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

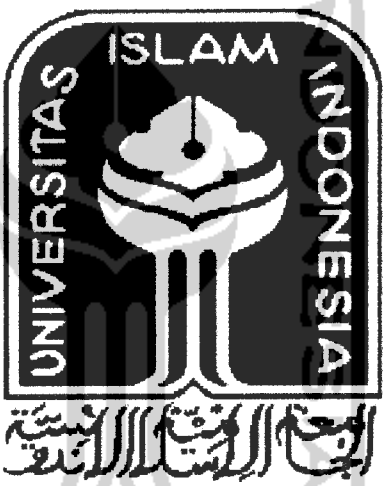
NIM : 06 523 058

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2010

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.

LEMBAR PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini,

Nama : Liza Afriyanti

NIM : 06 523 058

Tugas Akhir dengan judul :

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

Menyatakan bahwa seluruh komponen dan isi dalam Laporan Tugas Akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, maka saya akan siap menanggung resiko dan konsekuensi apapun.

Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 9 Juni 2010

Yang Membuat Pernyataan,

(Liza Afriyanti)

HALAMAN PERSEMBAHAN

Allahurabbi, The Guide of Every Single Guidnce,
Puji syukur kepada sumber dari suara-suara hati yang bersifat mulia, sumber ilmu pengetahuan, sumber dari segala kebenaran.
Sang Maha Cahaya, Sang Kekasih tercinta
yang tak terbatas pencahayaan cinta-Nya bagi umat-Nya,
Allah Subhanahu Wa Ta'ala.

Rasulullah Muhammad SAW. The Everlasting Prince,
Inspiration of Every Single Inspiration.
Teladan beliau yang menjadi penuntun dalam setiap langkah.

Special thanks to :
Untuk Ayah dan Mama, Special Thank's for do'a dan dukungan yang tiada henti, semoga persembahan ini dapat menjadi salah satu tanda bakti dan wujud kasih sayang ananda.
Saudara-saudari ku tercinta : Kak Tetty, Ardi, Arlan, dan Arfan serta keponakan ku Tasya. Kasih sayang antara kita tak perlu diucapkan dengan kata-kata. Hanya perhatian kalian yang mampu membuatku tersenyum. Thank's banget untuk keceriaan yang menghiasi hari-hari, juga dukungan dan motivasi yang sangat berarti.
Agus Alfiandri, seseorang yang selalu menjadi bintang yang menemani hari-hariku, menjadi cahaya setiap langkahku dan menjadi cermin untuk diriku.
Sahabatku tersayang : Special to Dewi "Kebo", Tifa "Cacing", besok kalian nginap lagi di kamarku y....^^. Dan sahabat seperjuangan Ita, Tyas, Dini, Indah, Kiki yang selalu memberikan pelajaran-pelajaran yang berharga untuk menjalani hidup selamat berjuang sobat dimanapun kalian berada kelak.
Teman Laboratorium KSC : Mas Bowo si manusia freak "mantan manajer", Robby "wrekudara" thanks banget y sob....^^, Kisti "kizzy" ayooo.. kita KP lg d indok sm mbak indok.. haha.... =p, Arief "ndutz 1" bersyukurlah kamu jd penerus mas bow, bukannya apa-apa, biar bisa kurus aj sih.... Hahaa..... =p, Hendra "tukul" pak maintenance tiada duanya, Henry "n4bil4" bantuin install Linux... =D. Terima kasih kebersamaan yang kalian berikan. Kalian adalah keluargaku^^.
Adik-adik KSC angkatan 2007 : Ifa "new buk min", dhika "kamu jadi potografer aj sih dik, hehehe....., Yogie "Pak Ben" manusia bertanduk, Yudha

"cah mendem", Ardhi "go research go...", fian "bantuin yudha maintence sana, jgn pacaran terus.... =P". Terus belajar dan mengajar yah !!! Selesaikan penelitian nya ya...^^.

Teman-teman Lab Informatika : dari lantai 3, GMM dan SISJARKOM sampai lantai 4, PIT dan SIRKEL. Tak lupa pula buat laboran mas Fahmi.

Teman-teman kampus ku : Alma, Yana, Diska, Andi, Yuli, Rona, selamat berjuang kawan....!!! Semoga d lain hari kita dapat bertemu kembali... =)

Anak-anak KKN Ekstensi Tematik Konversi Energi Unit 30 : Rio "Pak Ketua", Rizky "Sekretaris", Vicky "Bendahara", Donny, Latif, Bintang, Pandu, Rahmat, Kutut dan Rudi. Mau ngudek2 t*ik lagi?? Hahaa.... =))

Fungsionaris LEM FTI UII : Bang Rudy "makasih ya atas kebersamaannya, maaf tidak bisa menjalankan tugas hingga akhir periode", Robert "mantan Sekretaris Umum", Ronny "Bendahara hebat, jangan korupsi y....^^", Unik, Mukhlis, Andi, Ardhy, dan anggota lainnya yang gak bias disebutkan namanya karena lupa... hehehe..... Perjuangan Adalah Pelaksanaan Kata-kata!!!.

Penghuni Pondok Shafa 1 : Fiqa, Icha, Dilla, Dwi, Danar, dan anak lantai atas lainnya. Fania, Ratih, Evi, Tessa, Affa, Lia, jadi mau pindah g?? hahahhaaa..... =p. Tak lupa pula pemilik sodiq shop, mas sodiq dan mbak gayuk..... =)

Bapak Ibu guru ku dari TK – SMA, karena ilmu dari kalian lah aku bisa melanjutkan pendidikan hingga ke perguruan tinggi.

Bapak Ibu Dosen, yang memberikan ilmunya sehingga aku bisa menyelesaikan skripsi ku ini.

Terakhir buat semua temen2 ku, sodara ku, keluarga ku, yang gak bisa aku sebutin satu per satu.. terimakasi doanya ya.... Semoga kalian selalu dalam lindungan-NYA.

MOTTO

"...Dan Tuhanmu tidak akan lalai atas segala yang kamu kerjakan".
(Q.S. An Naml 93)

"... dan kamupun akan menuai apa yang telah kamu lakukan....."
Q.S. Al Baqoroh 134)

"Sungguh, bersama kesukaran itu pasti ada kemudahan"
(Q.S. Al Insyirah 5)

"Kemenangan paling berharga dalam hidup bukanlah tidak pernah gagal, melainkan bagaimana kita bisa bangkit ssetiap kali menemui kegagalan".
(Nelson Mandela, Presiden Afrika Selatan)

"Anda akan selalu melewati kegagalan untuk mencapai kesuksesan"
(Mickey Rooney, Aktor AS)

"Tujuan utama pendidikan bukanlah ilmu pengetahuan, melainkan aksi nyata"
(Herbert Spencer, Filusuf Inggris)

"Saya belajar selama saya hidup. Batu nisan akan menjadi ijazah saya".
(Anonymous)

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamu'alaikum Wr.Wb

Alhamdulillah, segala puji bagi Allah SWT atas segala rahmat, hidayah dan inayah-Nya, sehingga penulisan laporan tugas akhir yang berjudul **Rancang Bangun *Tool* Untuk Jaringan Syaraf Tiruan (JST) Model Perceptron** dapat penulis selesaikan dengan baik.

Laporan tugas akhir ini disusun sebagai salah satu syarat guna memperoleh gelar Sarjana Teknik Informatika pada Universitas Islam Indonesia. Dan juga sebagai sarana untuk mempraktekkan secara langsung ilmu dan teori yang telah diperoleh selama menjalani masa studi di Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

Penyusunan laporan tugas akhir ini tidak lepas dari bimbingan, dukungan dan bantuan baik materiil maupun spirituil dari berbagai pihak. Oleh karena itu dalam kesempatan ini dengan segala kerendahan hati, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada :

a. Allah SWT, Tuhan bagi seluruh alam yang melimpahkan rahmat dan karuniannya sehingga penulis selalu diberi kesehatan dan kemudahan selama pembuatan tugas akhir ini.
b. Ayah dan Mama yang telah memberikan seluruh do'a dan restu, serta dorongan baik spirituil maupun materiil sehingga penulis dapat menyelesaikan studi dengan baik.
c. Kakak dan Adik-adikku tercinta yang selalu memberikan semangat dan memberikan inspirasi untuk terus maju.
d. Seluruh keluarga besar dimanapun berada. Terimakasih atas doa dan dukungannya.
e. Bapak Edy Suandi Hamid, selaku Rektor Universitas Islam Indonesia dan seluruh jajaran Rektorat Universitas Islam Indonesia.

f. Bapak Fathul Wahid, ST., M.Sc. selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
g. Bapak Yudi Prayudi, S.Si., M.Kom, selaku Ketua Jurusan Teknik Informatika.
h. Ibu DR. Sri Kusumadewi, S.Si, MT selaku dosen pembimbing yang telah memberikan pengarahan, bimbingan, serta masukan selama pelaksanaan tugas akhir dan penulisan laporan.
i. Dosen-dosen Jurusan Teknik Informatika. Terima kasih atas semua ilmu pengetahuan dan motivasi serta bantuannya.
j. Sobat-sobatku yang jauh disana dan selalu mendoakanku, terima kasih atas semuanya. Semoga kebaikan kalian selama ini dapat dibalas oleh Allah SWT. Amin.
k. Serta semua pihak yang memberikan dukungan, yang tidak dapat saya sebutkan satu per satu.

Terima kasih kepada semua pihak yang telah membantu terselesaikannya penulisan laporan tugas akhir ini Semoga Allah SWT melimpahkan rahmat dan hidayahnya dan membalas semua kebaikan kalian.

Penulis menyadari bahwa dalam penyusunan laporan tugas akhir ini masih banyak terdapat kekeliruan dan kekurangan. Untuk itu penulis menyampaikan permohonan maaf sebelumnya serta sangat diharapkan kritik dan saran yang sifatnya membangun untuk penyempurnaan di masa mendatang.

Akhir kata semoga laporan ini dapat bermanfaat bagi penulis dan semua pembaca.

Wassalamu'alaikum Wr.Wb.

Yogyakarta, 9 Juni 2010

Liza Afriyanti

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

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAKSI

Jaringan Syaraf Tiruan (JST) merupakan salah satu topik penelitian yang menarik di bidang Kecerdasan Buatan. Hal ini disebabkan karena kemampuan JST untuk meniru sifat sistem yang dimasukkan. Pada dasarnya JST mencoba meniru cara kerja otak makhluk hidup, yaitu bentuk *neuron*-nya (sel syaraf). Faktor kecerdasan dari syaraf tidak ditentukan di dalam sel tetapi terletak pada bentuk dan topologi jaringannya.

Salah satu model JST yang sering digunakan untuk pembelajaran adalah perceptron. Metode perceptron merupakan metode pembelajaran dengan pengawasan dalam sistem jaringan syaraf. Dalam merancang jaringan neuron yang perlu diperhatikan adalah banyaknya spesifikasi yang akan diidentifikasi. Jaringan neuron terdiri dari sejumlah neuron dan sejumlah masukan.

Penelitian dilakukan untuk membangun sebuah *tool* yang digunakan untuk membuat struktur Jaringan Syaraf Tiruan model perceptron. Sehingga dapat membantu *user* dalam memahami teori sekaligus dapat memahami struktur JST secara visual.

Hasil akhir dari penelitian skripsi ini adalah sebuah *tool* untuk membuat struktur Jaringan Syaraf Tiruan model perceptron. *Tool* untuk struktur Jaringan Syaraf Tiruan (JST) model perceptron dapat digunakan untuk melakukan proses pelatihan dan pengujian serta menampilkan struktur jaringan perceptron.

Kata-kunci : Jaringan Syaraf Tiruan, Perceptron, Struktur Jaringan.

1

B

2.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Komputer terpilih sebagai salah satu alternatif solusi dalam pengambilan keputusan yang tepat dan akurat. Hasil kerja sistem komputer ini diakui lebih cepat, teliti dan akurat dibandingkan dengan manusia, hal inilah yang mendorong lahirnya Kecerdasan Buatan atau *Artificial Intellegence.*

Jaringan Syaraf Tiruan (*Artificial Neural Networks*) adalah salah satu cabang ilmu dari bidang ilmu Kecerdasan Buatan. Jaringan Syaraf Tiruan (JST) merupakan topik yang bagus untuk dijadikan bahan penelitian dibidang Kecerdasan Buatan. Hal ini disebabkan karena kemampuan JST untuk meniru sifat sistem yang dimasukkan. Pada dasarnya JST mencoba meniru cara kerja otak makhluk hidup, yaitu bentuk *neuron*-nya (sel syaraf). Faktor kecerdasan dari syaraf tidak ditentukan di dalam sel tetapi terletak pada bentuk dan topologi jaringannya.

Salah satu model JST yang sering digunakan untuk pembelajaran adalah perceptron. Metode perceptron merupakan metode pembelajaran dengan pengawasan dalam sistem jaringan syaraf. Dalam merancang jaringan neuron yang perlu diperhatikan adalah banyaknya spesifikasi yang akan diidentifikasi. Jaringan neuron terdiri dari sejumlah neuron dan sejumlah masukan. [SIA05]

Salah satu *toolbox* yang sering digunakan dalam menyelesaikan model Jaringan Syaraf Tiruan (JST) adalah Matlab. Kelebihan daripada Matlab yaitu membantu seorang *user* dalam memecahkan suatu masalah khususnya dalam perceptron, dimana *software* ini menyediakan fasilitas jenis-jenis *toolbox* yang dapat mempermudah pekerjaan seorang *user*. Sedangkan kekurangannya adalah data yang dimasukan disajikan dalam bentuk kurva, sehingga *user* hanya dapat melihat hasil dari data yang dibuat tanpa mengetahui struktur jaringan pada jaringannya.

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.

1.4 Tujuan Penelitian

Tujuan yang diharapkan dari penulisan tugas akhir ini adalah membangun aplikasi untuk mengimplementasikan struktur Jaringan Syaraf Tiruan model Perceptron dengan menggunakan bahasa pemrograman Java untuk meningkatkan pemahaman user mengenai Jaringan Syaraf Tiruan model Perceptron.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini antara lain sebagai media pembelajaran bagi *user* yang dapat membantu dalam memahami struktur Jaringan Syaraf Tiruan model Perceptron.

1.6 Metodologi Penelitian

Dalam suatu penelitian, diperlukan metodologi agar data yang diperlukan dalam penelitian sesuai dengan data yang ada di lapangan. Metodologi penelitian adalah ilmu mengenai jalan yang dilewati untuk mencapai pemahaman [MAR07]. Metodologi penelitian yang akan dilakukan yaitu :

1.6.1 Studi Pendahuluan

Dalam studi pendahuluan, yang menjadi sasaran pokok adalah melihat bagaimana variabel-variabel yang akan dipelajari. Pada objek penelitian, variabel-variabel tersebut dipelajari melalui dokumentasi yang ada, seperti buku-buku referensi mengenai JST model perceptron, menggunakan berbagai macam literatur yang berhubungan dengan JST dan internet, selanjutnya sekaligus dipilih sampel studi.

1.6.2 Pengumpulan Data

Metode pengumpulan data yang digunakan dalam penelitian tugas akhir ini adalah metode kepustakaan, yaitu mengumpulkan data yang diperlukan dari buku-buku referensi yang relevan dengan permasalahan yang dihadapi, serta data-data yang didapatkan dari internet.

1.6.3 Perancangan Model

Setelah dilakukan studi pendahuluan, dapat diketahui variabel mana yang sesuai antara teori dan kenyataan serta variabel mana yang tidak sesuai. Variabel-variabel yang sesuai selanjutnya digunakan untuk pembuatan model dan program komputer yang menyangkut hal-hal penerapan masukan dan keluaran.

1.6.4 Metode Pengembangan Perangkat Lunak

Setelah pengumpulan data, diperlukan metode untuk perancangan dan pembuatan perangkat lunak. Metode pembuatan perangkat lunak yang digunakan pada tugas akhir ini adalah :

a. Analisis Data

   Analisis data dilakukan untuk mengolah data yang sudah didapat dan mengelompokkan data sesuai dengan kebutuhan perancangan. Analisis data yang akan dilakukan adalah pembuatan neuron, analisis jaringan, membangun struktur Jaringan Syaraf Tiruan serta membangun algoritma pembelajaran.

b. Desain

   Tahap ini merupakan tahap penerjemahan dari keperluan data yang telah dianalisis ke dalam bentuk antarmuka yang mudah dimengerti oleh *user*. Desain yang akan dilakukan adalah desain struktur Jaringan Syaraf Tituan dan desain visual jaringan.

c. Implementasi (*coding*).

   Implementasi pada perangkat lunak (*software*) menggunakan teknologi Java.

d. Proses pengujian sistem

   Proses pengujian dilakukan terhadap perangkat lunak yang telah dibangun.

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.

BAB II
PERCEPTRON

2.1 Teori Jaringan Syaraf Tiruan (JST)

2.1.1 Definisi Jaringan Syaraf Tiruan (JST)

Saat ini bidang kecerdasan buatan dalam usahanya menirukan intelegensi manusia, belum mengadakan pendekatan dalam bentuk fisiknya melainkan dari sisi yang lain. Pertama-tama diadakan studi mengenai teori dasar mekanisme proses terjadinya intelegensi. Bidang ini disebut *'Cognitive Science'*. Dari teori dasar ini dibuatlah suatu model untuk disimulasikan pada komputer, dan dalam perkembangannya yang lebih lanjut dikenal berbagai sistem kecerdasan buatan yang salah satunya adalah jaringan syaraf tiruan. Dibandingkan dengan bidang ilmu yang lain, jaringan syaraf tiruan relatif masih baru. Sejumlah literatur menganggap bahwa konsep jaringan syaraf tiruan bermula pada makalah Waffen McCulloch dan Walter Pitts pada tahun 1943. Dalam makalah tersebut mereka mencoba untuk memformulasikan model matematis sel-sel otak. Metode yang dikembangkan berdasarkan sistem syaraf biologi ini, merupakan suatu langkah maju dalam industri komputer. [NAS09]

Elemen yang paling mendasar dari jaringan syaraf adalah sel syaraf. Sel-sel syaraf inilah membentuk bagian kesadaran manusia yang meliputi beberapa kemampuan umum. Pada dasarnya sel syaraf biologi menerima masukan dari sumber yang lain dan mengkombinasikannya dengan beberapa cara, melaksanakan suatu operasi yang non-linear untuk mendapatkan hasil dan kemudian mengeluarkan hasil akhir tersebut.

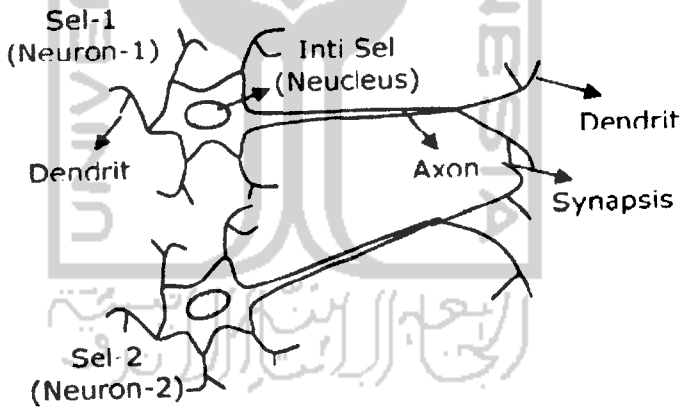
Jaringan syaraf tiruan dapat dibayangkan seperti otak buatan yang dapat berpikir seperti manusia dan juga sepandai manusia dalam menyimpulkan sesuatu dari potongan-potongan informasi yang diterima. Khayalan manusia tersebut mendorong para peneliti untuk mewujudkannya. Komputer diusahankan agar bisa berpikir sama seperti cara berpikir manusia. Caranya adalah dengan melakukan

peniruan terhadap aktivitas-aktivitas yang terjadi di dalam sebuah jaringan syaraf biologis. [KUS03]

Salah satu contoh pengambilan ide dari jaringan syaraf biologis adalah adanya elemen-elemen pemrosesan pada jaringan syaraf tiruan yang saling terhubung dan beroperasi secara paralel. Ini meniru jaringan syaraf biologis yang tersusun dari sel-sel syaraf (neuron). Cara kerja dari elemen-elemen pemrosesan jaringan syaraf tiruan juga sama seperti meng-*encode* informasi yang diterimanya.

Jaringan syaraf biologis merupakan kumpulan sel-sel syaraf (neuron). Neuron mempunyai tugas mengolah informasi. Komponen-komponen utama dari sebuah neuron dapat dikelompokkan menjadi tiga bagian, yaitu:

1. *Dendrit*. Dendrit bertugas untuk menerima informasi.
2. *Badan sel (soma)*. Badan sel berfungsi sebagai tempat pengolahan informasi.
3. *Akson (neurit)*. Akson mengirimkan impuls-impuls ke sel syaraf lainnya.

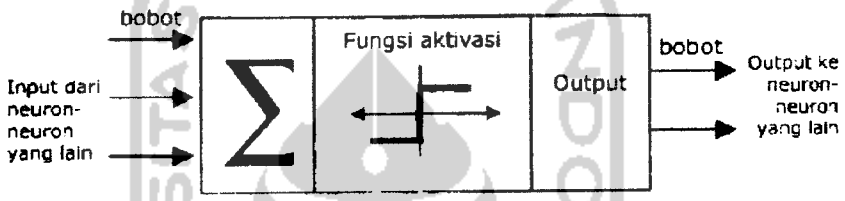


Gambar 2. 1 Sel Syaraf Biologis [KUS03]

Pada gambar 2.1 dapat diperhatikan sebuah neuron menerima impuls-impuls sinyal dari neuron yang lain melalui dendrit dan mengirimkan sinyal yang dibangkitkan oleh badan sel malalui akson akson dari sel syaraf biologis ini bercabang-cabang dan berhubungan dengan dendrit dari sel syaraf lainnya dengan cara mengirimkan impuls melaui *sinapsis*. Sinapsis adalah unit fungsional antara dua buah sel syaraf, katakanlah A dan B, di mana yang satu adalah serabut akson dari neuron A dan satunya lagi dendrit dari neuron B. [KUS03]

2.1.2 Cara Kerja Komponen Jaringan Syaraf Tiruan

Ada beberapa tipe jaringan syaraf tiruan, tetapi hampir semuanya memiliki komponen yang sama. Sama halnya seperti otak manusia, jaringan syaraf juga terdiri dari beberapa neuron, dan ada hubungan antara neuron tersebut. Neuron-neuron tersebut akan mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju ke neuron-neuron yang lain. Pada JST hubungan ini dikenal dengan nama bobot. Informasi tersebut disimpan pada suatu nilai tertentu pada bobot tersebut. [KUS03]

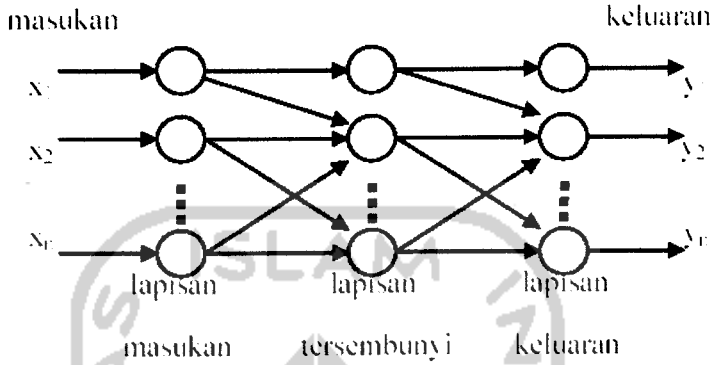


Gambar 2. 2 Struktur Neuron Jaringan Syaraf Tiruan [KUS03]

Informasi (disebut dengan *input*) akan dikirim ke neuron dengan bobot kedatangan tertentu. *Input* ini akan diproses oleh suatu fungsi perambatan yang akan menjumlahkan nilai-nilai semua bobot yang datang. Hasil penjumlahan ini kemudian akan dibandingkan dengan suatu nilai ambang *(threshold)* tertentu melalui fungsi aktivasi setiap neuron. Apabila *input* tersebut melewati suatu nilai ambang tertentu, maka neuron tersebut akan diaktifkan, tetapi jika tidak maka neuron tersebut tidak akan diaktifkan. Apabila neuron tersebut diaktifkan, maka neuron tersebut akan mengirimkan *output* melalui bobot-bobot *output*-nya ke semua neuron yang berhubungan dengannya. Hal ini dilakukan secara terus menerus. [KUS03]

Pada jaringan syaraf, neuron-neuron akan dikumpulkan dalam lapisan-lapisan (*layer*) yang disebut dengan lapisan neuron (*neuron layers*). Biasanya neuron-neuron pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya (kecuali lapisan *input* dan lapisan *output*). Informasi yang diberikan pada jaringan syaraf akan dirambatkan dari lapisan ke lapisan,

mulai dari lapisan *input* sampai ke lapisan *output* melalui lapisan yang lainnya, yang sering dikenal dengan nama lapisan tersembunyi (*hidden layer*), tergantung pada algoritma pembelajarannya, bisa jadi informasi tersebut akan dirambatkan secara mundur pada jaringan.



