filternya adalah gradien *feature map* $\frac{\partial X[1]}{\partial P[1]}$, $\frac{\partial X[2]}{\partial P[2]}$, dan $\frac{\partial X[3]}{\partial P[3]}$. Hasilnya adalah gradien filter $\frac{\partial P[1]}{\partial F[1]}$, $\frac{\partial P[2]}{\partial F[2]}$ dan $\frac{\partial P[3]}{\partial F[3]}$. Berikut adalah hasil konvolusinya.

Tabel 3. 46 Matriks Gradien Filter $\frac{\partial P[1]}{\partial F[1]}$

x/y	1	2	3
1	0.065398437	0.033170344	0.052927158
2	-0.01975681	-0.041315732	-0.013732316
3	0.041234284	0.0688177	-0.009814486

Tabel 3. 47 Matriks Gradien Filter $\frac{\partial P[2]}{\partial F[2]}$

x/y	1	2	3
1	0	0,004526216	0,003651795
2	-0,05300423	-0,084650132	-0,048088479
3	-0,00658688	0	0

Tabel 3. 48 Matriks Gradien Filter $\frac{\partial P[3]}{\partial F[3]}$

x/y	1	2	3
1	0.022085118	0.074251652	0
2	0.08623157	0.048120784	0.028476678
3	0.05538236	0.05538236	0.047718172

5. *Update* nilai parameter

Setelah semua gradien parameter dihitung, *update* parameter-parameter dengan menggunakan *Stochastic Gradient Descent*. Pada tahap ini akan digunakan *learning rate* $\alpha=0,1$. Berikut adalah perhitungan untuk *update* nilai parameter bobot W_{11} .

$$W_{1,1}' = W_{1,1} - \alpha \left(\frac{\partial L}{\partial W_{1,1}} \right)$$

$$W_{1,1}' = -0,13 - 0.1(0)$$

$$W_{1,1}' = -0,13$$

Hasil penghitungan W_{ji}' yang lainnya dapat dilihat pada Tabel 3.49.

Tabel 3. 49 Matriks W_{ji}'

i/j	1	2	3
1	-0.13	0.35	0.3
2	0.492476697	0.267364354	-0.18984105
3	0.31	0.24	0.33
4	0.39	-0.06	-0.36
5	0.023314616	-0.139755362	-0.34355925
6	0.34	-0.35	0.47
7	-0.26625022	-0.217627373	0.323877589
8	-0.43	-0.26	-0.49
9	0.285184092	-0.307054921	0.371870829
10	0.24	-0.09	-0.08

Selanjutnya berikut adalah perhitungan untuk *update* nilai parameter bobot V_{11} .

$$V_{1,1}' = V_{1,1} - \alpha \left(\frac{\partial L}{\partial V_{1,1}} \right)$$

$$V_{1,1}' = 0,36 - 0.1(0)$$

$$V_{1,1}' = 0.36$$

Hasil penghitungan V_{kj}' yang lainnya dapat dilihat pada Tabel 3.50.

Tabel 3. 50 Matriks V_{kj}'

k/j	1	2	3	4	5	6	7	8	9	10
1	0.36	0.456487 035	-0.43	-0.35	0.049468 29	0.25	-0.1675 77	-0.41	0.516002 303	-0.19
2	-0.17	0.465303 318	-0.48	0.31	0.355919 561	-0.19	-0.3401 3012	0.24	-0.1784 96921	-0.36
3	-0.14	0.124692 367	-0.49	-0.1	0.324087 959	-0.46	0.248552 138	-0.11	0.489503 479	-0.46
4	0.25	-0.051 18372	-0.22	-0.43	-0.1235 4873	0	-0.1425 5312	0.23	0.265500 776	0.41
5	-0.21	0.036296 113	0.01	-0.43	-0.1811 0409	-0.34	-0.1779 888	-0.19	0.118502 428	0.35
6	0.32	-0.1937 0389	-0.47	-0.36	0.408895 914	-0.38	0.182011 204	-0.24	-0.1414 97572	0.28
7	0.04	0.156868 879	0.23	-0.21	-0.2793 8696	-0.32	-0.136 75341	0.3	0.461002 053	0.05
8	-0.44	-0.234 31484	0.02	0.48	0.027064 312	0.22	0.440693 465	-0.38	0.256502 829	-0.05
9	0.49	-0.081 90922	-0.16	-0.14	0.464276 244	0.19	0.495882 064	0.29	-0.094 998748	-0.34
10	0.22	-0.1711 8372	-0.14	0.46	0.396451 272	-0.44	-0.072 55312	-0.25	-0.124 499224	0.05
11	-0.33	0.178816 283	0.41	0.13	-0.233 54873	-0.13	-0.182 55312	0.16	0.475500 776	0.15

k/j	1	2	3	4	5	6	7	8	9	10
12	-0.1	-0.043 78026	-0.41	-0.06	0.188666 964	-0.48	0.061846 487	-0.13	0.449502 478	-0.48
13	0.31	-0.37	-0.23	-0.48	-0.32	0.09	0.47	-0.02	-0.1	-0.05
14	-0.09	0.178434 439	0.45	0.22	0.305306 52	0.07	-0.3733 7671	-0.36	0.330501 026	-0.13
15	0.09	0.369618 156	0.16	-0.01	0.388855 249	-0.28	0.209176 413	-0.35	0.295000 25	-0.31
16	-0.44	0.259541 787	-0.19	-0.35	-0.431 3737	0.22	-0.500 9883	-0.22	-0.003 9997	0.3
17	0.11	0.398816 283	-0.04	-0.33	-0.113 54873	0.19	0.447446 88	-0.5	0.465500 776	-0.33
18	-0.14	0.247212 538	-0.42	-0.11	-0.268 35668	-0.4	-0.096 01219	0.44	0.376501 827	-0.16
19	-0.13	-0.04	0	-0.13	0.19	0.39	-0.12	0.21	-0.01	-0.47
20	-0.15	-0.501 10735	0.27	-0.43	-0.093 31978	-0.09	0.307611 597	-0.01	-0.255 499274	-0.2
21	-0.24	0.049274 496	0.01	0.14	-0.082 17503	-0.45	-0.001 56482	-0.38	-0.400 499524	0.41
22	0.06	-0.461 90922	-0.19	0.42	-0.0357 2376	0.16	0.485882 064	0.24	0.375001 252	-0.06
23	-0.23	-0.261 64193	-0.17	-0.12	0.225077 57	0.33	0.106458 575	-0.03	0.151501 076	-0.07
24	-0.19	0.168358 071	-0.46	0.43	-0.324 92243	0.16	-0.26 354142	0.5	0.281501 076	-0.27
k/j	1	2	3	4	5	6	7	8	9	10
25	-0.3	0.378969 021	-0.18	-0.29	0.486909 172	0.23	-0.342 22369	0.42	0.193500 676	-0.19
26	0.08	0.257518 014	0.1	-0.44	0.432559 118	0.13	0.434646 683	-0.47	0.062501 627	-0.33
27	0.27	-0.090 91643	-0.21	-0.23	0.097252 597	-0.15	-0.441 97661	0.36	-0.437 999399	0.15

Selanjutnya hitung *update* nilai parameter filter F[1], F[2], dan F[3]. Berikut adalah perhitungan untuk *update* nilai parameter bobot F[1]_{1,1}.

$$F[1]_{1,1}' = F[1]_{1,1} - \alpha \left(\frac{\partial P[1]}{\partial F[1]_{1,1}} \right)$$

$$F[1]_{1,1}' = 0,42 - 0.1(0,065398437)$$

$$F[1]_{1,1}' = 0,413460156$$

Hasil penghitungan F' yang lainnya dapat dilihat pada tabel-tabel berikut.

Tabel 3. 51 Matriks Filter F[1]'

x/y	1	2	3
1	0.413460156	0.486682966	0.004707284
2	0.481975681	-0.325868427	0.181373232
3	0.365876572	0.12311823	-0.339018551

Tabel 3. 52 Matriks Filter F[2]'

x/y	1	2	3
1	-0.36	-0.190452622	-0.290365179
2	0.295300423	0.448465013	-0.415191152
3	0.260658688	-0.23	-0.44

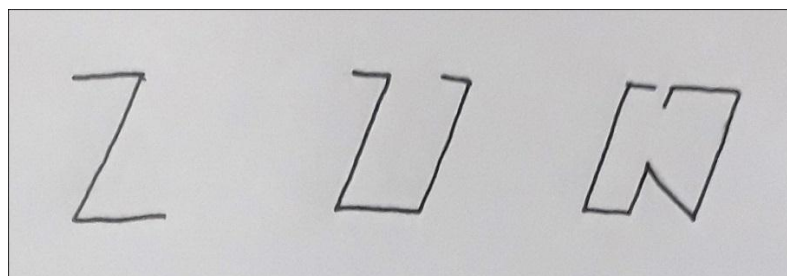
Tabel 3. 53 Matriks Filter F[3]'

x/y	1	2	3
1	-0.24220851	0.272574835	-0.46
2	0.141376843	0.085187922	-0.452847668
3	-0.50553824	0.214461764	0.495228183

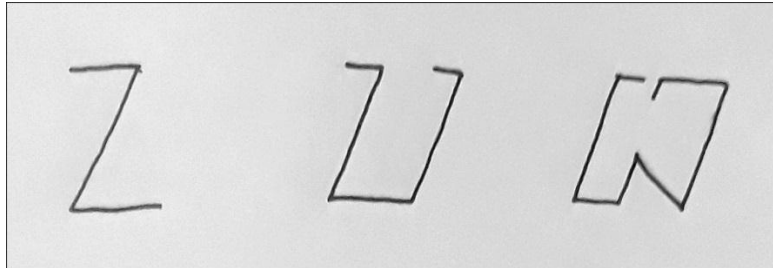
Hasil *update* parameter-parameter bobot V' , bobot W' dan filter F' akan digunakan pada iterasi *feedforward* selanjutnya.

3.2.4 Analisis Rencana Pengujian

Pengujian CNN yang dilakukan sama seperti pelatihan pada CNN. Perbedaannya adalah citra uji tidak disimpan dan hanya untuk mengetahui karakter tulisan mana saja yang dapat dikenali sesuai dengan sudah dilatih. Berikut adalah citra masukan untuk pengujian.

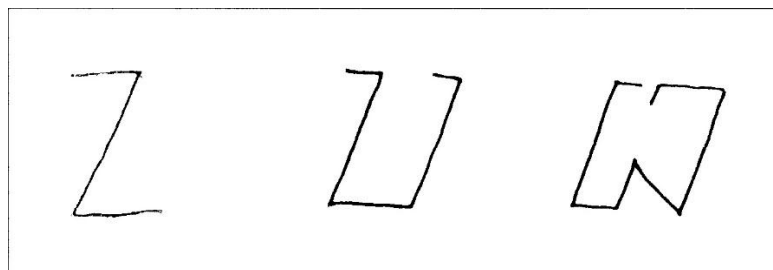
**Gambar 3. 19 Citra Uji Masukan**

Kemudian citra masukan akan masuk ke tahap *preprocessing* berikut adalah hasil dari *grayscale*.



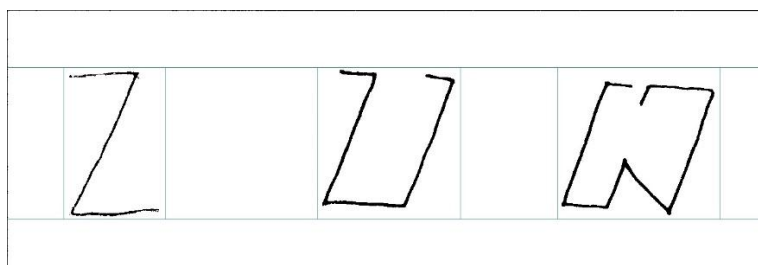
Gambar 3. 20 Citra Uji Hasil Grayscale

Setelah citra *grayscale* didapatkan proses selanjutnya adalah *thresholding*. Berikut adalah hasil dari proses *thresholding*.



Gambar 3. 21 Citra Uji Hasil Thresholding

Setelah dilakukan proses *thresholding*, tahap *preprocessing* selanjutnya adalah segmentasi. Segmentasi yang dilakukan segmentasi baris dan segmentasi kolom. Berikut gambar hasil segmentasi.



Gambar 3. 22 Citra Uji Hasil Segmentasi

Berdasarkan hasil segmentasi citra pada tahap *preprocessing* terdapat 3 karakter tulisan tangan segmentasi. Setelah itu masuk ke tahap pengujian dengan metode

CNN, dimana pada proses ini hanya melakukan perhitungan matriks sampai pada tahap *feedforward* saja untuk menampilkan hasil akhir tidak menggunakan *backpropagation* yang berfungsi untuk merubah bobot matriks.

