SKRIPSI

RANCANG BANGUN APLIKASI JARINGAN SARAF TIRUAN UNTUK DIAGNOSA PENYAKIT PADA SPECIES BURUNG PUNGLOR BERBASIS WEB

Disusun Oleh

BAGUS SANTOSO
07. 12. 630

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

2012

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI JARINGAN SARAF TIRUAN UNTUK DIAGNOSA PENYAKIT PADA SPECIES BURUNG PUNGLOR BERBASIS WEB

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Komputer dan Informatika Strata Satu (S-1)

Disusun oleh :

BAGUS SANTOSO

07.12.630

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
M. Ibrahim Ashari, ST, MT	**Sotyohadi, ST**
NIP. P.1030100358	**NIP.Y. 1039700309**

Mengetahui,

Ketua Jurusan Teknik Elektro S-1

Ir. Yusuf Ismail Nakhoda, MT

NIP.Y.1018800189

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2012

ABSTRAK

RANCANG BANGUN APLIKASI JARINGAN SARAF TIRUAN UNTUK DIAGNOSA PENYAKIT PADA SPECIES BURUNG PUNGLOR BERBASIS WEB

Bagus Santoso, NIM 07.12.630
Dosen Pembimbing : M. Ibrahim Ashari, ST, MT dan Sotyohadi, ST

Semua species burung punglor terkenal mudah terserang penyakit, penyakit tersebut dapat diketahui dari gejala-gejala yang ditimbulkan. Sehingga dibutuhkan suatu aplikasi perangkat lunak untuk mengidentifikasi jenis penyakit pada burung punglor dan memberikan solusi kepada *user* cara menangani penyakit tersebut.

Aplikasi ini dibangun menggunakan algoritma jaringan saraf tiruan *backpropagation* yang sudah dilatih mampu memberikan diagnosa penyakit dengan cukup baik. Serta untuk membantu *user* mendiagnosa penyakit pada punglor peliharaanya, sistem juga dibangun berbasis *web* agar dapat diakses dengan mudah.

Aplikasi ini dapat mengidentifikasi 9 penyakit dengan 22 jenis gejala pada burung punglor. Dan terdapat beberapa informasi mengenai burung punglor yang dapat menambah pengetahuan pengguna aplikasi.

Kata Kunci: jaringan saraf tiruan, backpropagation, web, penyakit burung punglor

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa, yang telah memberikan berkat-Nya, sehingga penulis dapat menyelesaikan laporan Skripsi ini dengan baik dan lancar.

Laporan Skripsi ini merupakan salah satu persyaratan akademik dalam menyelesaikan program Strata 1 Jurusan Teknik Elektro, Konsentrasi Komputer & Informatika, Institut Teknologi Nasional Malang. Adapun judul laporan Skripsi ini adalah:

RANCANG BANGUN APLIKASI JARINGAN SARAF TIRUAN UNTUK DIAGNOSA PENYAKIT PADA SPECIES BURUNG PUNGLOR BERBASIS WEB

Selanjutnya pada kesempatan ini penulis juga menyampaikan rasa terimakasih yang sebesar-besarnya kepada pihak–pihak yang telah banyak membantu penulis selama penyusunan tugas akhir, diantaranya :

1. Bapak Ir. Yusuf Ismail Nakhoda, MT selaku Ketua Jurusan Teknik Elektro S-1 ITN Malang.
2. Bapak Dr. Aryuanto Soetedjo, ST, MT selaku Sekertaris Jurusan Teknik Elektro S-1 ITN Malang.
3. Bapak M. Ibrahim Ashari, ST, MT selaku Dosen Pembimbing I
4. Bapak Sotyohadi, ST selaku Dosen Pembimbing II
5. Bapak Ir. Yusuf Ismail Nakhoda, MT, selaku Dosen Wali.
6. Orang tua serta keluarga yang telah memberikan dukungan untuk selalu berdoa, berusaha dan nasehat yang telah diberikan sampai saat ini.
7. Seluruh dosen dan pegawai ITN Kampus 2 Malang.
8. Semua teman-teman Mahasiswa Elektro ITN Malang.
9. Semua pihak yang telah membantu penulis dalam menyelesaikan skripsi ini yang tidak bisa penulis sebutkan satu persatu.

Penulis berharap agar buku laporan Skripsi ini dapat memberikan banyak manfaat bagi semua pihak yang membutuhkan, khususnya bagi rekan-rekan mahasiswa. Penulis menyadari bahwa dalam penyusunan laporan ini masih

banyak kekurangan, oleh karena itu mohon maaf apabila dalam buku ini terdapat hal-hal yang kurang berkenan dihati para pembaca.

Penulis juga mengharap koreksi, kritik serta saran-saran yang bermanfaat demi kesempurnaan buku Laporan Skripsi ini.

Malang, Agustus 2012

Penulis

DAFTAR ISI

DAFTAR TABEL

BAB III PERANCANGAN SISTEM

DAFTAR GAMBAR

BAB II TINJAUAN PUSTAKA

BAB III PERANCANGAN SISTEM

BAB IV IMPLEMENTASI DAN PEMBAHASAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Hobi memelihara dan merawat burung berkicau sudah sejak lama digemari oleh masyarakat Indonesia. Banyak alasan kenapa burung khususnya dari species punglor menjadi burung peliharaan yang paling diminati. Beberapa alasannya adalah sekedar untuk mendengar kicauannya, usaha *breeding* burung, bisnis burung, melatih burung, dll. Selain itu popularitas burung punglor cenderung stabil karena meliki ciri khas yang unik saat berkicau.

Ada satu faktor yang menghambat perkembangan dari species punglor yaitu virus dan penyakit. Hal tersebut dapat disebabkan oleh beberapa faktor yaitu pola makan yang tidak tepat, kebersihan kandang yang tidak higienis, faktor lingkungan, atau bahkan disebabkan oleh virus.

Sebenarnya tiap penyakit pada punglor sebelum menjadi parah umumnya menunjukkan gejala-gejala penyakit yang diderita, tetapi masih dalam tahap yang ringan. Namun pemilik sering mengabaikan hal ini karena tidak mengetahuinya, dan menganggap gejala tersebut sudah biasa terjadi, sampai pada suatu saat timbul gejala yang sangat parah, sehingga sudah terlambat untuk ditangani.

Para ahli dalam hal ini mempunyai kemampuan untuk menganalisa gejala-gejala penyakit pada burung tersebut, tetapi untuk mengatasi semua persoalan yang dihadapi pemilik atau peternak terkendala oleh waktu dan banyaknya pemilik yang mempunyai masalah dengan peliharaannya. Oleh karena itu, pada penelitian ini akan dibuat suatu aplikasi jaringan saraf tiruan yang memberikan informasi mengenai penyakit pada burung dan dapat mendiagnosa gejala-gejala penyakit pada burung berkicau, khususnya pada species punglor, sekaligus memberikan solusi penanggulangannya, yang nantinya dapat mengurangi atau memperkecil risiko kerugian bagi pemilik atau sampai menimbulkan kematian. Implementasi jaringan saraf tiruan ini dibuat berbasis *Web* agar dapat diakses dan dimanfaatkan masyarakat secara luas.

Pada penelitian ini digunakan metode *Backpropagation*, sedangkan persoalan yang dibahas mengenai penyakit pada species punglor dan gejala-gejala yang menyertainya. Jaringan saraf tiruan ini dibuat berbasis *Web* dengan menggunakan PHP dan basis data Mysql yang sangat ringan dan mudah diakses. Hasil penelitian ini diharapkan dapat dimanfaatkan oleh pemilik untuk mendiagnosa penyakit pada *variant* punglornya, sehingga pemilik tidak harus menunggu kehadiran seorang pakar hewan untuk mendiagnosa penyakit pada burung.

Dalam skripsi ini akan diterapkan algoritma pembelajaran *Backpropagation* untuk memprediksi penyakit pada burung punglor. *Backpropagation* merupakan suatu metode untuk melakukan pelatihan terhadap lapisan-lapisan kompetitif yang terawasi, dimana diperlukan data pelatihan dalam prosesnya. Jaringan dalam *Backpropagation* bersifat *multilayer*. Jadi, minimal terdapat sebuah *input layer, hidden layer,* dan *output layer.*

Penggunaan algoritma *Backpropagation* dalam sistem ini ditujukan untuk meminimalkan error pada keluaran yang dihasilkan jaringan dengan targetnya. Apabila keluaran jaringan memberikan hasil yang salah, maka penimbang (*weight*) dikoreksi agar error dapat diperkecil sehingga tanggapan jarigan saraf tiruan selanjutnya diharapkan dapat mendekati nilai yang benar.

1.2 Rumusan Masalah

Dalam skripsi ini, ruang lingkup permasalahan dibatasi pada bagaimana membuat program aplikasi dalam bentuk perangkat lunak (*software*) yang dipergunakan untuk mengidentifikasi penyakit pada burung punglor beserta solusi penanganannya dalam usaha memberikan masukan bagi pemilik maupun pihak-pihak yang terkait.

Karena permasalahan penyakit pada burung punglor cukup kompleks maka permasalahan pada skripsi ini dapat dirumuskan sebagai berikut :

1. Bagaimana mengidentifikasi penyakit pada burung punglor serta penanggulangannya?
2. Bagaimana membangun perangkat lunak program aplikasi dalam menanggulangi penyakit pada burung punglor?

1.3 Batasan Masalah

Permasalahan yang akan dibahas dalam penulisan ini dibatasi pada:

1. Aplikasi dibuat dengan jaringan saraf tiruan menggunakan metode *Backpropagation.*
2. Program aplikasi yang dibuat hanya berfungsi untuk mengidentifikasi penyakit pada burung punglor setelah terjadi gejala-gejala.
3. Solusi yang dihasilkan hanya berupa cara penanganan setelah penyakit teridentifikasi.
4. Aplikasi ini dirancang hanya untuk species burung punglor.

1.4 Tujuan Penelitian

Tujuan penulisan skripsi ini adalah :

1. Mengidentifikasi pada burung punglor dengan menggunakan aplikasi jaringan saraf tiruan, beserta solusi penanggulangannya.
2. Membuat program aplikasi sebagai pengganti pakar dengan mensubtitusikan pengetahuan manusia ke dalam bentuk sistem sehingga dapat dipakai orang banyak untuk menangani pada burung punglor

1.5 Manfaat Penelitian

Manfaat yang dapat diambil dari pelaksanaan penelitian ini adalah:

1. Bagi para Ahli peternakan dan Masyarakat
    a. Membantu tugas para ahli peternakan untuk memudahkan pengenalan jenis penyakit serta tepat dalam mengambil tindakan.
    b. Membantu masyarakat, khususnya pemilik untuk mengetahui penyakit yang menyerang burung punglor, sehingga upaya pengobatan penyakit dapat secara cepat dan tepat.
    c. Sistem yang dibuat diharapkan dapat membantu meringankan beban tenaga dan meningkatkan produktifitas kerja ahli peternakan.

2. Bagi Penulis
    a. Untuk menambah wawasan dan ilmu pengetahuan di luar lingkungan kampus yang berhubungan dengan program studi yang dipilih.
    b. Untuk menambah pengetahuan sebelum terjun langsung ke dunia kerja dan ke tengah masyarakat untuk mengabdikan ilmu yang dipelajari di bangku kuliah.
3. Bagi Ilmu Pengetahuan
    a. Hasil penelitian ini diharapkan dapat memberikan sumbangan bagi pengembangan ilmu di bidang informatika.
    b. Menggali solusi alternatif dan menambah kepustakaan di bidang ilmu komputer.
    c. Menambah kepustakaan di bidang ilmu peternakan.

1.6 Metodologi Penelitian

1. Studi literatur dan survei

    Pengumpulan data yang dilakukan dengan mencari bahan-bahan kepustakaan,referensi, dan survei lapangan dari berbagai sumber sebagai landasan teori yang ada hubunganya dengan permasalahan yang dijadikan objek penelitian.

2. Analisa Kebutuhan Sistem

    Data dan informasi yang telah diperoleh akan dianalisa agar didapatkan kerangka global yang bertujuan untuk mendefinisikan kebutuhan sistem di mana nantinya akan digunakan sebagai acuan perancangan sistem.

3. Perancangan dan Implementasi

    Berdasarkan data dan informasi yang telah diperoleh serta analisa kebutuhan untuk membangun sistem ini, akan dibuat rancangan kerangka global yang menggambarkan mekanisme dari sistem yang akan dibuat dan diimplementasikan kedalam sistem.

4. Eksperimen dan Evaluasi

   Pada tahap ini, sistem yang telah selesai dibuat akan diuji coba, yaitu pengujian berdasarkan fungsionalitas program, dan akan dilakukan koreksi dan penyempurnaan program jika diperlukan.

1.7 Sistematika Penulisan

Untuk mempermudah dan memahami pembahasan penulisan skripsi ini, maka sistematika penulisan disusun sebagai berikut :

Bab I : Pendahuluan

Berisi Latar Belakang, Rumusan Masalah, Tujuan Penelitian, Pembatasan Permasalahan, Metode Penelitian dan Sistematika Penulisan.

Bab II : Tinjauan Pustaka

Berisi tentang landasan teori mengenai permasalahan yang berhubungan dengan penelitian yang dilakukan.

Bab III : Perancangan dan Analisa Sistem

Dalam bab ini berisi mengenai analisa kebutuhan sistem baik *software* maupun *hardware* yang diperlukan untuk membuat kerangka global yang menggambarkan mekanisme dari sistem yang akan dibuat.

Bab IV : Pembuatan dan Pengujian Sistem

Berisi tentang implementasi dari perancangan sistem yang telah dibuat serta pengujian terhadap sistem tersebut.

Bab V : Penutup

Merupakan bab terakhir yang memuat intisari dari hasil pembahasan yang berisikan kesimpulan dan saran yang dapat digunakan sebagai pertimbangan untuk pengembangan penulisan selanjutnya.

BAB II

TINJAUAN PUSTAKA

2.1 Punglor

2.1.1 Definisi

Punglor atau biasa disebut anis adalah salah satu species burung yang hidup di berbagai Negara di asia seperti Bangladesh, Bhutan, Cambodia, China, India, Laos, Malaysia, Myanmar, Nepal, Pakistan, SriLanka, Thailand, dan Vietnam. Punglor jadi primadona pecinta burung karena memiliki ciri khas yang menarik minat pecinta burung. Setiap punglor di suatu Negara akan berbeda karakteristiknya karena dipengaruhi oleh suhu dan iklim di Negara tersebut. Punglor memiliki beberapa jenis, yaitu:

1. Anis Merah (*Zoothera citrina*) yaitu punglor yang paling banyak diminati untuk dikembangkan. Selain karena suaranya yang mendayu-dayu, gaya ocehannya bisa membuat orang takjub. Kepalanya bergerak kekiri dan kekanan alias bergaya teler. Berwarna merah bata dan hitam kebiru-biruan serta warna putih di bagian sayap.
2. Anis Kembang (*Zoothera interpres*) yaitu punglor yang memiliki ciri warna putih totol hitam di bagian dada, serta hitam pekat pada sayap dan warna merah di bagian kepala. Habitat aslinya berada di hutan primer di daerah Jawa Barat, Kalimantan, NTB, Cina dan India.
3. Anis Siberia (*Zoothera sibirica*) yaitu punglor yang memiliki ciri warna biru kehitaman dan memiliki ciri khas alis putih di dekat mata. Anis ini berasal dari Cina dan bermigrasi saat musim dingin. Biasa ditemukan di daerah pegunungan Jawa, Sumatra, dan sebagian Kalimantan.
4. Anis Sisik (*Zoothera dauma*) yaitu memilliki ciri warna menyerupai sisik dan memiliki suara lembut, pendek dan mengalun. Biasa ditemukan di pegunungan Sumatra.
5. Anis Kuning (*Zoothera obscurus*) yaitu burung anis yang memiliki warna sesuai namanya yaitu kuning. Bersuara berisik saat berkelompok dan

bersuara lemah pendek saat menyendiri. Biasa ditemukan di hutan sekunder Siberia hingga Mongolia.

6. Anis Hibrida yaitu hasil perkawinan antara anis merah dan anis kembang yang dilakukan pecinta anis di Indonesia. Di pasaran biasa disebut anis blorok atau anis macan.

