

Bab 2

Intrusion Detection System

2.1 Pengertian Intrusion Detection System

Intrusion detection atau pendeteksi gangguan memang sejak lama sudah menjadi bagian dari penelitian para pakar teknologi informasi, dimana diawali sekitar tahun 1980 atas publikasi seorang pengamat keamanan komputer yang bernama *John Anderson* .

Intrusion Detection System (IDS) merupakan suatu sistem yang mampu melakukan pendeteksian terhadap suatu gangguan, yang berupa penyerangan. Adapun tipe pendeteksian gangguan ini terbagi menjadi dua bagian, yaitu pertama Host Based Attack Detection (HBAD) atau pendeteksian gangguan yang dilakukan terhadap suatu komputer tempat disimpannya aplikasi ini. Kedua adalah Network based Attack Detection (NBAD), biasa dikatakan sebagai pendeteksian yang dilakukan terhadap suatu kondisi dengan lingkup jaringan, misalnya saja gangguan yang dilakukan oleh aplikasi paket *sniffer*.

Sistem yang berbasis IDS hanya mampu mendeteksi saja namun kelanjutan aksi yang akan dilakukan terhadap suatu gangguan tergantung pada aplikasi lainnya, misalnya saja *Firewall*, dengan menggunakan perangkat ini aksi akan lebih lengkap dan berkelanjutan, sehingga dapat terhindar dari setiap gangguan.

Umumnya aplikasi IDS mampu memberikan suatu indikasi apabila adanya gangguan untuk kemudian akan disampaikan ke administrator agar dapat dilakukan aksi terhadap gangguan tersebut.

2.2 Klasifikasi IDS

Berdasarkan sistem penganalisaan terhadap suatu gangguan, IDS memiliki kategori sebagai berikut, [1,2,3]:

- *Misuse Detection Model*, merupakan sistem deteksi yang melihat suatu aktivitas yang secara jelas dianggap sebagai *Pattern Signature* suatu gangguan dan biasanya deteksi ini cenderung untuk kondisi internal sistem.

- *Anomaly Detection Model*, merupakan sistem deteksi yang cenderung melihat suatu gangguan karena suatu aktifitas yang dianggap sebagai kondisi yang tidak normal dan biasanya untuk kondisi yang disebabkan oleh eksternal sistem.
- Umumnya aplikasi IDS menggunakan metoda *Misuse Detection*, dimana pola-pola gangguan selalu di *update* oleh pihak *vendor* pembuat aplikasi tersebut. bila dibandingkan dengan *Anomaly Detection Model*, metode ini mampu melakukan analisa tanpa melihat secara spesifik terhadap pola suatu gangguan dan terkadang melakukan suatu kesalahan akibat sensitifitas yang ditimbulkan.

2.3 Sistem Kerja IDS

Sistem IDS bekerja secara keseluruhan dari aktivitas suatu jaringan, yang mana sistem ini kemudian menyimpan informasi serta membandingkan secara paket dengan beberapa jumlah pola yang diketahui sebagai bentuk serangan, sebagai contoh, intruksi SYN yang dilakukan secara terus menerus dengan tanpa menghiraukan kondisi balasan dari host sumber, maka akan dijadikan suatu catatan atau indikasi awal oleh Sistem IDS sebagai kondisi yang tidak normal. "Sistem IDS yang bisa dikatakan baik adalah Sistem IDS yang mampu menyimpan jenis atau bentuk serangan di dalam suatu database, dengan perkiraan jumlah lebih dari 100 jenis[1]. Bentuk reaksi yang diberikan oleh Sistem IDS sangatlah bervariasi, bergantung pada bentuk serta konfigurasi Sistem IDS itu sendiri, biasanya yang akan dijadikan sebagai suatu catatan adalah berupa *log* kejadian-kejadian yang dianggap mencurigakan atau dikatakan kondisi yang tidak normal. Paket-paket yang sifatnya masih sulit untuk diterka oleh sistem biasanya akan tetap disimpan sebagai bahan pertimbangan secara manual yang dilakukan oleh *Administrator* suatu jaringan. Namun untuk Sistem IDS yang sudah cukup baik, biasanya telah diintegrasikan dengan *Router* atau aplikasi *Firewall* untuk melakukan blokir terhadap penyerangan suatu *Host*. Secara mendasar, Sistem IDS terdiri dari sebuah *engine* dan *console*, *engine* bertugas untuk memperoleh serta menganalisa suatu kejadian pada jaringan, sedangkan *console*, bertugas untuk mengelola *engine* dan