Data masukan yang digunakan dalam analisis ini adalah data citra hitam putih huruf NA, PA, dan BA pada tabel Gambar 3.23, Gambar 3.24, dan Gambar 3.25 yang telah di-*resize* sehingga menjadi berukuran 6*6 pixel. Berikut adalah matriks citranya.

Tabel 3. 54 Matriks Citra Huruf Masukan NA

x/y	1	2	3	4	5	6
1	1	1	1	1	1	0
2	0	0	0	1	0	0
3	0	0	0	1	0	0
4	0	0	1	0	0	0
5	0	1	0	0	0	0
6	1	1	1	1	1	1

Tabel 3. 55 Matriks Citra Huruf Masukan PA

x/y	1	2	3	4	5	6
1	0	1	1	0	1	1
2	0	0	1	0	0	1
3	0	0	1	0	0	1
4	0	1	0	0	1	0
5	0	1	0	0	1	0
6	1	1	1	1	0	0

Tabel 3. 56 Matriks Citra Huruf Masukan BA

x/y	1	2	3	4	5	6
1	0	1	1	1	1	1
2	0	1	0	0	0	1
3	0	1	0	0	0	1
4	1	0	0	0	1	0
5	1	0	1	0	1	0
6	1	1	0	1	1	0

Inisialisasi Bobot

Dalam analisis ini kelas bobot-bobot yang akan diinisialisasi adalah nilai-nilai bobot dan bias filter *convolutional layer* F' , bobot dan bias *hidden layer* V' , dan bobot dan bias *output layer fully-connected layer* W' , yang diambil dari nilai-nilai bobot dan bias yang telah diperbarui pada tahap pelatihan *backpropagation*.

Filter *convolutional layer* F' berjumlah 3 buah dengan masing-masing berukuran 3×3 pixel, maka jumlah bobot yang harus diisi adalah 27 buah. Berikut adalah matriks-matriks inisialisasi filternya.

Filter F[1]			
x/y	1	2	3
1	0,4135	0,4867	0,0047
2	0,4820	-0,3259	0,1814
3	0,3659	0,1231	-0,3390

Filter F[2]			
x/y	1	2	3
1	-0,3600	-0,1905	-0,2904
2	0,2953	0,4485	-0,4152
3	0,2607	-0,2300	-0,4400

Filter F[3]			
x/y	1	2	3
1	-0,2422	0,2726	-0,4600
2	0,1414	0,0852	-0,4528
3	-0,5055	0,2145	0,4952

Gambar 3. 23 Inisialisasi Matriks-Matriks Filter F'

Sedangkan berikut adalah bias-bias filter *convolutional layer*.

4. Bias $F'[1]$ ($bF'[1]$) = 0,1769
5. Bias $F'[2]$ ($bF'[2]$) = -0,1842
6. Bias $F'[3]$ ($bF'[3]$) = 0,4176

Matriks-matriks filter F' akan digunakan pada proses konvolusi citra masukan pada *convolutional layer*.

Berikut adalah matriks V' . $V'_{0,y}$ menandakan nilai bias (semua diberi nilai 0) dan $V'_{x,y}$ menandakan nilai bobot.

Tabel 3. 57 Inisialisasi Matriks Bobot Hidden Layer V'

x/y	1	2	3	4	5	6	7	8	9	10
0	0	-0,0038	0	0	-0,0114	0	-0,0082	0	0,0500	0
1	0,36	0,4565	-0,43	-0,35	0,0495	0,25	-0,1676	-0,41	0,5160	-0,19
2	-0,17	0,4653	-0,48	0,31	0,3559	-0,19	-0,3401	0,24	-0,1785	-0,36
3	-0,14	0,1247	-0,49	-0,1	0,3241	-0,46	0,2486	-0,11	0,4895	-0,46
4	0,25	-0,0512	-0,22	-0,43	-0,1235	0	-0,1426	0,23	0,2655	0,41
5	-0,21	0,0363	0,01	-0,43	-0,1811	-0,34	-0,1780	-0,19	0,1185	0,35
6	0,32	-0,1937	-0,47	-0,36	0,4089	-0,38	0,1820	-0,24	-0,1415	0,28

x/y	1	2	3	4	5	6	7	8	9	10
7	0,04	0,1569	0,23	-0,21	-0,2794	-0,32	-0,1368	0,3	0,4610	0,05
8	-0,44	-0,2343	0,02	0,48	0,0271	0,22	0,4407	-0,38	0,2565	-0,05
9	0,49	-0,0819	-0,16	-0,14	0,4643	0,19	0,4959	0,29	-0,0950	-0,34
10	0,22	-0,1712	-0,14	0,46	0,3965	-0,44	-0,0726	-0,25	-0,1245	0,05
11	-0,33	0,1788	0,41	0,13	-0,2335	-0,13	-0,1826	0,16	0,4755	0,15
12	-0,1	-0,0438	-0,41	-0,06	0,1887	-0,48	0,0618	-0,13	0,4495	-0,48
13	0,31	-0,3700	-0,23	-0,48	-0,3200	0,09	0,4700	-0,02	-0,1000	-0,05
14	-0,09	0,1784	0,45	0,22	0,3053	0,07	-0,3734	-0,36	0,3305	-0,13
15	0,09	0,3696	0,16	-0,01	0,3889	-0,28	0,2092	-0,35	0,2950	-0,31
16	-0,44	0,2595	-0,19	-0,35	-0,4314	0,22	-0,5010	-0,22	-0,0040	0,3
17	0,11	0,3988	-0,04	-0,33	-0,1135	0,19	0,4474	-0,5	0,4655	-0,33
18	-0,14	0,2472	-0,42	-0,11	-0,2684	-0,4	-0,0960	0,44	0,3765	-0,16
19	-0,13	-0,0400	0	-0,13	0,1900	0,39	-0,1200	0,21	-0,0100	-0,47
20	-0,15	-0,5011	0,27	-0,43	-0,0933	-0,09	0,3076	-0,01	-0,2555	-0,2
21	-0,24	0,0493	0,01	0,14	-0,0822	-0,45	-0,0016	-0,38	-0,4005	0,41
22	0,06	-0,4619	-0,19	0,42	-0,0357	0,16	0,4859	0,24	0,3750	-0,06
23	-0,23	-0,2616	-0,17	-0,12	0,2251	0,33	0,1065	-0,03	0,1515	-0,07
24	-0,19	0,1684	-0,46	0,43	-0,3249	0,16	-0,2635	0,5	0,2815	-0,27
25	-0,3	0,3790	-0,18	-0,29	0,4869	0,23	-0,3422	0,42	0,1935	-0,19
26	0,08	0,2575	0,1	-0,44	0,4326	0,13	0,4346	-0,47	0,0625	-0,33
27	0,27	-0,0909	-0,21	-0,23	0,0973	-0,15	-0,4420	0,36	-0,4380	0,15

Matriks V' akan digunakan sebagai bobot dari *input layer* ke *hidden layer* pada *fully-connected layer*.

Berikut adalah matriks W' . $W'_{0,y}$ menandakan nilai bias (semua diberi nilai 0) dan $W'_{x,y}$ menandakan nilai bobot.

Tabel 3. 58 Matriks Inisialisasi Hidden Layer W'

x/y	1	2	3
0	0,06528844	-0,01138244	-0,053906
1	-0,13	0,35	0,3
2	0,492476697	0,267364354	-0,18984105
3	0,31	0,24	0,33
4	0,39	-0,06	-0,36

x/y	1	2	3
5	0,023314616	-0,139755362	-0,34355925
6	0,34	-0,35	0,47
7	-0,26625022	-0,217627373	0,323877589
8	-0,43	-0,26	-0,49
9	0,285184092	-0,307054921	0,371870829
10	0,24	-0,09	-0,08

Matriks W' akan digunakan sebagai bobot dari *hidden layer* ke *output layer* pada *fully-connected layer*.

Feedforward

Pada tahap *feedforward* akan dilakukan proses CNN dari awal masukan melewati *convolutional*, *pooling*, dan *fully-connected layer* hingga dihasilkan sebuah vektor *one-hot* klasifikasi kelas huruf, seperti pada tahap pelatihan. Lakukan untuk ketiga gambar masukan. Berikut adalah hasil penghitungan vektor *one-hot* matriks Y' untuk ketiga gambar masukan.

Tabel 3. 59 Matriks Y' Gambar NA

Y_i	Nilai
Y_1	0.347115605
Y_2	0.113824397
Y_3	0.539059999

Tabel 3. 60 Matriks Y' Gambar PA




Y_i	Nilai
Y_1	0,590299948
Y_2	0,122792889
Y_3	0,286907163

Tabel 3. 61 Matriks Y' Gambar BA

Y_i	Nilai
Y_1	0,724978807
Y_2	0,10091383
Y_3	0,174107362

Berikut adalah tabel hasil pengujian yang dilakukan.

Tabel 3. 62 Rencana Pengujian

Citra uji	Kelas	Hasil Pengenalan	Keterangan
	NA	55% NA 6% PA 39% BA	Hasil Pengenalan NA
	PA	59% NA 12% PA 29% BA	Hasil Pengenalan NA
	BA	72% NA 10% PA 17% BA	Hasil Pengenalan NA

Dari tabel analisis pengujian di atas, didapatkan hasil sebagai berikut.

- a) Jumlah karakter sama = 1
- b) Jumlah karakter Tidak sama = 2
- c) Jumlah keseluruhan karakter = 3

Kemudian untuk menghitung nilai akurasi pengenalannya digunakan persamaan 2.45, sehingga hasilnya sebagai berikut.

$$\text{Akurasi} = \frac{\text{Jumlah karakter sama}}{\text{Jumlah keseluruhan karakter}} \times 100\%$$

$$\text{Akurasi} = \frac{1}{3} \times 100\%$$

$$\text{Akurasi} = 33,33\%$$

3.2.5 Analisis Kebutuhan Non Fungsional

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

3.2.5.1 UML (*Unified Modelling Language*)

UML merupakan bahasa pemodelan untuk sistem atau perangkat lunak yang berorientasi objek[21]. Versi UML (*Unified Modelling Language*) yang digunakan

pada versi ini adalah versi 1.4, digunakan karena dengan aplikasi *generate code* pada bahasa java, dan referensi yang diambil.

Dalam UML ada 13 diagram umum diantaranya, *use case diagram*, *activity diagram*, *sequence diagram*, *statemachine diagram*, *class diagram*, *communication diagram*, *deployment diagram*, *component diagram*, *object diagram*, *composite structure diagram*, *iinteraction overview diagram*, *package diagram*, *diagram timing*. Akan tetapi yang digunakan dalam penelitian ini hanya 4 *diagram* yaitu *use case*, *activity*, *sequence* dan *class diagram* karena sudah dapat mewakili pemodelan secara keseluruhan dalam penelitian ini.

3.2.5.2 Analisis Minimum 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. 63 Analisis Kebutuhan Perangkat Keras

No	Kebutuhan Perangkat Keras	Spesifikasi Minimum
1.	Processor	Dual Core2.1 Ghz
2.	Memory	RAM 512 Mb

3.2.5.3 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. 64 Analisis Kebutuhan Perangkat Lunak

No.	Kebutuhan Perangkat Lunak
1.	Sistem Operasi Windows XP
2.	JDK 7.10

3.2.5.4 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.

Tabel 3. 65 Analisis Kebutuhan Pengguna

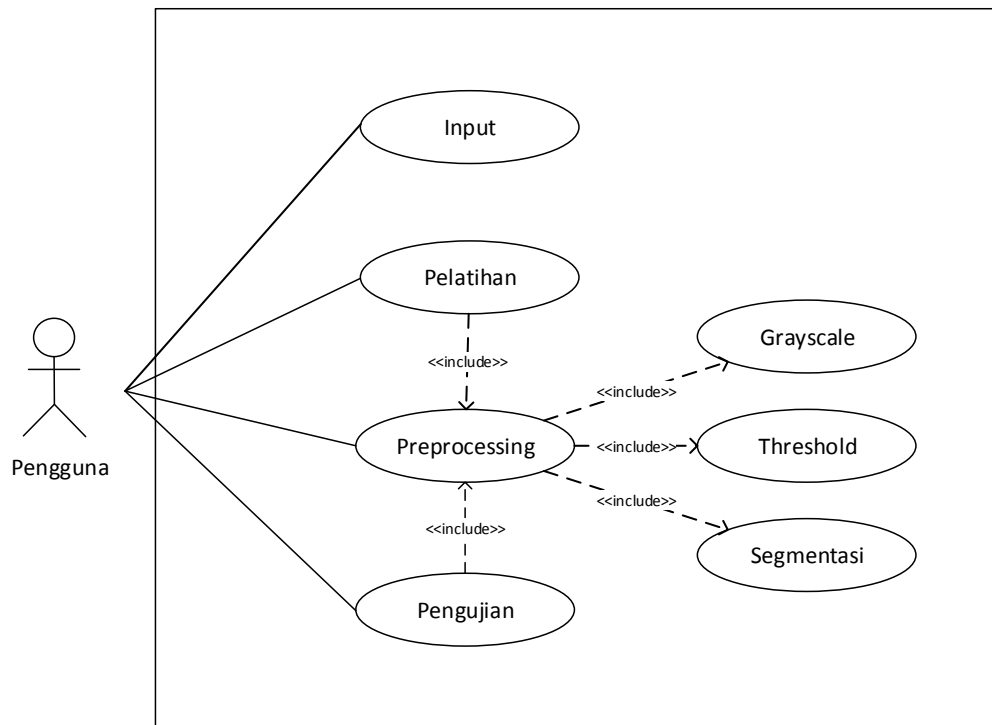
No.	Kebutuhan Pengguna
1.	Dapat mengoperasikan komputer dengan lancar
2.	Mengerti tentang penggunaan aplikasi

3.2.6 Analisis Kebutuhan Fungsional

Selain melakukan analisis terhadap kebutuhan non fungsional, juga dilakukan analisis terhadap kebutuhan fungsional yang berupa pemodelan lingkup *Object Oriented Programming*, meliputi *use case diagram*, *use case scenario*, *activity diagram*, *sequence diagram*, dan *class diagram*.

