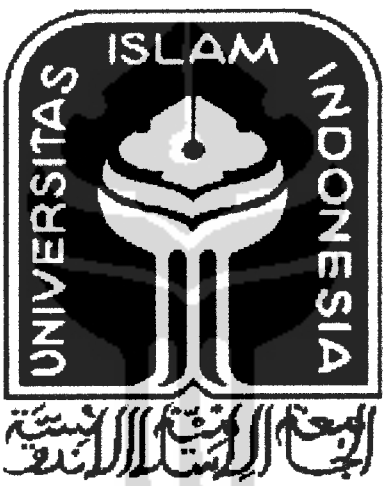
RANCANG BANGUN *TOOL* UNTUK JARINGAN SYARAF TIRUAN (JST) MODEL PERCEPTRON

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika



Oleh :

Nama : Liza Afriyanti
NIM : 06 523 058

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2010

LEMBAR PENGESAHAN PEMBIMBING

RANCANG BANGUN *TOOL* UNTUK JARINGAN SYARAF TIRUAN (JST) MODEL PERCEPTRON

TUGAS AKHIR

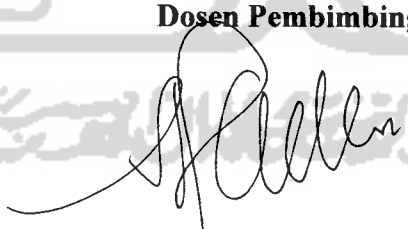
Disusun oleh :

Nama : Liza Afriyanti

NIM : 06 523 058

Yogyakarta, 9 Juni 2010

Dosen Pembimbing,



DR. Sri Kusumadewi, S.Si., MT

LEMBAR PENGESAHAN PENGUJI

RANCANG BANGUN *TOOL* UNTUK JARINGAN SYARAF TIRUAN (JST) MODEL PERCEPTRON

TUGAS AKHIR

Disusun oleh :

Nama : Liza Afriyanti

NIM : 06 523 058

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 28 Juni 2010

Tim Penguji,

DR. Sri Kusumadewi, S.Si., MT.
Ketua

Arwan Ahmad Khoiruddin, S.Kom, M.Cs.
Anggota I

Beni Suranto, S.T.
Anggota II

Mengetahui,
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

Yudi Prayudi, S.Si., M.Kom.

TAKARIR

Toolbox	:	Kotak peralatan/ Peralatan
Threshold	:	Batas ambang
Learning rate	:	Laju pemahaman
Neuron	:	Unit pemroses informasi yang menjadi dasar dalam pengoperasian jaringan syaraf tiruan
Setting	:	Pengaturan
Install	:	Pemasangan sistem
Neuron layers	:	Lapisan neuron
Error	:	Kesalahan
Epoch	:	Satu siklus pelatihan yang melibatkan semua pola
Swing	:	Satu framework java yang menggunakan komponen gui di dalam paradigma pemrograman java
AWT	:	(Abstract Window Toolkit), penyedia komponen window/GUI (Graphical User Interface) pada Java

DAFTAR ISI

Permasalahan dalam JST yang sering dihadapi *user* adalah tidak dapat memahami struktur pada jaringannya. Hal ini terjadi karena tidak didukung dengan tersedianya suatu aplikasi yang dapat membantu *user* dalam memahami struktur JST. Untuk dapat memahami struktur JST selain memahami teori juga diperlukan pemahaman secara visual. Untuk itu diperlukan sebuah aplikasi yang dapat membantu *user* dalam memahami struktur JST model Perceptron. Berdasarkan latar belakang tersebut, maka pada penelitian ini akan dibangun sebuah *tool* untuk Jaringan Syaraf Tiruan model perceptron.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas dapat dirumuskan permasalahan yang akan diselesaikan yaitu :

a. Bagaimana meningkatkan pemahaman user dalam memahami Jaringan Syaraf Tiruan model Perceptron.
b. Bagaimana membuat alat bantu (*tool*) untuk membuat Jaringan Syaraf Tiruan model perceptron yang dapat memberikan visualisasi secara grafis.
c. Bagaimana membangun aplikasi untuk mengimplementasikan struktur Jaringan Syaraf Tiruan model Perceptron dengan menggunakan Java.

1.3 Batasan Masalah

Dalam melaksanakan suatu penelitian diperlukan adanya batasan agar tidak menyimpang dari yang telah direncanakan sehingga tujuan yang sebenarnya dapat tercapai. Batasan masalah yang diperlukan yaitu :

1. Aplikasi ini dibuat untuk dijalankan pada desktop.
2. Program yang akan dibuat nantinya akan menampilkan struktur JST model Perceptron.
3. Masukan yang diperlukan antara lain jumlah variabel *input*, nilai variabel *input*, bobot, alpha (*learning rate)*, *threshold*, maksimum epoh dan target (*output*).
4. Iterasi dilakukan terus hingga semua pola memiliki keluaran jaringan yang sama dengan targetnya atau tercapainya epoh maksimum.