Klasifikasi punglor adalah sebagai berikut:

Kingdom : *Animalia*
Phylum : *Chordata*
Classis : *Aves*
Subclassis : *Carinatae*
Ordo : *Passeriformes*
Subordo : *Passeri*
Pavordo : *Passerida*
Superfamilia : *Muscicapoidea*
Familia : *Turdidae*
Genus : *Zoothera*

2.1.2 Jenis-jenis Penyakit pada Punglor

Mendeteksi penyakit pada punglor tidaklah mudah karena burung ini tidak selalu boleh dipegang dan melihatnya pun dari jarak yang tidak dekat, selain itu sama seperti burung yang lain punglor tidak memberikan isyarat atau gejala-gejala yang jelas dan setelah sakit baru nampak perubahannya. Penyakit bisa disebakan karena kondisi sekitar dan virus. Adapun jenis-jenis penyakit yang dapat menyerang punglor, yaitu:

1. Abes (Bengkak), yaitu penyakit pada kaki yang terluka akibat terkena bagian sangkar yang tajam dan menyebabkan infeksi. Gejala-gejalanya adalah:
    a) Kaki terlihat bengkak
    b) Burung tampak lesu
2. Berak Darah, yaitu penyakit yang menyerang usus hingga menyebabkan peradangan dan luka pada usus. Gejala-gejalanya adalah:
    a) Burung tampak lesu

b) Sayap terkulai dan jatuh kebawah
c) Kotoran berwarna kemerahan karena darah
d) Burung tidak mau bertengger dan berdiri di lantai sangkar

3. Berak Kapur, yaitu penyakit yang disebabkan kurangnya kebersihan tempat makan dan minum sehingga pakan dan minuman tercampur kuman dan bakteri. Gejala-gejalanya adalah:
   a) Burung tampak lesu
   b) Warna bulu menjadi kusam
   c) Nafsu makan menurun
   d) Kotoran menjadi encer dan berwarna putih
   e) Burung sering mengantuk dan kedinginan
4. Cacingan, yaitu penyakit yang disebabkan kotoran yang jarang dibersihkan yang mengundang kerumunan lalat. Gejala-gejalanya adalah:
   a) Burung tampak lesu
   b) Nafsu makan menurun
   c) Kotoran menjadi encer dan berlendir
   d) Burung menjadi lumpuh
5. *Bronchitis*, yaitu penyakit pada pernapasan. Pada burung penyakit ini disebabkan karena angin yang masuk pada sangkar terlalu berlebihan. Gejala-gejalanya adalah:
   a) Burung tampak lesu
   b) Batuk
   c) Sesak nafas
   d) Sewaktu bernafas berbunyi d tenggorokan
   e) Nafsu makan menurun
6. Susah Buang Kotoran, penyakit ini disebabkan karena sampah-sampah seperti kaki dan sayap serangga, biji buah yang keras yang tercampur pada pakan sehingga mengganggu system pencernaan burung ini. Gejala-gejalanya adalah:
   a) Burung tampak lesu
   b) Nafsu makan menurun
   c) Ada sisa kotoran pada anus burung

7. *Newcastle Deases* (ND), yaitu penyakit yang disebabkan oleh virus *Tutorfurens.* Penyakit ini menyebabkan burung tidak lagi mampu mengoceh atau berbunyi. Gejala-gejalanya adalah:
    a) Sesak nafas
    b) Nafsu makan menurun
    c) Kotoran berwarna hijau kekuningan
8. Masuk Angin, yaitu penyakit yang disebabkan suhu di sekitar sangkar yang terlalu dingin. Gejala-gejalanya adalah:
    a) Burung tampak lesu
    b) Bulu kusut dan tidak rapi
    c) Burung tidak mau makan sama sekali
    d) Kotoran menjadi berair
9. Radang Mata, penyakit ini disebabkan kotoran berupa debu, pasir atau serbuk yang mengenai mata burung sehingga menyebabkan luka. Gejala-gejalanya adalah:
    a) Mata membengkak dan berair
    b) Kelopak mata melekat dan susah terbuka
    c) Burung tampak lesu
    d) Burung tampak kebingungan

2.2 Jaringan Saraf Tiruan

2.2.1 Definisi

Jaringan Saraf Tiruan (JST) merupakan representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan saraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran.

Jaringan saraf tiruan disusun sama seperti jaringan pada saraf biologis, yaitu:

1. Pengolahan informasi terjadi pada elemen-elemen pemrosesan (*neuron-neuron*).
2. Sinyal antara dua buah *neuron* diteruskan melalui perantara koneksi.

3. Setiap perantara koneksi memiliki bobot terasosiasi.
4. Setiap *neuron* menerapkan sebuah fungsi aktivasi terhadap masukan jaringan (jumlah sinyal input berbobot). Tujuannya adalah untuk menentukan sinyal keluaran. Fungsi aktivasi yang digunakan biasanya adalah fungsi yang nonlinier.

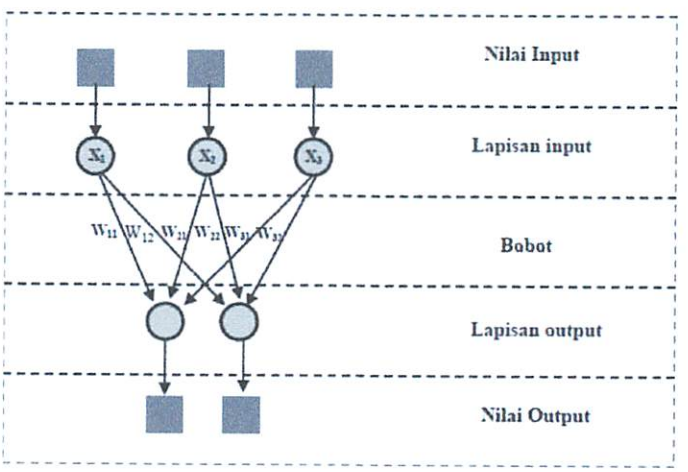
Adapun cara belajar jaringan saraf tiruan adalah, ke dalam jaringan saraf tiruan dimasukkan informasi yang sebelumnya telah diketahui hasil keluarannya. Masukan dilakukan lewat *node* (simpul) atau unit masukan. Bobot-bobot antar koneksi dalam suatu arsitektur diberi nilai awal, lalu jaringan saraf tiruan dijalankan. Bobot-bobot akan digunakan untuk proses belajar dan mengingat informasi. Pengaturan bobot dilakukan secara terus-menerus dengan menggunakan aturan tertentu sampai diperoleh keluaran yang diharapkan.

2.2.2 Arsitektur Jaringan Saraf Tiruan

Hubungan antar *neuron* dalam jaringan saraf mengikuti pola tertentu pada arsitektur jaringan sarafnya. Arsitektur jaringan saraf dibagi menjadi tiga, yaitu:

1. Jaringan saraf dengan lapisan tunggal (*single layer net*)

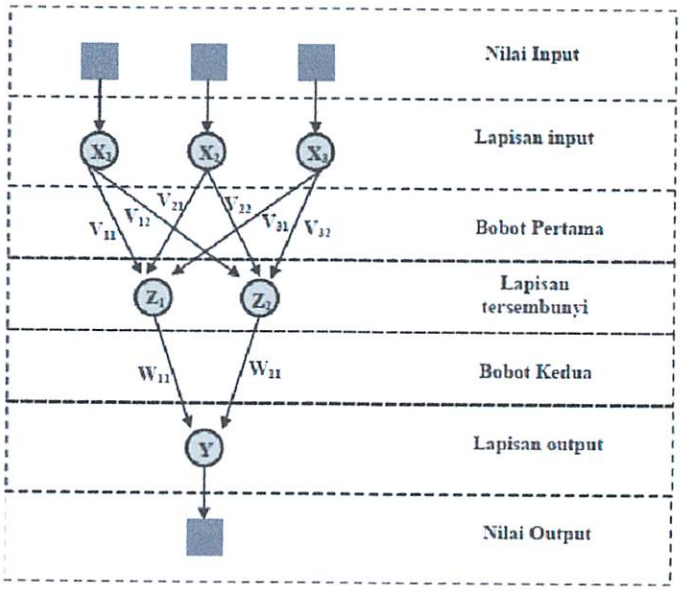
   Jaringan dengan lapisan tunggal hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima *input*, kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi. Dalam gambar 2.1 menunjukkan arsitektur jaringan saraf dengan lapisan tunggal.



Gambar 2.1 Jaringan Saraf dengan Lapisan Tunggal

2. Jaringan saraf dengan banyak lapisan (*multilayer net*)

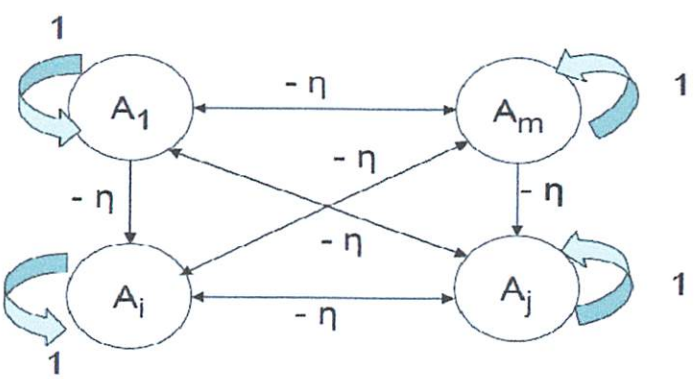
   Jaringan dengan banyak lapisan memiliki satu atau lebih lapisan yang terletak diantara lapisan *input* dan lapisan output (memiliki satu atau lebih lapisan tersembunyi). Jaringan saraf dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit, dengan pembelajaran yang lebih rumit pula. Dalam gambar 2.2 menunjukkan arsitektur jaringan saraf dengan banyak lapisan.



Gambar 2.2 Jaringan Saraf dengan Banyak Lapisan

3. Jaringan saraf dengan lapisan kompetitif (*competitive layer net*)

   Hubungan antar *neuron* pada lapisan kompetitif ini tidak diperlihatkan pada diagram arsitektur. Dalam gambar 2.3 menunjukkan contoh arsitektur jaringan dengan lapisan kompetitif yang memiliki bobot $-\eta$.



Gambar 2.3 Jaringan Saraf dengan Lapisan Kompetitif

2.2.3 Proses Pembelajaran

Jaringan saraf tiruan akan mencoba untuk mensimulasikan kemampuan otak manusia untuk belajar. Perubahan yang terjadi selama proses pembelajaran adalah perubahan nilai bobot. Nilai bobot akan bertambah, jika informasi yang diberikan oleh *neuron* tersampaikan, sebaliknya jika informasi tidak disampaikan oleh suatu *neuron* ke *neuron* lain, maka nilai bobot yang menghubungkan keduanya akan dikurangi. Pembelajaran dilakukan pada masukan yang berbeda, maka nilai bobot akan diubah secara dinamis hingga mencapai suatu nilai yang seimbang. Terdapat dua metode pembelajaran, yaitu:

1. Pembelajaran terawasi (*supervised learning*)

   Metode pembelajaran pada jaringan saraf tiruan disebut terawasi jika keluaran yang diharapkan telah diketahui sebelumnya.

2. Pembelajaran tak terawasi (*unsupervised learning*)

   Metode ini tidak memerlukan target keluaran. Selain itu tidak dapat ditentukan hasil yang diharapkan selama proses pembelajaran[1].

Dalam proses pembelajaran, *learning rate* adalah sebuah variabel yang nilainya diantara 0 sampai 1, yang berguna untuk menentukan laju pembelajaran. Sedangkan *max epoch* adalah batas maksimum perulangan untuk setiap pola pelatihan.

2.3 *Backpropagation* (Propagasi balik)

Backpropagation merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan *neuron-neuron* yang ada pada lapisan tersembunyi. Algoritma *Backpropagation* menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error* ini, tahap perambatan maju (*feedforward*) harus dikerjakan terlebih dahulu. Terdapat parameter-parameter penentu dalam pembelajaran Jaringan Saraf Tiruan, yaitu:

1. *Maximum epoch*

   Menentukan berapa iterasi pembelajaran yang akan dilakukan pada Jaringan Saraf Tiruan. Semakin besar *maximum epoch*, maka tingkat kesalahan Jaringan Saraf Tiruan akan semakin menurun. Namun,

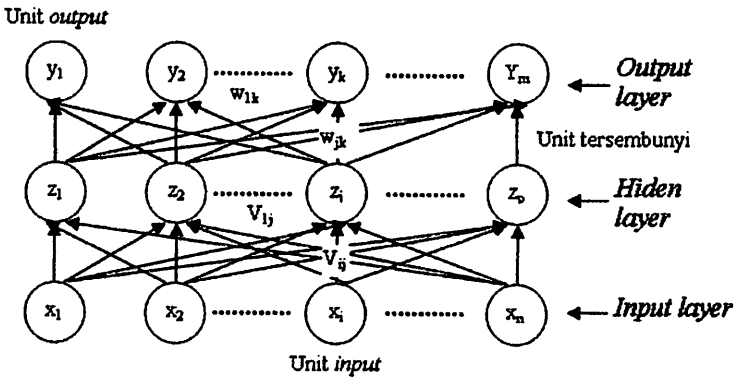
penentuan *maximum epoch* yang terlalu besar akan menyebabkan Jaringan Saraf Tiruan terlalu mengikuti pola data pelatihan dan meningkatkan kesalahan yang mungkin terjadi ketika Jaringan Saraf Tiruan diberikan data masukan dari permasalahan sebenarnya. Hal ini disebut *overfitting*.

2. Laju pembelajaran (α)

   Laju pembelajaran menunjukkan seberapa cepat Jaringan Saraf Tiruan akan menyesuaikan diri dengan data pelatihan yang diterimanya. *Learning rate* Merupakan salah satu parameter pelatihan untuk menghitung nilai koreksi bobot pada waktu proses pelatihan. Nilai α ini berada pada range 0 (nol) sampai 1 (satu). Semakin besar nilai dari *learning rate*, maka proses pelatihan akan berjalan semakin cepat. Namun, apabila nilai *learning rate* terlalu besar, pada umumnya proses pelatihan dapat melampaui keadaan optimal, yaitu pada saat dicapai nilai *error* yang paling minimal. Dengan kata lain, *learning rate* mempengaruhi ketelitian jaringan suatu sistem. Semakin besar *learning rate*, maka ketelitian jaringan akan semakin berkurang. Sebaliknya, apabila *learning rate*-nya semakin kecil, maka ketelitian jaringan akan semakin besar atau bertambah dengan konsekuensi proses pelatihan akan memakan waktu yang lama.

2.3.1 Arsitektur *Backpropagation*

Jaringan ini memiliki satu lapis masukan ditambah dengan sebuah bias, satu atau lebih lapisan tersembunyi (*hidden layer*) ditambah dengan sebuah bias, dan satu lapisan keluaran. Setiap lapis memiliki *neuron-neuron* (unit-unit). Di antara *neuron* pada lapis berikutnya dihubungkan dengan model koneksi yang memiliki bobot-bobot (*weights*).



Gambar 2. 4 Arsitektur *Backpropagation*

Berdasarkan gambar 2.4, v_{ij} merupakan bobot dari *neuron input layer* x_i ke *neuron hidden layer* z_i, dimana v_{1j} merupakan bobot yang menghubungkan bias dari *neuron input layer* ke *hidden layer*. Sedangkan w_{jk} merupakan bobot dari *neuron hidden layer* z_j ke *neuron output layer* y_k, dimana w_{1k} merupakan bobot dari bias di *neuron hidden layer* ke *neuron output layer*.

2.3.2 Fungsi Aktivasi

Ada beberapa fungsi aktivasi yang sering digunakan dalam Jaringan Saraf Tiruan. Penggunaan fungsi aktivasi dalam algoritma *Backpropagation* harus berupa fungsi yang dapat diturunkan. Biasanya menggunakan fungsi aktivasi Sigmoid. Dalam metode *Backpropagation*, fungsi aktivasi yang sering digunakan adalah fungsi Sigmoid Biner (*Binary Sigmoid*), yang mempunyai jangkauan nilai antara 0 (nol) sampai 1 (satu). Nilai dari keluaran dan masukan ada diantara 0 (nol) sampai 1 (satu). Persamaan fungsinya adalah :

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

fungsi turunannya adalah :

$$f(x) = \alpha f(x)[1 - f(x)] \tag{2.2}$$

Learning rate (α) merupakan laju pembelajaran dan e adalah bilangan euler.

2.3.3 Algoritma *Backpropagation*

Pelatihan sebuah jaringan *backpropagation* terdiri dari tiga langkah, yaitu pelatihan pola masukan secara *feedworward*, *backpropagation* dari kumpulan

kesalahan, dan penyesuaian bobot. 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 *Backpropagation* ditentukan dengan *Mean Square Error* (MSE). Semakin kecil nilai MSE maka dapat dianggap bahwa arsitektur jaringan semakin baik. MSE dihitung dengan persamaan 2.3

$$\text{MSE} = \frac{1}{2}\sum_{P}(t_p - y_p)^2 \qquad (2.3)$$

dimana p adalah jumlah *neuron* (unit) keluaran, t adalah target, dan y adalah keluaran.

Algoritma pelatihan *backpropagation* adalah sebagai berikut :

1. Bobot awal ditentukan secara acak dengan nilai nilai sekecil mungkin, misalnya [-0,01,0,01].
2. Inisialisasi *error* target, jumlah *hidden layer*, *learning rate*, dan jumlah maksimum epoch.
3. Selama (epoch<maksimum epoch) dan (MSE<target *error*), maka :
   a. Lakukan *feedforward* untuk setiap pasangan pelatihan :
      1) Tiap-tiap unit masukan (x_i, i=1,2,..,n) menerima sinyal dan menyalurkan sinyal ini ke semua unit lapisan tersembunyi.
      2) Tiap-tiap unit tersembunyi (z_j, j=1,2,…,p) menjumlahkan perkalian nilai masukan dan bobot sinyal masukan dengan Persamaan 2.4

$$z\_in_j = v_{oj} + \sum_{i=1}^{n} x_i v_{ij} \qquad (2.4)$$

Gunakan fungsi aktivasi untuk menghitung *output*nya yaitu :

$$z_j = f(z\_in_j) \qquad (2.5)$$

Lalu kirimkan sinyal tersebut ke semua unit di lapisan atasnya.

3) Tiap-tiap unit keluaran (y_k ; k=1,…,m) akan menjumlahkan bobot sinyal keluaran dengan Persamaan 2.6.

$$y\_in_k = v_{ok} + \sum_{i} z_j w_{jk} \qquad (2.6)$$

Gunakan fungsi aktivasi untuk menghitung sinyal keluaran.

$$y_k = f(y\_in_k) \quad (2.7)$$

b. Lakukan propagasi balik untuk masing-masing pasangan pelatihan :

1) Tiap-tiap unit keluaran (y_k ; k=1,...,m) menerima target yang berkesesuaian dengan pola pelatihan, hitung informasi kesalahan.

$$\delta_k = (t_k - y_k) f'(y\_in_k) \quad (2.8)$$

Kemudian hitung koreksi bobot, digunakan untuk memperbaharui w_{jk} (Persamaan 2.9).

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (2.9)$$

hitung koreksi bias untuk memperbaiki bobot bias (w_{ok}).

$$\Delta w_{ok} = \alpha \delta_k \quad (2.10)$$

Kemudian kirimkan nilai δ_k ke seluruh unit yang berada pada lapisan di bawahnya (*backward*).

2) Tiap-tiap unit tersembunyi (Z_j ; j=1,...,p) menjumlahkan *input delta* dari unit lapisan atasnya.

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (2.11)$$

Kalikan fungsi ini dengan turunan fungsi aktivasi untuk menghitung informasi kesalahan

$$\delta_j = \delta\_in_j f'(y\_in_j) \quad (2.12)$$

Hitung koreksi bobot untuk memperbaiki v_{ij}.

$$\Delta v_{ij} = \alpha \delta_j x_i \quad (2.13)$$

Hitung koreksi bobot bias untuk memperbaiki v_{0j}.

$$\Delta v_{oj} = \alpha \delta_j \quad (2.14)$$

c. Perbaiki bobot dan bias

Tiap-tiap unit keluaran (y_k ; k=1,...,m) memperbaiki bobot dan bias (j=0,...,p).