3.2.6.1 Use Case Diagram

Use Case Diagram merupakan pemodelan yang menunjukkan interaksi antara sistem dengan pengguna (aktor). Dalam *use case* yang dibuat menggunakan lima *use case*, dimana ada *use case* yang berelasi dengan *use case* lainnya. Relasi yang digunakan yaitu *include*, berikut adalah *use case diagram* untuk aplikasi yang dibangun.



Gambar 3. 24 Use Case Diagram

1. Definisi aktor

Aktor merupakan pihak yang mengakses *use case*. Tabel 3.66 merupakan deskripsi dari aktor

Tabel 3. 66 Deskripsi Aktor

No.	Aktor	Deskripsi
1	Pengguna	Pengguna aplikasi

2. Definisi *use case*

Use case menggambarkan apa saja yang dapat dilakukan oleh sistem tabel 3.67 merupakan deskripsi dari setiap *use case*.

Tabel 3. 67 Deskripsi Use Case

No.	Aktor	Deskripsi
1.	Input	Proses Input Citra
2.	Preprocessing	Proses Pengolahan Citra

No.	Aktor	Deskripsi
3.	Pelatihan	Proses Melatih data Citra Latih
4.	Pengujian	Proses Pengujian Citra Uji

3. Skenario *Use Case*

Skenario *Use case* menjelaskan bagaimana setiap *use case* bekerja. Berikut adalah skenario dari setiap *use case*.

Tabel 3. 68 Skenario *Use Case Input*

Nama <i>Use Case</i>	<i>Input</i>	
<i>Use Case Terkait</i>	Pelatihan, Pengujian, Pengolahan data <i>training</i>	
Tujuan	Memasukan Citra	
Kondisi Awal	Citra belum dimasukkan ke dalam aplikasi	
Kondisi Akhir	Citra sudah dimasukkan ke dalam aplikasi dan sudah siap	
Aktor	Pengguna	
Skenario Utama	Langkah	Aksi
	1	Pengguna memilih input citra
	2	Sistem menampilkan pilihan citra
	3	Pengguna memilih citra tulisan tangan
	4	Sistem menampilkan citra yang dipilih pengguna

Tabel 3. 69 Skenario Use Case Preprocessing

Nama Use Case	<i>Preprocessing</i>	
Use Case Terkait	Pengolahan data <i>training</i> , pengujian	
Tujuan	Mengolah Citra	
Kondisi Awal	Citra Belum dimasukan kedalam aplikasi	
Kondisi Akhir	Citra sudah siap dimasukan kedalam aplikasi dan sudah siap	
Aktor	Pengguna	
Skenario Aktor	Langkah	Aksi
	1	Pengguna Memilih Input Citra
	2	Sistem menampilkan citra
	3	Pengguna memilih <i>preprocessing</i>
	4	Sistem menampilkan hasil <i>preprocessing</i>

Tabel 3. 70 Skenario Use Case Pelatihan

Nama Use Case	Pelatihan	
Use Case Terkait	Pengolahan data <i>training</i> , pengujian	
Tujuan	Menambah data kelas hingga sesuai target	
Kondisi Awal	Data latih dan data kelas sudah tersedia	
Kondisi Akhir	Data bobot kelas berhasil dihitung hingga keluar nilai vector	
Skenario Aktor	Langkah	Aksi
	1	Pengguna memasukkan citra hasil <i>preprocessing</i>

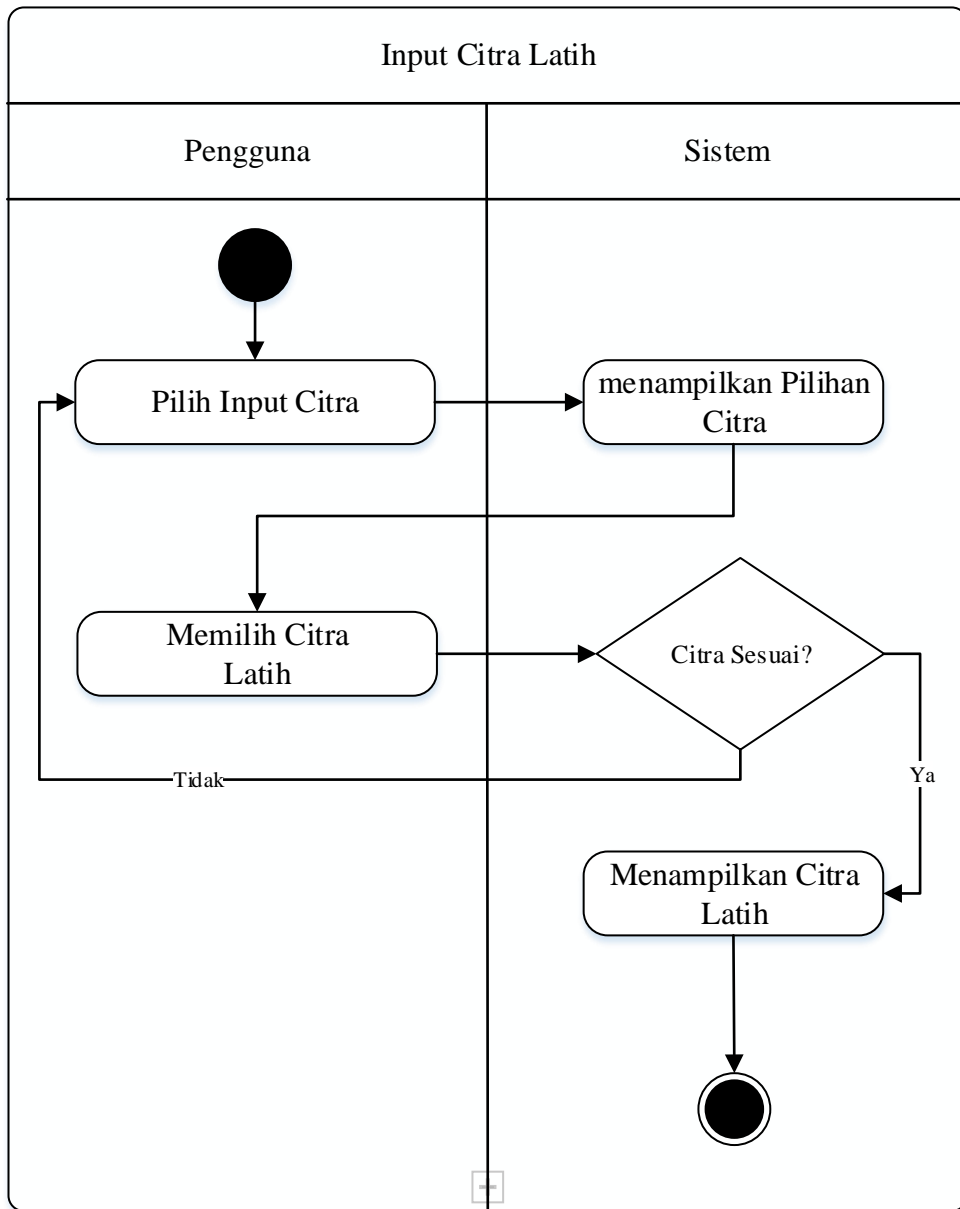
	Langkah	Aksi
	2	Pengguna memilih pelatihan
	3	Sistem melakukan perubahan bobot

Tabel 3. 71 Skenario *Use Case* Pengujian

Nama <i>Use Case</i>	Penilaian Aksara Sunda	
<i>Use Case</i> Terkait	Tidak ada	
Tujuan	Melakukan evaluasi tulisan aksara	
Kondisi Awal	Tulisan tangan sudah ada dan belum ditampilkan	
Kondisi Akhir	Tulisan tangan ditampilkan	
Aktor Skenario Utama	Langkah	Aksi
	1	Pengguna memilih tombol cek tulisan
	2	Sistem melakukan proses perhitungan dan menampilkan hasil tulisan tangan

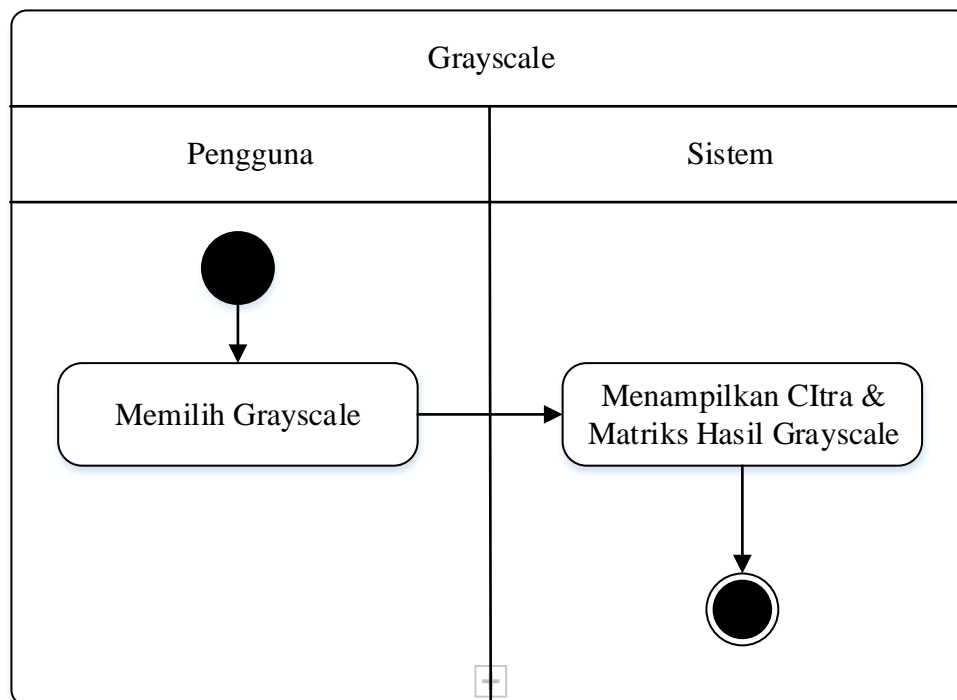
3.2.6.2 Activity Diagram

Activity diagram berfungsi memodelkan bagaimana aliran *use case* bekerja. Berikut adalah *activity* diagram dari *use case* sebelumnya yang berjumlah 10 *diagram*.



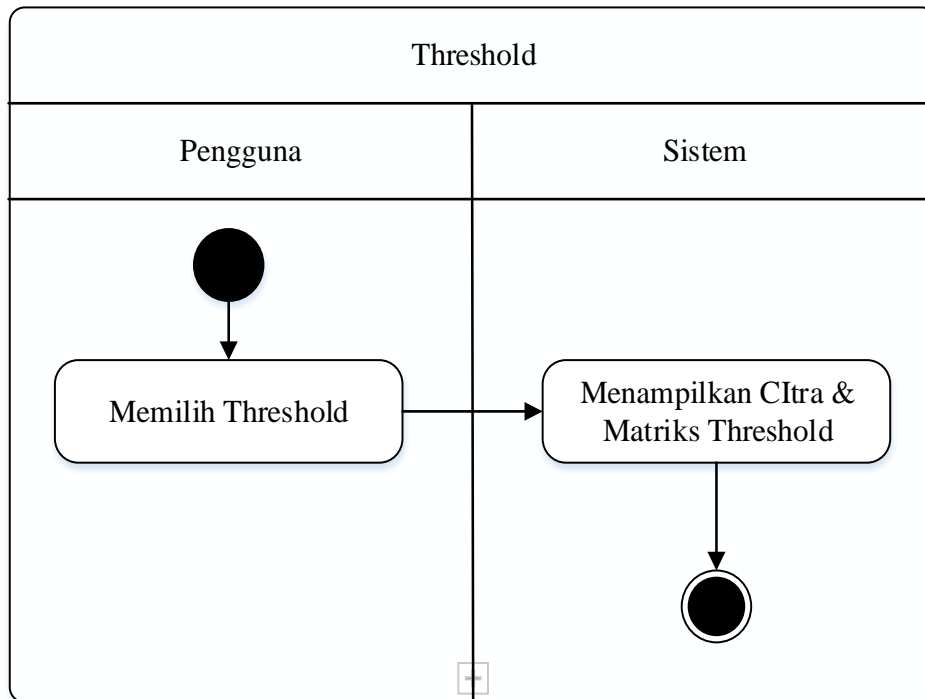
Gambar 3. 25 Activity Diagram *Input Citra*

Gambar 3.25 merupakan Activity diagram input citra merupakan aktifitas memilih citra dan menampilkan citra.