1.7 Sistematika Penulisan

Dalam penyusunan tugas akhir ini, sistematika penulisan dibagi menjadi beberapa bab sebagai berikut :

BAB I Pendahuluan, bab ini berisi pembahasan masalah umum yang meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB II Perceptron, bagian ini memuat landasan teori yang berfungsi sebagai sumber atau alat dalam memahami permasalahan yang berkaitan dengan teori Jaringan Syaraf Tiruan, definisi Jaringan Syaraf Tiruan, cara kerja komponen Jaringan Syaraf Tiruan, konsep belajar Jaringan Syaraf Tiruan, arsitektur Jaringan Syaraf Tiruan, fungsi aktivasi atau fungsi transfer, dan Jaringan Syaraf Tiruan model perceptron serta pelatihan perceptron.

BAB III Model Sistem, bagian ini memuat uraian tentang gambaran umum model sistem yang diusulkan meliputi proses *instalasi* sistem, proses pembuatan arsitektur jaringan, proses *setting* data, dan terakhir proses pelatihan dan pengujian pada sistem.

BAB IV Perancangan Sistem, pada bab ini membahas mengenai perancangan sistem diagram UML (*Unified Modelling Language*), perancangan struktur Jaringan Syaraf Tiruan, perancangan visual jarigan dan implementasi perangkat lunak yang dibuat dan memuat dokumentasi atau tampilan form-form yang telah dibangun.

BAB V Pengujian, bab ini membahas tentang analisis kinerja dari perangkat lunak. Pada bagian ini mengulas analisis hasil pengujian terhadap sistem yang dibandingkan dengan kebenaran dan kesesuaiannya dengan kebutuhan perangkat lunak yang telah dituliskan pada bagian sebelumnya.

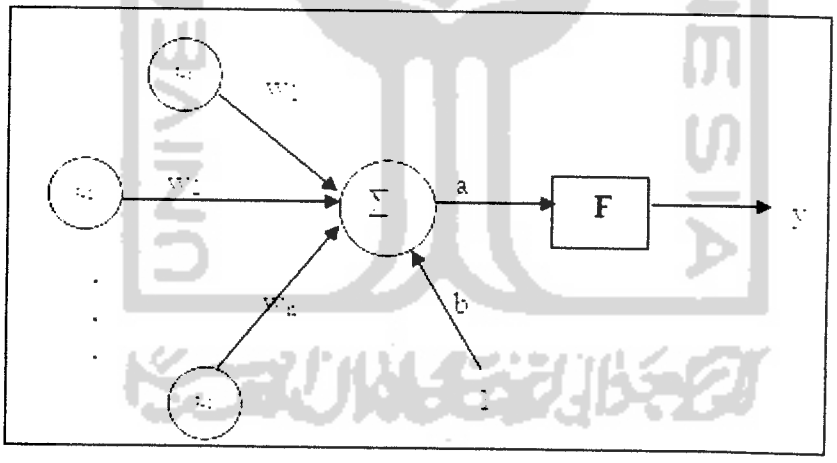
BAB VI Penutup, membuat kesimpulan-kesimpulan yang merupakan rangkuman dari hasil analisis kinerja pada bagian sebelumnya dan saran yang perlu diperhatikan berdasarkan keterbatasan yang ditemukan dan asumsi-asumsi yang dibuat selama pembuatan *tool* untuk JST model Perceptron.

2.1.5 Fungsi Aktivasi atau Fungsi Transfer

Fungsi Aktivasi adalah fungsi yang menggambarkan hubungan antara tingkat aktivasi internal (*summation function*) yang mungkin berbentuk linear atau *nonlinear*. Fungsi aktivasi lebih tepat digunakan untuk mengaktifkan neuron bila telah melewati suatu ambang tertentu (*threshold*), pengaktifan ini dilakukan terhadap bobot total penjumlahan beberapa nilai bobot yang datang.

Berikut fungsi-fungsi aktivasi yang biasanya digunakan dalam sistem jaringan syaraf :

a. Fungsi *linier* atau identitas
b. Fungsi *step biner*
c. Fungsi *bipolar*
d. Fungsi *saturating linear*
e. Fungsi *symmetric saturating linear*
f. Fungsi *sigmoid biner*
g. Fungsi *sigmoid bipolar*



Gambar 2. 7 Fungsi Aktivasi JST

3.3 Proses Setting Data

Setelah aplikasi dapat dijalankan, langkah selanjutnya memberikan data yang dibutuhkan pada aplikasi sehingga aplikasi dapat menjadi suatu sistem yang dapat digunakan untuk membangun struktur Jaringan Syaraf Tiruan (JST) model perceptron. Pada langkah ini, *user* harus memasukkan data yang sudah dirancang pada proses pembuatan arsitektur jaringan seperti jumlah variabel *input*, nilai variabel *input*, alpha, bobot, *threshold*, bias, dan target (*output*).

3.4 Proses Pelatihan dan Pengujian pada Sistem

3.4.1 Proses Pelatihan

Pada proses pelatihan setiap kali pola dimasukkan, hasil keluaran jaringan dibandingkan dengan target awal. Jika terdapat perbedaan, maka bobot akan dimodifikasi. Pelatihan terus dilakukan berulang-ulang untuk semua kemungkinan pola yang ada hingga jaringan dapat mengerti polanya (ditandai dengan samanya semua keluaran jaringan dengan target keluaran yang diinginkan).