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.15)$$

Tiap-tiap unit tersembunyi (Z ; 1,...,p) memperbaiki bobot dan bias (i ; 0,...,n).

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad (2.16)$$

1) Jika iterasi mencapai maksimum, maka berhenti.

Setelah proses pelatihan, tahap pengujian jaringan saraf *backpropagation* diaplikasikan dengan hanya menggunakan tahap perambatan maju (*feedforward*) dari algoritma pelatihan. Prosedur pelatihannya adalah sebagai berikut :

a) Inisialisasi bobot yang diperoleh dari proses pelatihan jaringan.

b) Tiap-tiap unit masukan (x_i; i=1,2,...,n) menerima sinyal dan menyalurkan ke semua unit lapisan tersembunyi.

c) Tiap-tiap unit tersembunyi (z_j; j=1,...,p) menjumlahkan perkalian nilai masukan dan bobot sinyal masukan.

$$z_{in_j} = v_{oj} + \sum_{i=1}^{n} x_i v_{ij} \qquad (2.17)$$

Gunakan fungsi aktivasi untuk menghitung keluaran, yaitu:

$$z_j = f(z\_in_j) \qquad (2.18)$$

d) Tiap-tiap unit keluaran (y_k; k=1,...,m) akan menjumlahkan bobot sinyal keluaran.

$$y\_in_k = v_{0k} + \sum_{i} z_k w_{jk} \qquad (2.19)$$

Gunakan fungsi aktivasi untuk menghitung nilai keluaran.

$$y_k = f(y\_in_k) \qquad (2.20)$$

Jika $y_k \geq 0{,}5$ maka $y_k=1$, *else* $y_k=0$.

Keterangan dari simbol-simbol :

t = keluaran vector target, t=(t1,...,xk,...,xm)

δk = informasi tentang kesalahan pada unit y_k yang disebarkan kembali ke unit tersembunyi

δj = informasi tentang kesalahan dari lapisan keluaran ke unit tersembunyi z_j

α = *learning rate*

x_i = unit masukan i

v_{0j} = bobot awal bias pada lapisan tersembunyi j

v_{ij} = bobot menuju *hidden*

z_j = unit tersembunyi j

z_in_j = masukan ke jaringan z_j

w_{0k} = bobot awal bias pada menuju keluaran

w_{jk} = bobot menuju keluaran

y_k = unit keluaran i

y_in_k = masukan jaringan ke y_k

e = error tiap pasangan

2.4 Optimalitas Arsitektur *Backpropagation*

2.4.1 Pemilihan Bobot dan Bias

Bobot awal akan mempengaruhi apakah jaringan mencapai titik minimum lokal atau global, dan seberapa cepat konvergensinya. Bobot yang menghasilkan nilai turunan fungsi aktivasi yang terlalu kecil sebaiknya dihindari, demikian pula nilai bobot awal tidak boleh terlalu besar karena nilai turunan fungsi aktivasinya menjadi sangat kecil.

Nguyen dan Widrow (1990) mengusulkan cara membuat inisialisasi bobot dan bias ke unit tersembunyi sehingga menghasilkan iterasi lebih cepat.

Bila diketahui : n = jumlah unit masukan

p = jumlah unit tersembunyi

β = faktor skala = $0.7\sqrt[n]{p}$

Algoritma penentuan bobot dan bias :

1. Inisialisasi semua bobot (v_{ij}(lama)) dengan bilangan acak dalam interval [-0,05,0,05]
2. Hitung

$$\| v_j \| = \sqrt{v_{1j}^2 + v_{2j}^2 + \cdots + v_{nj}^2} \qquad (2.21)$$

3. Bobot yang dipakai sebagai inisialisasi

$$v_{ij} = \frac{\beta v_{ij}(lama)}{\|v_j\|} \qquad (2.22)$$

4. Bias yang dipakai sebagai inisialisasi v_{0j} = bilangan acak antara $-\beta$ dan β.

2.4.2 Jumlah Unit Tersembunyi

Menentukan jumlah lapisan pada *hidden layer* dan menentukan jumlah unit tiap lapisannya adalah sangat sulit. Secara teoritis ditunjukkan bahwa jaringan dengan sebuah *hidden layer* sudah cukup bagi *backpropagation* 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.

2.4.3 Lama Iterasi

Tujuan utama penggunaan *backpropagation* adalah untuk mendapatkan keseimbangan antara 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 dikenali dengan benar. Tapi hal tersebut tidak menjamin jaringan akan mampu mengenali pola pengujian dengan tepat.

Umumnya data dibagi menjadi dua 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, kesalahan yang terjadi dihitung berdasarkan semua data (pelatihan dan pengujian). Selama kesalahan ini menurun, pelatihan terus dijalankan.Akan tetapi jika kesalahannya meningkat, pelatihan tidak ada gunanya untuk diteruskan lagi.

2.5 Borland Delphi

2.5.1 Pengenalan Delphi

Borland Delphi 7 adalah bahasa pemrograman visual yang dirilis oleh Borland Internasional, dan merupakan pengembangan dari bahasa pemrograman Pascal dan Borland Delphi sebelumnya. Delphi 7 mempunyai banyak tambahan fitur baru dibandingkan versi sebelumnya, sehingga Delphi 7 telah menjadi lingkungan pengembangan yang cukup kompleks.

Dalam penciptaan bahasa Borland Delphi 7 ini, Borland Internasional bertujuan menjadikannya sebagai solusi *e-business*, yang salah satu ujung

tombaknya adalah *Web-service*, disisi lain untuk aplikasi ber*database* Borland 7 sudah cukup proposional dibandingkan versi sebelumnya.

Borland Delphi adalah program untuk membuat aplikasi berbasis Microsoft Windows secara cepat dan mudah. Borland Delphi menyediakan *tool* untuk membuat aplikasi sederhana sampai aplikasi yang kompleks, baik untuk keperluan pribadi maupun untuk keperluan instansi/perusahaan dengan sistem yang lebih besar. Adapun kemampuan lain Borland Delphi 7 di antaranya :

1. Memiliki sarana pengembangan yang bersifat grafis (Visual).
2. Berorientasi objek (*object oriented*).
3. Dapat bekerja did dalam sistem operasi Windows.
4. Dapat menghasilkan program aplikasi berbasis Windows.
5. Mampu memanfaatkan program aplikasi berbasis Windows, seperti grafis, multimedia, internet, multitasking, dan sebagainya.

Borland Delphi berbasiskan *Object Oriented Programming* (OOP) dan dikembangkan dengan basis Visual yang berarti menggunakan sarana grafis untuk mengembangkannya. Borland Delphi berorientasi pada objek-objek yang dipisah-pisah, sehingga disebut pemrograman berorientasi objek. Borland Delphi juga bersifat modular programming, karena kode-kode program letaknya tersebar di dalam modul-modul (objek-objek) yang terpisah.

2.5.2 Lingkungan Kerja Delphi (IDE)

Lingkungan pengembangan terpadu atau *Intergrated Development Environment* (IDE) dalam program Delphi terbagi menjadi enam bagian utama, yaitu, Main Window, Toolbar, Component Palette, Form Designer, Code Editor, dan Object Inspector. IDE merupakan sebuah lingkungan di mana semua tombol perintah yang diperlukan untuk mendesain aplikasi, menjalankan,dan menguji sebuah aplikasi disajikan dengan baik untuk memudahkan pengembangan program. Sebagai sarana interaktif, Delphi menyediakan bermacam-macam komponen interface aplikasi yang berupa tombol, menu dropdown maupun pop-up, kotak teks, kotak gambar, radio button, checkbox, scrollbar, listbox, combobox, dan panel. Dalam pemrograman Delphi terdapat beberapa elemen pemrograman, antara lain: Identifier, tipe data, dan scope.

2.5.3 Delphi dan Aplikasi *Database*

Delphi menyediakan object yang sangat kuat, canggih, dan lengkap, sehingga memudahkan pemrogram dalam merancang, membuat, dan menyelesaikan aplikasi *database* yang diinginkan. Selain itu juga dapat menangani data dalam berbagai format *database*, misalnya format MS-Access, Sybase, Oracle, FoxPro, Informix, DB2, dan lain sebagainya. Format *database* yang dianggap asli dari Delphi adalah Paradox dan dBase.

2.6 MySQL

MySQL merupakan *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi *General Public Lisence* (GPL). Dimana setiap orange bebas untuk menggunakan MySQL, namun tidak boleh dijadikan sebagai produk yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama,yaitu *Structure Query Language* (SQL).

SQL adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

MySQL adalah salah satu jenis *database* yang sangat terkenal, karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses-nya. Keandalan suatu sistem *database* (DBMS) dapat diketahui dari cara kerja *optimizer*-nya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh *user* ataupun program-program aplikasinya. Sebagai *database server*, MySQL dapat dikatakan lebih unggul dibandingkan dengan *database server* yang lainnya dalam *query* data.

2.7 PHP

PHP adalah bahasa pemrograman yang memungkinkan para *web developer* untuk membuat aplikasi *web* yang dinamis dan cepat. PHP merupakan singkatan dari "Hypertext Prepocessor". PHP ditulis dan diperkenalkan pertama kali sekitar tahun 1994 oleh Rasmus Lerdorf melalui situsnya untuk mengetahui siapa saja yang telah mengakses ringkasan *online*-nya.

Untuk dapat berjalan, PHP membutuhkan *web server*, yang bertugas untuk memproses file-file php dan mengirimkan hasil pemrosesan untuk ditampilkan di *browser client*. Oleh karena itu, PHP termasuk *server-side scripting* (*script* yang diproses di sisi *server*). *Web server* sendiri adalah software yang diinstall pada computer local ataupun computer lain yang berada di jaringan intranet/internet yang berfungsi untuk melayani permintaan-permintaan *web* dari *client*.

Karakteristik yang paling unggul dan paling kuat dalam PHP adalah lapisan integrasi *database* (*database integration layer*). *Database* yang didukung PHP antara lain : Oracle, Adabas-D, Sybase, FilePro, mSQL, MySQL, Informix, Solid, dBase, ODBC, Unix dbm, dan PostgreSQL. Contoh script PHP sederhana terdapat pada Listing 1.

Listing 1 : Contoh Script PHP

```
<html>
<head>
<title>coba PHP </title>
</head>
<body>
<?
Echo "belajar PHP";
?>
</body>
</html>
```

2.8 Apache

Server HTTP Apache atau *Server Web*/WWW Apache adalah *web server* yang dapat dijalankan di banyak sistim operasi (Unix, BSD, Linux, Microsoft Windows dan Novell Netware serta platform lainnya) yang berguna untuk melayani dan memfungsikan situs *web*. Protokol yang digunakan untuk melayani fasilitas *web*/*www* ini menggunakan HTTP.

Apache memiliki fitur-fitur canggih seperti pesan kesalahan yang dapat dikonfigur, autentikasi berbasis basis data, serta didukung antarmuka pengguna berbasis grafik (GUI) yang memungkinkan penanganan server menjadi mudah.

BAB III

PERANCANGAN SISTEM

3.1 Deskripsi Sistem

Deskripsi sistem merupakan gambaran tentang sistem yang akan dibuat. Langkah-langkah untuk mendiagnosa penyakit pada burung punglor melalui sebuah sistem, sehingga dihasilkan keluaran berupa kesimpulan tentang penyakit yang menyerang punglor serta solusi untuk mengatasi penyakit tersebut.

3.2 Analisis Sistem

3.2.1 Analisa Kebutuhan Perangkat Lunak

Beberapa spesifikasi kebutuhan yang dibutuhkan dalam rancangan perangkat lunak ini antara lain :

1. Dapat menerima obyek dari pakar berupa nama penyakit, gejala penyakit, serta solusi sebagai masukan untuk pelatihan.
2. Dapat melatih sistem untuk mengenali pola-pola penyakit sesuai dengan aturan Jaringan Saraf Tiruan *Backpropagation.*
3. Dapat menyimpan bobot-bobot hasil pelatihan dalam file pelatihan.
4. Dapat menerima masukan dari *user* berupa gejala-gejala penyakit sebagai masukan untuk dapat mendiagnosa penyakit.
5. Dapat membaca pola pelatihan sesuai aturan Jaringan Saraf Tiruan *Backporpagation.*
6. Dapat menghasilkan keluaran jika aturan sudah sesuai dan masukan gejala yang dimasukkan telah dikenali dalam proses pelatihan.
7. Dapat memberikan laporan hasil diagnosa.

3.2.2 Pengolahan Data

Data yang digunakan dalam aplikasi adalah data gejala penyakit punglor. Data tidak dibatasi, karena tergantung dari jumlah data, dimana data langsung dimasukkan ke dalam aplikasi tanpa harus mengalami perubahan. Pengolahan data hanya mengubah data gejala penyakit ke dalam bentuk biner, yaitu bentuk matrik 1x22, untuk menghasilkan suatu pola yang digunakan sebagai masukan data Jaringan Saraf Tiruan. Dengan membedakan apakah gejala bernilai *true* (1) atau *false* (0). Alur pengolahan data yang dilakukan dalam aplikasi antara lain :

Langkah 1 : Mengambil data gejala yang sudah disimpan dalam berkas.

Langkah 2 : Setelah gejala diambil, maka masukan matrik akan mengubah gejala dalam bentuk matrik diikuti munculnya target yang telah ditentukan sebelumnya.

Langkah 3 : Selesai memasukkan gejala, lakukan dengan masukan untuk pelatihan jaringan saraf.

Langkah 4 : Masukkan jumlah *hidden layer*, maksimum epoh, target *error*, dan *learning rate*.

Langkah 5 : Setelah memasukkan data-data yang dibutuhkan dalam jaringan saraf, kemudian tekan tombol proses untuk memproses jaringan.

Langkah 6 : Kemudian muncul hasil dari segala proses yang telah dilakukan.

3.3 Perancangan Sistem

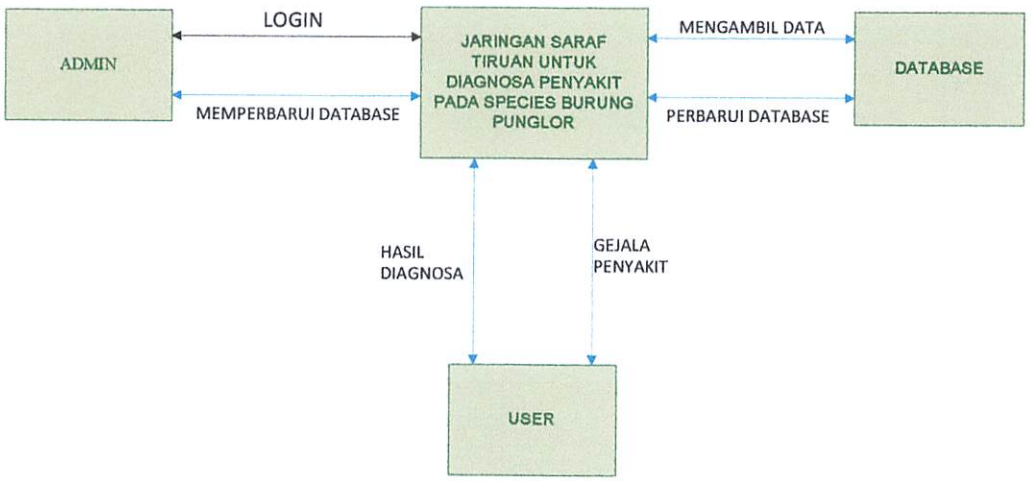
Rancangan dan langkah-langkah yang dilakukan dalam memberikan diagnosis penyakit burung punglor menggunakan Jaringan Saraf Tiruan *Backpropagation*.

Penelitian dilakukan dengan langkah-langkah sebagai berikut :

1. Mempelajari metode yang digunakan, yaitu *Backpropagation* dan penyakit pada burung punglor.
2. Menganalisa dan merancang sistem dengan menggunakan hasil pembelajaran pada tahap sebelumnya.
3. Membuat sistem berdasarkan analisis dan perancangan yang dilakukan.
4. Uji coba sistem.
5. Evaluasi hasil uji coba.

3.4 Desain Sistem

Sebelum membangun sebuah sistem, perlu merancang desain sistem terlebih dahulu, untuk dapat mempermudah pembuatan sistem, dan memberikan gambaran umum tentang sistem tersebut. Dijelaskan pada gambar 3.1



Gambar 3.1 Desain Sistem Jaringan Saraf Tiruan *Backpropagation* untuk Mendiagnosa Penyakit pada Burung Punglor

3.5 Gambaran Umum Perangkat Lunak

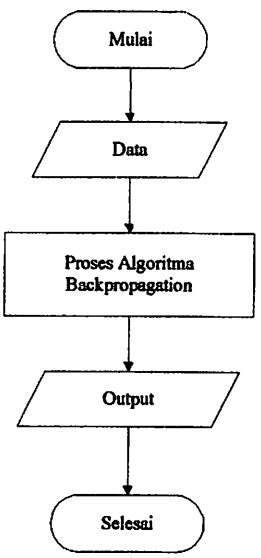
Sistem yang dikembangkan adalah implementasi dari algoritma *Backpropagation* dalam bentuk aplikasi untuk mendiagnosa jenis penyakit pada burung punglor. Sistem akan mendiagnosa gejala yang diberikan oleh *user* dan akan memberikan keluaran berupa hasil diagnosa penyakit terhadap gejala-gejala tersebut.

Dari masukan tersebut akan dikelompokkan ke dalam 9 jenis penyakit, sehingga akan terdapat 9 *neuron* pada lapisan keluaran jaringan saraf. Pada tahap pembelajaran menggunakan metode *feedforward*, lalu akan dihitung besarnya nilai *error*. Jika kesalahan yang ditargetkan belum didapat, maka dilakukan *backpropagation*. Jaringan terdiri dari satu lapisan masukan dengan 22 *neuron*, satu *hidden layer*, dan satu lapisan keluaran dengan 9 *neuron*. Data dilatih hingga mendapatkan *error* yang minimum, bila nilai *error* minimum, maka bobot akan disimpan dalam data penyimpanan. Tempat penyimpanan menggunakan *database* MySQL.

