BAB 2

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Susmikanthi dan Adhiyaksa (2005) dalam penelitiannya menggunakan metode *Perceptron* untuk mengidentifikasi huruf. Dari penelitiannya disimpulkan bahwa dalam merancang jaringan neural yang harus diperhatikan adalah spesifikasi yang akan diidentifikasi.

Pujiyanta (2009) dalam penelitiannya menggunakan metode *perceptron* untuk mengenali objek sederhana. Dari penelitiannya disimpulkan *perceptron* dapat digunakan untuk mengenali objek sederhana dengan ketepatan yang cukup baik.

Setiawan et al (2010) dalam penelitiannya menggunakan metode *Perceptron* untuk mengidentifikasi dokumen yang rusak karena banjir. Dalam penelitiannya disimpulkan bahwa tingkat akurasi tergantung dari banyaknya jumlah huruf yang dilatih.

Jayati (2008) dalam penelitiannya menggunakan metode *thinning* dalam pengolahan citra menyimpulkan metode *thinning* dapat diimplementasikan dalam OCR (*Optical Character Recognition*).

Fahmy (2000) dalam penelitiannya Automatic Recognition Of Handwritten Arabic Characters Using Their Geometrical Features menggunakan metode *thinningBinary Region* sebagai awal pengenalan karakter Arab.Hasilnya, kerangka gambar hasil *thinning* dapat digunakan untuk mewakili gambar saat dilakukan pelatihan dan pengenalan.

Wahyono et al (2009) dalam penelitiannya menggunakan algoritma *perceptron* untuk pengenalan huruf. Dari penelitiannya disimpulkan algoritma *perceptron* dapat digunakan untuk pengklasifikasian huruf.

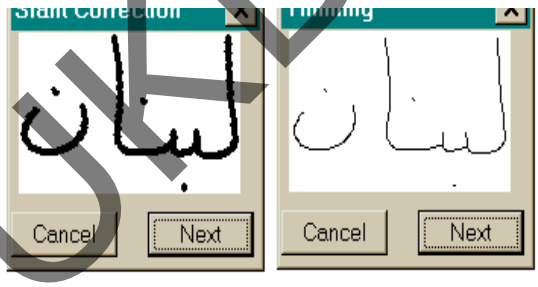
2.2. Landasan Teori

2.2.1. Gambaran Sistem

Penelitian ini akan membahas tentang pengenalan citra huruf yang menggunakan metode *thinningBinary Region* dan metode *Perceptron* pada proses pengenalan. Pada penelitian ini citra huruf yang akan dikenali akan melalui proses *thinning* untuk menipiskan citra huruf. Proses *thinning* ini akan menggunakan metode *Binary Region*. Setelah melalui proses *thinning*, citra hasil *thinning* akan dikenali dengan metode *Perceptron*.

2.2.2. *Thinning*

Thinning adalah metode yang digunakan untuk mencari dasar / rangka dari suatu gambar. *Thinning* dapat digunakan untuk mencari informasi dari suatu gambar dengan menghilangkan informasi – informasi yang tidak diperlukan tanpa menghilangkan informasi aslinya. Gambar 2.1 adalah contoh penerapan *thinning* pada citra.



Gambar 2.1.sebelum *thinning* (kiri), setelah *thinning* (kanan)

Dikutip dari Fahmy, Maged M.M. (2007) Automatic Recognition Of Handwritten Arabic CharactersUsing Their Geometrical Features. (http://sic.ici.ro/sic2001_2/art1.htm)

2.2.3. *Binary Region*

Binary Region adalah salah satu metode untuk melakukan *thinning* (Jayati,2008). Metode ini dapat diterapkan pada gambar biner yang berisi informasi 1 sebagai *foreground* dan 0 sebagai *background*. Dalam penerapannya metode ini ada 2 langkah dan menggunakan konsep *pixel – pixel* tetangga. Metode ini diterapkan pada *contour points* dari area dimana *contour points* adalah sembarang *pixel* dengan nilai 1 dan memiliki setidaknya

1 dari 8 tetangganya bernilai 0. Konsep Algoritma *Thinning Binary Region* dapat dilihat dari Gambar 2.2.

P9	P2	P3
P8	P1	P4
P7	P6	P5

Gambar 2.2.Konsep *pixel* tetangga untuk *Thinning Binary Region*

Langkah 1 :

1. Beri tanda *P1* jika memenuhi syarat berikut :

   a. $2 \leq N(P1) \leq 6$

   b. $S(P1) = 1$

   c. $P2 \cdot P4 \cdot P6 = 0$

   d. $P4 \cdot P6 \cdot P8 = 0$

keterangan :

a. N adalah *pixels* tetangga dari *P1* yang tidak bernilai 0.
   Sehingga N(P1) = P2+P3+P4+P5+P6+P7+P8+P9.

b. S(P1) adalah jumlah transisi dari 0 ke 1 dalam urutan P2,P3,P4,P5,P6,P7,P8,P9. Dapat dilihat pada Gambar 2.3.

1	0	1
1	1	0
1	1	1

Gambar 2.3. Contoh transisi dari 0 ke 1.

3. Jika telah memenuhi syarat – syarat diatas dan telah ditandai, maka P1 diset menjadi 0.

Langkah 2 :

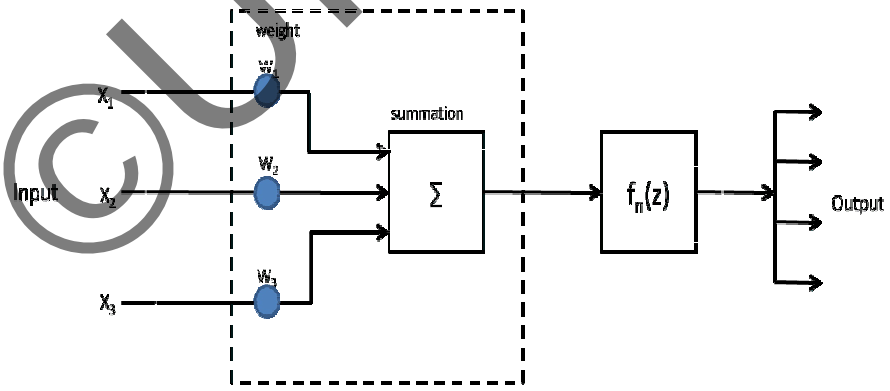
1. Beri tanda *P1* jika memenuhi syarat berikut :
   a. $2 \leq N(P1) \leq 6$
   b. $S(P1) = 1$
   c. $P2 . P4 . P8 = 0$
   d. $P2 . P6 . P8 = 0$
2. Jika telah memenuhi syarat – syarat diatas dan telah ditandai, maka P1 diset menjadi 0.

Dalam 1 kali iterasi dilakukan langkah 1 dan 2. Iterasi ini terus dilakukan sampai tidak ada *point* yang dapat dihapus lagi. Sehingga dihasilkan kerangka dari sebuah gambar.

2.2.4. *Perceptron*

Perceptron merupakan metode jaringan syaraf tiruan yang diperkenalkan oleh Rosenbllat 1959. Menurut Ben Krosedan Patrick Van Der Smagt, (1996,26) *perceptron* terdiri dari beberapa unit masukan dan ditambah 1 bias.

Arsitektur *Perceptron* dapatdilihatpadaGambar 2.4 :



Gambar 2.4 Arsitektur *Perceptron*

Dikutipdari: Graupe, Daniel. (2007). Principles Of Artificial neural Networks. 2nd Edition, Singapore

Algoritma Pembelajaran *Perceptron* :

Langkah 1 : Inisialisai$w$, b, α, θ.

Langkah 2 : Hitung unit *output*$y\_in = b + \sum_i x_i w_i$.

$$y = \begin{cases} 1 \text{ jika } y\_in > \Theta \\ 0 \text{ jika } -\Theta \le y\_in \le \Theta \\ -1 \text{ jika } y\_in < -\Theta \end{cases}$$

Jika ada y yang keluarannya tidak sama dengan target (t) maka lakukan langkah 3.

Langkah 3 : Update nilai $b$dan w

$w_i(baru) = w_i(lama) + \alpha t x_i$

$b(baru) = b(lama) + \alpha t$

Algoritma Pengenalan *Perceptron* :

Langkah 1 : Inisialisasi bobot dari hasil pelatihan sebelumnya.

$y\_in = b + \sum_i x_i w_i$

$$y = \begin{cases} 1 \text{ jika } y\_in > \Theta \\ 0 \text{ jika } -\Theta \le y\_in \le \Theta \\ -1 \text{ jika } y\_in < -\Theta \end{cases}$$

Langkah 2: Jika y sama dengan target (t) maka *input* dikenali.

Keterangan :

w = bobot

b = bias

α = laju pemahaman / *learning rate*

Θ = *treshold*

x = unit *input*

y = unit *output*

t = target

2.3. Implementasi *Perceptron* Pada Pengenalan Citra Huruf

0	1	1	0
1	0	0	1
1	1	1	1
1	0	0	1
1	0	0	1

Gambar 2.5 contoh huruf A training

Pada Gambar 2.5 adalah contoh dari gambar huruf A (4 x 5 *pixel*). Pada salah satu rumus *perceptron* ($y\_in = b + \sum_i x_i w_i$), $x_i$pada rumus diartikan *pixel* pada gambar. Karena gambar berukuran 4 x 5, maka ada $x_1 - x_{20}$. Maka rumus pelatihan akan menjadi seperti berikut :

$$y\_in = b + x_1w_1 + x_2w_2 + x_3w_3 + \ldots\ldots + x_{20}w_{20}$$

Jika target untuk huruf A terdiri dari 3 *bit* yaitu 1 -1 1, maka langkah diatas harus terus diulang pada proses pelatihan dengan syarat bobot (*w*) dan bias (*b*) harus diperbarui. Perulangan akan berhenti saat *y* = target (*t*) yaitu 1 -1 1. Karena 3 *bit* maka rumus pelatihan akan menjadi :

$$y\_in1 = b + x_1w_1 + x_2w_2 + x_3w_3 + \ldots\ldots + x_{20}w_{20}$$

$$y\_in2 = b + x_1w_1 + x_2w_2 + x_3w_3 + \ldots\ldots + x_{20}w_{20}$$

$$y\_in3 = b + x_1w_1 + x_2w_2 + x_3w_3 + \ldots\ldots + x_{20}w_{20}$$

y_in1 harus sama dengan 1, *y_in2* harus sama dengan -1, *y_in3* harus sama dengan 1.

Untuk proses pengenalan akan diterapkan seperti berikut :

0	1	1	0
1	0	0	1
1	1	1	1
1	0	0	1
1	0	0	1

Gambar 2.6 contoh huruf A uji

Pada Gambar 2.6 adalah contoh citra huruf A yang akan dikenali. Sebelumnya inisialisasi bobot dari hasil pelatihan sebelumnya. Setelah itu terapkan rumus pengenalan seperti berikut :

$$y\_in1 = b + x_1w_1 + x_2w_2 + x_3w_3 + \ldots\ldots + x_{20}w_{20}$$

$$y\_in2 = b + x_1w_1 + x_2w_2 + x_3w_3 + \ldots\ldots + x_{20}w_{20}$$

$$y\_in3 = b + x_1w_1 + x_2w_2 + x_3w_3 + \ldots\ldots + x_{20}w_{20}$$

Karena target A adalah 1 -1 1, agar *input* dikenali maka *y_in1* harus sama dengan 1, *y_in2* harus sama dengan -1, *y_in3* harus sama dengan 1. Jika salah satu dari *y_in* ada yang tidak sama dengan target, maka *input* tidak dikenali.