menampilkan hasil laporan atas kejadian yang diterima. Secara logika berarti kedua sistem ini bekerja secara serial atau berurutan, namun tidak menutup kemungkinan sehingga kedua sistem ini bekerja secara paralel, dengan konsekuensi maka sistem ini harus dibuat secara terpisah, sehingga dapat melakukan pengawasan sekaligus memberikan hasilnya yang berupa laporan. Dengan demikian maka akan dibutuhkan sistem komputasi yang besar [1,2], berdasarkan rekomendasi, komputer yang digunakan harus memiliki persyaratan sebagai berikut :

- Ram 128 Mb
- Intel Pentium II atau jenis yang setingkat
- Minimal *space disk* 100MB

Namun spesifikasi diatas merupakan kondisi normal, dengan arti bahwa penerimaan sistem terhadap kejadian pada jaringan tidak sibuk, sehingga akan membutuhkan spesifikasi yang lebih besar lagi bila lalu lintas pergerakan pada jaringan dapat dikatakan padat atau sibuk karena hal ini menyangkut permasalahan pada sistem *database* yang ada.

Batasan suatu Sistem IDS

Sistem kerja suatu IDS bersifat *real-time*, sehingga untuk melakukan suatu aksi dapat dikatakan lambat, karena aksi baru dilakukan setelah kejadian itu muncul, sebagai contoh pada jenis serangan DoS, atau Denial of Service, dimana sistem ini merupakan sistem penyerangan yang sudah umum dilakukan oleh para *hacker*. Setelah kejadian ini, IDS baru akan melakukan aksinya, namun sistem sudah dipengaruhi oleh serangan tersebut secara meluas. Begitupun untuk jenis serangan yang dinamakan *Teardrop attack*, kronologis serangan ini berupa sistem penyerangan yang membuat kondisi suatu sistem sangat sibuk atau dinamakan *Buffer overflow*, dimana sistem akan mengalami kondisi yang disebut crash, dan sebelum Sistem IDS mengambil tindakan kemungkinan besar sistem sudah *shot down*.

Pada tahun 1998, akhirnya para peneliti membuat kesimpulan bahwa Sistem IDS adalah sistem yang sifatnya *Vulnerability*, atau mudah terkena serangan. Suatu kesimpulan mengatakan bahwa para penyerang atau *intruder* mengetahui kelemahan pada Sistem IDS, yaitu sebagai contoh

Sistem IDS mengetahui bahwa PHF CGI sebagai suatu bentuk serangan melalui HTTP, namun saat hacker merubah keterangan *string PHF* menjadi PHOOF, maka hal ini akan dilewatkan oleh Sistem IDS, tetapi sistem penerima akan mendeteksi hal ini sebagai PHF, sehingga sistem akan terinfeksi tanpa adanya suatu hambatan[2].

2.4 Karakteristik IDS

Aplikasi sistem pendeteksi gangguan cenderung berbeda-beda dan memiliki fokus yang berlainan karena masing-masing lebih merujuk pada tujuan aplikasi yang akan ditempatkannya, dibawah ini hasil pengamatan yang dilakukan oleh *A Jackson*, seorang peneliti dari Los Alamos National Laboratory yang menyimpulkan karakteristik kebanyakan dari suatu produk IDS adalah seperti dibawah ini[2] :

➤ ***Suitability***

Aplikasi IDS yang cenderung memfokuskan berdasarkan skema manajemen dan arsitektur jaringan yang dihadapkannya.

➤ ***Flexibility***

Aplikasi IDS yang mampu beradaptasi dengan spesifikasi jaringan yang akan dideteksi oleh aplikasi tersebut.

➤ ***Protection***

Aplikasi IDS yang secara ketat memproteksi gangguan yang sifatnya utama dan berbahaya.

➤ ***Interoperability***

Aplikasi IDS yang secara umum mampu beroperasi secara baik dengan perangkat-perangkat keamanan jaringan serta manajemen jaringan lainnya.

