RANCANG BANGUN APLIKASI DIAGNOSA KERUSAKAN KOMPUTER DENGAN METODE BACKPROPAGATION

Dian Puspita Hapsari[1)], M. Tsalits Nururrohman[2)], Reza Uttungga[3)]
[1)] Program Studi Sistem Informasi, Institut Teknologi Adhi Tama Surabaya
Arief Rachman Hakim 100, Surabaya-60117
Email : hapsari_dp04@yahoo.com [1)]

Abstrak

Kerusakan komputer merupakan suatu masalah yang membutuhkan seorang teknisi untuk memperbaikinya. Seorang teknisi terkadang salah dalam mendiagnosa atau bahkan membutuhkan waktu yang cukup lama untuk mediagnosa kerusakan. Hal itu dipengaruhi oleh tingkat pengetahuan setiap teknisi yang berbeda. Untuk mengatasi semua permasalahan itu, dibutuhkan sebuah aplikasi berbasis web untuk mendiagnosa kerusakan pada komputer dengan metode Backpropagation.Setiap keluhan kerusakan diklasifikasikan menjadi sebuah pola biner, kemudian ditentukan target kerusakan dari setiap pola tersebut. Aplikasi kemudian dilatih berulangkali sampai Jaringan Saraf Tiruan dapat mengenali pola pelatihan sesuai dengan yang ditargetkan. Hasil analisis penelitian menunjukkan bahwa Jaringan Saraf Tiruan dengan metode Backpropagation adalah sistem pemrosesan informasi yang bertujuan untuk melatih jaringan agar mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan dan kemampuan jaringan untuk memberikan respon yang benar terhadap pola masukan yang serupa (tetapi tidak sama) dengan pola yang digunakan selama pelatihan. Dari 20 kali percobaan prediksi, aplikasi dapat melakukan prediksi dengan ketepatan 100%. Sedangkan prediksi secara manual didapatkan hasil ketepatan 75%. Dengan demikian, output dari masing-masing jaringan saraf tiruan dapat dijadikan sebagai pertimbangan teknisi dalam mengambil sebuah keputusan untuk mengatasi permasalahan kerusakan Komputer.
Kata kunci: *Kerusakan Komputer, Jaringan Saraf Tiruan, Backpropagation.*

Abstract

Computer damage is a problem requiring an intervention by a technician to fix it. However, a technician sometimes is wrong in diagnosing the damage and even takes longer time to do so. This is specifically due to the level of his knowledge, skill and experience. In order to cope with such problem, it requires a web-based application for diagnosing damages in computers by means of a backpropagation method. Initially each complaint on the damage is classified into a binary pattern. Next, based on the binary pattern, the damaged is identified. The application then goes on trial for sometimes until the Artificial Neural Network can identify the training pattern in compliance with the target. The results of the research analysis showed that the Artificial Neural Network with backpropagation method is an information processing system for network exercising to attain a balance between the ability of the network to identify the adopted pattern during the exercise and the ability of the network to give correct responses to patterns of input similar the ones used in the network exercising. The results of 20 predictive tests showed that the application could predict the problem accurately by 100%. On the other hands, the accuracy of the of manual prediction was 75%. In conclusion, the output of each Artificial Neural Network was adoptable as a consideration by a technician to take decision to cope with the computer problem.

Keyword : *computer damage, Artifical Neural Network, backpropagation*

1. Pendahuluan

Seiring dengan berkembangnya teknologi informasi, semakin tinggi pula tingkat penggunaanya. Salah satunya adalah komputer. Komputer memiliki tingkat penggunaan yang tinggi karena telah menjadi bagian dari kehidupan sehari-hari. Tingginya tingkat pemanfaatan komputer berbanding terbalik dengan pengetahuan pengguna mengenai masalah teknis komputer. Padahal komputer yang

digunakan dalam kegiatan sehari-hari dapat mengalami kerusakan sehingga tidak dapat menjalankan fungsinya dengan maksimal.

Masalah kerusakan komputer merupakan kasus yang membutuhkan seorang teknisi untuk menyelesaikan sebuah masalah dengan pengetahuan yang dimilikinya. Saat ini teknisi membutuhkan waktu yang cukup lama untuk mendiagnosa kerusakan komputer, apalagi seorang teknisi pemula. Bahkan sering kali seorang teknisi menunda pekerjaannya demi menemukan sebuah solusi untuk kerusakan komputer. Berdasarkan permasalahan diatas, dibutuhkan suatu aplikasi diagnosa kerusakan pada PC dengan menggunakan salah satu metode jaringan saraf tiruan yaitu Backpropagation yang dapat menghasilkan output hasil pendiagnosaan kerusakan komponen komputer beserta solusi yang akan diambil untuk mengatasi kerusakan. Aplikasi yang dibuat harus mampu menangani masalah pendiagnosaan kerusakan komputer, baik hardware dan software yang mengalami kerusakan.Untuk itu aplikasi jaringan saraf tiruan diagnosis berbasis web ini berfungsi sebagai solusi dari permasalahan kerusakan pada komputer, aplikasi ini akan membantu mengidentifikasi kerusakan pada komputer, menemukan penyebab kerusakan pada komputer, dan memberikan solusi sebagai problem solving kerusakan.

2. Metode

Jaringan saraf tiruan (artifical neural networks) atau disingkat JST adalahsistem komputasi dimana arsitektur dan operasi diilhami dari pengetahuan tentang sel saraf biologi di dalam otak [1]. Seperti halnya telah diungkapkan di atas, jaringan saraf tiruan merupakan salah satu representasi buatan dari otak manusia. Otak buatan ini dapat berpikir seperti manusia dan menyerupai kepandaian manusia dalam menyimpulkan sesuatu dan selalu mencoba mensimulasikan proses pembelajaran pada otak tersebut. Jaringan saraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran. Dengan cara melakukan peniruan terhadap aktivitas-aktivitas yang terjadi di dalam sebuah jaringan saraf biologis.