BAB 3

PERANCANGAN SISTEM

3.1. Spesifikasi Sistem

3.1.1. Perangkat Lunak

Dalam penelitian, pembangunan program menggunakan spesifikasi sebagai berikut :

1. Sistem opresai Windows 7 Ultimate 64 *bit*
2. Microsoft Visual Basic 2010
3. C#
4. Microsoft Acces

3.1.2. Perangkat Keras

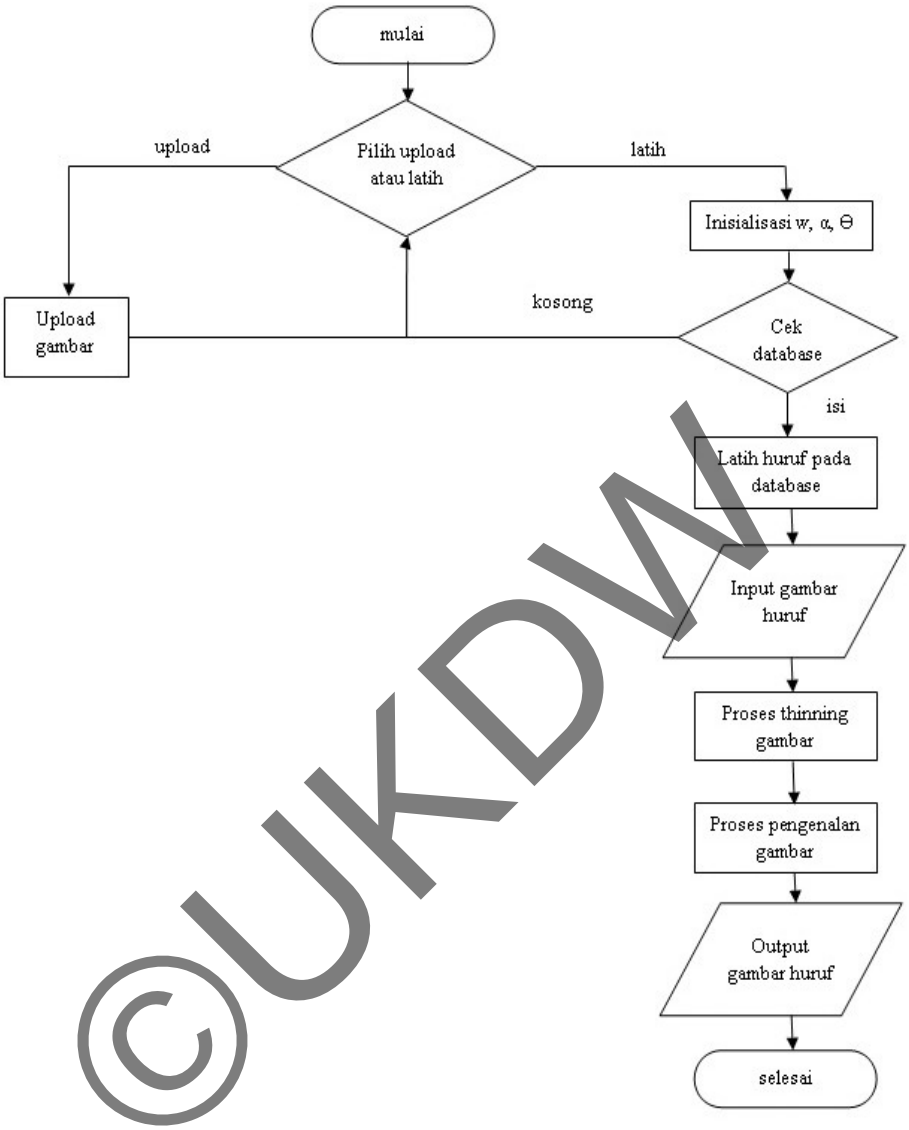
Spesifikasi perangkat keras yang digunakan dalam pembangunan program untuk penelitian sebagai berikut :

1. Processor Intel Core i3 2.00 GHz
2. Memory 2048 MB
3. Harddisk 320 MB

3.2. *Flowchart*

3.2.1. *Flowchart* Program

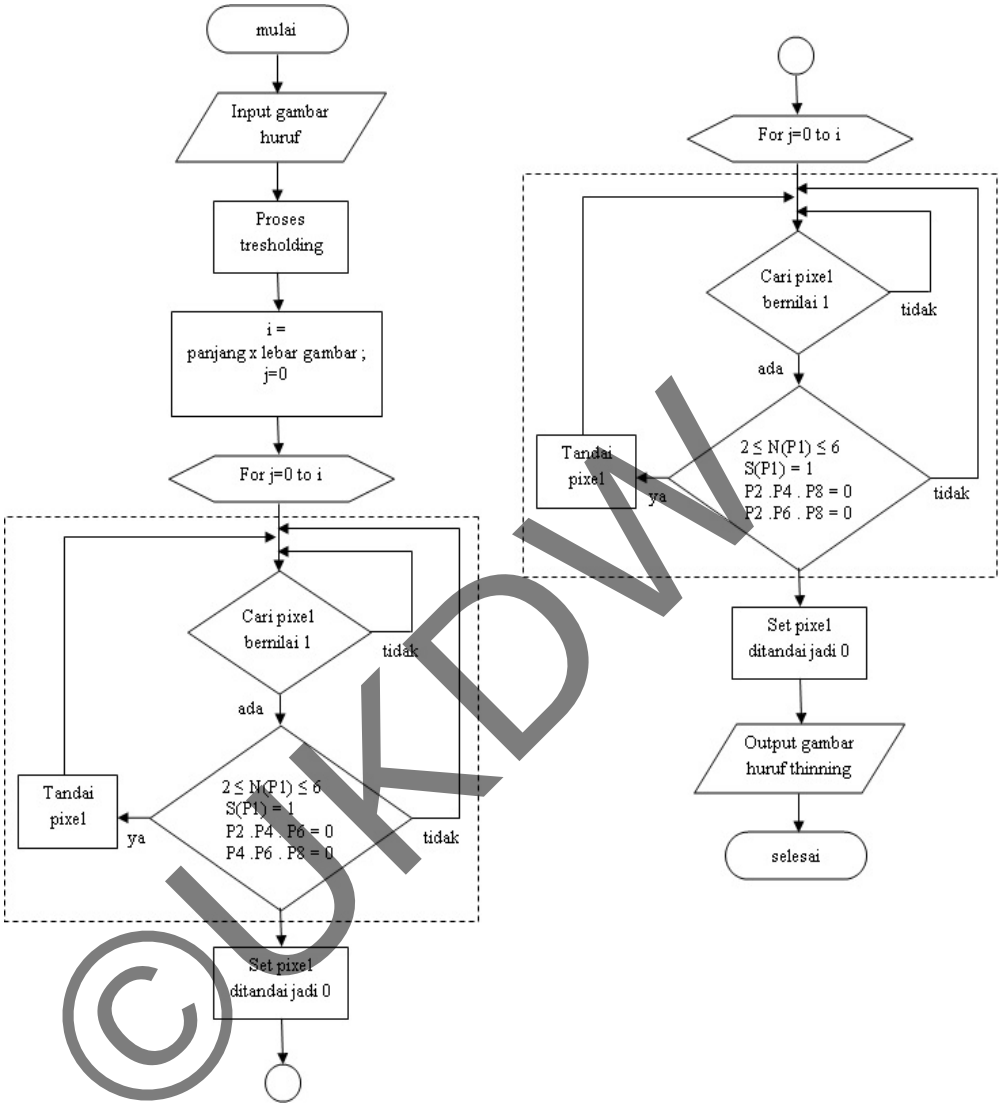
Flowchart program menggambarkan urutan langkah penggunaan program oleh *user*. *User* dapat memilih untuk meng*upload* huruf baru untuk pada *database* atau langsung melakukan proses pelatihan dan pengenalan. Jika *database* masih kosong maka *user* diharuskan untuk meng*upload* gambar terlebih dulu. Jika *database* telah terisi, maka *user* dapat langsung melakukan proses pelatihan dan pengenalan. *Flowchart* program dapat dilihat pada Gambar 3.1.



Gambar 3.1. Flowchart program

3.2.2. *Flowchart Thinning*

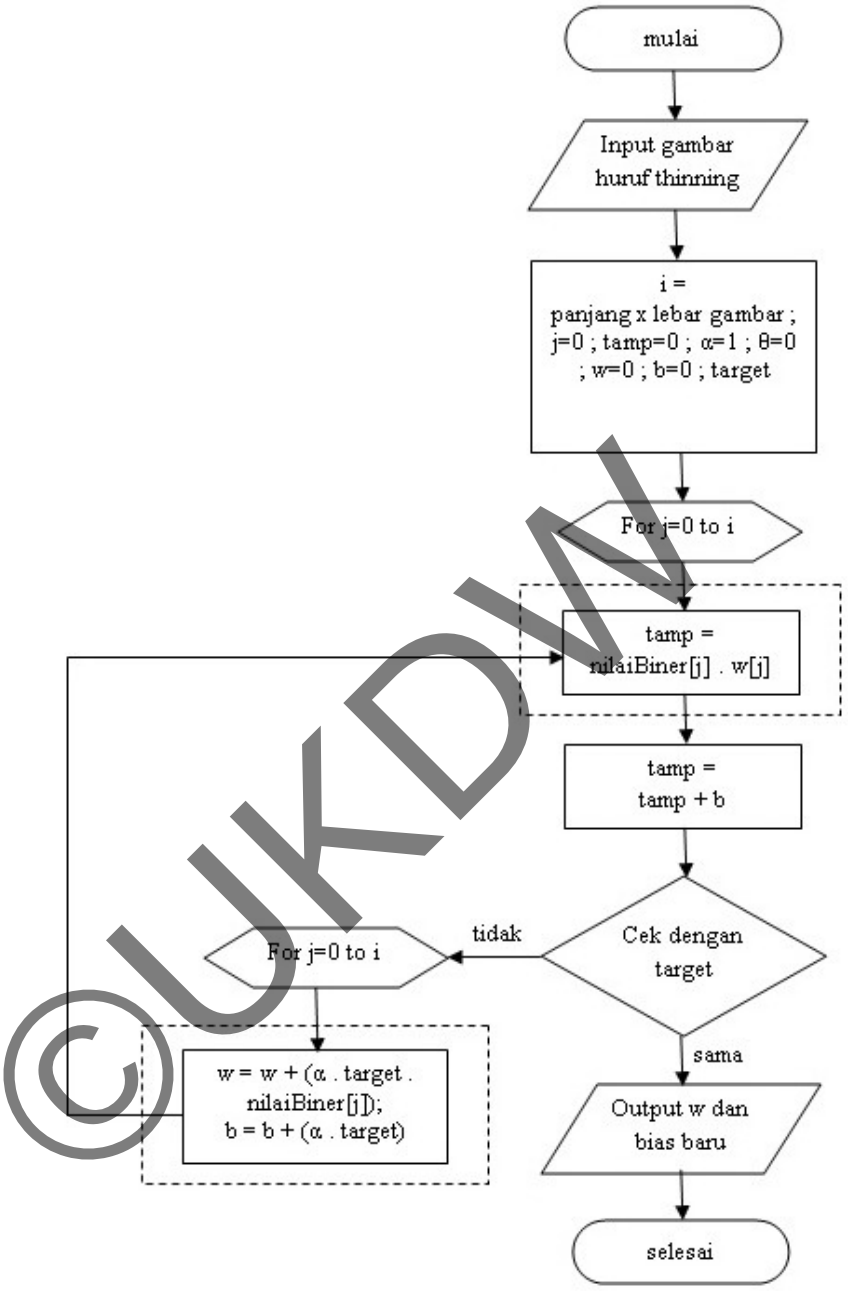
Flowchart thinning menggambarkan urutan langkah yang terjadi pada proses *thinning* gambar. Proses *thinning* ini terjadi pada bagian *upload* gambar untuk disimpan di *database* dan bagian pengenalan. *Flowchart thinning* dapat dilihat pada Gambar 3.2.



Gambar 3.2. Flowchart thinning

3.2.3. *Flowchart* Pelatihan *Perceptron*

Flowchart pelatihan *perceptron* menggambarkan urutan langkah yang terjadi pada proses pelatihan. Proses ini dilakukan pada gambar yang sudah di *thinning* yang tersimpan pada *database*. *Flowchart* pelatihan *perceptron* dapat dilihat pada Gambar 3.3.