➤ ***Comprehensiveness***

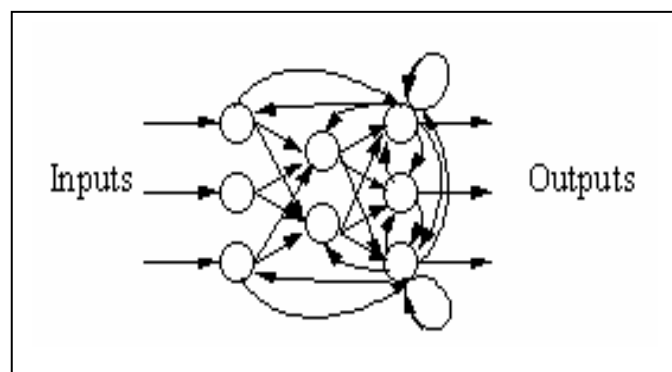
Kelengkapan yang dimiliki oleh aplikasi IDS ini mampu melakukan sistem pendeteksian secara menyeluruh seperti pemblokiran semua yang berbentuk *Java Applet*, memonitor isi dari suatu email serta dapat memblokir *address url* secara spesifik.

Keuntungan penggunaan Neural Network

- Perangkat yang mampu untuk mengenali suatu objek secara non-linier.
- Mempermudah pemetaan input menjadi suatu hasil tanpa mengetahui proses sebenarnya.
- Mampu melakukan pengadaptasian terhadap pengenalan suatu objek
- Perangkat yang memiliki toleransi terhadap suatu kesalahan dalam pengenalan suatu objek.
- Neural Network mampu diimplementasikan pada suatu *Hardware* atau perangkat keras.
- Perangkat yang mampu diimplementasikan secara parallel.

3.2 Arsitektur Neural Network

Bentuk dasar arsitektur suatu neural network adalah sebagai berikut :



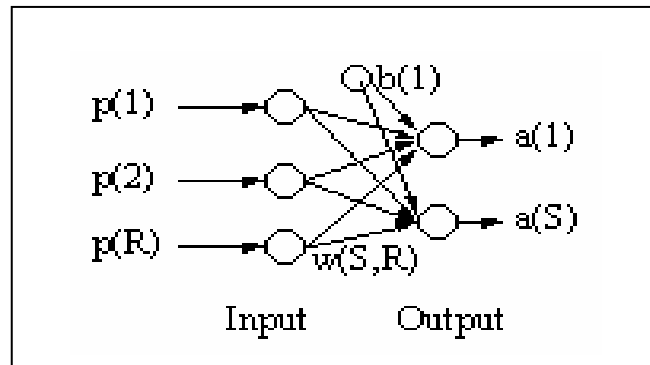
Gambar 2. Arsitektur dasar Neural Network

Secara umum, terdapat tiga jenis neural network [1] yang sering digunakan berdasarkan jenis *network*-nya, yaitu :

- Single-Layer Neural Network
- Multilayer Perceptron Neural Network
- Recurrent Neural Networks

3.2.1. Single-Layer Neural Network

Neural network jenis ini memiliki koneksi pada inputnya secara langsung ke jaringan output.

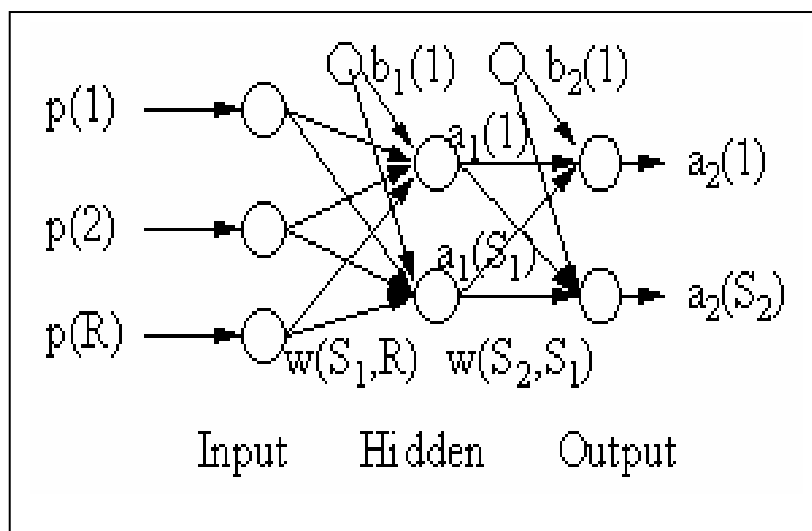


Gambar 3. Single-layer Neural Network

Jenis neural network ini sangatlah terbatas, hanya digunakan pada kasus-kasus yang sederhana.