Jaringan saraf tiruan tidak diprogram untuk menghasilkan keluaran tertentu. Semua kesimpulan atau keluaran yang ditarik oleh jaringan didasarkan pada pengalamannya selama mengikuti proses pembelajaran. Pada proses pembelajaran ini pola-pola input serta output dimasukkan kedalam jaringan saraf tiruan, lalu jaringan akan diajari untuk memberikan jawaban yang bisa diterima. Pengetahuan jaringan saraf tiruan diperoleh melalui sebuah proses pembelajaran kontinyu.Koneksi- koneksi antar unit-unit proses memiliki bobot atau nilai informasi yang digunakan untuk menyimpan hasil pembelajaran yang telah dilakukan oleh jaringan saraf tiruan tersebut. Algoritma backpropagation dibagi menjadi 2 bagian yaitu algoritma pelatihan dan algoritma pengujian, antara lain:

1. Algoritma Pelatihan

Algoritma pelatihan untuk jaringan dengan satu lapisan tersembunyi (dengan fungsi aktifasi *sigmoidbiner*) adalah sebagai berikut:

Langkah 0: Inisialisasi semua bobot (dengan bilangan acak kecil), bias, toleransi kesalahan, konstanta pembelajaran, momentum, maksimal iterasi dan parameter lainnya.

Langkah 1: Jika kondisi penghentian belum terpenuhi atau *stop condition* (kesalahan lebih kecil atau sama dengan batas toleransi kesalahan, iterasi lebih kecil atau sama dengan maksimal iterasi) lakukan langkah 2-9.

Langkah 2 : Untuk setiap pasang data pelatihan, lakukan langkah 3 - 8.

Fase 1: Propagasi Maju

Langkah 3: Masing-masing unit masukan x_i (i = 1, 2, ..., n) menerima sinyal masukan x_i dan sinyal tersebut disebarkan atau meneruskannya ke bagian berikutnya (unit-unit lapisan tersembunyi diatasnya).

Langkah 4: Masing-masing unit di lapisan tersembunyi z_j (j = 1, 2, ... , p). Menjumlahkan sinyal masukan berbobot, dengan persamaan 1:

$$z\_net_j = v_{j0} + \sum_{i=1}^{n} x_i v_{ji} \quad \text{..... (1)}$$

Dimana: z_net_j adalah jumlah total *input* untuk unit ke-j. v_{j0} adalah bias yang menghubungkan antara unit ke-0 dengan unit ke-j. n adalah jumlah unit pada lapisan tersembunyi sebelumnya. Untuk lapisan tersembunyi yang pertama, n adalah jumlah unit masukan. x_i adalah nilai masukan unit ke-I v_{ji} adalah bobot yang menghubungkan antara unit ke-i dengan unit ke-j. Kemudian menghitung keluaran setiap unit sesuai dengan fungsi aktifasi yang digunakan, dengan /persamaan 2:

$$z_j = f(z\_net_j) = \frac{1}{1+e^{-z\_net_j}} \quad \text{..... ……(2)}$$

Kemudian mengirim sinyal tersebut ke semua unit keluaran (unit *output*).

Langkah 5: Masing-masing unit keluaran y_k (k = 1, 2, ... , m) dikalikan dengan faktor penimbang, dan dijumlahkan, dengan persamaan 3:

$$y\_net_k = w_{k0} + \sum_{j=1}^{p} z_j w_{kj} \quad \text{……..... (3)}$$

Dimana: y_net_k adalah jumlah total masukan untuk unit ke-k (k = 1, ... , m). w_{k0} adalah bias yang menghubbungkan antara unit ke-0 dengan unit ke-k. w_{kj} adalah bobot yang menghubungkan antara unit ke-j dengan unit ke-k. Menghitung nilai keluaran setiap unit pada lapisan keluaran dengan fungsi aktivasi sesuai dengan fungsi aktivasi yang telah digunakan, dengan persamaan 4. y_k adalah nilai keluaran dari unit ke-k

$$y_k = f(y\_net_k) = \frac{1}{1+e^{-y\_net_k}} \quad \text{..... (4)}$$

Fase 2: Propagasi Mundur

Langkah 6: Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran y_k (k = 1, 2, m), dengan persamaan 5:

$$\delta_k = (t_k - y_k) f'(y\_net_k) = (t_k - y_k) y_k (1 - y_k) \quad \text{..... (5)}$$

Dimana: δ_k adalah unit kesalahan yang akan dipakai dalam perubahan bobot lapisan di bawahnya

Langkah 7: t_k adalah target ke-k. Hitung suku perubahan bobot w_{kj} (k = 1, 2, … ,m ; j = 0, 1, … , p) dengan laju pemahaman α, dengan persamaan 6:

$$\Delta w_{kj} = \alpha \delta_k z_j \quad \text{……..... (6)}$$

Dimana: Δw_{kj} adalah perubahan bobot yang menghubungkan unit ke-j dengan unit ke-k, α adalah laju pemahaman *(learning rate)* Hitung perubahan bias pada setiap unit pada lapisan tersembunyi, dengan persamaan 7:

$$\Delta w_{k0} = \alpha \delta_k \quad \text{……..... (7)}$$

Dimana: Δw_{k0} adalah perubahan bobot yang menghubungkan unit ke-0 dengan unit ke-k. Langkah 7: Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi z_j (j = 1, 2, ... , p) dikalikan delta dan dijumlahkan sebagai masukan ke unit-unit lapisan berikutnya, dengan persamaan 8:

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k w_{kj} \quad \text{…….. (8)}$$

Dimana: δ_net_j adalah jumlah total kesalahan untuk unit ke-j. Hitung actor δ setiap unit pada lapisan tersembunyi, dengan persamaan 9:

$$\delta_j = \delta\_net_j f'(z\_net_j) = \delta\_net_j z_j (1 - z_j) \quad \text{…….…..... (9)}$$

Kemudian menghitung suku perubahan bobot v_{ji} (j = 1,2,..,p;i=0,1,..,n), dengan persamaan 10:

$$\Delta v_{ji} = \alpha \delta_j x_i \quad \text{……..... (10)}$$

Kemudian menghitung perbaikan bias (untuk memperbaiki v_{j0}), dengan persamaan 11:

$$\Delta v_{j0} = \alpha \delta_i \quad \text{..... ……..(11)}$$

Dimana: Δv_{j0} adalah perubahan bias yang menghubungkan unit ke-j dengan unit ke-0.