Pada tahap pengujian sistem, dengan menggunakan bobot-bobot akhir yang telah diperoleh selama proses pembelajaran, maka dihasilkan pengelompokan terhadap data pengujian.

Langkah-langkah untuk mendiagnosa penyakit pada burung punglor menggunakan Jaringan Saraf Tiruan *Backpropagation* ditunjukkan pada diagram alir dalam gambar 3.2, dengan tahapan sebagai berikut :

1. Data masukan (*input*) adalah gejala-gejala klinis penyakit.
2. Proses algoritma *backpropagation* terdiri dari proses pelatihan dan pengujian data.
3. Keluaran (*output*), merupakan hasil diagnosis penyakit pada burung punglor.



Gambar 3.2 Diagram Alir Sistem Secara Umum

3.6 Proses Jaringan Saraf Tiruan *Backpropagation*

Proses dibagi menjadi tiga, yaitu proses membuat arsitektur jaringan, proses pelatihan, dan proses pengujian.

3.6.1 Arsitektur

Arsitektur dari *backpropagation* terdiri dari lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer*), dan lapisan keluaran (*output layer*). Arsitektur jaringan saraf yang digunakan terdiri dari 22 unit *neuron* pada lapisan masukan

dan 9 unit *neuron* pada lapisan keluaran. Jumlah *neuron* pada lapisan tersembunyi (*hidden layer*) adalah n, dimana n akan dicoba dalam beberapa nilai.

3.6.2 Penetapan Masukan (*Input*)

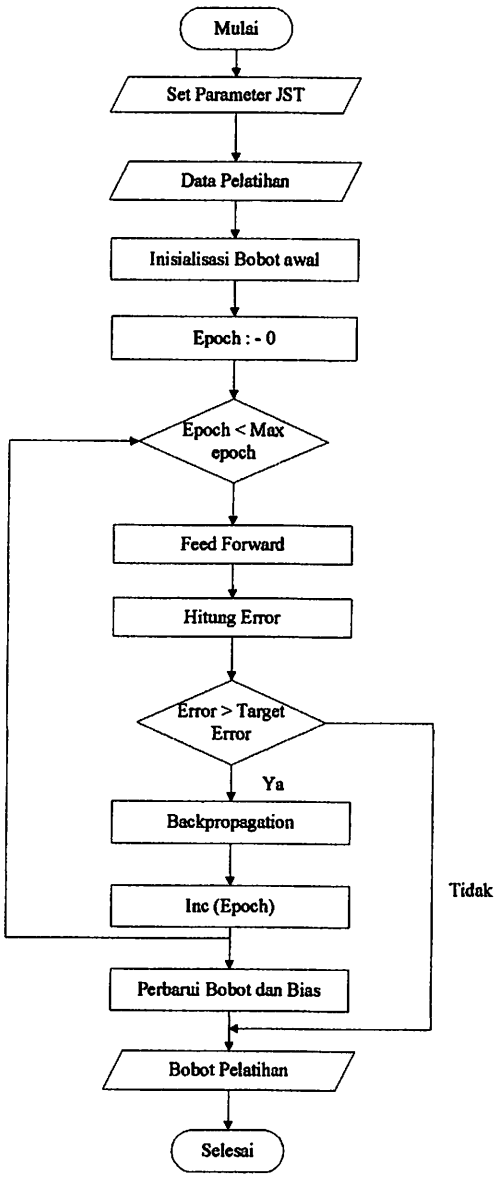
Data masukan yang digunakan dalam aplikasi ada dua jenis. Masukan data pertama adalah data gejala penyakit dan masukan data kedua adalah Jaringan Saraf Tiruan untuk pelatihan jaringan.

Masukan data gejala yang digunakan akan dijadikan data biner, berupa matrik ordo 1x22. Hasil pengolahan data gejala penyakit digunakan sebagai masukan data pelatihan jaringan saraf.

3.6.3 Proses Pelatihan

Algoritma pelatihan backpropagation terdiri dari tiga langkah, yaitu: langkah maju (*feedforward*), propagasi balik (*backpropagation*), dan perubahan bobot. Langkah-langkah terdapat pada gambar 3.3, yaitu :

1. Mulai.
2. Tentukan kesalahan yang ditargetkan, epoh maksimum, dan *neuron hidden.*
3. Masukan data yang akan dilatih.
4. Inisialisasi bobot awal, yang berkisar antara -0,05 sampai 0,05.
5. Tentukan parameter epoh sama dengan 0 (nol).
6. Lakukan *feedforward* setelah jaringan terisi bobot.
7. Hitung kesalahan antara pola keluaran dan pola target.
8. Apakah kesalahan keluaran lebih besar dari kesalahan yang ditargetkan? Jika ya, lakukan langkah 9, jika tidak, maka lakukan langkah 10.
9. Lakukan *backpropagation.*
10. Proses berhenti, bobot dan bias akhir pelatihan disimpan, dan siap untuk digunakan dalam proses pengujian.
11. Selesai.



Gambar 3.3 Diagram Alir Proses Pelatihan Jaringan Saraf Tiruan *Backpropagation*

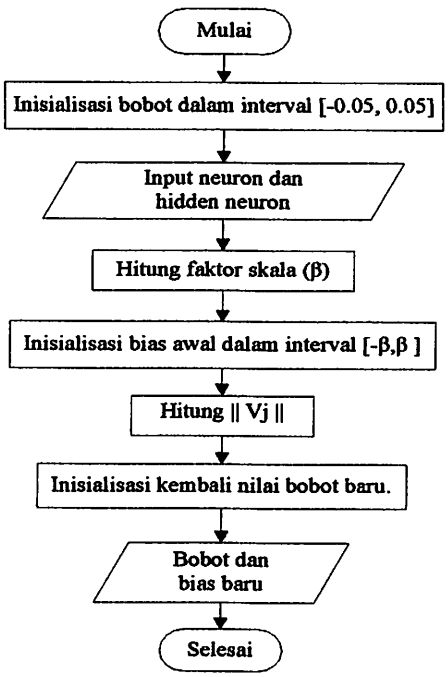
3.6.3.1 Inisialisasi Bobot dan Bias Awal Nguyen Widrow

Inisialisasi digunakan untuk menentukan nilai bobot dan bias awal dari unit masukan ke unit tersembunyi. Bertujuan untuk mempercepat iterasi.

Untuk proses inisialisasi terdapat pada gambar 3.4, yaitu :

1. Mulai.
2. Inisialisasi semua bobot dengan bilangan acak antara -0,05 sampai 0,05.
3. Tentukan jumlah unit masukan dan unit tersembunyi.

4. Hitung faktor skala menggunakan rumus $\beta=0.7\sqrt[n]{p}$.
5. Inisialisasi bias awal dari unit masukan ke unit tersembunyi, dengan nilai [-β, β].
6. Hitung $\| Vj \|$ menggunakan persamaan 2.21.
7. Inisialisasi kembali bobot baru menggunakan persamaan 2.22.
8. Selesai.



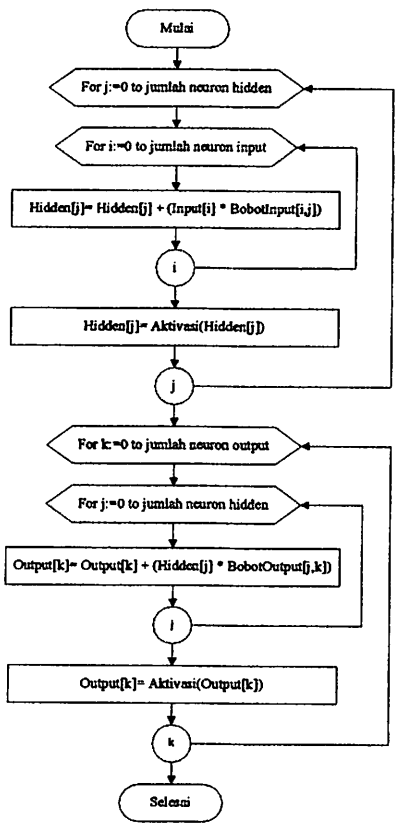
Gambar 3.4 Diagram Alir Proses Inisialisasi Bobot dan Bias Awal Nguyen Widrow

3.6.3.2 *Feedforward*

Proses yang dilakukan dalam tahap ini adalah menjumlahkan perkalian antara masukan dengan bobot yang ada dan menghitung nilai aktivasinya untuk kemudian hasilnya digunakan untuk masukan lapisan yang ada di atasnya. Langkah-langkahnya terdapat pada gambar 3.5, yaitu :

1. Mulai.
2. Kalikan seluruh data masukan dengan bobot acak pada masing-masing bobot koneksi yang terhubung dengan masukan *neuron*. Lalu jumlahkan seluruh bobot yang menuju unit tersembunyi yang sama.
3. Aktivasikan hasil penjumlahan pada masing-masing *neuron* di lapisan tersembunyi, sehingga keluaran berada antara 0 (nol) dan 1 (satu).

4. Kalikan seluruh data hasil aktivasi masing-masing *neuron* pada lapisan tersembunyi dengan bobot masing-masing bobot koneksinya. Lalu jumlahkan seluruh vektor bobot yang menuju keluaran.
5. Aktivasikan hasil penjumlahan pada tiap *neuron* lapisan keluaran, sehingga keluaran akan bernilai antara 0 (nol) dan 1 (satu).
6. Selesai.



Gambar 3.5 Diagram Alir Proses *Feedforward*

3.6.3.3 *Backpropagation*

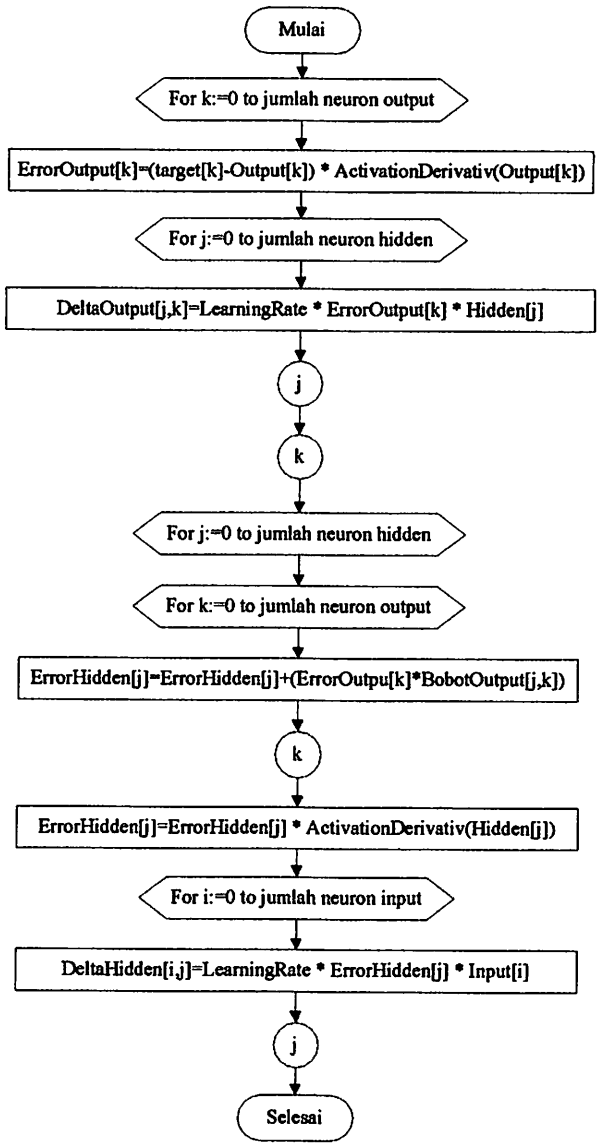
Proses yang dilakukan dalam tahap ini adalah perhitungan in*form*asi kesalahan pada tiap *neuron* masing-masing lapisan, dimulai dari kesalahan pada lapisan keluaran hingga lapisan tersembunyi. In*form*asi kesalahan berguna untuk menghitung faktor peubah bobot yang digunakan untuk memperbaiki bobot lama.

Langkah-langkah *backpropagation* terdapat pada gambar 3.6, yaitu :

1. Mulai.
2. Hitung selisih antara target pelatihan dengan keluaran. Selisih disebut *error*. Kalikan selisih dengan keluaran yang telah diaktivasi dengan fungsi

turunan aktivasi. Hasil perkalian merupakan faktor kesalahan pada lapisan keluaran dan akan digunakan untuk menghitung faktor kesalahan pada lapisan tersembunyi, serta untuk menghitung faktor peubah bobot.

3. Hitung besar faktor peubah bobot terbaru dengan mengalikan *learning rate* dengan lapisan tersembunyi dan *error* pada lapisan keluaran.
4. Masing-masing faktor kesalahan di keluaran dikalikan dengan bobot lama yang terkoneksi dengan *neuron* lapisan keluaran. Lalu hasil perkalian pada semua koneksi pada *neuron* dijumlahkan. Faktor kesalahan ini akan digunakan untuk menghitung peubah bobot dari lapisan masukan ke lapisan tersembunyi.
5. Mengalikan *learning rate* dengan lapisan masukan dan *error* (pada lapisan tersembunyi) untuk mendapatkan nilai faktor peubah bobot baru pada tiap vektor.
6. Selesai.



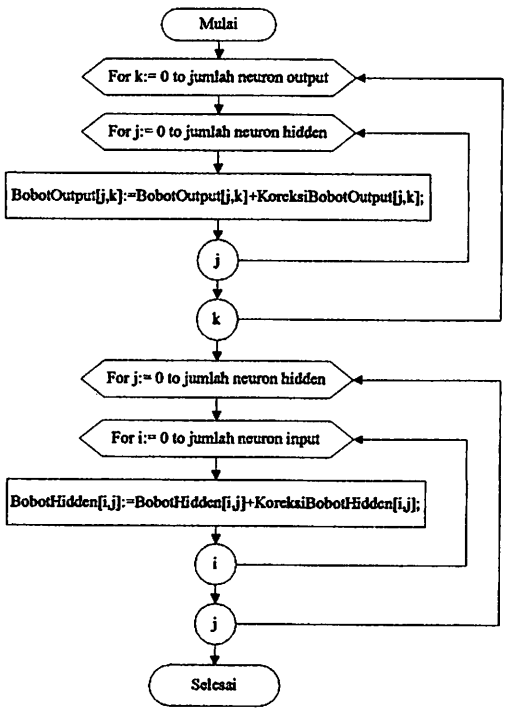
Gambar 3.6 Diagram Alir Proses *Backpropagation*

3.6.3.4 *Update* Bobot

Hasil dari proses *backpropagation* adalah nilai faktor peubah bobot yang digunakan untuk melakukan perubahan bobot baru. Langkah-langkah dalam *update* bobot terdapat pada gambar 3.7, yaitu :

1. Mulai.
2. Perbaiki nilai bobot untuk tiap koneksi menuju lapisan keluaran dengan cara menjumlahkan nilai bobot lama dengan peubah bobot yang telah dihitung pada proses *backpropagation*.

3. Perbaiki nilai bobot ke lapisan tersembunyi dengan menjumlahkan nilai bobot lama dengan peubah bobot yang telah dihitung pada proses *backpropagation.*
4. Selesai.



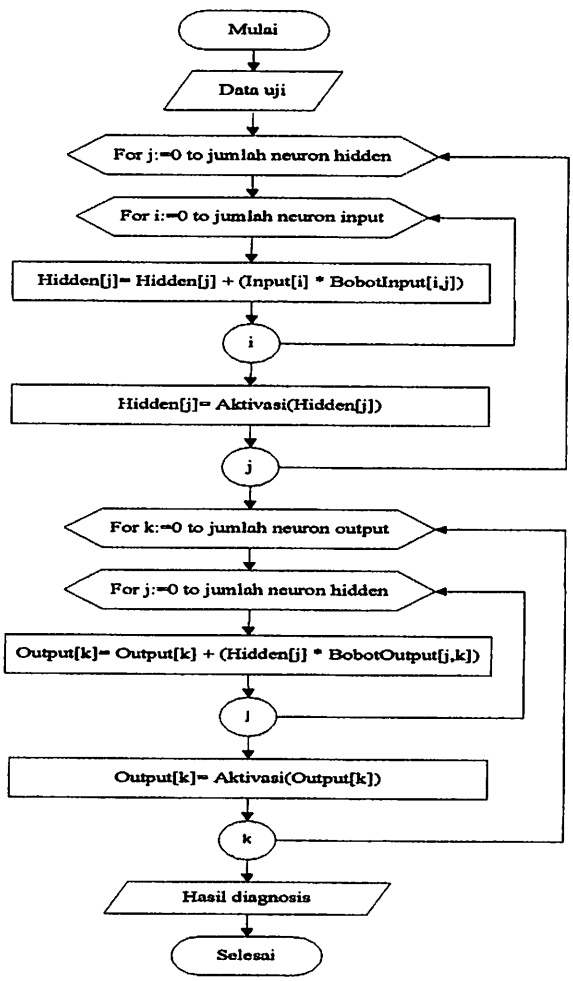
Gambar 3.7 Diagram Alir Proses Perubahan Bobot

3.6.4 Proses Pengujian

Tahap pengujian Jaringan Saraf Tiruan *Backpropagation* dilakukan dengan tahap *feedforward* dari algoritma pelatihan. Dalam proses pengujian, hasil keluaran jaringan tidak diproses ke *backpropagation.* Langkah-langkah pengujian terdapat pada gambar 3.8, yaitu :

1. Mulai.
2. Inisialisasi bobot-bobot yang diperoleh dari proses pelatihan jaringan.
3. Masukkan data pengujian.
4. Kalikan seluruh data masukan pada *neuron* masukan dengan bobot acak pada masing-masing bobot yang terhubung dengan *neuron* masukan. Lalu jumlahkan seluruh vektor bobot menuju *neuron* tersembunyi.

5. Aktivasikan hasil penjumlahan pada masing-masing *neuron* di lapisan tersembunyi, sehingga keluaran akan bernilai antara 0 (nol) dan 1 (satu).
6. Kalikan seluruh data hasil aktivasi masing-masing *neuron* lapisan tersembunyi dengan bobot masing-masing koneksi bobot keluaran. Lalu jumlahkan seluruh vektor bobot yang menuju *neuron* keluaran.
7. Aktivasikan hasil penjumlahan pada masing-masing *neuron* di lapisan keluaran, sehingga keluaran akan bernilai antara 0 (nol) dan 1 (satu).
8. Keluaran berupa hasil diagnosis penyakit.
9. Selesai.