Gambar 2. 3 Jaringan Syaraf Dengan Tiga Lapisan [KUS03]

Beberapa jaringan syaraf ada juga yang tidak memiliki lapisan tersembunyi, dan ada juga yang neuron-neuronnya disusun dalam bentuk matriks.

2.1.3 Konsep Belajar Jaringan Syaraf Tiruan

Ciri utama yang dimiliki oleh sistem jaringan syaraf tiruan adalah kemampuan untuk belajar. Agar berfungsi seperti yang diinginkan, jaringan tidak diprogram seperti yang dilakukan pada sistem komputer sekarang ini, melainkan harus diajari.

Berdasarkan fungsi masukan keluarannya, fungsi jaringan syaraf tiruan ditentukan oleh parameternya (bobot-bobot koneksi). Untuk kasus yang diketahui fungsi pemetaannya, bobot-bobot tersebut dapat berharga tetap dan ditentukan pada waktu perancangan. Tetapi pada kebanyakan kasus, parameter jaringan yang cocok belum diketahui, dan jaringan harus mencari sendiri besarnya bobot tersebut atau ditentukan secara acak.

Suatu proses penyesuaian parameter secara berurutan dilakukan, dengan tujuan mendekati fungsi yang diinginkan. Proses penyesuaian parameter inilah yang disebut dengan proses belajar dalam sistem jaringan syaraf tiruan. Proses belajar dikategorikan dalam dua jenis :

1. Dengan pengawasan (*supervised learning*),
2. Tanpa pengawasan (*unsupervised learning*).

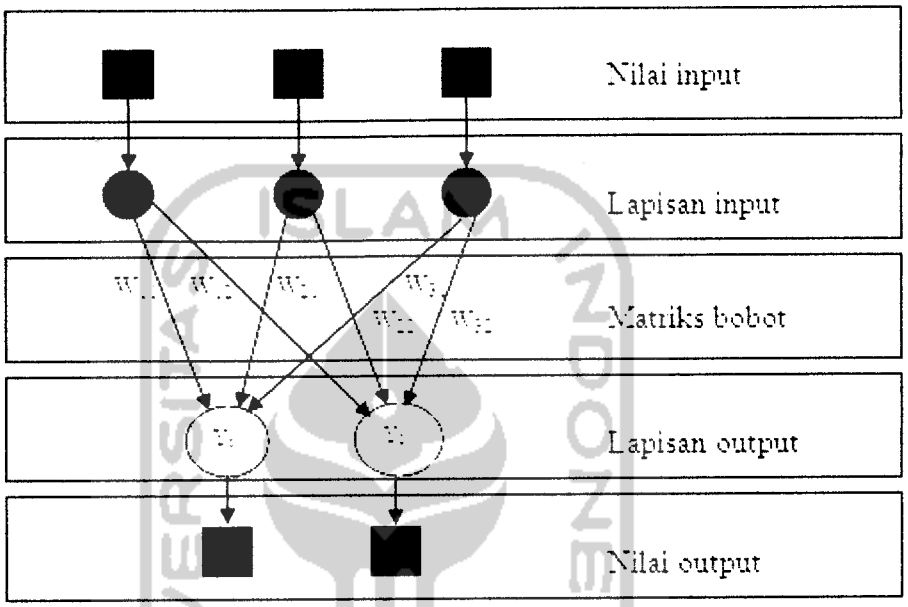
Proses belajar dengan pengawasan memerlukan keluaran target atau jawaban yang diperlukan dalam proses belajar sebagai dasar penghubung bobot. Jaringan diajar untuk menyelesaikan persoalan-persoalan yang terdapat dalam paket belajarnya. Selama belajar apabila jaringan mengeluarkan jawaban yang salah, maka besar kesalahan dapat dicari, yaitu beda keluaran aktual dan acuannya. Sedangkan dalam belajar tanpa pengawasan, jaringan akan mengubah bobot-bobotnya, sebagai tanggapan terhadap masukan, tanpa memerlukan keluaran acuan. [KUS03]

2.1.4 Arsitektur Jaringan Syaraf Tiruan

Dalam jaringan syaraf, neuron-neuron dikelompokan dalam lapisan-lapisan yang umumnya setiap lapisan yang sama akan memiliki keadaan yang sama pula. Apabila neuron-neuron dalam suatu lapisan (misalkan lapisan tersembunyi) akan dihubungkan dengan neuron-neuron pada lapisan lain (misalkan lapisan *output*), maka setiap neuron pada lapisan tersebut (misalkan lapisan tersembunyi) juga harus dihubungkan pada setiap neuron pada lapisan yang lainnya (misalkan lapisan *output*). Ada beberapa arsitektur jaringan syaraf, antara lain : [KUS03]

1. Jaringan lapis tunggal (*single layer net).*

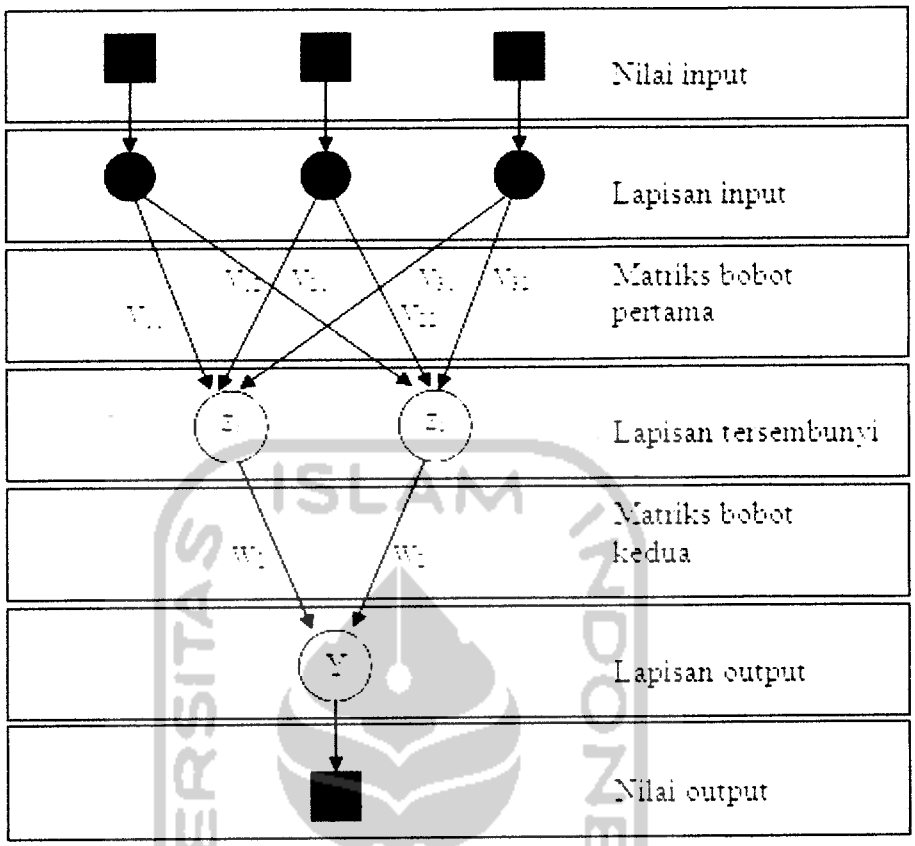
   Jaringan yang memiliki arsitektur ini hanya memiliki satu buah lapisan bobot koneksi. Jaringan lapisan tunggal terdiri dari unit-unit input yang menerima sinyal dari dunia luar, dan unit-unit output dimana kita bisa membaca respons dari jaringan syaraf tiruan tersebut.



Gambar 2. 4 Single Layer Net [KUS03]

2. Jaringan multilapis *(multi layer net).*

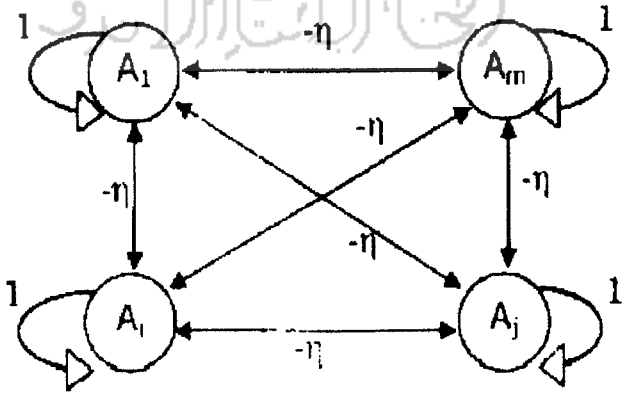
   Merupakan jaringan dengan satu atau lebih lapisan tersembunyi dan memiliki kemampuan lebih dalam memecahkan masalah bila dibandingkan dengan *single layer net*, namun pelatihannya lebih rumit.



Gambar 2. 5 Multi Layer Net [KUS03]

3. Jaringan kompetitif *(competitive net)*.

   Pada jaringan ini sekumpulan neuron bersaing untuk mendapatkan hak menjadi aktif. Dari gambar berikut bobot-bobotnya ditunjukan oleh $-\eta$.



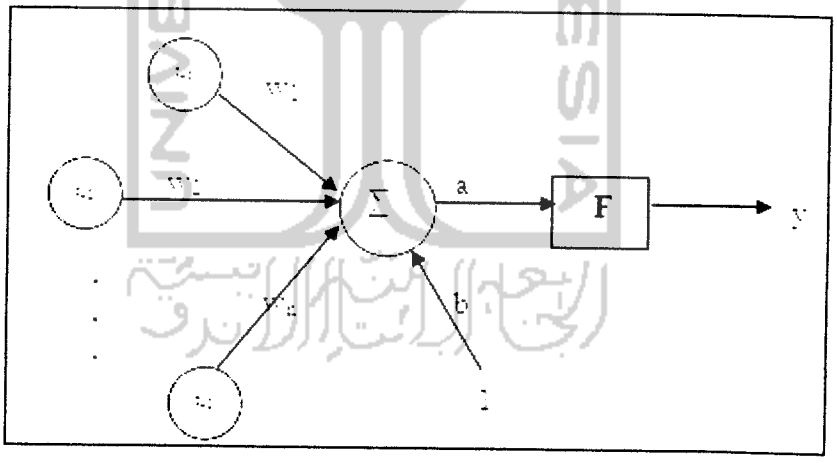
Gambar 2. 6 Competitive Net [KUS03]

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

2.2 Jaringan Syaraf Tiruan Model Perceptron

Model jaringan perceptron ditemukan pertama kali oleh Rosenbatt (1962) dan Minsky – Papert (1969). Model tersebut merupakan model yang memiliki aplikasi dan pelatihan yang paling baik pada era tersebut. Perceptron merupakan salah satu bentuk jaringan sederhana, perceptron biasanya digunakan untuk mengklasifikasikan suatu pola tipe tertentu yang sering dikenal dengan pemisahan secara linear. Pada dasarnya perceptron pada jaringan syaraf dengan satu lapisan memiliki bobot yang dapat diatur. Dapat digunakan dalam kasus untuk mengenali fungsi logika "dan" dengan masukan dan keluaran bipolar. [SIA05]

Arsitektur jaringan perceptron mirip dengan arsitektur jaringan Hebb. Jaringan terdiri dari beberapa unit masukan (ditambah sebuah bias), dan memiliki sebuah unit keluaran. Hanya saja fungsi aktivasi bukan merupakan fungsi biner (atau bipolar), tetapi memiliki kemungkinan nilai -1, 0 atau 1.

Algoritma yang digunakan oleh aturan perceptron ini akan mengatur parameter-parameter bebasnya melalu proses pembelajaran. Fungsi aktivasinya dibuat sedemikian rupa sehingga terjadi pembatasan antara daerah positif dan negatif.

Perceptron memiliki kemampuan lebih baik daripada algoritma pembelajaran *Hebb*. Perceptron memiliki karakteristik sebagai berikut :

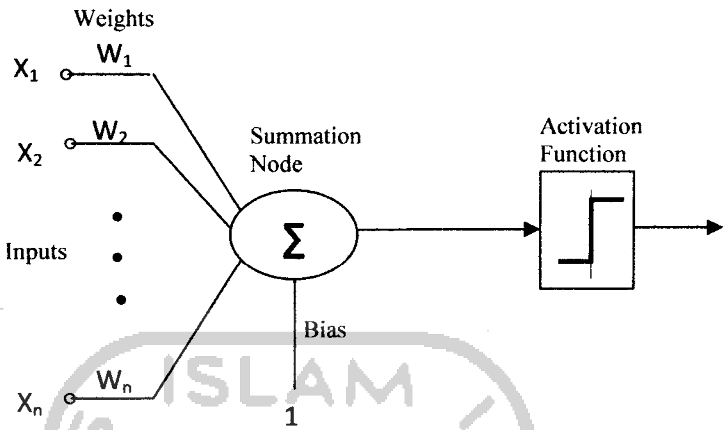
- Jaringan lapis tunggal
- Fungsi aktivasi : Fungsi tanga bipolar dengan suatu nilai batas tetap (q)

$$f(net) = \begin{cases} 1 & jika \quad net > \theta \\ 0 & jika \quad -\theta \le net \le \theta \\ 1 & jika \quad net < \theta \end{cases} \qquad \text{..... (2.1)}$$

- Apabila terjadi kesalahan untuk pola masukan pelatihan bobot akan disesuaikan dengan formula :

$$w_i(new) = w_i(old) + \alpha t x_i \qquad \text{..... (2.2)}$$

- T adalah nilai target +1 atau -1



Gambar 2. 8 Target

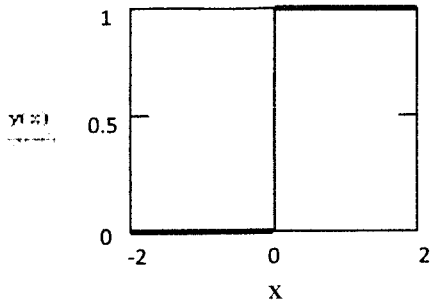
$a = W_1X_1 + W_2X_2 + \ldots + W_NX_N + \text{Bias}$

Output = Threshold (a)

$$\text{Dimana}\quad \text{Threshold (a)} = \begin{cases} -1, \text{ untuk semua } a \leq 0 \\ 1, \text{ untuk semua } a > 0 \end{cases} \qquad \text{(2.3)}$$

Perceptron merupakan pengembangan aturan *Hebb* dimana terdapat tiga bagian yaitu unit sensor (dengan aktivasi biner), unit-asociator (dengan aktivasi biner), dan unit respon (dengan aktivasi bipolar). Bobot dari unit sensor tetap, sedangkan unit asosiator berubah. Fungsi aktivasinya nilai bobot unit-asociator diperbaharui selama masih ada *error* berdasarkan, a adalah laju pembelajaran dan t adalah target keluaran (±1). Anggota kelas berespon 1 sedangkan bukan anggota kelas terespon -1. Dan fungsi aktifasi yang digunakan adalah *step function* (undak biner) untuk mengkonversi suatu variabel yang bernilai kontinue ke suatu *output* biner (0 atau 1), seperti pada gambar berikut ini :

$$y(x) := \left| \begin{array}{l} \text{temp} \leftarrow 1, \text{ jika } x > 0 \\ \text{temp} \leftarrow 0, \text{ otherwise} \end{array} \right. \qquad \text{(2.4)}$$



Gambar 2. 9 Fungsi Aktivasi Undak Biner JST

2.2.1 Algortima Pelatihan Perceptron

Misalkan

s adalah vektor masukan dan *t* adalah target keluaran

α adalah laju pemahaman (*learning rate*) yang ditentukan

θ adalah *threshold* yang ditentukan

Algoritma pelatihan perceptron adalah sebagai berikut :

1. Inisialisasi semua bobot dan bias (umumnya $w_i = b = 0$)

   Tentukan laju pemahaman (= α). Untuk penyederhanaa, biasanya α diberi nilai = 1
2. Selama ada elemen vektor masukan yang respon unit keluarannya tidak sama dengan target, lakukan :
   a. Set aktivasi unit masukan $x_i = s_i$ (i=1, …, n)
   b. Hitung respon unit keluaran :

$$net = \sum_i x_i w_i - b \qquad \text{..... (2.5)}$$

$$f(net) = \begin{cases} 1 & jika \quad net > \theta \\ 0 & jika \quad -\theta \le net \le \theta \\ -1 & jika \quad net < -\theta \end{cases} \qquad \text{..... (2.6)}$$

   c. Perbaiki bobot pola yang mengandung kesalahan ($y \neq t$) menurut persamaan :

   w_i (baru) = w_i (lama) + Δw (i=1, …, n) dengan $\Delta w = \alpha\, t\, x_i$ ..... (2.7)

   b (baru) = b (lama) + Δb dengan $\Delta b = \alpha\, t$ ..... (2.8)

Ada beberapa hal yang perlu diperhatikan dalam algoritma tersebut :

a. Iterasi dilakukan terus hingga semua pola memiliki keluaran jaringan yang sama dengan targetnya (jaringan sudah memahami pola). Iterasi tidak berhenti setelah semua pola dimasukkan seperti yang terjadi pada model Hebb.
b. Pada langkah 2(c), perubahan bobot hanya dilakukan pada pola yang mengandung kesalahan (keluaran jaringan $\neq$ target). Perubahan tersebut merupakan hasil kali unit masukan dengan target dan laju pemahaman. Perubahan bobot hanya akan terjadi kalau unit masukan $\neq 0$.
c. Kecepatan iterasi ditentukan pula oleh laju pemahaman ($=\alpha$ dengan $0 \leq \alpha \leq 1$) yang dipakai. Semakin besar harga α , semakin sedikit iterasi yang diperlukan. Akan tetapi jika α terlalu besar, maka akan merusak pola yang sudah benar sehingga pemahaman menjadi sedikit lambat.

Algoritma pelatihan perceptron lebih baik dibandingkan model Hebb karena :

1. Setiap kali sebuah pola dimasukkan, hasil kekurangan jaringan dibandingkan dengan target yang sesungguhnya. Jika terdapat perbedaan, maka bobot akan dimodifikasi. Jadi tidak semua bobot selalu dimodifikasi dalam setiap iterasinya.
2. Modifikasi bobot tidak hanya ditentukan oleh perkalian antara target dengan masukan, tapi juga melibatkan suatu laju pemahaman (*learning rate*) yang besarnya bisa diatur.
3. Pelatihan 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). Satu siklus pelatihan yang melibatkan semua pola disebut *epoch*. Dalam jaringan Hebb, pelatihan hanya dilakukan dalam satu epoch saja. Teorema konvergensi perceptron menyatakan bahwa apabila ada bobot yang tepat, maka proses pelatihan akan konvergen ke bobot yang tepat tersebut.

BAB III
MODEL SISTEM

Dalam penelitian ini akan dibangun sebuah sistem perangkat lunak untuk *tool* membangun struktur Jaringan Syaraf Tiruan (JST) model perceptron. Aplikasi yang dibangun merupakan aplikasi yang berbasis *desktop*. Model *tool* ini merupakan *tools* atau alat untuk membangun struktur Jaringan Syaraf Tiruan (JST) model perceptron.

Gambaran umum perangkat lunak *tool* untuk struktur Jaringan Syaraf Tiruan (JST) model perceptron dapat dilihat pada gambar 3.1.

Gambar 3. 1 Gambaran Umum Sistem

Dari gambar 3.1 diatas dapat dibagi menjadi beberapa bagian, yaitu instalasi sistem, arsitektur jaringan, *setting* data dan terakhir sistem dapat digunakan untuk membangun struktur jaringan melalui proses pengujian dan pelatihan.

3.1 Proses Instalasi Sistem

Untuk dapat menggunakan *tool* untuk struktur Jaringan Syaraf Tiruan (JST) model perceptron, maka sistem harus di-*install* terlebih dahulu. Pada tahap ini terdapat langkah-langkah yang harus dilakukan, antara lain :

1. *Install* java

   Aplikasi *desktop tool* ini memiliki ekstensi .jar yang artinya bahwa *tool* ini dikembangkan dengan bahasa pemrograman java, sehingga untuk menjalankannya dibutuhkan aplikasi java jre. Untuk menginstal jre dapat dilakukan dengan cara menginstal aplikasi seperti menginstal aplikasi biasa.

2. *Copy file* .jar

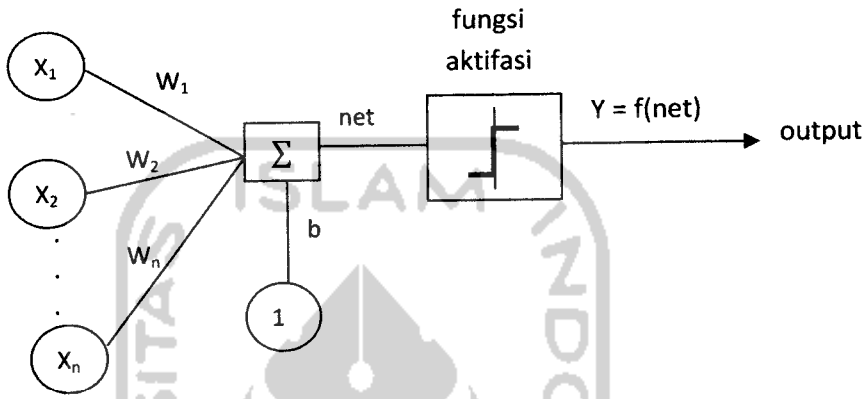
   Setelah jre terinstal maka langkah selanjutkan men*copy*-kan *file* .jar ke komputer. Setelah itu aplikasi dapat dijalankan dengan *double-klik* pada *file*.

Setelah melakukan proses di atas, maka sistem telah ter-*install* sepenuhnya dan siap untuk membangun struktur Jaringan Syaraf Tiruan (JST) model perceptron.

