PENERAPAN JARINGAN SYARAF TIRUAN *RESILIENT PROPAGATION* UNTUK DIAGNOSIS PENYAKIT DIABETES MELLITUS

SKRIPSI

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh :

SANDRODHIAZ VIRGIANDHANA

NIM.0710963041

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

PENERAPAN JARINGAN SYARAF TIRUAN *RESILIENT PROPAGATION* UNTUK DIAGNOSIS PENYAKIT DIABETES MELLITUS

SKRIPSI

Disusun oleh :

Sandrodhiaz Virgiandhana

NIM. 0710963041

Telah diperiksa dan disetujui oleh :

Pembimbing I,	Pembimbing II,
Drs. Achmad Ridok, M.Kom	**Nanang Yudi Setiawan, ST**
NIP. 196808251994031002	**NIP. 197711142003122001**

LEMBAR PENGESAHAN

PENERAPAN JARINGAN SYARAF TIRUAN *RESILIENT PROPAGATION* UNTUK DIAGNOSIS PENYAKIT DIABETES MELLITUS

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

SANDRODHIAZ VIRGIANDHANA

NIM. 0710963041

Skripsi ini telah diuji dan dinyatakan lulus tanggal 4 Januari 2013

Penguji I	Penguji II
Drs. Marji, M. T.	**Rekyan Regasari MP., ST.,MT.**
NIP. 196708011992031001	**NIK. 77041406120253**

Penguji III

Indriati, ST.,M.Kom
NIK. 83101306120035

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, MT.
NIP. 196708011992031001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 4 Januari 2013
Mahasiswa,

Sandrodhiaz Virgiandhana
NIM. 0710963041

PENERAPAN JARINGAN SYARAF TIRUAN *RESILIENT PROPAGATION* UNTUK DIAGNOSIS PENYAKIT DIABETES MELLITUS

ABSTRAK

Diabetes Mellitus (DM) merupakan masalah kesehatan yang sedang berkembang pesat di Indonesia. Ini merupakan merupakan ancaman serius bagi pembangunan kesehatan di Indonesia. Organisasi Kesehatan Dunia (World Health Organisation/WHO) memperkirakan jumlah penyandang diabetes di Indonesia akan melonjak drastis. Untuk menekan pertumbuhan penyakit ini dilakukan suatu penelitian. Pada penelitian ini mengimplementasikan sebuah sistem untuk mendiagnosis DM menggunakan algoritma Jaringan Saraf Tiruan (JST) *Resilient Propagation.* Dari penelitian Jaringan Saraf Tiruan mampu memprediksi penyakit DM dengan keakuratan sebesar 96.66%. Nilai akurasi ini diperoleh dari struktur JST dengan jumlah neuron pada *input layer* sebanyak 4 unit, pada *hidden layer* sebanyak 70 unit, jumlah neuron pada *output layer* sebanyak satu unit, nilai *learning rate* sebesar 0.9, dan maksimum *epoch* sebanyak 5000 dengan nilai MSE terkecil sebesar 0.0000999732406311761.

Kata Kunci : Diabetes Mellitus, *Resilient Propagation*, Jaringan Saraf Tiruan

APPLICATION OF ARTIFICIAL NEURAL NETWORK RESILIENT PROPAGATION FOR THE DIAGNOSIS OF DIABETES MELLITUS

ABSTRACT

Diabetes Mellitus (DM) is a health problem that is growing rapidly in Indonesia. This is a serious threat to health development in Indonesia. The World Health Organization (WHO) estimates the number of people with diabetes will rise dramatically in Indonesia. To suppress the growth of this disease conducted a study. In this study implements an application for diagnosing diabetes using Artificial Neural Network algorithm (ANN) Resilient Propagation. This study were obtained from ANN are able to predict disease DM formed with accuracy of 96.66%. Accuracy value is obtained from structure with the number of neurons in the input layer 4 units, the hidden layer as many as 70 units, the number of neurons in the output layer by one unit, the value of learning rate of 0.9, and the maximum epoch of 5000 with the smallest MSE value of 0.00009997324063117 61.

Key Word : Diabetes Mellitus, Resilient Propagation, Artificial Neural Network

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, hanya dengan rahmat dan karunia yang telah diberikan kepada penulis, sehingga dapat menyelesaikan skripsi yang berjudul "Penerapan Jaringan Syaraf Tiruan Resilient Propagation Untuk Diagnosis Penyakit Diabetes Mellitus".

Skripsi ini merupakan salah satu syarat untuk memenuhi persyaratan akademis untuk menyelesaikan studi di program Sarjana Ilmu Komputer Universitas Brawijaya.

Dalam penyelesaian tugas akhir ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Drs. Achmad Ridok, M.Kom, selaku dosen pembimbing utama. Terima kasih atas arahan serta bimbingannya dalam penyusunan skripsi ini,
2. Nanang Yudi S.,ST, selaku dosen pembimbing pendamping atas arahan serta bimbingannya dalam penyusunan skripsi ini.
3. Drs. Marji, M.T, selaku Ketua Program Studi Ilmu Komputer, Program Teknik Infomatika dan Ilmu Komputer.
4. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer, Program Teknik Infomatika dan Ilmu Komputer.
5. Segenap staf dan karyawan Program Teknik Infomatika dan Ilmu Komputer. Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan skripsi ini.
6. Orang tua Penulis yaitu Ibu dari Penulis atas segala doa restu, dukungan moril dan materiil kepada Penulis.
7. Rekan Alief Ramadanie yang telah banyak memberi dukungannya dalam menyelesaikan tugas akhir ini.
8. Rekan-rekan Ilmu Komputer Universitas Brawijaya yang telah memberikan dukungannya dalam penyusunan tugas akhir ini.

9. Semua pihak yang telah membantu terselesaikannya laporan ini yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa laporan tugas akhir ini tentunya tidak terlepas dari berbagai kekurangan dan kesalahan. Oleh karena itu, segala kritik dan saran yang bersifat membangun sangat Penulis harapkan dari berbagai pihak guna peningkatan kualitas penelitian
serupa di masa mendatang.

Malang, 4 Januari 2013

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SOURCE CODE

BAB I
PENDAHULUAN

1.1. Latar Belakang

Kesehatan merupakan hal yang begitu penting bagi manusia. Ironisnya banyak sekali penyakit-penyakit yang pada akhirnya terlambat didiagnosis sehingga mencapai tahap kronis yang membuatnya sulit untuk ditangani. Salah satunya yaitu penyakit Diabetes. Diabetes Mellitus (DM) merupakan masalah kesehatan yang sedang berkembang pesat di dunia. Sekarang ini diperkirakan terdapat kira-kira 171 - 194 juta orang di dunia ini menderita karena diabetes. Jumlah penderita diabetes diperkirakan akan meningkat lebih dari 330 juta pada tahun 2025. [KAH – 06]

Pernyataan umum dari diabetes yaitu dicirikan oleh produksi insulin yang tidak memadai dari pankreas atau ketidakefektifan penggunaan insulin yang diproduksi oleh pankreas (hyperglemia) [RAD – 07]. Hal tersebut menyebabkan obesitas, hipertensi, peningkatan kadar kolesterol dan penyakit lain yang merupakan prevalensi dari diabetes. Pengembangan masalah penyakit ginjal, kebutaan dan jantung koroner merupakan beberapa penyakit yang diakibatkan dari perawatan yang tidak tepat dan keterlambatan diagnosis diabetes.

Pada penelitian untuk diagnosis diabetes ini akan menggunakan metode kecerdasan buatan atau *Artificial Intelligence* (AI) yang sebelumnya telah banyak digunakan untuk kasus – kasus yang lain.

Saat ini Kecerdasan Buatan atau *Artificial Intelligence* (AI) telah banyak diterapkan diberbagai bidang, terutama dalam dunia medis. Teknologinya yang mengadopsi proses dan cara pikir manusia dalam pengambilan keputusan sangat membantu untuk penanganan masalah lebih lanjut. Salah satu cabang AI adalah Jaringan Saraf Tiruan atau *Artificial Neural Network*. Jaringan saraf tiruan merupakan salah satu sistem pemrosesan informasi yang didesain dengan menirukan cara kerja otak manusia dalam menyelesaikan suatu masalah dengan melakukan proses belajar melalui perubahan bobot sinapsisnya. Dengan menggunakan jaringan saraf tiruan dapat dilakukan metode pendekatan pembelajaran atau pelatihan untuk mengidentifikasi pola data dari sistem

diagnosis penyakit diabetes. Jaringan saraf tiruan memiliki kemampuan melakukan komputasi secara paralel dengan cara belajar dari pola-pola yang diajarkan. Dengan kemampuan tersebut jaringan saraf tiruan dapat melakukan regresi nonlinier terhadap pola-pola parameter penyakit, sehingga mampu memperkirakan risiko penyakit yang diderita secara lebih akurat. Terdapat tiga kelas pokok dari arsitektur jaringan saraf tiruan antara lain *single-layer feed-forward neural network, recurrent neural network*,dan *multi-layer feed-forward neural network* (*multi-layer perceptron*) [SRI – 09].

Pada penelitian sebelumnya untuk mendiagnosa suatu penyakit telah banyak dilakukan didalam dunia medis, termasuk di antaranya adalah mendiagnosis penyakit diabetes. Seperti penelitian yang dilakukan oleh Yoga (2010) yang mendiagnosa diabetes menggunakan *Fuzzy Database* dan memiliki keakuratan mencapai 86,7%.

Pada penelitian ini akan menggunakan algoritma *resilient propagation* (RPROP) yang merupakan perkembangan dari algoritma Backpropagation. Penelitian menggunakan algoritma resilient propagation sebelumnya telah dilakukan untuk pengenalan kata oleh Prameswari (2011) dan memiliki keakuratan yang tinggi yaitu mencapai 90%. Oleh karena itu, untuk mendiagnosis penyakit Diabetes menggunakan resilient propagation diharapkan dapat mencapai nilai keakuratan yang lebih baik dari metode yang digunakan sebelumnya.

Berdasarkan latar belakang tersebut, maka penulis mengambil judul pada skripsi ini **"Penerapan Jaringan Syaraf Tiruan Resilient Propagation Untuk Diagnosis Penyakit Diabetes Mellitus"**.

1.2 Rumusan Masalah

1. Bagaimana mengimplementasikan algoritma *Resilient Propagation* untuk mendiagnosa penyakit diabetes mellitus?
2. Bagaimana tingkat akurasi diagnosis sistem jika digunakan untuk mendiagnosa penyakit diabetes mellitus?

1.3 Batasan Masalah

Batasan masalah pada skripsi ini adalah :

1. Parameter atau indikasi yang digunakan yaitu kadar glukosa dalam darah, kolesterol, trigliserida, dan Ureum.
2. Tidak menangani data yang memiliki *missing value* pada data latih maupun data uji.
3. Pembanding diagnosa sistem adalah hasil diagnosa pakar.

1.4 Tujuan

Tujuan pembuatan skripsi ini adalah :

1. Menerapkan algoritma *Resilient Propagation* untuk mendiagnosa penyakit diabetes berdasarkan input parameter yang diberikan.
2. Mengetahui tingkat akurasi sistem jika digunakan untuk mendiagnosa penyakit diabetes.

1.5 Manfaat

Manfaat yang dapat diambil dari skripsi ini adalah Menghasilkan suatu aplikasi komputer yang dapat memberikan diagnosis pasien tersebut menderita penyakit diabetes atau tidak.

1.6 Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan skripsi ini adalah :

1. Studi Literatur

   Mempelajari tentang penyakit diabetes mellitus beserta faktor-faktor risikonya dan mempelajari teori jaringan saraf tiruan terutama yang berhubungan dengan algoritma *Resilient Propagation* dari berbagai referensi.
2. Pendefinisian dan Analisis Masalah

   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan Pembuatan Program

   Membuat perancangan perangkat lunak dengan analisis terstruktur dan mengimplementasikan hasil rancangan tersebut dalam suatu program computer

4. Uji Coba dan Analisa Hasil Implementasi

   Menguji perangkat lunak pada beberapa data yang berbeda dan menganalisa hasil dari implementasi tersebut kemudian melakukan evaluasi.

5. Penyusunan Laporan

   Membuat laporan tertulis mengenai hasil skripsi ini.

1.7 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut :

1. **BAB I PENDAHULUAN**

   Berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi pemecahan masalah, dan sistematika penulisan.

2. **BAB II TINJAUAN PUSTAKA**

   Menguraikan teori-teori yang berhubungan dengan penyakit diabetes mellitus dan algoritma *Resilient Propagation*.

3. **BAB III METODOLOGI DAN PERANCANGAN SISTEM**

   Pada bab ini akan dijelaskan mengenai metode-metode yang digunakan dalam mengimplementasikan algoritma *Resilient Propagation* untuk diagnosis risiko penyakit diabetes mellitus

4. **BAB IV PEMBAHASAN**

   Dalam bab ini dijelaskan mengenai implementasi program untuk diagnosis penyakit diabetes mellitus bedasarkan faktor risiko dan pengujian keakuratan sistem serta analisis sistem perangkat lunak yang dibangun.

5. **BAB V KESIMPULAN DAN SARAN**

   Berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan pembangunannya.

BAB II
TINJAUAN PUSTAKA

2.1 Jaringan Saraf Tiruan

2.1.1 Definisi

Jaringan saraf tiruan (*artificial neural network*) atau disingkat JST adalah sistem komputasi dimana arsitektur dan operasi diilhami dari pengetahuan tentang sel syaraf biologi di dalam otak [KRI – 04]. Jaringan saraf tiruan merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran [KUS – 03].

Menurut [HEC – 88] mendefinisikan bahwa jaringan syaraf tiruan adalah suatu struktur pemroses informasi yang terdistribusi dan bekerja secara paralel, yang terdiri atas elemen pemroses (yang memiliki memori lokal dan beroperasi dengan informasi lokal) yang diinterkoneksi bersama dengan alur sinyal searah yang disebut koneksi. Setiap elemen pemroses memiliki koneksi keluaran tunggal yang bercabang ke sejumlah koneksi kolateral yang diinginkan (setiap koneksi membawa sinyal yang sama dari keluaran elemen pemroses tersebut). Keluaran dari elemen pemroses tersebut dapat merupakan sebarang jenis persamaan matematis yang diinginkan. Seluruh proses yang berlangsung pada setiap elemen pemroses harus benar-benar dilakukan secara lokal, yaitu keluaran hanya bergantung pada nilai masukan pada saat itu yang diperoleh melalui koneksi dan nilai yang tersimpan dalam memori lokal.

Jaringan saraf tiruan digunakan untuk menyelesaikan masalah-masalah yang kompleks dan sulit dipahami, dimana sejumlah besar data mengenai masalah tersebut telah dikumpulkan. Jaringan saraf tiruan mencari pola dan hubungan dalam data yang sangat besar yang terlalu rumit dan sulit untuk dianalisis manusia. Jaringan saraf tiruan menemukan pengetahuan ini dengan menggunakan perangkat keras dan peranti lunak yang menyerupai pola-pola pemrosesan dalam otak manusia. Jaringan saraf tiruan “mempelajari” pola-pola dari jumlah data yang

banyak dengan menyaring data, mencari hubungan, membangun model, dan mengoreksi kesalahan model itu sendiri berkali-kali. Jaringan saraf tiruan memiliki sejumlah besar *node* perasa dan pemroses yang secara kontinu berinteraksi satu sama lain. Selain itu, jaringan saraf tiruan dapat mengorganisasikan dirinya sendiri dan dapat dilatih dengan memberikan sejumlah data dan membiarkan jaringan saraf tiruan itu sendiri mencari berbagai pola dan hubungan dalam data tersebut [LAU – 07].

Jaringan saraf tiruan dibentuk sebagai generalisasi model matematika dari jaringan syaraf biologi, dengan asumsi bahwa [SIA – 05] :

a. Pemrosesan informasi terjadi pada banyak elemen sederhana (neuron).
b. Sinyal dikirirnkan diantara neuron-neuron melalui penghubung-penghubung.
c. Penghubung antar neuron memiliki bobot yang akan memperkuat atau memperlemah sinyal.
d. Untuk menentukan output, setiap neuron menggunakan fungsi aktivasi (biasanya bukan fungsi linier) yang dikenakan pada jumlahan input yang diterima. Besarnya output ini selanjutnya dibandingkan dengan suatu batas ambang.

Jaringan saraf tiruan ditentukan oleh 3 hal [SIA – 05] :

a. Pola hubungan antar neuron (disebut arsitektur jaringan).
b. Metode untuk menentukan bobot penghubung (disebut metode *training/learning/* algoritma).
c. Fungsi aktivasi (fungsi transfer), yaitu fungsi yang digunakan untuk menentukan keluaran suatu neuron.

2.1.2 Konsep Dasar Jaringan Saraf Tiruan

Setiap pola-pola informasi input dan output yang diberikan ke dalam JST diproses dalam neuron. Neuron-neuron tersebut terkumpul di dalam lapisan-lapisan yang disebut *neuron layers*. Lapisan-lapisan penyusun JST tersebut dapat dibagi menjadi 3, yaitu:

1. Lapisan input :

   Unit-unit di dalam lapisan input disebut unit-unit input. Unit-unit

input tersebut menerima pola inputan data dari luar yang menggambarkan suatu permasalahan.

2. Lapisan Tersembunyi

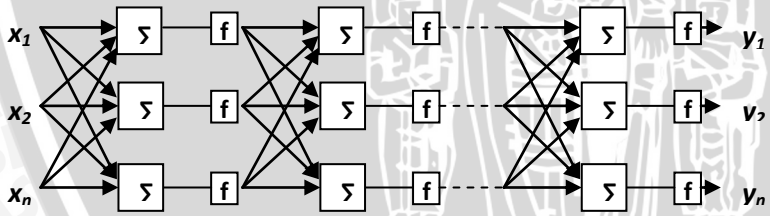
   Unit-unit di dalam lapisan tersembunyi disebut unit-unit tersembunyi. Dimana outputnya tidak dapat secara langsung diamati.

3. Lapisan output

   Unit-unit di dalam lapisan output disebut unit-unit output. Output dari lapisan ini merupakan solusi JST terhadap suatu permasalahan.