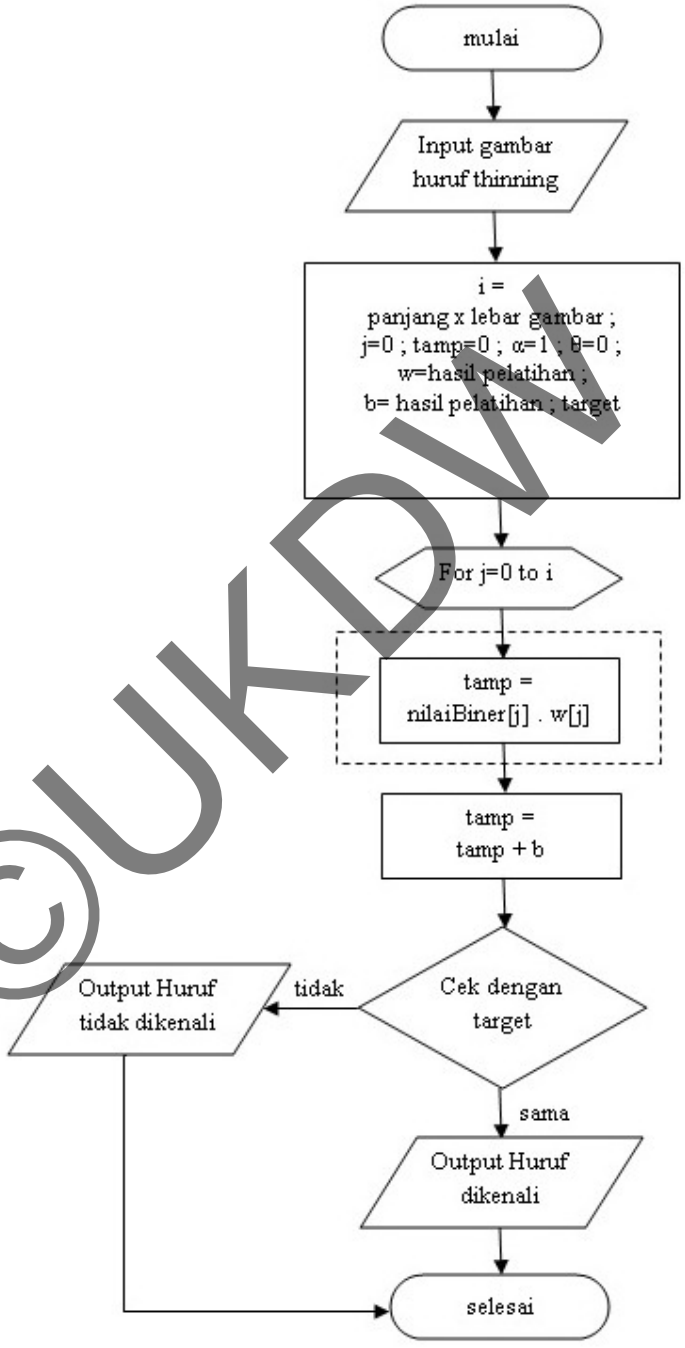


Gambar 3.3. Flowchart pelatihan pada algoritma perceptron

3.2.4. *Flowchart* Pengenalan *Perceptron*

Flowchart pelatihan *perceptron* menggambarkan urutan langkah yang terjadi pada proses pengenalan.

Proses ini dilakukan pada gambar hasil *thinning* yang di*input*kan *user* pada bagian pengenalan. *Flowchart* pengenalan *perceptron* dapat dilihat pada Gambar 3.4.



Gambar 3.4. Flowchart pengenalan pada algoritma perceptron

3.3. Rancangan *Database*

Pada penelitian ini *database* program akan menggunakan Microsoft Acces. *Database* terdiri dari 1 tabel yang menyimpan informasi dari huruf – huruf untuk proses pembelajaran. Berikut ini adalah rancangan tabel *database*.

Tabel 3.1. Rancangan tabel *database*

Nama	Tipe data
ID	Autonumber
Letter	Text
Target	Text
Font	Text
Binary_data	Memo

Keterangan :

1. ***ID***

   ID digunakan untuk menjadi pembeda untuk tiap data pada *database*. Karena menjadi pembeda, maka *ID* antara data satu dengan yang lain tidak sama. Digunakan *autonumber* agar otomatis sehingga *user* tidak perlu mengisi *ID* dan juga untuk menghindari kemungkinan salah *input* dari *user*.

2. ***Letter***

   Letter pada *database* merupakan huruf – huruf (A - Z) yang di*upload* oleh *user* untuk menjadi bahan pelatihan.

3. **Target**

   Target digunakan untuk pembeda antar huruf. Masing – masing huruf mempunyai target berbeda yang merupakan acuan bagi proses pelatihan *perceptron*. Dikarenakan jumlah huruf yang ada 26, maka

target yang digunakan terdiri dari 5 *bit*, artinya ada 2^5 kombinasi. Pemberian target pada masing – masing huruf dapat dilakukan secara acak, tidak ada aturan dalam pemberian target.Daftar target untuk masing – masing huruf dapat dilihat pada Tabel 3.2.

Tabel 3.2. Daftar target masing – masing huruf

Huruf	**Target**
A	1 1 1 1 1
B	1 1 1 1 -1
C	1 1 1 -1 1
D	1 1 1 -1 -1
E	1 1 -1 1 1
F	1 1 -1 1 -1
G	1 1 -1 -1 1
H	1 1 -1 -1 -1
I	1 -1 1 1 1
J	-1 1 -1 -1 -1
K	1 -1 1 -1 1
L	1 -1 1 -1 -1
M	1 -1 -1 1 1
N	-1 1 1 -1 -1
O	1 -1 -1 -1 1
P	1 -1 -1 -1 -1
Q	-1 1 1 1 1
R	-1 1 1 1 -1
S	-1 1 1 -1 1
T	1 -1 -1 1 -1
U	-1 1 -1 1 1
V	-1 1 -1 1 -1
W	-1 1 -1 -1 1
X	1 -1 1 1 -1
Y	-1 -1 1 1 1
Z	-1 -1 -1 -1 -1

4. ***Font***

   Jenis huruf (contoh : Arial, Times New Roman) untuk masing – masing *Letter*. Digunakan agar *user* tahu jenis huruf apa saja yang sudah disimpan di *database*.

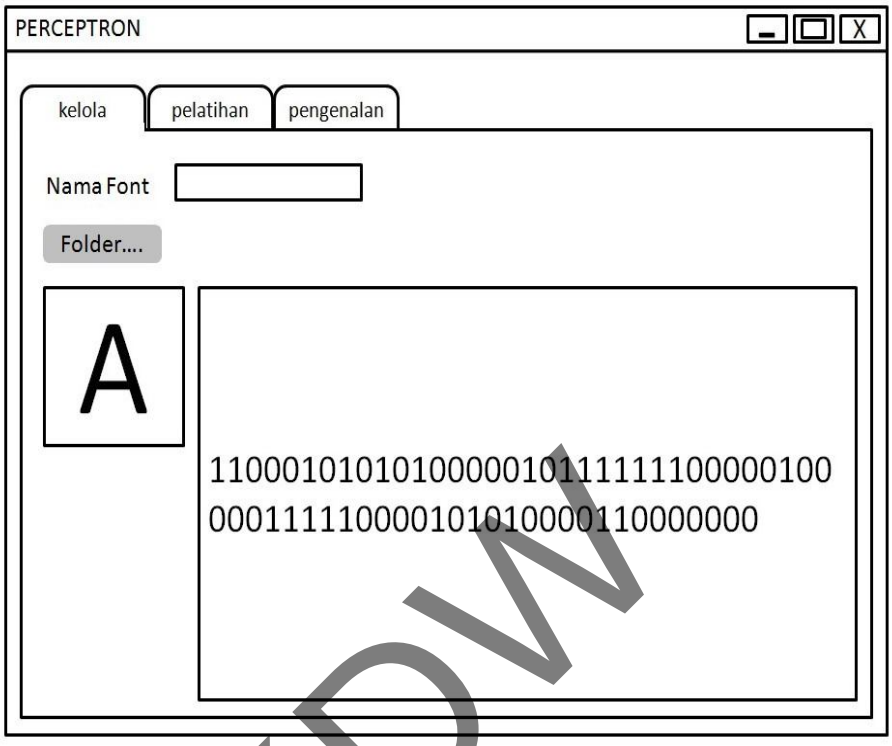
5. ***Binary data***

   Gambar huruf – huruf yang di*upload user* ke *database* akan disimpan dalam bentuk biner (1 dan 0). Hal ini dilakukan agar dapat diproses dengan *perceptron*.

3.4. Rancangan Antarmuka

3.4.1. Form Kelola Data *Font*

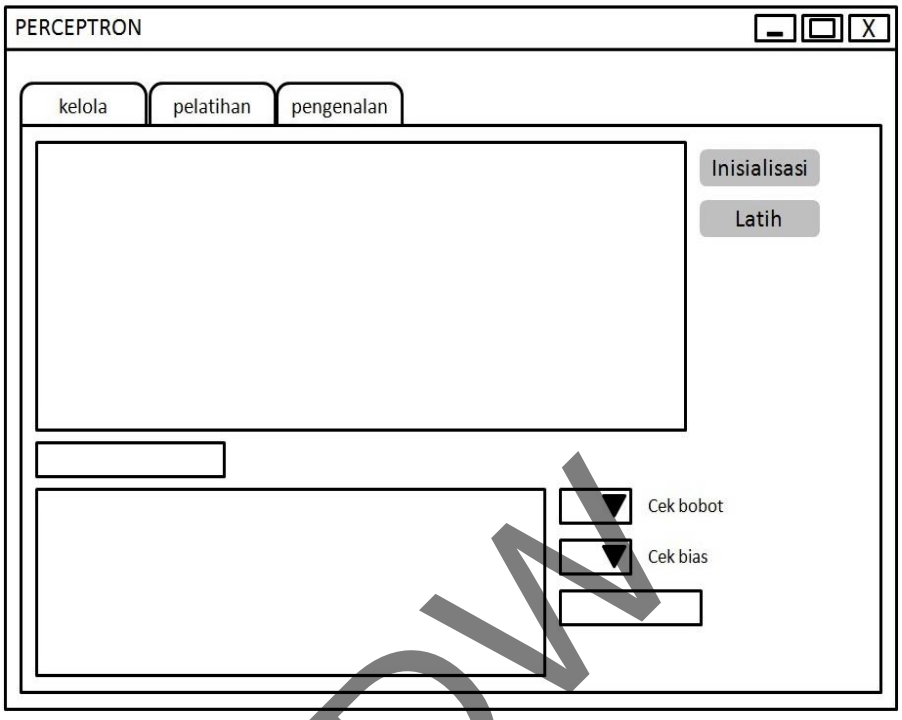
Form kelola digunakan untuk meng*upload* huruf ke *database*. *User* harus mempersiapkan gambar huruf – huruf lengkap dari A – Z dan diletakkan pada satu folder dan masing – masing gambar harus diberi nama sesuai dengan huruf yang ada pada gambar. Kemudian *user* harus meng*input*kan jenis huruf yang di*upload* pada kolom "Nama *Font*", setelah mengisi kolom "Nama *Font*", *user* dapat meng*upload* folder berisi huruf – huruf untuk disimpan ke *database* dengan menekan tombol "Folder". Jika *user* belum mengisi kolom "Nama *Font*" maka tombol "Folder" tidak dapat digunakan. Pada saat meng*upload* huruf akan di-*thinning* terlebih dahulu lalu disimpan ke *database*. Jika *database* masih kosong, *user* harus melakukan tahap *upload* gambar ini agar dapat melakukan proses pelatihan dan pengenalan. Rancangan form kelola data *font* dapat dilihat pada Gambar 3.5.



Gambar 3.5. Form Kelola Data Font

3.4.2. Form Pelatihan

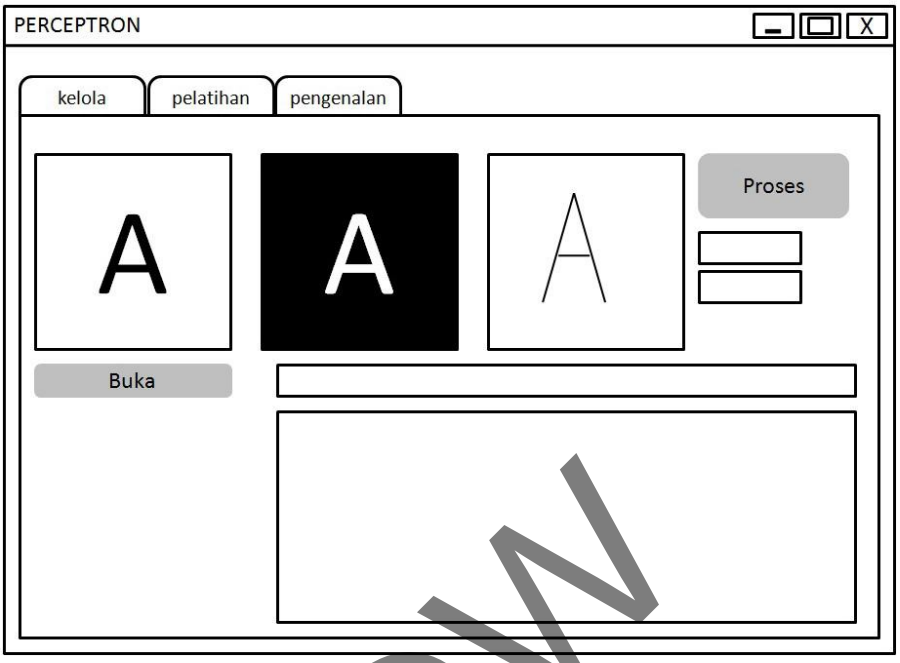
Form pelatihan digunakan untuk melatih huruf – huruf yang ada pada *database*. Sebelum dilatih *user* harus melakukan inisialisasi dengan menekan tombol ”Inisialisasi”. Hal ini dilakukan untuk inisialisasi w, α, θ, bias, dan mengambil nilai biner dari gambar pada *database*. Setelah melakukan inisialisasi, *user* dapat melakukan pelatihan dengan menekan tombol “Latih”. Tahap pelatihan ini harus dilakukan dulu sebelum melakukan tahap pengenalan. Hal ini dilakukan agar didapatkan bobot (*w*) baru yang akan digunakan pada saat melakukan pengenalan. Form pelatihan dapat dilihat pada Gambar 3.6.