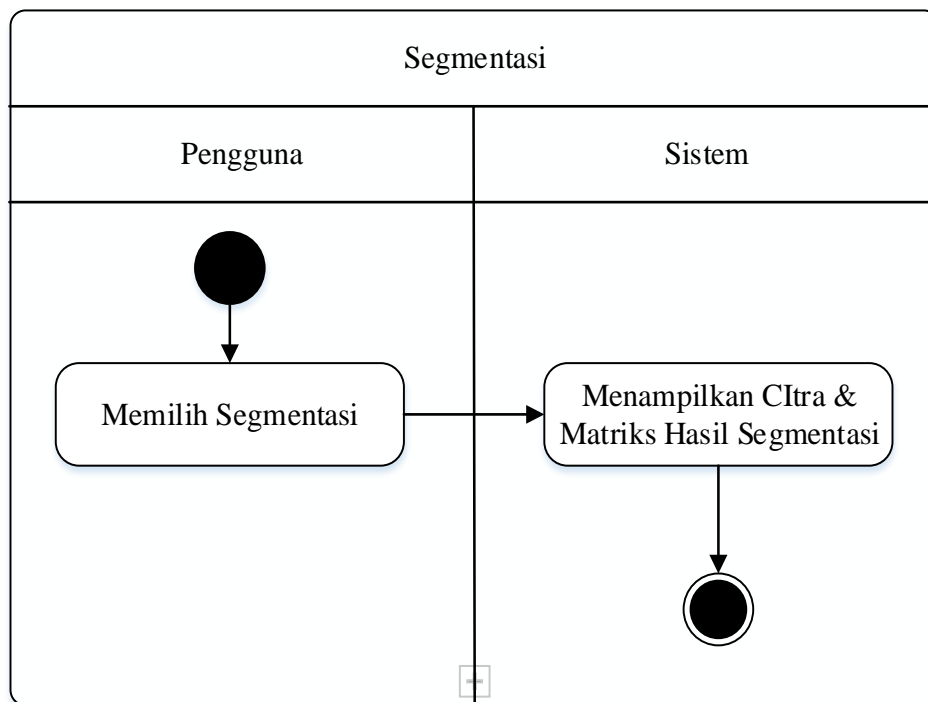
Gambar 3. 26 Activity Diagram *Grayscale*

Gambar 3.26 Merupakan Activity diagram proses grayscale dimana aktifitas ini termasuk dalam proses *preprocessing*.



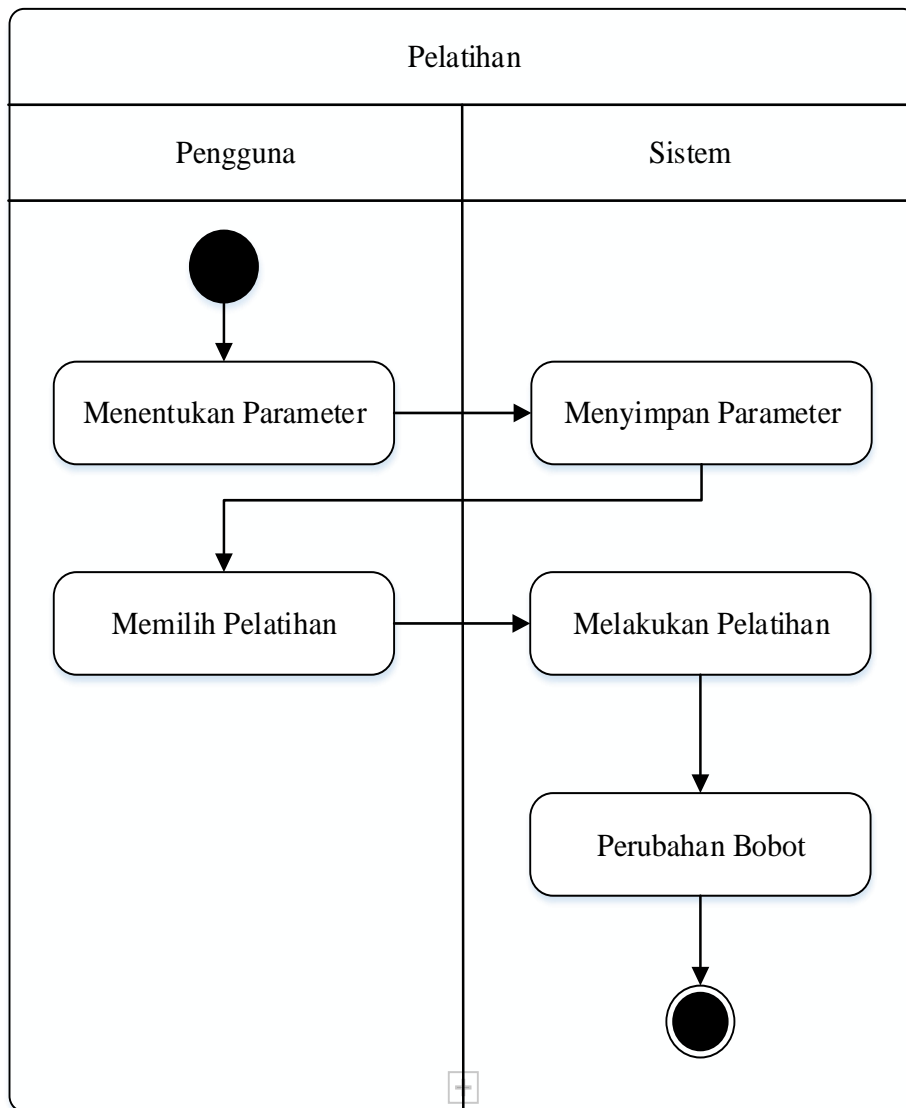
Gambar 3. 27 Activity Diagram Threshold

Gambar 3.27 merupakan *activity* proses *sauvola thresholding*, aktifitas ini merupakan bagian dari proses *preprocessing* dan dilakukan setelah proses *grayscale* selesai.



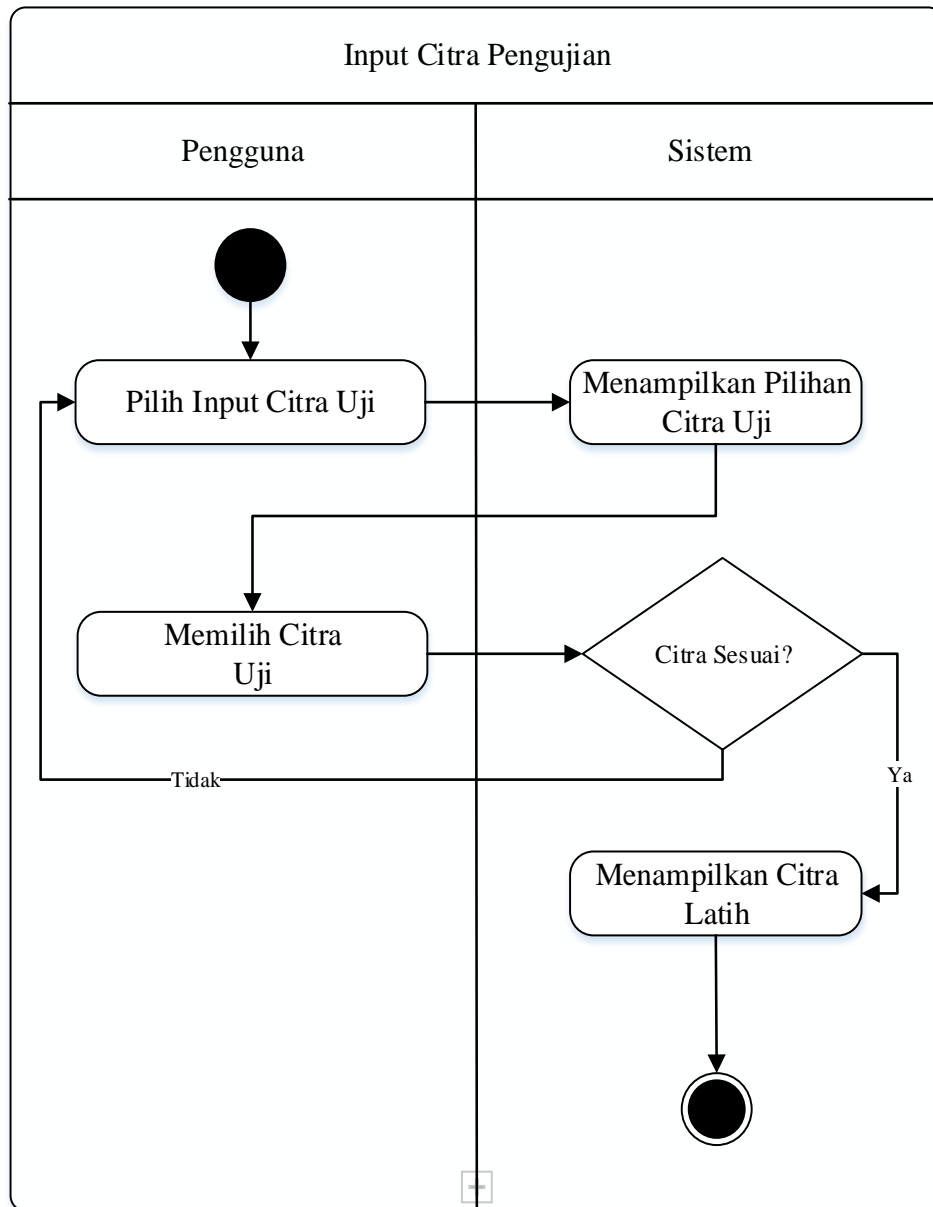
Gambar 3. 28 Activity Diagram Segmentasi

Gambar 3.28 merupakan *activity* proses Segmentasi, aktifitas ini merupakan bagian dari proses *preprocessing* dan dilakukan setelah proses *grayscale* selesai.



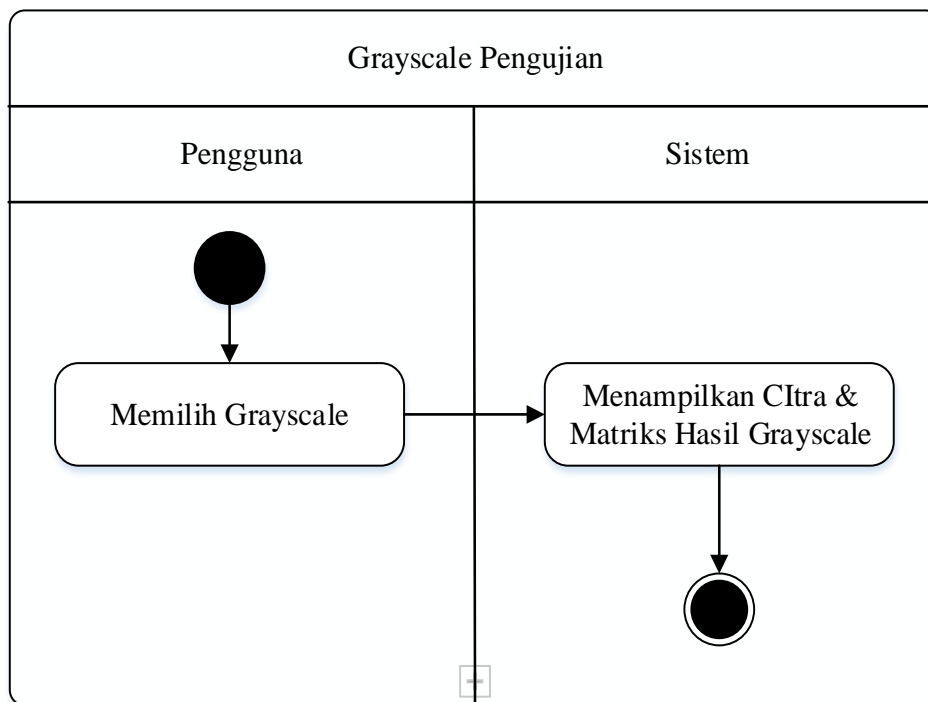
Gambar 3. 29 Activity Diagram Pelatihan

Gambar 3.29 merupakan *activity* proses pelatihan, aktifitas ini dilakukan setelah proses *preprocessing* selesai.