2.1.3 Arsitektur *Multilayer Perceptron Neural Network*

Multilayer Perceptron Neural Network (jaringan saraf banyak lapisan) merupakan salah satu model arsitektur dari jaringan saraf tiruan. Arsitektur dengan model ini sebenarnya merupakan model jaringan saraf satu lapisan yang jumlahnya banyak dan prinsip kerja dari model jaringan saraf lapisan ini sama dengan model jaringan saraf satu lapisan. Output dari tiap lapisan sebelumnya dalam model ini merupakan input dari lapisan sebelumnya. Pada penelitian ini, algoritma *resilient propagation* menggunakan arsitektur *multilayer perceptron neural network*. Gambar 2.2 menyatakan model jaringan saraf banyak lapisan :



Gambar 2.1 Arsitektur jaringan saraf banyak lapisan

[KRI – 04]

2.1.4 Metode Pelatihan/Pembelajaran

Cara berlangsungnya pembelajaran atau pelatihan jaringan saraf tiruan dikelompokkan menjadi tiga yaitu [PUS – 06] :

a. *Supervised learning* (pembelajaran terawasi)

   Pada metode ini, setiap pola yang diberikan ke dalam JST telah diketahui outputnya. Selisih antara pola output actual (output yang dihasilkan) dengan pola output yang dikehendaki (output target) yang disebut error digunakan untuk mengoreksi bobot JST sehingga JST mampu menghasilkan output sedekat mungkin dengan pola target yang telah diketahui oleh JST. Contoh algoritma JST yang menggunakan metode ini adalah : Hebbian, Perceptron, ADALINE, Boltzman, Hopfield, *Backpropagation*.

b. Unsupervised learning (pembelajaran tak terawasi)

   Pada metode ini, tidak memerlukan target output. Pada metode ini tidak dapat ditentukan hasil seperti apakah yang diharapkan selama proses pembelajaran. Selama proses pembelajran, nilai bobot disusun dalam suatu range tertentu tergantung pada nilai input yang diberikan. Tujuan pembelajran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya sangat cocok untuk klasifikasi pola. Contoh algoritma JST yang menggunakan metode ini adalah: *Competitive*, Hebbian, Kohonen, LVQ (*Learning Vector Quantization*), Neocognitron.

c. Hybrid Learning

   Merupakan kombinasi dari metode pembelajaran supervised learning dan unsupervised learning. Sebagian dari bobot-bobotnya ditentukan melalui pembelajaran terawasi dan sebagian lainnya melalui pembelajaran tak terawasi. Contoh algoritma JST yang menggunakan metode ini yaitu : algoritma RBF.

2.1.5 Fungsi Aktivasi

Fungsi aktivasi adalah fungsi matematis inputan agar mendekati nilai ambang tertentu. Fungsi aktifasi dalam JST berguna untuk menentukan nilai keluaran dari hasil kombinasi linier masukan dan bobot [SIA – 05].

Dalam *Backpropagation,* fungsi aktivasi yang digunakan harus memenuhi beberapa syarat yaitu : kontinu, terdeferensiasi dengan mudah dan merupakan fungsi yang monoton naik. Fungsi aktivasi yang sering digunakan adalah fungsi sigmoid biner dan fungsi sigmoid bipolar. Fungsi sigmoid biner memiliki *range* (0,1) sedangkan fungsi sigmoid bipolar memiliki *range* (-1,1).

a. Fungsi Sigmoid Biner

$$f(x) = \frac{1}{1 + e^{(-\alpha x)}} \quad (2.1)$$

dengan turunannya

$$f'(x) = \alpha f(x)[1 - f(x)] \quad (2.2)$$

b. Fungsi Sigmoid Bipolar

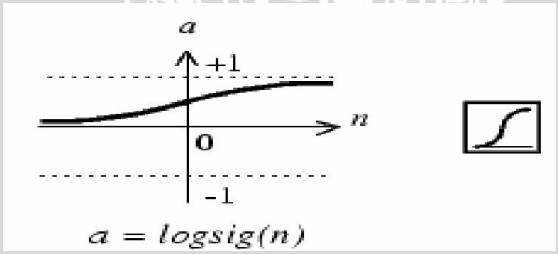
$$f(x) = \frac{1 - e(-x)}{1 + e(-x)} \quad (2.3)$$

dengan turunannya

$$f'(x) = \frac{1}{2}[1 + f(x)][1 - f(x)] \quad (2.4)$$

e merupakan bilangan eural yaitu 2.71828, sedangkan x adalah hasil penjumlahan dari sinyal-sinyal input terbobot.

Pada penelitian ini menggunakan fungsi aktivasi sigmoid biner. Grafik untuk fungsi sigmoid biner dapat dilihat dalam Gambar 2.2 dengan n sebagai masukan dan a sebagai keluaran dari neuron tersebut.



Gambar. 2.2 Grafik fungsi sigmoid biner dengan selang (0,1)

2.1.6 Normalisasi dan Denormalisasi Data

Normalisasi data digunakan untuk menyamakan skala data ke dalam jangkauan nilai tertentu, misal -1 sampai dengan 1. Misalkan suatu data akan diubah kedalam suatu jangkauan nilai antara 0.1 dan 0.8, dalam kasus diabetes ini

akan digunakan jangkuan nilai antara 0.1 sampai 0.9, maka dapat dihitung dengan persamaan 2.5 sebagai berikut :

$$\boldsymbol{x}' = \frac{0.8(x - \min \boldsymbol{values})}{(\max \boldsymbol{values} - \min \boldsymbol{values})} + 0.1 \quad (2.5)$$

Denormalisasi data adalah proses untuk mengembalikan data yang telah ternormalisasi. Perhitungan denormalisasi ditunjukkan pada persamaan

$$\boldsymbol{x} = \frac{(\max \boldsymbol{values} - \min \boldsymbol{values})(\boldsymbol{x}' - 0.1)}{0.8} + \min \boldsymbol{values} \quad (2.6)$$

dimana :

x’ = data hasil normalisasi

x = data awal

min values = nilai terkecil dari seluruh data

max values = nilai terbesar dari seluruh data

2.1.7 Inisialisasi Bobot dan Bias

Inisialisasi pembobot dari *neuron* input ke *neuron* tersembunyi bertujuan untuk meningkatkan kemampuan *neuron-neuron* tersembunyi untuk melakukan pembelajaran. Hal ini dilakukan dengan mendistribusikan pembobot dan bias awal sedemikian rupa sehingga dapat meningkatkan kemampuan lapisan tersembunyi dalam melakukan proses pembelajaran [FAU – 94].

Bobot awal akan memengaruhi apakah jaringan mencapai titik minimum lokal atau global, dan seberapa cepat konvergensinya. Nilai bobot awal tidak boleh terlalu besar karena nilai turunan fungsi aktivasinya akan menjadi kecil. Nguyen dan Widrow (1990) mengusulkan cara membuat inisialisasi bobot dan bias ke unit tersembunyi sehingga menghasilkan iterasi yang lebih cepat. Algoritma inisialisasi Nguyen-Widrow didefinisikan sebagai persamaan berikut :

1. Hitung harga faktor penskalaan β dengan persamaan 2.7

(2.7)

$$\beta = 0.7\sqrt[n]{p}$$

dengan :

β = faktor skala

n = jumlah unit masukan

p = jumlah unit tersembunyi

2. Inisialisasi semua bobot (vij(lama)) dengan bilangan acak dalam interval [-0.05,0.05]
3. Hitung nilai $\|v_j\|$ yang dinyatakan dalam persamaan berikut:

$$\|v_j\| = \sqrt{v^2{}_{1j} + v^2{}_{2j} + \cdots + v^2{}_{nj}} \quad (2.8)$$

4. Bobot yang dipakai sebagai inisialisasi dinyatakan dalam persamaan berikut:

$$v_{ij} = \frac{\beta v_{ij}(lama)}{\|v_j\|} \quad (2.9)$$

5. Bias yang dipakai sebagai inisialisasi v_{j0} = bilangan acak antara $-\beta$ dan β.

2.1.8 Jumlah Unit Tersembunyi

Menentukan jumlah lapis pada *hidden layer* dan menentukan jumlah unit per layernya sangat sulit. *Resilient propagation* dengan sebuah *hidden layer* sudah cukup untuk mengenali sembarang perkawanan antara masukan dan target dengan tingkat ketelitian yang ditentukan. Jumlah unit tersembunyi dapat ditentukan dengan melakukan beberapa percobaan (*trial and error*). Tetapi penambahan jumlah *hidden* layer kadangkala membuat pelatihan menjadi lebih mudah [SIA – 05].

2.1.9 Lama Iterasi

Tujuan utama penggunaan *resilient propagation* adalah mendapatkan keseimbangan antata pengenalan pola pelatihan secara benar dan respon yang baik untuk pola lain yang sejenis (disebut pengujian). Jaringan dapat dilatih terus

menerus hingga semua pola pelatihan dikenalai dengan benar. Akan tetapi hal itu tidak menjamin jaringan akan mampu mengenali pola pengujian dengan tepat.

Umumnya data dibagi menjadi 2 bagian saling asing, yaitu pola data yang dipakai sebagai pelatihan dan data yang dipakai untuk pengujian. Perubahan bobot dilakukan berdasarkan pola pelatihan. Akan tetapi selama pelatihan (misal setiap 10 epoch), kesalahan yang terjadi dihitung berdasarkan semua data (pelatihan dan pengujian). Selama kesalahan ini menurun, pelatihan terus dijalankan. Akan tetapi jika kesalahannya sudah meningkat, pelatihan tidak ada gunanya untuk diteruskan lagi [SIA – 05].

2.3 Algoritma *Resilient Propagation*

Jaringan Saraf Tiruan (JST) *Resilient Propagation* merupakan algoritma pelatihan pada JST yang dikembangkan untuk mengatasi kelemahan pada JST Propagasi Balik (*Backpropagation*) yang biasanya menggunakan fungsi aktivasi *sigmoid*. Fungsi aktivasi ini akan membawa *input* dengan *range* yang tidak terbatas ke nilai *output* dengan *range* yang terbatas antar 0 dan 1. Salah satu karakteristik dari fungsi *sigmoid* adalah gradiennya akan mendekati nol apabila *input* yang diberikan sangat banyak. Gradien yang mendekati nol ini akan berimplikasi pada rendahnya perubahan bobot. Apabila bobot-bobot tidak cukup mengalami perubahan, maka algoritma akan sangat lambat untuk mendekati nilai optimumnya [RIE – 94].

Sama seperti pada algoritma *gradien descent backpropagation*, algoritma RPROP melaksanakan dua tahap pembelajaran yaitu tahap perambatan maju (*forward*) untuk mendapatkan error output dan tahap perambatan mundur (*backward*) untuk mengubah nilai bobot-bobot [FEB – 07].

Algoritma *resilient propagation* melakukan perubahan bobot dan bias jaringan sesuai dengan perilaku *gradien* di setiap *epoch* pelatihan, sehingga jumlah *epoch* yang diperlukan untuk mencapai target yang diinginkan jauh lebih sedikit. Untuk setiap bobot w_{ij} terdapat nilai-update Δij yang berubah selama proses pelatihan sesuai dengan perilaku *gradien*. Nilai *update* dan bobot berubah setiap kali seluruh pola data pelatihan telah diproses ke dalam jaringan, yaitu setiap satu *epoch*. Bobot merupakan salah satu faktor penting agar jaringan dapat

melakukan generalisasi dengan baik terhadap data yang dilatihkan kedalamnya, karena bobot pada jaringan saraf tiruan mempengaruhi besarnya sinyal yang akan keluar dari setiap node yang ada pada *hidden layer* dan *output layer*.

Walaupun pada *resilient backpropagation* ada banyak parameter yang dapat diatur, tetapi sebagian besar dari parameter-parameter tersebut dapat digunakan dengan nilai yang diatur secara *default*. Hal ini disebabkan karena variasi nilai dari parameter-parameter tersebut tidak terlalu mempengaruhi waktu yang dibutuhkan untuk pelatihan [RIE – 94].

Berdasarkan hasil penelitian tersebut, pada penelitian ini digunakan nilai *default* pada parameter-parameter yang diperlukan. Nilai Δ_{max} dan Δ_{min} masing-masing sebesar 50 dan 0,1. Sedangkan untuk parameter η^+ dan η^- digunakan nilai 1,2 dan 0,5.

Pelatihan dilakukan berulang-ulang dan berhenti jika telah mencapai batas iterasi maksimum yang ditentukan dan nilai *error* kurang dari *Mean Square Error* (MSE). Ketepatan algoritma *resilient propagation* ditentukan dengan *Mean Square Error* (MSE). Semakin kecil nilai MSE maka dapat dianggap bahwa arsitektur jaringan semakin baik [FAU – 94]. MSE dihitung dengan persamaan 2.10.

$$MSE = \frac{\sum_P (t_p - y_p)^2}{N * K} \tag{2.10}$$

dimana p adalah jumlah *neuron* (unit) *output*, t adalah target, y adalah output, N adalah jumlah data latih dan K adalah banyaknya keluaran.

Proses perambatan maju (*forward*) pada algoritma RPROP sama dengan algoritma Propagasi Balik umumnya, sedangkan proses perambatan mundur (*backward*) berbeda. Algoritma pelatihan RPROP adalah adalah sebagai berikut :

1. Inisialisasi bobot

   Bobot awal ditentukan secara acak dengan nilai sekecil mungkin, misalkan [-0.05, 0.05].

2. Inisialisasi *error* target, jumlah *hidden layer*, *learning rate*, faktor penaik, faktor penurun, penyesuaian minimum, penyesuaian maksimum dan jumlah maksimum epoch.
3. Selama (Epoch < Maksimum epoch) dan (MSE >Target Error), maka:
   a. Lakukan *feedforward* untuk setiap pasangan pelatihan:
      1. Tiap unit masukan (x_i, i=1,…,n) menerima sinyal masukan x_i dan meneruskan sinyal tersebut ke semua unit pada *layer* di atasnya (*hidden layer*)
      2. Tiap-tiap unit pada *hidden layer* (z_j, j=1,….,p) menjumlahkan sinyal–sinyal input terbobot,

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i \, . \, v_{ij} \tag{2.11}$$

lalu menghitung sinyal keluarannya dengan fungsi aktivasi,

$$z_j = f(z\_in_j) \tag{2.12}$$

dan mengirimkan sinyal ini ke seluruh unit pada lapisan atasnya (lapisan *output*).

      3. Setiap unit output (y_k, k=1,…,m) menghitung total sinyal masukan terbobot,

$$y\_in_k = w_{ok} + \sum_{j=1}^{p} w_{jk} \, . \, z_j \tag{2.13}$$

lalu menghitung sinyal keluaran dengan fungsi aktivasi,

$$y_k = f(y\_in_k) \tag{2.14}$$

   b. Lakukan *resilient propagation* untuk masing-masing pasangan pelatihan:
      1. Menghitung nilai informasi error (δ_k) pada tiap-tiap unit output :

$$\delta_k = f'(y_k)\frac{1}{2}(t_k - y_k) \tag{2.15}$$

lalu hitung nilai informasi error (δ_j) pada tiap-tiap unit *hidden* dengan melakukan proses backpropagation terhadap error pada output (δ_k)

$$\delta_j = f'(z_j)\sum_{k} \delta_k w_{jk} \tag{2.16}$$

2. Menghitung nilai turunan pertama fungsi error terhadap bobot pada input ke *hidden layer* (ΔE_{ij})

$$\Delta E_{ij} = \Delta E_{ij} + \delta_j x_i \quad (2.17)$$

dan menghitung nilai turunan pertama fungsi error terhadap bobot pada *hidden* ke *output layer* (ΔE_{jk})

$$\Delta E_{jk} = \Delta E_{jk} + \delta_k z_j \quad (2.18)$$

c. Perbarui Bobot

1. Menghitung nilai learning rate yang baru pada bobot antara input ke *hidden layer* (Δ_{ij}).

Jika ΔE_{ij} lama * $\Delta E_{ij} > 0$ maka

$\Delta_{ij} = \min(\Delta_{ij}\text{lama} * \eta+, \Delta_{max})$

$\Delta V_{ij} = -\text{sign}(\Delta E_{ij} * \Delta_{ij})$

Jika ΔE_{ij} lama * $\Delta E_{ij} < 0$ maka

$\Delta_{ij} = \max(\Delta_{ij}\text{lama} * \eta-, \Delta_{min})$

$\Delta E_{ij} = 0$ (2.19)

Jika tidak maka

$\Delta V_{ij} = -\text{sign}(\Delta E_{ij} * \Delta_{ij})$

$\Delta E_{ij} lama = \Delta E_{ij}$ (2.19)

2. Perbarui nilai bobot antara input ke *hidden layer* (v_{ij})

Jika $\Delta E_{ij} < 0$ maka

v_{ij} = v_{ij} (lama) - ΔV_{ij}

Jika tidak maka

v_{ij} = v_{ij} (lama) + ΔV_{ij} (2.20)

3. Perbarui nilai *learning rate* sebelumnya (Δ_{ij} lama) dan nilai turunan pertama fungsi error terhadap bobot pada input ke *hidden layer* sebelumnya (ΔE_{ij}lama)

Δ_{ij}lama = Δ_{ij}

ΔE_{ij}lama = ΔE_{ij}

ΔE_{ij} = 0 (2.21)

4. Menghitung nilai *learning rate* yang baru pada bobot antara *hidden* ke *output layer* (Δ_{jk})

   Jika ΔE_{jk}lama * ΔE_{jk} > 0 maka

   Δ_{jk}= min(Δ_{jk}lama $*$ η+, Δ_{max})

   $\Delta W_{jk} = -\text{sign}(\Delta E_{jk} * \Delta_{jk})$

   $\Delta E_{jk}\text{lama} = \Delta E_{jk}$

   Jika ΔE_{jk}lama * ΔE_{jk} < 0 maka

   Δ_{jk}= max(Δ_{jk}lama $*$ η−, Δ_{min})

   $\Delta E_{jk} = 0$

   Jika tidak maka

   $\Delta W_{jk} = -\text{sign}(\Delta E_{jk} * \Delta_{jk})$ (2.22)

   $\Delta E_{jk}\text{lama} = \Delta E_{jk}$

5. Perbarui nilai bobot antara *hidden* ke *output layer* (w_{jk})

   Jika $\Delta E_{jk} < 0$ maka

   w_{jk} = w_{jk} (lama) - ΔW_{jk}

   Jika tidak maka

   w_{jk} = w_{jk} (lama) + ΔW_{jk} (2.23)

6. Perbarui nilai *learning rate* sebelumnya (Δ_{jk}lama) dan nilai turunan pertama fungsi error terhadap bobot pada input ke *hidden layer* sebelumnya (ΔE_{jk}lama)

   Δ_{jk}lama = Δ_{jk}

   ΔE_{jk}lama = ΔE_{jk}

   ΔE_{jk} = 0 (2.24)

Jika iterasi mencapai maksimum maka berhenti.