Gambar 3.6. Form Pelatihan

3.4.3. Form Pengenalan

Form pengenalan digunakan untuk melakukan pengenalan. *User* meng*upload* gambar yang akan dicoba untuk dikenali dengan menekan tombol "Buka". Gambar harus bertipe BMP. Setelah memilih gambar, *user* menekan tombol "Proses" untuk melakukan proses *thinning* dan kemudian dilanjutkan pengenalan dengan *perceptron*. Gambar asli akan ditampilkan pada kotak "Gambar Asli", gambar hasil *tresholding* akan ditampilkan pada kotak "Gambar Black White", gambar hasil *thinning* akan ditampilkan pada kotak "Gambar *Thinning*". Hasil perhitungan akan ditampilkan pada kotak "Hasil Hitung". Sedangkan hasil huruf akan ditampilkan dengan "MessageBox". Form pengenalan dapat dilihat pada Gambar 3.7.



Gambar 3.7. Form Pengenalan

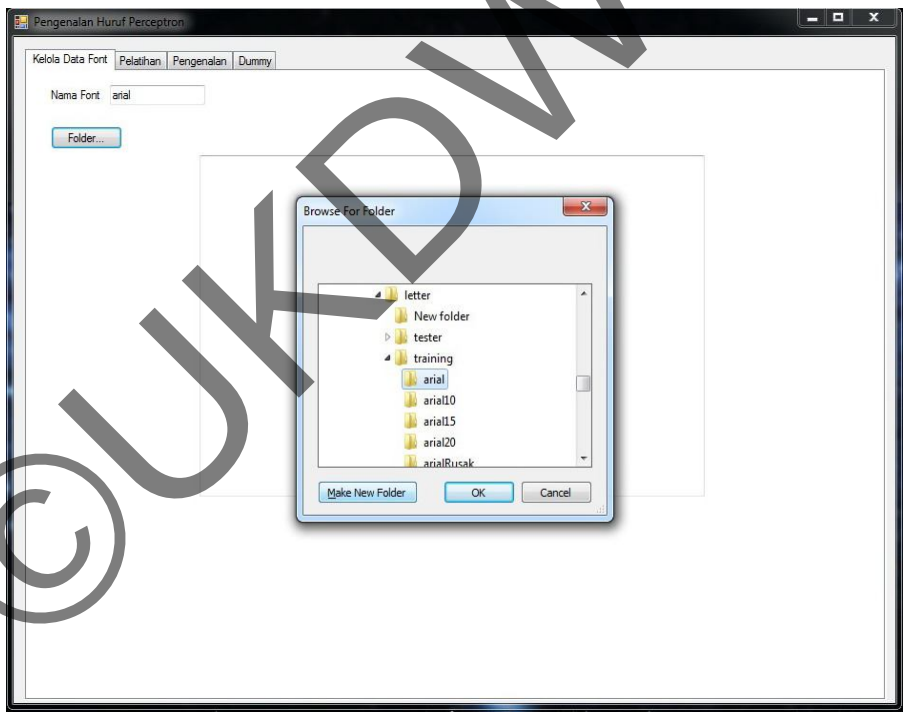
BAB 4

IMPLEMENTASI DAN ANALISIS SISTEM

4.1. Implementasi Sistem

Pada sub bab ini akan dijelaskan tentang antarmuka program dan cara penggunaannya mulai dari proses *upload font* ke *database*, inisialisasi, pelatihan, cek bobot dan bias, hingga proses pengenalan.

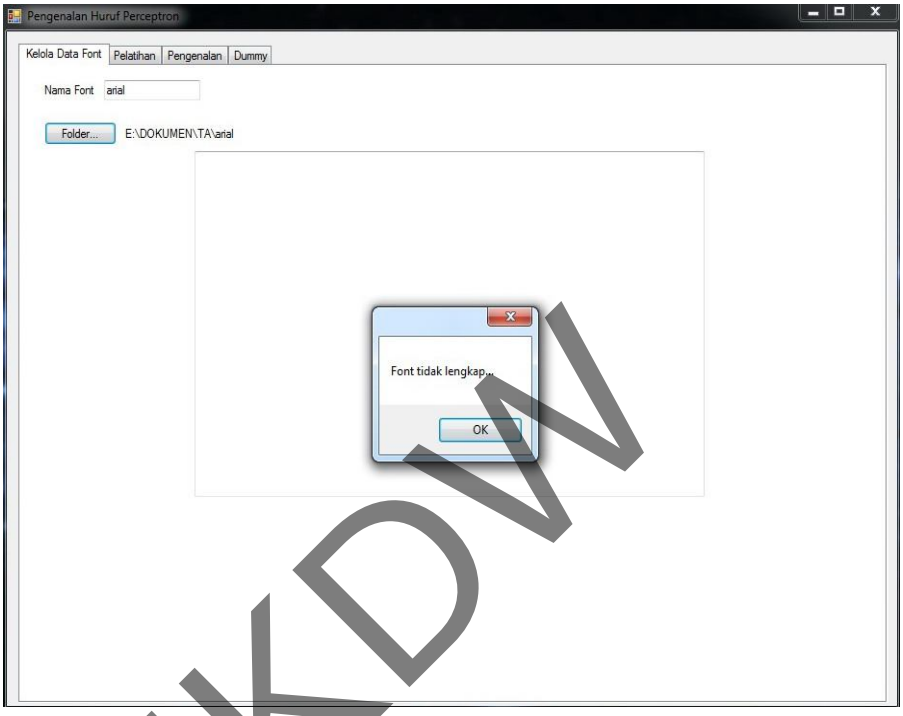
4.1.1. Implementasi Tab-Kelola Data *Font*



Gambar 4.1. Implementasi Form Tab Kelola Data Font – user mengisi nama font

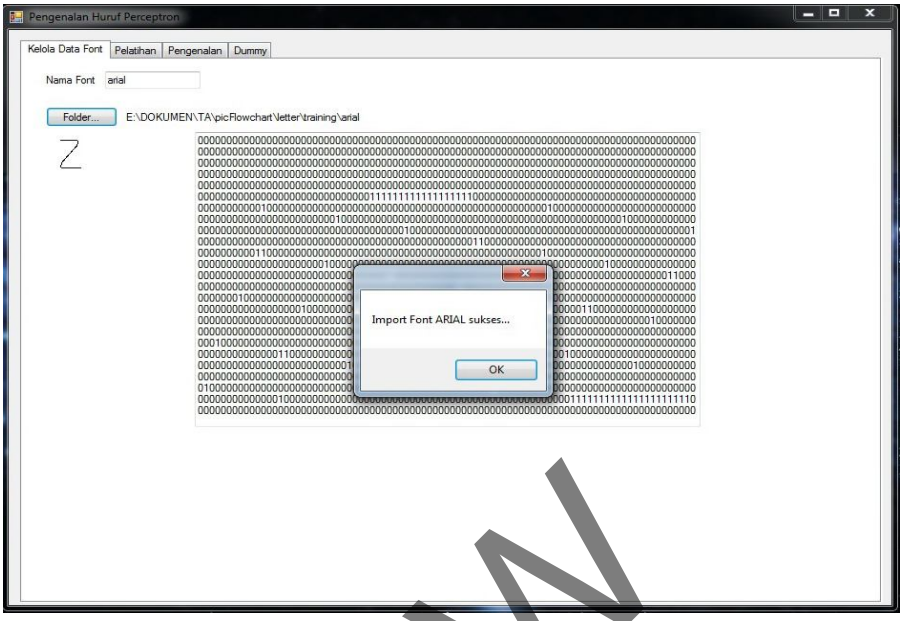
Tab `Kelola Data Font` akan tampil pertama kali saat program dijalankan. Tab ini digunakan *user* saat ingin meng*upload font* ke *database*. *User* mengisi nama *font* lalu tekan enter maka akan muncul tampilan seperti

pada Gambar 4.1. Setelah *user* menentukan jenis *font* yang akan di*upload*, *user* meng-klik tombol OK. Jika proses *upload font* tidak sukses maka akan muncul tampilan seperti Gambar 4.2.



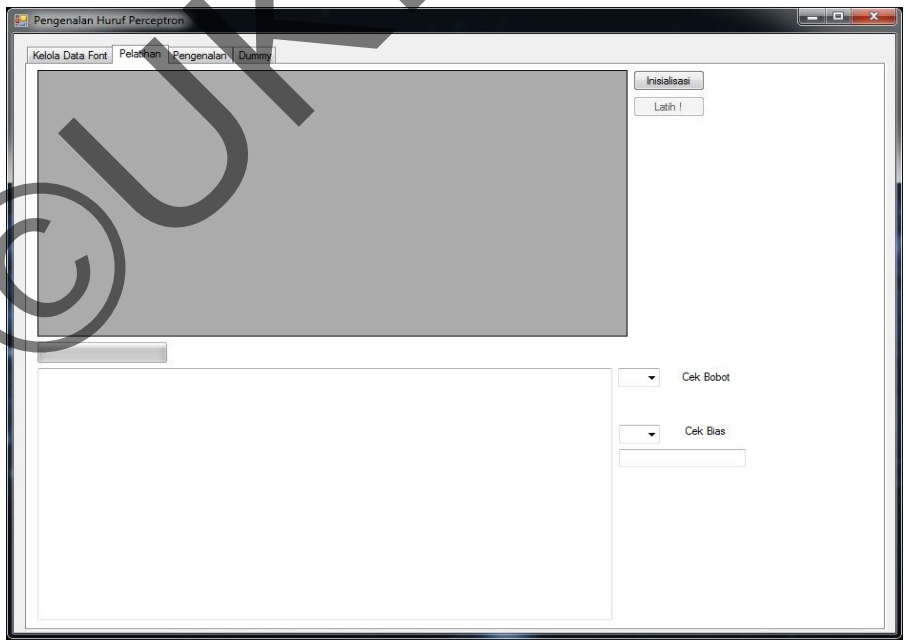
Gambar 4.2. Implementasi Form Tab Kelola Data Font – upload gagal

Kegagalan proses *upload* bisa disebabkan isi dalam folder *font* yang akan di*upload* tidak lengkap yaitu kurang dari 26 atau pemberian nama file yang salah. Jika proses *upload* sukses dilakukan maka *font* yang di*upload* akan otomatis di*thinning* dan akan ditampilkan gambar *font* dan data binernya, sehingga akan muncul tampilan seperti Gambar 4.3 dibawah ini yang menunjukkan proses *upload* telah sukses dilakukan.