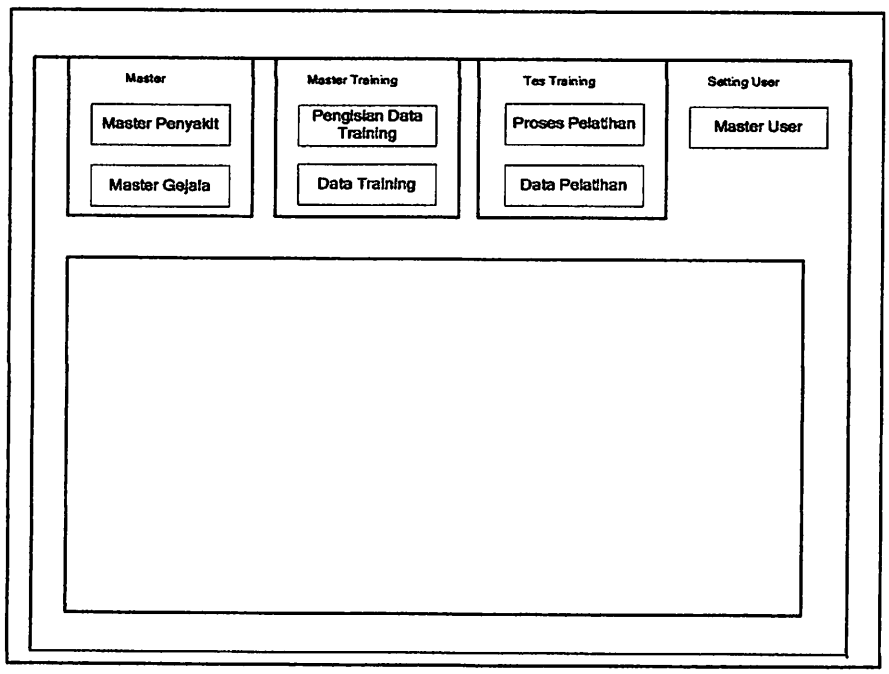
Gambar 3.8 Diagram Alir Proses Pengujian Jaringan Saraf Tiruan

3.7 Perancangan *Form* Utama

Perancangan *form* digunakan untuk menghubungkan perangkat lunak dengan pengguna, agar dapat dioperasikan. Berikut adalah gambar rancangan menu untuk mendiagnosa penyakit pada burung punglor. Terdapat pada gambar 3.9.

- Tampilan menu utama pada halaman admin

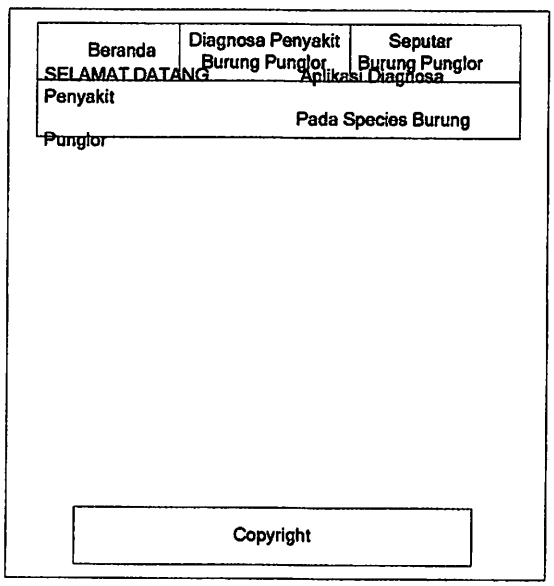
  Tampilan ini berada pada halaman admin (pada *software* Delphi 7).



Gambar 3.9 *Form* Menu untuk Diagnosa Penyakit Burung Punglor

- Tampilan menu utama pada halaman pengguna (*user*)

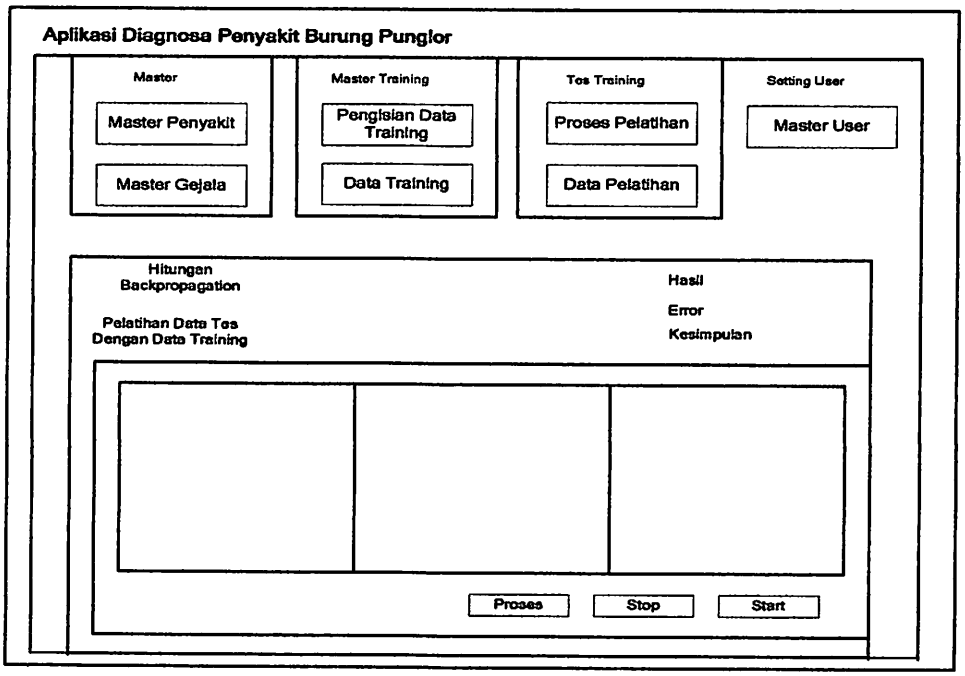
  Tampilan ini berada pada halaman *web*. Terdapat pada gambar 3.10.



Gambar 3.10 *Form* Menu *Web* untuk Diagnosa Penyakit Pada Burung Punglor

a. Rancangan *Form* Pelatihan

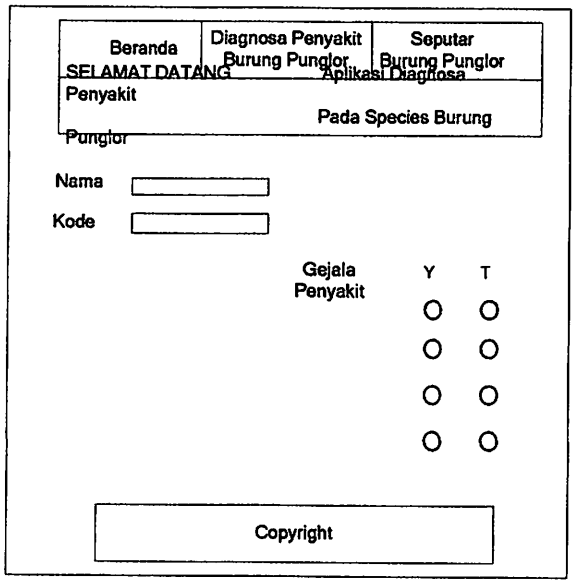
   Pada rancangan *form* ini terdiri dari tombol Master Penyakit, Master Gejala, Pengisian Data Training, Data Training, Proses Pelatihan, Data Pelatihan, dan Master User. Terdapat pada gambar 3.11.



Gambar 3.11 Rancangan *Form* Pelatihan

b. Rancangan *Form* Diagnosa Penyakit (terdapat pada halaman *web*)

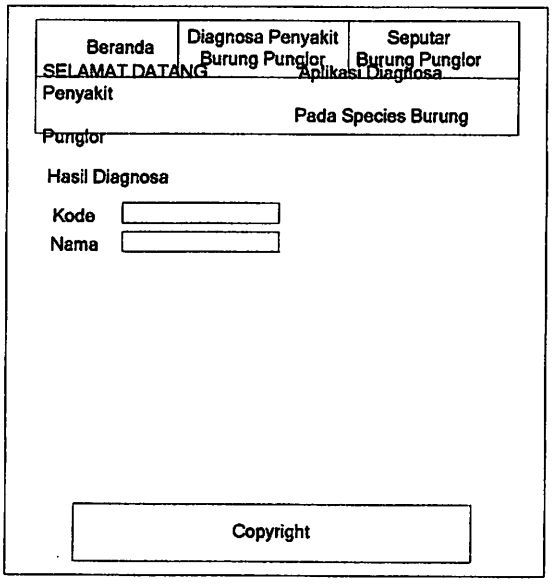
Pada rancangan ini berada pada halaman *web*. Terdiri dari 3 menu utama pada *web* (Beranda, Diagnosa Penyakit Pada Burung Punglor, dan In*form*asi Tentang Burung Punglor), radio button, dan tombol Simpan. Terdapat pada gambar 3.12.



Gambar 3.12 Rancangan *Form* Diagnosa Penyakit pada Tampilan *Web*

c. Rancangan *Form* Tampilan Hasil Diagnosa Penyakit (terdapat pada halaman *web*)

Pada rancangan ini berada pada halaman *web*. Terdiri dari 3 menu utama pada *web* (Beranda, Diagnosa Penyakit Burung Punglor, dan In*form*asi Tentang Burung Punglor), dan tombol Lihat Hasil Diagnosa. Terdapat pada gambar 3.13.



Gambar 3.13 Rancangan *Form* Hasil Diagnosa Penyakit pada Tampilan *Web*

3.8 Struktur Basis Pengetahuan (*Knowledge Base*)

Basis pengetahuan terdiri dari kumpulan data atau fakta-fakta yang digunakan untuk menyelesaikan permasalahan yang khusus. Basis pengetahuan yang digunakan dalam sistem ini berupa aturan nama penyakit dan gejala penyakit. Dijelaskan pada tabel 3.1 dan tabel 3.2 berikut.

Tabel 3.1 Pembentukan Aturan Penyakit Burung Punglor

No	Kode Penyakit	Nama Penyakit	Gejala
1	001	Abes (Bengkak)	a) Kaki terlihat bengkak b) Burung tampak lesu
2	002	Berak Darah	a) Burung tampak lesu b) Sayap terkulai dan jatuh kebawah c) Kotoran berwarna kemerahan karena darah d) Burung tidak mau bertengger dan berdiri di lantai sangkar
3	003	Berak Kapur	a) Burung tampak lesu

			b) Warna bulu menjadi kusam c) Nafsu makan menurun d) Kotoran menjadi encer dan berwarna putih e) Burung sering mengantuk dan kedinginan
4	004	Cacingan	a) Burung tampak lesu b) Nafsu makan menurun c) Kotoran menjadi encer dan berlendir d) Burung menjadi lumpuh
5	005	*Bronchitis*	a) Burung tampak lesu b) Batuk c) Sesak nafas d) Sewaktu bernafas berbunyi d tenggorokan e) Nafsu makan menurun
6	006	Susah Buang Kotoran	a) Burung tampak lesu b) Nafsu makan menurun c) Ada sisa kotoran pada anus burung
7	007	*Newcastle Deases* (ND)	a) Sesak nafas b) Nafsu makan menurun c) Kotoran berwarna hijau kekuningan
8	008	Masuk Angin	a) Burung tampak lesu b) Bulu kusut dan tidak rapi c) Burung tidak mau makan sama sekali d) Kotoran menjadi berair
9	009	Radang Mata	a) Mata membengkak dan berair b) Kelopak mata melekat dan susah terbuka c) Burung tampak lesu d) Burung tampak kebingungan

Tabel 3.2 Pengurutan Gejala untuk Penyakit Burung Punglor

No	Kode Gejala	Gejala
1	G01	Kaki terlihat bengkak
2	G02	Burung tampak lesu
3	G03	Sayap terkulai kebawah
4	G04	Kotoran berwarna kemerahan karena darah
5	G05	Burung tidak mau bertengger
6	G06	Warna bulu menjadi kusam
7	G07	Nafsu makan menurun
8	G08	Kotoran menjadi encer dan berwarna putih
9	G09	Burung sering mengantuk dan kedinginan
10	G10	Kotoran menjadi encer dan berlendir
11	G11	Burung menjadi lumpuh
12	G12	Batuk
13	G13	Sesak nafas
14	G14	Sewaktu bernafas berbunyi d tenggorokan
15	G15	Ada sisa kotoran pada anus burung
16	G16	Kotoran berwarna hijau kekuningan
17	G17	Bulu kusut dan tidak rapi
18	G18	Kotoran menjadi berair

19	G19	Burung tidak mau makan sama sekali
20	G20	Mata membengkak dan berair
21	G21	Kelopak mata melekat dan susah terbuka
22	G22	Burung tampak kebingungan

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi

Lingkungan implementasi yang akan dijelaskan pada sub bab ini adalah lingkungan perangkat keras dan lingkungan perangkat lunak yang digunakan untuk sistem diagnosis jenis penyakit pada burung punglor menggunakan JST *Backpropagation.*

4.1.1 Perangkat Keras

Perangkat keras yang digunakan dalam perancangan aplikasi diagnose penyakit pada burung punglor menggunakan JST *Backpropagation* adalah :

1. Prosesor Intel Atom Inside™ CPU N2550 @1,5 GHz 1MB L2 cache
2. Memori 1016 MB RAM
3. Harddisk kapasitas 250 GB
4. Monitor 10,1 ''
5. Keyboard
6. Mouse

4.1.2 Perangkat Lunak

Perangkat keras yang digunakan dalam perancangan aplikasi diagnosa penyakit pada burung punglor menggunakan JST *Backpropagation* ini adalah :

1. Sistem Operasi *Windows XP*
2. Borland Delphi 7
3. MySQL
4. Dreamweaver 8

4.2 Implementasi Program

Pada sub bab ini akan dibahas mengenai implementasi dari sistem diagnosa penyakit pada burung punglor menggunakan JST *Backpropagation.*

4.2.1.Struktur Data

Untuk memudahkan dalam implementasi jaringan saraf tiruan, penentuan struktur data harus dilakukan dengan baik. Struktur data yang digunakan dalam proses jaringan saraf tiruan dapat dilihat dalam Gambar 4.1.

```
const
  MaxIterasi = 63000;
  MaxUnitMasukan  = 42;
  MaxUnitHidden = 100;
  MaxUnitKeluaran = 12;
  MaxJumlahData = 10000;
var
   X              : array[1..MaxUnitMasukan ] of Extended;
   X_Data         : array[1..MaxUnitMasukan,1..MaxJumlahData] of double;

   T              : array[1..MaxUnitKeluaran ] of Extended;
   T_             : array[1..MaxUnitKeluaran ] of Integer;
   T_Data         : array[1..MaxUnitKeluaran,1..MaxJumlahData] of double;

   Y_hitung       : array[1..MaxUnitKeluaran,1..MaxJumlahData] of double;
   w,delta_w,w_min    : array[0..MaxUnitHidden,1..MaxUnitKeluaran] of
double;
   v,delta_v,v_min    : array[0..MaxUnitMasukan,1..MaxUnitHidden] of
double;
   std_min        : real;
   z,z_in         : array[1..MaxUnitHidden] of double;
   y_in,y         : array[1..MaxUnitKeluaran] of double;
   delta          : array[1..MaxUnitKeluaran] of double;
   delta_in       : array[1..MaxUnitHidden] of double;
   deltaHidden    : array[1..MaxUnitHidden] of double;

   JumlahData          : integer;
   JumlahMasukanUnit : integer;
   JumlahHiddenUnit : integer;
   JumlahKeluaranUnit  : integer;
```

```
  iterasi                : integer;

  MaxMasukan,
  MaxKeluaran      : real;

  Zj                     : array[0..MaxUnitMasukan-1,0..MaxUnitHidden-1] of
Extended;
  Vj             : array[0..MaxUnitMasukan] of Real;


Beta,maxError    : Extended;
  Alpha              : real;
  MaxEpoch       : integer;
  Epoch              : Integer;
  ErrorVal           : Extended;
  ThresHold      : Extended;
  str_               : string;
  showlogs       : Boolean;
```

Gambar 4.1 *Source code* Struktur Data Jaringan Saraf Tiruan

Berikut adalah penjelasan dari struktur data yang terdapat dalam Gambar 4.1 :

1. Ndata merupakan jumlah *neuron masukan*. Berupa ketetapan yang bernilai 22 dikarenakan jumlah masukan yang akan dimasukankan ke dalam sistem berjumlah 22 neuron.
2. Nhidden adalah jumlah *neuron hidden*. Berupa ketetapan yang bernilai 100 dikarenakan jumlah hidden neuron yang akan dimasukankan maksimal 100 neuron.
3. Ntarget merupakan jumlah *neuron keluaran*. Berupa ketetapan yang bernilai 9 dikarenakan jumlah keluaran yang akan dihasilkan oleh sistem sebanyak 9 neuron.
4. X berisi in*Form*asi mengenai nilai dari unit masukan.
5. Z,V,DeltaV

6. Vj,Voj,Zin,Zj berisi in*Forma*si mengenai nilai bobot, bias, dan nilai keluaran dari lapisan masukan ke lapisan tersembunyi
7. Wjk berisi in*Forma*si mengenai nilai bobot dari lapisan tersembunyi ke lapisan keluaran
8. Wok,Yin,Yk berisi in*Forma*si mengenai bias dan nilai keluaran dari lapisan hidden ke lapisan keluaran
9. Delta,DeltaWok berisi in*Forma*si mengenai faktor kesalahan unit keluaran dan nilai perubahan bias dari lapisan tersembunyi ke lapisan keluaran.

10.DeltaWjk berisi in*Forma*si mengenai nilai perubahan bobot dari lapisan tersembunyi ke lapisan keluaran.

11.DeltaIn,DeltaNet,DeltaVoj berisi in*Forma*si mengenai faktor kesalahan di unit tersembunyi serta in*Forma*si nilai perubahan bobot dan bias lapisan masukan ke lapisan tersembunyi.

12.T merupakan in*Forma*si mengenai nilai target.