Keterangan dari simbol :

t = target

δ_k = Informasi tentang kesalahan pada unit yk yang disebarkan kembali ke unit tersembunyi

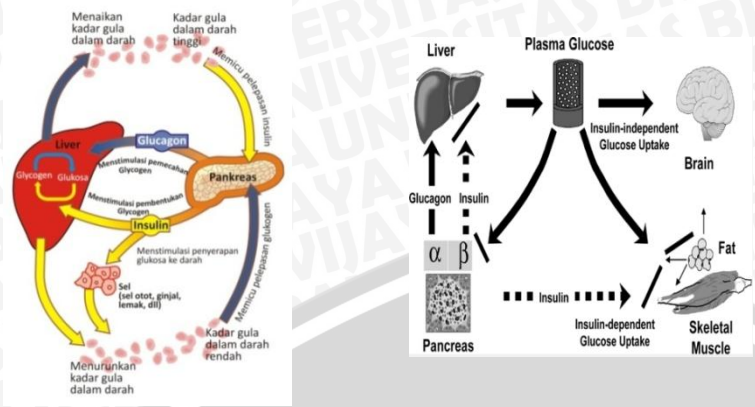
δ_j = Informasi tentang kesalahan dari lapisan *output* ke unit tersembunyi zj

α = *learning rate*
X_i = unit *input* i
V_{oj} = Bobot awal bias pada lapisan tersembunyi j
V_{ij} = Bobot menuju hidden
Z_j = Unit tersembunyi j
Z_in_j = *Input* jaringan ke zj
W_{ok} = Bobot awal bias pada menuju *output*
W_{jk} = Bobot menuju output
Y_k = Unit *output* i
Y_in_k = *Input* jaringan ke yk
η^+ = faktor penaik
η^- = faktor penurun
Δ_{min} = penyesuaian minimum
Δ_{max} = penyesuaian maksimum
E = Error tiap pasangan

2.3 Diabetes Mellitus

2.3.1 Pengertian Diabetes Mellitus

Diabetes Mellitus merupakan suatu penyakit dimana salah satu organ tubuh, dalam hal ini pankreas tidak dapat mengatur kadar gula (glukosa) dalam darah. Glukosa dalam tubuh memberikan energi untuk melakukan aktivitas seperti berjalan, berlari, mengendarai sepeda, dan aktivitas yang lain. Glukosa dalam darah diproduksi oleh hati dari makanan. Dalam orang yang sehat beberapa hormon insulin berfungsi untuk mengatur kadar glukosa darah. Insulin diproduksi oleh pankreas, organ kecil dekat perut yang juga mengeluarkan enzim penting yang membantu dalam proses pencernaan makanan. Insulin mengatur glukosa untuk bergerak dari darah ke dalam hati, otot dan sel lemak dimana ini digunakan untuk bahan bakar. Orang dengan penyakit diabetes terbagi menjadi dua yaitu tidak dapat memproduksi insulin yang cukup (*type 1 diabetes*) dan orang yang tidak dapat menggunakan insulin dengan baik (*type 2 diabetes*).



Gambar 2.1 *Diabetes Mellitus*

[Sumber : KAH – 06]

Dalam diabetes, glukosa dalam darah tidak dapat bergerak ke sel dan tinggal dalam darah. Ini tidak hanya dapat membahayakan sel yang membutuhkan glukosa untuk bahan bakar tetapi juga membahayakan organ yang lain dan jaringan otot apabila kadar glukosa dalam darah tersebut tinggi.

2.3.2 Tipe Diabetes

2.3.2.1 Tipe 1 diabetes

Pankreas berhenti memproduksi insulin atau memproduksi terlalu sedikit insulin untuk mengatur kadar glukosa darah. Type 1 diabetes dapat ditemui secara khas pada masa kanak-kanak atau masa remaja. Itu biasanya dikenal sebagai " Serangan Diabetes Anak Muda" atau "*Insulin Dependent Diabetes Mellitus* ".

Type 1 diabetes dapat terjadi pada usia tua yang diakibatkan kerusakan pankreas oleh alkohol, penyakit, operasi atau kesalahan progresif dari sel pankreas yang memproduksi insulin. Orang dengan type 1 diabetes umumnya membutuhkan terapi insulin setiap hari untuk menopang hidupnya.

2.3.2.2 Type 2 diabetes

Organ pankreas menghasilkan insulin, tetapi sebagian tubuh atau seluruhnya tidak dapat menggunakan insulin dengan baik. Biasanya diartikan sebagai perlawanan insulin. Tubuh mencoba mengatasi perlawanan ini dengan menghasilkan insulin lebih banyak lagi. Dengan perlawanan insulin tersebut otomatis akan mengalami kerusakan sehingga tidak dapat melanjutkan untuk

menghasilkan insulin yang cukup untuk mengatasi permintaan tinggi. Paling sedikit 90 % pasien memiliki type 2 diabetes. Type 2 diabetes secara khas dapat diketahui dalam usia dewasa, biasanya sekitar umur 45 ke atas. Ini biasanya disebut dengan "Serangan Dewasa Diabetes Mellitus" atau "*Non Insulin Dependent Diabetes Mellitus"*. Nama ini sudah tidak digunakan lagi karena type 2 diabetes kerap terjadi pada remaja dan beberapa orang dengan type 2 diabetes yang membutuhkan insulin. Type 2 diabetes sering dikontrol dengan diet, olahraga dan pengobatan oral. Lebih dari setengah orang dengan type 2 diabetes membutuhkan insulin untuk mengontrol kadar gula darah pada saat perawatan sakitnya.

2.3.3 Faktor Resiko Diabetes

Faktor resiko penyakit diabetes ada banyak sekali. Adapun dalam penelitian ini akan difokuskan dan dibahas 4 (lima) faktor resiko yaitu mengenai kadar glukosa dalam darah, Kolesterol, Trigliserida, dan Ureum.

2.3.3.1 Glukosa

Kadar gula darah, atau jumlah gula dalam darah merupakan suatu indikator yang dapat menjadi penentu apakah seseorang sehat atau tidak. Jumlah gula darah berhubungan dengan terjadinya penyakit *diabetes mellitus*. Gula darah adalah istilah yang mengacu pada tingkat glukosa yang terdapat di dalam darah, dimana gula darah ini secara ketat diatur dalam tubuh. Glukosa yang dialirkan melalui darah kemudian berfungsi sebagai sumber utama energi untuk sel-sel tubuh. Itulah mengapa kemudian kadar gula darah sangat penting untuk kelangsungan hidup sel-sel tubuh.

Tingkat normal gula dalam darah sebelum makan (stabil) adalah antara 70 dan 110 mg/dl dan setelah makan akan naik ke tingkat di antara 100-140 mg/dl. Peningkatan kadar gula darah di atas 140 mg/dl dianggap sebagai gejala-gejala penderita diabetes (*hiperglikemia*). Kadar gula darah yang rendah yang berkisar dibawah 70 juga hal berbahaya. Hal ini dapat menyebabkan seseorang cepat pingsan (*hipoglikemia*).

2.3.3.2 Kolesterol

Kolesterol merupakan salah satu komponen dari lemak. Adapun lemak sendiri adalah cadangan energi yang memberikan kontribusi kalori paling tinggi untuk berbagai proses metabolisme di dalam tubuh.

Timbulnya kolesterol dalam jumlah yang kelewat tinggi, diantaranya disebabkan oleh terlampau berlebihnya asupan makanan yang berasal dari lemak hewani, telur dan serta makanan-makanan yang dewasa ini disebut sebagai makanan sampah (*junkfood*).

Kadar kolesterol normal yaitu berada di bawah 200 mg/dl. Adapun apabila kadar kolesterol seseorang melampaui atau kadarnya di atas 200 mg/dl maka resiko terkena diabetes akan semakin besar. Dikarenakan kadar kolesterol yang tinggi, maka kolesterol tersebut akan mengendap pada arteri sehingga berakibat akan menghambat proses pengambilan glukosa.

2.3.3.3 Trigliserida

Trigliserida merupakan jumlah lemak yang bersirkulasi dalam darah. Dengan kata lain trigliserida sederhananya dapat disebut sebagai lemak. Lemak yang ada dalam makanan dan yang ada dalam tubuh merupakan bentuk dari trigliserida. Nilai trigliserida yang tinggi (hipertrigliseridemia) memiliki kaitan yang erat dengan timbulnya penyakit jantung dan diabetes. Banyak orang dengan penyakit jantung atau diabetes memiliki kadar trigliserida tinggi.

Peningkatan trigliserida dapat disebabkan oleh kelebihan berat badan / obesitas, aktivitas fisik, merokok, konsumsi alkohol berlebihan dan diet tinggi karbohidrat, kelainan genetik dan lain sebagainya.

Kadar Trigliserida normal yaitu kurang dari 150mg/dl. Jika seseorang melampaui dari kadar normal tersebut maka resiko terkena diabetes pun menjadi besar.

2.3.3.3 Ureum

Hampir seluruh ureum dibentuk di dalam hati, dari metabolisme protein (asam amino). Urea berdifusi bebas masuk ke dalam cairan intra sel dan ekstrasel. Zat ini dipekatkan dalam urin untuk diekskresikan. Pada keseimbangan nitrogen

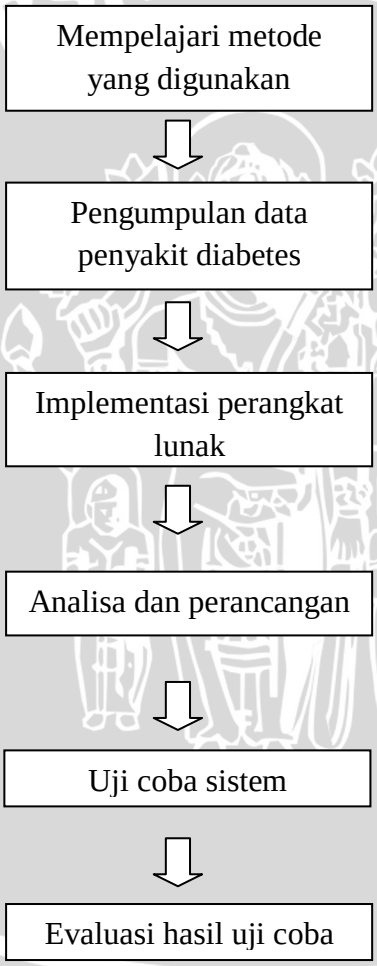
yang stabil, sekitar 25 gram urea diekskresikan setiap hari. Kadar dalam darah mencerminkan keseimbangan antara produksi dan ekskresi urea.

Ureum berasal dari penguraian protein, terutama yang berasal dari makanan. Pada orang sehat yang makanannya banyak mengandung protein, ureum biasanya berada di atas rentang normal. Kadar rendah biasanya tidak dianggap abnormal karena mencerminkan rendahnya protein dalam makanan atau ekspansi volume plasma. Namun, bila kadarnya sangat rendah bisa mengindikasikan penyakit hati berat. Kadar urea bertambah dengan bertambahnya usia, juga walaupun tanpa penyakit ginjal.

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab metodologi dan perancangan ini akan dibahas metode, rancangan yang digunakan, dan langkah – langkah yang dilakukan dalam memberikan diagnosis penyakit diabetes mellitus berdasarkan faktor risiko menggunakan jaringan syaraf tiruan *Resilient Propagation*.

Langkah – langkah penelitian ini dapat digambarkan seperti pada Gambar 3.1.



Gambar 3.1 Blok diagram penelitian

Penelitian dilakukan dengan tahapan – tahapan sebagai berikut :

1. Mempelajari literatur atau sumber-sumber yang terkait dengan metode yang digunakan pada sistem ini (*Resilient Propagation*) dan objek penelitian (penyakit diabetes mellitus).
2. Mengumpulkan data pasien yang berasal dari data rekam medik pasien RSUD Pare dan laboratorium cek fisik kesehatan.
3. Menganalisa dan merancang sistem dengan menggunakan hasil pembelajaran pada tahap sebelumnya.
4. Membuat sistem berdasarkan analisis dan perancangan yang dilakukan.
5. Uji coba sistem.
6. Evaluasi hasil uji coba.

3.1 Sumber Data

Data yang digunakan dalam penelitian ini merupakan data yang terdiri dari data rekam medik pasien RSUD Pare dan laboratorium cek fisik kesehatan. Data yang digunakan sebanyak 100 data pasien 50% pasien diabetes dan 50% tidak terkena diabetes.

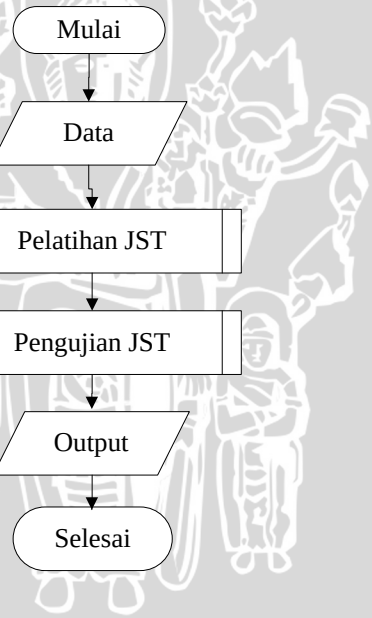
3.2 Gambaran Umum Perangkat Lunak

Sistem yang dikembangkan merupakan implementasi dari algoritma *Resilient Propagation* dalam bentuk aplikasi untuk memberikan diagnosis penyakit diabetes dan di dalamnya terdapat penentuan diagnosis penyakit tersebut apakah terkena penyakit diabetes atau tidak. Sistem akan mendiagnosis atau mengolah *input* yang diberikan oleh *user* dan akan memberikan *output* berupa hasil diagnosisnya.

Dari *input-input* tersebut akan dilakukan pelatihan dan memberikan *output* berupa nilai risiko yang kemudian dikategorikan, sehingga akan terdapat 1 neuron pada lapisan keluaran jaringan saraf. Pada tahap pembelajaran menggunakan algoritma *feedforward*, kemudian akan dihitung besarnya nilai *error*. Jika kesalahan yang ditargetkan belum didapat maka dilakukan *resilient propagation*. Data yang dipakai pada pelatihan ini akan diujikan yaitu dengan menggunakan 70 data latih, kemudian dilakukan pengujian dengan data uji dari sisa keseluruhan

data terhadap data latih. Jaringan terdiri dari satu lapisan *input* dengan 4 *neuron*, satu lapisan *hidden* (5, 10, 20, 30, 40, 50, 60, 70, 80, 90 *neuron*) dan satu lapisan *output* dengan 1 *neuron*. Data dilatihkan ke jaringan hingga *error* minimum, jika kesalahan mencapai nilai minimum maka bobot akan disimpan dalam data penyimpanan (*dataset*). Pada penelitian ini akan digunakan database Microsoft Excel 2007 sebagai tempat penyimpanan (*dataset*). Selanjutnya pada tahap pengujian sistem, dengan menggunakan bobot-bobot akhir yang telah diperoleh selama proses pembelajaran, maka dihasilkan klasifikasi terhadap data pengujian.

Secara keseluruhan, tahap pengembangan sistem diagnosis penyakit diabetes berdasarkan faktor risiko menggunakan jaringan syaraf tiruan *resilient propagation* dari tahap awal sampai akhir diilustrasikan pada Gambar 3.2.



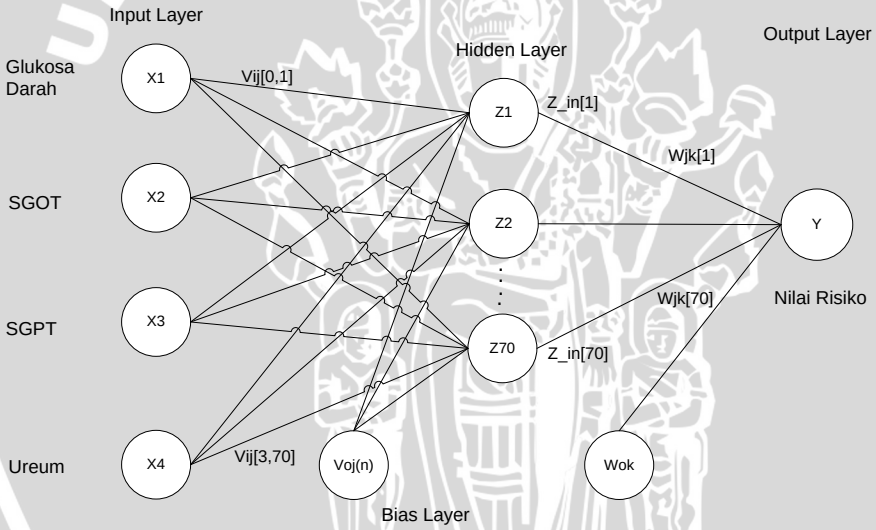
Gambar 3.2 *Flowchart* sistem secara umum

3.3 Proses Jaringan Syaraf Tiruan *Resilient Propagation*

Proses ini dapat dibagi menjadi proses membuat arsitektur jaringan, proses pelatihan, dan proses pengujian.

3.3.1 Arsitektur

Arsitektur dari *resilient propagation* menggunakan *multilayer perceptron neural network* yang terdiri dari lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer*) dan lapisan keluaran (*output layer*). Arsitektur jaringan syaraf yang digunakan adalah 4 unit *neuron* pada lapisan *input* yang menyatakan faktor risiko penyakit diabetes dan 2 unit *neuron* pada lapisan *output* yang menyatakan diagnosis penyakit diabetes penderita. Sedangkan jumlah neuron pada *hidden layer* adalah n, dimana nilai n nantinya akan diujicobakan beberapa nilai. Hal ini dilakukan untuk mendapatkan arsitektur jaringan yang sesuai untuk melakukan pengenalan. Arsitektur JST diagnosis penyakit diabetes berdasarkan faktor risiko dapat dilihat pada Gambar 3.3.

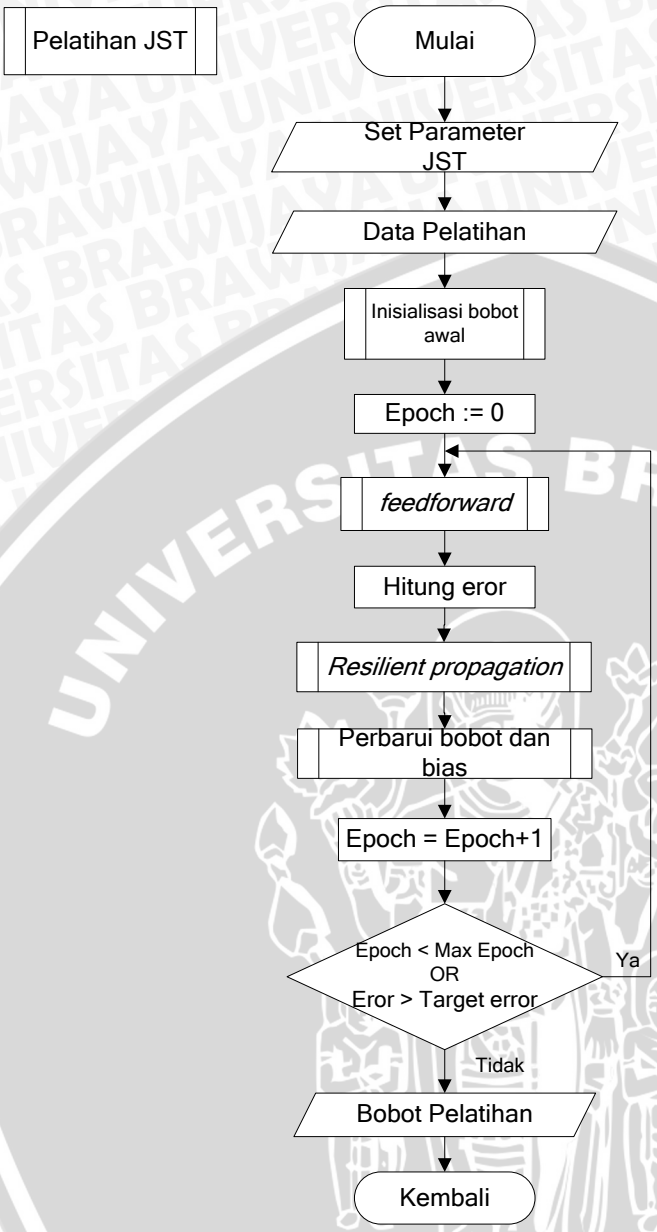


Gambar 3.3 Arsitektur Jaringan Saraf Tiruan *Resilient Propagation*

3.3.2 Proses Pelatihan

Algoritma pelatihan *resilient propagation* pada dasarnya terbagi menjadi 3 langkah, yaitu: langkah maju (*feedforward)*, propagasi balik (*resilient propagation)* dan perubahan bobot. Secara keseluruhan langkah-langkah pelatihan algoritma *resilient ropagation* diilustrasikan pada Gambar 3.4 dan langkah-langkahnya adalah sebagai berikut:

1. Mulai.
2. Set parameter JST yaitu *learning rate*, kesalahan yang ditargetkan, *maksimum epoch*, faktor penaik, faktor penurun, penyesuaian minimum, penyesuaian maksimum dan *neuron hidden*.
3. Masukkan data *input* yang akan dilatihkan.
4. Inisialisasi bobot dan bias awal dengan menggunakan algoritma inisialisai Nguyen Widrow.
5. Setting parameter iterasi awal (*epoch*) sama dengan 0.
6. Setelah seluruh koneksi jaringan terisi bobot, lakukan *feedforward*.
7. Lakukan perhitungan kesalahan antara pola *output* JST dan pola target.
8. Lakukan langkah *resilient propagation*.
9. Perbarui bobot dan bias
10. Periksa apakah kesalahan *output* lebih besar dari kesalahan yang ditargetkan atau *epoch* lebih besar dari maksimum *epoch*? Jika ya, maka lakukan langkah 11, jika tidak, maka lakukan langkah 12.
11. Naikkan *epoch* 1,
12. Proses berhenti, bobot dan bias akhir pelatihan disimpan pada dataset, data siap digunakan untuk proses pengujian.
13. Selesai.