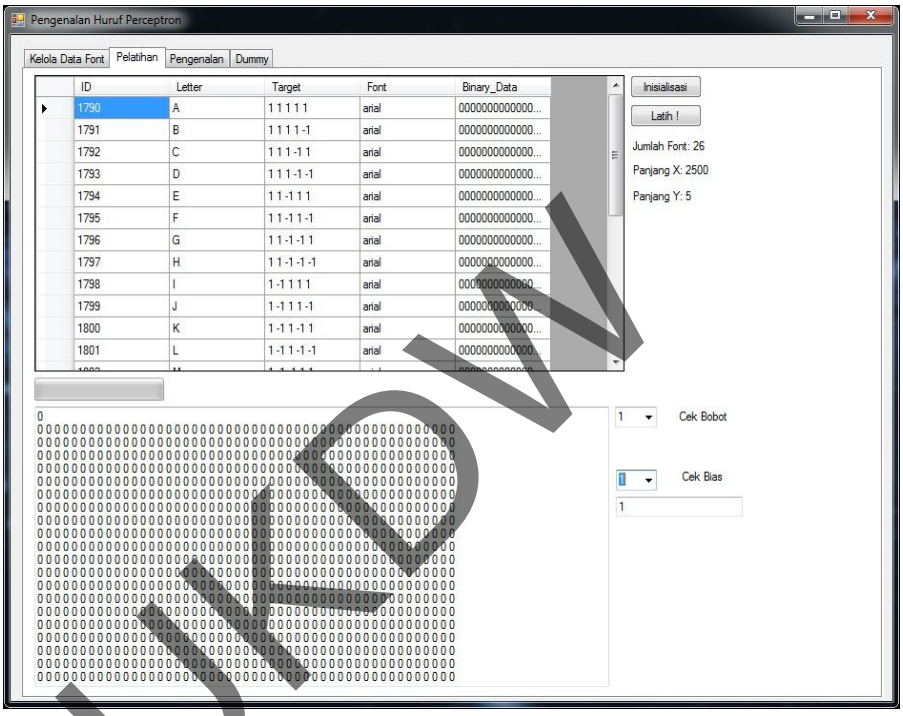
Gambar 4.3. Implementasi Form Tab Kelola Data Font – upload sukses

4.1.2. Implementasi Tab - Pelatihan



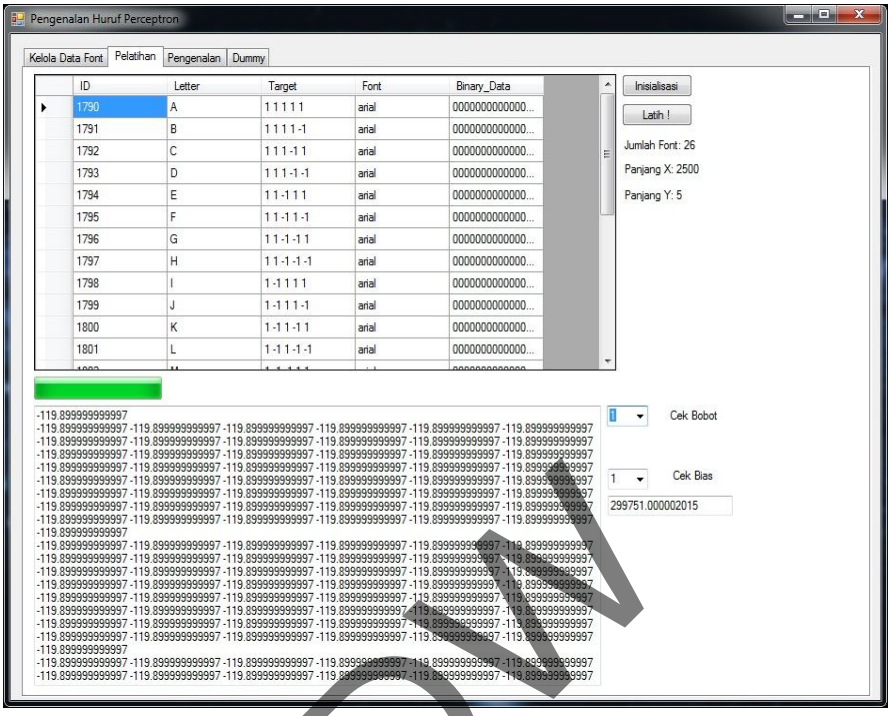
Gambar 4.4. Implementasi Form Tab Pelatihan - awal

Gambar 4.4. menunjukkan tampilan Tab `Pelatihan` saat pertama kali dipilih. Dapat dilihat tombol `Latih` masih tidak bisa digunakan. Untuk memulai pelatihan *user* harus melakukan inisialisasi dengan menekan tombol `Inisialisasi`. Setelah tombol `Inisialisasi` ditekan maka akan tampil seperti Gambar 4.5. dibawah ini.



Gambar 4.5. Form Tab Pelatihan – Inisialisasi

Dari Gambar 4.5. dapat dilihat *font* yang sudah ada di *database* hasil dari proses *upload* yang dilakukan oleh *user* dan tombol `Latih` sudah dapat digunakan. Kolom bobot semua masih berisi nilai 0 dan kolom bias masih berisi nilai 1. Hal ini dikarenakan belum dilakukan pelatihan untuk menentukan bobot dan bias yang baru. Bias mempunyai pengaruh dalam meningkatkan atau menurunkan input dari fungsi aktivasi. Untuk melakukan pelatihan *user* harus menekan tombol `Latih`. Saat *user* menekan tombol `Latih`, proses pelatihan akan dilakukan pada semua data *font* yang ada di *database*. Hasil setelah tekan tombol Pelatihan akan muncul tampilan seperti Gambar 4.6.

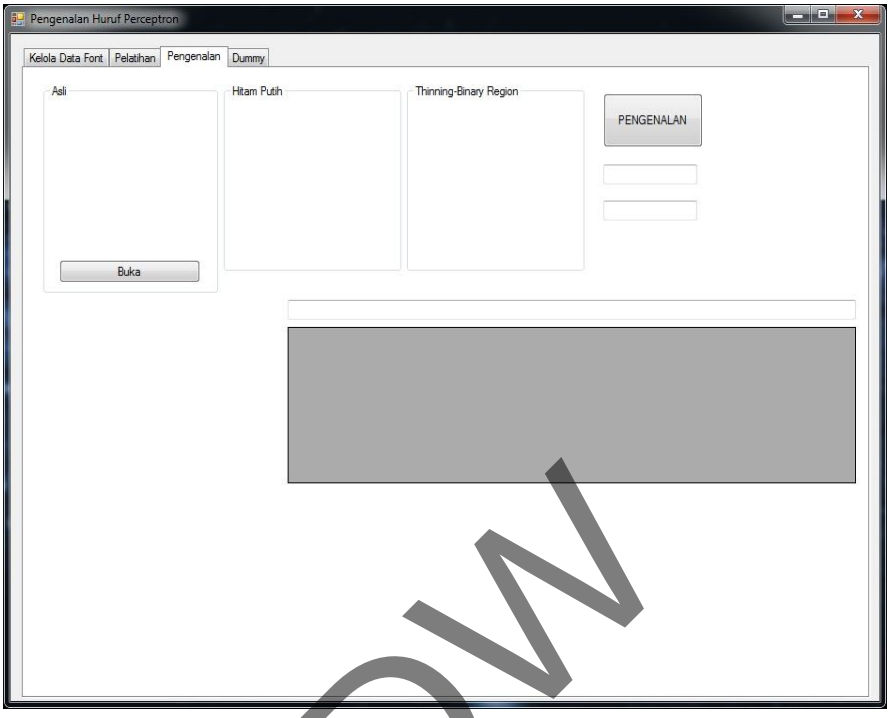


Gambar 4.6. Form Tab Pelatihan – Latih

Dari Gambar 4.6 dapat dilihat nilai bobot dan bias telah berubah. Nilai bobot dan bias inilah yang akan digunakan untuk proses pengenalan.

4.1.3. Implementasi Tab – Pengenalan

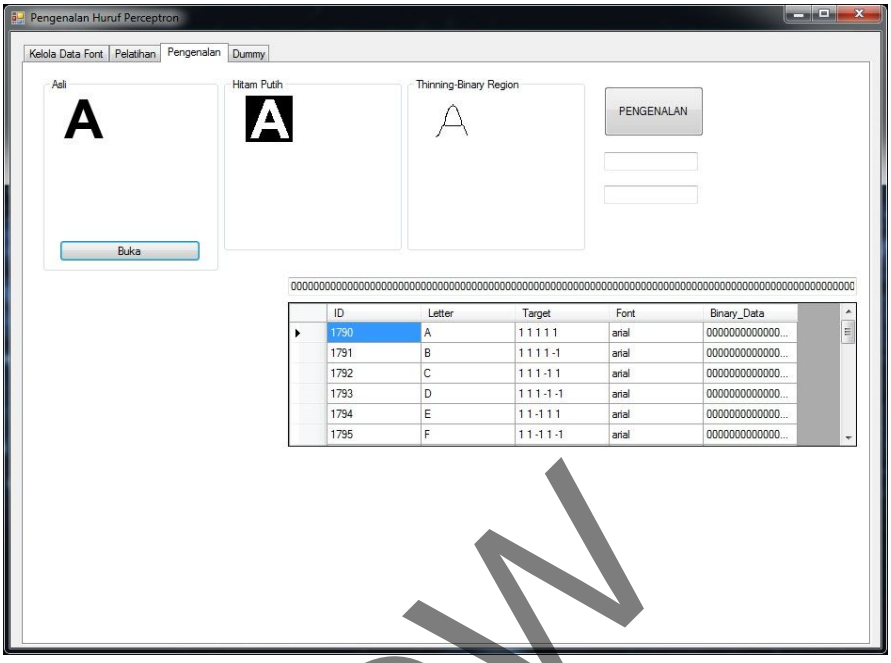
Tab Pengenalan digunakan pada saat *user* akan melakukan proses pengenalan. Proses pengenalan ini dapat dilakukan setelah sebelumnya *user* melakukan pelatihan. Jika tidak melalui proses pelatihan maka proses pengenalan akan gagal dilakukan. Tampilan awal tab Pengenalan dapat dilihat pada Gambar 4.7. dibawah ini.



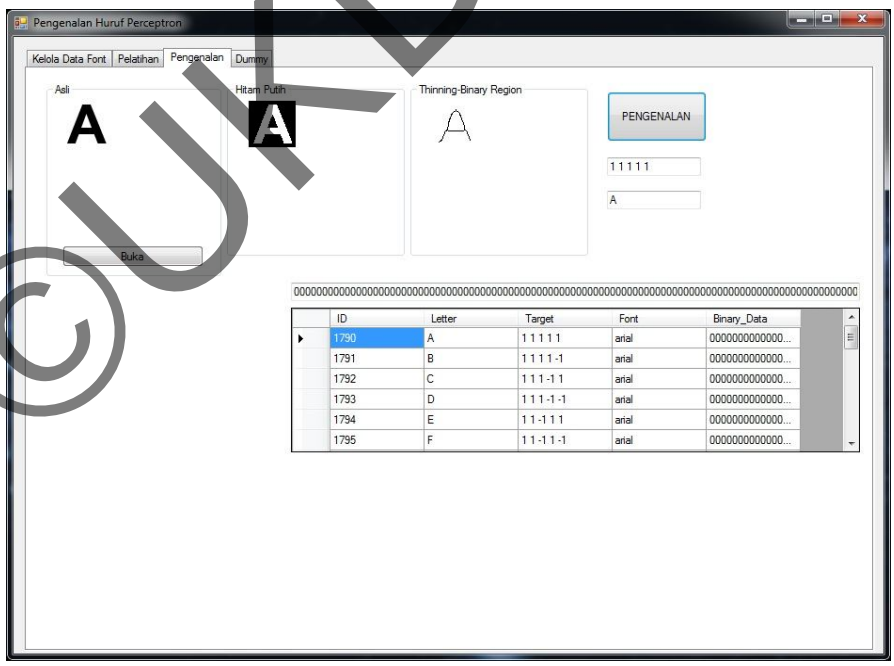
Gambar 4.7. Form Tab Pengenalan – awal

Saat *user* akan melakukan proses pengenalan, pertama kali *user* harus menekan tombol Buka untuk mencari file gambar yang akan dicoba untuk dikenali. Setelah memilih gambar yang diinginkan maka akan ditampilkan 3 gambar yaitu gambar asli, gambar hasil tresholding, dan gambar hasil thinning. Akan ditampilkan juga data biner dari gambar huruf yang dipilih *user*. Data biner ini yang digunakan untuk proses pengenalan.

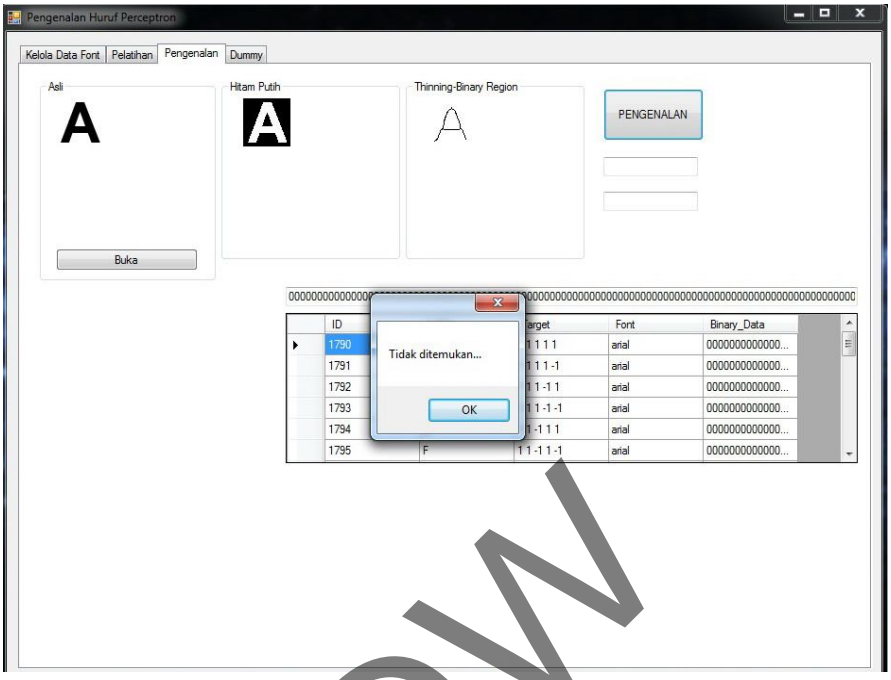
Setelah gambar yang dipilih muncul maka *user* harus menekan tombol Pengenalan untuk melakukan proses pengenalan. Jika gambar dikenali maka akan keluar huruf dari hasil perhitungan. Jika tidak dikenali maka akan muncul pesan “tidak dikenali”. Untuk masing- masing tampilan dapat dilihat seperti Gambar 4.8., Gambar 4.9., Gambar 4.10.