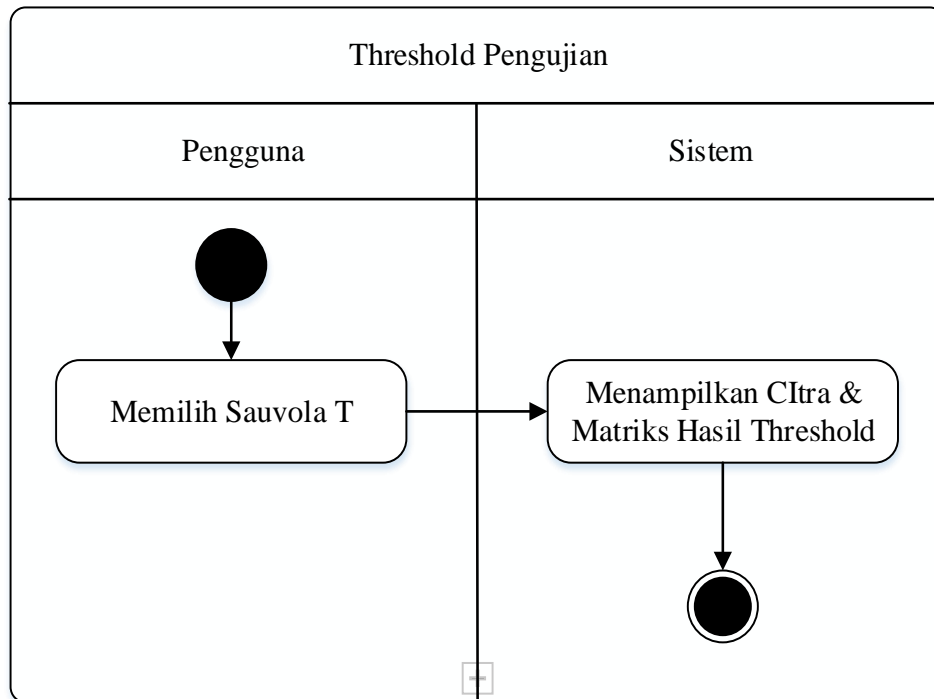
Gambar 3. 30 Activity Diagram *Input Citra Pengujian*

Gambar 3. 30 merupakan *Activity* diagram input citra pengujian merupakan aktifitas memilih citra dan menampilkan citra sebelum dilakukan pengujian.



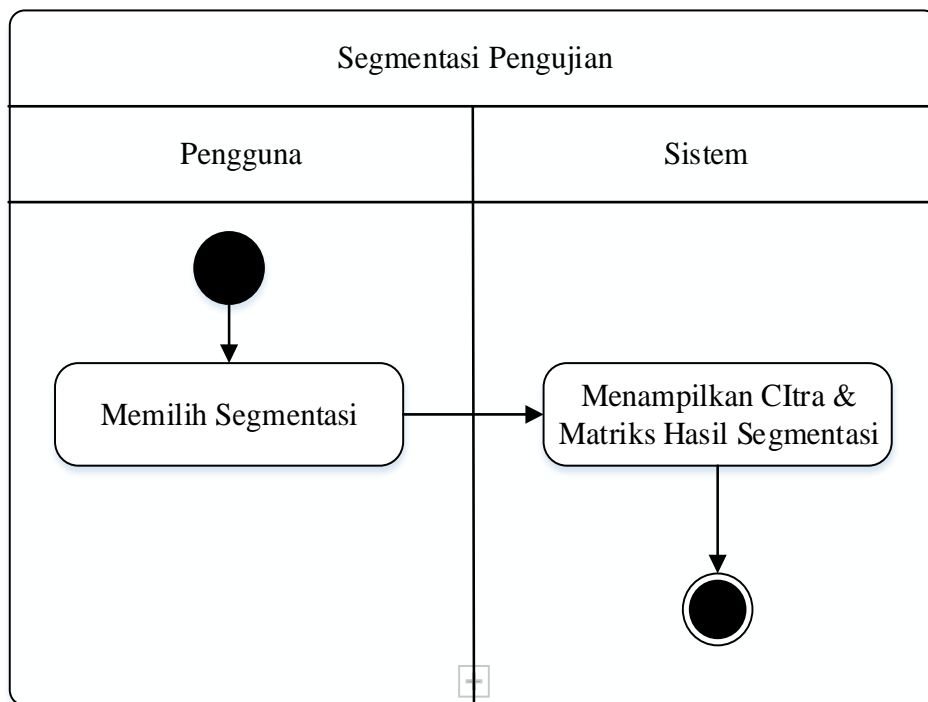
Gambar 3. 31 Activity Diagram Grayscale Pengujian

Gambar 3.31 Merupakan Activity diagram proses *grayscale* dimana aktifitas ini termasuk dalam proses *preprocessing*.



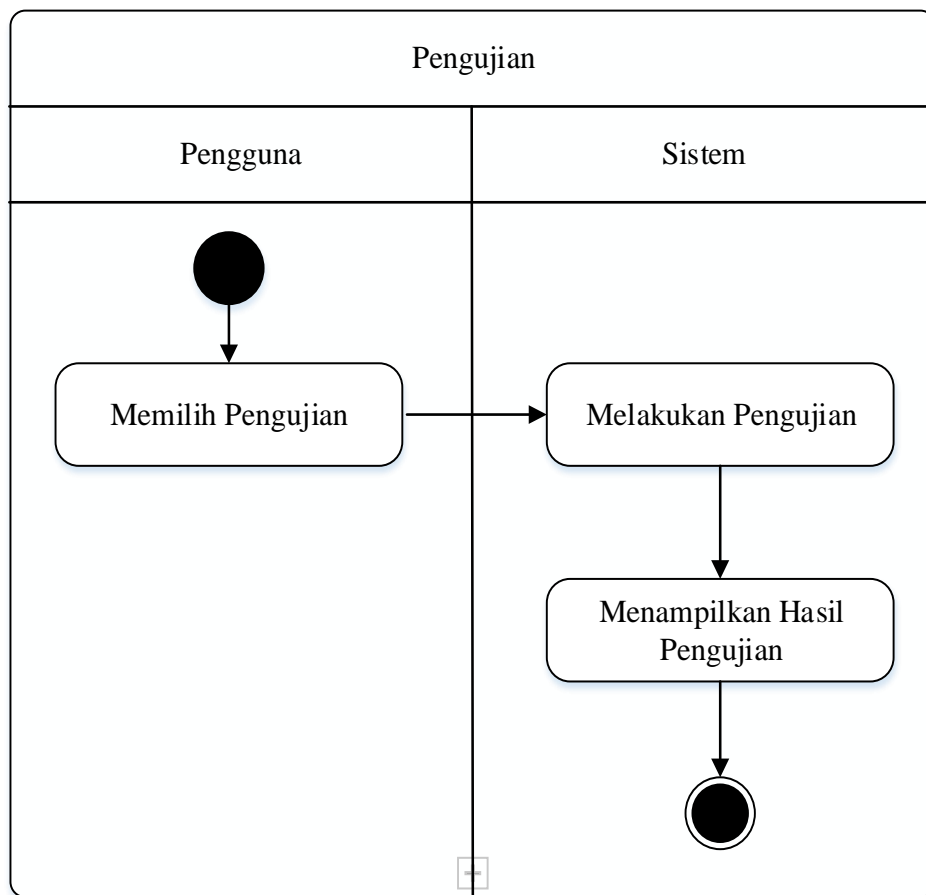
Gambar 3. 32 Activity Diagram *Threshold Pengujian*

Gambar 3.32 merupakan *activity* proses *sauvola thresholding*, aktifitas ini merupakan bagian dari proses *preprocessing* dan dilakukan setelah proses *grayscale* selesai.



Gambar 3. 33 Activity Diagram Segmentasi Pengujian

Gambar 3.33 merupakan *activity* proses Segmentasi, aktifitas ini merupakan bagian dari proses *preprocessing* dan dilakukan setelah proses *grayscale* selesai.



Gambar 3. 34 Activity Diagram Pengujian

Gambar 3.34 merupakan *activity* proses pengujian, aktifitas ini dilakukan setelah proses *preprocessing* selesai, di aktifitas ini kita sudah melakukan tahap akhir pada program dan dapat mengetahui hasil akhir akurasi dari proses pengujian.

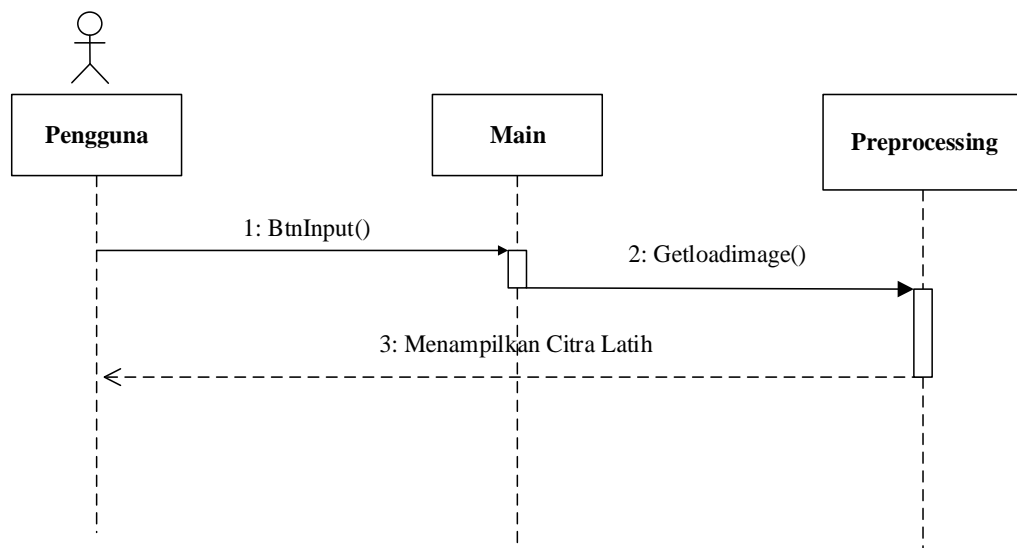
3.2.6.3 Class Diagram

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 3.35.

banyak class dengan berbagai fungsinya. Akan tetapi yang digunakan dalam aplikasi ini hanya *Main* saja. *Main*, *Preprocessing* dan Pengujian merupakan class secara umum yang menyediakan *trigger* kepada semua class yang ada.

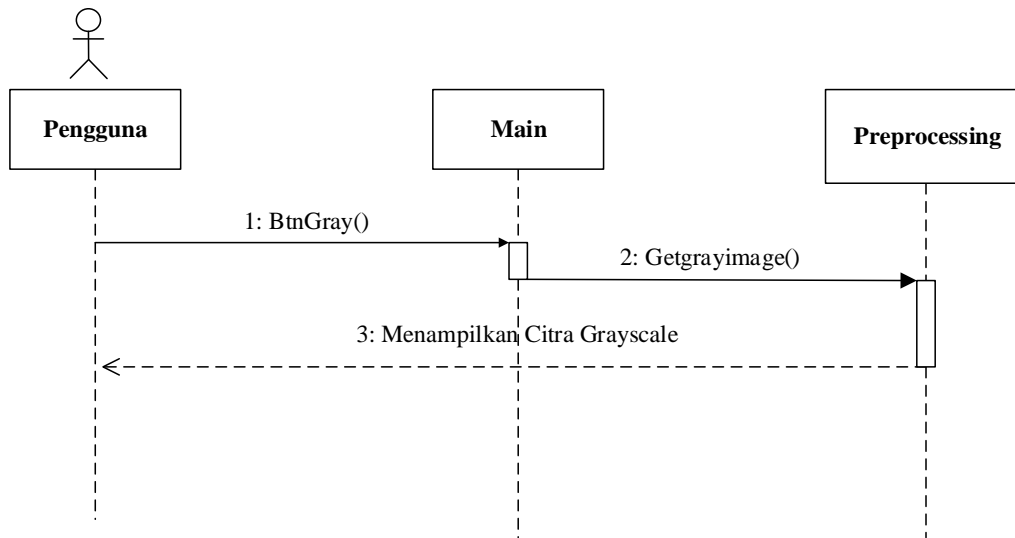
3.2.6.4 Sequence Diagram

Diagram sekuen menggambarkan kelakuan/perilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Berikut merupakan Diagram *Sequence* dari program yang digunakan dipenelitian ini.