Misalkan s sebagai vektor masukan, t adalah target keluaran, α adalah laju pemahaman, θ adalah nilai threshold. Perhatikan algoritma untuk pelatihan perceptron di bawah ini :

Langkah 0 : Inisialisasi semua bobot dan bias (umumnya $w_i = b = 0$). Set laju pembelajaran α ($0 < \alpha \leq 1$) (untuk penyederhanaan set $\alpha = 1$). Kemudian set epoh = 0.

Langkah 1 : Selama kondisi berhenti benilai FALSE atau selama ada elemen vektor masukan yang respon unit keluarannya tidak sama dengan target ($y \neq t$), lakukan langkah-langkah 2 – 6.

Langkah 2 : Untuk setiap pasangan (s, t), kerjakan langkah 3 – 5. Pada langkah ini epoh = epoh + 1. Epoh atau iterasi akan berhenti jika y = t atau tercapainya epoh maksimum.

Langkah 3 : Set aktivasi unit masukan $x_i = s_i$ (i = 1, ..., n)

a. Antar muka halaman *home*, sebagai halaman utama sistem.
b. Antar muka untuk *input* data, halaman untuk memasukkan data seperti jumlah variabel *input*, nilai variabel *input*, bobot, alpha (*learning rate)*, *threshold*, maksimum epoh dan target (*output*).
c. Antar muka untuk informasi hasil perhitungan, halaman untuk menampilkan hasil perhitungan epoh .
d. Antar muka untuk tabel perhitungan epoh terkahir, halaman untuk menampilkan tabel hasil perhitungan epoh yang terkahir.
e. Antar muka untuk proses pengujian, halaman yang berguna untuk melakukan proses pengujian terhadap proses pembelajaran perceptron.
f. Antar muka untuk struktur jaringan, halaman yang menampilkan struktur jaringan.
g. Antar muka halaman *about*, merupakan halaman untuk memberikan informasi mengenai program dan *programmer*.
h. Antar muka halaman *help*, merupakan halaman untuk panduan penggunaan *tool* untuk JST model perceptron.

4.1.6 Kebutuhan Perangkat Lunak yang Dibutuhkan

Perangkat lunak yang dibutuhkan untuk pembuatan *tool* JST model perceptron adalah sebagai berikut :

1. Sun Microsystem Java 2 Standard Edition SDK (J2SDK 1.6), berfungsi untuk menjalankan aplikasi Java.
2. *Integrated Development Environment* (IDE) Netbeans 6.8, berfungsi sebagai *text editor* dan *compiler* and *debugging* program.
3. Rational Rose Enterprise Edition, berfungsi untuk perancangan UML.
4. Adobe Photoshop CS3, berfungsi untuk mendesain icon, judul, dan tampilan.

4.1.7 Kebutuhan Perangkat Keras yang Dibutuhkan

Perangkat keras komputer yang dibutuhkan adalah perangkat keras yang dapat mendukung perangkat lunak yang memiliki kemampuan atau tampilan grafis yang cukup baik. Perangkat keras minimum yang digunakan pada *tool* untuk JST model perceptron adalah :

1. Processor intel Pentium.
2. Memori 520 MB
3. Hardisk 120 GB
4. VGA 128 MB
5. Monitor resolusi 1280 x 800
6. Mouse dan keyboard

4.2 Perancangan Sistem

4.2.1 Use Case Diagram

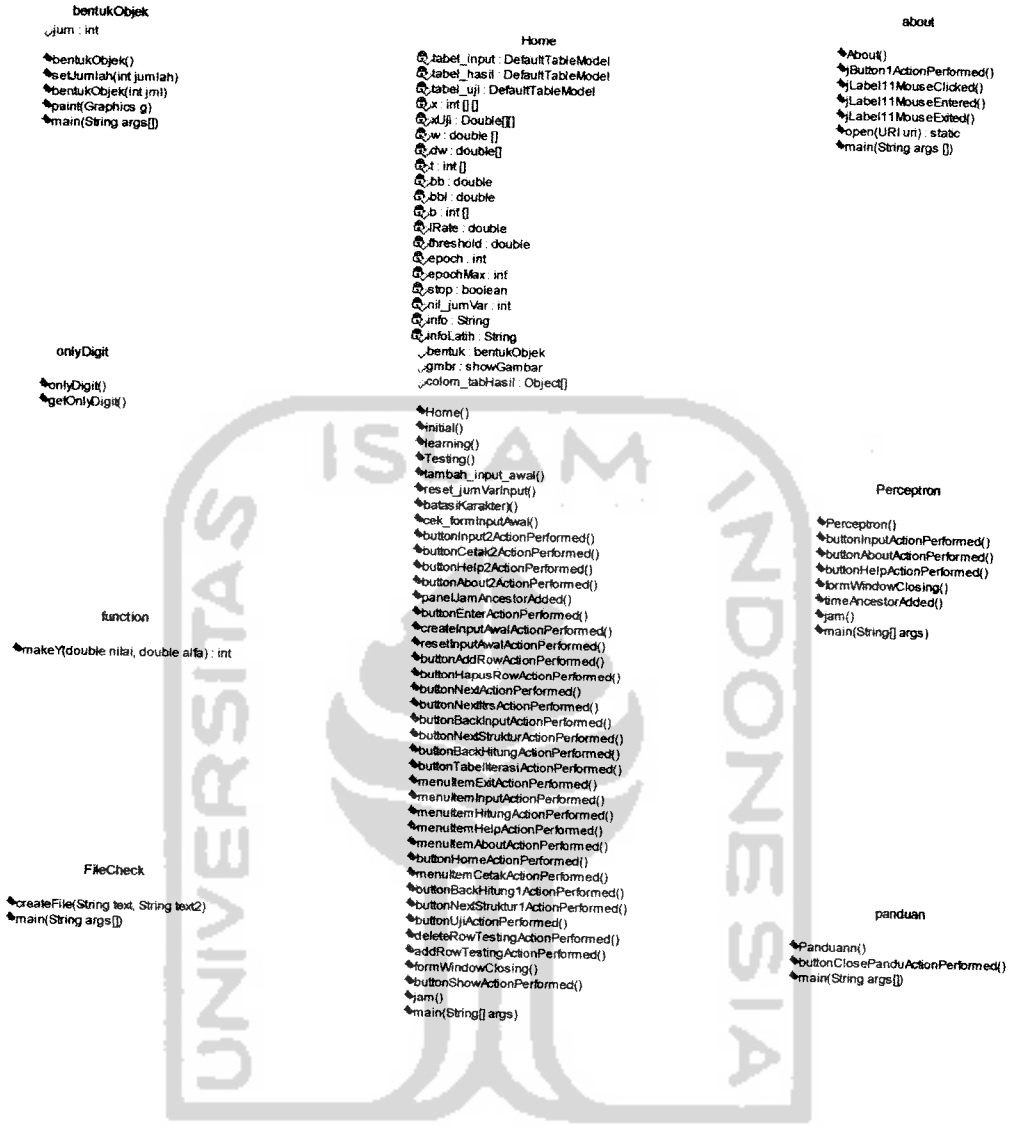
Pada *tool* untuk JST model perceptron, *use case* menjelaskan tentang hubungan antara sistem dengan aktor. Hubungan ini dapat berupa *input* aktor ke sistem ataupun *output* ke aktor. Gambar 4.1 berikut ini akan dijelaskan *Use case diagram* yang terdapat dalam rancang bangun *tool* untuk JST model perceptron.