3.2 Proses Pembuatan Arsitektur Jaringan

Pada proses pembuatan arsitektur jaringan, user harus mempunyai rancangan data yang akan digunakan untuk membangun sebuah struktur jaringan. Data (*input*) yang dibutuhkan seperti alpha, bobot, *threshold*, bias, jumlah variabel input, nilai variabel input dan target (*output*) yang nantinya akan digunakan dalam proses pelatihan dan pengujian sistem.

Pada gambar 3.2 menjelaskan bagaimana struktur akan dibangun berdasarkan *input* data yang akan dimasukan oleh *user*. *User* akan menentukan jumlah variabel input (X_1, X_2, …, X_n) yang akan dibangun, setelah itu *user* harus memasukan nilai variabel input berupa bilangan biner ataupun bipolar.



Gambar 3. 2 Arsitektur JST Model Perceptron

Kemudian *user* memasukkan nilai bobot (umumnya nilai bobot adalah 0) dimana sistem akan memperbaharui nilai bobot selama masih ada *error* berdasarkan α adalah laju pembelajaran dan t adalah target keluaran (± 1). Fungsi aktifasi mempunyai kemungkinan nilai -1, 0, dan 1. Untuk nilai *threshold* θ yang ditentukan adalah berdasarkan pada persamaan 3.2.

$$\text{Net} = \sum_i \quad x_i w_i + b \qquad \text{.....(3.1)}$$

$$f(net) = \begin{cases} 1 & jika \quad net > \theta \\ 0 & jika \quad -\theta \le net \le \theta \\ -1 & jika \quad net < -\theta \end{cases} \qquad \text{..... (3.2)}$$

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)

Langkah 4 : Hitung respon untuk unit output :

$$net = \sum_i x_i w_i + b \quad \text{..... (3.3)}$$

$$f(net) = \begin{cases} 1 & jika \quad net > \theta \\ 0 & jika \quad \theta \geq net \geq -\theta \\ -1 & jika \quad net < -\theta \end{cases} \quad \text{..... (3.4)}$$

Langkah 5 : Perbaiki bobot dan bias pola jika terjadi kesalahan, $y \neq t$. Jika pada setiap epoh diketahui bahwa keluaran jaringan tidak sama dengan target yang diinginkan, maka bobot harus di ubah menggunakan rumus :

$\Delta w_i = \alpha\, t\, x_i = t\, x_i$ (karena $\alpha = 1$). Bobot baru = bobot(lama) + Δw_i

Langkah 6 : Test kondisi berhenti, jika tidak terjadi perubahan bobot pada epoh tersebut maka kondisi berhenti TRUE, namun jika masih terjadi perubahan maka kondisi berhenti FALSE

3.4.2 Proses Pengujian

Proses pengujian merupakan tahap penyesuaian terhadap bobot yang telah terbentuk pada proses pelatihan. Algoritma untuk proses pengujian adalah sebagai berikut :

Langkah 0 : Ambil bobot dari hasil pembelajaran,

Langkah 1 : Untuk setiap vektor x, lakukan langkah 2 – 4,

Langkah 2 : Set nilai aktivasi dari unit masukan, $x_i = s_i$; i=1,….,n,

Langkah 3 : Hitung total masukan ke unit keluaran, Net = $\sum_i \; x_i w_i + b$,

Langkah 4 : Gunakan fungsi aktivasi, Y = f(net).

BAB IV
PENGEMBANGAN SISTEM

4.1 Analisis Kebutuhan Sistem

4.1.1 Analisis Kebutuhan Masukan

Analisis kebutuhan masukan dibutuhkan untuk membangun Jaringan Syaraf Tiruan model Perceptron. *Tool* untuk struktur JST model perceptron hanya memiliki satu level akses, yaitu pengguna. Kebutuhan masukan dari *user* dibutuhkan untuk menentukan struktur jaringan. Masukan yang dibutuhkan adalah:

a. Jumlah variabel *input*.
b. Nilai variabel *input*.
c. Bobot awal.
d. Alpha.
e. *Threshold*.
f. Maksimum epoh.
g. Target (*output*).

4.1.2 Analisis Kebutuhan Keluaran

Analisis kebutuhan keluaran *tool* untuk JST model perceptron yaitu hasil perhitungan proses pelatihan dan pengujian serta gambar struktur Jaringan Syaraf Tiruan (JST) model perceptron yang dapat membantu *user* dalam memahami JST model perceptron.

4.1.3 Analisis Kebutuhan Proses

Kebutuhan proses dalam pembuatan *tool* untuk struktur Jaringan Syaraf Tiruan (JST) model perceptron antara lain proses *input* data untuk pembentukan jaringan. Kebutuhan proses *tool* untuk JST model perceptron adalah sebagai berikut :

a. Proses *input* data ke sistem. *Input* berupa parameter JST model perceptron seperti jumlah variabel *input*, nilai variabel *input*, bobot, alpha (*learning rate)*, *threshold*, maksimum epoh dan target (*output*).
b. Proses perhitungan data dan pembentukan jaringan. Setelah semua data dimasukkan kemudian diproses untuk menghasilkan perhitungan perceptron dan struktur jaringan dapat divisualisasikan ke bidang yang ada.

4.1.4 Analisis Kebutuhan Use Case

Use case adalah proses-proses yang terjadi dalam suatu sistem. *Use case* menggambarkan bagaimana seseorang akan menggunakan/memanfaatkan sistem. Kebutuhan *use case* yang diperlukan pada *tool* untuk JST model perceptron dapat dilihat pada tabel 4.1 dibawah ini.

Tabel 4. 1 Tabel Kebutuhan *Use Case*

No	Nama Use Case	Requirement	Aktor
1.	Manajemen *input* data awal	*User* dapat memasukkan data seperti jumlah variabel *input*, nilai variabel *input*, bobot, alpha (*learning rate)*, *threshold*, maksimum epoh dan target (*output*).	*User* (pengguna)
2.	Lihat *about*	*User* dapat melihat halaman *About* yang berisi informasi mengenai program dan *programmer*.	*User* (pengguna)
3.	Lihat *help*	*User* juga dapat melihat halaman *Help* untuk melihat panduan penggunaan *tool* untuk JST model perceptron.	*User* (pengguna)

4.	Proses pelatihan	*User* dapat melakukan proses pelatihan perceptron setelah melakukan memasukkan data.	*User* (pengguna)
5.	*Input* nilai variabel uji	*User* dapat memasukkan data nilai variabel *input* uji.	*User* (pengguna)
6.	Proses pengujian	*User* dapat melakukan proses pengujian setelah melakukan proses pelatihan.	*User* (pengguna)
7.	Membuat struktur jaringan	*User* dapat membuat struktur jaringan setelah memasukkan kebutuhan *input* data seperti jumlah variabel *input*, nilai variabel *input*, bobot, alpha (*learning rate)*, *threshold*, maksimum epoh dan target (*output*).	*User* (pengguna)
8.	Simpan hasil pelatihan dan pengujian	*User* dapat melakukan penyimpanan file hasil perhitungan pelatihan dan pengujian perceptron.	*User* (pengguna)

4.1.5 Kebutuhan Antar Muka

Perancangan antar muka dilakukan dengan menggunakan komponen Java (Swing dan AWT) dan Netbeans 6.8. Penggunaan komponen Java Swing, AWT dan Netbeans 6.8 merupakan pilihan yang tepat untuk mengimplementasikan perangkat lunak, dengan tampilan yang indah dan memudahkan pengguna untuk menggunakan sistem, dan sifat orientasi objek dan modularisasi dari Java juga membuat *programmer* lebih mudah untuk memisahkan tampilan dari kode, sehingga memudahkan pengembangan perangkat lunak. Kebutuhan antar muka yang diperlukan antara lain :

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

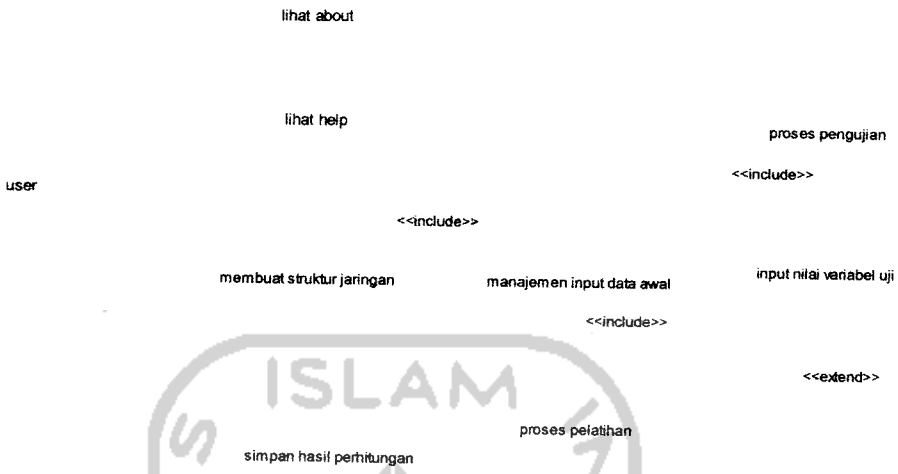
Perangkat keras komputer yang dibutuhkan adalah perangkat keras yang dapat mendukung perangkat lunak yang memiliki kemampuan atau tampilan grafis yang cukup baik. Perangkat keras minimum yang digunakan pada *tool* untuk JST model perceptron adalah :

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.

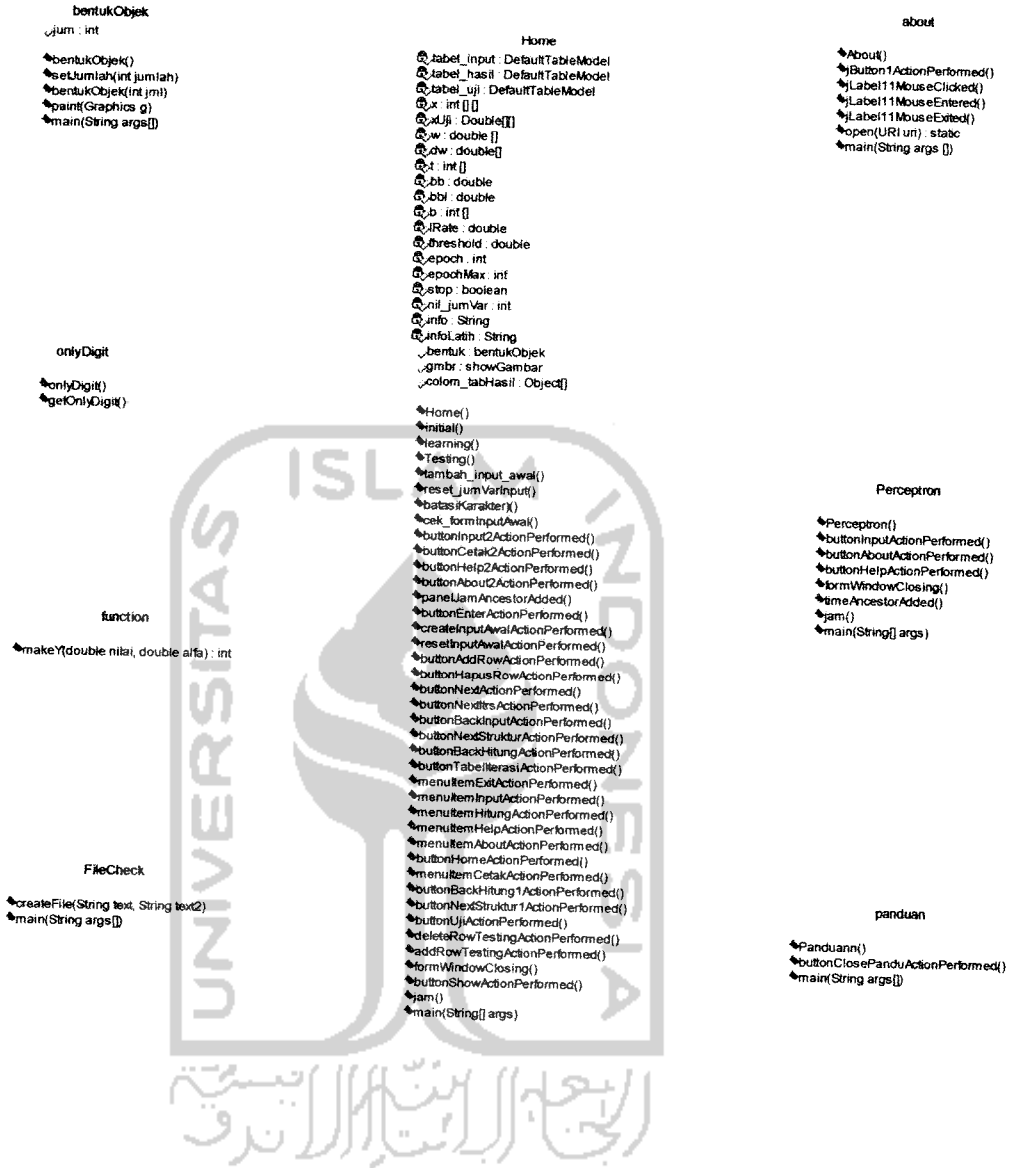


Gambar 4. 1 *Use Case Diagram Tool* Untuk JST Model Perceptron

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

4.2.3 *Sequence Diagram*

Sequence diagram digunakan untuk menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus berasosiasi dengan *use case*. *Sequence diagram* juga dapat memperlihatkan tahap demi tahap proses yang seharusnya terjadi untuk menghasilkan sesuatu di dalam *use case*.

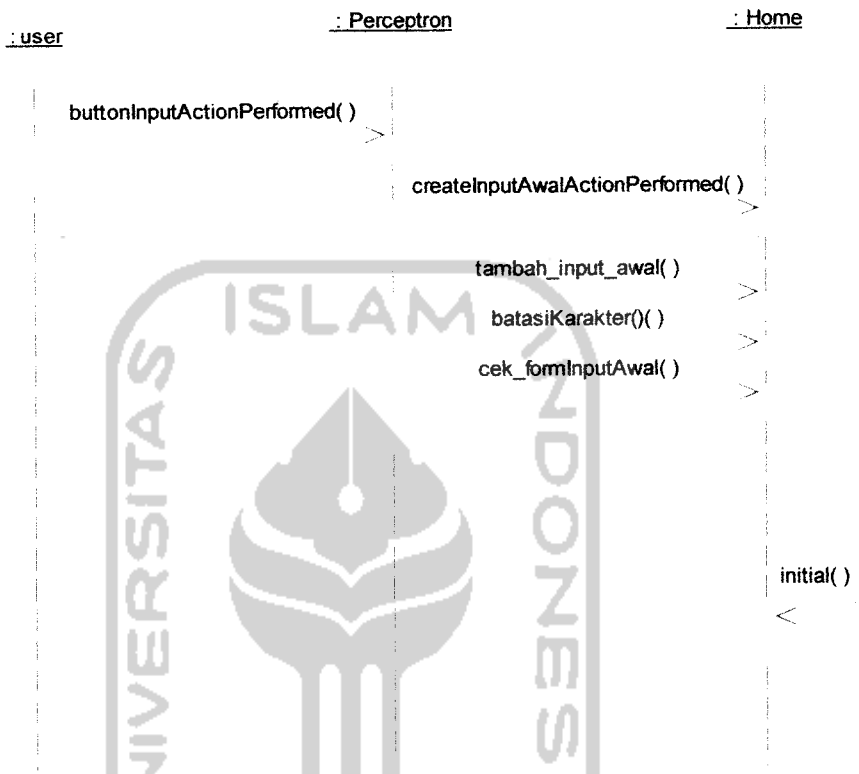
Pada bagian ini akan dijelaskan *sequence diagram* dari rancang bangun *tool* untuk JST model perceptron. Dalam *sequence diagram* ini menggambarkan interaksi antar objek pada sistem secara berurutan.

a. *Sequence Diagram* Manajemen *Input* Data Awal

Sequence Diagram manajemen *input* data awal menunjukkan interaksi *user* dengan perangkat lunak untuk melakukan *input* data awal pada *tool* untuk JST model perceptron. Objek yang berkaitan dengan *sequence* ini adalah sebagai berikut :

Aktor	:	*User*
ClassBoundary	:	*Home*
Clas Control	:	create*Input*AwalActionPerformed()
Clas Entity	:	Tab *Input*Data
Keterangan	:	1. *User* mengawali *sequence* ini dengan memanggil method *buttonInputActionPerformed* ().
		2. Sistem melakukan instansiasi dan memanggil *method createInputAwalActionPerformed*() untuk memproses *input* dari *user*.

Sequence diagram manajemen *input* data awal dapat dilihat pada gambar 4.3.



Gambar 4. 3 *Sequence Diagram* Manajemen *Input* Data Awal

b. *Sequence Diagram* Proses Pelatihan

Sequence Diagram proses pelatihan menunjukkan interaksi *user* dengan perangkat lunak untuk melakukan pelatihan perceptron berdasarkan *input* data oleh *user*. Objek yang berkaitan dengan *sequence* ini adalah sebagai berikut :

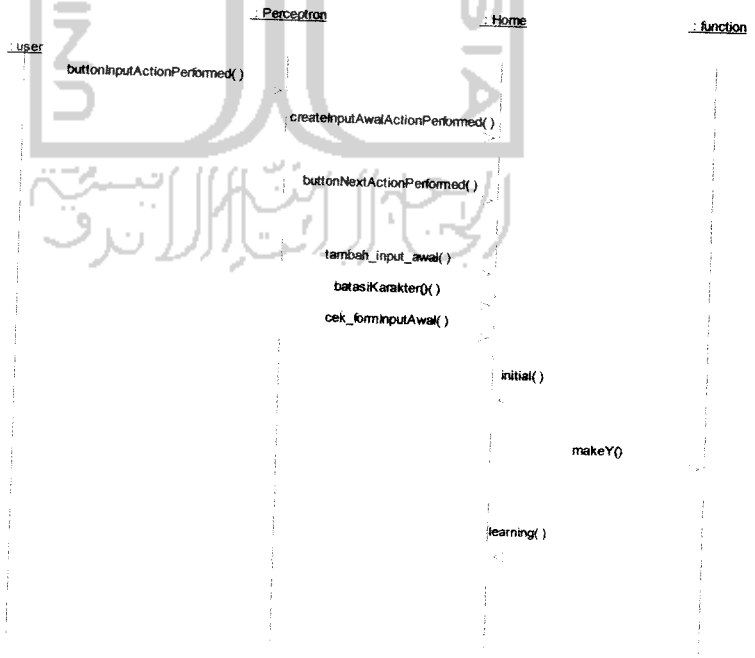
Aktor	:	*User*
Class Boundary	:	*Home*
Class Control	:	*buttonNextActionPerformed* ()

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.



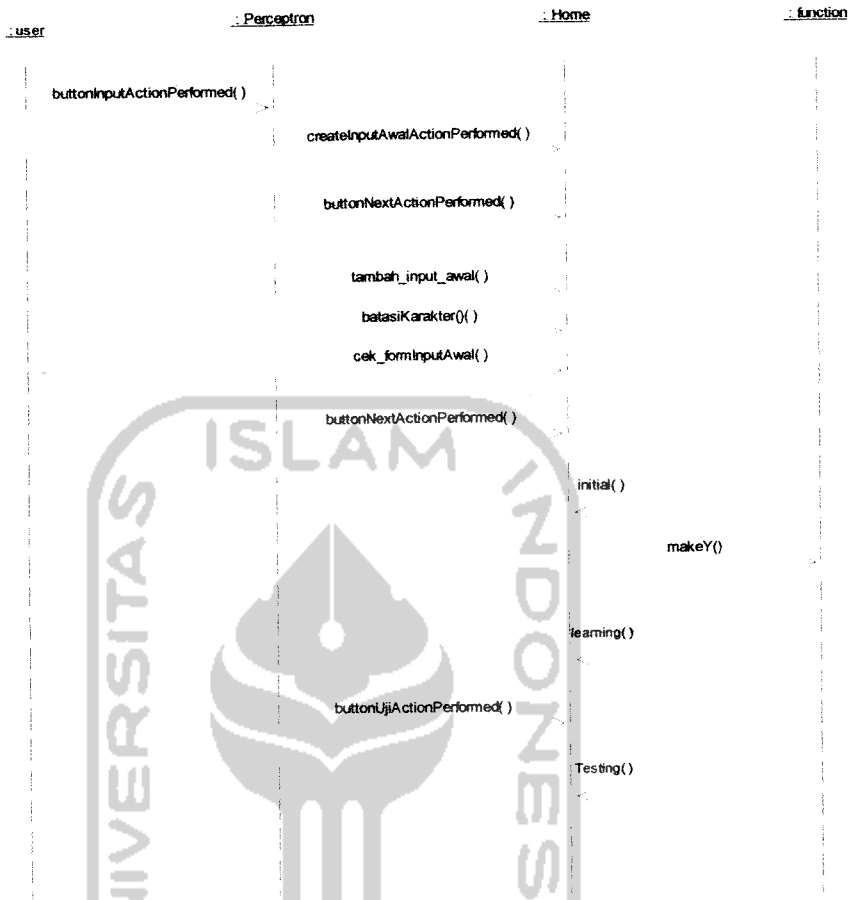
Gambar 4. 4 *Sequence Diagram* Proses Pelatihan

c. *Sequence Diagram* Proses Pengujian

Sequence Diagram proses pengujian menunjukkan interaksi *user* dengan perangkat lunak untuk melakukan pengujian perceptron berdasarkan *input* data oleh *user*. Objek yang berkaitan dengan *sequence* ini adalah sebagai berikut :

Aktor	:	*User*
Class Boundary	:	*Home*
Class Control	:	*buttonUjiActionPerformed()*
Class Entity	:	Tab *Testing Data*
Keterangan	:	1. *User* mengawali *sequence* ini dengan memanggil *method buttonNextActionPerformed* () pada tab *Input Data* dan sistem akan melakukan proses pelatihan. 2. Setelah itu *user* dapat melakukan proses pengujian dengan memasukkan nilai variabel *input* baru dan memanggil method *buttonUjiActionPerformed*(). 3. Kemudian tab *Testing Data* akan menampilkan hasil pengujian perceptron.

Sequence diagram proses pengujian dapat dilihat pada gambar 4.5.



Gambar 4. 5 *Sequence Diagram* Proses Pengujian

d. *Sequence Diagram* Struktur Jaringan

Sequence Diagram struktur jaringan menunjukkan interaksi *user* dengan perangkat lunak untuk melakukan pembuatan struktur jaringan berdasarkan *input* data oleh *user*. Objek yang berkaitan dengan *sequence* ini adalah sebagai berikut :

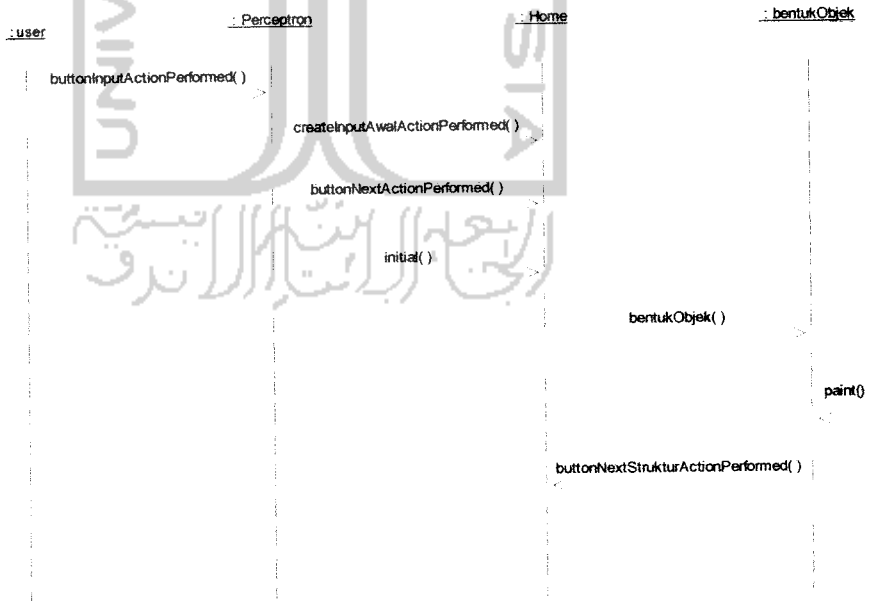
Aktor	:	*User*
Class Boundary	:	*Home*
Class Control	:	*buttonNextActionPerformed*()

Class Entity : Tab *Network Structure*

Keterangan :

1. *User* mengawali *sequence* ini dengan memanggil method *buttonInputActionPerformed*().
2. *Home* melakukan instansiasi dan memanggil method *createInputAwalActionPerformed*() untuk memproses *input* dari *user* dan sistem akan membentuk struktur jaringan.
3. Kemudian sistem memanggil method *buttonNextStrukturActionPerformed*() dan sistem akan menampilkan struktur jaringan.