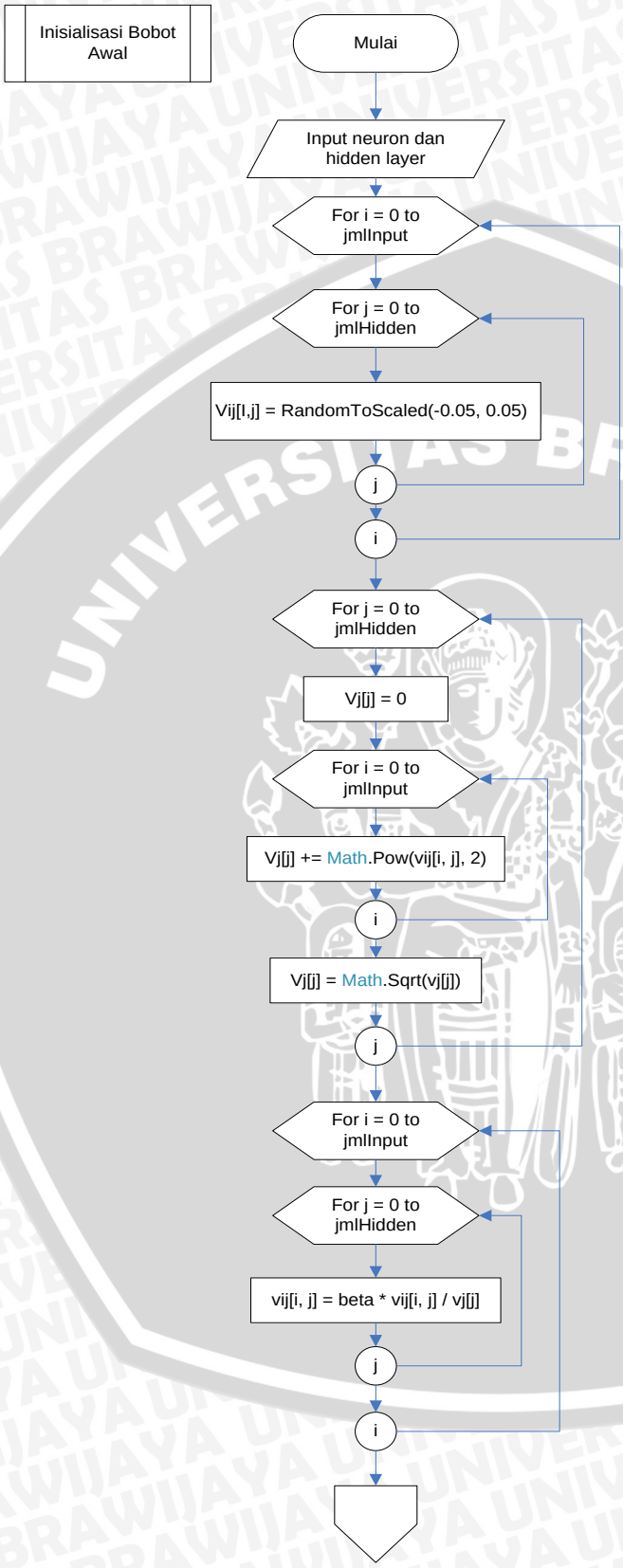
Gambar 3.4 *Flowchart* proses pelatihan JST *Resilient Propagation*

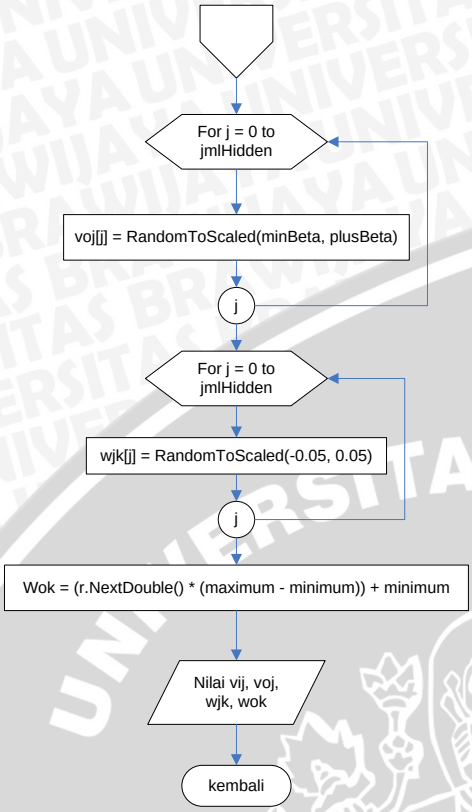
3.3.2.1 Inisialisasi Bobot dan Bias Awal

Inisialisasi bobot dan bias awal Nguyen Widrow digunakan untuk menentukan nilai bobot dan bias awal dari unit masukan ke unit tersembunyi. Tujuannya adalah untuk mempercepat iterasi. Karena bobot awal akan mempengaruhi apakah jaringan mencapai titik minimum lokal atau global, dan seberapa cepat konvergensinya Untuk proses Inisialisasi Bobot dan Bias Awal Nguyen Widrow dapat dilihat pada Gambar 3.5 dan langkah-langkahnya adalah sebagai berikut:

1. Mulai
2. Pemberian nilai awal pada semua bobot dari unit masukan ke unit tersembunyi dengan bilangan acak dalam interval -0,05 sampai 0,05
3. Tentukan jumlah unit masukan dan unit tersembunyi.
4. Kemudian hitung faktor skala menggunakan rumus

   $\beta = 0.7\sqrt[n]{p}$
5. Setelah diperoleh hasil perhitungan faktor skala, selanjutnya adalah Pemberian nilai bias awal dari unit masukan ke unit tersembunyi
6. Hitung $\| V_j \|$
7. Selanjutnya, pemberian nilai bobot kembali dari unit masukan ke unit tersembunyi.
8. Selesai


Inisialisasi Bobot Awal
Mulai
Input neuron dan hidden layer
For i = 0 to jmlInput
For j = 0 to jmlHidden
Vij[I,j] = RandomToScaled(-0.05, 0.05)
j
i
For j = 0 to jmlHidden
Vj[j] = 0
For i = 0 to jmlInput
Vj[j] += Math.Pow(vij[i, j], 2)
i
Vj[j] = Math.Sqrt(vj[j])
j
For i = 0 to jmlInput
For j = 0 to jmlHidden
vij[i, j] = beta * vij[i, j] / vj[j]
j
i



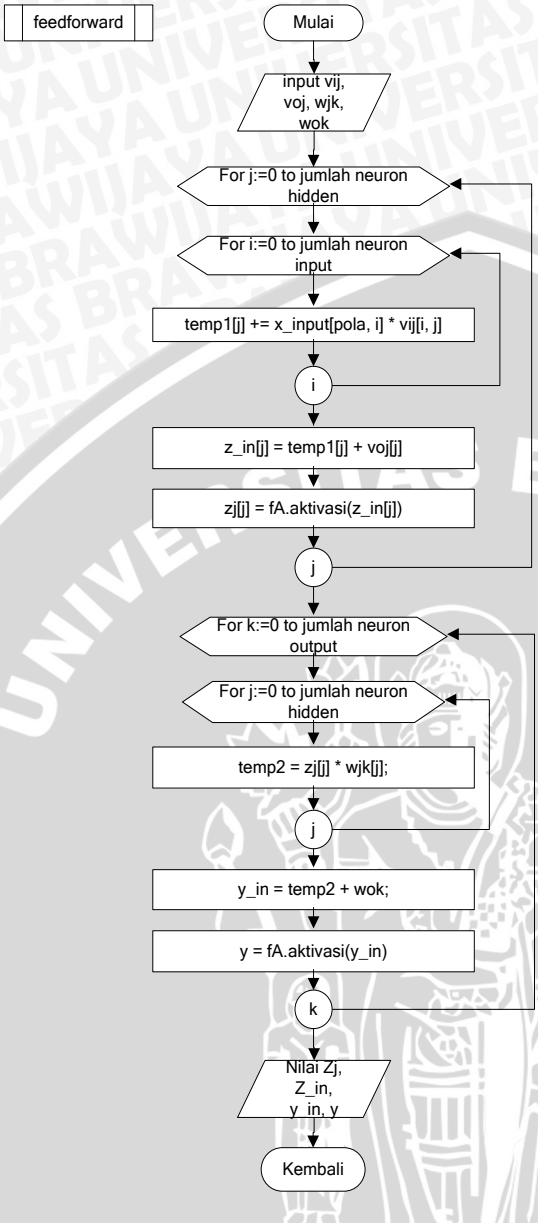
Gambar 3.5 *Flowchart* proses inisialisasi bobot dan bias awal Nguyen Widrow

3.3.2.2 *Feedforward*

Pada proses pelatihan JST terdapat sub proses *feedforward* dan *resilient propagation*. Proses yang dilakukan dalam fase *feedforward* adalah menjumlahkan perkalian antara masukan dengan bobot yang ada dan menghitung nilai aktivasinya untuk kemudian hasil dari perhitungan tersebut dijadikan masukan oleh lapisan yang berada diatasnya. Untuk proses *feedforward* dapat dilihat pada Gambar 3.6 dan langkah-langkah *feedforward* adalah sebagai berikut:

1. Mulai
2. Kalikan seluruh data *input(x)* pada *neuron input* dengan bobot random(Vij) pada masing-masing bobot koneksi bobot *Input* yang terhubung dengan *neuron input.* Kemudian jumlahkan seluruh vektor bobot yang menuju neuron *hidden* yang sama(Z_in).

3. Lakukan aktivasi(Zj) hasil penjumlahan tersebut pada masing-masing *neuron* di lapisan tersembunyi, sehingga *output* pada lapis ini berada pada kisaran 0 dan 1.
4. Kalikan seluruh data hasil aktivasi masing-masing neuron lapis *hidden* pada *neuron hidden* dengan bobot pada masing-masing koneksi bobot *output* yang terhubung dengan *neuron* pada lapis *hidden*. Kemudian jumlahkan seluruh vektor bobot yang menuju *neuron output* yang sama(Y_in).
5. Lakukan aktivasi(Y) hasil penjumlahan tersebut pada masing masing neuron di lapisan *output*, sehingga *output* pada lapis ini berada pada kisaran 0 dan 1.
6. Selesai.



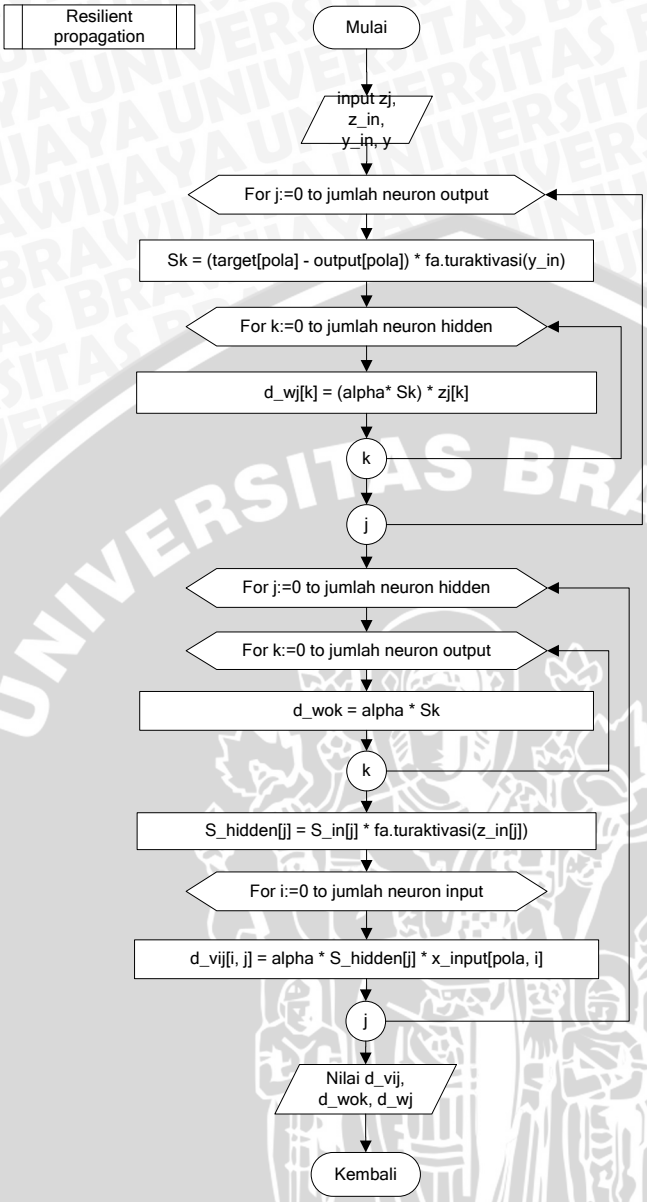
Gambar 3.6 *Flowchart* proses *feedforward*

3.3.2.3 *Resilient Propagation*

Sedangkan untuk langkah *resilient propagation* adalah perhitungan informasi kesalahan pada tiap *neuron* pada masing-masing lapisan dimulai dari kesalahan pada lapis *output* hingga lapis *hidden* terdekat dengan lapis *input*. Informasi kesalahan berguna untuk menghitung faktor peubah bobot yang akan digunakan untuk perbaikan bobot lama. Algoritma *backpropagation* diperlihatkan pada diagram alir pada Gambar 3.7

Langkah-langkah *resilient propagation* :

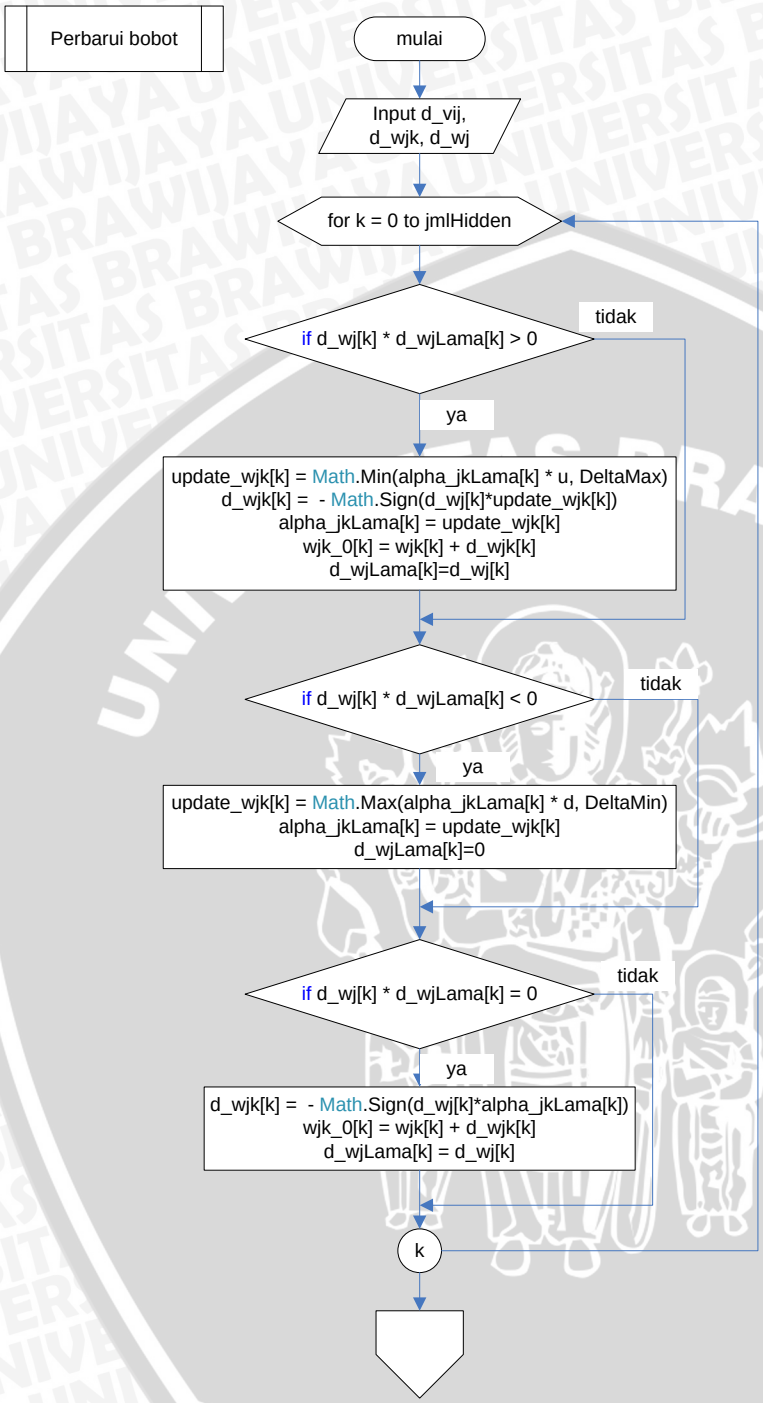
1. Mulai
2. Pada lapisan *output*. Pertama hitung selisih antara target pelatihan dengan *output* dan dikalikan 0,5. Selisih ini disebut sebagai *Error*. Kalikan selisih ini dengan *output* yang telah diaktivasi dengan fungsi turunan aktivasi. Hasil perkalian ini merupakan faktor kesalahan pada lapis *output* dan akan digunakan untuk menghitung faktor kesalahan pada lapisan *hidden* dan untuk menghitung faktor peubah bobot pada vektor bobot menuju *output*.
3. Hitung besar faktor peubah bobot baru pada vektor yang menuju lapisan output dengan cara mengalikan hidden layer dengan *error* di output layer.
4. Pada lapisan *hidden*. Untuk menghitung faktor kesalahan masing-masing neuron lapisan *hidden* dilakukan: Masing-masing faktor kesalahan di *output* kalikan dengan bobot lama yang terkoneksi dengan neuron lapisan *output*. Kemudian hasil perkalian pada seluruh koneksi yang terhubung dengan masing-masing neuron pada lapis *hidden* akan dijumlahkan. Faktor kesalahan pada neuron lapis hidden akan digunakan untuk menghitung peubah bobot pada koneksi dari lapisan *input* menuju lapisan *hidden.*
5. Hitung nilai faktor peubah bobot baru pada tiap vektor yang menuju lapisan hidden dengan cara mengalikan *input layer* dan *error* di *hidden layer*.
6. Selesai.