Gambar 4. 1 *Use Case Diagram Tool* Untuk JST Model Perceptron

4.2.2 *Class Diagram*

Class diagram digunakan untuk melakukan visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak digunakan. *Class diagram* juga dapat memperlihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain (*logical view*) dari suatu sistem. Selama proses desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

Gambar 4.2 berikut ini adalah akan dijelaskan *class diagram* yang digunakan untuk melakukan visualisasi struktur kelas-kelas yang terdapat dalam rancang bangun *tool* untuk JST model perceptron.



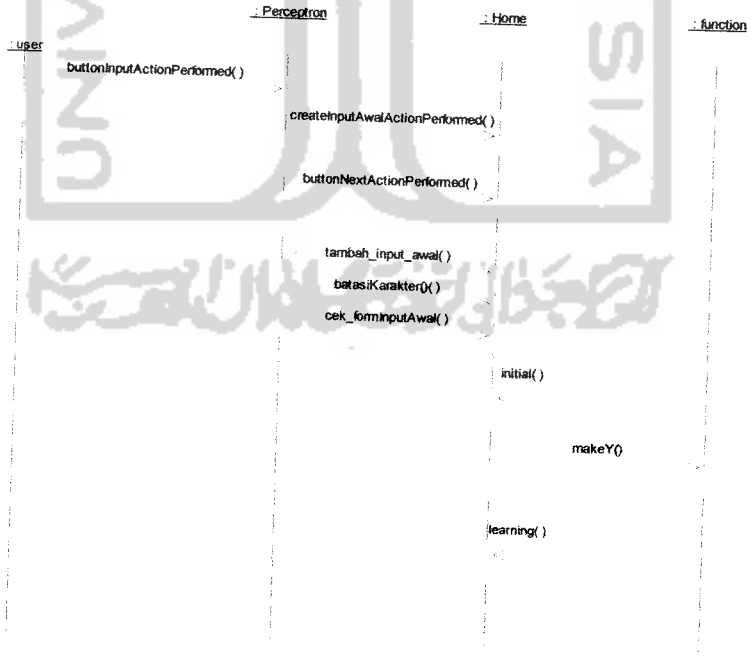
Gambar 4. 2 *Class Diagram Tool* untuk JST Model Perceptron

Class Entity : Tab Result Data

Keterangan :

1. *User* mengawali *sequence* ini dengan memanggil *method* *buttonInputActionPerformed* ().
2. *Home* melakukan instansiasi dan memanggil *method* *createInputAwalActionPerformed*() untuk memproses *input* dari *user*. Setelah itu *user* memanggil method *buttonNextActionPerformed* () untuk melakukan proses pelatihan pada sistem.
3. Kemudian tab *Result Data* akan menampilkan hasil pelatihan perceptron.

Sequence diagram proses pelatihan dapat dilihat pada gambar 4.4.