Fase 3: Perubahan Bobot

Langkah 8: Hitung semua perubahan bobot. Perubahan bobot garis yang menuju ke unit keluaran (k = 1,2,…,m ;j=0,1,…,p), pada persamaan 12:

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj} \quad \text{..... (12)}$$

Perubahan bobot garis yang menuju ke unit tersembunyi (j = 1,2,..,p ; i = 0, 1, ... , n), dengan persamaan 13:

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji} \quad \text{..... (13)}$$

Langkah 9: Uji kondisi pemberhentian (akhir iterasi)

Setelah pelatihan selesai dilakukan, jaringan dapat dipakai untuk pengenalan pola. Dalam hal ini, hanya propagasi maju (langkah 4 dan 5) saja yang dipakai untuk menentukan keluaran jaringan. Apabila fungsi aktivasi yang dipakai bukan *sigmoidbiner*, maka langkah 4 dan 5 harus disesuaikan. Demikian juga turunannya pada langkah 6 dan 7.

2. Algoritma pengujian

Yang digunakan hanya tahap umpan maju saja. Berikut ini adalah algoritma pengujian atau juga disebut dengan algoritma aplikasi:

Langkah 0: Inisialisasi semua bobot (dengan bilangan acak kecil), bias, toleransi kesalahan, konstanta pembelajaran, momentum, maksimal iterasi dan parameter lainnya.

Langkah 1: Jika kondisi penghentian belum terpenuhi atau *stop condition* (kesalahan lebih kecil atau sama dengan batas toleransi kesalahan,iterasi lebih kecil atau sama dengan maksimal iterasi), lakukan langkah 2-9.

Langkah 2 : Untuk setiap pasang data pelatihan, lakukan langkah 3 - 8.

Fase 1: Propagasi Maju

Langkah 3: Masing-masing unit masukan x_i (i = 1, 2, ..., n) menerima sinyal masukan x_i dan sinyal tersebut disebarkan atau meneruskannya ke bagian berikutnya (unit-unit lapisan tersembunyi diatasnya).

Langkah 4: Masing-masing unit di lapisan tersembunyi z_j (j = 1, 2, ... , p). Menjumlahkan sinyal masukan berbobot, dengan persamaan 14:

$$\text{z\_net}_\text{j} = v_{j0} + \sum_{i=1}^{n} x_i v_{ji} \quad \text{..... (14)}$$

Kemudian hitung keluaran setiap unit sesuai dengan fungsi aktifasi yang digunakan, persamaan 15:

$$z_j = \text{f}\,(\text{z\_net}_\text{j}) = \frac{1}{1+e^{-z\_net_j}} \quad \text{..... (15)}$$

Kemudian mengirim sinyal tersebut ke semua unit keluaran (unit *output*).

Langkah 5: Masing-masing unit keluaran y_k (k = 1, 2, ... , m) dikalikan dengan faktor penimbang dan dijumlahkan, dengan persamaan 16:

$$\text{y\_net}_\text{k} = w_{k0} + \sum_{j=1}^{p} z_j w_{kj} \quad \text{..... (16)}$$

Dimana: W_{kj} adalah bobot yang menghubungkan antara unit ke-j dengan unit ke-k. Menghitung nilai keluaran setiap unit pada lapisan keluaran dengan fungsi aktivasi sesuai dengan fungsi aktivasi yang telah digunakan, dengan persamaan 17:

$$y_k = \text{f}\,(\text{y\_net}_\text{k}) = \frac{1}{1+e^{-y\_net_k}} \quad \text{..... (17)}$$

3. Perancangan Sistem

Perancangan ini merupakan implementasi dari bagian struktur program yang dibuat untuk mencapai tujuan perangkat lunak yang diinginkan.Rancangan diagram alir sistem *(flowchart sistem)* dari aplikasi diagnosa komputer ini dapat dilihat pada gambar 1.

Flowchart sistem pada gambar 1 menjelaskan bahwa terdapat interaksi antara 3 (tiga) entitas dalam sistem, antara lain adalah: Tenaga ahli yang berperan sebagai *trainer* sekaligus admin dari aplikasi, sistem (aplikasi diagnosa kerusakan komputer), dan Teknisi yang berperan sebagai *user* aplikasi.

Dari *flowchart* sistem pada gambar 1, terdapat 2 (dua) proses perhitungan algoritma, yaitu:

1. Proses Pelatihan (*Learning*)

Proses pelatihan merupakan tahap penyesuaian bobot-bobot jaringan dan hasil dari pelatihan bobot-bobot koneksi antar sel. Proses pelatihan dilakukan berulang-ulang sehingga dihasilkan jaringan yang memberikan tanggapan yang benar terhadap semua masukannya. Proses pelatihan dengan metode *backpropagation* ditunjukkan oleh gambar 3. Proses pelatihan diawali dengan *input* maksimum *epoch*, α (*learning rate*) dan target *error* dari pengguna program (dengan level kuasa 'Tenaga Ahli'). Selanjutnya bobot awal dari lapisan *input* ke lapisan tersembunyi, bias ke lapisan tersembunyi, lapisan tersembunyi ke lapisan *output*, dan bias ke lapisan *output* diinisialisasikan secara acak dengan nilai 0 sampai 1.Setelah tahap penetapan bobot awal, tahap selanjutnya adalah membuka data pelatihan, dimana data pelatihan ini merupakan keputusan diagnosa kerusakan terdahulu yang telah ditetapkan dan terget kerusakan terdahulu