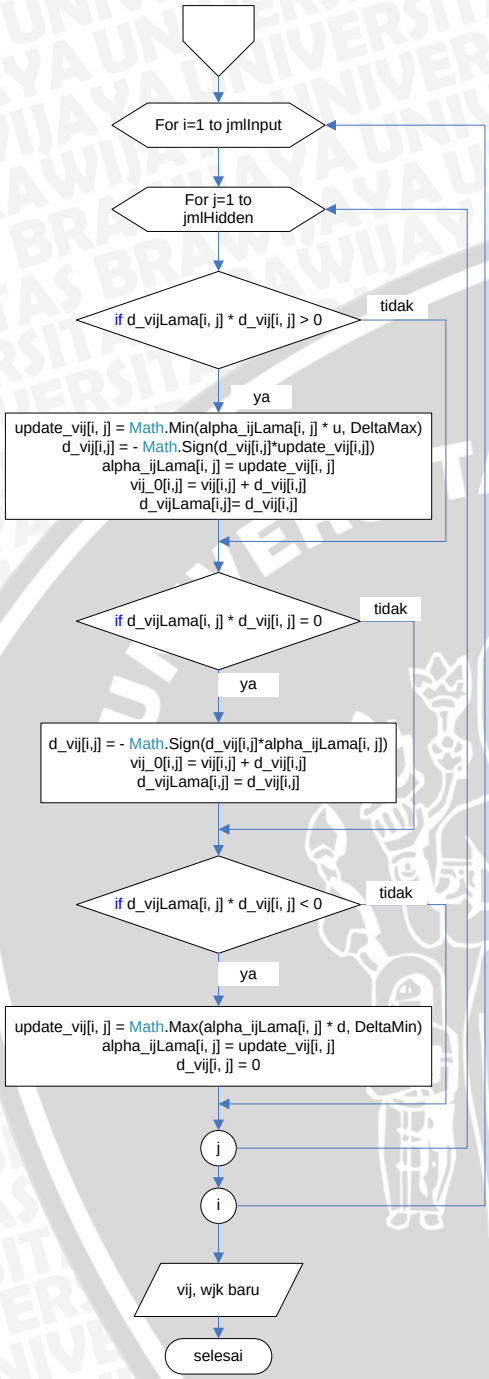


Gambar 3.7 *Flowchart* proses *Resilient Propagation*

3.3.2.4 Perbarui Bobot

Hasil dari proses *backpropagation* adalah nilai faktor peubah bobot yang digunakan untuk melakukan perubahan bobot. Proses perubahan bobot dilakukan untuk mendapatkan nilai bobot baru. Proses ini digambarkan dengan diagram alir pada Gambar 3.8.


Perbarui bobot
mulai
Input d_vij,
d_wjk, d_wj
for k = 0 to jmlHidden
if d_wj[k] * d_wjLama[k] > 0
tidak
ya
update_wjk[k] = Math.Min(alpha_jkLama[k] * u, DeltaMax)
d_wjk[k] = - Math.Sign(d_wj[k]*update_wjk[k])
alpha_jkLama[k] = update_wjk[k]
wjk_0[k] = wjk[k] + d_wjk[k]
d_wjLama[k]=d_wj[k]
if d_wj[k] * d_wjLama[k] < 0
tidak
ya
update_wjk[k] = Math.Max(alpha_jkLama[k] * d, DeltaMin)
alpha_jkLama[k] = update_wjk[k]
d_wjLama[k]=0
if d_wj[k] * d_wjLama[k] = 0
tidak
ya
d_wjk[k] = - Math.Sign(d_wj[k]*alpha_jkLama[k])
wjk_0[k] = wjk[k] + d_wjk[k]
d_wjLama[k] = d_wj[k]
k



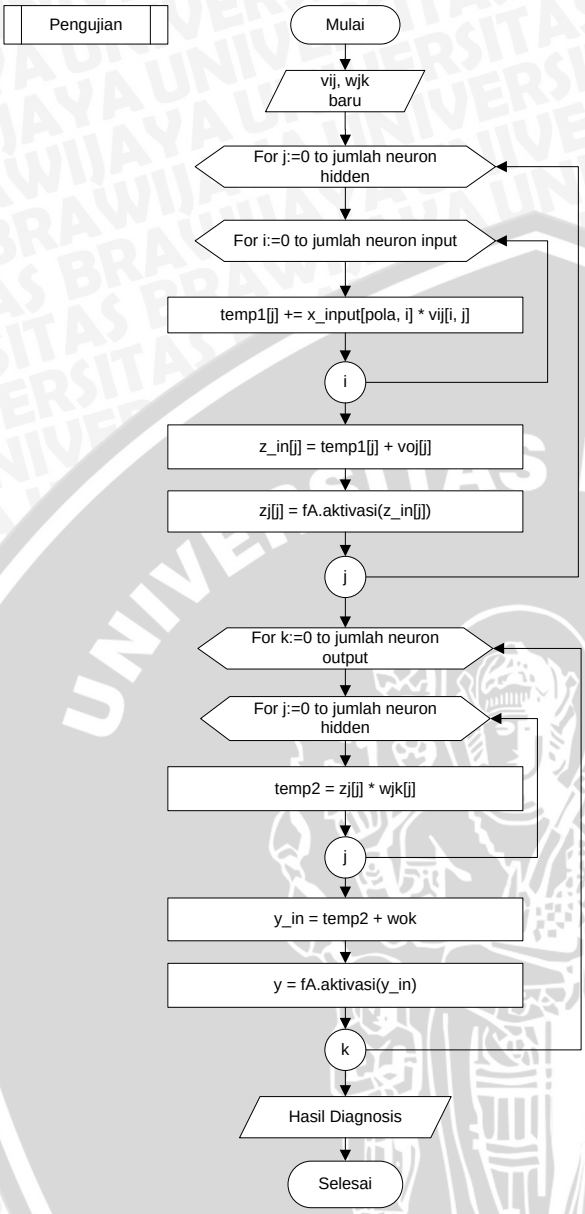
Gambar 3.8 *Flowchart* Perbarui Bobot

3.3.3 Proses pengujian

Tahap pengujian jaringan syaraf tiruan *backpropagation* diaplikasikan dengan hanya menggunakan tahap perambatan maju (*feed fordward*) dari algoritma pelatihan. Di dalam proses pengujian, *output* yang dikeluarkan oleh

jaringan tidak akan diproses lagi menuju ke *resilient propagation*. Secara keseluruhan langkah langkah pengujian algoritma *resilient propagation* diilustrasikan pada Gambar 3.9 dan langkah-langkahnya adalah sebagai berikut:

1. Mulai
2. Inisialisasi bobot-bobot yang diperoleh dari proses pelatihan jaringan.
3. Masukkan data pengujian.
4. Kalikan seluruh data *input* pada *neuron input* dengan bobot random pada masing-masing bobot koneksi bobot *Input* yang terhubung dengan *neuron input*. Kemudian jumlahkan seluruh vektor bobot yang menuju neuron *hidden* yang sama.
5. Lakukan aktivasi hasil penjumlahan tersebut pada masing-masing *neuron* di lapisan tersembunyi.
6. Kalikan seluruh data hasil aktivasi masing-masing neuron lapis *hidden* pada *neuron hidden* dengan bobot pada masing-masing koneksi bobot *output* yang terhubung dengan *neuron* pada lapis *hidden*. Kemudian jumlahkan seluruh vektor bobot yang menuju *neuron output* yang sama.
7. Lakukan aktivasi hasil penjumlahan tersebut pada masing-masing neuron di lapisan *output*.
8. Diperoleh keluaran yang berupa hasil diagnosis tingkatan risiko penyakit jantung koroner.
9. Selesai.



Gambar 3.9 *Flowchart* proses pengujian JST

3.4 Contoh Perhitungan Manual

Diketahui 5 data sampel penelitian berupa faktor risiko penyakit jantung koroner beserta nilai risikonya yang ditunjukkan pada Tabel 3.1.

Tabel 3.1 Tabel data masukan

	Glukosa Darah	Kolesterol	Trigliserida	Ureum
1	954	666,2	390	215
2	351	37,5	71,8	42
3	259	80,9	47,8	45
4	205	28,3	23,5	26
5	404	35,3	26,1	165

Langkah yang pertama yaitu untuk menentukan pola data pelatihan yaitu x1 adalah Glukosa darah, x2 adalah SGOT, x3 adalah SGPT, x4 adalah Ureum. Kemudian dilakukan normalisasi data input dengan persamaan 2.5 dan skala menggunakan normalisasi antara 0.1 sampai 0.9. Hasil dari normalisasi data ditunjukkan pada Tabel 3.2.

Tabel 3.2 Tabel hasil normalisasi data awal

	X1	X2	X3	X4
1	0.9	0.9	0.9	0.9
2	0.255941	0.1115379	0.20543	0.167725
3	0.157677	0.1659665	0.153042	0.180423
4	0.1	0.1	0.1	0.1
5	0.31255	0.1087788	0.105675	0.68836

Parameter awal untuk pelatihan yaitu :

Jumlah unit input = 4

Jumlah unit hidden = 5

Jumlah unit output = 1

u = 1.2

d = 0.5

Δ_0 = 0.1

Δ_{min} = 0 dan Δ_{max} = 50

Inisialisasi bobot dan bias awal dengan menggunakan metode Nguyen Widrow. Inisialisasi bobot-bobot dari *input layer* ke *hidden layer* mula-mula diberi nilai acak antara -0,05 hingga 0,05.

Tabel 3.3 Bobot awal ke lapisan *hidden* (v_{ij})

v_{ij}	Z_1	Z_2	Z_3	Z_4	Z_5
x_1	-0.04	0.02	-0.03	0.04	0.03
x_2	-0.01	-0.03	-0.03	-0.02	0.02
x_3	0.01	0.01	0.04	0.01	0.05
x_4	0.02	-0.01	0.03	0.01	0.02

β = faktor skala = $0.7\sqrt[n]{p}$ = $0.7\sqrt[4]{5}$ = 1.04674414

n = jumlah unit masukan = 4

p = jumlah unit tersembunyi = 5

Bias awal yang dipakai adalah bilangan acak antara -1.04674414 hingga 1.04674414

Menghitung nilai v_j

$$\|v_j\| = \sqrt{v^2{}_{1j} + v^2{}_{2j} + \cdots + v^2{}_{nj}}$$

$\|v_1\|$ =SQRT $[(-0,03)^2+(0,02)^2+(0,05)^2+(-0,01)^2]$

= 0.04690416

.

.

.

$\|v_5\|$ =SQRT $[(0,01)^2+(0,02)^2+(0,03)^2+(0,01)^2]$

= 0.06480741

Sehingga dihasilkan inisialisasi nilai v_j yang ditunjukkan pada Tabel 3.4.

Tabel 3.4 Inisialisasi nilai Vj

j	vj
1	0.04690416
2	0.03872983
3	0.06557439
4	0.04690416
5	0.06480741

Berdasarkan persamaan 2.9, maka diperoleh nilai bobot yang digunakan inisialisasi (V_{ij}) adalah sebagai berikut :

$$v_{11}(\text{baru}) = \frac{\beta v_{11}(lama)}{\|v_1\|} = \frac{1.04674414 \text{ x } (-0{,}04)}{0.04690416}$$

$$= -0{,}035751008$$

.

.

$$v_{65}(\text{baru}) = \frac{\beta v_{65}(lama)}{\|v_5\|} = \frac{1.04674414 \text{ x } (0{,}02)}{0.06480741}$$

$$= 0{,}478032395$$

Nilai bobot baru yang digunakan inisialisasi *input layer* ke *hidden layer* (V_{ij}) dapat dilihat pada Tabel 3.5.

Tabel 3.5 Nilai bobot baru V_{ij}

V_{ij}	Z_1	Z_2	Z_3	Z_4	Z_5
x_1	-0.89266639	0.54053636	-0.47888096	0.89266639	0.48454838
x_2	-0.2231666	-0.81080453	-0.47888096	-0.4463332	0.32303225
x_3	0.223166599	0.27026818	0.638507942	0.2231666	0.80758064
x_4	0.446333197	-0.27026818	0.478880956	0.2231666	0.32303225

Bias awal yang dipakai adalah bilangan acak antara -1.04674414 hingga 1.04674414. Nilai V_{oj} dapat dilihat pada Tabel 3.6.

Tabel 3.6 Nilai bias awal (V_{oj})

	V_{0j}
1	-1.04674414
2	-0.65673927
3	0.278637258
4	0.756732843
5	1.04674414

Inisialisasi bobot awal lapisan tersembunyi ke lapisan output (W_{jk}) diperoleh dari proses acak antara -0,05 hingga 0,05. Inisialisasi nilai W_{jk} dapat dilihat pada tabel 3.7.

Tabel 3.7 Bobot awal ke lapisan *output* (W_{jk})

	W_{jk}
Z_1	-0,04
Z_2	0,03
Z_3	-0,01
Z_4	0,02
Z_5	0,03

Sedangkan inisialisasi bias awal *hidden layer* ke *output layer* (W_{ok}) diperoleh dari proses acak antara -0,05 hingga 0,05. Inisialisasi nilai W_{ok} dapat dilihat pada tabel 3.8.

Tabel 3.8 Bias awal *hidden layer* ke lapisan *output layer* (W_{ok})

	W_{ok}
1	-0,04

Proses *Feedforward*

Hitung keluaran unit *hidden* (Z_j)

Masing-masing unit *hidden* menjumlahkan bobot sinyalnya, pada langkah ini menggunakan persamaan 2.11, maka diperoleh nilai keluaran unit hidden (Z_in_j) adalah sebagai berikut :

z_in_1 = -1.04674414 + ((0.9x -0.89266639) + (0.9x -0.2231666) + (0.9x 0.223166599) + (0.9x 0.446333197))

= -1.44844402

.

.

z_in_5 = -1.04674414 + ((0.9x 0.484548) + (0.9x 0.323032) + (0.9x 0.807581) + (0.9x 0.323032))

= 0.697630035

Hasil keluaran unit hidden (Z_in_j) ditunjukkan pada Tabel 3.9.

Tabel 3.9 Operasi pada *hidden*

j	Z_in
1	-1.44844402
2	-1.2899855
3	-0.90307985
4	-0.24334439
5	0.697630035

Kemudian hitung aktivasinya dengan menggunakan persamaan 2.12

Hasil aktivasi unit *hidden* dapat dilihat pada tabel 3.10.

Tabel 3.10 Hasil Aktivasi Operasi *hidden*

j	Z
1	0.8097587

2	0.78414459
3	0.71158188
4	0.56053761
5	0.33233799

Hitung keluaran unit Y_k

Masing-masing unit *output* menjumlahkan bobot sinyal *input*. Untuk menghitung nilai keluaran menggunakan persamaan 2.13.

y_in_k = -0,04 + ((0.8097587x-0,04) + (0.78414459x0,03) + (0.71158188x-0,01) + (0.56053761x0,02) + (0.33233799x0,03))

= -0.03480094

Hasil operasi pada *output neuron* dapat dilihat pada tabel 3.11.

Tabel 3.11 Operasi pada *output layer*

k	Y_in
1	-0.03480094

Kemudian hitung aktivasinya menggunakan persamaan 2.14. Hasil aktivasinya dapat dilihat pada tabel 3.12.

Tabel 3.12 Hasil aktivasi y_in

k	Y
1	0.50869935

Hitung faktor δ di unit *output*

Setiap unit *output* y menerima vektor hasil yang diinginkan (t) untuk data masukan tersebut. Perhitungan nilai faktor δ di unit *output* menggunakan persamaan 2.15. Nilai faktor δ di unit *output* dapat dilihat pada Tabel 3.13.

Tabel 3.13 Nilai faktor δ di unit *output* y_k

k	δ
1	0.06139399

Hitung nilai ΔE_{jk} menggunakan persamaan 2.18. Nilai ΔE_{jk} dapat dilihat pada Tabel 3.14

Tabel 3.14 Nilai ΔE_{jk}

	ΔE_{jk}
Z1	0.04971432
Z2	0.04814177
Z3	0.04368685
Z4	0.03441364
Z5	0.02040356

Hitung nilai informasi error δj pada tiap-tiap unit *hidden* dengan menggunakan persamaan 2.16. Hasil perhitungan kesalahan di unit *hidden* dapat dilihat pada tabel 3.15.

Tabel 3.15 Nilai informasi error δj

	δj
Z1	-0.00037831
Z2	0.00031175
Z3	-0.000126
Z4	0.00030247
Z5	0.00040868

Hitung nilai ΔE_{ij} menggunakan persamaan 2.17. Nilai ΔE_{ij} dapat dilihat pada Tabel 3.16.

Tabel 3.16 Nilai ΔE_{ij}

ΔE_{ij}	Z_1	Z_2	Z_3	Z_4	Z_5
x_1	-0.00034048	0.00028057	-0.0001134	0.00027222	0.00036781
x_2	-0.00034048	0.00028057	-0.0001134	0.00027222	0.00036781
x_3	-0.00034048	0.00028057	-0.0001134	0.00027222	0.00036781
x_4	-0.00034048	0.00028057	-0.0001134	0.00027222	0.00036781

Hitung nilai *learning rate* yang baru pada bobot antara *input* ke *hidden layer* dengan menggunakan persamaan 2.19.

Karena $\Delta E_{ijlama} . \Delta E_{ij} \geq 0$, maka

$\Delta_{ij} = \min (\Delta_{ijlama} . u, \Delta_{max}) = \min(0.012, 50) = 0.012$

Perbarui nilai bobot antara *input* ke *hidden layer* menggunakan persamaan 2.18.

Tabel 3.17 Bobot V_{ij} baru

Vij	Z_1	Z_2	Z_3	Z_4	Z_5
x_1	-0.90466639	0.55253636	-0.49088096	0.90466639	0.49454838
x_2	-0.2351666	-0.79880453	-0.49088096	-0.4343332	0.33303225
x_3	0.211166599	0.28226818	0.626507942	0.2351666	0.81758064
x_4	0.434333197	-0.25826818	0.466880956	0.2351666	0.33303225

Menghitung nilai *learning rate* yang baru pada bobot antara *hidden* ke *output layer* (Δ_{jk}) menggunakan persamaan 2.22.