Gambar 4.8. Form Tab Pengenalan – setelah memilih gambar



Gambar 4.9. Form Tab Pengenalan – gambar dikenali



Gambar 4.10. Form Tab Pengenalan – gambar tidak dikenali

4.2. Pengujian dan Analisis Sistem

Pada sub bab ini akan dijelaskan tentang analisis dan pengujian sistem yang telah dibuat untuk mengenali pola – pola huruf yang di*input*kan *user*. Pada pengujian ini dalam *database* sistem telah terdapat huruf A – Z dengan jenis *font Arial Bold* yang akan digunakan untuk pelatihan. Pengujian akan dilakukan sebanyak 4 kali dengan rincian sebagai berikut.

1. *Font Arial Bold* atau seperti pada *database*.
2. *Font Arial Bold* dengan variasi *Noise*.
3. *Font Arial Bold Italic*.
4. *Font Arial Regular*.

Contoh *font Arial Bold* dapat dilihat pada lampiran-1. Hasil pengujian pertama yang diterapkan pada *font Arial Bold* dapat dilihat pada Tabel 4.1.

Tabel 4.1. Hasil pengujian pada *font Arial Bold* di database

Arial Bold		
Huruf	**Dikenali sebagai**	**Status**
A	A	✓
B	B	✓
C	C	✓
D	D	✓
E	F	×
F	F	✓
G	G	✓
H	H	✓
I	B	×
J	H	×
K	C	×
L	H	×
M	E	×
N	D	×
O	G	×
P	H	×
Q	A	×
R	B	×
S	C	×
T	F	×
U	H	×
V	F	×
W	G	×
X	B	×
Y	B	×
Z	H	×

Dari data Tabel 4.1 terlihat sistem tidak dapat mengenali semua huruf, walaupun *font Arial Bold* adalah *font* yang digunakan untuk proses pelatihan. Tingkat keberhasilan sistem hanya 26,9%. Sistem hanya dapat mengenali huruf A,B,C,D,F,G,H. Sedangkan huruf lainnya hasilnya tidak sesuai dengan yang diharapkan *user* walaupun *font* yang diuji sama dengan *font* yang dilatih.

4.2.1. Pengujian Pada *FontArial* Dengan Variasi *Noise*

Pengujian kedua diterapkan pada *font Arial Bold* dengan variasi *Noise*. Pemberian *Noise* dilakukan menggunakan *software Adobe Photoshop* dengan metode Gaussian dan tingkat *Noise* 21%. Contoh *font Arial Bold* dengan variasi *Noise* dapat dilihat pada lampiran-2. Hasil pengujian kedua yang diterapkan dapat dilihat pada Tabel 4.2.

Tabel 4.2. Hasil pengujian pada *font Arial Bold Noise*

Arial Bold Noise		
Huruf	**Dikenali sebagai**	**Status**
A	A	✓
B	B	✓
C	D	×
D	D	✓
E	F	×
F	F	✓
G	G	✓
H	H	✓
I	B	×
J	H	×
K	D	×
L	H	×
M	E	×
N	H	×
O	G	×
P	H	×
Q	C	×
R	B	×
S	D	×
T	F	×
U	H	×
V	F	×
W	G	×
X	B	×
Y	B	×
Z	H	×

Dari data tabel 4.2 diatas terlihat sistem hanya dapat mengenali huruf A,B,D,F,G,H. Hasil ini lebih sedikit dari pengujian pertama karena pada pengujian kedua ini huruf C dikenali menjadi huruf D. Sedangkan huruf

lainnya hasilnya tidak sesuai dengan yang diharapkan *user*. Pada pengujian kedua ini tingkat keberhasilan turun menjadi 23,1%. *Noise* tidak terlalu berpengaruh pada sistem selama *noise* tidak terlalu banyak, karena jika terlalu banyak *noise* maka akan mengubah bentuk kerangka huruf.

4.2.2. Pengujian Pada *Font Arial Bold Italic*

Pengujian ketiga diterapkan pada *font Arial Bold Italic*. Contoh *font Arial Bold Italic* dapat dilihat pada lampiran-3. Hasil pengujian dapat dilihat pada tabel 4.3 dibawah ini.

Tabel 4.3. Hasil pengujian *font Arial Bold Italic*

Arial Bold Italic		
Huruf	**Dikenali sebagai**	**Status**
A	H	×
B	H	×
C	H	×
D	D	✓
E	F	×
F	D	×
G	H	×
H	H	✓
I	F	×
J	H	×
K	B	×
L	H	×
M	G	×
N	H	×
O	G	×
P	H	×
Q	G	×
R	F	×
S	H	×
T	F	×
U	H	×
V	F	×
W	H	×
X	F	×
Y	F	×
Z	F	×

Dari data Tabel 4.3 dapat terlihat sistem hanya dapat mengenali huruf D dan H, sedangkan pengujian huruf lainnya hasilnya tidak sesuai dengan yang diharapkan *user*. Tingkat keberhasilan sistem hanya 7,7%. Mayoritas huruf yang diuji dikenali sistem sebagai huruf H dan F.

4.2.3. Pengujian Pada *Font Arial Regular*

Pengujian keempat diterapkan pada *font Arial Regular*. Contoh *font Arial* Regular dapat dilihat pada lampiran-4. Hasil dari pengujian dapat dilihat pada Tabel 4.4.

Tabel 4.4.Hasil pengujian *font Arial Regular*

Arial Regular		
Huruf	**Dikenali sebagai**	**Status**
A	F	×
B	D	×
C	G	×
D	G	×
E	D	×
F	F	✓
G	F	×
H	D	×
I	H	×
J	H	×
K	D	×
L	D	×
M	H	×
N	D	×
O	H	×
P	F	×
Q	G	×
R	F	×
S	H	×
T	F	×
U	D	×
V	H	×
W	G	×
X	B	×
Y	F	×
Z	H	×

Dari data Tabel 4.4 dapat terlihat sistem hanya dapat mengenali huruf F, sedangkan pengujian pada huruf lainnya hasilnya tidak sesuai yang diharapkan *user*. Tingkat keberhasilan sistem pun hanya 3,8% atau paling kecil dari semua pengujian yang telah dilakukan. Mayoritas huruf yang diuji dikenali sistem sebagai huruf D, F, dan H.

4.2.4. Pengaruh Target, Jumlah Huruf, dan Variasi Huruf Pada *Perceptron*

Pemberian nilai target juga mempengaruhi proses pelatihan *perceptron*. Hal ini dilakukan penulis dengan membuat dua daftar target. Rincian target dapat dilihat pada Tabel 4.5.

Tabel 4.5.Daftar pengujian variasi target

Huruf	Target 1	Target 2
A	1 1 1 1 1	1 1 1 1 1
B	1 1 1 1 -1	1 1 1 1 -1
C	1 1 1 -1 1	1 1 1 -1 1
D	1 1 1 -1 -1	1 1 1 -1 -1
E	1 1 -1 1 1	1 1 -1 1 1
F	1 1 -1 1 -1	1 1 -1 1 -1
G	1 1 -1 -1 1	1 1 -1 -1 1
H	1 1 -1 -1 -1	1 1 -1 -1 -1
I	1 -1 1 1 1	1 -1 1 1 1
J	-1 1 -1 -1 -1	1 -1 1 1 -1
K	1 -1 1 -1 1	1 -1 1 -1 1
L	1 -1 1 -1 -1	1 -1 1 -1 -1
M	1 -1 -1 1 1	1 -1 -1 1 1
N	-1 1 1 -1 -1	1 -1 -1 1 -1
O	1 -1 -1 -1 1	1 -1 -1 -1 1
P	1 -1 -1 -1 -1	1 -1 -1 -1 -1
Q	-1 1 1 1 1	-1 1 1 1 1
R	-1 1 1 1 -1	-1 1 1 1 -1
S	-1 1 1 -1 1	-1 1 1 -1 1
T	1 -1 -1 1 -1	-1 1 1 -1 -1
U	-1 1 -1 1 1	-1 1 -1 1 1
V	-1 1 -1 1 -1	-1 1 -1 1 -1
W	-1 1 -1 -1 1	-1 1 -1 -1 1

Tabel 4.5.Daftar pengujian variasi target (lanjutan)

Huruf	Target 1	Target 2
X	1 -1 1 1 -1	-1 1 -1 -1 -1
Y	-1 -1 1 1 1	-1 -1 1 1 1
Z	-1 -1 -1 -1 -1	-1 -1 1 1 -1

Kolom Target 1 pada Tabel 4.5 merupakan target awal yang digunakan untuk melakukan pengujian sebelumnya. Kolom Target 2 merupakan target variasi yang akan diuji. Data target huruf pada *database* akan diubah sesuai kolom Target 2 lalu dilakukan pengujian. Hasil pengujian menggunakan Target 2 dapat dilihat pada Tabel 4.6.

Tabel 4.6. Hasil pengujian menggunakan Target 2

Arial Bold		
Huruf	**Dikenali sebagai**	**Status**
A	A	✓
B	B	✓
C	A	×
D	B	×
E	B	×
F	B	×
G	A	×
H	B	×
I	B	×
J	B	×
K	A	×
L	B	×
M	A	×
N	B	×
O	A	×
P	B	×
Q	A	×
R	B	×
S	A	×
T	B	×
U	B	×
V	B	×
W	A	×
X	B	×
Y	B	×

Tabel 4.6. Hasil pengujian menggunakan Target 2 (lanjutan)

Huruf	Dikenali sebagai	Status
Z	B	×

Jenis *font* pada *database* tetap sama yaitu *Arial Bold*. Pengujian diterapkan menggunakan jenis *font Arial Bold* seperti yang telah dilatih. Dari data Tabel 4.6 dapat terlihat jika Tabel 4.1 memiliki hasil yang lebih baik dari Tabel 4.6, sistem hanya dapat mengenali huruf A dan B, sedangkan huruf lainnya tidak sesuai dengan yang diharapkan *user* walaupun jenis *font* yang diuji sama dengan yang dilatih.

Jumlah huruf pada juga berpengaruh pada *perceptron*. Semakin banyak huruf yang dilatih maka sistem akan lebih sulit mengenali huruf – huruf yang diuji. Hal ini disebabkan karena akan makin banyak huruf maka akan semakin sulit untuk mendapat bobot yang tepat. Sedangkan semakin sedikit huruf yang dilatih maka sistem akan lebih dapat untuk mengenali huruf – huruf yang diuji. Hasil pengujian dapat dilihat pada Tabel 4.7.

Tabel 4.7. Hasil pengujian dengan variasi jumlah huruf

Jenis *Font* / **Jumlah Huruf**	***Arial Bold***	***Arial Bold Noise***	***Arial Bold Italic***	***Arial Regular***
4	64,3%	64,3%	39,3%	35,7%
8	50%	50%	15,6%	15,6%
12	16,7%	16,7%	2,8%	2,8%
26	26,9%	23,1%	7,7%	3,8%

Variasi huruf juga berpengaruh pada *perceptron*. Jika terlalu banyak variasi huruf yang diuji maka sistem kurang dapat mengenali huruf yang diuji. Hal ini dikarenakan bentuk dari kerangka akan menjadi berbeda jauh dari yang huruf yang dilatih. Dari Tabel 4.7 hasil pengujian paling baik selain pada jenis *font* yang ada pada database adalah jenis *font* dengan variasi *Noise*. Hasil pengujian menunjukkan bahwa *Noise* tidak terlalu mempengaruhi sistem. Hasil pengujian yang memiliki prosentase paling kecil adalah pada *font Arial Regular*. Pengujian pada *font Arial Regular* hasilnya tidak pernah melebihi dari prosentase pengujian pada *font Arial Bold Italic*.