Sequence diagram struktur jaringan dapat dilihat pada gambar 4.6.



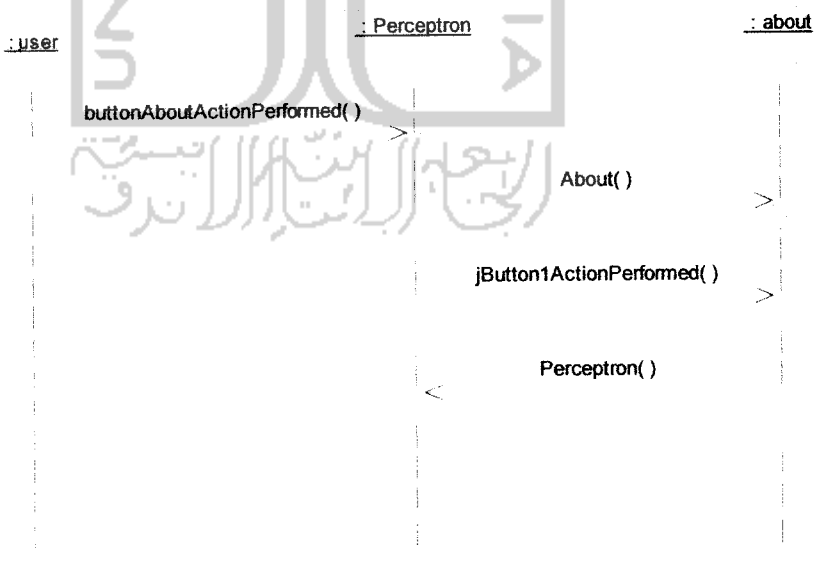
Gambar 4. 6 *Sequence Diagram* Membuat Struktur Jaringan

e. *Sequence Diagram* Lihat *About.*

Sequence Diagram lihat *about* menunjukkan interaksi *user* dengan perangkat lunak untuk menuju halaman *about*. Objek yang berkaitan dengan *sequence* ini adalah sebagai berikut :

Aktor	:	*User*
Class Boundary	:	*About*
Class Control	:	*buttonAboutActionPerformed*()
Class Entity	:	Halaman *About*
Keterangan	:	1. *User* mengawali *sequence* ini dengan memanggil *method* *buttonAboutActionPerformed*().
		2. Kemudian sistem akan menampilkan halaman *about*.

Sequence diagram lihat *about* dapat dilihat pada gambar 4.7.



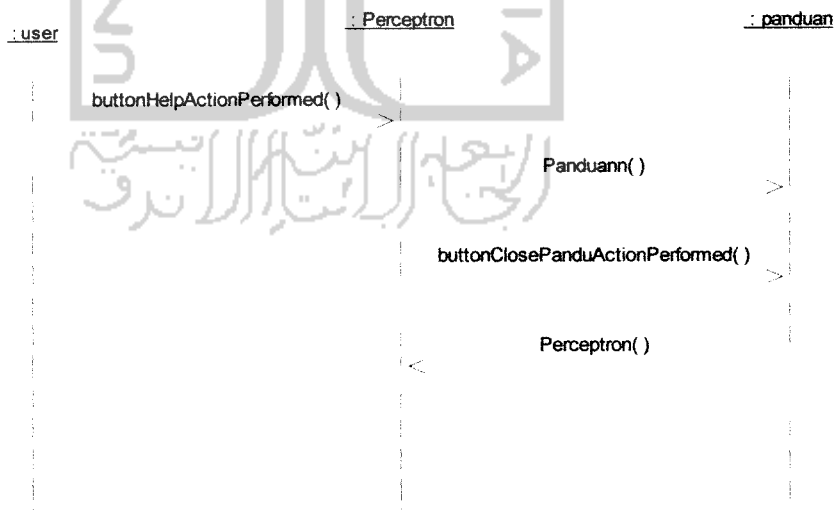
Gambar 4. 7 *Sequence Diagram* Lihat *About*

f. *Sequence Diagram* Lihat *Help.*

Sequence Diagram lihat *help* menunjukkan interaksi *user* dengan perangkat lunak untuk menuju halaman *help*. Objek yang berkaitan dengan *sequence* ini adalah sebagai berikut :

Aktor	:	*User*
Class Boundary	:	*About*
Class Control	:	*buttonHelpActionPerformed*()
Class Entity	:	Halaman *Help*
Keterangan	:	3. *User* mengawali *sequence* ini dengan memanggil *method* *buttonHelpActionPerformed*(). 4. Kemudian sistem akan menampilkan halaman *help*.

Sequence diagram lihat *help* dapat dilihat pada gambar 4.8.



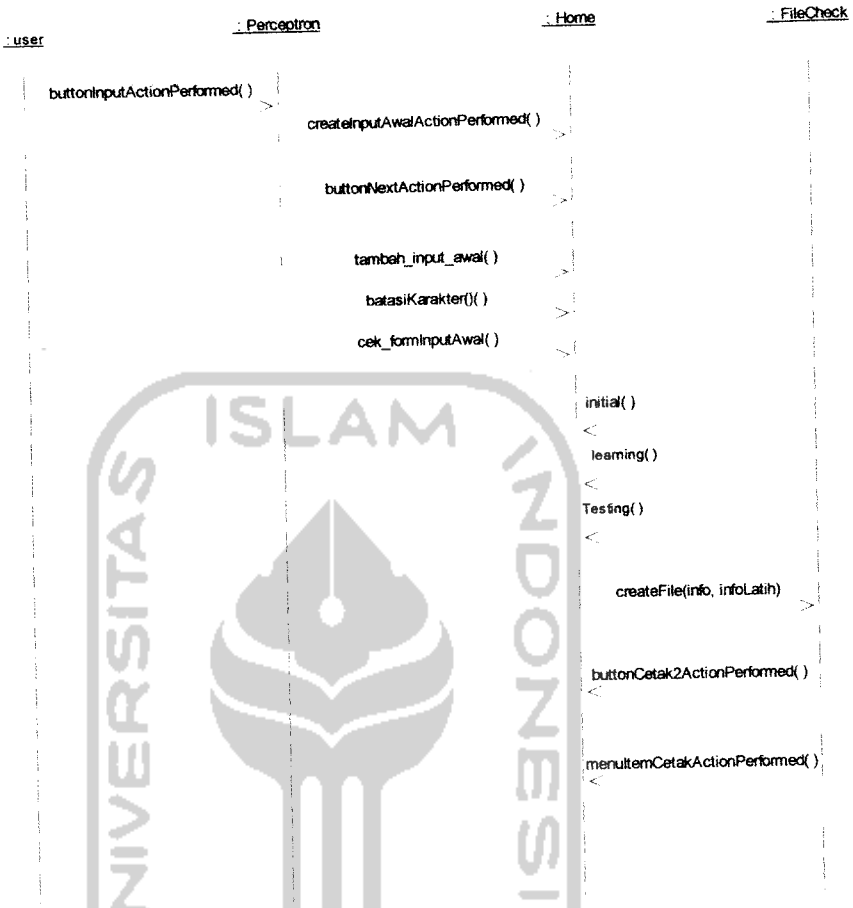
Gambar 4. 8 *Sequence Diagram* Lihat *Help*

g. *Sequence Diagram* Simpan Hasil Perhitungan.

Sequence Diagram simpan hasil perhitungan menunjukkan interaksi *user* dengan perangkat lunak untuk melakukan penyimpanan hasil perhitungan setelah melakukan proses pelatihan dan pengujian. File akan disimpan dalam bentuk dokumen. Objek yang berkaitan dengan *sequence* ini adalah sebagai berikut :

Aktor	:	*User*
Class Boundary	:	*Home*
Class Control	:	*buttonCetak2ActionPerformed()*
Class Entity	:	Tab *Testing Data*
Keterangan	:	1. Setelah melakukan proses pelatihan dan pengujian, *user* dapat melakukan penyimpanan data dengan memanggil *buttonCetak2ActionPerformed*(). 2. Kemudian sistem akan memberikan form untuk konfirmasi lokasi penyimpanan file dan file akan tersimpan.

Sequence diagram simpan hasil perhitungan dapat dilihat pada gambar 4.9.



Gambar 4. 9 *Sequence Diagram* Simpan Hasil Perhitungan

h. ***Sequence Diagram Input* Nilai Variabel Uji.**

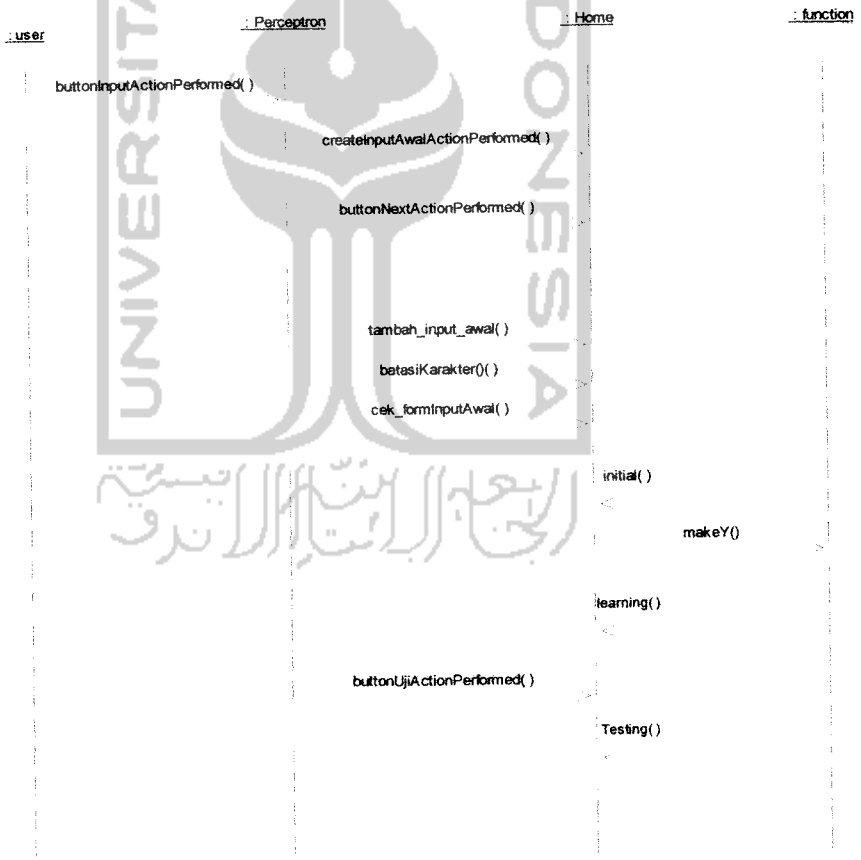
Sequence Diagram input nilai variabel uji menunjukkan interaksi *user* dengan perangkat lunak untuk melakukan proses pengujian data. Objek yang berkaitan dengan *sequence* ini adalah sebagai berikut :

Aktor	:	*User*
Class Boundary	:	*Home*
Class Control	:	*buttonUjiActionPerformed*()
Class Entity	:	Tab *Testing Data*

Keterangan : 1. Setelah melakukan proses pelatihan, *user* dapat melakukan pengujian data dengan memasukkan *input* nilai variabel uji dan memanggil *buttonUjiActionPerformed*().

2. Kemudian sistem akan melakukan proses pengujian berdasarkan *input* nilai variabel dan bobot hasil pelatihan.

Sequence diagram input nilai variabel uji dapat dilihat pada gambar 4.10.



Gambar 4. 10 *Sequence Diagram Input* Nilai Variabel Uji

4.2.4 *Activity Diagram*

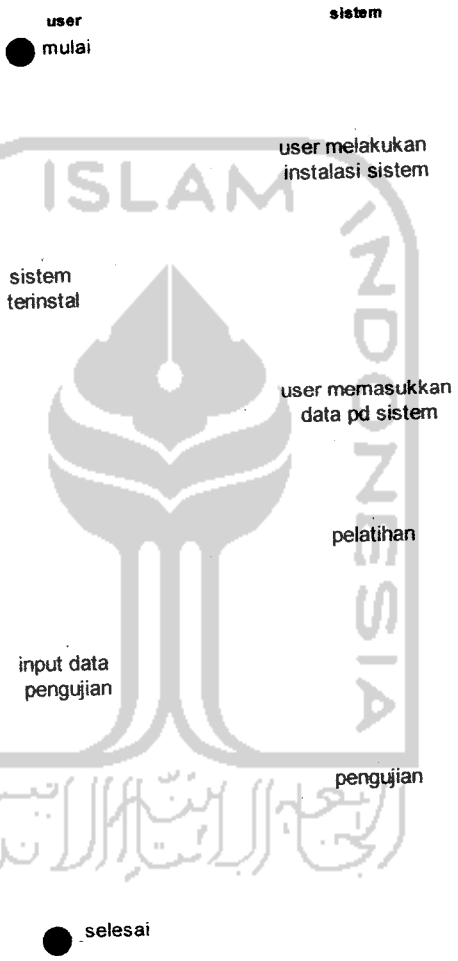
Activity diagram merupakan diagram yang menggambarkan alur kerja dari sistem. Diagram ini merupakan suatu diagram dinamis yang menunjukkan aktivitas beserta kejadian yang menyebabkan suatu objek berada dalam *state* tertentu. *Activity Diagram* ini lebih menggambarkan transisi-transisi dan aktivitas-aktivitas 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.

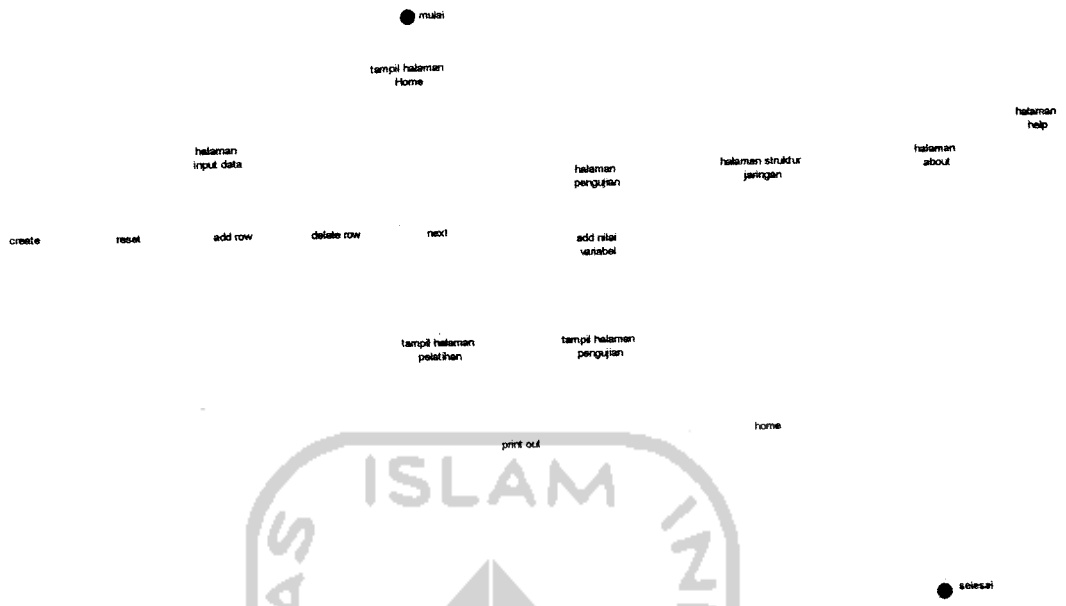
1. Pelatihan data. Urutan aktivitas proses pelatihan data pada sistem dapat dijelaskan sebagai berikut :
   a. *User* melakukan instalasi sistem.
   b. Setelah sistem terpasang *user* akan melakukan *setting* pada data.
   c. Kemudian *User* membuat struktur jaringan dengan memasukkan data pada sistem 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 :

a. *User* memasukkan nilai variabel (nilai untuk x_1, x_2, x_n) untuk proses pengujian.
b. Kemudian sistem akan melakukan proses pengujian data berdasarkan nilai variabel dan nilai bobot pada proses pelatihan.
c. Kemudian sistem mengakhiri kegiatan ini.



Gambar 4. 11 *Activity Diagram* instalasi sistem



Gambar 4. 12 *Activity Diagram tool* untuk JST model perceptron

Pada gambar 4.12 diatas dijelaskan bahwa pada saat *user* pertama kali menjalankan sistem *user* akan dihadapkan pada tampilan halaman *Home*, kemudian sistem akan memberikan pilihan *halaman input data, About dan Help*. Pada halaman *input data*, *user* memasukkan data yaitu jumlah variabel *input*, nilai variabel *input*, alpha, *threshold* dan maksimum epoh. Sistem memberikan 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 pelatihan. Pada halaman pengujian terdapat tabel untuk memasukkan nilai variabel untuk proses pengujian. Halaman struktur jaringan akan memberikan gambar struktur jaringan perceptron.

Pada halaman ini terdapat beberapa tombol, yaitu tombol *print* untuk menyimpan hasil perhitungan pelatihan dan pengujian kedalam bentuk dokumen, tombol *home* untuk kembali ke halaman *home*, tombol *about* untuk menampilkan halaman *about* dan terakhir tombol *help* untuk memanggil halaman panduan penggunaan program.

4.2.5 Perancangan Struktur Jaringan JST Model Perceptron

Langkah awal dalam perancangan neuron untuk membuat struktur JST adalah representasi jumlah variabel *input,* alpha, *threshold* dan maksimum epoh. Kemudian dilanjutkan dengan memasukkan nilai variabel *input* dan melakukan inisialisasi nilai bias dan memasukkan target (*output*).

a. Penentuan Parameter Jaringan Syaraf Tiruan

Sebagai contoh akan dibuat perceptron untuk mengenali fungsi logika AND dengan masukan dan keluaran bipolar. Untuk inisialisasi awal tiap parameter yang digunakan adalah bobot awal (w) = 0, Alpha (α) = 1, dan Threshold (θ) = 0. Contoh untuk perancangan neuron dan nilai neuron terdapat pada tabel 4.2 berikut :

Tabel 4. 2 Tabel Parameter Masukan dan Target

Masukan		Bias	Target
X_1	X_2	b	t
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

Untuk threshold θ = 0, maka fungsi aktifasinya :

$$f(net) = \begin{cases} 1 & jika \quad net > \theta \\ 0 & jika \quad -\theta \le net \le \theta \\ -1 & jika \quad net < -\theta \end{cases} \quad \text{..... (4.1)}$$

b. Proses Epoh

Epoh dilakukan terus hingga semua pola memiliki keluaran jaringan yang sama dengan targetnya (jaringan sudah memahami pola). Perubahan bobot hanya dilakukan pada pola yang mengandung kesalahan (keluaran jaringan ≠ target). Perubahan tersebut merupakan hasil kali unit masukan dengan target dan laju pemahaman. Perubahan bobot hanya akan terjadi kalau unit masukan ≠ 0.

Kecepatan epoh ditentukan pula oleh laju pemahaman (=α dengan $0 \leq \alpha \leq 1$) yang dipakai. Semakin besar harga α, semakin sedikit epoh yang diperlukan. Akan tetapi jika α terlalu besar, maka akan merusak pola yang sudah benar sehingga pemahaman menjadi sedikit lambat. Epoh untuk seluruh pola yang ada disebut **Epoh** . Tabel 4.3 berikut menunjukkan hasil pada epoh pertama.

Tabel 4. 3 Epoh Pertama

Masukan	target			Perubahan Bobot	Bobot Baru
(x_1 x_2 1)	t	net	Y=f(net)	(Δw_1 Δw_2 Δb)	(w_1 w_2 b)
Inisialisasi					(0 0 0)
(1 1 1)	1	0	0	(1 1 1)	(1 1 1)
(1 -1 1)	-1	1	1	(-1 1 -1)	(0 2 0)
(-1 1 1)	-1	2	1	(1 -1 -1)	(1 1 -1)
(-1 -1 1)	-1	-3	-1	(0 0 0)	(1 1 -1)

Pada *input* yang pertama (x_1 x_2 1) = (1 1 1). Nilai net dihitung berdasarkan bobot yang sudah ada sebelumnya yaitu (w_1 w_2 b) = (0 0 0). Maka net = $\sum_i xi\ wi + b$ = 1(0) + 1(0) + 0 = 0, sehingga f(net) = f(0) = 0.

c. Proses Perubahan Bobot

Dari hasil perhitungan diatas diketahui bahwa keluaran jaringan (f(net)=0) tidak sama dengan target yang diinginkan (dalam epoh ini target = 1), maka bobot harus di ubah menggunakan rumus :

$\Delta w_i = \alpha\ t\ x_i = t\ x_i$ (karena $\alpha = 1$). Bobot baru = bobot (lama) + Δw_i .....(4.2)

Input pola kedua dan seterusnya dihitung dengan cara yang sama. Pada pola terakhir (x_1 x_2 1) = (-1 -1 1), nilai f(net) = -1 yang sama dengan targetnya. Maka bobot tidak diubah. Hal ini dinyatakan dengan kondisi $\Delta\ w_i = 0$. Garis pemisah pola terbentuk dari persamaan :

$x_1 w_1 + x_2 w_2 + b = 0$ (karena $\theta = 0$), jadi terbentuk sebuah garis saja ..... (4.3)

Persamaan garis pemisah untuk setiap pola hasil epoh dapat dilihat pada table 4.4 berikut :

Tabel 4. 4 Persamaan Garis Hasil Epoh

Masukan	Bobot Baru	Persamaan Garis
(x_1 x_2 1)	(w_1 w_2 b)	
(1 1 1)	(1 1 1)	$x_1 + x_2 = -1$
(1 -1 1)	(0 2 0)	$2x_2 = 0$
(-1 1 1)	(1 1 -1)	$x_1 + x_2 = 1$
(-1 -1 1)	(1 1 -1)	$x_1 + x_2 = 1$

Mengingat tidak semua f(net) pada tabel di atas sama dengan t, maka epoh dilanjutkan pada epoh kedua. Semua pola kembali dimasukkan ke jaringan dengan menggunakan bobot terakhir yang diperoleh sebagai bobot awalnya. Diperoleh hasil epoh pada tabel 4.5 sebagai berikut :

Tabel 4. 5 Epoh Kedua

Masukan			target			Perubahan Bobot			Bobot Baru		
(x_1	x_2	1)	t	net	Y=f(net)	(Δw_1	Δw_2	Δb)	(w_1	w_2	b)
Bobot yang diperoleh dari epoh pertama									(1	1	-1)
(1	1	1)	1	1	1	(0	0	0)	(1	1	-1)
(1	-1	1)	-1	-1	-1	(0	0	0)	(1	1	-1)
(-1	1	1)	-1	-1	-1	(0	0	0)	(1	1	-1)
(-1	-1	1)	-1	-3	-1	(0	0	0)	(1	1	-1)

Dalam epoh di atas, untuk semua pola nilai f(net) = t, sehingga tidak dilakukan perubahan terhadap bobot. Karena f(net) = t untuk semua pola maka jaringan sudah mengenal semua pola sehingga epoh dihentikan.