Gambar 4. 4 *Sequence Diagram* Proses Pelatihan

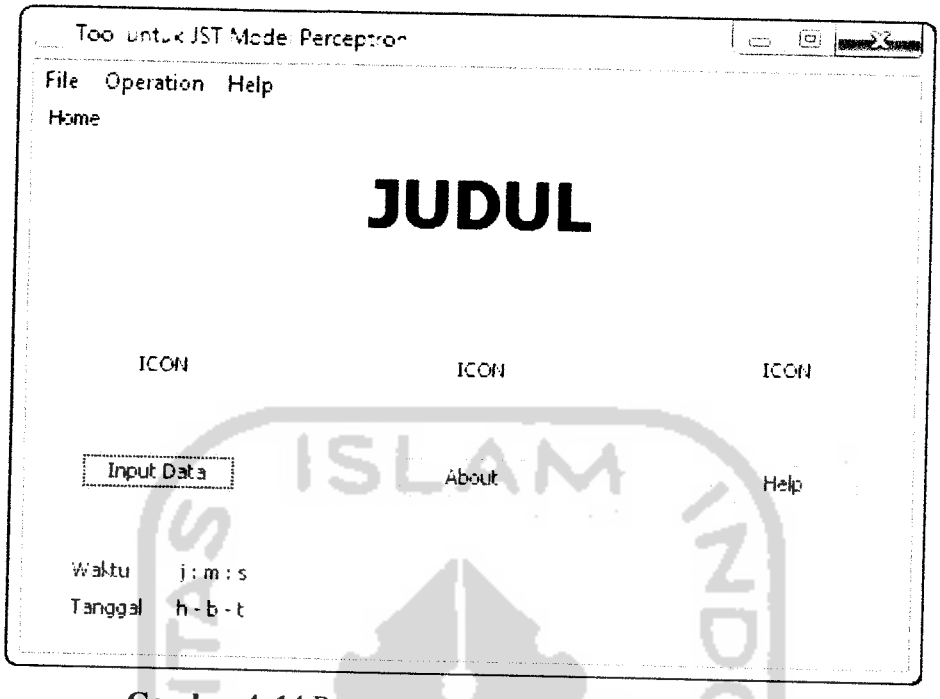
4.2.4 *Activity Diagram*

Activity diagram merupakan diagram yang menggambarkan alur kerja dari sistem. Diagram ini merupakan suatu diagram dinamis yang menunjukkan aktivitas beserta kejadian yang menyebabkan suatu objek berada dalam *state* tertentu. *Activity Diagram* ini lebih menggambarkan transisi-transisi dan aktivitas-aktivitas yang menyebabkan perubahan pada *state* objek. Simbol lingkaran berisi warna hitam menandakan awal *state* sedangkan simbol lingkaran berisi warna hitam yang dilingkari oleh lingkaran bergaris hitam menandakan akhir *state*.

Pada bagian ini akan dijelaskan tentang aktivitas yang dilakukan oleh *user*. *Activity Diagram* pembuatan *tool* untuk JST model perceptron kegiatan di mulai dari isntalasi sistem, sampai pelatihan dan pengujian.

1. Pelatihan data. Urutan aktivitas proses pelatihan data pada sistem dapat dijelaskan sebagai berikut :
   a. *User* melakukan instalasi sistem.
   b. Setelah sistem terpasang *user* akan melakukan *setting* pada data.
   c. Kemudian *User* membuat struktur jaringan dengan memasukkan data pada sistem seperti jumlah variabel *input*, nilai variabel *input*, bobot, alpha (*learning rate)*, *threshold*, maksimum epoh dan target (*output*).
   d. Kemudian sistem akan meminta *user* untuk memasukkan nilai variabel (nilai untuk x_1, x_2, x_n) dan target (*Output*).
   e. Sistem akan melakukan pelatihan berdasarkan data yang telah dimasukkan.
   f. Sistem menghasilkan keluaran berupa hasil perhitungan epoh terakhir dan struktur jaringan yang di hasilkan.
   g. Kemudian sistem mengakhiri kegiatan ini.

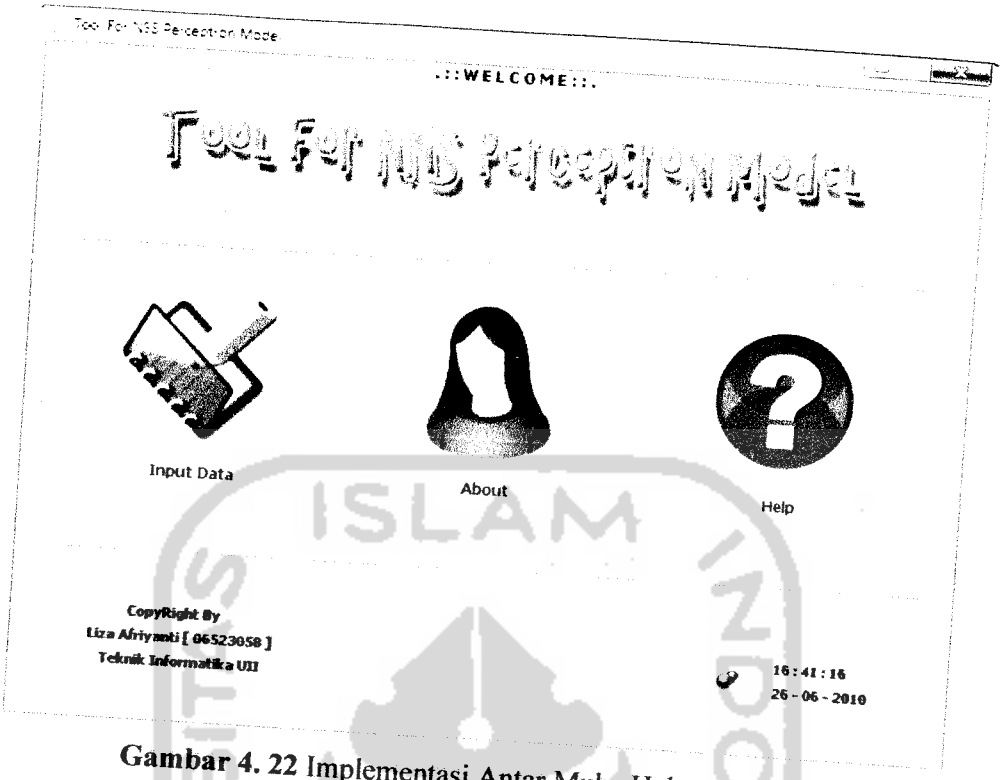
2. Pengujian data. Urutan untuk proses pengujian data sama seperti pada proses pelatihan, namun pada proses pengujian data, sistem tidak melakukan iterasi dan bobot yang digunakan adalah bobot dari hasil pelatihan. Urutan proses pengujian data adalah sebagai berikut :