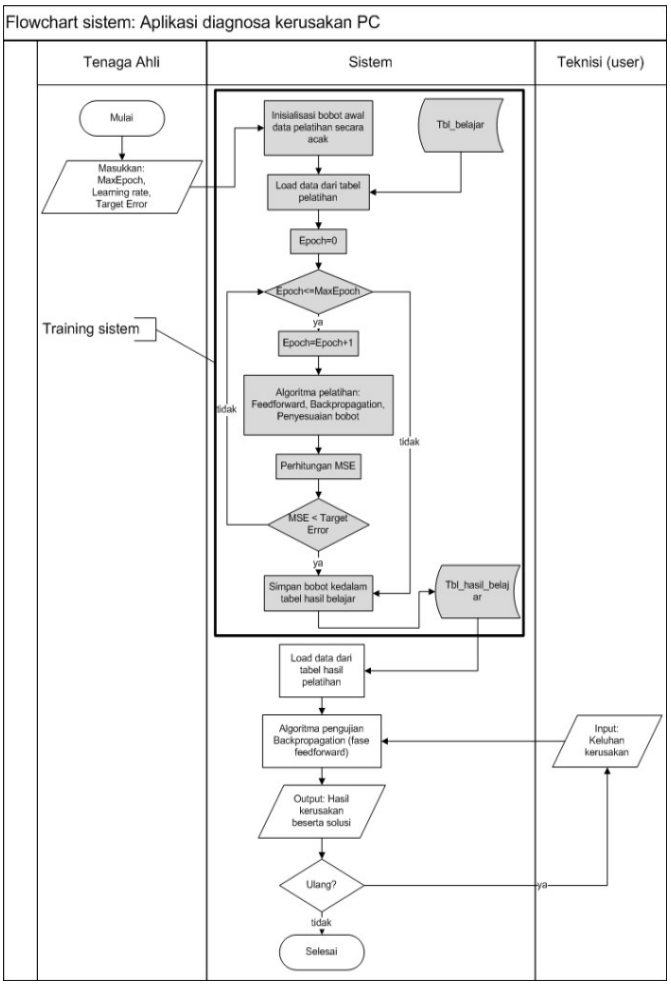
Data pelatihan dapat di*input*kan menggunakan fasilitas *input* data pembelajaran pada program aplikasi. Dimana data pelatihan untuk JST Diagnosa kerusakan hardware pada PC adalah tabel belajar masukan 1 dan tabel belajar keluaran 1, data data pelatihan untuk JST Diagnosa kerusakan software pada PC adalah tabel belajar masukan 2 dan tabel keluaran 2. Tahap berikutnya adalah peng-*assign*-an *epoch* dengan nilai 0. Setelah tahap tersebut, maka ada tes percabangan yang mempunyai 2 kondisi sebagai berikut:

***Epoch*<= Maksimal *Epoch*:** Jika kondisi ini terpenuhi maka *epoch* akan di-*increment* dengan jarak 1 (*epoch*=*epoch*+1) dan berlaku pada setiap iterasi sampai berhenti pada batas maksimal *epoch* yang di-*input*-kan. Kemudian dalam setiap *epoch* dilakukan perhitungan algoritma pelatihan.

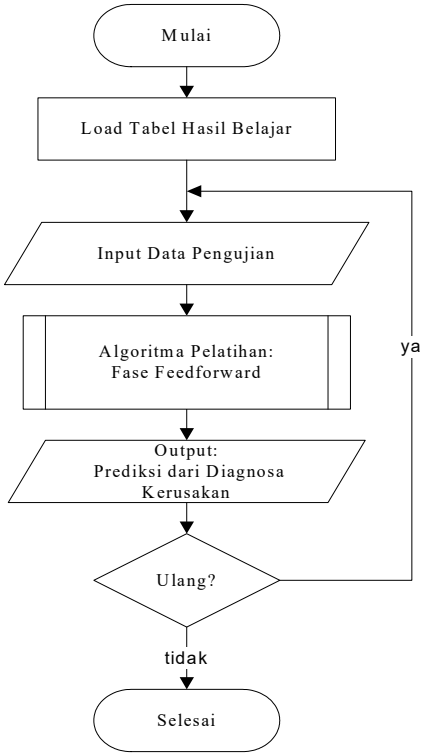
***Epoch*> Max*Epoch*:** Jika kondisi ini terpenuhi, maka hasil pelatihan disimpan ke tabel hasil belajar.

2. Proses Pengujian (*Testing*)

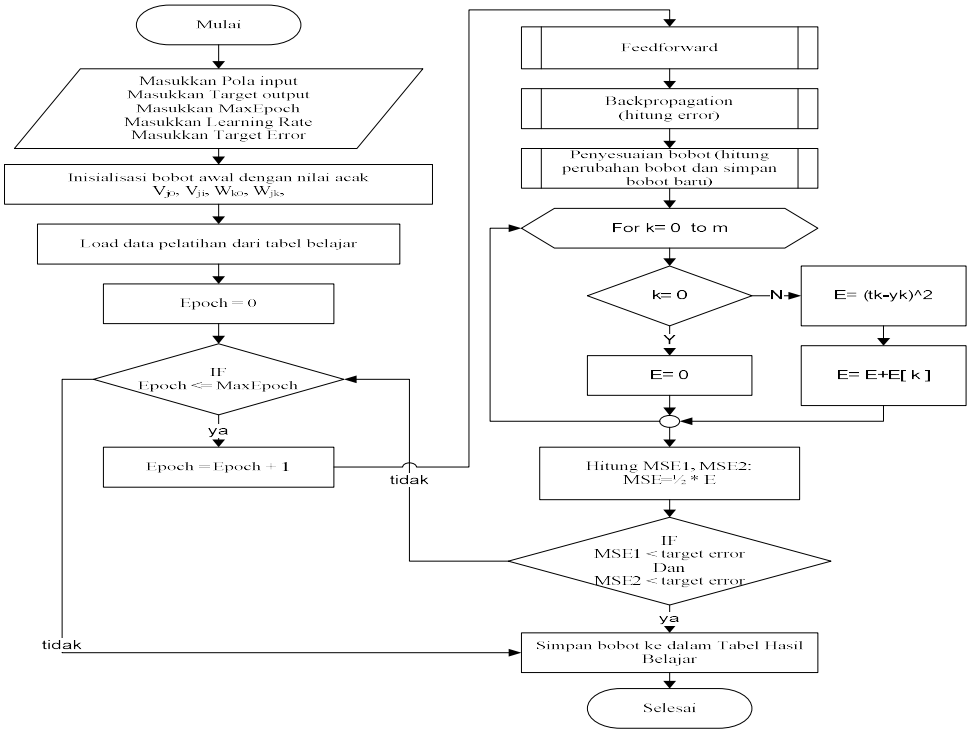
Proses pengujian bertujuan untuk menguji jaringan apakah jaringan dapat memberikan respon yang benar terhadap data baru berdasarkan pelatihan yang telah diberikan, algoritma pengujian dijelaskan pada gambar 3.Proses pengujian diawali dengan membuka tabel hasil belajar, yang berisi nama variabel bobot unit masukan dan nama variabel bobot unit tersembunyi beserta nilai bobotnya. Selanjutnya masukkan data pengujian sebagai sinyal *input*-an x_i. Pada proses pengujian, hanya dijalankan fase feedforward dari algoritma pelatihan. Setiap unit masukan Xi mengirimkan sinyal xi ke semua unit tersembunyi Zj yang terhubung dengannya. Setiap unit tersembunyi kemudian menjumlahkan setiap sinyal input terboboti yang masuk padanya. Hasilnya digunakan unuk menghitung sinyal keluaran Zj dengan menggunakan fungsi aktifasinya, kemudian mengirimkan ke setiap unit keluaran Yk. Pada setiap unit keluaran menjumlahkan setiap sinyal input terboboti yang masuk padanya. Hasilnya digunakan untuk menghitung sinyal keluaran yk dengan menggunakan fungsi aktifitasnya.Selanjutnya sinyal keluaran yk tersebut dibandingkan dengan nilai target yang telah ditentukan sebelumnya untuk mendapatkan output dari proses pengujian.