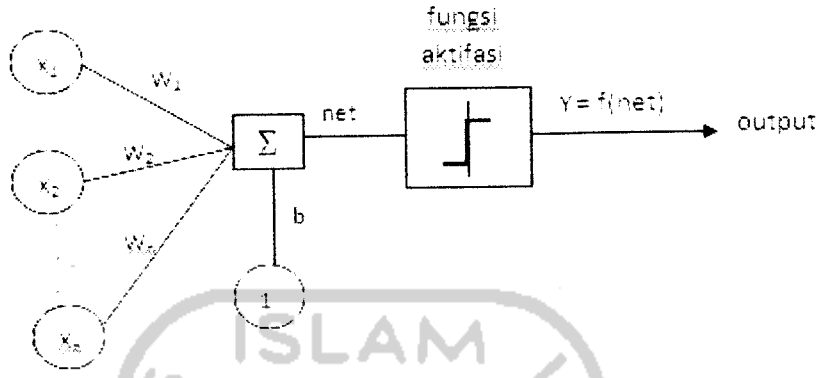
4.2.6 Perancangan Visual Jaringan

Desain visualisasi Jaringan Syaraf Tiruan Model Perceptron didasarkan pada rancangan struktur jaringan yang terdiri atas jumlah variabel (node/neuron) dan nilai dari tiap-tiap variabel (node/neuron) yang dimasukkan oleh *user* sesuai dengan kebutuhannya. Berdasarkan perancangan struktur jaringan, selanjutnya *input* data digunakan sebagai data visual untuk struktur jaringan syaraf tiruan model perceptron.

Seperti yang telah dibahas sebelumnya, sistem Jaringan Saraf Tiruan merupakan analogi yang berkaitan erat dengan proses berpikir dalam otak manusia. Jaringan Saraf Tiruan ditentukan oleh 3 hal yang paling mendasar :

1. Pola hubungan antarneuron (arsitektur jaringan),
2. Metode untuk menentukan bobot penghubung (*learning* atau *training method*), dan
3. Fungsi aktivasi.

Berdasarkan contoh pada penentuan parameter diatas, dapat kita tentukan visual struktur jaringan seperti gambar 4.13 di bawah ini :



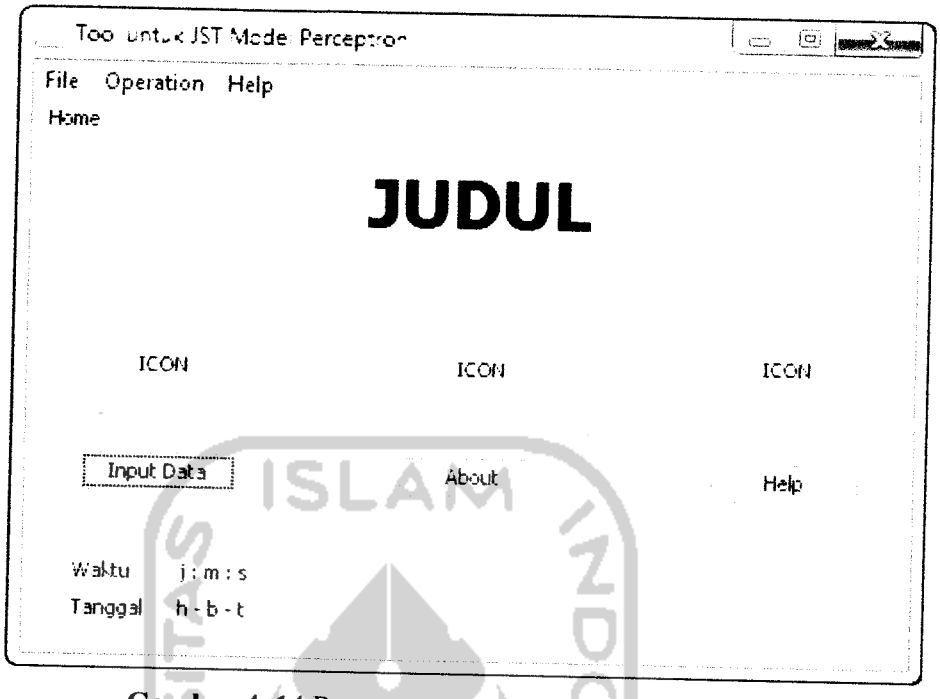
Gambar 4. 13 Perancangan Visual Jaringan

4.2.7 Perancangan Antar Muka

Rancangan antar muka dari rancang bangun tool untuk struktur JST model Perceptron terdiri dari antar muka halaman *Home* dan halaman Proses. Pada halaman proses terdapat 6 (enam) tab pane, yaitu tab *Input Data*, tab hasil pelatihan, tab tabel epoh terkahir, tab pengujian, dan tab struktur jaringan.

a. Antar Muka Halaman *Home*

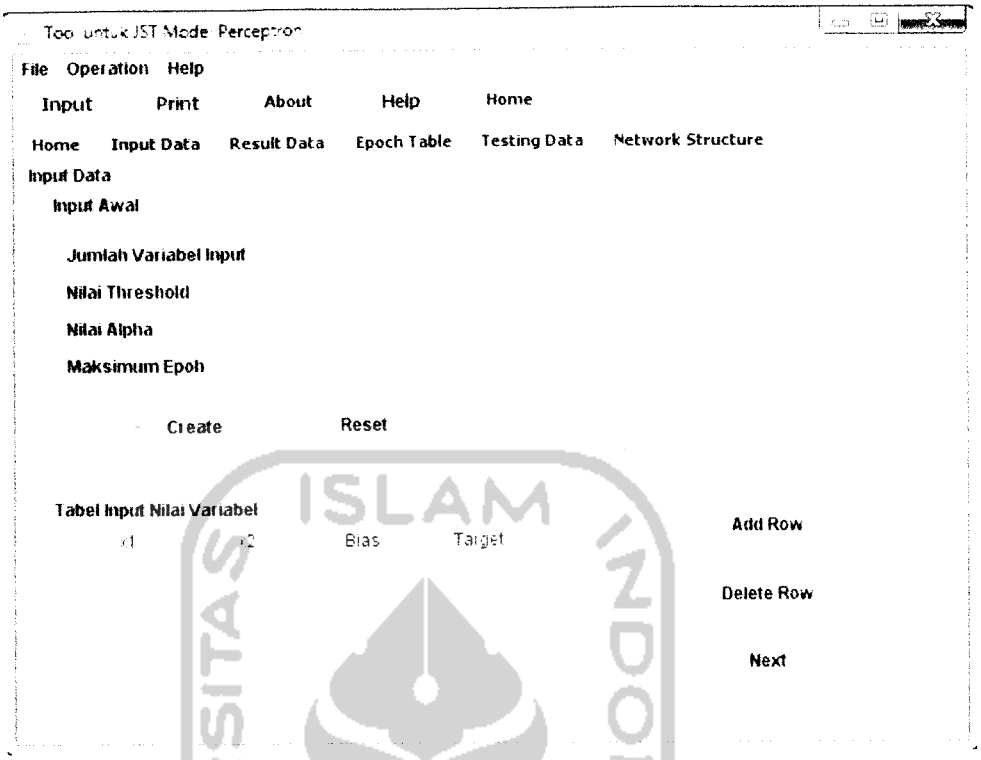
Halaman *Home* merupakan halaman yang muncul pertama kali ketika *user* menggunakan aplikasi. Pada rancang bangun tool untuk struktur JST model Perceptron, halaman utama terdiri atas 3 menu, yaitu menu *Input Data* (untuk inisialisasi data), menu *About* (tentang program dan *programmer*) dan menu *Help* (panduan penggunaan program). Gambar 4.14 menunjukkan rancangan antar muka halaman *Home* rancang bangun *tool* untuk struktur JST model Perceptron.



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.*

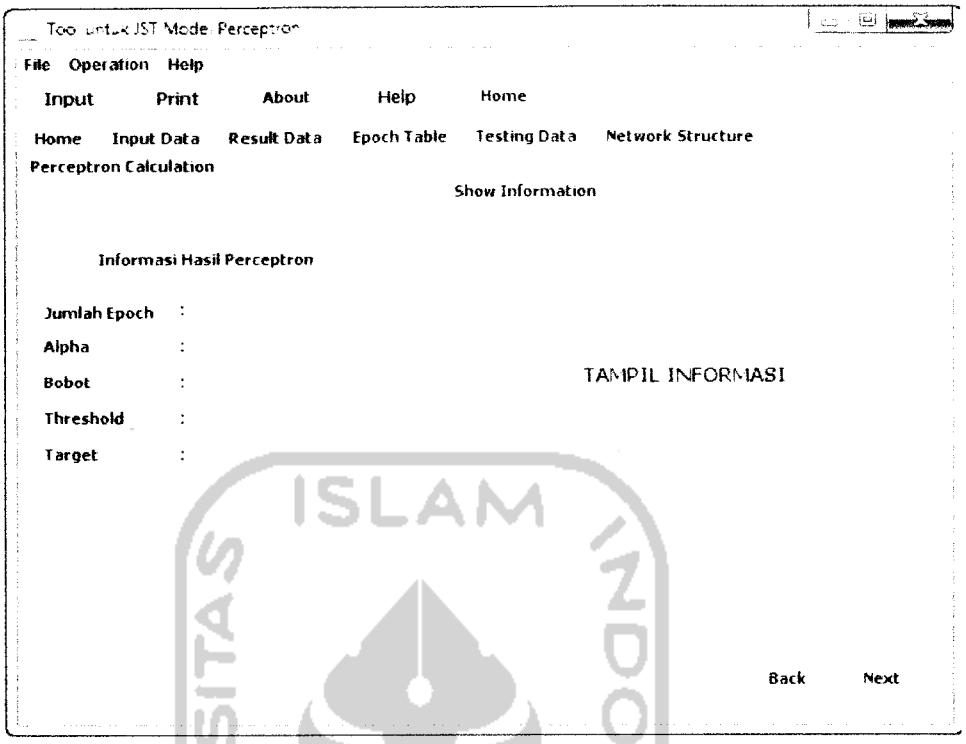


Gambar 4. 15 Rancangan Antar Muka Halaman *Input Data*

Pada rancangan antar muka *Input Data*, terdapat beberapa untuk pengisian parameter pembuatan jaringan, tombol *Add Baris* untuk menambah nilai dari masing-masing variabel. Tombol Next untuk melanjutkan ke halaman hasil data.

c. Antar Muka Halaman Hasil Data

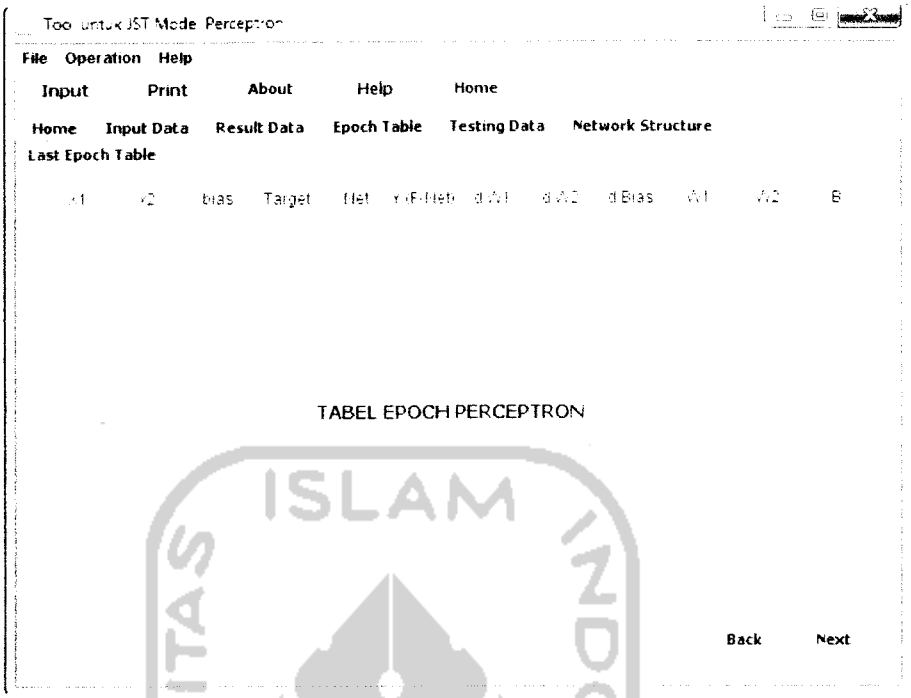
Halaman hasil data adalah halaman hasil perhitungan pelatihan data dari rancang bangun *tool* untuk struktur JST model Perceptron. Gambar 4.16 menunjukkan rancangan antar muka hasil data.



Gambar 4. 16 Rancangan Antar Muka Informasi Hasil Perhitungan

d. Antar Muka Halaman Tabel Epoh

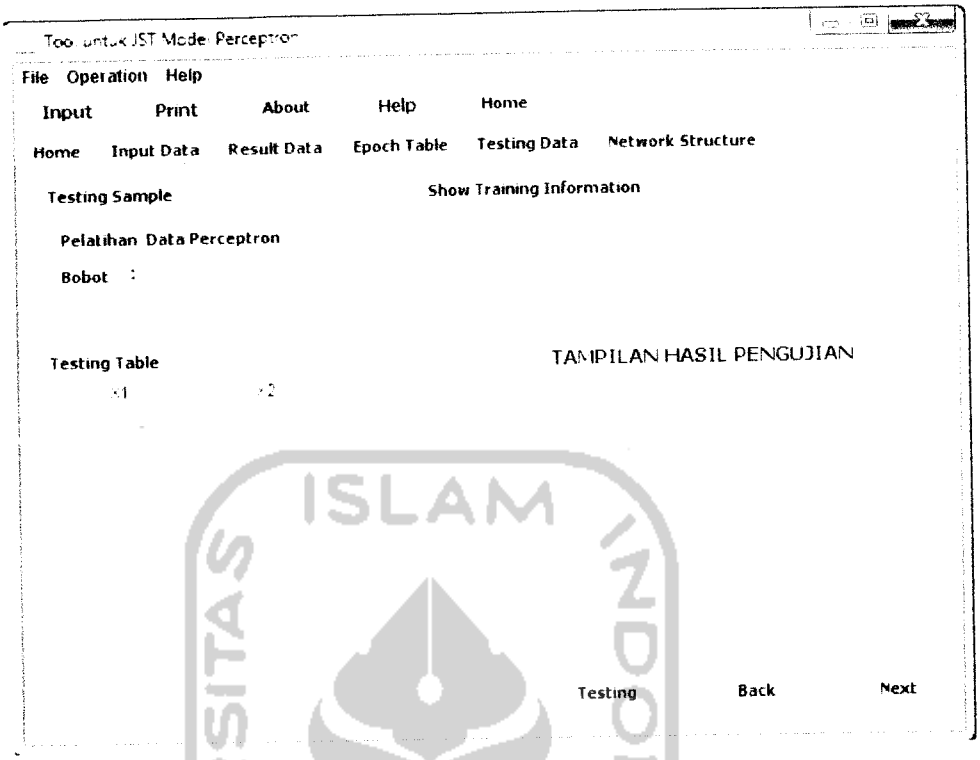
Halaman tabel epoh ini akan menampilkan tabel hasil perhitungan proses pelatihan epoh pada sistem. Gambar 4.17 menunjukkan rancangan antar muka Tabel Epoh .



Gambar 4. 17 Rancangan Antar Muka Halaman Tabel Epoh

e. Antar Muka Halaman Pengujian Data

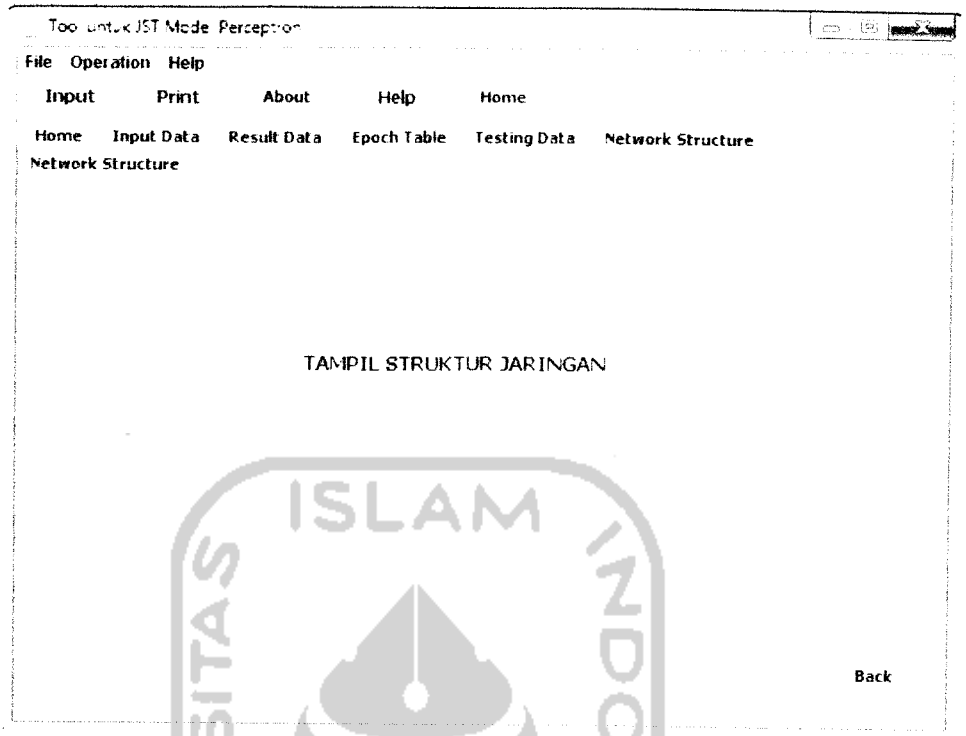
Halaman pengujian data adalah halaman untuk melakukan proses pengujian data dari hasil pelatihan perceptron. Pada halaman ini terdapat tabel untuk memasukkan data pengujian. Setelah data dimasukkan, kemudian *user* mengklik tombol *testing* untuk melakukan proses pengujian data pada sistem. Gambar 4.18 menunjukkan rancangan antar muka pengujian data.



Gambar 4. 18 Rancangan Antar Muka Halaman Pengujian Data

f. Antar Muka Halaman Struktur Jaringan.

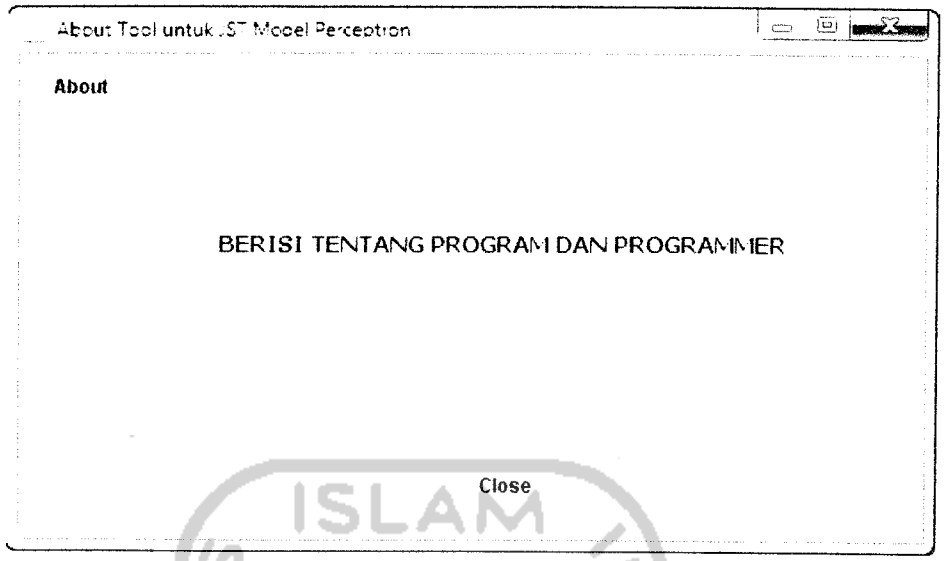
Pada halaman struktur jaringan terdapat sebuah panel untuk menampilkan struktur jaringan. Gambar 4.19 menunjukkan rancangan antar muka pengujian data.



Gambar 4. 19 Antar Muka Halaman Struktur Jaringan

g. Antar Muka Halaman *About*

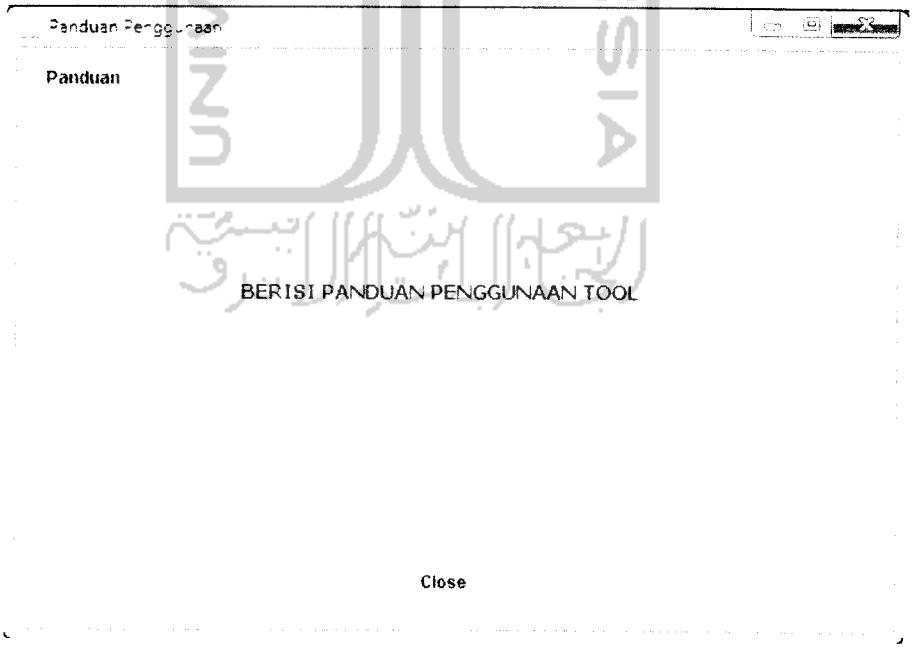
Halaman *about* adalah halaman yang berisikan data tentang *tool* untuk struktur JST model Perceptron dan programmer. *User* dapat menghubungi programmer jika terdapat kesulitan dalam menjalankan atau mengembangkan program. Gambar 4.20 menunjukkan rancangan antarmuka halaman *about.*



Gambar 4. 20 Rancangan Antar Muka Halaman *About*

h. Antar Muka Halaman *Help*

Halaman *help* adalah halaman yang berisikan manual dan cara penggunaan *tool* untuk struktur JST model Perceptron. Gambar 4.21 menunjukkan rancangan antarmuka halaman *help.*



Gambar 4. 21 Rancangan Antar Muka Halaman *Help*

4.3 Implementasi Perangkat Lunak

Pada tahap implementasi, sistem dioperasikan dalam keadaan sesungguhnya. Tujuan implementasi ini adalah untuk mengetahui apakah sistem yang dibuat benar dan sesuai dengan perancangan yang disiapkan.