Karena ΔE_{jk}lama * $\Delta E_{jk} \geq 0$ maka

Δ_{jk}= min(Δ_{jk}lama u, Δ_{max}) = min (0.0144, 50) = 0.0144

Perbarui nilai bobot antara *hidden* ke *output layer* menggunakan persamaan 2.23.

Tabel 3.17 Bobot W_{jk} baru

	W_{jk}
Z_1	-0.0544
Z_2	0.0156
Z_3	-0.0244
Z_4	0.0056
Z_5	0.0156

Proses Pengujian

Inisialisasi bobot yang diperoleh dari proses pelatihan

Hitung keluaran unit *hidden* (Z_j)

Masing-masing unit *hidden* menjumlahkan bobot sinyalnya, pada langkah ini menggunakan persamaan 2.11, maka diperoleh nilai keluaran unit hidden (Z_in_j) adalah sebagai berikut :

z_in_1 = -1.44844402+ ((0.9x-0.90466639) + (0.9x-0.2351666) + (0.9x0.211166599) + (0.9x0.434333197))

= -1.49164402

.

.

z_in_5 = -1.44844402+ ((0.9x0.49454838) + (0.9x0.33303225) + (0.9x0.81758064) + (0.9x0.33303225))

= 0.733630035

Hasil keluaran unit hidden (Z_in_j) ditunjukkan pada Tabel 3.18.

Tabel 3.18 Operasi pada *hidden*

j	Z_in
1	-1.49164402
2	-1.2467855
3	-0.94627985

4	-0.20014439
5	0.733630035

Kemudian hitung aktivasinya dengan menggunakan persamaan 2.12

Hasil aktivasi unit *hidden* dapat dilihat pada tabel 3.19.

Tabel 3.19 Hasil Aktivasi Operasi *hidden*

j	Z
1	0.816324752
2	0.776742774
3	0.720366283
4	0.549869701
5	0.324398755

Hitung keluaran unit Y_k

Masing-masing unit *output* menjumlahkan bobot sinyal *input*. Untuk menghitung nilai keluaran menggunakan persamaan 2.13.

y_in_k = -0.0544 + ((0.816324752x-0,0544) + (0.776742774x0.0156) + (0.720366283x-0.0244) + (0.549869701x0.0056) + (0.324398755x0.0156))

= -0.09612793

Hasil operasi pada *output neuron* dapat dilihat pada tabel 3.20.

Tabel 3.20 Operasi pada *output layer*

k	Y_in
1	-0.09612793

Kemudian hitung aktivasinya menggunakan persamaan 2.14. dengan menggunakan fungsi sigmoid biner pada proses aktivasi nya. Hasil aktivasinya dapat dilihat pada tabel 3.21.

Tabel 3.21 Hasil aktivasi y_in

k	Y
1	0.524013477

Misalkan threshold = 0,5, jika y > 0,5 maka output yang diberikan adalah 1, namun jika nilai y , 0,5 maka output yang diberikan 0. Dengan demikian dari proses pengujian, diperoleh nilai keluaran dimana y = 0,524013477 dan > 0,5 maka diprediksi pasien terkena diabetes.

3.5 Rancangan Uji Coba

3.5.1 Rancangan Uji Coba Pengaruh Jumlah *Neuron Hidden* pada layer

Uji coba ini dilakukan dengan menggunakan 5 (lima jumlah *Hidden Neuron* yang berbeda yaitu 5, 15, 30, 50, dan 70. Uji coba ini dirancang dengan menggunakan Tabel 3.22.

Tabel 3.22 Rancangan uji coba pengaruh Jumlah *Neuron Hidden* pada layer

Percobaan ke -	Jumlah *neuron hidden*	MSE
1		
2		
3		
4		
5		

3.5.1 Rancangan uji coba pengaruh nilai *learning rate*

Uji coba ini dilakukan dengan menggunakan beberapa nilai *learning rate* yang berbeda untuk mengetahui pengaruhnya terhadap nilai MSE. Percobaan dilakukan sebanyak 9 kali dimulai dengan *learning rate* terkecil sampai learning rate terbesar yaitu 0,1 – 0,9**.** Uji coba ini dirancang dengan menggunakan Tabel 3.23.

Tabel 3.23 Hasil percobaan pengaruh *learning rate*

Percobaan ke	*Learning Rate*	MSE
1	0.1	

2	0.2	
3	0.3	
4	0.4	
5	0.5	
6	0.6	
7	0.7	
8	0.8	
9	0.9	

3.6 Hasil Pengujian

3.6.1 Rancangan Uji Coba Pengaruh Nilai *Threshold*

Hasil dari model jaringan saraf tiruan berupa nilai kontinyu, sedangkan nilai respon pada penelitian ini bersifat binary, yaitu 1 untuk diabetes dan 0 untuk non diabetes. Untuk mengatasi nilai kontinyu yang dihasilkan oleh jaringan saraf tiruan, dilakukan proses diskritisasi. Proses ini dilakukan dengan cara menentukan batas/*threshold* tertentu untuk tiap nilai respon, dikarenakan nilai respon adalah 1 dan 0 maka nilai *threshold* berada antara interval tersebut.Penentuan nilai ini dilakukan secara *trial* dan *error*.

Uji coba pada penelitian ini dilakukan dengan menggunakan beberapa nilai yang berbeda untuk mengetahui pengaruhnya terhadap keakurat prediksi. Pengujian dilakukan sebanyak 9 kali dimulai dengan nilai *threshold* terkecil sampai nilai *threshold* terbesar yaitu 0,1 – 0,9. Uji coba ini dirancang dengan menggunakan Tabel 3.24.

Tabel 3.40 Hasil percobaan pengaruh *threshold* terhadap keakuratan prediksi

Percobaan ke	*Threshold*	Jumlah Benar	Jumlah Salah	Keakuratan
1	0.1			
2	0.2			
3	0.3			

4	0.4			
5	0.5			
6	0.6			
7	0.7			
8	0.8			
9	0.9			

BAB IV
HASIL DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan pada sub bab ini adalah lingkungan perangkat keras dan lingkungan perangkat lunak yang digunakan untuk mendiagnosis penyakit diabetes mellitus (DM) bagi penderita DM berdasarkan faktor risiko menggunakan metode jaringan saraf tiruan *resilient propagation*.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan system diagnosis penyakit diabetes mellitus (DM) bagi penderita DM berdasarkan faktor risiko menggunakan jaringan saraf tiruan *resilient propagation* adalah

1. Prosesor Intel® Core™ 2 Duo T5800 2.0Ghz
2. Memori 3072MB RAM
3. Harddisk dengan kapasitas 320GB
4. Monitor 14” Widescreen
5. Keyboard
6. Mouse

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem mendiagnosis penyakit diabetes mellitus (DM) bagi penderita DM berdasarkan faktor risiko menggunakan jaringan saraf tiruan *resilient propagation* adalah

1. Sistem Operasi *Microsoft Windows 7 Ultimate*
2. Microsoft Visual C# 2010 Express Edition

4.2 Implementasi Program

Berdasarkan analisa dan perancangan proses yang telah dipaparkan pada Bab III, maka pada bab ini akan dijelaskan proses-proses implementasinya.

4.2.1 Struktur Data

Untuk memudahkan dalam implementasi jaringan saraf tiruan, penentuan struktur data harus dilakukan dengan baik. Struktur data yang digunakan dalam sistem merupakan *array* 1 dimensi dan *array* 2 dimensi. Struktur data tersebut dapat dilihat pada *source code* 4.1.

```
double[,] x_input, vij_0,vij, d_vij, Tempdata;
double[] target, vj, voj_0,voj, d_voj;
double[] wjk_0,wjk,S_hidden,S_in,z_in,zj,d_wj;
double[] output,MSE;
double[,] vij_Optimum;
double[] voj_Optimum,wjk_Optimum;
```

Source code 4.1 Struktur Data Jaringan Saraf Tiruan

4.2.2 Proses Mencari Nilai Maksimum & Minimum

Fungsi untuk mencari nilai minimum dapat dilihat pada *source code* 4.2.

```
public void MaxMin(double[] TempData)
{
   double max = TempData[0];
   double min = TempData[0];

   for (int j = 0; j < TempData.Length; ++j)
   {
    if (max < TempData[j])
        max = TempData[j];

    if (min > TempData[j])
        min = TempData[j];
   }
   maxi = max;
   mini = min;
}
```

Source Code 4.2 Proses Mencari Nilai Minimum dan Maksimum

Fungsi ini digunakan untuk mencari nilai minimum dan maksimum dari data pasien penyakit diabetes mellitus yang telah di-*load*. Nilai tersebut nantinya akan digunakan dalam proses normalisasi.

4.2.3 Proses Normalisasi Data

Fungsi untuk melakukan proses normalisasi dapat dilihat pada *source code* 4.3

```
public void ProsesNormalisasi(double[] Tempdata)
{
    //normalisasi data pada tempdata
    for (int j = 0; j < Tempdata.Length; j++)
    {
     Tempdata[j] = 0.8 * (Tempdata[j] - mini) / (maxi - mini) + 0.1;
    }
}
```

Source Code 4.3 Proses Normalisasi Data

Fungsi tersebut digunakan untuk melakukan normalisasi terhadap data penyakit diabetes yang diderita oleh pasien agar *range* data yang didapatkan berkisar antara 0,1 hingga 0,9.

4.2.4 Proses Inisialisasi Bobot Awal

Proses ini bertugas untuk menginisialiasasi nilai awal bobot dari *unit-unit* yang berada pada *layer input* ke *unit-unit* yang berada pada *layer hidden,* demikian juga dengan bobot dari *unit-unit* yang berada pada *layer hidden* ke *unit-unit* yang berada pada *layer output.* Bobot diinisialisasi secara acak dengan nilai berkisar antara -0,05 sampai 0,05 dengan menggunakan algoritma inisialisasi Nguyen Widrow. Fungsinya ditunjukkan pada *source code* 4.4

```
public void ProsesRandomBobot(int jmlInput, int jmlHidden){

    //inisialisasi bobot awal dengan nguyen widrow
    double a = 1 / Convert.ToDouble(jmlInput);
    double beta = 0.7 * Math.Pow(jmlHidden, a);
    double plusBeta = beta;
    double minBeta = -1 * beta;

    vij = new double[jmlInput, jmlHidden];
    vj = new double[jmlHidden];
    voj = new double[jmlHidden];
    wjk = new double[jmlHidden];

    //inisialisasi bobot awal vij diantara nilai -0.05 dan 0.05
    for (int i = 0; i < jmlInput; i++)
    {
     for (int j = 0; j < jmlHidden; j++)
     {
       vij[i, j] = RandomToScaled(-0.05, 0.05);
     }
    }
```

```
//hitung ||vj||
for (int j = 0; j < jmlHidden; j++)
{
 vj[j] = 0;
 for (int i = 0; i < jmlInput; i++)
 {
   vj[j] += Math.Pow(vij[i, j], 2);
 }
 vj[j] = Math.Sqrt(vj[j]);
}

//nilai bobot yg digunakan inisialisasi (vij)
for (int i = 0; i < jmlInput; i++)
{
 for (int j = 0; j < jmlHidden; j++)
 {
  vij[i, j] = beta * vij[i, j] / vj[j];
 }
}

//inisialisasi bias awal(voj) menggunakan bilangan acak
for (int j = 0; j < jmlHidden; j++)
{
 voj[j] = RandomToScaled(minBeta, plusBeta);
}

//inisialisasi bobot awal hidden ke output (Wjk)
for (int j = 0; j < jmlHidden; j++)
{
 wjk[j] = RandomToScaled(-0.05, 0.05);
}

//inisialisasi bias awal hidden layer ke output(Wok)
 {
  wok = RandomToScaled(-0.05, 0.05);
 }
```

Source Code 4.4 Proses Inisialisasi Bobot Awal

4.2.5 Proses feedforward

Proses *feedforward* digunakan untuk mencari nilai *output* dari *neuron hidden* dan *neuron output.* Nilai tersebut kemudian akan diaktivasi dengan menggunakan fungsi *sigmoid*. Fungsinya ditunjukkan pada *source code* 4.5.

```
public void Prosesfeedfoward(int pola,double[,] x_input,double[,]
vij, double[] voj, double[] wjk, double wok)
{
  int jmlHidden = vij.GetLength(1);
  int jmlInput = vij.GetLength(0);
  double[] temp1 = new double[jmlHidden];
   z_in = new double[jmlHidden];
  zj= new double[jmlHidden];

  for (int j = 0; j < jmlHidden; j++)
  {
     temp1[j] = 0;
     for (int i = 0; i < jmlInput; i++)
     {
       temp1[j] += x_input[pola, i] * vij[i, j];
     }
     z_in[j] = temp1[j] + voj[j];
     zj[j] = fs.aktivasi(z_in[j]);
  }
  double temp2 = 0;
  for (int j = 0; j < jmlHidden; j++)
  {
     temp2 += zj[j] * wjk[j];
  }
  y_in = temp2 + wok;
  y = fs.aktivasi(y_in);
}
```

Source Code 4.5 Fungsi *Feedfoward*

4.2.6 Proses *Resilient propagation*

Proses *resilient propagation* terdiri dari perhitungan faktor kesalahan di unit keluaran, perhitungan nilai perubahan bobot lapisan tersembunyi ke lapisan keluaran, perhitungan perubahan bias lapisan tersembunyi ke lapisan keluaran, perhitungan faktor kesalahan di unit tersembunyi, perhitungan nilai perubahan bobot lapisan masukan ke lapisan tersembunyi, dan perhitungan perubahan bias lapisan masukan ke lapisan tersembunyi. Fungsi *resilient propagation* dapat dilihat pada *source code* 4.6.

```
public void resilientpropagation()
 {
  zj = ff.Get_Zj();
  z_in = ff.Get_Z_in();
  y_in = ff.Get_Y_in();
  y = ff.Get_y();

  //hitung informasi kesalahan , Sk = erroroutput
  Sk = (target[pola] - output[pola]) * fa.turaktivasi(y_in);

  //hitung koreksi bobot
  for (int k = 0; k < this.jmlHidden; k++)
  {
   d_wj[k] = (alpha* Sk) * zj[k];
  }

  //hitung koreksi bias u/ perbaiki bobot bias(wok)
  d_wok = alpha * Sk;

  //hitung kesalahan di hidden layer
  for (int j = 0; j < this.jmlHidden; j++)
  {
   S_in[j] = Sk * wjk[j];
   S_hidden[j] = S_in[j] * fa.turaktivasi(z_in[j]);
  }
  for (int j = 0; j < this.jmlHidden; j++)
  {
    for (int i = 0; i < this.jmlInput; i++)
    {
     d_vij[i, j] = alpha * S_hidden[j] * x_input[pola, i];
    }
    d_voj[j] = alpha * S_hidden[j];
  }
 }
```

Source Code 4.6 Fungsi *Resillientpropagation*

4.2.7 Proses update bobot dan bias

Proses *update* bobot dan bias digunakan untuk menghitung bobot dan bias baru dari unit-unit yang berada pada *layer input* ke unit-unit yang berada pada layer *hidden*, dan juga bobot dan bias baru dari unit-unit yang berada pada layer *hidden* ke unit-unit yang berada pada layer *output*. Fungsi perbarui bobot dan bias dapat dilihat pada *source code* 4.7.

```
public void perbaruibobotresillient(){
 for (int k = 0; k < jmlHidden; k++)
 {
  if (d_wj[k] * d_wjLama[k] > 0)
  {
    update_wjk[k] = Math.Min(alpha_jkLama[k] * u, DeltaMax);
    d_wjk[k] =  - Math.Sign(d_wj[k]*update_wjk[k]);
    alpha_jkLama[k] = update_wjk[k];
    wjk_0[k] = wjk[k] + d_wjk[k];
    d_wjLama[k]=d_wj[k];
  }

  else if (d_wj[k] * d_wjLama[k] < 0)
  {
    update_wjk[k] = Math.Max(alpha_jkLama[k] * d, DeltaMin);
    alpha_jkLama[k] = update_wjk[k];
    d_wjLama[k]=0;
  }

  else if(d_wj[k] * d_wjLama[k] == 0)
  {
    d_wjk[k] =  - Math.Sign(d_wj[k]*alpha_jkLama[k]);
    wjk_0[k] = wjk[k] + d_wjk[k];
    d_wjLama[k] = d_wj[k];
  }
 }

 for (int i = 1; i < jmlInput; i++)
 {
    for (int j = 1; j < jmlHidden; j++)
    {
      if (d_vijLama[i, j] * d_vij[i, j] > 0)
      {
       update_vij[i, j] = Math.Min(alpha_ijLama[i, j] * u, DeltaMax);
       d_vij[i,j] = - Math.Sign(d_vij[i,j]*update_vij[i,j]);
       alpha_ijLama[i, j] = update_vij[i, j];
       vij_0[i,j] = vij[i,j] + d_vij[i,j];
       d_vijLama[i,j]= d_vij[i,j];
      }
    else if (d_vijLama[i, j] * d_vij[i, j] == 0)
    {
      d_vij[i,j] = - Math.Sign(d_vij[i,j]*alpha_ijLama[i, j]);
      vij_0[i,j] = vij[i,j] + d_vij[i,j];
      d_vijLama[i,j] = d_vij[i,j];
    }
    else if (d_vijLama[i, j] * d_vij[i, j] < 0)
    {
      update_vij[i, j] = Math.Max(alpha_ijLama[i, j] * d, DeltaMin);
      alpha_ijLama[i, j] = update_vij[i, j];
      d_vij[i, j] = 0;
    }
   }
  }
```

Source Code 4.7 Fungsi Perbarui bobot

4.2.8 Proses nilai MSE (*Mean Square Error*)

Proses menghitung nilai MSE (*Mean Square Error*) digunakan menilai kinerja jaringan yang dilatih. Semakin kecil nilai MSE maka dapat dianggap bahwa arsitektur jaringan semakin baik. Fungsi MSE dapat diliat pada *source code* 4.8.

```
public void getMSE(double[] target, double[] y)
{
  int jmlData = target.Length;
  double TempMSE = 0.0;
  for (int i = 0; i < jmlData; i++)
  {
    TempMSE += Math.Pow(target[i] - y[i], 2);
  }
  mse = TempMSE / jmlData;
 }
```

Source Code 4.8 Fungsi MSE

4.2.9 Proses Pelatihan

Dalam proses pelatihan terdapat beberapa fungsi yang dijalankan diantaranya inisialisasi bobot awal, *feedfoward*, *resillientpropagation*, perbaikan bobot dan perhitungan MSE(*Mean Square Error*). Implementasi proses pelatihan dapat dilihat pada *source code* 4.9.

```
private void BtProses_Click(object sender, EventArgs e){
  //proses pelatihan
  alpha_ijLama[0, 0] = alpha;
  alpha_jkLama[0] = alpha;
  biasjLama[0] = alpha;
  biaskLama = alpha;
  inisialisasiBobotAwal();
  tabPelatihan.SelectedIndex = 2;
  BtStop.Enabled = true;
  epoch = 0;

  while (status == false)
  {
    pola = 0;
    while (pola < jmlData)
     {
       ff.Prosesfeedforward(pola, x_input, vij, voj, wjk, wok);
       output[pola] = ff.Get_y();
       resilientpropagation();
       perbaruibobotresilient();
       pola++;
     }
```

```
    m.getMSE(target,output);
    MSE[epoch] = m.get_MSE();

    DGProses.Rows.Add();
    DGProses.Rows[epoch].Cells[0].Value = epoch + 1;
    DGProses.Rows[epoch].Cells[1].Value = MSE[epoch];
    Application.DoEvents();

    if (MSE[epoch] < target_error)
    {
      txtMSE.Text = Convert.ToString(MSE[epoch]);
      txtCEpoch.Text = Convert.ToString(epoch + 1);
      status = true;
      BtSimpan.Enabled = true;
     }

    if (epoch == maxEpoch - 1)
    {
      txtMSE.Text = Convert.ToString(MSE[epoch]);
      txtCEpoch.Text = Convert.ToString(epoch + 1);
      status = true;
      BtSimpan.Enabled = true;
      BtStop.Enabled = false;
     }
     epoch++;
   }
```