4.2.2 Prosedur Menghitung Faktor Skala (β)

Prosedur untuk menghitung nilai faktor skala (β) dapat dilihat dalam Gambar 4.2.

```
procedure initParam();
begin
    Randomize;
    DecimalSeparator :='.';
    beta          := ( 0.7 *(power(JumlahHiddenUnit,(1/42))));
    epoch         := 0;
    std_min       := 1;
    iterasi       := 0;
    JumlahInputUnit  := 42;
    JumlahHiddenUnit := 15;
    JumlahOutputUnit := 12;
    MaxEpoch := MaxIterasi;
end;
```

Gambar 4.2 *Source Code* Hitung Faktor Skala

Prosedur di atas berfungsi untuk menghitung faktor skala yang akan digunakan pada inisialisasi bobot dan bias awal Nguyen Widrow.

4.2.3 Prosedur Inisialisasi Bobot Awal Nguyen Widrow

Prosedur InitBobot dalam Gambar 4.3 merupakan prosedur inisialisasi bobot awal Nguyen Widrow yang bertugas untuk membuat inisialisasi bobot dan bias awal dari unit-unit yang berada pada lapisan masukan ke unit-unit yang berada pada lapisan tersembunyi sehingga meghasilkan iterasi lebih cepat. Bobot diinisialisasi secara acak dengan rentang nilai antara -0.05 sampai 0.05 dapat dilihat dalam Gambar 4.3.

```
procedure initBobot();
var i,j: integer;
 temp : real;
begin
 //init bobot awal untuk metode nguyen widrow
 for i:= 0 to JumlahMasukanUnit-1 do
 for j:= 0 to JumlahHiddenUnit-1 do
 begin
    Randomize;
    temp := RandomRange(-5,5)/100;
    Zj[i,j] := (temp);
 end;
end;
```

Gambar 4.3 Prosedur Inisialisasi Bobot Awal Nguyen Widrow

4.2.4 Prosedur Inisialisasi Bobot

Setelah diperoleh nilai bobot dan bias acak awal dari lapisan masukan ke lapisan tersembunyi, maka nilai bobot dan bias tersebut dimodifikasi menggunakan metode Nguyen Widrow. Tahap selanjutnya adalah menginisialiasai nilai awal bobot dan bias dari unit-unit yang berada pada lapisan tersembunyi ke unit-unit yang berada pada layer keluaran. Bobot diinisialisasi

secara acak dengan nilai berkisar antara -0,05 sampai 0,05. Prosedurnya ditunjukkan dalam Gambar 4.4.

```
procedure NewBobot();

var i,j: integer;
 temp : real;
 tempBeta :real;
begin
 //init bobot awal untuk metode nguyen widrow
  initBobot();

//hitung VJ
  for i:= 0 to JumlahMasukanUnit-1 do
  begin
     temp := 0;
     for j:= 0 to JumlahHiddenUnit-1 do
     begin
       temp :=  temp + sqr(Zj[i,j]) ;
     end;
     Vj[i] := roundVal( sqrt(temp)) ;
  end;

//Hitung VJ Baru
  for i:= 0 to JumlahMasukanUnit-1 do
  for j:= 0 to JumlahHiddenUnit-1 do
  begin
     V[i,j]  := ( (Beta * Zj[i,j])/Vj[i] );
  end;
//Generate random value Wjk
  for i:= 0 to JumlahHiddenUnit-1  do
  for j:= 1 to JumlahKeluaranUnit-1 do
  begin
     Randomize;
     temp :=  RandomRange(-5,5)/100;
```

```
    W[i,j] := (temp);
  end;
end;
```

Gambar 4.4 Prosedur Inisialisasi Bobot dan Bias Awal

4.2.5 Prosedur *Feedforward*

Prosedur *Feedforward* merupakan prosedur yang bertugas untuk mengaktifkan unit-unit yang berada pada lapisan tersembunyi dan lapisan keluaran dengan menggunakan fungsi aktivasi pada persamaan 2.1. Prosedurnya dapat dilihat dalam Gambar 4.5.

```
procedure feedforward();

var temp: Real;
   temp2 : Extended ;
   temp3 :Extended;
   i,j: integer;
begin
    Application.ProcessMessages;
    for j := 0 to NLayer-1 do
    begin
     temp :=0;
     for i := 0 to Ndata-1 do
     begin
        temp:= temp+(X[i]*V[j,i]);
     end;
      Zin[j] := roundVal( voj[j] + temp );
    end;

    for j := 0 to NLayer-1 do
    begin

       temp2 :=1/(1+exp(-Zin[j])) ;
       Zj[j] :=roundVal( temp2 );
```

```
end;

for j := 0 to NKeluaran-1 do
begin
 temp :=0;
 for i := 0 to Nhidden-1 do
 begin
    temp:= temp+(Zj[i]*Wjk[j,i]);
 end;
 Yin[j] :=  roundVal(Wok[j] + temp);;
end;

for j := 0 to NKeluaran-1 do
begin
temp := roundVal(  1/(1+exp(-(  Yin[j]))) ) ;
 Yk[j] := temp ;
end;
```

Gambar 4.5 Prosedur *Feedforward*

4.2.6 Prosedur Hitung *Error*

Nilai keluaran (y) yang diperoleh dari proses feedforward, digunakan untuk menghitung in*Form*asi error pada lapisan keluaran. Dalam Gambar 4.6 menunjukkan prosedur untuk menghitung error pada lapisan keluaran. Nilai error ini digunakan untuk mendapatkan nilai koreksi bobot pada lapisan keluaran yang nantinya digunakan untuk menghitung nilai bobot baru pada lapisan keluaran.

```
procedure hitungError() ;

var
 temp :  real;
 j:integer;
begin
    temp :=0;
    for j := 0 to NKeluaran-1 do
```

```
    begin
      temp := temp + sqr(T[j] - Yk[j]);
    end;
    ErrorVal :=(temp/2);
end;
```

Gambar 4.6 Prosedur Hitung *Error*

4.2.7 Prosedur *Backward*

Prosedur untuk melakukan proses *bacward* 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. Prosedur *backward* dapat dilihat dalam Gambar 4.7.

```
procedure backward();
var i,j : integer;
  temp : Extended;
begin
    Application.ProcessMessages;
    for j := 0 to NKeluaran-1 do
    begin
    Delta[j]:=roundVal((T[j]-Yk[j])*( Yk[j])* (1-Yk[j]));
    end;

    for j := 0 to NKeluaran-1 do
    begin
     for i := 0 to NLayer - 1 do
      begin
   DeltaWjk[j,i]:=roundVal(Delta[j]*Zj[i]*Aplha);
     end;
```

```
    end;

    for j := 0 to NKeluaran-1 do
    begin
    DeltaWok[j]:=roundVal( Aplha * Delta[j]);
    end;

    for j := 0 to NLayer - 1 do
    begin
    temp := 0;
     for i := 0 to NKeluaran - 1 do
      begin
      temp:=temp+(Delta[i]*Wjk[i,j]* Aplha);
      end;
      DeltaIn[j] :=roundVal(temp);
    end;

    for j := 0 to NLayer-1 do
    begin
    DeltaNet[j]:=roundVal(DeltaIn[j]*((Zj[j])*(1-Zj[j])));
    end;

   for i:= 0 to Ndata-1 do
   for j:= 0 to NLayer-1 do
   begin
   DeltaV[j,i]:=roundVal(Aplha*X[i]*DeltaNet[j]);
   end;
end;
```

Gambar 4.7 Prosedur *backward*

4.2.8 Prosedur *Update* Bobot

Prosedur update bobot digunakan untuk menghitung bobot dan bias baru dari unit-unit yang berada pada lapisan masukan ke unit-unit yang berada pada lapisan tersembunyi, dan juga bobot dan bias baru dari unit-unit yang berada

pada lapisan tersembunyi ke unit-unit yang berada pada layer keluaran. Prosedur perbarui bobot dan bias dapat dilihat dalam Gambar 4.8

```
procedure PerbaruiBobot ();

var i,j : integer;
 temp : Extended;
begin
    for i:= 0 to NLayer-1 do
    begin
    DeltaVoj[i]:=roundVal(Aplha*DeltaNet[i]);
    end;

   for i:= 0 to Ndata-1 do
   for j:= 0 to NLayer-1 do
   begin
   V[j,i]:= roundVal(V[j,i]+ DeltaV[j,i]);
   end;

    for i:= 0 to NLayer-1 do
    begin
    Voj[i]:=roundVal(Voj[i] + DeltaV[j,i] );
    end;

    for i:= 0 to NKeluaran-1 do
    for j:= 0 to NLayer-1 do

   begin
   Wjk[i,j]:=roundVal(Wjk[i,j]+DeltaWjk[i,j]);
   end;

    for  j := 0 to NKeluaran-1 do
    begin
    wok[j] := roundVal(wok[j] + DeltaWok[j]);
    end;
```

```
end;
```

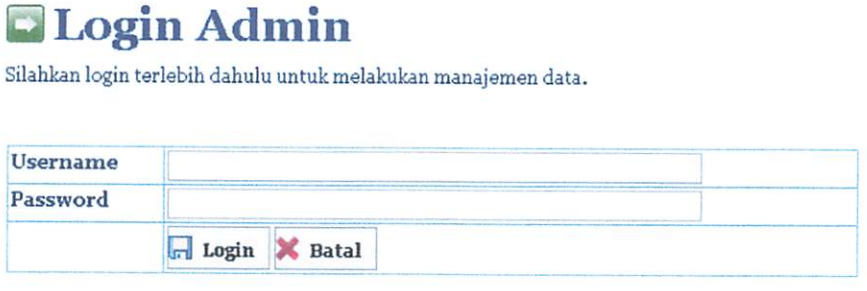
Gambar 4.8 Prosedur perbarui bobot

4.3 Implementasi Program

Berdasarkan perancangan tampilan pada sub bab 3, maka dihasilkan *Form* yang terdiri dari :

1. Tampilan *Form* Admin

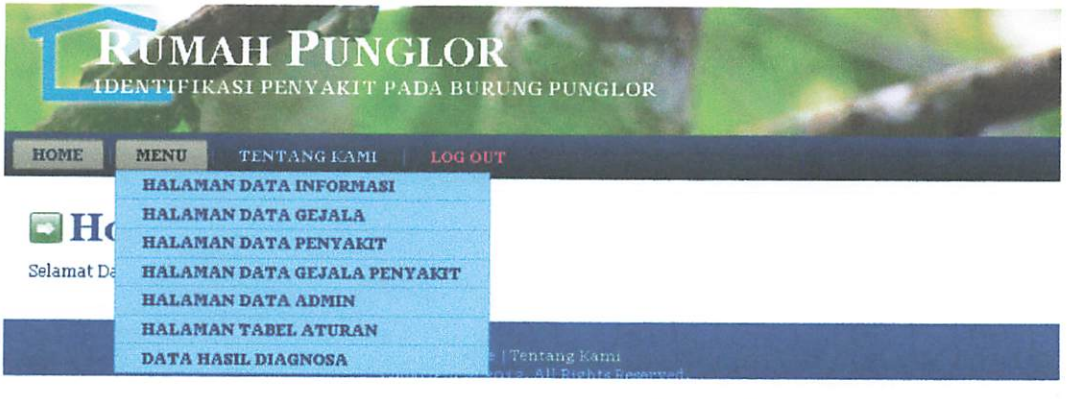
   Pada halaman *login*, akan diminta nama *user* dan *password* untuk dapat melakukan manajemen data. *Form login* dapat dilihat Dalam Gambar 4.9



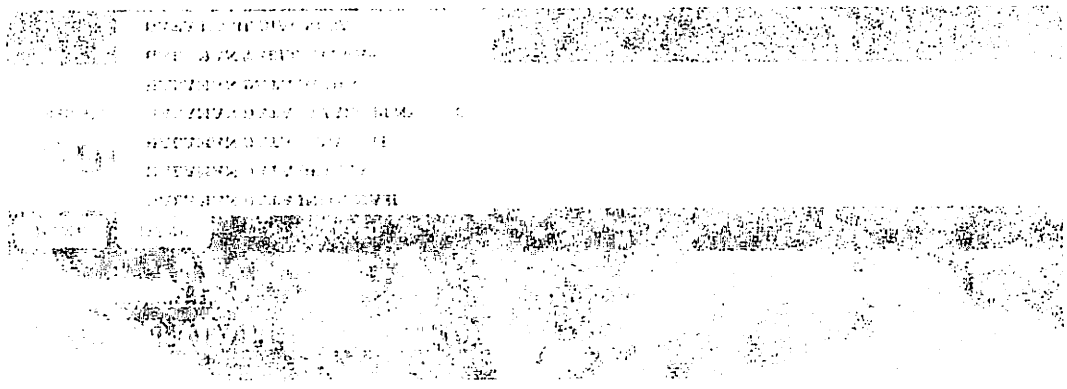
Gambar 4.9 Tampilan *Form* Admin

2. Tampilan *Form* Utama

   Menampilkan menu-menu yang dapat dipilih oleh admin untuk dapat menjalankan aplikasi. Terdapat pada gambar 4.10.



Gambar 4.10 Tampilan *Form* Utama



3. Tampilan *Form* Utama pada Halaman *Web*

   Menampilkan menu-menu yang dapat dipilih oleh pengguna untuk dapat menjalankan program. Terdapat pada gambar 4.11.

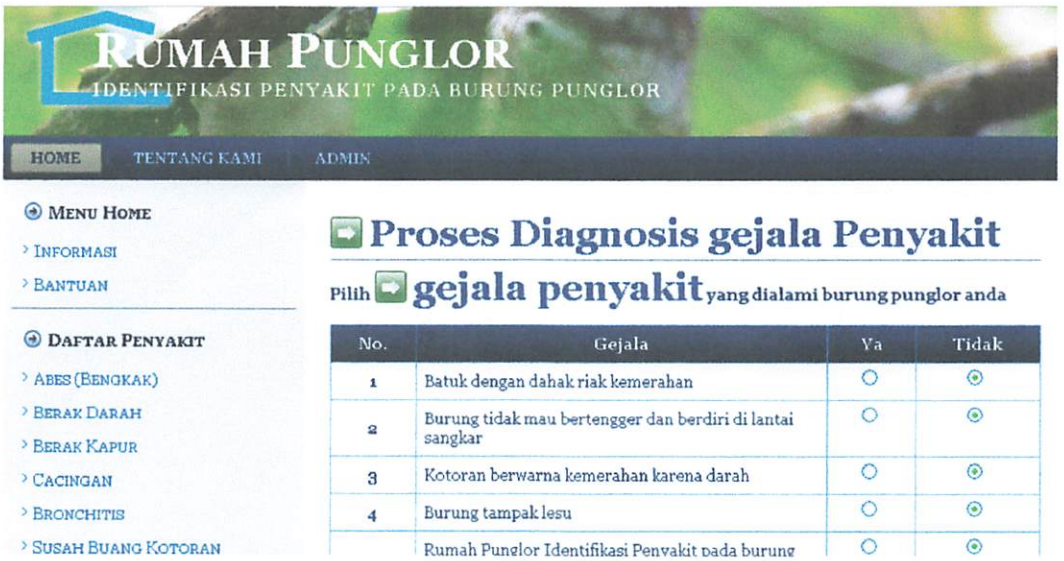
Gambar 4.11 Tampilan *Form* Utama pada Halaman *Web*

1. Tampilan pada halaman pengguna (*web*)

   Langkah-langkahnya adalah :

   1) Pilih menu Konsultasi.
   2) Masukkan nama, alamat, umur, dan pekerjaan.
   3) Pilih gejala yang sesuai pada *radio button* yang ada. Setelah itu klik tombol Simpan.

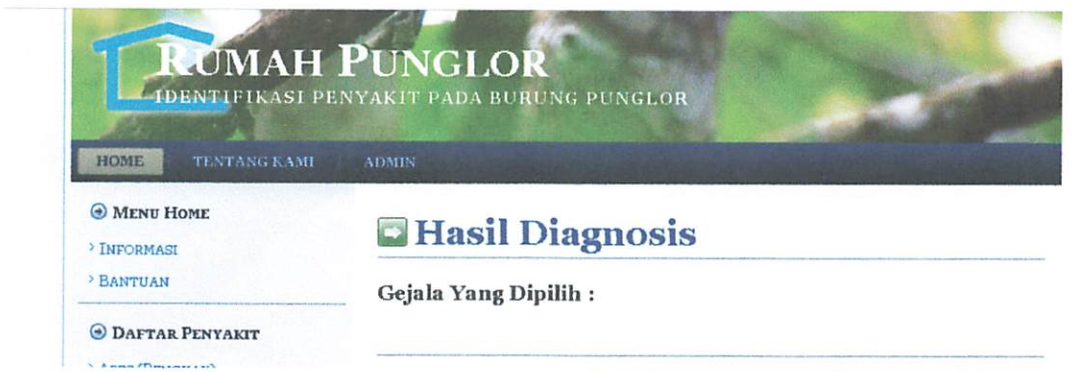
   Terdapat pada gambar 4.12.



Gambar 4.12 Tampilan Halaman *Web* untuk Mendiagnosa Penyakit Punglor

2. Tampilan *Form* Hasil Diagnosa Penyakit pada halaman *web*

   Setelah memasukkan gejala-gejala penyakit, lalu menekan tombol Simpan, maka akan ditampilkan hasil diagnosa penyakit beserta solusi untuk menanganinya. Terdapat pada gambar 4.13.