4.3.1 Batasan Implementasi

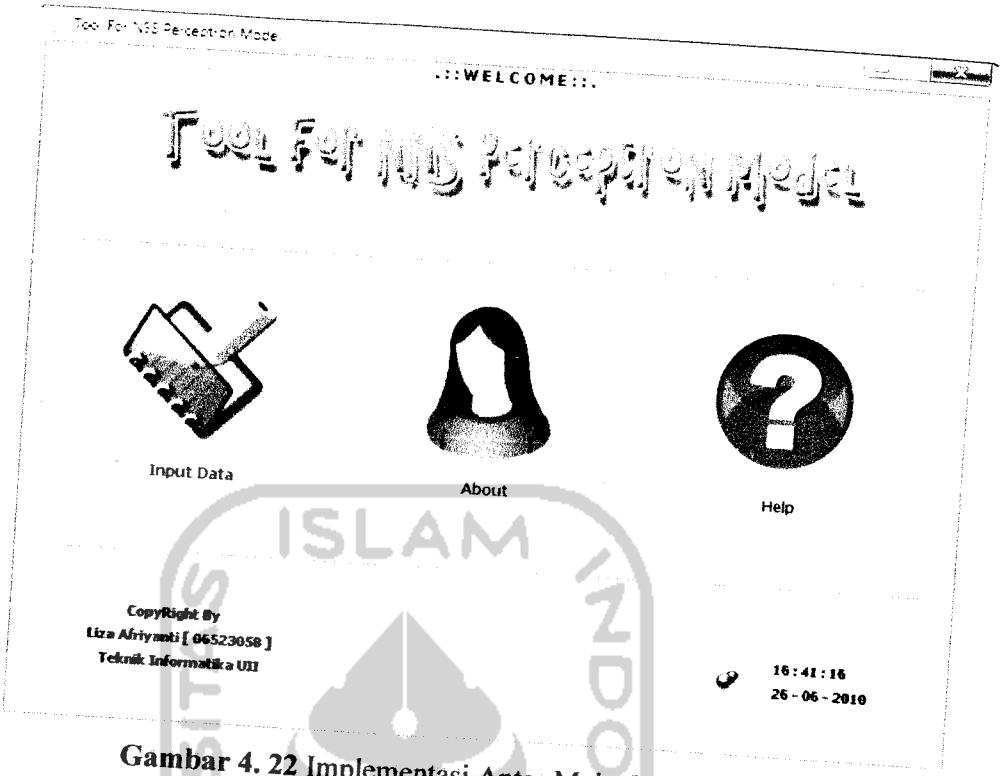
Pada implementasi *tool* untuk Jaringan Syaraf Tiruan (JST) model perceptron akan dijelaskan bagaimana sistem ini bekerja, dengan memberikan tampilan form-form dan tampilan output yang dibuat. Implementasi terdiri dari implementasi antar muka dan implementasi prosedural.

4.3.2 Implementasi Antar Muka

Implementasi antarmuka merupakan tampilan antar muka sistem sesuai dengan perancangannya. Implementasi antar muka dari tool untuk JST model perceptron terdiri dari antar muka halaman *Home*, halaman *Input Data*, halaman *Result Data*, halaman *Table Epoch*, halaman *Testing Data*, halaman *Network Structure*, halaman *About* dan halaman *Help*.

a. Halaman *Home*

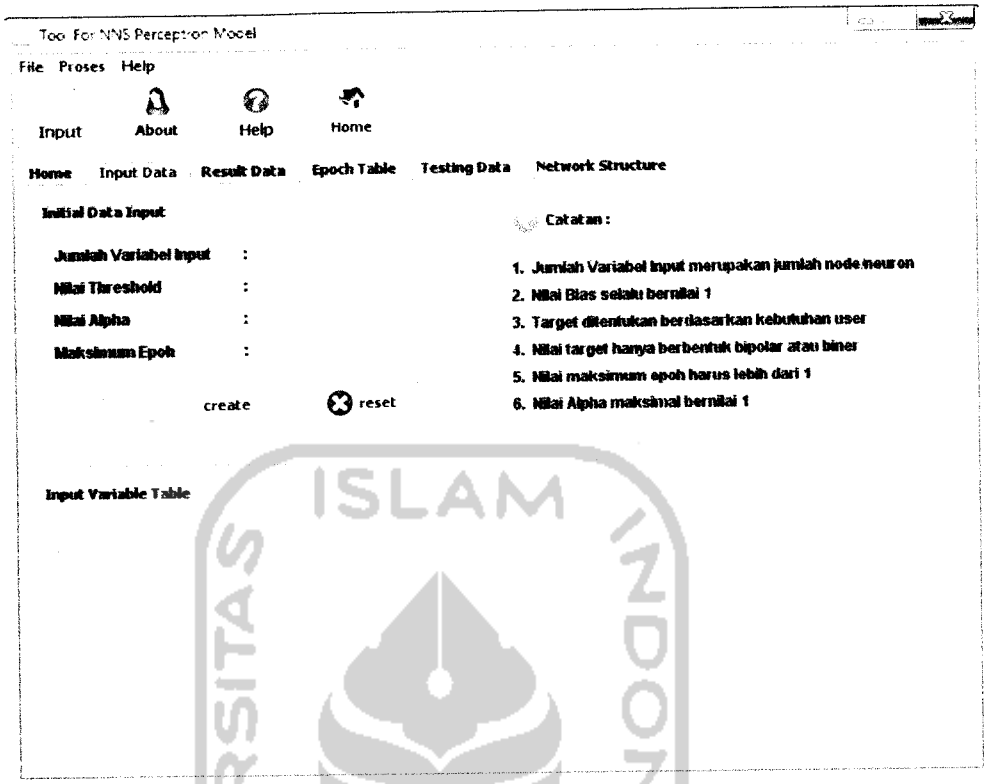
Halaman utama merupakan halaman yang muncul pertama kali ketika *user* menggunakan aplikasi. Pada tool untuk JST model perceptron, halaman utama terdiri atas tombol 3 menu, yaitu menu *Input Data* (untuk memasukkan parameter), menu *About* (tentang program dan *programmer*) dan menu *Help* (manual dan dokumentasi program). Gambar 4.22 menunjukkan implementasi antar muka halaman *Home* tool untuk JST model perceptron.



Gambar 4. 22 Implementasi Antar Muka Halaman *Home*

b. Halaman *Input* Data

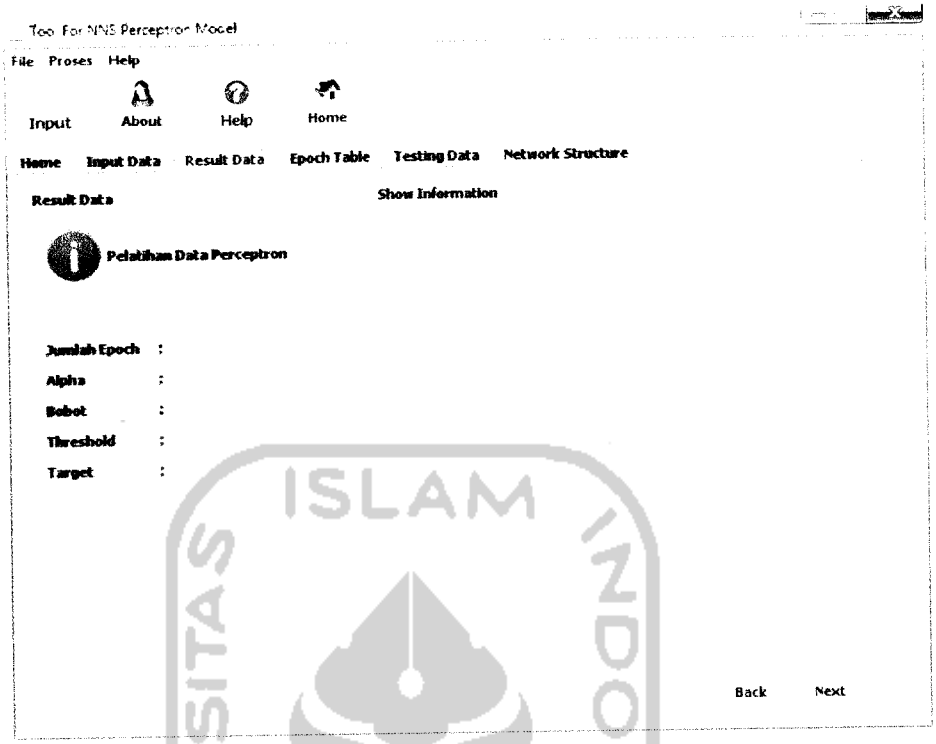
Pada *tool* untuk JST model perceptron, halaman *input* data terdiri dari beberapa pilihan tombol, yaitu tombol *create* untuk menginstansiasi *input*, tombol *reset* untuk menset ulang *input* awal, tombol *add row* untuk menambah baris pada tabel *input*, tombol *delete row* untuk menghapus baris pada 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. 23 Implementasi Antar Muka Halaman *Input* Data

c. Halaman *Result* Data

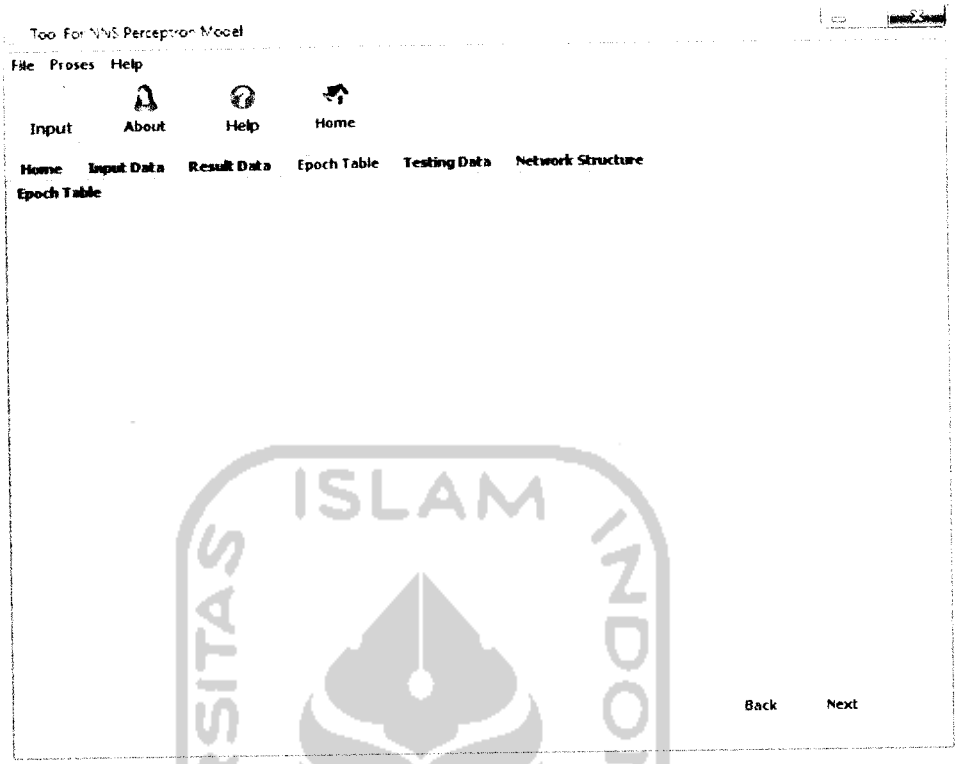
Pada halaman *result* data terdapat informasi hasil perhitungan perceptron seperti jumlah epoh, alpha, bobot, threshold, dan target. Untuk hasil perhitungan secara lengkap dapat dilihat pada *text area show information.* Gambar 4.24 menunjukkan implementasi antar muka halaman *result* data *tool* untuk JST model perceptron.



Gambar 4. 24 Implementasi Antar Muka Halaman Informasi

d. Halaman *Epoch Table*

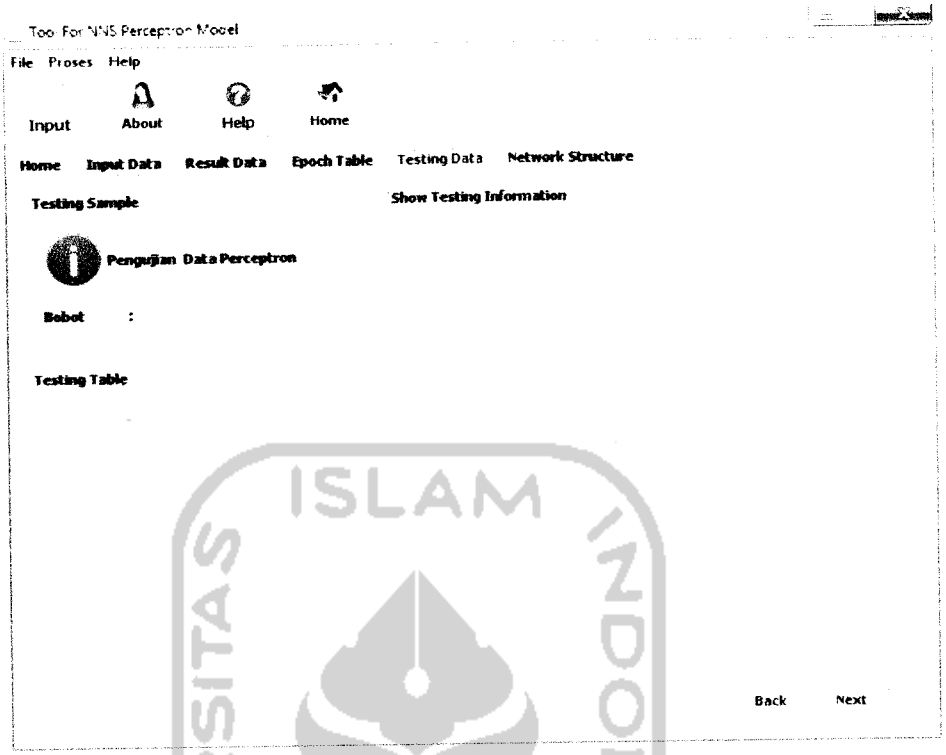
Pada halaman *epoch table* terdapat informasi hasil perhitungan proses pelatihan perceptron pada epoh. Gambar 4.25 menunjukkan implementasi antar muka halaman *epoch table tool* untuk JST model perceptron.



Gambar 4. 25 Implementasi Antar Muka Halaman Tabel Epoh

e. Halaman *Testing Data*

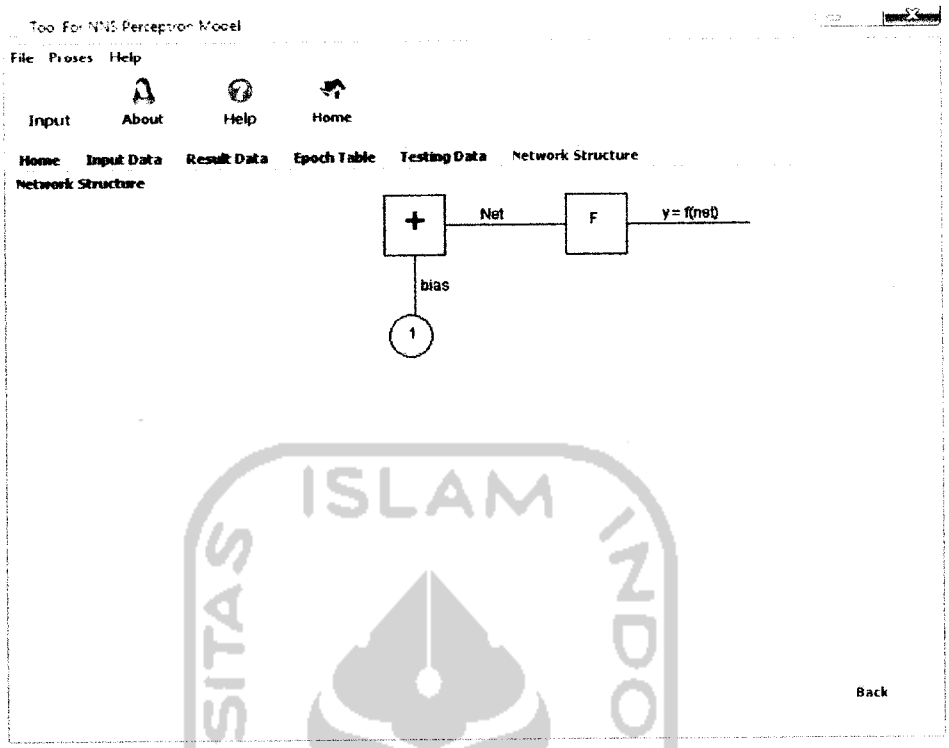
Pada halaman *result* data terdapat informasi hasil perhitungan pengujian perceptron. Sebelum melakukan pengujian, *user* diharuskan untuk memasukkan nilai variabel *input* untuk pengujian. Kemudian sistem akan melakukan proses pengujian perceptron dengan menggunakan bobot hasil pelatihan. Untuk hasil perhitungan secara lengkap dapat dilihat pada *text area show Training information.* Gambar 4.26 menunjukkan implementasi antar muka halaman *training* data *tool* untuk JST model perceptron.



Gambar 4. 26 Implementasi Antar Muka Halaman *Testing Data*

f. Halaman *Network Structure*

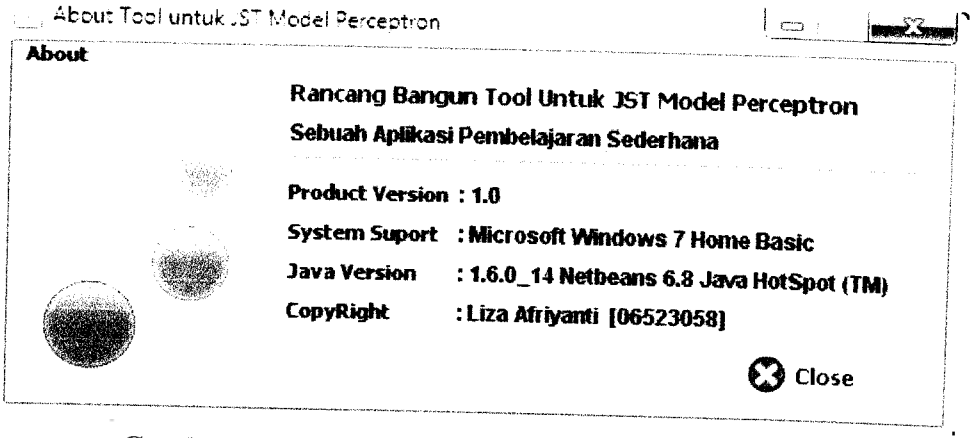
Pada halaman *network structure* terdapat gambar struktur jaringan perceptron. Jumlah node/neuron tergantung pada jumlah *input variable* yang dimasukkan oleh *user* pada halaman *input* data. Gambar 4.27 menunjukkan implementasi antar muka halaman *network structure tool* untuk JST model perceptron.



Gambar 4. 27 Implementasi Antar Muka Halaman *Network Structure*

g. Halaman *About*

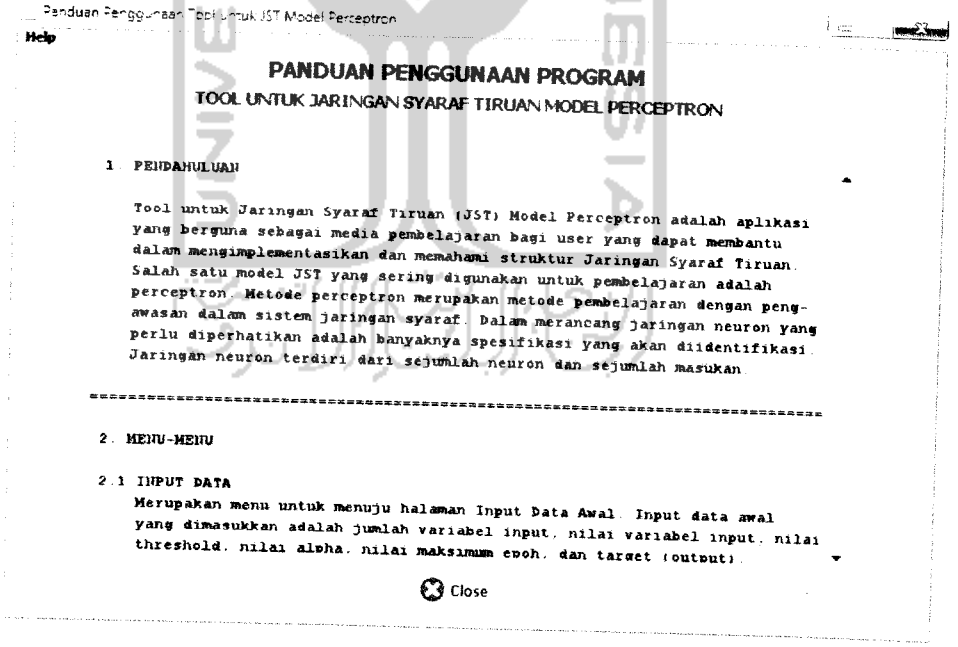
Halaman *about* adalah halaman yang berisikan data tentang *tool* untuk JST model perceptron dan *programmer*. *User* dapat menghubungi *programmer* jika terdapat kesulitan dalam menjalankan atau mengembangkan program. Gambar 4.28 menunjukkan implementasi antar muka halaman *about.*



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*

4.3.3 Implementasi Prosedural

Implementasi prosedural merupakan implementasi pada pemrograman sistem. Namun yang dijelaskan di sini adalah pemrograman Jaringan Syaraf Tiruan pada Java, yang merupakan proses inti dalam pembuatan *tool* untuk JST model perceptron.