Source Code 4.9 Fungsi Proses Pelatihan

4.2.10 Proses Pengujian

Proses pengujian merupakan tahap dimana nilai bobot optimum yang dihasilkan dari proses pelatihan digunakan sebagai bobot awal untuk menguji data diabetes. Proses pengujian ini dilakukan menggunakan proses *feedforward*. Implementasi proses pengujian dapat dilihat pada *source code* 4.10.

```
private void BtPrediksi_Click(object sender, EventArgs e){
   threshold = Convert.ToDouble(txtThreshold.Text);
   tabControl1.SelectedIndex = 2;
   z_in = new double[jmlHidden];
   zj = new double[jmlHidden];
   y = new double[jmlData];
   prediksi = new string[jmlData];

   pola = 0;
   while (pola < jmlData)
   {
    ff.Prosesfeedforward(pola, x_input, vij, voj, wjk, wok);
    y[pola] = ff.Get_y();
```

```
    //penggunaan threshold
    if(y[pola] < threshold)
    {
     prediksi[pola] = "non diabetes";
    }
    else
    {
     prediksi[pola] = "diabetes";
    }

    DGAkurasi.Rows.Add();
    DGAkurasi.Rows[pola].Cells[0].Value = no[pola];
    DGAkurasi.Rows[pola].Cells[1].Value = Keterangan[pola];
    DGAkurasi.Rows[pola].Cells[2].Value = prediksi[pola];
    pola++;
    Application.DoEvents();
    }
     BtSimpan.Enabled = true;
     BtPrediksi.Enabled = false;
  }
```

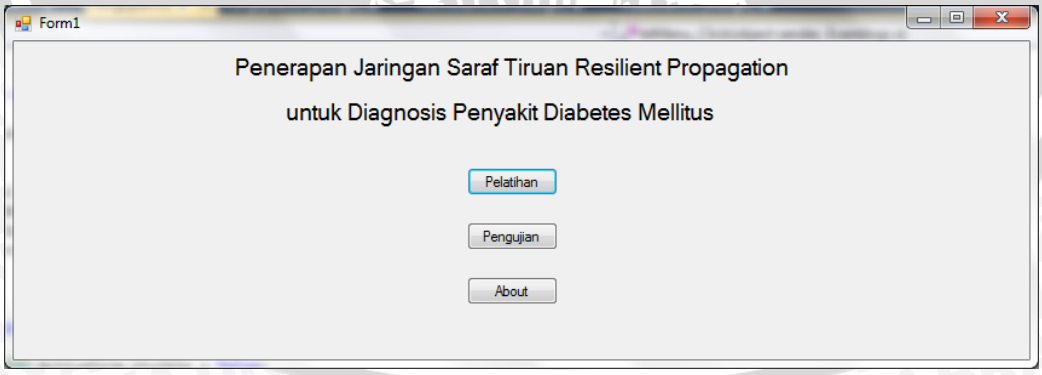
Source Code 4.10 Fungsi Proses Pengujian

4.3 Implementasi Antarmuka

Berdasarkan rancangan antarmuka pada sub bab 3.4 maka dihasilkan antarmuka yang terdiri dari :

4.4.1 *Form* Utama

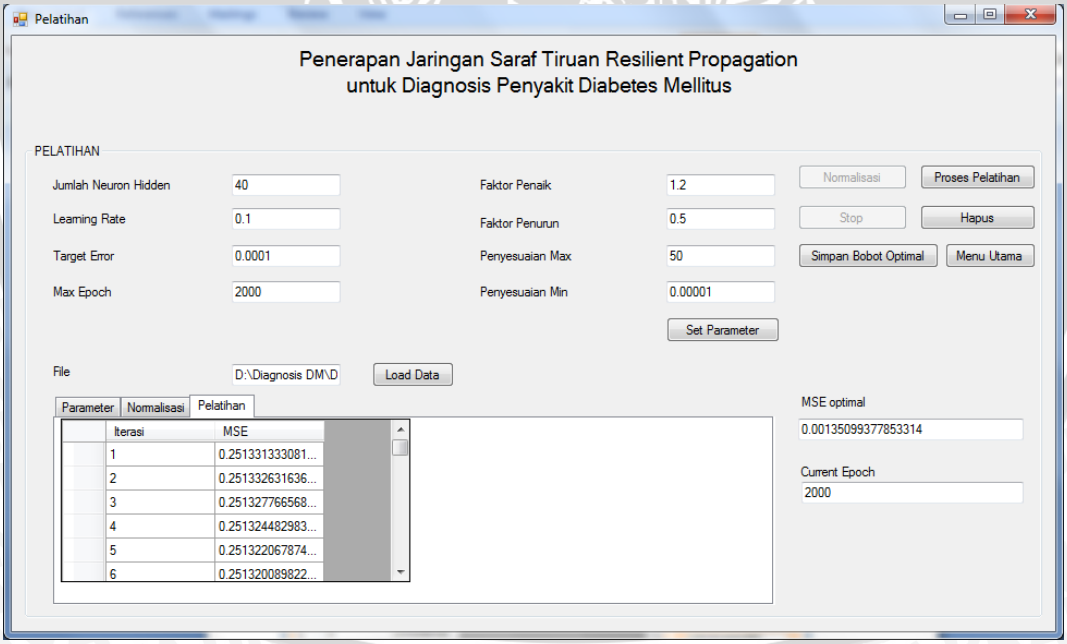
Form utama memiliki dua menu utama yang bias dipilih, yaitu menu Pelatihan dan Menu Pengujian. Form utama dapat dilihat pada gambar 4.1.



Gambar 4.1 *Form* Utama

4.3.2 *Form* Pelatihan

Pada form pelatihan JST, pertama yang dilakukan *user* adalah membuka *file* data hasil pemeriksaan pasien penderita penyakit diabetes yang akan dilatih. Data tersebut kemudian ditampilkan kedalam *datagridview* pada tab pelatihan. *User* dapat melakukan proses pelatihan pada sistem dengan menentukan parameter-parameter jaringan yang terdiri dari jumlah unit *hidden*, *learning rate*, target *error*, faktor penaik, faktor penurun, penyesuaian maksimum, penyesuaian minimum dan maksimum *epoch* yang diinginkan. Jika parameter sudah diatur, maka *user* dapat menekan tombol *Normalisasi* untuk proses normalisasi data latih. Setelah dilakukan proses normalisasi data latih, selanjutnya dilakukan proses pelatihan data dengan menekan tombol *Proses Pelatihan*. Proses pelatihan data ini akan ditampilkan nilai MSE (*Mean Square Error*) pada tiap iterasi hingga mencapai batas target *error* atau maksimum *epoch* yang sudah ditentukan. Setelah proses pelatihan selesai, maka bobot akan disimpan dengan menekan tombol *Simpan Bobot*. Form pelatihan data dapat dilihat pada gambar 4.2.

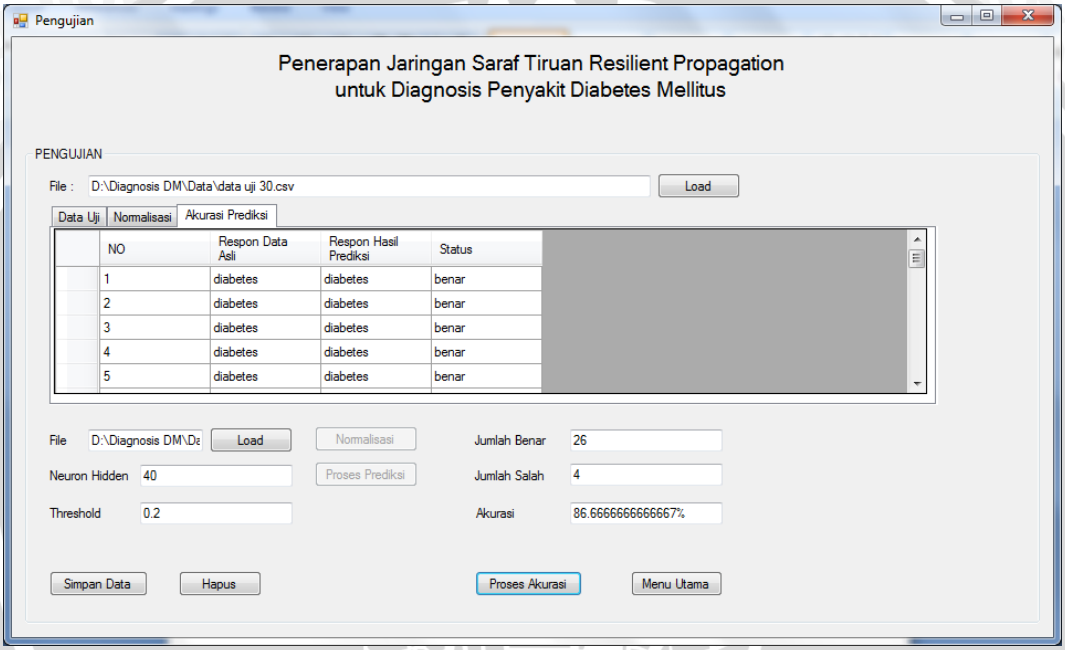


Gambar 4.2 *Form* Pelatihan

4.3.3 *Form* Pengujian

Pada form pengujian JST , *user* dapat membuka *file* data pasien penderita penyakit diabetes yang akan diuji. Kemudian parameter-parameter yang telah

ditentukan dan bobot pelatihan yang telah disimpan dari hasil pelatihan pada *form* pelatihan akan di-*load* pada *form* pengujian. Lalu dilakukan pengujian *threshold* dengan menginput *range* nilai 0,1 – 0,9. Setelah itu proses pengujian dilakukan dengan menekan tombol *Normalisasi* terlebih dahulu untuk menormalisasi data yang akan diuji dilanjutkan dengan menekan tombol *Proses Prediksi* untuk mengetahui hasil *output* JST. Hasil akurasi untuk data mana saja yang sesuai atau tidak dengan hasil diagnosis yang sebenarnya dapat dilihat dengan menekan tombol *Proses Akurasi*. Form pengujian dapat dilihat pada gambar 4.3.



Gambar 4.3 *Form* Pengujian

4.4 Uji Coba dan Analisa Sistem

Untuk memperoleh struktur jaringan saraf tiruan terbaik yang digunakan untuk mendiagnosis diabetes, maka dilakukan uji coba terhadap sistem. Uji coba ini dilakukan dengan melatih jaringan saraf tiruan dengan parameter-pamameter yang berbeda, yang nantinya akan diambil satu struktur jaringan yang terbaik yang digunakan untuk melakukan diagnosis diabetes.

4.4.1 Hasil Percobaan

Percobaan dilakukan guna menentukan jumlah *hidden neuron*, nilai *learning rate* agar didapatkan struktur jaringan saraf tiruan yang terbaik.

4.4.1.1 Pengaruh Jumlah *Neuron* pada *Hidden Layer*

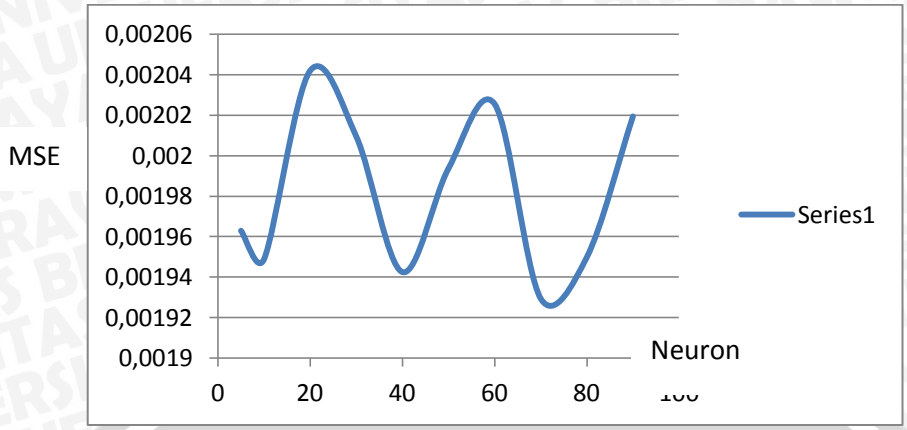
Untuk menentukan jumlah *hidden neuron* pada sistem dilakukan melalui proses *trial and error*. Percobaan ini dilakukan dengan cara membandingkan nilai MSE yang diperoleh pada tiap jumlah *hidden neuron*, kombinasi jumlah *hidden neuron* yang terbaik nantinya akan menghasilkan MSE yang terkecil.

Dalam percobaan ini , jumlah *neuron* pada *hidden layer* yang diujikan yaitu 5, 10, 20, 30, 40, 50, 60, 70, 80, 90 dengan menggunakan nilai parameter-parameter tetap lainnya yaitu *learning rate* 0,1; target *error* 0,0001; dan iterasi sebanyak 5.000 . Data hasil percobaan untuk mengetahui pengaruh jumlah unit *neuron* pada *hidden layer* dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil percobaan pengaruh neuron pada *hidden layer*

Percobaan ke	Jumlah *neuron* pada *hidden layer*	MSE
1	5	0.00196291090135528
2	10	0.00194886233785978
3	20	0.00204198112652202
4	30	0.0020094655418535
5	40	0.00194248081453194
6	50	0.00199369900743437
7	60	0.00202551887867916
8	70	0.00192939138697815
9	80	0.00194983684856778
10	90	0.00201939810988267

Berdasarkan tabel 4.1 dapat dilihat bahwa nilai MSE yang terkecil pada percobaan ke – 8 yaitu sebesar 0.00192939138697815 dengan jumlah neuron pada hidden layer sebanyak 70 unit. Pada Gambar 4.4 disajikan grafik pengaruh penambahan jumlah *neuron* pada *hidden layer* terhadap perubahan MSE.



Gambar 4.4 Grafik pengaruh penambahan *neuron hidden* terhadap perubahan MSE.

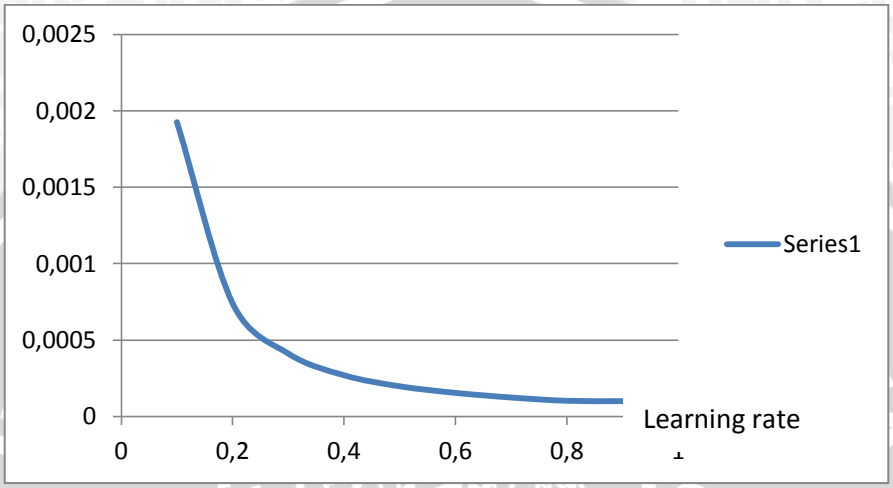
4.4.1.2 Pengaruh Nilai *Learning Rate*

Pada percobaan ini akan dilakukan perbandingan nilai MSE yang diperoleh dari uji coba beberapa nilai *learning rate* yaitu antara 0 sampai 1. Percobaan dilakukan sebanyak 9 kali dimulai dengan *learning rate* terkecil sampai *learning rate* terbesar yaitu 0,1 – 0,9. Percobaan ini menggunakan jumlah *neuron hidden* sebanyak 70 unit karena pada percobaan sebelumnya diketahui model ini lebih optimal. Percobaan dilakukan dengan maksimum *epoch* sebanyak 5.000 iterasi. Data hasil percobaan untuk mengetahui pengaruh *learning rate* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil percobaan pengaruh *learning rate*

Percobaan ke	*Learning Rate*	MSE
1	0.1	0.00192674272052631
2	0.2	0.0007391084477891 11
3	0.3	0.0004110531090236 02
4	0.4	0.0002693615704824 99
5	0.5	0.0001969500258376 06
6	0.6	0.0001540826936949 74
7	0.7	0.0001237002063614 84
8	0.8	0.0001022193677639 55
9	0.9	0.0000999732406311761

Berdasarkan tabel 4.2 dapat dilihat bahwa nilai MSE terkecil berada pada *learning rate* 0.9, yaitu sebesar 0.0000999732406311761. Sehingga *learning rate* optimal yang digunakan untuk pelatihan jaringan saraf tiruan adalah 0.9. Pada Gambar 4.5 disajikan grafik pengaruh *learning rate* terhadap perubahan MSE.



Gambar 4.5 Grafik pengaruh penambahan *learning rate* terhadap perubahan MSE.

4.4.3 Hasil Pengujian

Dari hasil pelatihan didapatkan bobot terbaik yang akan digunakan untuk menguji kemampuan sistem dalam memberikan prediksi diabetes. Pada pengujian ini yang dilakukan adalah membandingkan *output* sistem dengan data *real* sehingga dapat diketahui nilai akurasi keberhasilan sistem.

Berikut ini dilakukan pengujian kebenaran prediksi terhadap 30 data uji menggunakan arsitektur JST optimal yang didapatkan dari pelatihan yang sudah dilakukan yaitu dengan menggunakan parameter *neuron hidden* sejumlah 70 unit, dan nilai learning sebesar 0.9. Untuk mendapatkan tingkat akurasi kebenaran yang maksimal, pengujian ini akan dilakukan perbandingan tingkat akurasi kebenaran terhadap pengaruh nilai *threshold* yang berbeda-beda. Hasil pengujian dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil percobaan pengaruh *threshold* terhadap keakuratan data uji

Percobaan ke	*Threshold*	Jumlah Benar	Jumlah Salah	Keakuratan
1	0.1	28	2	93.33%
2	0.2	28	2	93.33%
3	0.3	29	1	96.66%
4	0.4	29	1	96.66%
5	0.5	29	1	96.66%
6	0.6	29	1	96.66%
7	0.7	28	2	93.33%
8	0.8	28	2	93.33%
9	0.9	28	2	93.33%

Berdasarkan hasil tabel 4.3 dapat dilihat threshold antara 0,3 sampai 0,6 mempunyai akurasi 96.66%. Tetapi ketika nilai threshold meningkat tingkat akurasi nya menjadi berkurang.