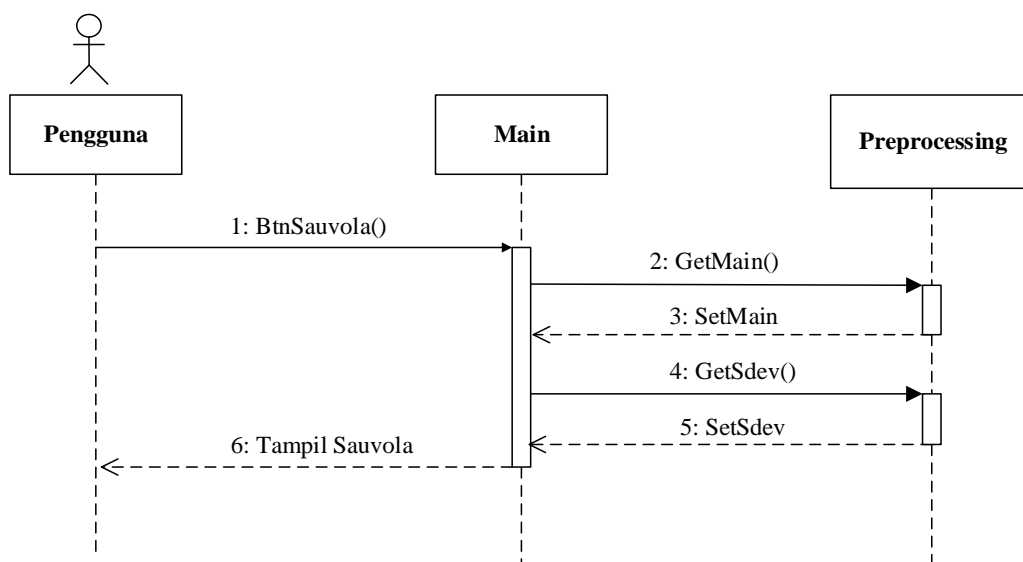
Gambar 3. 36 Sequence Diagram Input Citra

Gambar 3.36 merupakan *sequence* diagram pada proses *Input* Citra. Dalam proses ini kelas-kelas yang digunakan adalah kelas *Main* dan *preprocessing*.



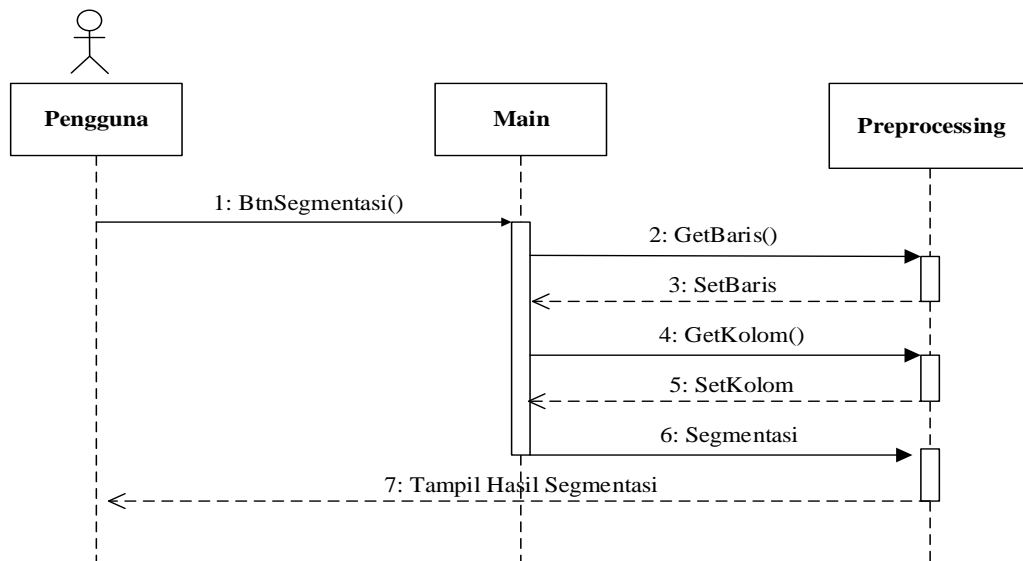
Gambar 3. 37 Sequence Diagram Grayscale

Gambar 3.37 merupakan *sequence* diagram pada proses *grayscale*. Dalam proses ini kelas-kelas yang digunakan adalah kelas **Main** dan *preprocessing*.



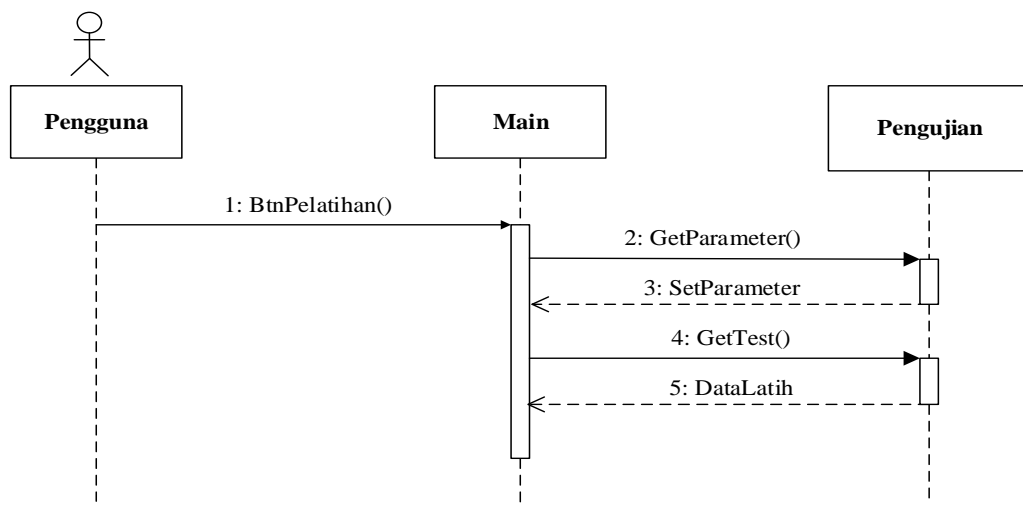
Gambar 3. 38 Sequence Diagram Threshold

Gambar 3.38 merupakan *sequence* diagram pada proses *Threshold*. Dalam proses ini kelas-kelas yang digunakan adalah kelas *Main* dan *preprocessing*.



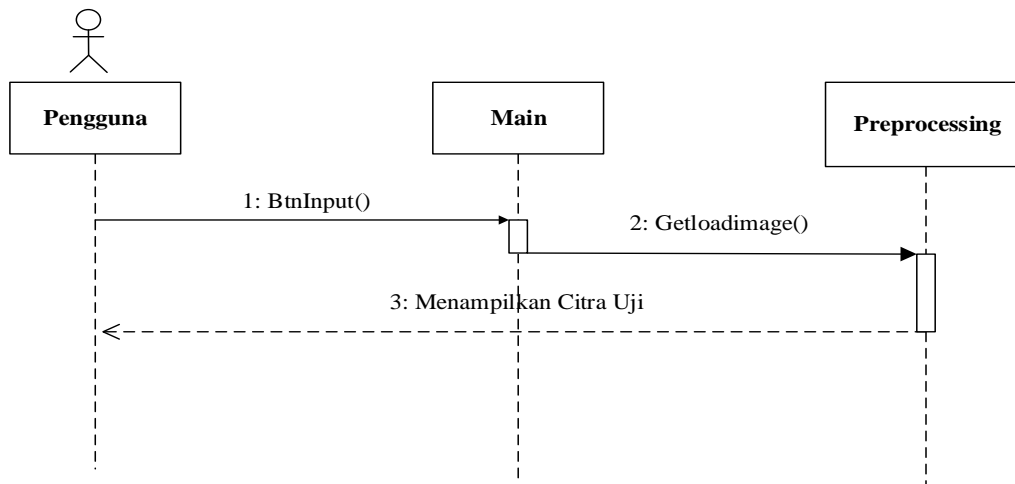
Gambar 3. 39 Sequence Diagram Segmentasi

Gambar 3.39 merupakan *sequence* diagram pada proses Segmentasi. Dalam proses ini kelas-kelas yang digunakan adalah kelas *Main* dan *preprocessing*.



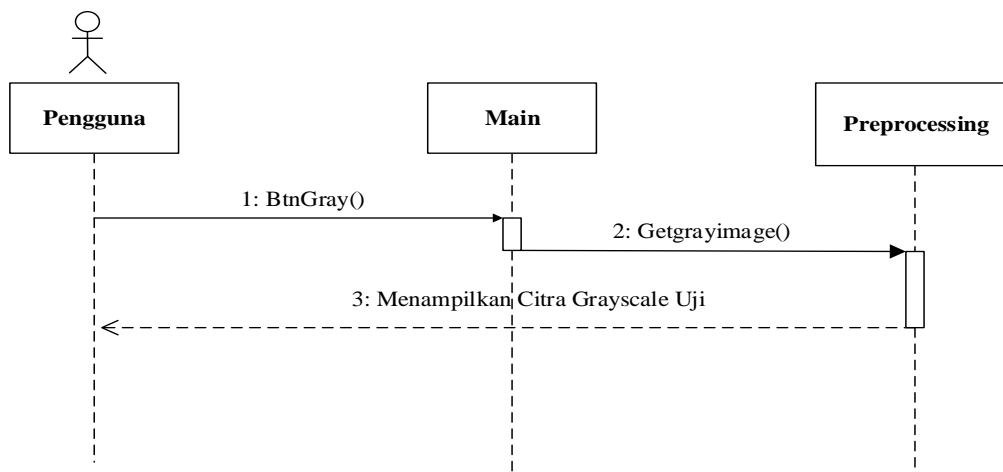
Gambar 3. 40 Sequence Diagram Pelatihan

Gambar 3.40 merupakan *sequence* diagram pada proses pelatihan data *training*. Dimana proses yang dilakukan adalah melakukan pelatihan hasil dari *preprocessing* dan secara otomatis merubah bobot awal dari data training yang digunakan.



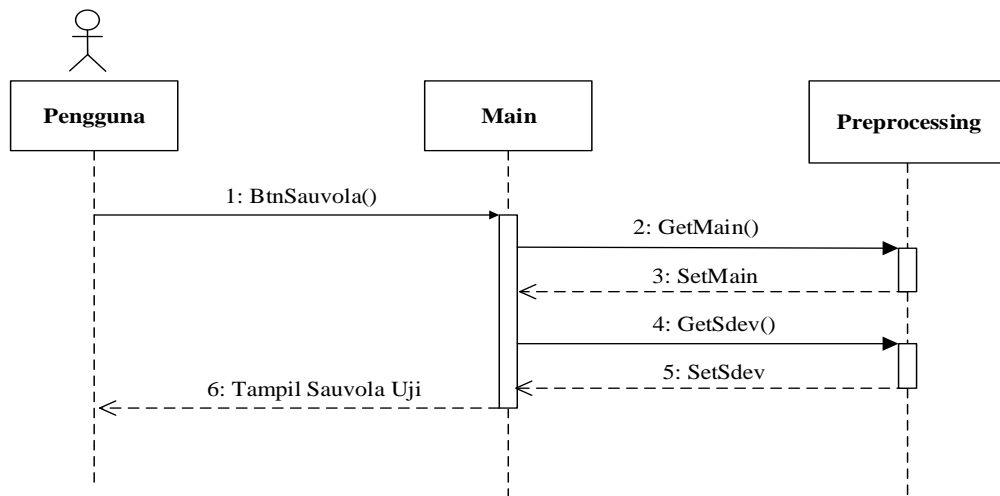
Gambar 3. 41 Sequence Diagram *Input Citra Uji*

Gambar 3.41 merupakan *sequence* diagram pada proses *Input Citra uji*. Dalam proses ini kelas-kelas yang digunakan adalah kelas **Main** dan *preprocessing*.



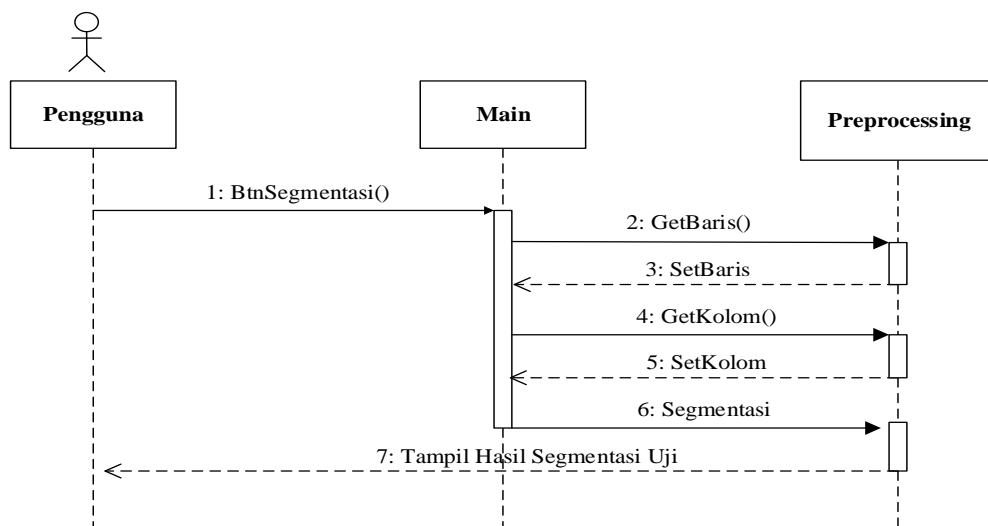
Gambar 3. 42 Sequence Diagram *Grayscale Uji*

Gambar 3.42 merupakan *sequence* diagram pada proses *grayscale* Uji. Dalam proses ini kelas-kelas yang digunakan adalah kelas *Main* dan *preprocessing*.