a. Method Proses Inisialisasi

Method ini berfungsi untuk menginisialisasi semua data yang telah dimasukkan oleh *user*. Berikut adalah implementasi method proses inisialisasi.

```
private void initial() {
        info += "\n >>>>>>>>>>> PROSES INISIALISASI
<<<<<<<<<<< \n\n";
        //inisial isi node/nilai variabel input
        try {
            x = new
int[tabelInputAwal.getRowCount()][Integer.parseInt(jumVarInp
ut.getText())];
            for (int i = 0; i <
tabelInputAwal.getRowCount(); i++) {
                for (int j = 0; j <
Integer.parseInt(jumVarInput.getText()); j++) {
                    x[i][j] =
Integer.parseInt(tabelInputAwal.getValueAt(i,
j).toString());
                }
            }
            //inisial bias
            b = new int[tabelInputAwal.getRowCount()];
            for (int i = 0; i < b.length; i++) {
                b[i] =
Integer.parseInt(tabelInputAwal.getValueAt(i,
Integer.parseInt(jumVarInput.getText())).toString());
            }
            // inisial target;
            t = new int[tabelInputAwal.getRowCount()];
            for (int i = 0; i <
tabelInputAwal.getRowCount(); i++) {
                t[i] =
Integer.parseInt(tabelInputAwal.getValueAt(i,
(tabelInputAwal.getColumnCount() - 1)).toString());
            }
            //menampilkan nama node pada text area
            for (int i = 0; i < x[0].length; i++) {
                info += " x" + (i + 1) + " ";
            }
```

```
            //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);
        }
    }
```

b. Method Proses Pelatihan

Method ini berfungsi untuk melakukan proses pelatihan data yang telah diinisialisasikan. Berikut adalah implementasi method pelatihan.

```
private void learning() {
        DecimalFormat df = new
DecimalFormat("#########.#####");
        Object[] hsl_akhr = new
Object[tabel_hasil.getColumnCount()];
        Object[] barisKosong = new
Object[tabel_hasil.getColumnCount()];
        Object[] pemisah = new Object[1];
        Function fct = new Function();
        int jml_node =
Integer.parseInt(jumVarInput.getText());
        tabel_hasil.setRowCount(0);

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

        info += "\n\n >>>>>>>>>> PROSES PEMBELAJARAN
<<<<<<<<<< \n";
        stop = false;
        epoch = 0;
        int ha;
        try {
            while (stop == false) {
                epoch++;
                int sama = 0;
                double y_in = 0;
                info += "\n ---> EPOCH KE-" + epoch + "\n";
                for (int i = 0; i < x.length; i++) {
                    y_in = 0;
                    info += " Data ke- " + (i + 1) + "\n";
                    info += " y_in = " + bb + " + "; //
menampilkan hasil net k text area
                    for (int j = 0; j < x[0].length; j++) {
                        y_in += (w[j]) * (x[i][j]);
                        hsl_akhr[j] = x[i][j];
                        if (j == 0) {
                            info += w[j] + "*" + x[i][j];
                        } else {
                            info += " + " + w[j] + "*" +
x[i][j];

                        }
                    }
                    hsl_akhr[jml_node] = b[i];
                    hsl_akhr[(jml_node + 1)] = t[i];
```

```
                    y_in =
Double.parseDouble(df.format(y_in).replace(',', '.'));
                    y_in = Double.parseDouble(df.format(y_in
+ bb).replace(',', '.')); // mencari nilai net
                    hsl_akhr[(jml_node + 2)] = y_in;
                    info += " = " + y_in + "\n";
                    ha = fct.makeY(y_in, threshold); //
menentukan nilai output jaringan
                    hsl_akhr[(jml_node + 3)] = ha;
                    info += " Hasil aktivasi = " + ha +
"\n";
                    info += " Target = " + t[i] + "\n";

                    //Jika output jaringan != target(output)
                    if (ha != t[i]) {
                        //nyari bobot baru
                        info += " Bobot Baru : \n";
                        for (int j = 0; j < x[0].length;
j++) {
                            barisKosong[(jml_node + 3 +
jml_node + 1 + 1 + j)] = w[j];
                            dw[j] =
Double.parseDouble(df.format(lRate * t[i] *
x[i][j]).replace(',', '.'));
                            hsl_akhr[(jml_node + 4 + j)] =
dw[j];
                            info += " W" + (j + 1) + " = " +
w[j] + " + " + lRate + " * " + t[i] + " * " + x[i][j] + " =
" + df.format(w[j] + (lRate * t[i] * x[i][j])) + "\n";
                            w[j] =
Double.parseDouble(df.format(w[j] + (lRate * t[i] *
x[i][j])).replace(',', '.'));
                            hsl_akhr[(jml_node + 3 +
jml_node + 1 + 1 + j)] = w[j];

                        }
                        //nyari bobot bias baru
                        info += " Bobot Bias Baru :\n";
                        info += " b = " + bb + " + " + lRate
+ " * " + t[i] + " = " + df.format(bb + (lRate * t[i])) +
"\n";
                        //masukin nilai bbl k tabel_hasil
(dBB)
                        barisKosong[(jml_node + 3 + jml_node
+ 1 + jml_node + 1)] = bb;
                        bbl = (lRate * t[i]);
                        hsl_akhr[(jml_node + 3 + jml_node +
1)] = bbl;
                        bb += bbl;
                        bb =
Double.parseDouble(df.format(bb).replace(',', '.'));
                        //masukin nilai bbb k tabel_hasil
(BB)
                        hsl_akhr[(jml_node + 3 + jml_node +
1 + jml_node + 1)] = bb;
```

```
                    } 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));
```

```
                    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);
                }
                if (epoch == epohMaksimum) {
                    stop = true;
                }
            }
        } catch (Exception e) {
            //     JOptionPane.showMessageDialog(null,
"Terjadi Kesalahan Pelatihan Pada : \n" + e);
        }
    }
```

c. Method Proses Pengujian.

Method ini berfungsi untuk melakukan pengujian perceptron pada sistem setelah proses pelatihan. Berikut adalah implementasi method proses pengujian.

```
private void Testing() {
        DecimalFormat df = new
DecimalFormat("#########.#####");
        infoLatih += "\n >>>>>>>>>>> PROSES PENGUJIAN
<<<<<<<<<<< \n\n";


        try {
            xUji = new
Double[tabel_uji.getRowCount()][Integer.parseInt(jumVarInput
.getText())];
            for (int i = 0; i < tabel_uji.getRowCount();
i++) {
                for (int j = 0; j <
Integer.parseInt(jumVarInput.getText()); j++) {
                    xUji[i][j] =
Double.parseDouble(tabel_uji.getValueAt(i, j).toString());
                }
            }
```

```
            for (int i = 0; i < xUji[0].length; i++) {
                infoLatih += " x" + (i + 1) + "    ";
            }
            infoLatih += "\n";
            for (int i = 0; i < xUji.length; i++) {
                for (int j = 0; j < xUji[0].length; j++) {
                    //inisialisasi node
                    infoLatih += xUji[i][j] + "    ";
                }
                infoLatih += "\n";
            }

            int haUji;
            Function fct = new Function();
            double y_inUji = 0;
            for (int i = 0; i < xUji.length; i++) {
                y_inUji = 0;
                infoLatih += " \n Data ke- " + (i + 1) +
"\n\n";
                infoLatih += " y_in = " + bb + " + ";
                for (int j = 0; j < xUji[0].length; j++) {
                    y_inUji += (w[j]) * (xUji[i][j]);
                    if (j == 0) {
                        infoLatih += w[j] + "*" +
xUji[i][j];
                    } else {
                        infoLatih += " + " + w[j] + "*" +
xUji[i][j];
                    }
                }
                y_inUji =
Double.parseDouble(df.format(y_inUji).replace(',', '.'));
                y_inUji =
Double.parseDouble(df.format(y_inUji + bb).replace(',',
'.')); //mencari nilai net

                infoLatih += " = " + y_inUji + "\n";
                haUji = fct.makeY(y_inUji, threshold); //
mencari output jaringan
                infoLatih += " Output Jaringan = " + haUji +
"\n";
                TextInfoUji.setText(infoLatih);
            }
            TextInfoUji.setText(infoLatih);
        } catch (Exception e) {
            //     JOptionPane.showMessageDialog(null,
"Terjadi Kesalahan Pengujian Pada : \n" + e);
        }
    }
```

d. Method Pembuatan Struktur Jaringan.

Method ini berfungsi untuk membuat struktur jaringan perceptron. Jumlah neuron berdasarkan pada jumlah variabel input. Berikut adalah implementasi method pembuatan struktur jaringan.

```
/*
 * DILARANG MENGHAPUS ATAU MENGEDIT COPYRIGHT INI.
 *
 * Copyright 2010 zhazha_moet@yahoo.co.id
 * All rights reserved.
 *
 * Semua isi dalam file ini adalah hak milik dari
zhazha_moet@yahoo.co.id
 * Anda tak diperkenankan untuk menggunakan file atau
mengubah file
 * ini kecuali anda tidak menghapus atau merubah lisence
ini.
 *
 * File ini dibuat menggunakan :
 * IDE         : NetBeans
 * NoteBook    : Compaq CQ41
 * OS          : Windows 7 Home Basic
 * Java        : Java 1.6
 */
package Core;

import java.awt.Font;
import java.awt.Graphics;

/**
 *
 * @author Liza Afriyanti
 */
public class bentukObjek extends javax.swing.JPanel {

    public int jum;

    public bentukObjek() {
    }

    public void setJumlah(int jumlah) {
        jum = jumlah;
    }

    public bentukObjek(int jml) {
        jum = jml;
    }

    @Override
    public void paint(Graphics g) {

        // atribute lingkaran/node
```

```
        int diameterX = 50;
        int diameterY = 50;
        int jarakAntarLingkaran = 30;

        //border
        int jarakPalingAtas = 40;
        int jarakPalingKiri = 90;

        //kotak y_in
        int sisiKotak = 50;
        int jarakLingDanKotak = 150;
        int mulaiKotakX = jarakPalingKiri + diameterX +
jarakLingDanKotak;
        int mulaiKotakY = (int) ((0.5 * ((jum * diameterY) +
((jum - 1) * jarakAntarLingkaran)) + jarakPalingAtas) - (0.5
* sisiKotak));

        //node dan garis
        int akhirGarisX = mulaiKotakX;
        int akhirGarisY = (int) (0.5 * ((jum * diameterY) +
((jum - 1) * jarakAntarLingkaran)) + jarakPalingAtas);

        for (int i = 0; i < jum; i++) {
            int jarakAtas = jarakPalingAtas + i * (diameterY
+ jarakAntarLingkaran);

            if (i == 0) {
                jarakAtas = jarakPalingAtas;
            }

            int mulaiLingkaranX = jarakPalingKiri;
            int mulaiLingkaranY = jarakAtas;

            g.drawOval(mulaiLingkaranX, mulaiLingkaranY,
diameterX, diameterY);
            g.drawString("X" + (i + 1), mulaiLingkaranX +
20, mulaiLingkaranY + 30);

            int mulaiGarisX = mulaiLingkaranX + diameterX;
            int mulaiGarisY = (int) (mulaiLingkaranY + (0.5
* diameterY));

            g.drawLine(mulaiGarisX, mulaiGarisY,
akhirGarisX, akhirGarisY);
            g.drawString("W" + (i + 1), mulaiGarisX + 15,
mulaiGarisY + 5);
        }

        g.drawRect(mulaiKotakX, mulaiKotakY, sisiKotak,
sisiKotak);
        g.setFont(new Font("", Font.PLAIN, 30));
        g.drawString("+", mulaiKotakX + 17, mulaiKotakY +
32);

        //garis y_in
        int mulaiGaris2X = mulaiKotakX + sisiKotak;
```

```
        int mulaiGaris2Y = (int) (mulaiKotakY + (0.5) *
sisiKotak);
        g.drawLine(mulaiGaris2X, mulaiGaris2Y, mulaiGaris2X
+ 100, mulaiGaris2Y);
        g.setFont(new Font("Arial", Font.PLAIN, 12));
        g.drawString("Net", mulaiGaris2X + 30, mulaiGaris2Y
- 4);

        //kotak fungsi aktifasi
        g.drawRect(mulaiKotakX + sisiKotak + 100,
mulaiKotakY, sisiKotak, sisiKotak);

        //lingkaran bawah/ bias
        g.drawOval(mulaiKotakX + 4, mulaiKotakY + 100,
sisiKotak - 15, sisiKotak - 15);
        g.drawString("1", mulaiKotakX + 20, mulaiKotakY +
120);

        //garis ke 3/ garis bias
        int mulaiGaris3X = (int) (mulaiKotakX + (0.5 *
sisiKotak));
        int mulaiGaris3Y = mulaiKotakY + sisiKotak;
        g.drawLine(mulaiGaris3X, mulaiGaris3Y, mulaiGaris3X,
mulaiGaris3Y + 100 - sisiKotak);
        g.drawString("bias", mulaiGaris3X + 5, mulaiGaris3Y
+ 30);

        //garis ke 4/ output
        int mulaiGaris4X = mulaiKotakX + sisiKotak + 100 +
sisiKotak;
        int mulaiGaris4Y = (int) (mulaiKotakY + (0.5) *
sisiKotak);
        g.drawLine(mulaiGaris4X, mulaiGaris4Y, mulaiGaris4X
+ 100, mulaiGaris4Y);
        g.drawString("y = f(net)", mulaiGaris4X + 30,
mulaiGaris4Y - 4);
        g.drawString("F", mulaiGaris4X - 30, mulaiGaris4Y);
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new bentukObjek(4).setVisible(true);
            }
        });
    }
}
```

BAB V
PENGUJIAN

5.1 Pengujian Sistem

Sebelum sistem diterapkan pada lingkungan sebenarnya, maka diperlukan evaluasi/pengujian terhadap berbagai aspek. Pengujian ini dilakukan agar kemungkinan terjadinya kesalahan/*error* pada sistem dapat diidentifikasi sejak awal. Pada tahap pengujian dan analisis *tool* untuk Jaringan Syaraf Tiruan (JST) model Perceptron, dilakukan pembandingan antara kebenaran serta kesesuaian program dengan kebutuhan sistem.

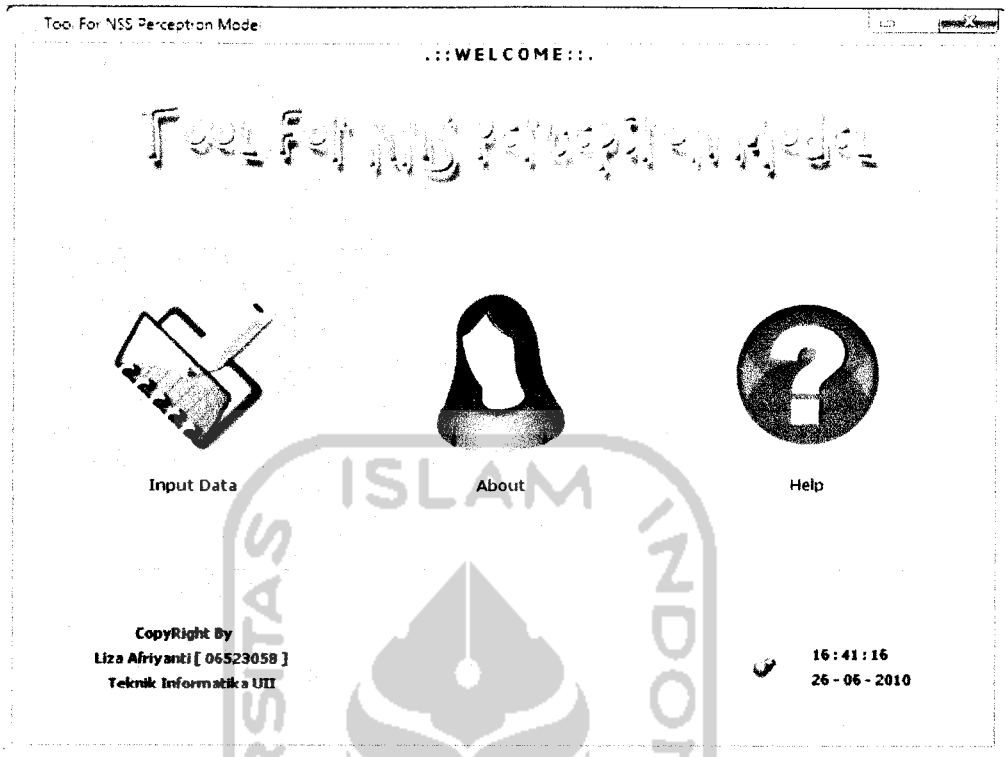
Pengujian dilakukan dengan mengisikan *input* ke dalam form-form yang sudah dijelaskan pada BAB IV. Metode pengujian dengan melakukan pengujian normal dan pengujian tidak normal.

5.1.1 Pengujian Normal

Dilakukan dengan memberikan masukan yang menurut spesifikasi awal dan pengetahuan yang diijinkan. Setelah diberikan masukan yang sesuai, dilakukan analisis pembandingan antara kebenaran masukan serta kesesuaian program dengan kebutuhan sistem.

a. Halaman *Home*

Menu ini adalah menu untuk tampilan utama. Menu terdiri dari tombol pilihan *Input Data*, *About* dan *Help*. Tampilan pada sistem dapat dilihat pada gambar 5.1.

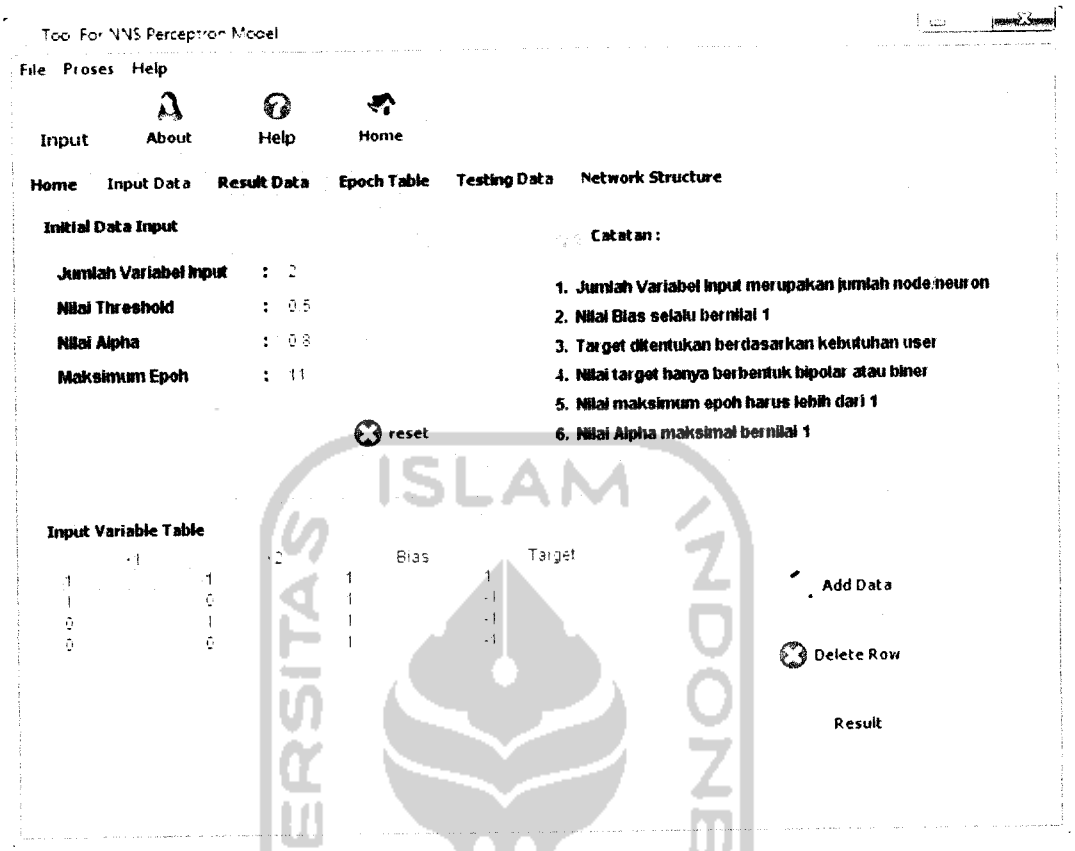


Gambar 5. 1 Tampilan Halaman *Home*

b. Hamalan *Input Data* Awal

Menu ini adalah menu untuk melakukan *input* data awal. Untuk melakukan proses pelatihan dan pengujian perceptron, langkah pertama yang harus dilakukan oleh *user* adalah memasukkan data awal seperti jumlah variabel *input*, nilai variabel *input*, alpha (*learning rate*), threshold, maksimum epoh dan target (*output*).

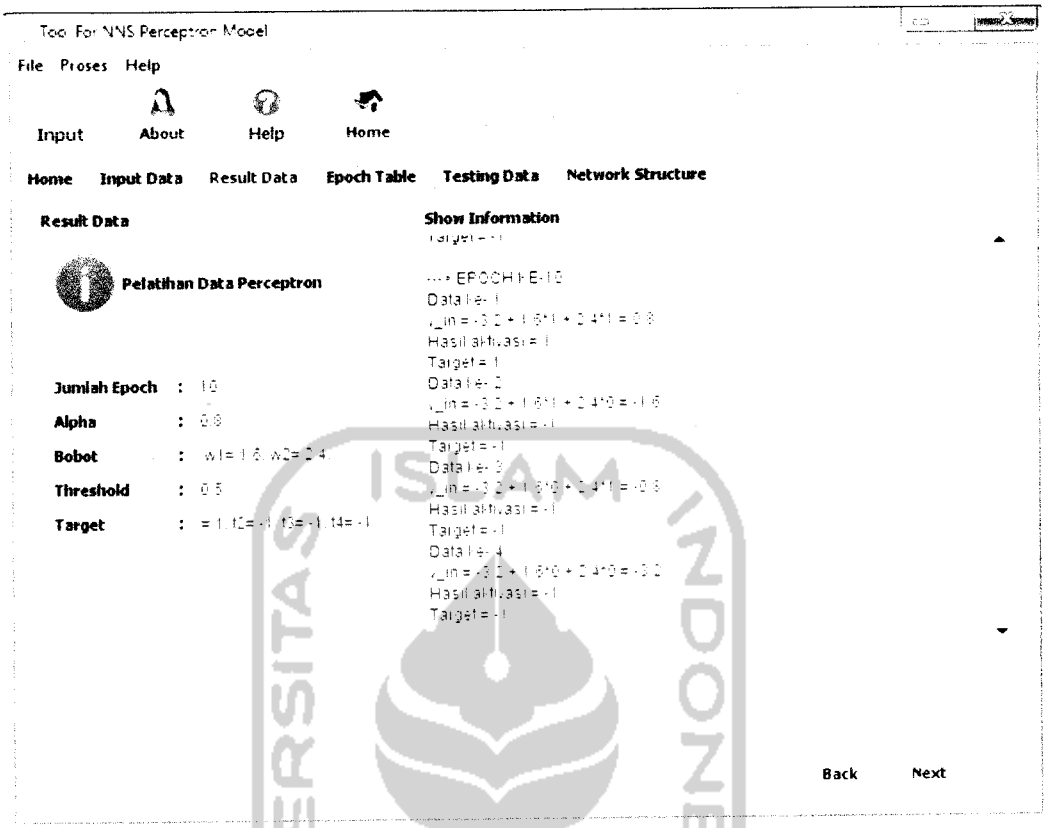
Misalkan untuk inisialisasi awal tiap parameter yang digunakan adalah jumlah variabel *input* (x) = 2, bobot awal (w) = 0, alpha (α) = 0.8, threshold (θ) = 0.5 dan maksimum epoh adalah 11 dengan *input* biner dan target bipolar. Tampilan *input* data awal dapat dilihat pada gambar 5.2.