Setelah melakukan pengujian threshold kita lakukan pengujian terhadap pengaruh pada jumlah data latih. Dengan menggunakan arsitektur JST optimal yang didapatkan dari pelatihan yang sudah dilakukan yaitu dengan menggunakan parameter *neuron hidden* sejumlah 70 unit dan nilai *learning rate* sebesar 0.9. Pengujian dilakukan dengan data latih yang berbeda-beda. Data latih yang digunakan mulai dari 40, 50, 60 dan data 70. Satu kali percobaan yaitu dengan menguji data yang sama dengan data uji sebanyak 30. Dalam pembagian data yang digunakan untuk pelatihan dan pengujian data sebanyak 100 dibagi menjadi 2, yaitu 70 untuk data latih dan 30 untuk data uji. Dan untuk variasi data latih nya dikurangi setiap 10 data dari 70 data latih. Data uji yang digunakan adalah 30 data yang belum pernah dilatih sebelumnya. Hasil pengujian pengaruh jumlah data latih terhadap akurasi sistem terlihat pada tabel 4.4.

Tabel 4.4 Hasil Percobaan pengaruh jumlah data latih terhadap akurasi sistem

Jumlah Data Latih	Jumlah Data Uji	Jumlah Benar	Jumlah Salah	Akurasi
10	30	29	1	96.66%
20	30	29	1	96.66%
30	30	29	1	96.66%
40	30	29	1	96.66%
50	30	29	1	96.66%
60	30	29	1	96.66%
70	30	29	1	96.66%

Berdasarkan tabel 4.4 dapat dilihat bahwa akurasi pada tiap data latih yang berbeda tidak berubah. Dengan jumlah benar sebanyak 29 dan jumlah salah 1 sehingga mendapatkan akurasi sebesar 96.66%.

4.4.4 Analisa Hasil

Jaringan Saraf Tiruan *Resilient propagation* dengan tiga lapisan neuron, yaitu satu lapisan masukan, satu lapisan tersembunyi dan satu lapisan keluaran mampu memberikan tingkat keberhasilan yang cukup bagus dalam memberikan prediksi diabetes. Memperbanyak jumlah neuron pada lapisan tersembunyi, dan menambah nilai *learning rate* diharapkan mampu meningkatkan ketelitian jaringan (diperoleh MSE yang konvergen), akan tetapi tingkat ketelitian juga dipengaruhi oleh faktor lain yaitu dan nilai bobot awal yang diperoleh secara acak.

Bedasarkan pada tabel dan grafik percobaan *neuron* pada *hidden layer* bahwa peningkatan neuron tidak berbanding lurus pada peningkatan atau penurunan MSE. Nilai error mencapai titik kandidat minimum saat mendapatkan neuron sejumlah 10, 40, dan 70 neuron. Penambahan jumlah neuron tidak disarankan dikarenakan akan membuat nilai MSE menjadi semakin besar.

Bedasarkan pada tabel dan grafik percobaan *learning rate* bahwa peningkatan nilai *learning rate* berbanding terbalik terhadap MSE. Besarnya peningkatan nilai learning rate akan menghasilkan MSE yang kecil. Penggunaan

nilai learning rate antara 0.4 sampai 0.9 digunakan untuk mencapai nilai MSE kurang dari 0.000269361570482499. Penyebab nilai MSE menjadi semakin kecil dikarenakan semakin besar nilai pembelajaran (*learning rate*) maka nilai eror menjadi semakin sedikit.

Bedasarkan pada tabel percobaan pengaruh threshold nilai akurasi yang diperoleh tidak dipengaruhi oleh nilai threshold, namun terdapat kecenderungan akurasi akan semakin baik pada range threshold bagian tengah seperti grafik sinus. Hal ini dapat dilihat bahwa nilai threshold antara 0.3 sampai 0.6 memiliki nilai akurasi yang tinggi sebesar 96.66% dibandingkan lainnya yang hanya sebesar 93.33%. penyebab nilai akurasi tertinggi antara nilai 0.3 sampai 0.6 karena pada range tersebut merupakan titik tengah nilai antara nilai threshold 0.1 sampai 0.9 sehingga pembagian nilai ambang menjadi baik.

Bedasarkan pada tabel percobaan data latih dapat diketahui bahwa besarnya kurasi tidak dipengaruhi oleh variasi dari data uji dan data latih. Hal ini dapat ditunjukkan bahwa data latih sebesar 10 sampai 70 dengan data uji sebesar 30 diperoleh nilai akurasi sebesar 96.66%. Hasil pengenalan sistem dalam memprediksi diabetes yang terbesar adalah sebesar 96.66% didapat dari jumlah prediksi yang benar 29 dan jumlah prediksi yang salah 1. Penyebab nilai akurasi tidak ada yang berubah karena pada pelatihan telah dilakukan percobaan untuk neuron dan learning rate yang membuat pembelajaran atau pelatihan menjadi bagus. Sehingga pada data latih sedikit pun sudah didapatkan nilai akurasi yang bagus.

Dengan maksimal epoch sebesar 5000 , diperoleh MSE yang nilainya mendekati target *error* yaitu sebesar 0.0000999732406311761. Jika jumlah iterasi semakin diperbanyak, dapat mengakibatkan waktu komputasi akan semakin lama karena di setiap iterasi, jaringan melakukan proses *resilient propagation* secara berulang-ulang. Dengan menggunakan nilai MSE yang mendekati target *error* tersebut, jaringan dapat mengenali dengan baik data uji. Semakin kecil nilai MSE, maka tingkat keakuratan sistem dalam mengenali data diabetes semakin tinggi.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Metode jaringan syaraf tiruan (JST) dengan algoritma *Resilient propagation* dapat di implementasikan untuk mendiagnosa kasus diabetes dengan model jaringan yang digunakan yaitu 4 pada layer input, 70 unit neuron pada hidden layer, dan 1 unit neuron pada layer output. Kombinasi parameter yang terbaik hasil implementasi metode ini yaitu dengan nilai *learning rate* sebesar 0.9 dan nilai neuron hidden sebanyak 70 unit. Nilai error mencapai titik kandidat minimum saat mendapatkan neuron sejumlah 10, 40, dan 70 neuron.

Sistem mampu memprediksi dengan baik menggunakan nilai range threshold sebesar 0.3 sampai 0.6 dengan tingkat akurasi prediksi sebesar 96.66% dari data uji sebanyak 30 data. Pada kasus ini dengan *learning rate* 0,4 sampai 0,9 mengalami penurunan yang signifikan, sehingga semakin besar *learning rate* semakin kecil pula MSE pelatihan yang dihasilkan. Dengan maksimal epoch sebesar 5000 , diperoleh MSE yang nilainya mendekati target *error* yaitu sebesar 0.0000999732406311761.

5.2 Saran

Pada penelitian jaringan syaraf tiruan ini perlu dilakukan pembanding dengan menggunakan jumlah data yang lebih banyak dan pola data yang bervariasi. Dengan keberadaan jumlah data semakin banyak dan pola yang semakin bervariasi diharapkan sistem masih dapat mengenali dengan baik pola data uji yang berbeda dan memberikan kemampuan bagi system untuk melakukan pembelajaran yang lebih banyak sehingga hasil analisa bisa lebih terinci.

DAFTAR PUSTAKA

[FAU – 94] Fausett, L. 1994. *Fundamentals of Neural Network: Architectures, Algorithms, and Applications*. New Jersey:Prentice-Hall,Inc.

[FEB – 07] Febrianty, Devy, 2007, *Analisis Jaringan Saraf Tiruan Rprop Untuk Mengenali Pola Elektrokardiografi Dalam Mendeteksi Penyakit Jantung Koroner*, Seminar Nasional Aplikasi Teknologi Informasi.

[HAI – 05] Haidar, andry, 2005, *Studi Kasus Mengenai Aplikasi Multilayer Perceptron Neural Network Pada Sistem Pendeteksi Gangguan (IDS) Berdasarkan Anomali Suatu Jaringan*.

[HEC – 88] Hecht-Nielsend, R., 1988, *Applications of counterpropagation networks*, San Diego : Elsevier Ltd.

[KAH – 06] Kahn, C, et al. 2006. *Joslin's Diabetes Mellitus Fourteenth Edition.* Harvard Medical School. Boston, Massachusetts.

[KRI – 04] Kristanto, A., 2004, *Jaringan Syaraf Tiruan; Konsep Dasar, Algoritma dan Aplikasinya*, Yogyakarta : Gava Media.

[KUS – 03] Kusumadewi, S., 2003, *Artificial Intelligence (Teknik dan Aplikasinya)*, Yogyakarta : Graha Ilmu.

[LAU – 07] Laudon, Jane P., Kenneth C. Laudon, 2007, *Management Information System, 10th ed*, New Jersey : Pearson Education, Inc.

[PUS – 06] Puspitaningrum, Dyah, 2006, *Pengantar Jaringan Saraf Tiruan*, Yogyakarta : Andi.

[RAD – 07] Radha. R. 2007. *Fuzzy Logic Approach for Diagnosis of Diabetics*. Information Technology Journal 6 (1), Pp. 96-102. India

[RIE – 94] Riedmiller, Martin, 1994, *Rprop-Description and Implementation Details*, Jerman : University of Karlsruhe.

[SET – 10] Setia K, Yoga, 2010, Diagnosis diabetes menggunakan fuzzy database model tahani dengan inferensi fuzzy mamdani, Malang : Universitas Brawijaya.

[SIA – 05] Siang, Jong Jek, (2005), *Jaringan Syaraf Tiruan dan Pemrograman Menggunakan Matlab,1st edition*, Yogyakarta : Andi.

[SRI – 09] Srivastava, Manjita, M C Srivastava, Smriti Bathnagar, 2009, *Control Systems*, India : Tata McGraw-Hill.

LAMPIRAN 1

Data hasil tes akurasi dengan menggunakan data latih 10.

No	Respon data asli	Respon hasil prediksi	status
1	diabetes	diabetes	benar
2	diabetes	diabetes	benar
3	diabetes	diabetes	benar
4	diabetes	diabetes	benar
5	diabetes	diabetes	benar
6	diabetes	diabetes	benar
7	diabetes	diabetes	benar
8	diabetes	diabetes	benar
9	diabetes	diabetes	benar
10	diabetes	diabetes	benar
11	diabetes	diabetes	benar
12	diabetes	diabetes	benar
13	diabetes	diabetes	benar
14	diabetes	non diabetes	salah
15	diabetes	diabetes	benar
16	non diabetes	non diabetes	benar
17	non diabetes	non diabetes	benar
18	non diabetes	non diabetes	benar
19	non diabetes	non diabetes	benar
20	non diabetes	non diabetes	benar
21	non diabetes	non diabetes	benar
22	non diabetes	non diabetes	benar
23	non diabetes	non diabetes	benar
24	non diabetes	non diabetes	benar
25	non diabetes	non diabetes	benar
26	non diabetes	non diabetes	benar
27	non diabetes	non diabetes	benar
28	non diabetes	non diabetes	benar
29	non diabetes	non diabetes	benar
30	non diabetes	non diabetes	benar

LAMPIRAN 2

Data hasil tes akurasi dengan menggunakan data latih 20.

No	Respon data asli	Respon hasil prediksi	status
1	diabetes	diabetes	benar
2	diabetes	diabetes	benar
3	diabetes	diabetes	benar
4	diabetes	diabetes	benar
5	diabetes	diabetes	benar
6	diabetes	diabetes	benar
7	diabetes	diabetes	benar
8	diabetes	diabetes	benar
9	diabetes	diabetes	benar
10	diabetes	diabetes	benar
11	diabetes	diabetes	benar
12	diabetes	diabetes	benar
13	diabetes	diabetes	benar
14	diabetes	non diabetes	salah
15	diabetes	diabetes	benar
16	non diabetes	non diabetes	benar
17	non diabetes	non diabetes	benar
18	non diabetes	non diabetes	benar
19	non diabetes	non diabetes	benar
20	non diabetes	non diabetes	benar
21	non diabetes	non diabetes	benar
22	non diabetes	non diabetes	benar
23	non diabetes	non diabetes	benar
24	non diabetes	non diabetes	benar
25	non diabetes	non diabetes	benar
26	non diabetes	non diabetes	benar
27	non diabetes	non diabetes	benar
28	non diabetes	non diabetes	benar
29	non diabetes	non diabetes	benar
30	non diabetes	non diabetes	benar

LAMPIRAN 3

Data hasil tes akurasi dengan menggunakan data latih 30.

No	Respon data asli	Respon hasil prediksi	status
1	diabetes	diabetes	benar
2	diabetes	diabetes	benar
3	diabetes	diabetes	benar
4	diabetes	diabetes	benar
5	diabetes	diabetes	benar
6	diabetes	diabetes	benar
7	diabetes	diabetes	benar
8	diabetes	diabetes	benar
9	diabetes	diabetes	benar
10	diabetes	diabetes	benar
11	diabetes	diabetes	benar
12	diabetes	diabetes	benar
13	diabetes	diabetes	benar
14	diabetes	non diabetes	salah
15	diabetes	diabetes	benar
16	non diabetes	non diabetes	benar
17	non diabetes	non diabetes	benar
18	non diabetes	non diabetes	benar
19	non diabetes	non diabetes	benar
20	non diabetes	non diabetes	benar
21	non diabetes	non diabetes	benar
22	non diabetes	non diabetes	benar
23	non diabetes	non diabetes	benar
24	non diabetes	non diabetes	benar
25	non diabetes	non diabetes	benar
26	non diabetes	non diabetes	benar
27	non diabetes	non diabetes	benar
28	non diabetes	non diabetes	benar
29	non diabetes	non diabetes	benar
30	non diabetes	non diabetes	benar

LAMPIRAN 4

Data hasil tes akurasi dengan menggunakan data latih 40.

No	Respon data asli	Respon hasil prediksi	status
1	diabetes	diabetes	benar
2	diabetes	diabetes	benar
3	diabetes	diabetes	benar
4	diabetes	diabetes	benar
5	diabetes	diabetes	benar
6	diabetes	diabetes	benar
7	diabetes	diabetes	benar
8	diabetes	diabetes	benar
9	diabetes	diabetes	benar
10	diabetes	diabetes	benar
11	diabetes	diabetes	benar
12	diabetes	diabetes	benar
13	diabetes	diabetes	benar
14	diabetes	non diabetes	salah
15	diabetes	diabetes	benar
16	non diabetes	non diabetes	benar
17	non diabetes	non diabetes	benar
18	non diabetes	non diabetes	benar
19	non diabetes	non diabetes	benar
20	non diabetes	non diabetes	benar
21	non diabetes	non diabetes	benar
22	non diabetes	non diabetes	benar
23	non diabetes	non diabetes	benar
24	non diabetes	non diabetes	benar
25	non diabetes	non diabetes	benar
26	non diabetes	non diabetes	benar
27	non diabetes	non diabetes	benar
28	non diabetes	non diabetes	benar
29	non diabetes	non diabetes	benar
30	non diabetes	non diabetes	benar

LAMPIRAN 5

Data hasil tes akurasi dengan menggunakan data latih 50.

No	Respon data asli	Respon hasil prediksi	status
1	diabetes	diabetes	benar
2	diabetes	diabetes	benar
3	diabetes	diabetes	benar
4	diabetes	diabetes	benar
5	diabetes	diabetes	benar
6	diabetes	diabetes	benar
7	diabetes	diabetes	benar
8	diabetes	diabetes	benar
9	diabetes	diabetes	benar
10	diabetes	diabetes	benar
11	diabetes	diabetes	benar
12	diabetes	diabetes	benar
13	diabetes	diabetes	benar
14	diabetes	non diabetes	salah
15	diabetes	diabetes	benar
16	non diabetes	non diabetes	benar
17	non diabetes	non diabetes	benar
18	non diabetes	non diabetes	benar
19	non diabetes	non diabetes	benar
20	non diabetes	non diabetes	benar
21	non diabetes	non diabetes	benar
22	non diabetes	non diabetes	benar
23	non diabetes	non diabetes	benar
24	non diabetes	non diabetes	benar
25	non diabetes	non diabetes	benar
26	non diabetes	non diabetes	benar
27	non diabetes	non diabetes	benar
28	non diabetes	non diabetes	benar
29	non diabetes	non diabetes	benar
30	non diabetes	non diabetes	benar

LAMPIRAN 6

Data hasil tes akurasi dengan menggunakan data latih 60.

No	Respon data asli	Respon hasil prediksi	status
1	diabetes	diabetes	benar
2	diabetes	diabetes	benar
3	diabetes	diabetes	benar
4	diabetes	diabetes	benar
5	diabetes	diabetes	benar
6	diabetes	diabetes	benar
7	diabetes	diabetes	benar
8	diabetes	diabetes	benar
9	diabetes	diabetes	benar
10	diabetes	diabetes	benar
11	diabetes	diabetes	benar
12	diabetes	diabetes	benar
13	diabetes	diabetes	benar
14	diabetes	non diabetes	salah
15	diabetes	diabetes	benar
16	non diabetes	non diabetes	benar
17	non diabetes	non diabetes	benar
18	non diabetes	non diabetes	benar
19	non diabetes	non diabetes	benar
20	non diabetes	non diabetes	benar
21	non diabetes	non diabetes	benar
22	non diabetes	non diabetes	benar
23	non diabetes	non diabetes	benar
24	non diabetes	non diabetes	benar
25	non diabetes	non diabetes	benar
26	non diabetes	non diabetes	benar
27	non diabetes	non diabetes	benar
28	non diabetes	non diabetes	benar
29	non diabetes	non diabetes	benar
30	non diabetes	non diabetes	benar

LAMPIRAN 7

Data hasil tes akurasi dengan menggunakan data latih 70.

No	Respon data asli	Respon hasil prediksi	status
1	diabetes	diabetes	benar
2	diabetes	diabetes	benar
3	diabetes	diabetes	benar
4	diabetes	diabetes	benar
5	diabetes	diabetes	benar
6	diabetes	diabetes	benar
7	diabetes	diabetes	benar
8	diabetes	diabetes	benar
9	diabetes	diabetes	benar
10	diabetes	diabetes	benar
11	diabetes	diabetes	benar
12	diabetes	diabetes	benar
13	diabetes	diabetes	benar
14	diabetes	non diabetes	salah
15	diabetes	diabetes	benar
16	non diabetes	non diabetes	benar
17	non diabetes	non diabetes	benar
18	non diabetes	non diabetes	benar
19	non diabetes	non diabetes	benar
20	non diabetes	non diabetes	benar
21	non diabetes	non diabetes	benar
22	non diabetes	non diabetes	benar
23	non diabetes	non diabetes	benar
24	non diabetes	non diabetes	benar
25	non diabetes	non diabetes	benar
26	non diabetes	non diabetes	benar
27	non diabetes	non diabetes	benar
28	non diabetes	non diabetes	benar
29	non diabetes	non diabetes	benar
30	non diabetes	non diabetes	benar