Gambar 1. *Flowchart Sistem Aplikasi Diagnosa Kerusakan PC*



Gambar 2. *Flowchart Algoritma Pengujian*

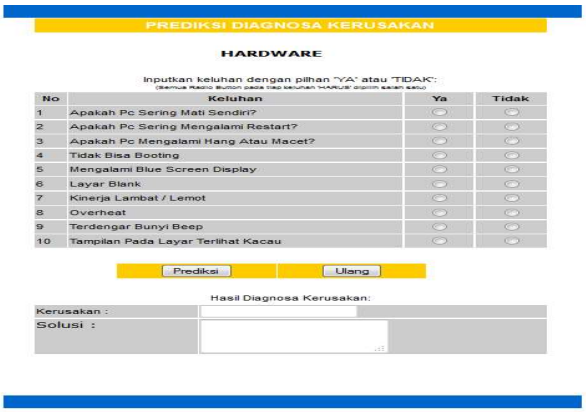


Gambar 3. *Flowchart Algoritma Pelatihan*

4. Hasil dan Pembahasan

Hasil pelatihan tersimpan pada tabel belajar 1, data ini digunakan sebagai input pola pelatihan JST untuk mengenali pola kerusakan yang mendekati pola target yang sudah ditentukan. Untuk melatih JST, tenaga ahli dapat membuka *submenu* Pelatihan yang terdapat pada menu Diagnosa *Hardware*. *Submenu* ini menampilkan form 'Pelatihan JST Diagnosa Kerusakan *Hardware'*. Dalam form ini menampilkan tabel pola target kerusakan dan tiga *textbox* variabel data pelatihan yaitu, Jumlah Max *Epoch* (diisikan nilai berupa angka diantara *range* 2-100.000), Laju Pemahaman (diisi nilai berupa angka pecahan kecil diantara *range* 0-1), dan Target Error (diisikan nilai berupa angka pecahan kecil diantara 0-1). Setelah mengisikan semua variabel data pelatihan, kemudian Tenaga ahli dapat melatih JST dengan meng-klik tombol 'Latih'. Untuk melihat hasil pelatihan dengan meng-klik tombol 'Hasil', dan tombol '*Refresh'* untuk menyegarkan tabel pola target yang ditampilkan. Hasil pelatihan akan disimpan pada tabel hasil belajar1. Form Pelatihan JST Diagnosa Kerusakan *Hardware* ini dapat dilihat pada Gambar 9.

Gambar 9.*Halaman Pelatihan JST*



Gambar 10.*Halaman Prediksi*

Setelah melakukan pelatihan, Tenaga Ahli dapat menguji apakah JST sudah mampu mengenali pola yang telah dilatihkan. Untuk mengujinya dapat dimulai dengan membuka *submenu* 'Prediksi' pada menu Diagnosa *Hardware*. Kemudian akan tampil form 'Prediksi Diagnosa Kerusakan *Hardware'*. Pada form ini terdapat 10 (sepuluh) pertanyaan keluhan kerusakan yang harus dipilih sesuai dengan pilihan 'Ya' atau 'Tidak'. Setelah itu, untuk mulai melihat hasil prediksi, user dapat meng-klik tombol 'Prediksi'. Pada *textfield* Hasil Diagnosa Kerusakan akan tampil hasil kerusakan sesuai keluhan yang telah dipilih oleh *user* beserta dengan solusi untuk mengatasi permasalahan kerusakan. Form ini dapat diakses oleh *user* dengan *level* Tenaga Ahli ataupun Teknisi. Tampilan form 'Prediksi Diagnosa Kerusakan *Hardware'* ini dapat dilihat seperti pada Gambar 10.

Tahap yang dilakukan setelah implementasi perangkat lunak pada model waterfall adalah tahap testing (pengujian) algoritma program. Untuk pengujian perangkat lunak pada skripsi ini telah disediakan 10 buah data pelatihan yaitu, 6 buah data untuk JST Diagnosa kerusakan *Hardware*, dan 4 buah data untuk JST Diagnosa kerusakan *Software*. Data-data ini didapatkan dari tenaga ahli, dimana tenaga ahli yang bersangkutan adalah teknisi yang sudah cukup lama bekerja dalam bidang komputer.

Pada JST Diagnosa kerusakan *Hardware* akan dilakukan beberapa pengujian, antara lain:

1. **Apakah JST dapat memprediksi kerusakan *Hardware* terhadap tingkat ketepatan prediksi mendekati 100%?**

   Pada percobaan ini akan digunakan *learning rate* = 0,1, jumlah epoch maksimal = 10.000 dan 40.000, serta *target error* = 0,1 hingga 0,00001. Pada tabel 1 dapat dilihat tabel hasil percobaan ini. Pada Tabel 1 menunjukkan bahwa dari 6 kali pelatihan JST Diagnosa kerusakan *Hardware* ini mencapai tingkat akurasi 99%. Dan dapat disimpulkan bahwa dengan penambahan MaxEpoch dari 10.000 menjadi 40.000 JST ini mampu mendekati tingkat akurasi 100% dengan berhenti pada iterasi ke 29.683, laju pemahaman (*learning rate*) 0,1 dan *Target error* 0,00001.