LAMPIRAN

Lampiran 1

List Program

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
using System.IO;

namespace PengenalanHurufPerceptron
{
    public partial class frmUtama : Form
    {
        OleDbConnection thisConn = Konfigurasi.lettersConnection;
        OleDbDataAdapter thisAdapter;
        OleDbCommand thisCommand;
        OleDbDataReader thisReader;
        DataSet thisDataset = new DataSet();
        DataSet thisDatasetPengenalan = new DataSet();
        string namaFont = "";
        int jumlahFont = 0, jumlahY = 0, jumlahX = 0, counter=0;
        string y_training = "";
        string[] target2 = new string[52];
        string[] target = new string[26];
        double[,] x = new double[52, 2500];
        double[,] t = new double[52, 5];
        double[,] y = new double[52, 5];
        double[] bias = { 1, 1, 1, 1, 1 };//new double[5];
        double[,] w_lama = new double[5, 2500];
        double[,] w_baru = new double[5, 2500];
        double[] x_pengenalan = new double[2500];
        bool[,] is_sama = new bool[260, 7];
        double alfa = 0.1;
        double tetha = 0;
        double ySumDummy = 0;
        double[] data = new double[12500];
        double[] frekuensi = new double[12500];
        double modus;

        int treshold = 127;
        int[,] arrayTreshold = new int[50, 50];
        string delPixel = "";
        List<string> list_pixel_del = new List<string>();
        int[,] arrayTreshold2;
        string hitungBobot = "";
        List<string> list_hitung_bobot = new List<string>();
        List<int> list_jml_bobot = new List<int>();


        public frmUtama()
```

```
        {
            InitializeComponent();
        }

        public Image Treshold(Image im, int treshold)
        {
            Bitmap arrayTresholdInput = (Bitmap)im;
            Bitmap arrayTresholdOutput = new Bitmap(50, 50);
            for (int i = 0; i < arrayTresholdInput.Height; i++)
            {
                for (int j = 0; j < arrayTresholdInput.Width; j++)
                {
                    if (((arrayTresholdInput.GetPixel(j, i).B +
arrayTresholdInput.GetPixel(j, i).R + arrayTresholdInput.GetPixel(j, i).G) / 3) >
treshold)
                    {
                        arrayTresholdOutput.SetPixel(j, i, Color.Black);
                        arrayTreshold[j, i] = 0;
                    }
                    else
                    {
                        arrayTresholdOutput.SetPixel(j, i, Color.White);
                        arrayTreshold[j, i] = 1;
                    }
                }
            }
            //textBox2.Text = arrayTreshold;
            return (Image)arrayTresholdOutput;
        }

        private int[,] thinning_gambar(int width, int height)
        {
            int status1 = 1;
            int status2 = 1;
            int iterasi = 0;
            do
            {
                for (int i = 1; i < height - 1; i++)
                {
                    for (int j = 1; j < width - 1; j++)
                    {
                        if (cari_pixel1(j, i) == 1)
                        {
                            delPixel = Convert.ToString(j) + "," +
Convert.ToString(i);
                            list_pixel_del.Add(delPixel);
                        }
                    }
                }
                if (delete_pixel() == 0) { status1 = 0; }

                for (int i = 1; i < height - 1; i++)
                {
                    for (int j = 1; j < width - 1; j++)
                    {
                        if (cari_pixel2(j, i) == 1)
                        {
                            delPixel = Convert.ToString(j) + "," +
Convert.ToString(i);
                            list_pixel_del.Add(delPixel);
                        }
                    }
```

```
        }
        if (delete_pixel() == 0) { status2 = 0; }
    } while (status1 != 0 || status2 != 0);  //1 || 0
    return arrayTreshold;
}

private int cari_pixel1(int cari_j, int cari_i)
{
    if (arrayTreshold[cari_j, cari_i] == 0)
        return 0;
    else
    {
        if (cek_pixel_1(cari_j, cari_i) == 1)
        {
            if (cek_pixel_2(cari_j, cari_i) == 1)
            {
                return 1;
            }
            else { return 0; }
        }
        else
        {
            return 0;
        }
    }
    //return 0;
}

private int cari_pixel2(int cari_j, int cari_i)
{
    if (arrayTreshold[cari_j, cari_i] == 0)
        return 0;
    else
    {
        if (cek_pixel_1(cari_j, cari_i) == 1)
        {
            if (cek_pixel_3(cari_j, cari_i) == 1)
            {
                return 1;
            }
            else { return 0; }
        }
        else
        {
            return 0;
        }
    }
    //return 0;
}

/*
0 P2 -- del_j , del_i - 1
1 P3 -- del_j + 1 , del_i - 1
2 P4 -- del_j + 1 , del_i
3 P5 -- del_j + 1 , del_i + 1
4 P6 -- del_j , del_i + 1
5 P7 -- del_j - 1 , del_i + 1
6 P8 -- del_j - 1 , del_i
7 P9 -- del_j - 1 , del_i - 1
*/

private int cek_pixel_1(int del_j, int del_i)
```

```
        {
            int countNp1 = 0;
            int countSp1 = 0;
            int[] Np1 = new int[] {
                                    arrayTreshold[del_j, del_i - 1],
                                    arrayTreshold[del_j + 1, del_i - 1],
                                    arrayTreshold[del_j + 1, del_i],
                                    arrayTreshold[del_j + 1, del_i + 1],
                                    arrayTreshold[del_j, del_i + 1],
                                    arrayTreshold[del_j - 1, del_i + 1],
                                    arrayTreshold[del_j - 1, del_i],
                                    arrayTreshold[del_j - 1, del_i - 1],
                                    arrayTreshold[del_j, del_i - 1]};
            for (int x = 0; x <= 7; x++)
            {
                if (Np1[x] == 1) { countNp1++; }
            }
            if (countNp1 < 2 || countNp1 > 6)
            {
                return 0;
            }

            for (int x = 0; x <= 7; x++)
            {
                if (Np1[x] == 0 && Np1[x + 1] == 1) { countSp1++; }
            }
            if (countSp1 != 1)
            {
                return 0;
            }
            else { return 1; }

            //return 0;
        }

        private int cek_pixel_2(int del_j, int del_i)
        {
            int[] Np1 = new int[] {
                                    arrayTreshold[del_j, del_i - 1],
                                    arrayTreshold[del_j + 1, del_i - 1],
                                    arrayTreshold[del_j + 1, del_i],
                                    arrayTreshold[del_j + 1, del_i + 1],
                                    arrayTreshold[del_j, del_i + 1],
                                    arrayTreshold[del_j - 1, del_i + 1],
                                    arrayTreshold[del_j - 1, del_i],
                                    arrayTreshold[del_j - 1, del_i - 1],
                                    arrayTreshold[del_j, del_i - 1] };
            if ((Np1[0] == 0 || Np1[2] == 0 || Np1[4] == 0) && (Np1[2] == 0 || Np1[4]
== 0 || Np1[6] == 0))
            {
                return 1;
            }
            else { return 0; }
            //return 0;
        }

        private int cek_pixel_3(int del_j, int del_i)
        {
            int[] Np1 = new int[] {
                                     arrayTreshold[del_j, del_i - 1],
                                     arrayTreshold[del_j + 1, del_i - 1],
                                     arrayTreshold[del_j + 1, del_i],
```

```
                                    arrayTreshold[del_j + 1, del_i + 1],
                                    arrayTreshold[del_j, del_i + 1],
                                    arrayTreshold[del_j - 1, del_i + 1],
                                    arrayTreshold[del_j - 1, del_i],
                                    arrayTreshold[del_j - 1, del_i - 1],
                                   arrayTreshold[del_j, del_i - 1] };
            if ((Np1[0] == 0 || Np1[2] == 0 || Np1[6] == 0) && (Np1[0] == 0 || Np1[4]
== 0 || Np1[6] == 0))
            {
                return 1;
            }
            else { return 0; }
            //return 0;
        }

        private int delete_pixel()
        {
            if (list_pixel_del.Count == 0)
            {
                return 0;
            }
            else
            {
                int i, j;
                char[] delimeter2 = { ',' };
                foreach (string s in list_pixel_del)
                {
                    string[] pixeldel = s.Split(delimeter2);
                    j = Convert.ToInt32(pixeldel[0]);
                    i = Convert.ToInt32(pixeldel[1]);
                    arrayTreshold[j, i] = 0;
                }
                list_pixel_del.Clear();
                return 1;
            }
        }

        Bitmap ToBitmap(int[,] imgBinary)
        {
            Bitmap imgThinning = new Bitmap(50, 50);
            for (int i = 0; i < imgThinning.Height; i++)
            {
                for (int j = 0; j < imgThinning.Width; j++)
                {
                    if (imgBinary[j, i] == 1)
                    {
                        imgThinning.SetPixel(j, i, Color.Black);
                    }
                }
            }

            return imgThinning;
        }

        private void OutputData(TextBox txtOutput, int[,] imgBinaries)
        {
            StringBuilder sb = new StringBuilder();

            for (int x = 0; x < imgBinaries.GetLength(0); x++)
            {
                for (int y = 0; y < imgBinaries.GetLength(1); y++)
                {
```

```
            if (imgBinaries[y, x] == 1)
            {
                sb.Append("1");
            }
            else
            {
                sb.Append("0");
            }
        }
    }
    txtOutput.Text = sb.ToString();
}

private void btnBukaGambar_Click(object sender, EventArgs e)
{
    try
    {
        thisDataset.Reset();
        string query;
        thisConn.Open();
        query = "select * from mst_letter order by font,letter";
        thisAdapter = new OleDbDataAdapter(query, thisConn);
        thisAdapter.Fill(thisDataset);
        thisConn.Close();
        dgvTampilPengenalan.DataSource = null;
        dgvTampilPengenalan.Refresh();
        dgvTampilPengenalan.DataSource = thisDataset.Tables[0];

        ofdLoadImage.FileName = "";
        ofdLoadImage.ShowDialog();
        if (ofdLoadImage.FileName == "")
        {
            return;
        }
        pbBefore.Image = Image.FromFile(ofdLoadImage.FileName);

        //START
        pbBW.Image = Treshold(pbBefore.Image, treshold);
        Bitmap bmp = new Bitmap(pbBW.Image);
        arrayTreshold2 = thinning_gambar(bmp.Width, bmp.Height);
        bmp = ToBitmap(arrayTreshold2);
        pbThinning.Image = bmp;
        OutputData(txtDummy, arrayTreshold2);
        for (int i = 0; i < txtDummy.Text.Length; i++)
        {
            x_pengenalan[i] = Convert.ToDouble(txtDummy.Text.Substring(i,1));
        }
        //END
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }

}

private void Form1_Load(object sender, EventArgs e)
{

    //bipolar

    target[0] = "1 1 1 1 1";
```

```
    target[1] = "1 1 1 1 -1";
    target[2] = "1 1 1 -1 1";
    target[3] = "1 1 1 -1 -1";
    target[4] = "1 1 -1 1 1";
    target[5] = "1 1 -1 1 -1";
    target[6] = "1 1 -1 -1 1";
    target[7] = "1 1 -1 -1 -1";
    target[8] = "1 -1 1 1 1";
    target[9] = "1 -1 1 1 -1";
    target[10] = "1 -1 1 -1 1";
    target[11] = "1 -1 1 -1 -1";
    target[12] = "1 -1 -1 1 1";
    target[13] = "1 -1 -1 1 -1";
    target[14] = "1 -1 -1 -1 1";
    target[15] = "1 -1 -1 -1 -1";
    target[16] = "-1 1 1 1 1";
    target[17] = "-1 1 1 1 -1";
    target[18] = "-1 1 1 -1 1";
    target[19] = "-1 1 1 -1 -1";
    target[20] = "-1 1 -1 1 1";
    target[21] = "-1 1 -1 1 -1";
    target[22] = "-1 1 -1 -1 1";
    target[23] = "-1 1 -1 -1 -1";
    target[24] = "-1 -1 1 1 1";
    target[25] = "-1 -1 1 1 -1";