Gambar 5. 2 Tampilan *Input* Data Awal

c. Halaman *Result data*

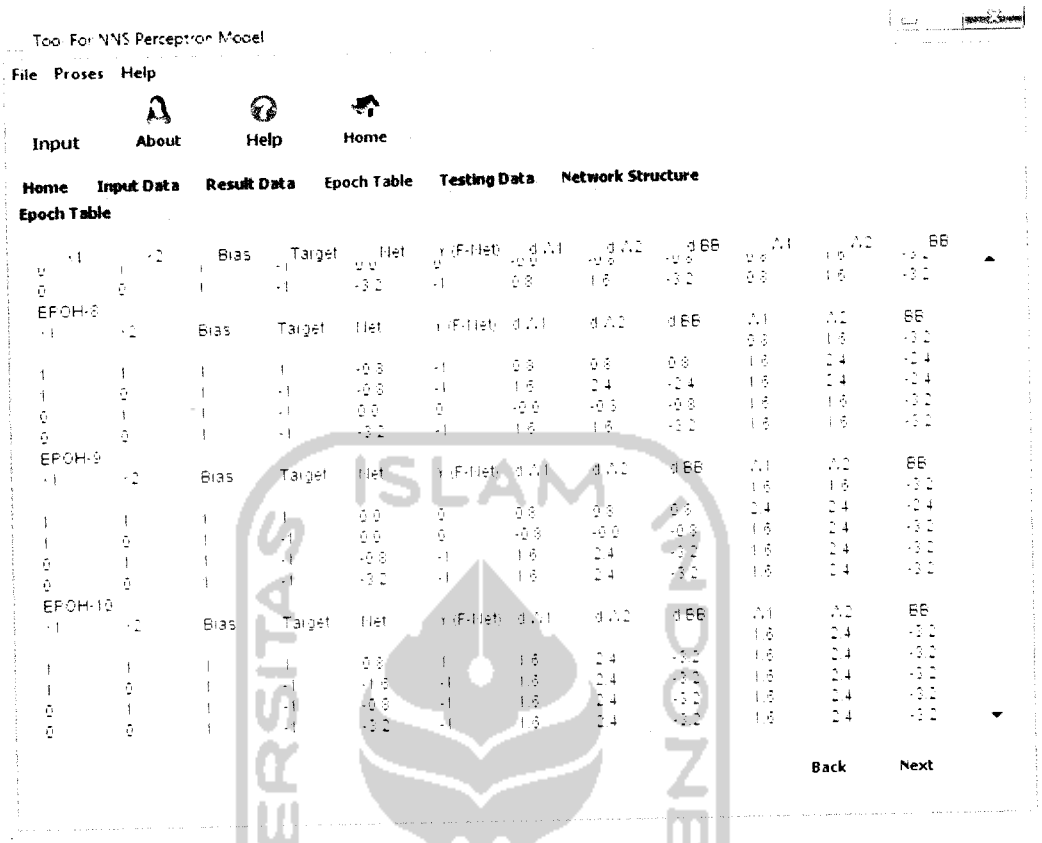
Setelah *user* adalah memasukkan data awal seperti jumlah variabel *input*, nilai variabel *input*, alpha (*learning rate*), threshold, maksimum epoh dan target (*output*). *User* dapat mengklik tombol *Next*. Kemudian sistem akan melakukan proses pelatihan data dan menampilkan hasilnya pada text area *show information*. Tampilan *result data* dapat dilihat pada gambar 5.3.



Gambar 5. 3 Tampilan *Result data*

d. Halaman *Epoch Table*

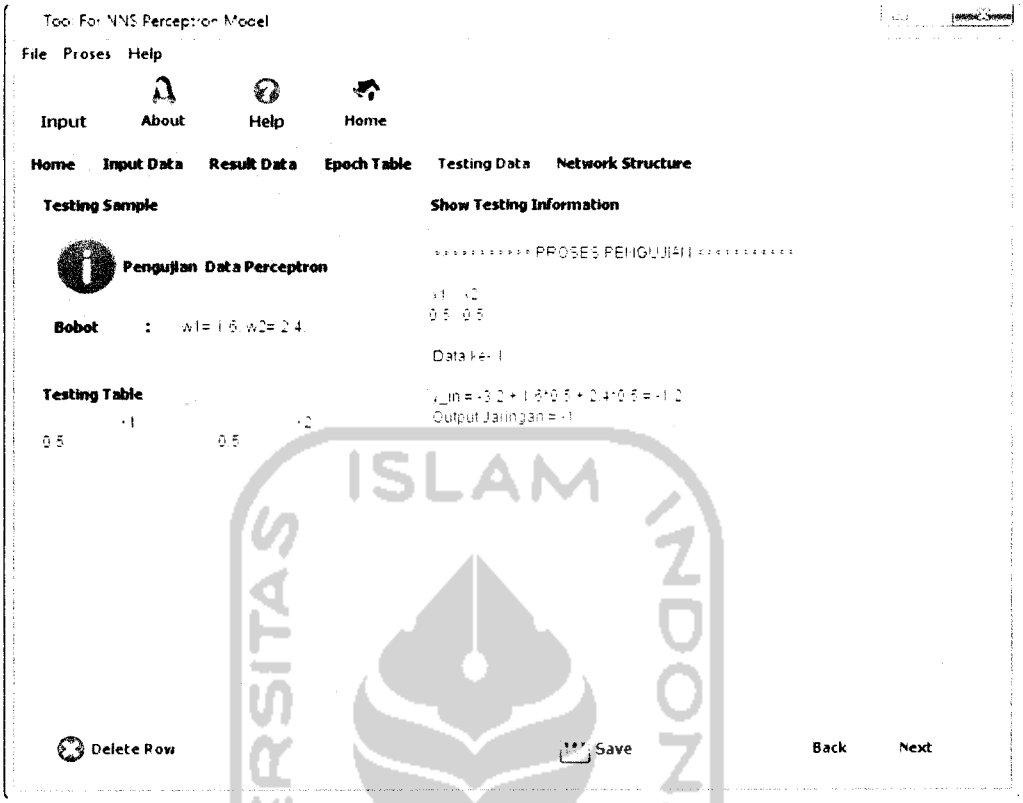
Setelah sistem melakukan proses pelatihan data, sistem akan menampilkan perhitungan pelatihan data epoh tiap iterasi pada halaman *epoch table*. Tampilan *epoch table* dapat dilihat pada gambar 5.4.



Gambar 5. 4 Tampilan *Epoh Table*

e. Halaman *Testing Data*

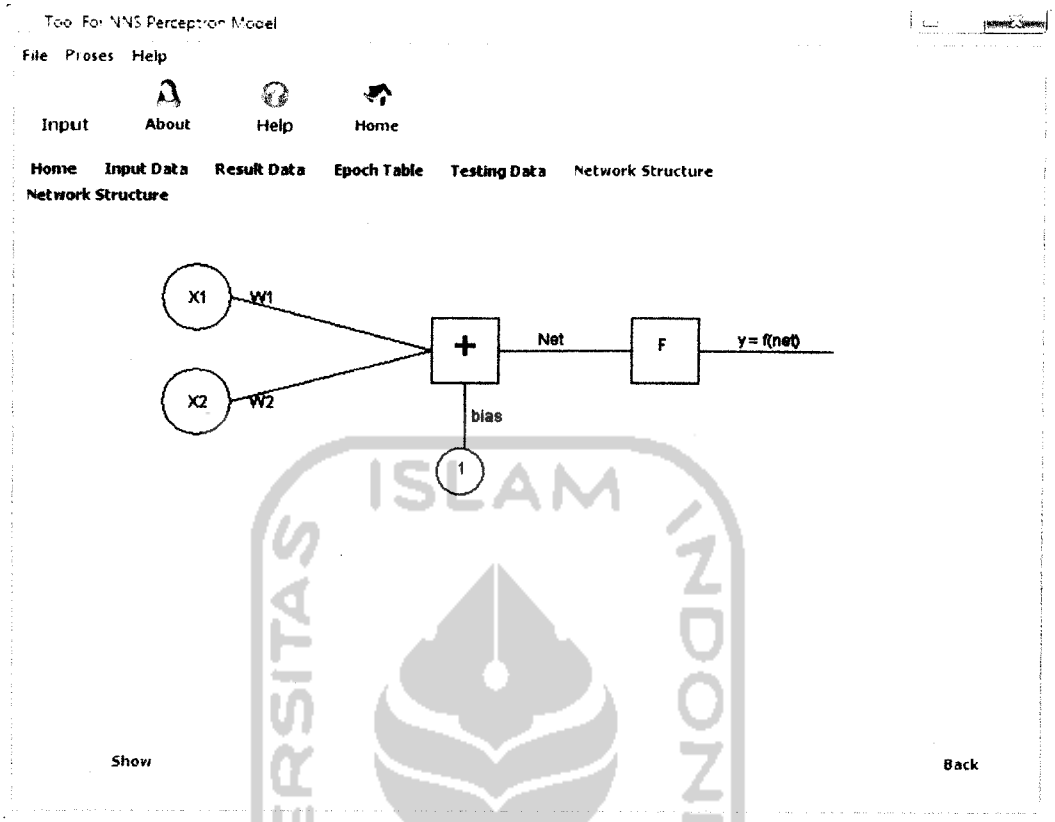
User dapat melakukan proses pengujian data pada sistem setelah melakukan proses pelatihan data. Pada halaman *testing data* terdapat tabel untuk memasukkan nilai variabel untuk proses pengujian. Nilai bobot pada proses pengujian didapat dari hasil proses pelatihan. Kemudian sistem akan menampilkan hasil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



Gambar 5. 5 Tampilan *Testing Data*

f. Halaman *Network Structure*

Pada halaman ini *user* dapat melihat struktur jaringan. Struktur jaringan terbentuk berdasarkan *input* data awal. Jumlah neuron terbentuk berdasarkan jumlah variabel *input* yang dimasukkan oleh *user*. Panel *network structure* hanya dapat menampilkan neuron kurang dari 5, akan tetapi *user* dapat mengklik tombol *show* jika memasukkan jumlah neuron lebih dari 4 untuk menampilkan form *network struktur* secara lengkap. Tampilan *network structure* dapat dilihat pada gambar 5.6.

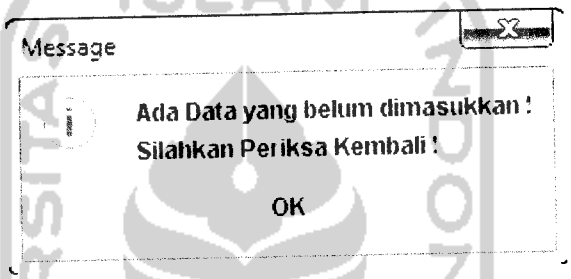


Gambar 5. 6 Tampilan *Network Structure*

5.1.2 Pengujian Tidak Normal

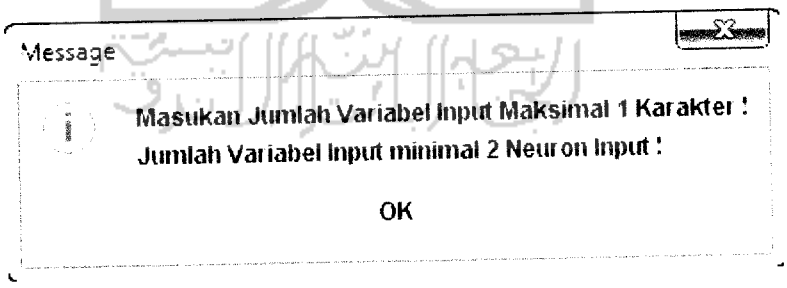
Dilakukan dengan memberikan masukan dengan spesifikasi yang tidak diijinkan sehingga sistem akan memberikan reaksi lain. Reaksi sistem berupa berupa peringatan *(alert)* atau penanganan kesalahan *(error handling)*.

Penanganan kesalahan ini dilakukan untuk menangkap error yang terjadi ketika salah satu *field* pada form masukan kosong atau ketidaksesuaian panjang karakter. Contoh penanganan kesalahan *input* terdapat salah satu *field* kosong, maka akan muncul peringatan seperti pada gambar 5.7.



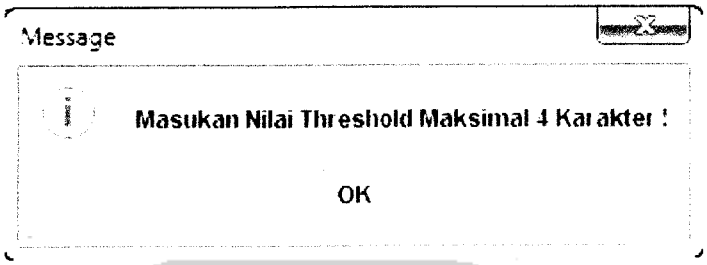
Gambar 5. 7 Peringatan Jika *Field* Kosong

Contoh penanganan kesalahan *input* jumlah variabel kurang dari 2, maka akan muncul peringatan seperti pada gambar 5.8.



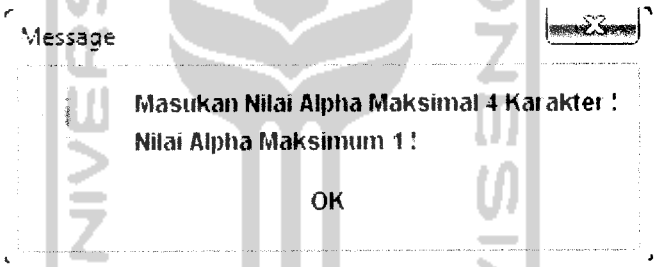
Gambar 5. 8 Peringatan Jika Jumlah *Input* Variabel Kurang dari 2

Contoh penanganan kesalahan *input* jumlah karakter untuk *field* nilai threshold lebih dari 4, maka akan muncul peringatan seperti pada gambar 5.9.



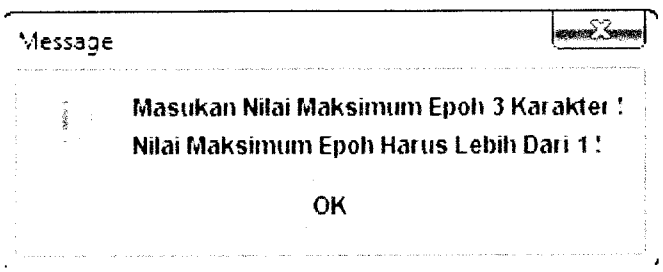
Gambar 5. 9 Peringatan Jumlah Karakter Nilai Threshold

Contoh penanganan kesalahan *input* untuk *field* nilai alpha lebih dari 1, maka akan muncul peringatan seperti pada gambar 5.10.



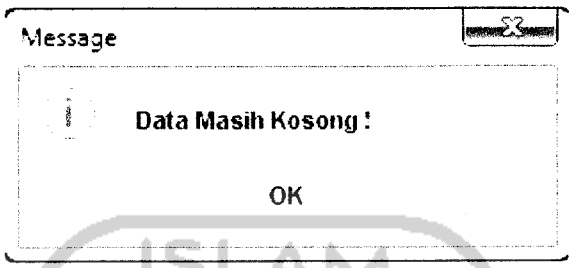
Gambar 5. 10 Peringatan Nilai Alpha Maksimum

Contoh penanganan kesalahan *input* untuk *field* nilai maksimum epoh harus lebih dari 1, maka akan muncul peringatan seperti pada gambar 5.11.



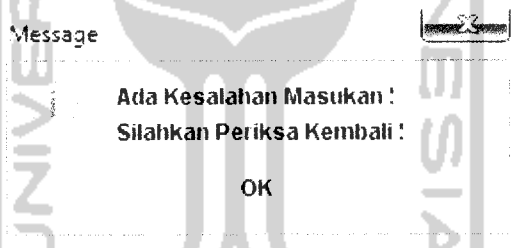
Gambar 5. 11 Peringatan Nilai Epoh Maksimum

Contoh penanganan kesalahan jika data pelatihan dan pengujian masih kosong ketika *user* ingin mencetak hasil perhitungan, maka akan muncul peringatan seperti pada gambar 5.12.



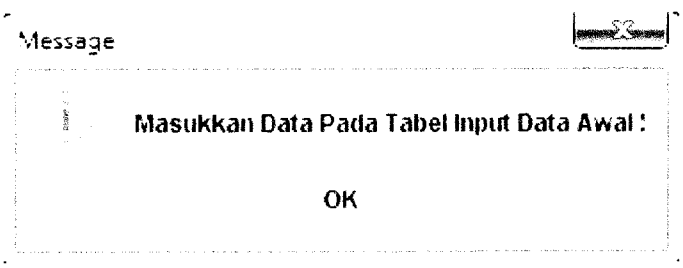
Gambar 5. 12 Peringatan Cetak Data

Contoh penanganan kesalahan salah satu masukan data adalah String, maka akan muncul peringatan seperti pada gambar 5.13.



Gambar 5. 13 Peringatan *Input* Data Bertipe Data String

Contoh penanganan kesalahan salah satu masukan pada tabel *input* data adalah String, maka akan muncul peringatan seperti pada gambar 5.14.



Gambar 5. 14 Peringatan *Input* Data Tabel Bertipe Data String

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.

DAFTAR PUSTAKA

[HAR03] Hariyanto, *Bambang.Esensi-esensi Bahasa Pemrograman JAVA.* Penerbit Informatika. Bandung. 2003.

[HAK09] Hakim, Rachmad. *Mastering Java (Konsep Pemrogramana Java dan Penerapannya untuk membuat software aplikasi).* Elex Media Komputindo. 2009.

[KUS03] Kusumadewi, Sri. *Artificial Intelligence(teknik dan aplikasinya).* Graha Ilmu, Yogyakarta. 2003.

[MAR07] Marbuko, Cholid dan Achmadi, Abu Haji. *Metodologi Penelitian : Memberikan bekal teoritis pada mahasiswa tentang metodologi penelitian serta diharapkan dapat melaksanakan penelitian dengan langkah-langkah yang benar*. Penerbit Bumi Aksara. Jakarta. 2007.

[NAS09] Nasution, Fithlail Gudie. *Jaringan saraf tiruan (Neural Network).* http://fithlail.web.id/category/neuralnetwork/ diakses tanggal 17 Maret 2009.

[RIC02] Rickyanto, Isak. *Dasar Pemrograman Berorientasi Objek Dengan JAVA 2(JDK 1.4)*. Penerbit Andi. Yogyakarta. 2002.

[SIA05] Siang, Jong Jek. *Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan Matlab*. Penerbit Andi. Yogyakarta. 2005.

LAMPIRAN

FORM-KP TF-A1

UNIVERSITAS ISLAM INDONESIA
Jurusan Teknik Informatika FTI

No.:

USULAN PENELITIAN TUGAS AKHIR

Nama : Liza Afriyanti No. Mhs : 06523058
Contact. : 085265732993 (alamat, telp., email, untuk kemudahan menghubungi
zhazha_moet@yahoo.co.id jika ada perubahan contact mohon menghubungi jurusan)

A. JUDUL / TOPIK

Rancang Bangun Tool untuk Jaringan Syaraf Tiruan Model Perceptron

B. MASALAH

1. Latar Belakang

Jaringan Syaraf Tiruan (JST) merupakan topik yang hangat dibicarakan dan mengundang banyak kekaguman dalam dasa warsa terakhir. Hal ini disebabkan karena kemampuan JST untuk meniru sifat sistem yang di-*input*-kan. Pada dasarnya JST mencoba meniru cara kerja otak makhluk hidup, yaitu bentuk *neuron*-nya (sel syaraf). Faktor kecerdasan dari syaraf tidak ditentukan di dalam sel tetapi terletak pada bentuk dan topologi jaringannya.

Salah satu permasalahan dalam JST yang 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 handal yang dapat membantu *user* dalam memahami struktur JST model Perceptron.

Salah satu bahasa pemrograman yang banyak digunakan adalah Java. Java telah menerapkan konsep pemrograman berorientasi objek yang modern dalam implementasinya. Dengan kemudahannya, Java dapat digunakan dalam perancangan pembuatan tool untuk JST model Perceptron.

2. Rumusan Masalah

Bagaimana membangun aplikasi untuk mengimplementasikan struktur Jaringan Syaraf Tiruan model Perceptron dengan menggunakan Java.

3. Batasan Masalah

a. Aplikasi ini dibuat untuk dijalankan pada desktop.
b. Program yang akan dibuat nantinya akan menampilkan struktur JST model Perceptron.
c. Perceptron pada JST memiliki bobot yang bisa diatur dan suatu nilai ambang (*threshold*).

C. PENYELESAIAN MASALAH

1. Usulan

Membangun aplikasi untuk mengimplementasikan struktur JST model Perceptron dengan menggunakan Java, meliputi :

a. Pembuatan Neuron.
b. *Entry* data.
c. Membangun struktur Jaringan Syaraf Tiruan.
d. Membangun algoritma pembelajaran.
e. Proses pengujian.

2. Langkah Penyelesaian

1. Analisis jaringan.
2. Desain struktur Jaringan Syaraf Tituan.
3. Desain visual jarigan.
4. *Coding.*
5. Pengujian sistem.

D. DAFTAR PUSTAKA UTAMA

1. Siang, Jong Jek. 2005. *Jaringan Syaraf Tiruan & Pemrogramannya Menggunakan Matlab.* Yogyakarta : Andi.
2. Sri Kusumadewi, Beni Suranto. 2009. *Modul Praktikum Pemrograman Berorientasi Objek.* Yogyakarta : Laboratorium KSC, Jurusan Teknik Informatika, UII.
3. Wahana Komputer. 2006. *Membuat Aplikasi Database dengan Java 2.* Yogyakarta : Andi.

E. USULAN DOSEN PEMBIMBING

Dr. Sri Kusumadewi, S.Si., MT.

Yogyakarta, 13 Oktober 2009

Nama : Liza Afriyanti

UNIVERSITAS ISLAM INDONESIA
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK INFORMATIKA
JL. KALI URANG KM.14, 5 JOGJAKARTA

PRESENSI PRESENTASI KEMAJUAN TA
LAMPIRAN USULAN TA

Nama : LIZA AFRIYANTI
No. Mahasiswa : 06523058

nggal	Judul TA	No. Mahasiswa	Nama Mhs Yg Presentasi	T. T Dosen*
-09	alat bantu ajar bagi & fungsi trimbuhan utk SD	05523252	Firda Zulvia Abrahams	
-09	Sistem pembuatan jdwl pelajaran dan skln scr otomatis dg algen	05523006	ALi nfni	
-9	penggunaan RGSS dlm pembuatan menu game sistem pd RPG maker xp	05523123	fazri azis	
-09	alat bantu ajar sains utk SD menggunakan bilingual basis multimedia	0552308	Oka pujianto	
-09	aplikasi denah ruangan pasien rawat inap di RS Islam Kartasi	05523070	Zesyar Erlinda	
-09	game EPS menggunakan Java utk keperluan outdoor	05523033	Ari Magetan W	

Catatan :
.....
.....

Jogjakarta, 13-10-2009
Ka. Prodi T. Informatika

Yudi Prayudi, S,Si, M.Kom

UNIVERSITAS ISLAM INDONESIA
Jurusan Teknik Informatika FTI

SARAN/USULAN PRESENTASI KEMAJUAN TUGAS AKHIR

Nama Mhs. : Liza afriyanti
No. Mhs. : 06523058
Judul TA : Rancang Bangun Tool untuk Jaringan Syaraf tiruan .

lanjutkan !! !

Nilai kemajuan Tugas Akhir: _____ (0 - 100)
(studi pustaka, perancangan, penguasaan materi, ketepatan)

Yogyakarta, 12 Jan 10

Dosen,

(Azmi Arifyah)
(nama terang)

Dilampirkan pada Laporan TA yang diajukan untuk pendadaran

UNIVERSITAS ISLAM INDONESIA
Jurusan Teknik Informatika FTI

SARAN/USULAN PRESENTASI KEMAJUAN TUGAS AKHIR

Nama Mhs. : Liza Afriyanti
No. Mhs. : 06 523 058
Judul TA : Rancang Bangun Tool untuk JST Model Perceptron

- Batasan Masalah.
- revisi poin2nya

- Aplikasi
- lanjutkan, karena sdh masuk bln ke-3

Nilai kemajuan Tugas Akhir: _____ (0 - 100)
(studi pustaka, perancangan, penguasaan materi, ketepatan)

Yogyakarta, 12 Jan 2010

Dosen,

Ridho. R.
(nama terang)

Dilampirkan pada Laporan TA yang diajukan untuk pendadaran