Tabel 1. *Data prediksi diagnosa Hardware ketepatan mendekati 100%*

No	Kerusakan	Max. Epoch	Epoch	L.R.	T.E.	MSE1	MSE2	Akurasi (%)
1	Power Supply	10.000	19	0,1	0,1	0,24144	0,09991	57
	Prosesor							57
	Motherboard							57
	RAM							56
	VGA							56
	Harddisk							57
2	Power Supply	10.000	90	0,1	0,01	0,23307	0,00973	86
	Prosesor							85
	Motherboard							85
	RAM							85
	VGA							85
	Harddisk							85
3	Power Supply	10.000	478	0,1	0,001	0,23672	0,00099	95
	Prosesor							94
	Motherboard							94
	RAM							95
	VGA							95
	Harddisk							94
4	Power Supply	10.000	3.255	0,1	0,0001	0,23238	9,9E-05	98
	Prosesor							98
	Motherboard							98
	RAM							98
	VGA							98
	Harddisk							98
5	Power Supply	10.000	10.000	0,1	0,00001	0,22687	4E-05	99
	Prosesor							99
	Motherboard							99
	RAM							99
	VGA							99
	Harddisk							99
6	Power Supply	40.000	29.683	0,1	0,00001	0,23047	1E-05	99
	Prosesor							99
	Motherboard							99
	RAM							99
	VGA							99
	Harddisk							99

Tabel 2.*Data percobaan efek penurunan Learning rate terhadap ketepatan prediksi*

No	LR.	Kerusakan	Max. Epoch	Epoch	T.E.	MSE1	MSE2	Akurasi (%)
1	0,8	Power Supply	10000	18	0,1	0,239	0,0995	56
		Prosesor						55
2	0,7	Power Supply	10000	31	0,1	0,3086	0,0965	57
		Prosesor						57
3	0,6	Power Supply	10000	24	0,1	0,2714	0,093	57
		Prosesor						56
4	0,5	Power Supply	10000	26	0,1	0,2788	0,098	57
		Prosesor						56
5	0,4	Power Supply	10000	24	0,1	0,2808	0,0988	57
		Prosesor						57
6	0,3	Power Supply	10000	13	0,1	0,189	0,0948	56
		Prosesor						57
7	0,2	Power Supply	10000	12	0,1	0,1847	0,0975	57
		Prosesor						56
8	0,1	Power Supply	10000	33	0,1	0,3247	0,0994	56
		Prosesor						57

2. **Apakah penurunan *learning rate* berpengaruh terhadap tingkat ketepatan prediksi?**
   Pada percobaan ini dilakukan menggunakan 2 data pola kerusakan, jumlah MaxEpoch = 10.000 dan *Target error* = 0,1. Percobaan dilakukan berulang-ulang dengan merubah *learning rate* dengan kisaran dari 0,8-0,1 dimana *learning rate* akan diturunkan dengan jarak 0,1.Jika diamati pada tabel 2, dapat dilihat bahwa setiap kali *learning rate* diturunkan maka yang terjadi pada MSE2 (akhir) dan Akurasi ketepatan berubah naik-turun secara tidak beraturan. Hal ini menunjukkan bahwa perubahan *learning rate* akan mempengaruhi cepat atau lambatnya JST dalam mengenali pola yang dilatihkan. Pada Tabel 4 menunjukkan bahwa dari 6 kali pelatihan JST Diagnosa kerusakan *Hardware* mencapai tingkat akurasi 99%. Dan dapat disimpulkan bahwa dengan penambahan MaxEpoch dari 10.000 menjadi 40.000 JST mampu mendekati akurasi 100% dengan berhenti pada iterasi ke 29.683, (*learning rate*) 0,1 dan *Target error* 0,00001. Pada pelatihan ke-5 dengan MaxEpoch 10.000, pelatihan JST menemui kegagalan. Dikarenakan untuk penurunan *Target error* menjadi 0,00001, membutuhkan iterasi lebih dari 10.000. Dan itu mengakibatkan MSE2 terpaksa berhenti dalam keadaan masih melebihi *Target error* yang ditetapkan.
3. **Apakah penurunan *Target Error* berpengaruh terhadap tingkat ketepatan prediksi?**
   Pada percobaan ini dilakukan dengan menggunakan 2 data pola kerusakan, jumlah MaxEpoch = 10.000 dan *learning rate* = 0,1. Percobaan akan dilakukan berulang-ulang dengan merubah *Target error* dengan kisaran dari 0,1 sampai 0,01 dimana *Target error* akan diturunkan dengan jarak 0,01.

Tabel 3. *Data percobaan efek penurunan Target error Software terhadap ketepatan prediksi Hardware*

No	T.E	Kerusakan	Max. Epoch	Epoch	L.R.	MSE1	MSE2	Akurasi (%)
1	0,1	Power Supply	10000	27	0,1	0,2814	0,09976	56
		Prosesor						56
2	0,09	Power Supply	10000	23	0,1	0,2589	0,09878	58
		Prosesor						58
3	0,08	Power Supply	10000	38	0,1	0,3306	0,07772	60
		Prosesor						61
4	0,07	Power Supply	10000	24	0,1	0,2145	0,06333	65
		Prosesor						64
5	0,06	Power Supply	10000	20	0,1	0,1769	0,05906	65
		Prosesor						66
6	0,05	Power Supply	10000	31	0,1	0,2385	0,04999	69
		Prosesor						69
7	0,04	Power Supply	10000	38	0,1	0,2586	0,03918	72
		Prosesor						72
8	0,03	Power Supply	10000	59	0,1	0,2763	0,02853	76
		Prosesor						76
9	0,02	Power Supply	10000	82	0,1	0,3023	0,0198	80
		Prosesor						80
10	0,01	Power Supply	10000	117	0,1	0,2716	0,00991	85
		Prosesor						85