Gambar 4. 14 Rancangan Antar Muka Halaman *Home*

b. Antar Muka Halaman *Input Data*

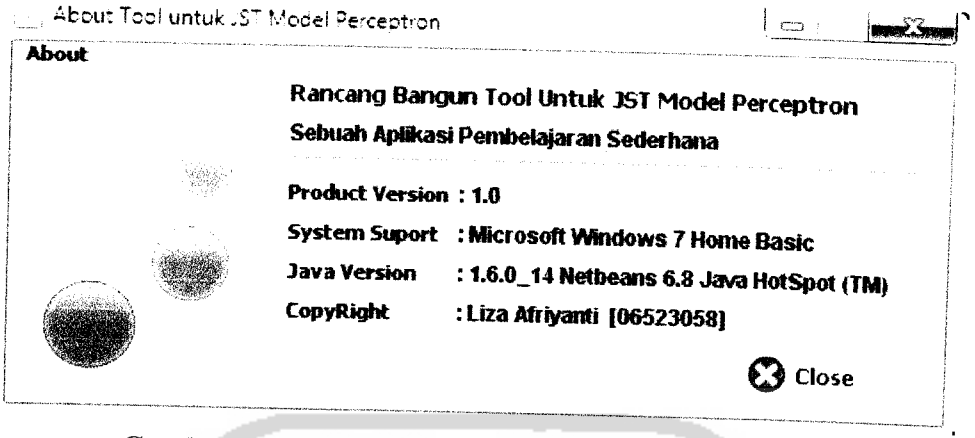
Pada halaman *input data user* diberikan form untuk menentukan jumlah variabel, nilai threshold dan nilai alpha. Setelah itu *user* mengisi tabel data variabel. Gambar 4.15 menunjukkan rancangan antar muka *Input Data.*



Gambar 4. 22 Implementasi Antar Muka Halaman *Home*

b. Halaman *Input* Data

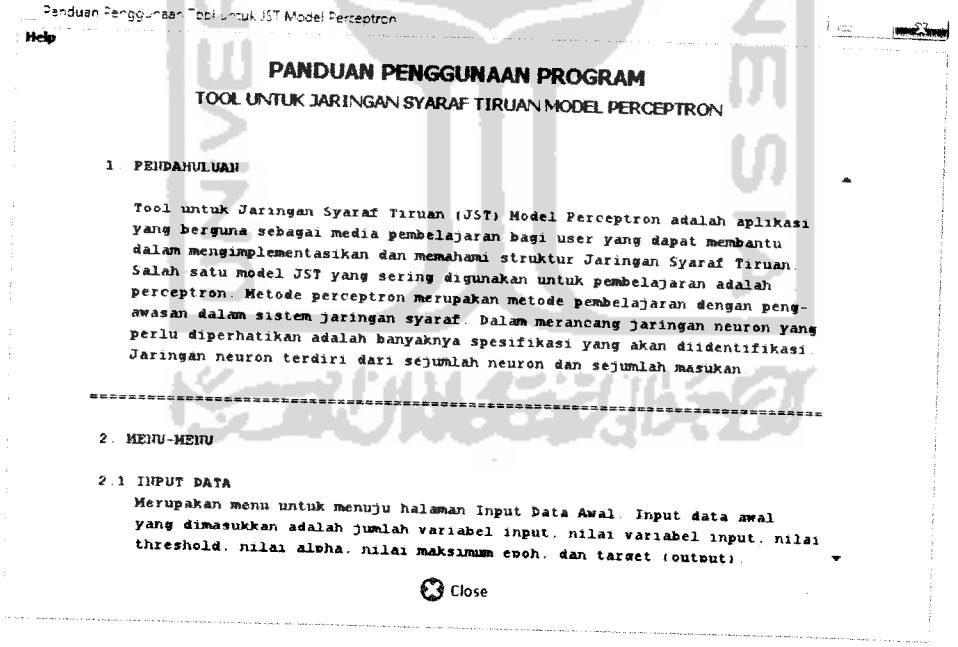
Pada *tool* untuk JST model perceptron, halaman *input* data terdiri dari beberapa pilihan tombol, yaitu tombol *create* untuk menginstansiasi *input*, tombol *reset* untuk menset ulang *input* awal, tombol *add row* untuk menambah baris pada tabel *input*, tombol *delete row* untuk menghapus baris pada tabel *input*, dan tombol *next* untuk menuju ke halaman hasil. Gambar 4.23 menunjukkan implementasi antar muka halaman *Home tool* untuk JST model perceptron.