    /*
    target[0] = "1 1 1 1 1";        // A
    target[1] = "1 1 1 1 -1";       // B
    target[2] = "1 1 1 -1 1";       // C
    target[3] = "1 1 1 -1 -1";      // D
    target[4] = "1 1 -1 1 1";       // E
    target[5] = "1 1 -1 1 -1";      // F
    target[6] = "1 1 -1 -1 1";      // G
    target[7] = "1 1 -1 -1 -1";     // H
    target[8] = "1 -1 1 1 1";       // I
    target[9] = "1 -1 1 1 -1";      // J
    target[10] = "1 -1 1 -1 1";     // K
    target[11] = "1 -1 1 -1 -1";    // L
    target[12] = "1 -1 -1 1 1";     // M
    target[13] = "1 -1 -1 1 -1";    // N
    target[14] = "1 -1 -1 -1 1";    // O
    target[15] = "1 -1 -1 -1 -1";   // P
    target[16] = "-1 1 1 1 1";      // Q
    target[17] = "-1 1 1 1 -1";     // R
    target[18] = "-1 1 1 -1 1";     // S
    target[19] = "1 -1 -1 1 -1";    // T
    target[20] = "-1 1 -1 1 1";     // U
    target[21] = "-1 1 -1 1 -1";    // V
    target[22] = "-1 1 -1 -1 1";    // W
    target[23] = "1 -1 1 1 -1";     // X
    target[24] = "-1 -1 1 1 1";     // Y
    target[25] = "-1 -1 -1 -1 -1";  // Z
    */

}

private void button1_Click(object sender, EventArgs e)
{
    DialogResult result = folderBrowserDialog1.ShowDialog();
    if (result == DialogResult.OK)
    {
```

```
                string[] files =
Directory.GetFiles(folderBrowserDialog1.SelectedPath);
                lblFolderFont.Text = folderBrowserDialog1.SelectedPath;

                bool cek, cekTotal = true;
                for (int i = 65; i < 91; i++)
                {
                    cek = File.Exists(folderBrowserDialog1.SelectedPath + "/" +
(char)i + ".bmp");
                    if (!cek)
                    {
                        cekTotal = false;
                        break;
                    }
                }
                if (cekTotal)
                {
                    thisConn.Open();
                    thisCommand = new OleDbCommand("delete from mst_letter where
ucase(font)='" + namaFont.ToUpper() + "'", thisConn);
                    thisCommand.ExecuteNonQuery();
                    for (int i = 65; i < 91; i++)
                    {
                        File.Copy(folderBrowserDialog1.SelectedPath + "/" + (char)i +
".bmp", AppDomain.CurrentDomain.BaseDirectory + "letters/" + (char)i + "-" + namaFont
+ ".bmp", true);
                        pictureBox1.Image =
Image.FromFile(folderBrowserDialog1.SelectedPath + "/" + (char)i + ".bmp");
                        Bitmap bmp = new Bitmap(Treshold(pictureBox1.Image,
treshold));
                        arrayTreshold2 = thinning_gambar(bmp.Width, bmp.Height);
                        bmp = ToBitmap(arrayTreshold2);
                        pictureBox1.Image = bmp;

                        OutputData(txtOutputData, arrayTreshold2);
                        string binerFinal = txtOutputData.Text;

                        thisCommand = new OleDbCommand("insert into
mst_letter(letter,target,font,binary_data) values('" + (char)i + "','" + target[i -
65] + "','" + namaFont + "','" + binerFinal + "')", thisConn);
                        thisCommand.ExecuteNonQuery();

                    }
                    thisConn.Close();

                    MessageBox.Show("Import Font " + namaFont.ToUpper() + "
sukses...");
                }
                else
                {
                    MessageBox.Show("Font tidak lengkap...");
                }
            }
        }

        private void txtNamaFont_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                if (txtNamaFont.Text.Trim() == String.Empty)
                {
                    MessageBox.Show("Anda belum mengisi nama font...");
```

```
            txtNamaFont.Text = "";
        }
        else
        {
            namaFont = txtNamaFont.Text.Trim();
            button1.Enabled = true;
            button1.Focus();
            button1_Click(null, null);
        }
    }
}

private void Form1_Activated(object sender, EventArgs e)
{
    txtNamaFont.Focus();
}

private void txtNamaFont_Click(object sender, EventArgs e)
{
    txtNamaFont.Text = "";
    lblFolderFont.Text = "";
    button1.Enabled = false;
}

private void button2_Click(object sender, EventArgs e)
{
    btnLatih.Enabled = false;
    thisDataset.Reset();
    string query;
    thisConn.Open();
    query = "select * from mst_letter order by font,letter";
    thisAdapter = new OleDbDataAdapter(query, thisConn);
    thisAdapter.Fill(thisDataset);
    thisConn.Close();
    dgvTampil.DataSource = null;
    dgvTampil.Refresh();
    dgvTampil.DataSource = thisDataset.Tables[0];

    jumlahFont = dgvTampil.Rows.Count;
    if (jumlahFont > 0)
    {
        btnLatih.Enabled = true;
        jumlahX = dgvTampil.Rows[0].Cells[4].Value.ToString().Trim().Length;
        jumlahY = 5;
        char[] delimiter = { ' ' };
        lblJumlahFont.Text = "Jumlah Font: " + jumlahFont.ToString();
        lblPanjangX.Text = "Panjang X: " + jumlahX.ToString();
        lblPanjangY.Text = "Panjang Y: " + jumlahY.ToString();
        for (int f = 0; f < jumlahFont; f++)
        {
            //untuk mendapatkan Y tersimpan disetiap font
            target2[f]=dgvTampil.Rows[f].Cells[2].Value.ToString();
            string text = target2[f];
            string[] words = text.Split(delimiter);

            for (int i = 0; i < jumlahY; i++)
            {
                t[f, i] = Convert.ToInt32(words[i]);
                //MessageBox.Show(f+">>"+t[f,i].ToString());
            }

            //untuk menampung biner pixel X disetiap font
```

```
                    for (int i = 0; i < jumlahX; i++)
                    {
                        x[f, i] =
Convert.ToInt32(dgvTampil.Rows[f].Cells[4].Value.ToString().Substring(i, 1));
                        if (x[f, i] == 0) { x[f, i] = -1; }
                        //MessageBox.Show(f + ">>" + x[f, i].ToString());
                    }
                }
                //MessageBox.Show("Data X dan Y dari seluruh font sudah disimpan dalam
array" + Environment.NewLine + "Silakan lanjut ke Proses Latih");
                 for (int i = 0; i < 3; i++)
                    {
                        //MessageBox.Show(target2[i].ToString());
                    }

            }
            else
            {
                MessageBox.Show("Data kosong...");
            }
        }

        private void tabPage2_Click(object sender, EventArgs e)
        {

        }

        private void Pilih(DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {

            }
        }

        private void dgvTampil_Click(object sender, EventArgs e)
        {

        }

        private void dgvTampil_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            DataGridViewRow row = this.dgvTampil.Rows[e.RowIndex];
        }

        private void dgvTampil_CellEnter(object sender, DataGridViewCellEventArgs e)
        {
            Pilih(e);
        }


        private void button3_Click(object sender, EventArgs e)
        {
            Cursor.Current = Cursors.WaitCursor;
            progressBar2.Maximum = 200;
            int cek_w;
            double moduscek;
            do
            {
                //cek_w = 0;
                for (int i = 0; i < jumlahFont; i++) //i < jumlahFont, tapi bisa
diganti ke 1 untuk cek huruf pertama saja
```

```
                {
                    //cek_w = 0;
                    for (int k = 0; k < jumlahY; k++)
                    {

                        for (int j = 0; j < jumlahX; j++)
                        {
                            w_lama[k, j] = w_baru[k, j];
                            y[i, k] += w_lama[k, j] * x[i, j];
                        }
                        y[i, k] += bias[k];

                        if (y[i, k] > tetha) { y[i, k] = 1; }
                        else if (y[i, k] <= tetha && y[i, k] >= tetha * -1) { y[i, k]
= 0; }
                        else if (y[i, k] < tetha * -1) { y[i, k] = -1; }

                        if (k == jumlahY - 1) { y_training += y[i, k].ToString(); }
                        else { y_training += y[i, k].ToString() + " "; }

                    }

                    if (y_training != target2[i])
                    {
                        for (int k = 0; k < jumlahY; k++)
                        {
                            for (int j = 0; j < jumlahX; j++)
                            {
                                w_baru[k, j] = w_lama[k, j] + alfa * t[i, k] * x[i,
j];

                                bias[k] = bias[k] + alfa * t[i, k];
                                //w_lama[k, j] = w_baru[k, j];
                            }
                        }
                        //cek_w=1;
                    }
                    //textBox2.Text = counter.ToString();
                } counter++;
                progressBar2.Value = counter;
                //moduscek = cekBobot2();
                //MessageBox.Show(moduscek.ToString());
            } while (counter < 200);

            Cursor.Current = Cursors.Default;
            MessageBox.Show("Done...");
        }

        private void cekBobot()
        {
            int neuron=0,s=49;
            StringBuilder bobot2 = new StringBuilder();
            neuron = Convert.ToInt32(comboBox1.Text);
            for (int i = 0; i < jumlahX; i++ )
            {
                bobot2.Append(w_baru[neuron-1, i].ToString()+" ");
                if (i % s == 0) { bobot2.AppendLine(); }
            }
            textBox1.Text = bobot2.ToString();
        }

        private void cekBias()
        {
```

```
            int neuron = 0;
            StringBuilder bias2 = new StringBuilder();
            neuron = Convert.ToInt32(comboBox2.Text);
            bias2.Append(bias[neuron-1].ToString());
            textBox19.Text = bias2.ToString();
        }

        private void btnPengenalan_Click(object sender, EventArgs e)
        {
            PengenalanCaraLama();
        }


        private void PengenalanCaraLama()
        {
            Cursor.Current = Cursors.WaitCursor;

            double[] y_in = new double[5];
            string tamp = "";
            for (int k = 0; k < jumlahY; k++)
            {
                for (int j = 0; j < jumlahX; j++)
                {
                    if (x_pengenalan[j] == 0) { x_pengenalan[j] = -1; }
                    y_in[k] += x_pengenalan[j] * w_baru[k, j];
                }
                y_in[k] += bias[k];

                if (y_in[k] > tetha) { y_in[k] = 1; }
                else if (y_in[k] <= tetha && y_in[k] >= tetha * -1) { y_in[k] = 0; }
                else if (y_in[k] < tetha * -1) { y_in[k] = -1; }

                if (k == jumlahY - 1) { tamp += y_in[k].ToString(); }
                else { tamp += y_in[k].ToString() + " "; }
            }


            Cursor.Current = Cursors.Default;
            thisDatasetPengenalan.Clear();
            string query;
            thisConn.Open();
            query = string.Format("select letter from mst_letter where target='{0}'",
tamp);
            thisAdapter = new OleDbDataAdapter(query, thisConn);
            thisAdapter.Fill(thisDatasetPengenalan);
            thisConn.Close();
            txtHasilHitung.Text = tamp.ToString();//MessageBox.Show(tamp);
            if (thisDatasetPengenalan.Tables[0].Rows.Count > 0)
            {
                txtHasilHuruf.Text =
thisDatasetPengenalan.Tables[0].Rows[0][0].ToString();//MessageBox.Show("Hasil: " +
thisDatasetPengenalan.Tables[0].Rows[0][0].ToString());
            }
            else
            {
                MessageBox.Show("Tidak ditemukan...");
            }
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            cekBobot();
```

```
        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            cekBias();
        }

    }
}
```

Lampiran 2

Font Arial Bold pada Database

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T
U	V	W	X
Y	Z		

Lampiran 3

Font Arial Bold dengan Variasi Noise

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T
U	V	W	X
Y	Z		

Lampiran 4

Font Arial Bold Italic

A	*B*	*C*	*D*
E	*F*	*G*	*H*
I	*J*	*K*	*L*
M	*N*	*O*	*P*
Q	*R*	*S*	*T*
U	*V*	*W*	*X*
Y	*Z*		

Lampiran 5

Font Arial Regular

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T
U	V	W	X
Y	Z		

Universitas Kristen Duta Wacana
Fakultas Teknologi Informasi Program Studi Teknik Informatika
Jl. Dr. Wahidin Sudirahusada 5-25 Yogyakarta 55224
Telp.: (0274)563929 ext. 321 Faks.: (0274)513235

FORMULIR PERBAIKAN (REVISI) SKRIPSI

Dicetak tanggal: 24-04-2014 19:04:54

Yang bertanda tangan di bawah ini:

Nama	: STEPY KUMARA KALANGIE
N I M	: 22084552
Judul Skripsi	: IMPLEMENTASI ALGORITMA THINNING BINARY REGIONS DAN ALGORITMA PERCEPTRON UNTUK PENGENALAN CITRA HURUF
Tanggal Pendadaran	: Jumat, 4 April 2014 pukul 08:00 WIB

Telah melakukan perbaikan tugas akhir dengan lengkap.

Demikian pernyataan kami agar dapat dipergunakan sebagaimana mestinya.

Yogyakarta, Kamis, 24 April 2014

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Sri Suwarno, M.Eng.	Dra. Widi Hapsari, M.T.