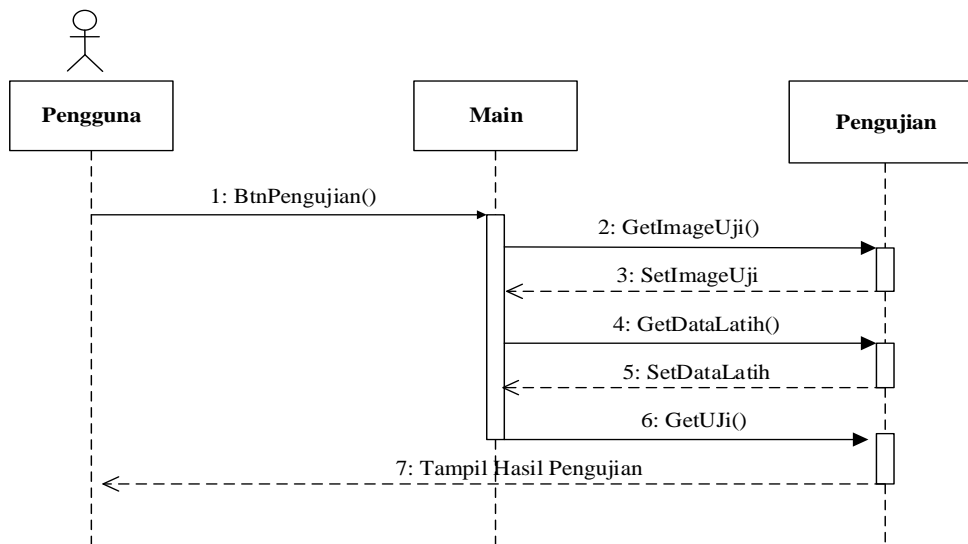
Gambar 3. 43 Sequence Diagram Threshold Uji

Gambar 3.43 merupakan *sequence* diagram pada proses *Threshold* Uji. Dalam proses ini kelas-kelas yang digunakan adalah kelas *Main* dan *preprocessing*.



Gambar 3. 44 Sequence Diagram Segmentasi Uji

Gambar 3.44 merupakan *sequence* diagram pada proses Segmentasi uji. Dalam proses ini kelas-kelas yang digunakan adalah kelas *Main* dan *preprocessing*.



Gambar 3. 45 Sequence Diagram Penguujian

Gambar 3.45 merupakan *sequence* diagram pada proses penguujian CNN. Dalam proses ini, kelas-kelas yang digunakan adalah *Main*, *Preprocessing*.

3.3 Perancangan Sistem

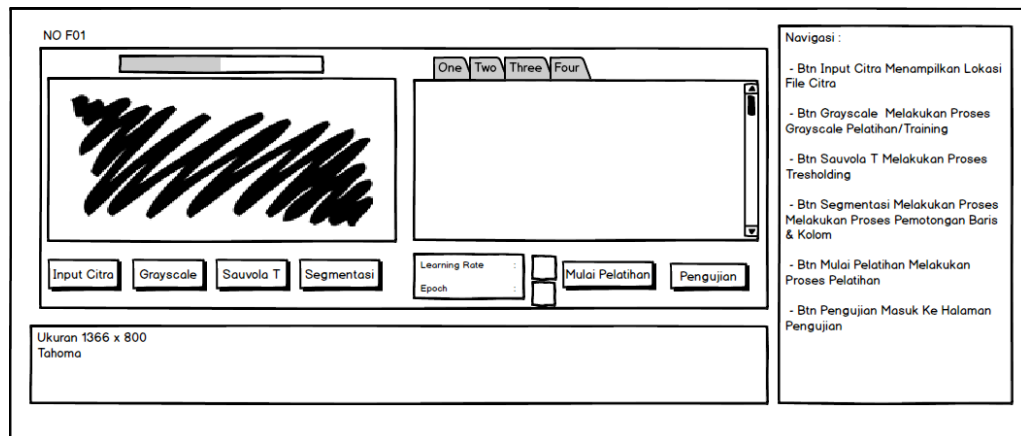
Dalam perancangan sistem ini, dibuat beberapa perancangan seperti perancangan antar muka, pesan (*Alert*) dan jaringan semantik.

3.3.1 Perancangan Antar Muka

Perancangan antar muka merupakan perancangan *Interface* yang dilakukan dari aplikasi yang dibuat. Dalam penelitian ini menggunakan 3 antar muka yang bisa dilihat pada gambar berikut.

3.3.2.3 Antar Muka Pelatihan

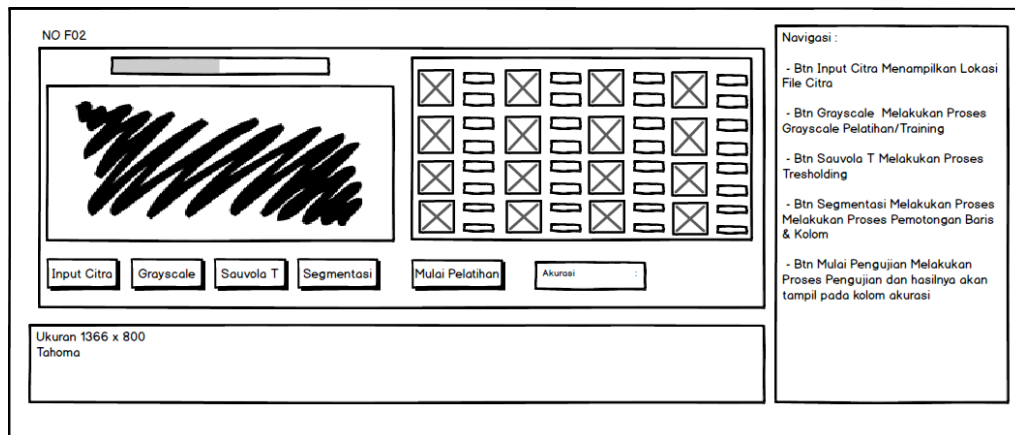
Pada menu pelatihan/*training* akan menampilkan tahapan-tahapan yang sama dengan *preprocessing*, namun pada menu inilah proses pelatihan citra tulisan tangan dilakukan.



Gambar 3. 46 Perancangan Menu Antar Muka Pelatihan

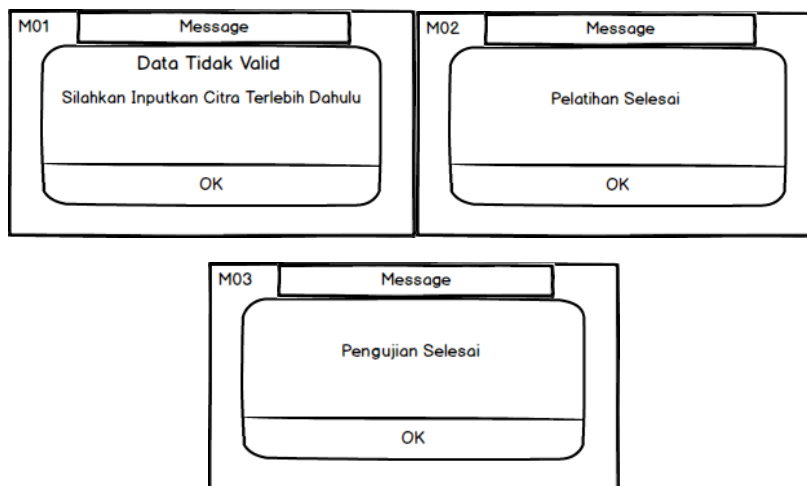
3.3.2.4 Antar Muka Pengujian

Pada menu pengujian/*testing* akan menampilkan proses pengujian, dimana proses ini juga yang menjadi hasil akhir pengenalan tulisan tangan aksara sunda. Pada menu ini tahapan-tahapannya tidak beda jauh dari proses pelatihan, dimana didalamnya kita harus menginputkan citra tulisan tangan yang akan diuji setelah citra berhasil diinputkan maka masuk ke tahap *preprocessing* kemudian citra siap untuk diuji.



Gambar 3. 47 Perancangan Antar Muka Pengujian

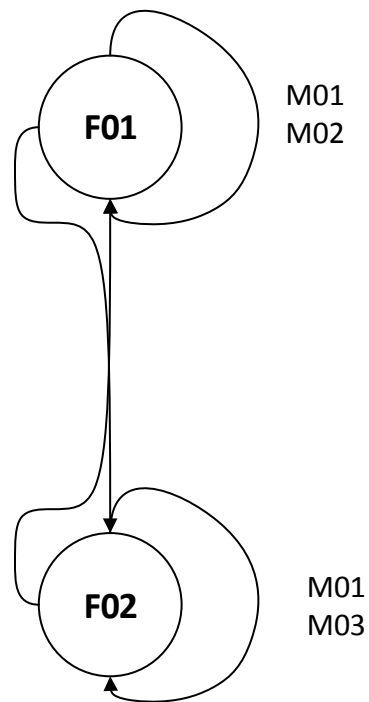
3.3.2 Perancangan Pesan



Gambar 3. 48 Perancangan Pesan

3.3.3 Jaringan Semantik

Jaringan semantik merupakan arsitektur lanjutan yang dibuat untuk memperjelas alur dari perancangan sistem antar muka. Berikut adalah jaringan semantik pada sistem ini :



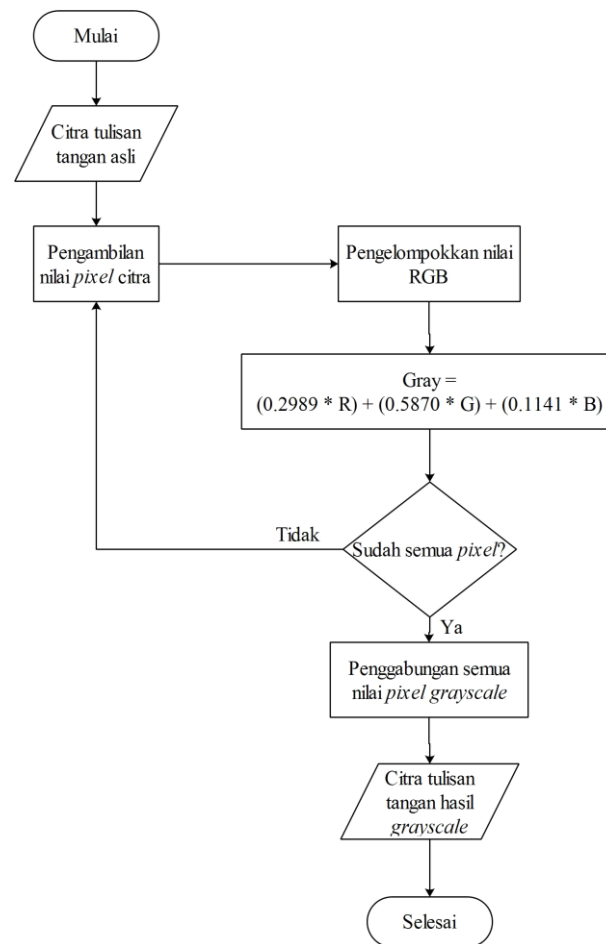
Gambar 3. 49 Jaringan Semantik pada Sistem

3.3.4 Perancangan Prosedural

Pada bagian ini akan dibuat *flowchart* yang didalamnya terdapat deskripsi rinci dari perangkat lunak. Deskripsi tersebut adalah sebagai berikut.

3.3.4.1 Flowchart *Grayscale*

Grayscale berfungsi untuk mengecilkan range warna menjadi 0 sampai dengan 255.