Gambar 4. 28 Implementasi Antar Muka Halaman *About*

h. Halaman *Help*

Halaman *help* adalah halaman yang berisikan manual dan cara penggunaan *tool* untuk JST model perceptron. Gambar 4.29 menunjukkan implementasi antar muka halaman *help tool* untuk JST model perceptron.



Gambar 4. 29 Implementasi Antar Muka Halaman *Help*

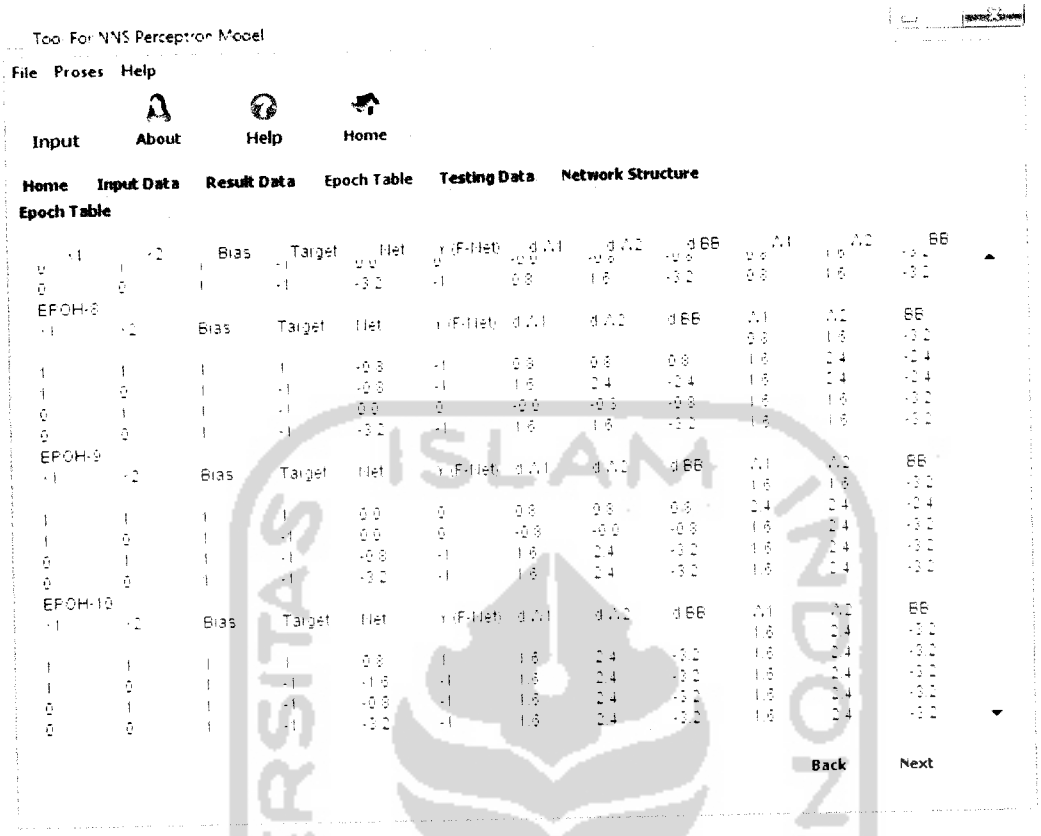
```
            //menampilkan nilai bias dan target pada text
area
            info += " b  t \n";
            for (int i = 0; i < x.length; i++) {
                for (int j = 0; j < x[0].length; j++) {
                    //inisialisasi node
                    info += x[i][j] + "   ";
                }
                //inisialisasi bias dan target
                info += b[i] + "   " + t[i] + "\n";
            }
            // initial bobot awal
            w = new
double[Integer.parseInt(jumVarInput.getText())];
            dw = new
double[Integer.parseInt(jumVarInput.getText())]; // variabel
untuk delta w
            for (int i = 0; i < w.length; i++) {
                dw[i] = 0;
                w[i] = 0;
                if (i == 0) {
                    info += "\n Bobot Awal (w) = [ " + w[i]
+ " ";
                } else if (i == (w.length - 1)) {
                    info += w[i] + " ]\n";
                } else {
                    info += w[i] + " ";
                }
            }
            // initial bobot bias
            bb = 0;
            info += " Bobot bias awal (bb) = [ " + bb + "
]\n";
            // initial threshold dan learning rate (alfa)
            lRate =
Double.parseDouble(nilAlpha.getText().trim());
            info += " Learning rate (alfa) = " + lRate +
"\n";
            threshold =
Double.parseDouble(nilThreshold.getText().trim());
            info += " Threshold (tetha) = " + threshold +
"\n";
        } catch (Exception e) {
            //  JOptionPane.showMessageDialog(null, "Terjadi
Kesalahan Inisialisasi Pada : \n" + e);
        }
    }
```

```
                    } else {
                        sama += 1;
                        for (int j = 0; j < x[0].length;
j++) {
                            hsl_akhr[(jml_node + 4 + j)] =
w[j];
                            hsl_akhr[(jml_node + 3 +
jml_node + 1 + 1 + j)] = w[j];
                            barisKosong[(jml_node + 3 +
jml_node + 1 + 1 + j)] = w[j];
                        }
                        hsl_akhr[(jml_node + 3 + jml_node +
1)] = bb;
                        hsl_akhr[(jml_node + 3 + jml_node +
1 + jml_node + 1)] = bb;
                        barisKosong[(jml_node + 3 + jml_node
+ 1 + jml_node + 1)] = bb;
                    }
                    tabel_hasil.addRow(hsl_akhr);
                }

                for (int i = 0; i < pemisah.length; i++) {
                    pemisah[i] = ("EPOH-" + (epoch + 1));
                }

                epohMaksimum =
Integer.parseInt(MaxEpoch.getText());
                if (sama == tabelInputAwal.getRowCount()) {
                    stop = true;
                    String bbt = "", trg = "";
                    jumIterasi.setText(df.format(epoch));
                    nilAlphaInfo.setText(df.format(lRate));
                    for (int i = 0; i < w.length; i++) {
                        bbt += " w" + (i + 1) + "= " + w[i]
+ ",";
                    }
                    bobotAkhir.setText(bbt);
                    bobotUji.setText(bbt);

nilThresholdInfo.setText(df.format(threshold));
                    for (int i = 0; i < t.length; i++) {
                        trg += " t" + (i + 1) + "= " + t[i]
+ ",";
                    }
                    nilTarget.setText(trg);
                    TextAreaInfo.setText(info);
                } else {
                    if (epoch < epohMaksimum) {
                        tabel_hasil.addRow(pemisah);
                        tabel_hasil.addRow(colom_tabHasil);
//mengisi data pada tabel _hasil utk epoh i
                        tabel_hasil.addRow(barisKosong);
                    }
                    String bbt = "", trg = "";
                    jumIterasi.setText(df.format(epoch));
                    nilAlphaInfo.setText(df.format(lRate));
```