3.2.2. Multilayer Perceptron Neural Network

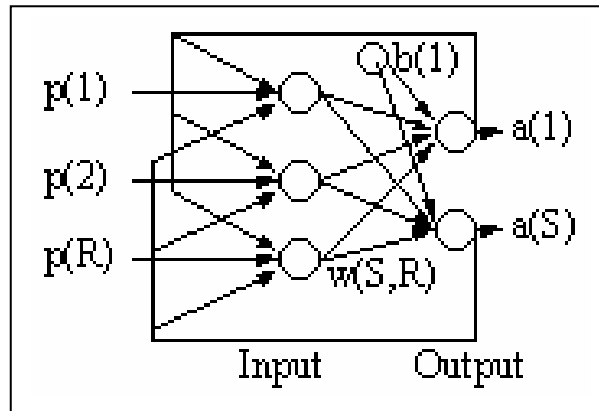
Jenis neural network ini memiliki layer yang dinamakan "hidden", ditengah layer input dan output. Hidden ini bersifat variable, dapat digunakan lebih dari satu hidden layer.



Gambar 4. Multilayer Perceptron Neural Network

3.2.3. Recurrent Neural Network

Neural network jenis ini memiliki ciri, yaitu adanya koneksi umpan balik dari output ke input.



Gambar 5. Recurrent Network

Kelemahan dari jenis ini adalah Time Delay akibat proses umpan balik dari *output* ke titik *input*.

3.3 Proses Pembelajaran pada Neural Network

Proses pembelajaran merupakan suatu metoda untuk proses pengenalan suatu objek yang sifatnya kontinuitas yang selalu direspon secara berbeda dari setiap proses pembelajaran tersebut. Tujuan dari pembelajaran ini sebenarnya untuk memperkecil tingkat suatu error dalam pengenalan suatu objek. Secara mendasar, neural network memiliki sistem pembelajaran yang terdiri atas beberapa jenis berikut :

- Supervised Learning
- Unsupervised Learning

3.3.1. Supervised Learning

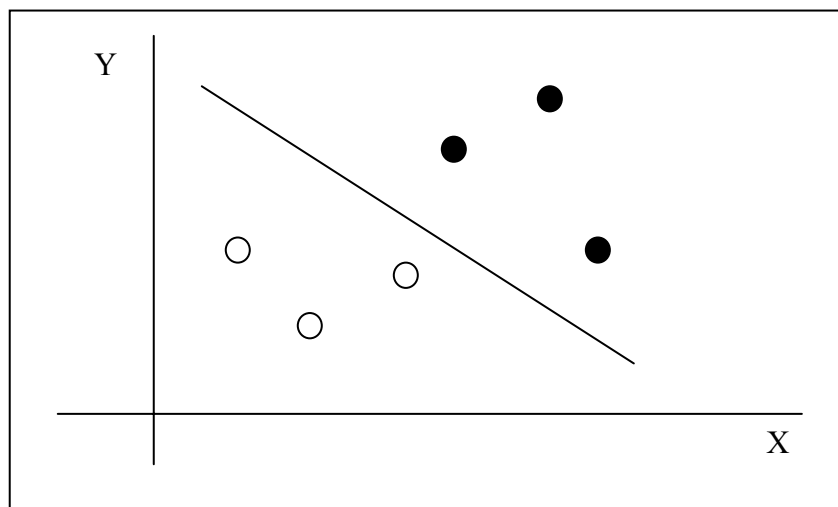
Sistem pembelajaran pada metoda Supervised learning adalah sistem pembelajaran yang mana, setiap pengetahuan yang akan diberikan kepada sistem, pada awalnya diberikan suatu acuan untuk memetakan suatu masukan menjadi suatu keluaran yang diinginkan. Proses pembelajaran ini akan terus dilakukan selama kondisi *error* atau kondisi yang diinginkan belum tercapai. Adapun setiap perolehan error akan dikalkulasikan untuk setiap pemrosesan hingga data atau nilai yang diinginkan telah tercapai.