Gambar 4.13 Tampilan Hasil Diagnosa pada Halaman *Web*

4.4 Penetapan Keluaran (*Output*)

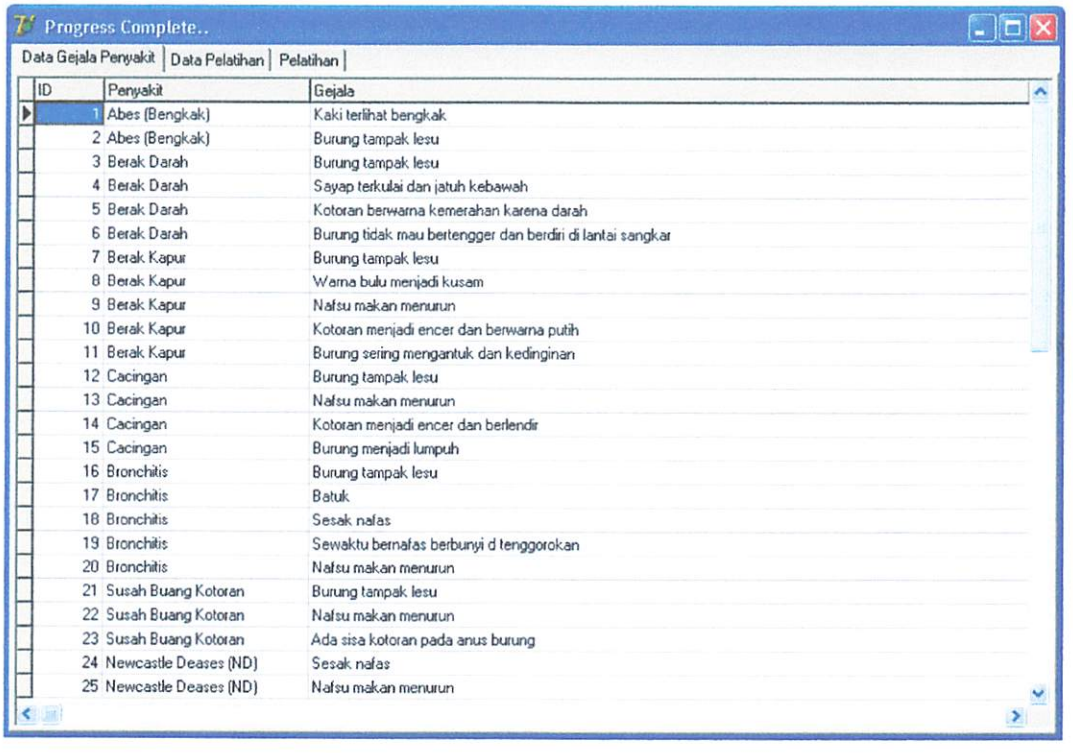
Jika hasil *output* berada pada kisaran $0 \leq output \leq 0{,}5$ maka hasil *output* tersebut dapat dikategorikan pada nilai 0 atau *false*, sementara jika nilai *output* berada pada kisaran $0{,}5 > output \leq 1$ maka hasil *output* tersebut dikategorikan pada nilai 1 atau *true*. Penentuan nilai inilah yang nantinya akan menentukan pola gejala penyakit pada masing-masing penyakit apakah bernilai *false* atau *true*.

4.5 Hasil dan Pembahasan

Untuk memperoleh struktur Jaringan Saraf Tiruan yang terbaik yang digunakan untuk diagnosis penyakit, maka dilakukan pengujian terhadap sistem. Pengujian dilakukan dengan melatih jaringan Saraf tiruan dengan parameter-pamameter yang berbeda, yang nantinya akan diambil satu struktur jaringan yang terbaik yang digunakan untuk melakukan prediksi jenis penyakit pada burung punglor.

4.6 Proses Pelatihan

Untuk proses pelatihan, langkah yang harus dilakukan adalah memasukkan data-data penyakit, beserta gejala-gejalanya. Seperti yang terdapat pada gambar 4.14.

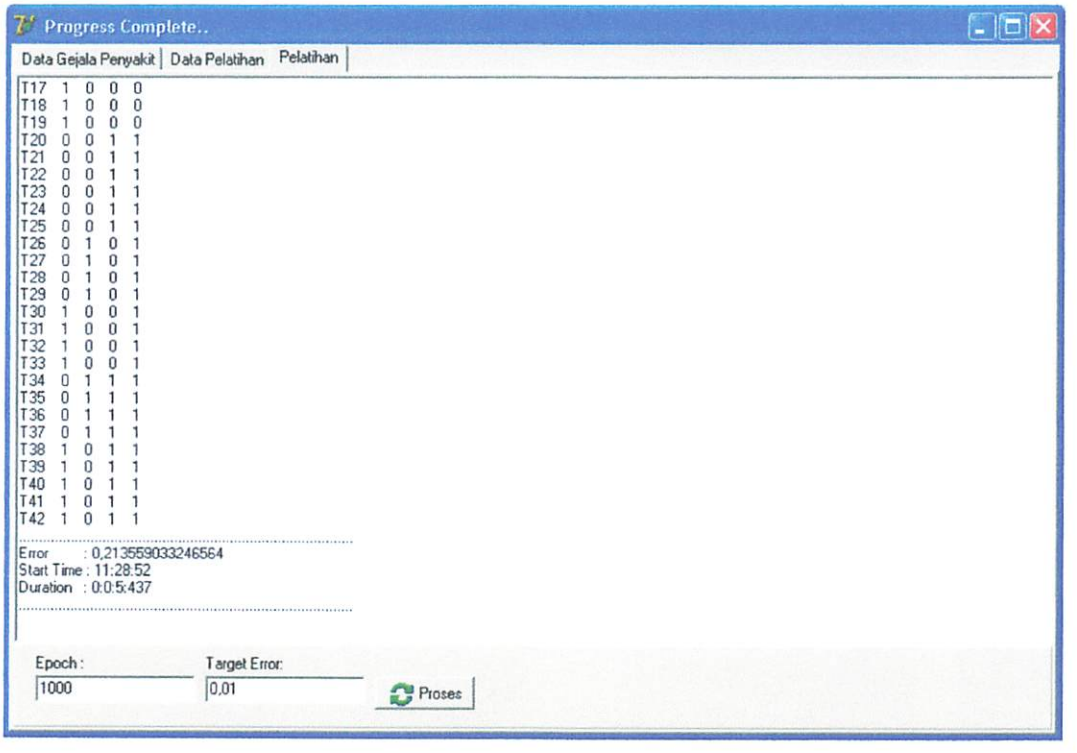


ID	Penyakit	Gejala
1	Abes (Bengkak)	Kaki terlihat bengkak
2	Abes (Bengkak)	Burung tampak lesu
3	Berak Darah	Burung tampak lesu
4	Berak Darah	Sayap terkulai dan jatuh kebawah
5	Berak Darah	Kotoran berwarna kemerahan karena darah
6	Berak Darah	Burung tidak mau bertengger dan berdiri di lantai sangkar
7	Berak Kapur	Burung tampak lesu
8	Berak Kapur	Warna bulu menjadi kusam
9	Berak Kapur	Nafsu makan menurun
10	Berak Kapur	Kotoran menjadi encer dan berwarna putih
11	Berak Kapur	Burung sering mengantuk dan kedinginan
12	Cacingan	Burung tampak lesu
13	Cacingan	Nafsu makan menurun
14	Cacingan	Kotoran menjadi encer dan berlendir
15	Cacingan	Burung menjadi lumpuh
16	Bronchitis	Burung tampak lesu
17	Bronchitis	Batuk
18	Bronchitis	Sesak nafas
19	Bronchitis	Sewaktu bernafas berbunyi d tenggorokan
20	Bronchitis	Nafsu makan menurun
21	Susah Buang Kotoran	Burung tampak lesu
22	Susah Buang Kotoran	Nafsu makan menurun
23	Susah Buang Kotoran	Ada sisa kotoran pada anus burung
24	Newcastle Deases (ND)	Sesak nafas
25	Newcastle Deases (ND)	Nafsu makan menurun

Gambar 4.14 Proses Memasukkan Data Penyakit dan Gejala

Setelah memasukkan seluruh data penyakit beserta gejalanya, maka data tersebut akan dilatih. Proses pelatihan tersebut dilakukan dengan cara memilih gejala yang sesuai dengan jenis penyakit yang akan dilatih satu persatu. Pola yang dimasukkan adalah : Ya (bernilai 1), dan Tidak (bernilai 0). Setelah selesai, maka

hasil kesimpulan, *error*, dan hasil perhitungan akan ditampilkan. Terdapat pada gambar 4.15



Gambar 4.15 Tampilan Hasil Perhitungan

4.6 Analisa Hasil

Setelah mendapatkan arsitektur jaringan yang optimal, kemudian dilanjutkan dengan pengujian terhadap data yang sudah pernah dilakukan pembelajaran. Terdapat 9 data pengujian yang sudah dilakukan pembelajaran sebelumnya. Bobot yang dipakai untuk pengujian adalah bobot dari hasil pembelajaran dengan jumlah unit pada *hidden layer* sebesar 100 unit, nilai *learning rate* sebesar 0.8, target *error* sebesar 0.01, dan *max epoch* sebesar 1000 (yang sudah ditentukan sebelumnya).

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil percobaan dan pembahasan dapat disimpulkan beberapa hal sebagai berikut:

1. Dihasilkan perangkat lunak untuk mendiagnosa jenis penyakit pada burung punglor yang dibangun menggunakan algoritma jaringan saraf tiruan *backpropagation* yang sudah dilatih mampu memberikan diagnosa penyakit dengan cukup baik.
2. Adanya solusi dari seluruh penyakit pada burung punglor, yang terdapat pada halaman *web*, yang dapat memberikan informasi tambahan pagi para pengguna aplikasi.
3. Dari hasil penelitian, arsitektur Jaringan Saraf Tiruan terbaik yang berhasil disusun adalah jaringan saraf *backpropagation* yang terdiri atas lapisan *input* dengan 22 *neuron*, satu lapisan tersembunyi dengan 100 *neuron* dan lapisan *output* dengan 9 *neuron*.
4. Dalam Jaringan Saraf Tiruan, semakin banyak jumlah input yang dihitung semakin lama proses perhitungannya.

5.2 Saran

Adapun saran penulis untuk pengembangan penelitian di bidang yang sama adalah:

1. Penggunaan data pelatihan yang semakin banyak akan memberikan keakuratan jaringan terhadap data uji. Pada penelitian ini masih membutuhkan data training dari masing-masing penyakit yang lebih diperbanyak lagi sehingga dapat mewakili karakteristik data yang akan diujikan.
2. Uji coba aplikasi dibandingkan dengan aplikasi sejenis dengan algoritma atau teknik-teknik klasifikasi data yang lain (misalnya pohon keputusan, *bayesian belief network* dan lain lain) dengan data yang sama sehingga dapat diketahui perbandingan kinerja satu sama lain.
3. Aplikasi dapat dikembangkan lebih lanjut. Dimana jumlah masukan yang berupa variabel pemeriksaan dan keluaran penyakit lebih diperbanyak.

DAFTAR PUSTAKA

[1] Kusumadewi, *Artificial Intelligence,* Graha Ilmu, Yogyakarta, 2003

[2] Puspitaningrum, Pengantar Jaringan Saraf Tiruan, Penerbit Andi, Yogyakarta,2006.

[3] Divisi Penelitian dan Pengembangan, 2003, *Pemrograman Borland Delphi 7*, Andi, Yogyakarta.

[4] S, Husni, 2003, *Pemrograman Database dengan Delphi*, Graha Ilmu, Yogyakarta.

[5] Kadir, Abdul., 2001, "*Pemrograman Database Menggunakan Delphi Jilid 1*", Salemba Infotek, Jakarta.

[6] Kadir, abdul, 2009, *"From Zero To Hero Membuat Aplikasi Web Dengan PHP + Database MySQL"*, Andi, Yogyakarta.

[7] Dewanto, Anang & Sitanggang, Maloedyn, 2009, "Merawat Dan Melatih Burung Kicauan", Agromedia Pustaka, Jakarta

[8] Ragam Jenis Burung Anis, http://yurosie.blogspot.com/2011/07/ragam-jenis-burung-anis.html
Tanggal akses : 04 Mei 2012

[9] http://zulkiflilubis-longbird.blogspot.com/2010/05/katarak-anis-merah-ternyata-bisa.html
Tanggal akses : 04 Mei 2012

[10] Purnomo, H. dan Agus, K, 2006. *Supervised Neural Networks dan Aplikasinya*. Graha Ilmu : Yogyakarta.

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA : BAGUS SANTOSO
NIM : 07.12.630
JURUSAN : Teknik Elektro S-1
KONSENTRASI : Teknik Komputer dan Informatika
MASA BIMBINGAN : Semester Genap Tahun Akademik 2011-2012
JUDUL : **RANCANG BANGUN APLIKASI JARINGAN SARAF TIRUAN UNTUK DIAGNOSA PENYAKIT PADA SPECIES BURUNG PUNGLOR BERBASIS WEB.**

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari : Rabu
Tanggal : 08 Agustus 2012
Dengan Nilai : B+ (73,8)

PANITIA UJIAN SKRIPSI

Ketua Majelis Penguji

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y.1018800189

Sekretaris Majelis Penguji

Dr. Eng. Aryuanto Soetedjo, ST, MT
NIP.P.1030800417

ANGGOTA PENGUJI

Dosen Penguji I

Bambang Prio Hartono, ST, MT
NIP.Y. 1028400082

Dosen Penguji II

Bima Aulia Firmandani, ST

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

FORMULIR PERBAIKAN SKRIPSI

Dalam pelaksanaan ujian skripsi jenjang Strata Satu (S-1) Jurusan Teknik Elektro Konsentrasi Teknik Komputer dan Informatika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : BAGUS SANTOSO
NIM : 07.12.630
JURUSAN : Teknik Elektro S-1
KONSENTRASI : Teknik Komputer dan Informatika
MASA BIMBINGAN : Semester Genap Tahun Akademik 2011-2012
JUDUL : **RANCANG BANGUN APLIKASI JARINGAN SARAF TIRUAN UNTUK DIAGNOSA PENYAKIT PADA SPECIES BURUNG PUNGLOR BERBASIS WEB.**

No	Penguji	Tanggal	Uraian	Paraf
1	Penguji I	8 Agustus 2012	Abstrak Disempurnakan	
2	Penguji II	8 Agustus 2012	Nomor Halaman Disesuaikan Dengan Format Skripsi Yang Benar	
3	Penguji II	8 Agustus 2012	Kesimpulan Nomor 3 Dibuktikan	

Disetujui,

Dosen Penguji I

Bambang Prio Hartono, ST, MT
NIP.Y. 1028400082

Dosen Penguji II

Bima Aulia Firmandani, ST

Mengetahui,

Dosen Pembimbing I

M.Ibrahim Ashari, ST, MT
NIP.P.1030100358

Dosen Pembimbing II

Sotyohadi, ST
NIP.Y. 1039700309

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo, KM 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Bagus Santoso
Nim : 07.12.630
Masa Bimbingan : Semester Genap Tahun Akademik 2011-2012
Judul Skripsi : RANCANG BANGUN APLIKASI JARINGAN SARAF TIRUAN UNTUK DIAGNOSA PENYAKIT PADA SPECIES BURUNG PUNGLOR BERBASIS WEB.

No	Tanggal	Uraian	Paraf Pembimbing
1	10 Juli 2012	REVISI BAB 1 & 2, spasi dan format penulisan dibetulkan.	
2	25 Juli 2012	ACC BAB 1, 2 & 3	
3	27 Juli 2012	ACC BAB 4	
4	30 Juli 2012	Revisi Makalah Seminar Hasil	
5	31 Juli 2012	ACC Makalah Seminar Hasil	
6			
7			
8			
9			
10			

Malang, Juli 2012
Dosen Pembimbing I

M. Ibrahim Ashari,ST, MT.
NIP. Y. 1030100358

Form S-4b

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo, KM 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Bagus Santoso
Nim : 07.12.630
Masa Bimbingan : Semester Genap Tahun Akademik 2011-2012
Judul Skripsi : RANCANG BANGUN APLIKASI JARINGAN SARAF TIRUAN UNTUK DIAGNOSA PENYAKIT PADA SPECIES BURUNG PUNGLOR BERBASIS WEB.

No	Tanggal	Uraian	Paraf Pembimbing
1	4 Juli 2012	Revisi BAB I, II	
2	7 Juli 2012	Acc Laporan BAB I, BAB II.	
3	25 Juli 2012	Revisi Laporan BAB III.	
4	26 Juli 2012	Acc Laporan BAB III, Revisi Laporan BAB IV dan V.	
5	28 Juli 2012	Acc Laporan BAB IV dan V	
7			
8			

Malang, Juli 2012
Dosen Pembimbing II

Sotyohadi, ST
NIP.Y. 1039700309

Form S-4b

SURAT PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini :

Nama	: Bagus Santoso
NIM	: 0712630
Program Studi	: Teknik Elektro S-1
Konsentrasi	: Teknik Komputer & Informatika

Dengan ini menyatakan bahwa Skripsi yang saya buat adalah hasil karya sendiri, tidak merupakan plagiasi dari karya orang lain. Dalam Skripsi ini tidak memuat karya orang lain, kecuali dicantumkan sumbernya sesuai dengan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat, dan apabila di kemudian hari ada pelanggaran atas surat pernyataan ini, saya bersedia menerima sangsinya.

Malang, Agustus 2012

Bagus Santoso
0712630

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Nomor Surat : ITN-205/EL-FTI/2012
Lampiran : -
Perihal : BIMBINGAN SKRIPSI

Kepada : Yth. Bapak/Ibu **M. Ibrahim Ashari, ST, MT**
Dosen Teknik Elektro S-1
ITN MALANG

Dengan Hormat

Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi untuk mahasiswa :

Nama	: **BAGUS SANTOSO**
Nim	: **0712630**
Fakultas	: **Teknologi Industri**
Program Studi	: **Teknik Elektro S-1**
Konsentrasi	: **Teknik Komputer & Informatika**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu :

" **Semester Genap Tahun Akademik 2011-2012** "

Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Ketua Program Studi Teknik Elektro S-1

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y. 1018800189

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

II (PERSERO) MALANG
NK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

omor Surat : ITN-205/EL-FTI/2012
ampiran : -
erihal : BIMBINGAN SKRIPSI

epada : Yth. Bapak/Ibu **Sotyohadi, ST**
Dosen Teknik Elektro S-1
ITN MALANG

Dengan Hormat

Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi untuk mahasiswa :

Nama	: **BAGUS SANTOSO**
Nim	: **0712630**
Fakultas	: **Teknologi Industri**
Program Studi	: **Teknik Elektro S-1**
Konsentrasi	: **Teknik Komputer & Informatika**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu :

" Semester Genap Tahun Akademik 2011-2012 "

Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Ketua Program Studi Teknik Elektro S-1

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y. 1018800189

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Boaz Santoso
NIM : 0712630
Perbaikan meliputi :

Abstrak disempurnakan

Malang,

(_____)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Bagus Santoso
NIM : 07.12.638
Perbaikan meliputi :

- nomer halaman disesuaikan format skripsi yg benar.
- kesimpulan no. 3 dibuktikan !