Gambar 5. 4 Tampilan *Epoh Table*

e. Halaman *Testing Data*

User dapat melakukan proses pengujian data pada sistem setelah melakukan proses pelatihan data. Pada halaman *testing data* terdapat tabel untuk memasukkan nilai variabel untuk proses pengujian. Nilai bobot pada proses pengujian didapat dari hasil proses pelatihan. Kemudian sistem akan menampilkan hasil proses pengujian data pada text area *show testing information.* Tampilan *testing data* dapat dilihat pada gambar 5.5.

5.2 Kelebihan Sistem

Berdasarkan pengujian sistem diatas dan dikomparasikan dengan alat bantu (*tool*) Matlab, dapat diketahui kelebihan dari *tool* untuk Jaringan Syaraf Tiruan (JST) model perceptron adalah :

1. Sistem dapat melakukan proses pelatihan dan pengujian berdasarkan *input* data dari *user* seperti jumlah variabel *input*, nilai variabel *input*, alpha (*learning rate)*, *threshold*, maksimum epoh dan target (*output*).
2. Sistem dapat membuat struktur Jaringan Syaraf Tiruan Model Perceptron. Jumlah neuron dapat dibentuk berdasarkan jumlah variabel *input*.
3. Sistem dapat membantu seorang *user* dalam memecahkan suatu masalah khususnya dalam perceptron, dimana *software* ini menyediakan fasilitas menu-menu yang dapat mempermudah pekerjaan seorang *user*.

5.3 Kelemahan Sistem

Sedangkan kekurangan dari *tool* untuk Jaringan Syaraf Tiruan (JST) model perceptron adalah antarmuka yang kurang dinamis sehingga tampilan data pada tabel epoh dan struktur jaringan tidak tersusun rapi jika data masukan terlalu banyak (lebih dari 4 neuron).

BAB VI
PENUTUP

6.1 Kesimpulan

Dari hasil penelitian dan pembahasan yang telah dilakukan, dapat diambil kesimpulan bahwa *tool* untuk Jaringan Syaraf Tiruan (JST) model Perceptron :

1. Mampu melakukan proses pelatihan data berdasarkan nilai masukkan dari user seperti jumlah variabel *input*, nilai variabel *input*, alpha (*learning rate)*, *threshold*, maksimum epoh dan target (*output*).
2. Mampu melakukan proses pengujian data berdasarkan nilai bobot yang didapat dari proses pelatihan.
3. Sistem dapat membuat struktur Jaringan Syaraf Tiruan model perceptron. Jumlah neuron terbentuk berdasarkan jumlah variabel *input*.

6.2 Saran

Saran-saran untuk pengembangan *tool* untuk Jaringan Syaraf Tiruan (JST) model Perceptron berdasarkan kesimpulan yang diperoleh adalah :

1. Aplikasi akan lebih bagus lagi jika memiliki antarmuka yang dinamis seperti dapat di-*resizeable* atau *full screen*, sehingga data-data dapat tertata dengan rapi.
2. Diharapkan dalam pengembangan sistem selanjutnya dapat mencakup beberapa metode Jaringan Syaraf Tiruan (JST) seperti model Hebb, model Adaline, Back Propagation, dan lain-lain.