Tabel 4. *Data prediksi diagnosa dengan ketepatan mendekati 100%*

No	Kerusakan	Max. Epoch	Epoch	L.R.	T.E.	MSE1	MSE2	Akurasi (%)
1	Bios	10.000	11	0,1	0,1	0,19068	0,09822	57
	OS							56
	Driver							56
	Aplikasi							56
2	Bios	10.000	89	0,1	0,01	0,25183	0,00991	85
	OS							86
	Driver							85
	Aplikasi							85
3	Bios	10.000	573	0,1	0,001	0,23899	0,00099	95
	OS							95
	Driver							95
	Aplikasi							95
4	Bios	10.000	3.689	0,1	0,0001	0,27692	1E-04	98
	OS							98
	Driver							98
	Aplikasi							98
5	Bios	10.000	10.000	0,1	0,00001	0,24286	4,1E-05	99
	OS							99
	Driver							99
	Aplikasi							99
6	Bios	40.000	29.430	0,1	0,00001	0,23089	1E-05	99
	OS							99
	Driver							99
	Aplikasi							99

Dapat dilihat pada tabel 3, bahwa semakin *Target error* diturunkan secara beraturan, maka MSE juga akan menurun secara beraturan. Dengan demikian, maka Akurasi ketepatan juga naik secara beraturan pula. Pada JST Diagnosa kerusakan *Software* dilakukan beberapa pengujian, yaitu:

1. **Apakah JST dapat memprediksi kerusakan *Software* terhadap tingkat ketepatan prediksi mendekati 100%?**
   Pada percobaan ini akan digunakan *learning rate* = 0,1, jumlah epoch maksimal = 10.000 dan 40.000, serta *Target error* = 0,1 hingga 0,00001. Pada tabel 4.4 dapat dilihat tabel hasil percobaan ini.
2. **Apakah penurunan *learning rate* berpengaruh terhadap tingkat ketepatan prediksi?**
   Pada percobaan ini dilakukan dengan menggunakan 2 data pola kerusakan, jumlah MaxEpoch = 10.000 dan *Target error* = 0,1. Percobaan akan dilakukan berulang-ulang dengan merubah *learning rate* dengan kisaran dari 0,8 sampai 0,1 dimana *learning rate* akan diturunkan dengan jarak 0,1. Pada tabel 5, dapat dilihat bahwa setiap kali *learning rate* diturunkan maka yang terjadi pada MSE2 (akhir) dan Akurasi ketepatan berubah naik-turun secara tidak beraturan. Hal ini menunjukkan bahwa perubahan *learning rate* (laju pemahaman) akan mempengaruhi cepat atau lambatnya JST dalam mengenali pola yang dilatihkan.
3. **Apakah penurunan *Target error* berpengaruh terhadap tingkat ketepatan prediksi?**
   Pada percobaan ini dilakukan dengan menggunakan 2 data pola kerusakan, jumlah MaxEpoch = 10.000 dan *learning rate* = 0,1. Percobaan akan dilakukan berulang-ulang dengan merubah *Target error* dengan kisaran dari 0,1 sampai 0,01 dimana *Target error* akan diturunkan dengan jarak 0,01. Dapat dilihat pada tabel 6, *Target error* diturunkan secara beraturan, maka MSE juga akan menurun secara beraturan.

Tabel 5 *Data percobaan efek penurunan Learning rate terhadap ketepatan prediksi*

No	L.R.	Kerusakan	Max. Epoch	Epoch	T.E.	MSE1	MSE2	Akurasi (%)
1	0,8	Bisa	10000	13	0,1	0,269218	0,09973	57
		OS						57
2	0,7	Bisa	10000	14	0,1	0,265688	0,09422	58
		OS						58
3	0,6	Bisa	10000	20	0,1	0,309475	0,09436	58
		OS						57
4	0,5	Bisa	10000	22	0,1	0,314466	0,09634	57
		OS						57
5	0,4	Bisa	10000	24	0,1	0,312289	0,09715	57
		OS						57
6	0,3	Bisa	10000	9	0,1	0,181085	0,08965	56
		OS						59
7	0,2	Bisa	10000	21	0,1	0,260818	0,0942	57
		OS						57
8	0,1	Bisa	10000	24	0,1	0,285922	0,09722	57
		OS						56

Tabel 6. *Data percobaan efek penurunan Target error terhadap ketepatan prediksi*

No	T.E	Kerusakan	Max. Epoch	Epoch	L.R.	MSE1	MSE2	Akurasi (%)
1	0,1	Bisa	10000	19	0,1	0,25706	0,097015	58
		OS						57
2	0,09	Bisa	10000	21	0,1	0,24228	0,087076	59
		OS						59
3	0,08	Bisa	10000	29	0,1	0,28277	0,078552	61
		OS						61
4	0,07	Bisa	10000	33	0,1	0,3083	0,060797	64
		OS						65
5	0,06	Bisa	10000	30	0,1	0,278	0,058208	65
		OS						65
6	0,05	Bisa	10000	40	0,1	0,27795	0,04817	69
		OS						69
7	0,04	Bisa	10000	40	0,1	0,26227	0,0387	72
		OS						72
8	0,03	Bisa	10000	58	0,1	0,31281	0,029851	75
		OS						75
9	0,02	Bisa	10000	63	0,1	0,23661	0,018921	80
		OS						80
10	0,01	Bisa	10000	109	0,1	0,28365	0,009638	85
		OS						86

Dengan demikian, akurasi ketepatan juga naik secara beraturan pula. Analisis pengujian sistem ini dilakukan beberapa pengujian. Dimana aplikasi yang diusulkan harus dapat memberikan solusi atau perkembangan yang lebih baik dari sistem yang sebelumnya. Pengujian dilakukan dengan membandingkan tingkat akurasi pendiagnosaan yaitu, pendiagnosaan kerusakan komputer secara manual (kondisi sebelum ada sistem yang diusulkan) dengan pendiagnosaan kerusakan komputer menggunakan aplikasi JST (sistem yang diusulkan). Hasil pengujian diagnosa kerusakan dengan cara manual dilihat pada tabel 7.