Malang, 8 Agustus 2011

(_____)

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo, Km. 2
MALANG

PERMOHONAN PERSETUJUAN SKRIPSI

Yang betanda tangan dibawah ini :

Nama	: Bagus Santoso
NIM	: 0712630
Semester	: 10
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: ~~TEKNIK ELEKTRONIKA~~ ~~TEKNIK ENERGI LISTRIK~~ **TEKNIK KOMPUTER DAN INFORMATIKA** ~~TEKNIK KOMPUTER~~ ~~TEKNIK TELEKOMUNIKASI~~
Alamat	: .....

Dengan ini kami mengajukan permohonan untuk mendapatkan persetujuan untuk membuat ***SKRIPSI Tingkat Sarjana.*** Untuk melengkapi permohonan tersebut, bersama kami lampirkan persyaratan-persyaratan yang harus dipenuhi.
Adapun persyaratan-persyaratan pengambilan ***SKRIPSI*** adalah sebagai berikut :

1. Telah melaksanakan semua praktikum sesuai dengan konsentrasinya (.....)
2. Telah lulus dan menyerahkan Laporan Praktek Kerja (.....)
3. Telah lulus seluruh mata kuliah keahlian (MKB) sesuai konsentrasinya (.....)
4. Telah menempuh mata kuliah ≥ 134 sks dengan IPK ≥ 2 dan tidak ada nilai E (.....)
5. Telah mengikuti secara aktif kegiatan seminar skripsi yang diadakan Jurusan (.....)
6. Memenuhi persyaratan administrasi (.....)

Demikian permohonan ini untuk mendapatkan penyelesaian lebih lanjut dan atas perhatiannya kami ucapkan terima kasih.

Telah diteliti kebenaran data tersebut diatas
Recording Teknik Elektro

(Juni Handayani)

Malang, ...2012
Pemohon

(Bagus Santoso)

Disetujui
Ketua Jurusan Teknik Elektro

Ir. Yusuf Ismail Nakhoda, MT
NIP. Y. 1018800189

Mengetahui
Dosen Wali

(Ir. Yusuf Ismail Nakhoda, MT.)

Catatan :
Bagi mahasiswa yang telah memenuhi persyaratan mengambil SKRIPSI agar membuat proposal dan mendapat persetujuan dari Ketua Jurusan/Sekretaris Jurusan T. Elektro S-1

1. PP 364.5 / = 2 ....
2. ..... 135 .....
3. EL 5354 (SD) = E ....

(Sistem Informasi)

LISTING PROGRAM PROSES PERHITUNGAN JST BACKPROPAGATION

```
<?php
error_reporting(1);
class NeuralNetwork {
        var $nodecount = array ();
        var $nodevalue = array ();
        var $nodethreshold = array ();
        var $edgeweight = array ();
        var $learningrate = array (0.1);
        var $layercount = 0;
        var $previous_weightcorrection = array ();
        var $momentum = 0.8;
        var $is_verbose = true;
        var $trainInputs = array ();
        var $trainOutput = array ();
        var $trainDataID = array ();
        var $controlInputs = array ();
        var $controlOutput = array ();
        var $controlDataID = array ();
        var $weightsInitialized = false;
        var $epoch;
        var $error_trainingset;
        var $error_controlset;
        var $success;

        function NeuralNetwork($nodecount) {
                if (!is_array($nodecount)) {
                        $nodecount = func_get_args();
                }
                $this->nodecount = $nodecount;
                 $this->layercount = count($this->nodecount);
        }
        function setLearningRate($learningrate) {
                if (!is_array($learningrate)) {
                        $learningrate = func_get_args();
                }
                $this->learningrate = $learningrate;
        }
        function getLearningRate($layer) {
                if (array_key_exists($layer, $this->learningrate)) {
                        return $this->learningrate[$layer];
                }
                return $this->learningrate[0];
        }
                function setMomentum($momentum) {
                $this->momentum = $momentum;
        }
        function getMomentum() {
                return $this->momentum;
        }
        function calculate($input) {
                foreach ($input as $index => $value) {
                        $this->nodevalue[0][$index] = $value;
                }
                // iterate the hidden layers
                for ($layer = 1; $layer < $this->layercount; $layer ++) {
```

```
                        $prev_layer = $layer -1;
                        // iterate each node in this layer
                        for ($node = 0; $node < ($this->nodecount[$layer]); $node ++) {
                                $node_value = 0.0;
                                // each node in the previous layer has a connection to this node
                                // on basis of this, calculate this node's value

for ($prev_node = 0; $prev_node < ($this->nodecount[$prev_layer]); $prev_node ++) {

\$inputnode_value = $this->nodevalue[$prev_layer][$prev_node];

$edge_weight = $this->edgeweight[$prev_layer][$prev_node][$node];
$node_value = $node_value + ($inputnode_value * $edge_weight);
        }
                                // apply the threshold
                                $node_value = $node_value - $this->nodethreshold[$layer][$node];

                                // apply the activation function
                                $node_value = $this->activation($node_value);

                                // remember the outcome
                                $this->nodevalue[$layer][$node] = $node_value;
                        }
                }
                // return the values of the last layer (the output layer)
                return $this->nodevalue[$this->layercount - 1];
        }
        function activation($value) {
                return tanh($value);
                }
        function derivative_activation($value) {
                $tanh = tanh($value);
                return 1.0 - $tanh * $tanh;
                //return $value * (1.0 - $value);
        }
        function addTestData($input, $output, $id = null) {
                $index = count($this->trainInputs);
                foreach ($input as $node => $value) {
                        $this->trainInputs[$index][$node] = $value;
                }

                foreach ($output as $node => $value) {
                        $this->trainOutput[$index][$node] = $value;
                }
        $this->trainDataID[$index] = $id;
        }
        function getTestDataIDs() {
                return $this->trainDataID;
        }
        function addControlData($input, $output, $id = null) {
                $index = count($this->controlInputs);
                foreach ($input as $node => $value) {
                        $this->controlInputs[$index][$node] = $value;
                }
                foreach ($output as $node => $value) {
                        $this->controlOutput[$index][$node] = $value;
```

```
            }
            $this->controlDataID[$index] = $id;
        }
        function getControlDataIDs() {
            return $this->controlDataID;
        }
        function showWeights($force = false) {
            if ($this->isVerbose() || $force) {
                echo "<hr>";
                echo "<br />Weights: <pre>".serialize($this->edgeweight)."</pre>";
                echo "<br />Thresholds: <pre>".serialize($this->nodethreshold)."</pre>";
            }
        }
        function setVerbose($is_verbose) {
            $this->is_verbose = $is_verbose;
        }
        function isVerbose() {
            return $this->is_verbose;
        }
        function load($filename) {
            if (file_exists($filename)) {
                $data = parse_ini_file($filename);
                if (array_key_exists("edges", $data) && array_key_exists("thresholds", $data)) {
                    // make sure all standard preparations performed
                    $this->initWeights();
                    // load data from file
                    $this->edgeweight = unserialize($data['edges']);
                    $this->nodethreshold = unserialize($data['thresholds']);
                    $this->weightsInitialized = true;
                    // load IDs of training and control set
                    if (array_key_exists("training_data", $data) &&
array_key_exists("control_data", $data)) {
                        // load the IDs
                        $this->trainDataID = unserialize($data['training_data']);
                        $this->controlDataID = unserialize($data['control_data']);
                        $this->controlInputs = array ();
                        $this->controlOutput = array ();
                        $this->trainInputs = array ();
                        $this->trainOutput = array ();
                    }
                    return true;
                }
            }
            return false;
        }
        function save($filename) {
            $f = fopen($filename, "w");
            if ($f) {
                fwrite($f, "[weights]");
                fwrite($f, "\r\nedges = \"".serialize($this->edgeweight)."\"");
                fwrite($f, "\r\nthresholds = \"".serialize($this->nodethreshold)."\"");
                fwrite($f, "\r\n");
                fwrite($f, "[identifiers]");
                fwrite($f, "\r\ntraining_data = \"".serialize($this->trainDataID)."\"");
                fwrite($f, "\r\ncontrol_data = \"".serialize($this->controlDataID)."\"");
                fclose($f);
```

```
                return true;
            }
            return false;
        }
        function train($maxEpochs = 500, $maxError = 0.01) {
            if (!$this->weightsInitialized) {
                $this->initWeights();
            }
            if ($this->isVerbose()) {
                /*echo "<table>";
                echo
"<tr><th>#</th><th>error(trainingdata)</th><th>error(controldata)</th><th>slope(error(controldata))</th></tr>";*
/
            }
            $epoch = 0;
            $errorControlSet = array ();
            $avgErrorControlSet = array ();
            define('SAMPLE_COUNT', 10);
            do {
//              echo "<tr><td colspan=10><b>epoch $epoch</b></td></tr>";
                for ($i = 0; $i < count($this->trainInputs); $i ++) {
                    // select a training pattern at random
                    $index = mt_rand(0, count($this->trainInputs) - 1);
                    // determine the input, and the desired output
                    $input = $this->trainInputs[$index];
                    $desired_output = $this->trainOutput[$index];
                    // calculate the actual output
                    $output = $this->calculate($input);
//                  echo "<tr><td></td><td>Training set $i</td><td>input = (" . implode(", ", $input) . ")</td>";
//                  echo "<td>desired = (" . implode(", ", $desired_output) . ")</td>";
//                  echo "<td>output = (" . implode(", ", $output) .")</td></tr>";
                    // change network weights
                    $this->backpropagate($output, $desired_output);
                }

                // buy some time
                set_time_limit(300);
                //display the overall network error after each epoch
                $squaredError = $this->squaredErrorEpoch();
                if ($epoch % 2 == 0) {
                    $squaredErrorControlSet = $this->squaredErrorControlSet();
                    $errorControlSet[] = $squaredErrorControlSet;
                    if (count($errorControlSet) > SAMPLE_COUNT) {
                        $avgErrorControlSet[] = array_sum(array_slice($errorControlSet, -
SAMPLE_COUNT)) / SAMPLE_COUNT;
                    }
                    list ($slope, $offset) = $this->fitLine($avgErrorControlSet);
                    $controlset_msg = $squaredErrorControlSet;
                } else {
                    $controlset_msg = "";
                }
                if ($this->isVerbose()) {
                    /*echo
"<tr><td><b>$epoch</b></td><td>$squaredError</td><td>$controlset_msg";
                    echo "<script type='text/javascript'>window.scrollBy(0,100);</script>";
                    echo "</td><td>$slope</td></tr>";
```

```
                        echo "</td></tr>";*/
                        flush();
                        ob_flush();
                }
                // conditions for a 'successful' stop:
                // 1. the squared error is now lower than the provided maximum error
                $stop_1 = $squaredError <= $maxError || $squaredErrorControlSet <= $maxError;
                // conditions for an 'unsuccessful' stop
                // 1. the maximum number of epochs has been reached
                $stop_2 = $epoch ++ > $maxEpochs;
                // 2. the network's performance on the control data is getting worse
                $stop_3 = $slope > 0;
        } while (!$stop_1 && !$stop_2 && !$stop_3);

        $this->setEpoch($epoch);
        $this->setErrorTrainingSet($squaredError);
        $this->setErrorControlSet($squaredErrorControlSet);
        $this->setTrainingSuccessful($stop_1);
        if ($this->isVerbose()) {
        /*      echo "</table>";*/
        }
        return $stop_1;
}
function setEpoch($epoch) {
        $this->epoch = $epoch;
}
function getEpoch() {
        return $this->epoch;
}
function setErrorTrainingSet($error) {
        $this->error_trainingset = $error;
}
function getErrorTrainingSet() {
        return $this->error_trainingset;
}
function setErrorControlSet($error) {
        $this->error_controlset = $error;
}
function getErrorControlSet() {
        return $this->error_controlset;
}
function setTrainingSuccessful($success) {
        $this->success = $success;
}
function getTrainingSuccessful() {
        return $this->success;
}
function fitLine($data) {
        $n = count($data);
        if ($n > 1) {
                $sum_y = 0;
                $sum_x = 0;
                $sum_x2 = 0;
                $sum_xy = 0;
                foreach ($data as $x => $y) {
                        $sum_x += $x;
```

```
                        $sum_y += $y;
                        $sum_x2 += $x * $x;
                        $sum_xy += $x * $y;
                }
                // implementation of formula (12)
                $offset = ($sum_y * $sum_x2 - $sum_x * $sum_xy) / ($n * $sum_x2 - $sum_x *
$sum_x);
                // implementation of formula (13)
                $slope = ($n * $sum_xy - $sum_x * $sum_y) / ($n * $sum_x2 - $sum_x * $sum_x);
                return array ($slope, $offset);
        } else {
                return array (0.0, 0.0);
        }
    }
    function getRandomWeight($layer) {
        return ((mt_rand(0, 1000) / 1000) - 0.5) / 2;
    }
    function initWeights() {
        // assign a random value to each edge between the layers, and randomise each threshold
        //
        // 1. start at layer '1' (so skip the input layer)
        for ($layer = 1; $layer < $this->layercount; $layer ++) {

                $prev_layer = $layer -1;
                // 2. in this layer, walk each node
                for ($node = 0; $node < $this->nodecount[$layer]; $node ++) {
                        // 3. randomise this node's threshold
                        $this->nodethreshold[$layer][$node] = $this->getRandomWeight($layer);
                        // 4. this node is connected to each node of the previous layer
                        for ($prev_index = 0; $prev_index < $this->nodecount[$prev_layer];
$prev_index ++) {
                                // 5. this is the 'edge' that needs to be reset / initialised
                                $this->edgeweight[$prev_layer][$prev_index][$node] = $this-
>getRandomWeight($prev_layer);
                                // 6. initialize the 'previous weightcorrection' at 0.0
                                $this->previous_weightcorrection[$prev_layer][$prev_index] = 0.0;
                        }
                }
        }
    }
    function backpropagate($output, $desired_output) {
        $errorgradient = array ();
        $outputlayer = $this->layercount - 1;
        $momentum = $this->getMomentum();
        // Propagate the difference between output and desired output through the layers.
        for ($layer = $this->layercount - 1; $layer > 0; $layer --) {
                for ($node = 0; $node < $this->nodecount[$layer]; $node ++) {
                        // step 1: determine errorgradient
                        if ($layer == $outputlayer) {
                                // for the output layer:
                                // 1a. calculate error between desired output and actual output
                                $error = $desired_output[$node] - $output[$node];
                                // 1b. calculate errorgradient
                                $errorgradient[$layer][$node] = $this-
>derivative_activation($output[$node]) * $error;
                        } else {
```

```
                                        // for hidden layers:
                                        // 1a. sum the product of edgeweight and errorgradient of the 'next'
layer
                                        $next_layer = $layer +1;
                                        $productsum = 0;
                                        for ($next_index = 0; $next_index < ($this->nodecount[$next_layer]);
$next_index ++) {
                                                $_errorgradient = $errorgradient[$next_layer][$next_index];
                                                $_edgeweight = $this-
>edgeweight[$layer][$node][$next_index];
                                                $productsum = $productsum + $_errorgradient *
$_edgeweight;
                                        }
                                        // 1b. calculate errorgradient
                                        $nodevalue = $this->nodevalue[$layer][$node];
                                        $errorgradient[$layer][$node] = $this-
>derivative_activation($nodevalue) * $productsum;
                                }
                                // step 2: use the errorgradient to determine a weight correction for each node
                                $prev_layer = $layer -1;
                                $learning_rate = $this->getLearningRate($prev_layer);
                                for ($prev_index = 0; $prev_index < ($this->nodecount[$prev_layer]);
$prev_index ++) {
                                        // 2a. obtain nodevalue, edgeweight and learning rate
                                        $nodevalue = $this->nodevalue[$prev_layer][$prev_index];
                                        $edgeweight = $this->edgeweight[$prev_layer][$prev_index][$node];
                                        // 2b. calculate weight correction
                                        $weight_correction = $learning_rate * $nodevalue *
$errorgradient[$layer][$node];
                                        // 2c. retrieve previous weight correction
                                        $prev_weightcorrection = $this-
>previous_weightcorrection[$layer][$node];
                                        // 2d. combine those ('momentum learning') to a new weight
                                        $new_weight = $edgeweight + $weight_correction + $momentum *
$prev_weightcorrection;
                                        // 2e. assign the new weight to this edge
                                        $this->edgeweight[$prev_layer][$prev_index][$node] = $new_weight;
                                        // 2f. remember this weightcorrection
                                        $this->previous_weightcorrection[$layer][$node] =
$weight_correction;
                                }
                                // step 3: use the errorgradient to determine threshold correction
                                $threshold_correction = $learning_rate * -1 * $errorgradient[$layer][$node];
                                $new_threshold = $this->nodethreshold[$layer][$node] + $threshold_correction;

                                $this->nodethreshold[$layer][$node] = $new_threshold;
                        }
                }
        }
        function squaredErrorEpoch() {
                $RMSerror = 0.0;
                for ($i = 0; $i < count($this->trainInputs); $i ++) {
                        $RMSerror += $this->squaredError($this->trainInputs[$i], $this->trainOutput[$i]);
                }
                $RMSerror = $RMSerror / count($this->trainInputs);
                return sqrt($RMSerror);
```

```
	}
	function squaredErrorControlSet() {
		if (count($this->controlInputs) == 0) {
			return 1.0;
		}
		$RMSerror = 0.0;
		for ($i = 0; $i < count($this->controlInputs); $i ++) {
			$RMSerror += $this->squaredError($this->controlInputs[$i], $this->controlOutput[$i]);
		}
		$RMSerror = $RMSerror / count($this->controlInputs);
		return sqrt($RMSerror);
	}
	function squaredError($input, $desired_output) {
		$output = $this->calculate($input);
		$RMSerror = 0.0;
		foreach ($output as $node => $value) {
			//calculate the error
			$error = $output[$node] - $desired_output[$node];
			$RMSerror = $RMSerror + ($error * $error);
		}
		return $RMSerror;
	}
}
?>
```