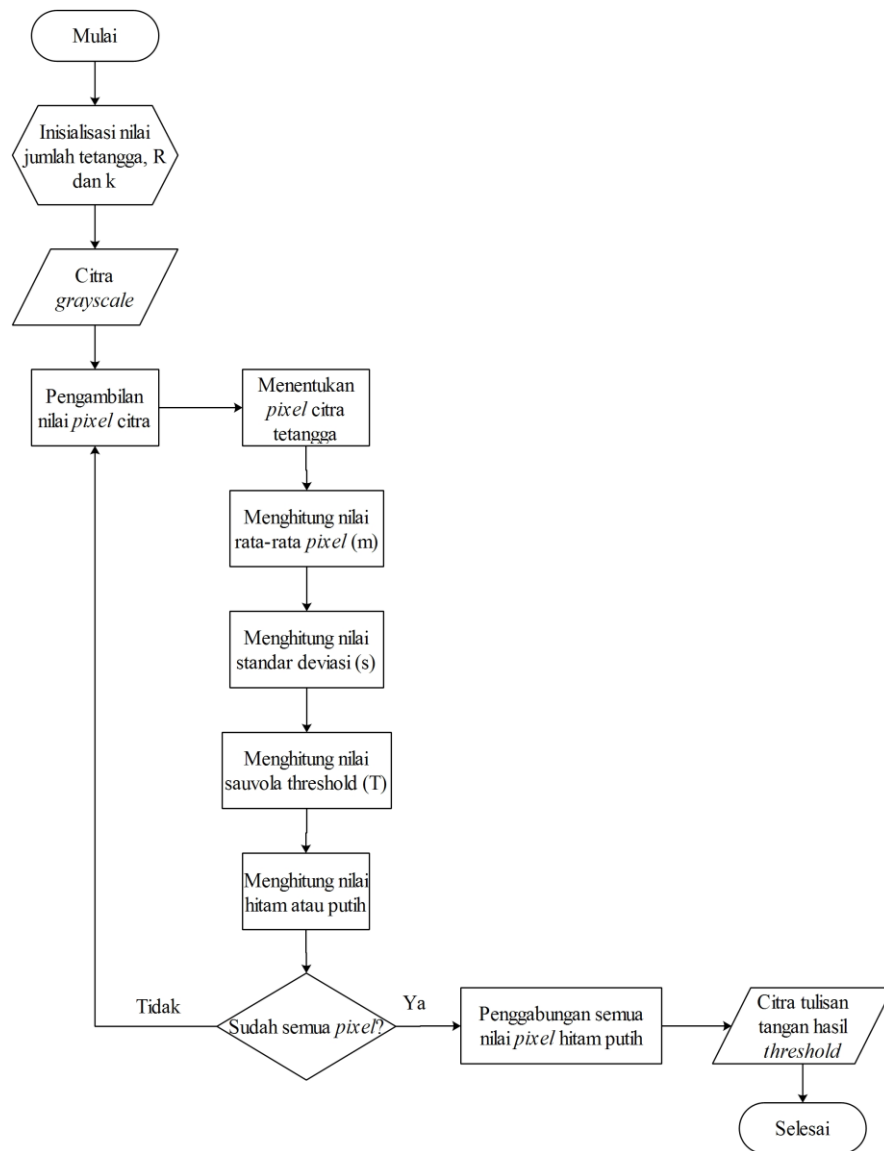
Gambar 3. 50 Flowchart Grayscale

Adapun langkah-langkah yang dilakukan sebagai berikut.

1. Warna citra dikelompokkan berdasarkan nilai *red*, *green* dan *blue*.
2. Kemudian menggunakan persamaan 2.1 , maka akan didapatkan nilai warna *grayscale* citra.

3.3.4.2 Flowchart Threshold

Threshold bertujuan untuk membedakan objek dan *Background* dari citra agar lebih mudah dikenali pada tahapan selanjutnya.

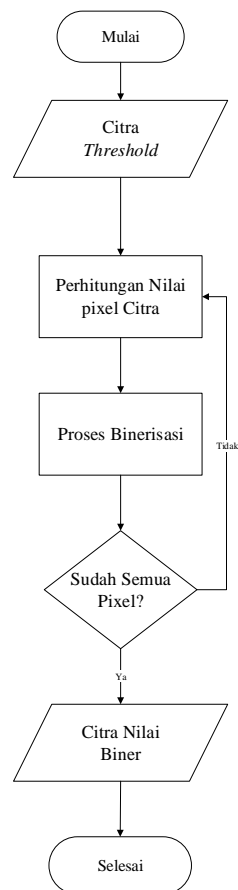


Gambar 3. 51 Flowchart Threshold

Metode *threshold* yang digunakan dalam penelitian ini yaitu menggunakan metode *Sauvola Threshold*. Sauvola merupakan kategori dari metode *local threshold*, dimana nilai ambang pada setiap *pixel* tergantung dari jumlah tetangga yang digunakan. Gambar 3.51 merupakan *flowchart* proses *sauvola threshold*.

3.3.4.3 Flowchart Binerisasi

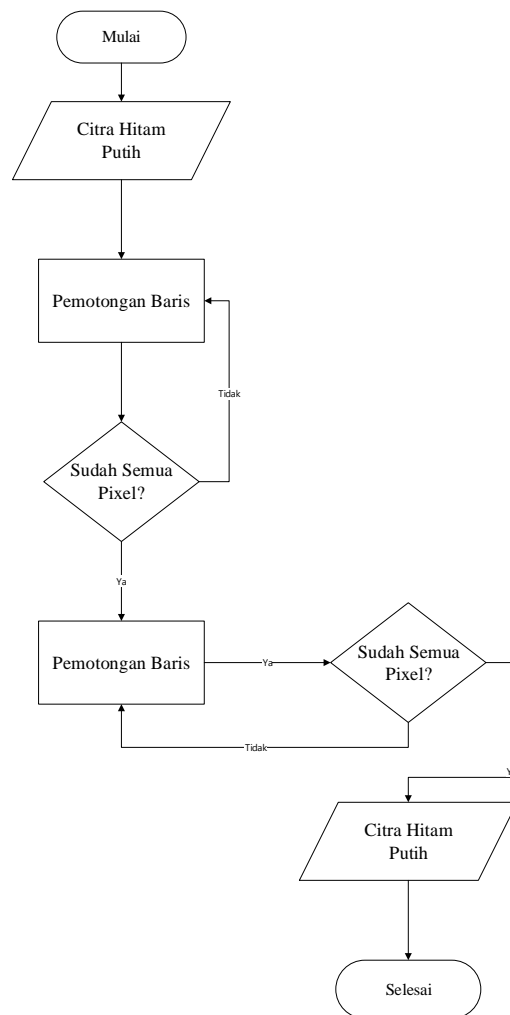
Binerisasi adalah merubah nilai *threshold* ke dalam nilai biner dengan merubah nilai 255 menjadi angka 0 dan nilai 0 menjadi angka 1.



Gambar 3. 52 Flowchart Binerisasi

3.3.4.4 Flowchart Segmentasi

Segmentasi merupakan proses pemotongan dilakukan untuk setiap baris (*horizontal*) pada citra input terlebih dahulu, kemudian melakukan pemotongan setiap kolom (*vertical*) pada setiap citra hasil pemotongan secara baris.

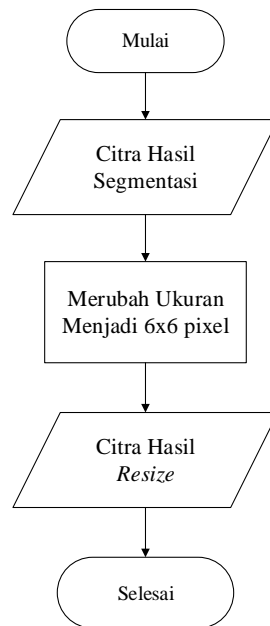


Gambar 3. 53 Flowchart Segmentasi

Pemotongan untuk setiap baris (horizontal) dilakukan dengan menelusuri pixel citra dari pixel baris ke-1. Untuk pemotongan setiap kolom (vertical) sama seperti pemotongan baris, hanya saja penelusuran citra dari pixel kolom ke-1.

3.3.4.5 Flowchart Resize

Resize adalah proses mengubah ukuran suatu citra menjadi lebih besar atau kecil dari ukuran citra awal dengan ukuran yang telah ditetapkan sebelumnya.

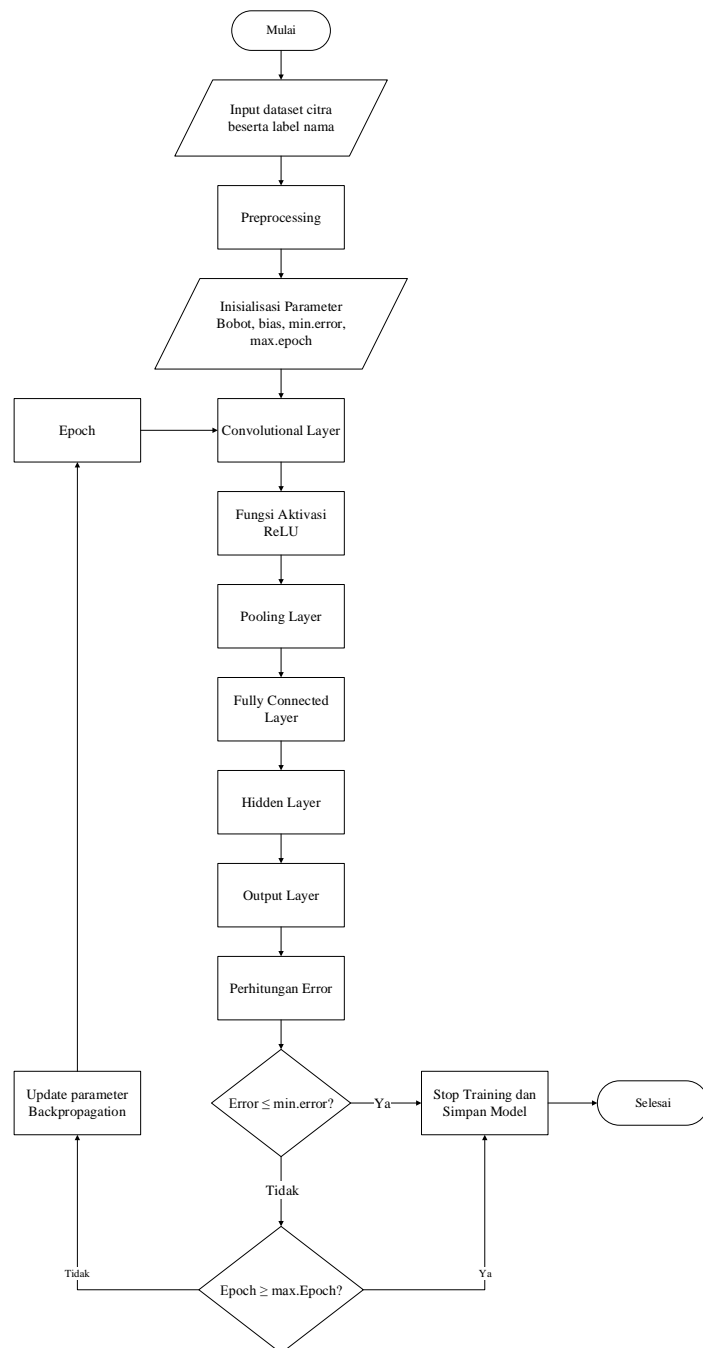


Gambar 3. 54 *Flowchart Resize*

Dalam penelitian ini karakter-karakter yang sudah tersegmen pada tahap segmentasi di-resize menjadi ukuran 6x6 pixel, ukuran tersebut digunakan berdasarkan penelitian yang telah dilakukan sebelumnya agar ukuran setiap segmentasi bernilai sama.

3.3.5.3 *Flowchart CNN Training*

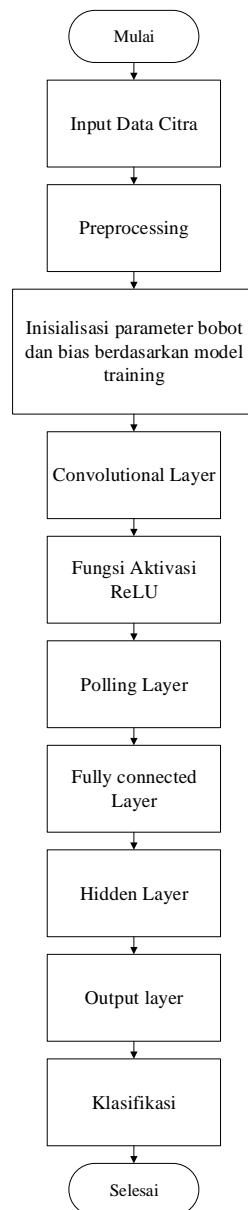
CNN Training terdiri dari beberapa tahap, yaitu tahap inisialisasi, tahap *feedforward*, tahap *backpropagation*, dan tahap *update* bobot.



Gambar 3. 55 *Flowchart CNN Training*

3.3.5.4 Flowchart CNN Testing

Pada proses testing tidak jauh berbeda dengan proses training, perbedaan hanya pada proses training kita bertujuan untuk mengupdate bobot namun pada proses testing tidak mengambil bobot melainkan menggunakan bobot hasil training.



Gambar 3. 56 Flowchart CNN Testing