3.3.2. Unsupervised Learning

Sistem pembelajaran pada *neural network*, yang mana sistem ini memberikan sepenuhnya pada hasil komputasi dari setiap pemrosesan, sehingga pada sistem ini tidak membutuhkan adanya acuan awal agar perolehan nilai dapat dicapai. Meskipun secara mendasar, proses ini tetap mengkalkulasikan setiap langkah pada setiap kesalahannya dengan mengkalkulasikan setiap nilai *weight* yang didapat

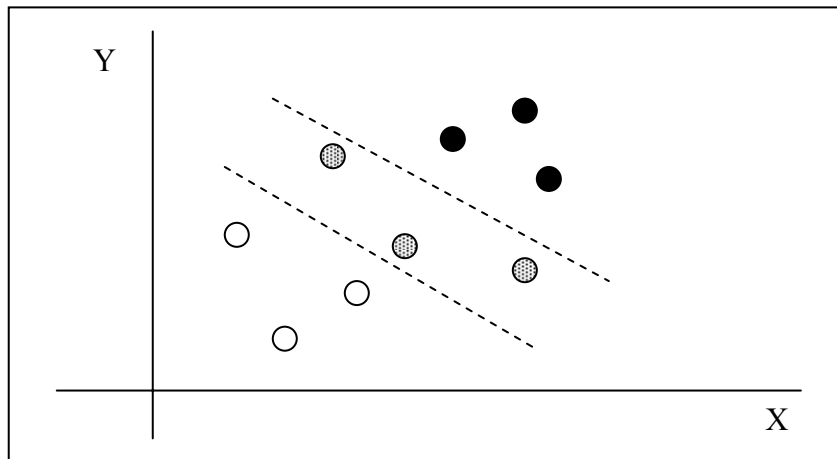
3.4. Mekanisme Kerja Multilayer Perceptron Neural Network

Sesuai dengan karakteristik *neural network*, pada dasarnya Multilayer Perceptron memiliki kecenderungan yang sama dengan jenis *neural network* lainnya, namun setiap jenis memiliki karakteristik masing-masing, seperti halnya *Single layer Neural Network*, biasanya hanya digunakan untuk memberikan solusi yang sifatnya hanya sederhana saja, sebagai contoh berikut ini.



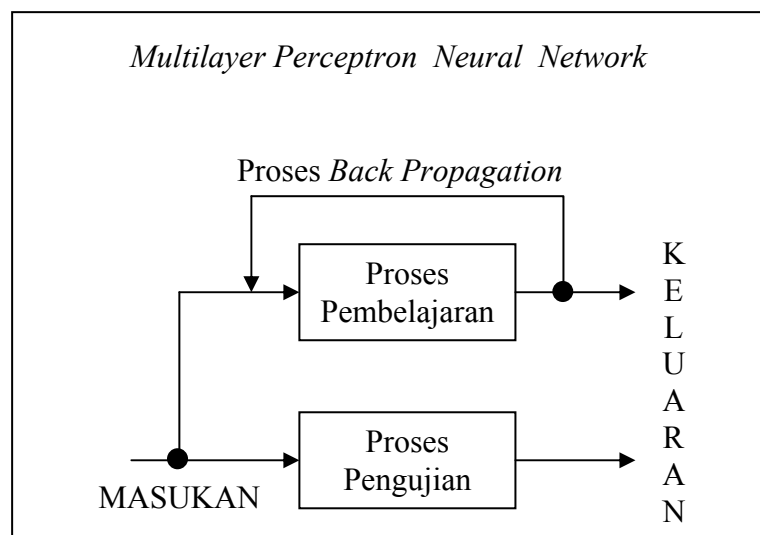
Gambar 6. Penggunaan *Single Layer Neural Network*

Gambar diatas menunjukkan bahwa *single layer neural network* digunakan untuk menganalisa dua bagian yang berbeda saja, yaitu agar dapat mengetahui posisi lingkaran hitam dan lingkaran yang berwarna putih. Lain halnya dengan dengan kondisi pada gambar dibawah ini.