Pada tabel 7 dapat dilihat bahwa dengan melakukan prediksi kerusakan sebanyak 20 (dua puluh) kali pada pengujian secara manual, telah diketahui hasil diagnosa benar sebanyak 15 (lima belas). Sedangkan hasil diagnosa yang salah sebanyak 5 (lima) kali. Dari hasil tersebut diketahui prosentase akurasi dapat dihitung dengan persamaan 18:

Tabel 7. *Pengujian Diagnosa Kerusakan Dengan Secara Manual*

No	Tanggal	Keluhan	Prediksi Kerusakan		Hasil
			Awal	Akhir	
1	04/09/14	X3 X4 X8	Hardisk	Mainboard	Salah
2	04/09/14	X1 X2	Power Suply	Power Suply	Benar
3	04/09/14	X4 X10	Vga	Vga	Benar
4	04/09/14	X1 X7	Power Suply	Power Suply	Benar
5	04/09/14	X6 X8	Ram	Vga	Salah
6	05/09/14	X1 X2 X7	Power Suply	Power Suply	Benar
7	05/09/14	X1 X5 X8	Mainboard	Prosessor	Salah
8	05/09/14	X2 X5	Ram	Ram	Benar
9	05/09/14	X3 X5	Hardisk	Hardisk	Benar
10	06/09/14	X1 X4 X8	Mainboard	Mainboard	Benar
11	06/09/14	X1 X2 X8	Bios	Bios	Benar
12	06/09/14	X6 X10	Aplikasi	Aplikasi	Benar
13	06/09/14	X4 X5 X9	Os	Os	Benar
14	06/09/14	X9 X10	Aplikasi	Aplikasi	Benar
15	06/09/14	X5 X7 X8	Driver	Driver	Benar
16	06/09/14	X2 X3 X10	Aplikasi	Os	Salah
17	08/09/14	X3 X5 X7	Os	Driver	Salah
18	08/09/14	X9 X10	Aplikasi	Aplikasi	Benar
19	08/09/14	X2 X5	Os	Os	Benar
20	08/09/14	X1 X8	Bios	Bios	Benar

Tabel 8. *Pengujian Diagnosa Kerusakan Aplikasi JST*

No	Tanggal	Keluhan	Prediksi Kerusakan		Hasil
			Awal	Akhir	
1	04/09/14	X3 X4 X8	Mainboard	Mainboard	Benar
2	04/09/14	X1 X2	Power Suply	Power Suply	Benar
3	04/09/14	X4 X10	Vga	Vga	Benar
4	04/09/14	X1 X7	Power Suply	Power Suply	Benar
5	04/09/14	X6 X8	Vga	Vga	Benar
6	05/09/14	X1 X2 X7	Power Suply	Power Suply	Benar
7	05/09/14	X1 X5 X8	Prosessor	Prosessor	Benar
8	05/09/14	X2 X5	Ram	Ram	Benar
9	05/09/14	X3 X5	Hardisk	Hardisk	Benar
10	06/09/14	X1 X4 X8	Mainboard	Mainboard	Benar
11	06/09/14	X1 X2 X8	Bios	Bios	Benar
12	06/09/14	X6 X10	Aplikasi	Aplikasi	Benar
13	06/09/14	X4 X5 X9	Os	Os	Benar
14	06/09/14	X9 X10	Aplikasi	Aplikasi	Benar
15	06/09/14	X5 X7 X8	Driver	Driver	Benar
16	06/09/14	X2 X3 X10	Os	Os	Benar
17	08/09/14	X3 X5 X7	Driver	Driver	Benar
18	08/09/14	X9 X10	Aplikasi	Aplikasi	Benar
19	08/09/14	X2 X5	Os	Os	Benar
20	08/09/14	X1 X8	Bios	Bios	Benar

$$\text{Akurasi Prediksi (\%)} = \frac{\text{Jumlah Hasil Benar}}{\text{Jumlah Prediksi}} \text{ X } 100 \quad \text{..... (18)}$$

Hasil Perhitungan:

$$\text{Akurasi Prediksi (\%)} = \frac{15}{20} \text{ X } 100$$
$$= 75$$

Hasil pengujian 221 diagnose kerusakan dengan menggunakan aplikasi JST dapat dilihat pada tabel 2. Pada tabel 8 dapat dilihat bahwa dengan melakukan 221 prediksi kerusakan sebanyak 20 (dua puluh) kali pada pengujian dengan menggunakan aplikasi JST, telah diketahui hasil 221 diagnose benar sebanyak 20 (dua puluh). Sedangkan untuk hasil prediksi yang salah tidak terjadi pada pengujian aplikasi ini. Dari hasil tersebut diketahui prosentase akurasi dihitung dengan persamaan 19:

$$\text{Akurasi Prediksi (\%)} = \frac{\text{Jumlah Hasil Benar}}{\text{Jumlah Prediksi}} \text{ X } 100 \quad \text{..... (19)}$$

Hasil Perhitungan:

$$\text{Akurasi Prediksi (\%)} = \frac{20}{20} \text{ X } 100$$
$$= 100$$

Dengan hasil perhitungan akurasi prediksi dari masing-masing pengujian tersebut, dapat diketahui bahwa diagnosa kerusakan komputer dengan aplikasi JST mencapai tingkat akurasi 100%. Sedangkan jika dilakukan pendiagnosaan secara manual, diketahui hanya dapat mencapai tingkat akurasi 75%.

5. DaftarPustaka

[1] Kristanto, Andri.(2004). Jaringan Saraf Tiruan, Konsep Dasar, Algoritma Dan Aplikasinya. Yogyakarta : Gava Media.

[2] Fausett, L. (1994). Fundamentals of Neural Network: Architectures, Algorithms, and Applications. New Jersey : Prentice-Hall,Inc.

[3] Haykin, Simon.(1999).Neural Networks: A Comprehensive foundation. Canada : Pearson Education,Inc.

[4] Kusumadewi, Sri.(2002).Membangun Jaringan Saraf Tiruan Menggunakan Matlab dan Excel Link. Yogyakarta : Graha Ilmu.

[5] Nguyen, D. dan Widrow, B., (1989). The truck backer-upper: An Example of self learning in neural networks, International Joint Conference on neural Networks, Vol. 2, pp.357 – 363, Washington, DC.

[6] Puspitaningrum, Dyah. (2006). Pengantar Jaringan Saraf Tiruan. Yo