Gambar 7. Penggunaan *Multilayer Perceptron Neural Network*

Pada Gambar 7, menunjukkan bahwa dengan karakteristik *Single Layer Neural Network* yang hanya mampu mendeteksi dua daerah saja membuat kasus ini sulit untuk dapat diselesaikan. *Multilayer Perceptron Neural Network* adalah jenis neural network yang memiliki kemampuan untuk mendeteksi atau melakukan analisa untuk permasalahan yang sifatnya cukup atau bahkan sangat kompleks, seperti pada masalah Pemrosesan Bahasa, Pengenalan suatu Pola serta Pemrosesan suatu *Image* atau gambar. Adapun Proses yang terjadi Pada *Multilayer Perceptron Neural Network*, adalah sebagai berikut :



Gambar 8. Proses *Multilayer Perceptron Neural network*

target adalah pemilihan metoda *Back Propagation*, yang akan dijelaskan pada sub bab berikut.

3.4.1 Back Propagation Multilayer Perceptron Neural Network

Back Propagation adalah istilah dalam penggunaan metoda MLP-NN untuk melakukan proses *update* pada nilai vektor *weight* dan *bias*.

Adapun bentuk metoda *weight* ini memiliki beberapa macam, antara lain adalah sebagai berikut[5].

- Gradient Descent Back Propagation (GD)
- Gradient Descent Back Propagation dengan Momentum (GDM)
- Variable Learning Rate Back Propagation dengan Momentum (GDX)
- Conjugate Gradient Back Propagation (CGP)

3.4.1.1 Gradient Descent Back Propagation (GD)

Metoda ini merupakan proses *update* untuk nilai *weight* dan *bias* dengan arah propagasi fungsinya selalu menurunkan nilai *weight* sebelumnya.

Bentuk vektor *weight* tersebut berlaku seperti metoda berikut.

$$W_{k+1} = W_k - \alpha \cdot g_k \quad (1)$$

Dimana α , merupakan *Learning rate* serta g , merupakan *gradient* yang berhubungan dengan nilai *error* yang diakibatkan oleh *weight* tersebut.

3.4.1.2 Gradient Descent Back Propagation dengan Momentum

Penggunaan Momentum pada Metoda ini memberikan nilai tambah dimana hasil *update* diharapkan tidak berhenti pada kondisi yang dinamakan "Local Minimum", sehingga proses penelusuran hingga mencapai nilai minimum yang paling puncak dalam pengertian nilai *error* yang paling kecil dapat tercapai. Adapun bentuk metoda penggunaan Momentum ini adalah seperti dibawah ini.

$$W_{k+1} = W_k - \alpha \cdot g_k + \mu \cdot W_{k-1} \quad (2)$$

3.4.1.3 Variabel Learning Rate Back Propagation dengan Momentum

Penggunaan metoda ini bertujuan untuk mempercepat waktu penyelesaian sehingga proses mendapatkan nilai error yang paling kecil dapat tercapai dengan cepat serta penelusuran yang lebih singkat. Sebaliknya jika nilai yang digunakan dalam praktisnya maka hasil yang didapatkan biasanya akan memperlambat proses penelusuran nilai *error* yang paling kecil.

Dalam penggunaan metoda ini para peneliti biasanya menggunakan cara memperbesar nilai dari *Variabel Learning Rate* saat hasil yang dicapai jauh dari target, dan sebaliknya saat hasil yang dicapai dekat dengan nilai target. Secara perhitungan metoda ini memang tidak begitu jauh dari metoda yang telah dijelaskan sebelumnya, namun perbedaannya adalah seperti dibawah ini.

$$W_{k+1} = W_k - \alpha_{k+1} \cdot g_k + \mu \cdot W_{k-1} \quad (3)$$

$$\alpha_{k+1} = \beta \cdot \alpha_k \quad (4)$$

$$\beta = \begin{cases} 0.7 & \text{jika nilai } new\ error > 1.04 \text{ (old error)} \\ 1.05 & \text{jika nilai } new\ error < 1.04 \text{ (old error)} \end{cases} \quad (5)$$

$$\beta = \begin{cases} 0.7 & \text{jika nilai } new\ error > 1.04 \text{ (old error)} \\ 1.05 & \text{jika nilai } new\ error < 1.04 \text{ (old error)} \end{cases} \quad (6)$$

3.4.1.4 Conjugate Gradient Back Propagation (CGX)

Conjugate Gradient Back Propagation memiliki perbedaan dibandingkan dengan metoda GD yaitu pada saat melakukan proses *update*, dimana untuk metoda GD proses tersebut dilakukan setiap penggunaan rumus sedangkan pada proses CGX, *update* dilakukan setiap iterasi dilakukan. Berikut ini merupakan proses *update* nilai *weight*.

$$W_{k+1} = W_k + \alpha \cdot p_k \quad (7)$$

Dimana : $p_k = -g_k + \beta_k \cdot p_{k-1} \quad (8)$

$$\beta = \frac{\Delta g_{k-1}^T \cdot g_k}{g_{k-1}^T \cdot g_{k-1}} \quad (9)$$

dan
$$\Delta \mathbf{g}_{k-1}^T = \mathbf{g}_k^T - \mathbf{g}_{k-1}^T \quad (10)$$

3.4.1.5 Quasi- Newton Back Propagation (BPGS)

Metoda *Newton* ini merupakan improvisasi dari metoda CGX, dimana pencapaian nilai konfigurasi dapat dilakukan lebih cepat.

Metoda yang digunakan adalah sebagai berikut :

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mathbf{A}_k^{-1} \cdot \mathbf{g}_k \quad (11)$$

\mathbf{A}_k merupakan Hessian Matrix untuk nilai *wieght* dan *Bias*.

Tabel 1. Bentuk Numerik Masukan dan Keluaran

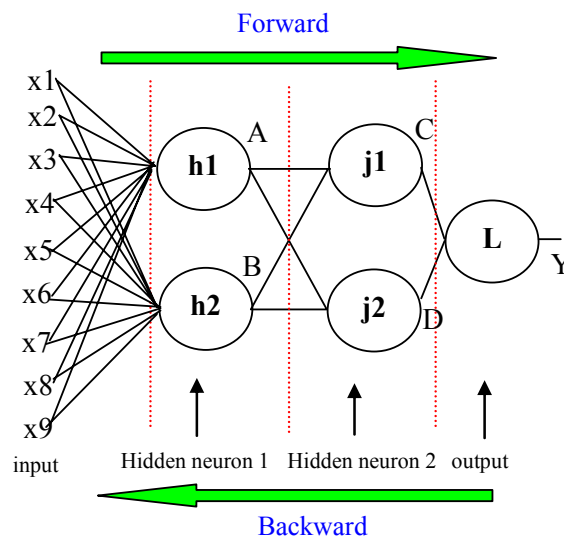
Protocol ID	Source Port	Destination Port	Source Address	Destination Address	ICMP Type ID	ICMP Code ID	Raw Data Length	Data ID	Attack
0	2314	80	1573638018	-1580478590	1	1	401	3758	0
0	1611	6101	801886082	-926167166	1	1	0	2633	1

Tahapan yang ketiga adalah mengkonversikan masukan-keluaran pada Tabel 1 menjadi nilai-nilai sebagai masukan-keluaran seperti pada Tabel 2 dibawah.

Tabel 2. Nilai-nilai Numerik Masukan dan Keluaran

0,2314,80,1573638018,-1580478590,1,1,401,3758,0 0,1611,6101,801886082,-926167166,1,1,0,2633,1
--

Sesuai dengan kriteria masukan pada proses pengumpulan data maka akan dibentuk tiga layer *Neural Network* dengan banyaknya masukan sejumlah sembilan kategori, maka bentuk jaringan yang dibuat adalah seperti pada Gambar 11 berikut.



Gambar 11. Rancangan *Multilayer Perceptron Neural Network*

4.4 Proses Perhitungan saat Pengujian

Proses ini dinamakan proses "forward", karena alur perhitungan dilakukan dari layer awal hingga ke bagian keluaran[6].

Hidden neuron 1

$$\mathbf{h1} = W_{h1x1} \cdot x1 + W_{h1x2} \cdot x2 + W_{h1x3} \cdot x3 + W_{h1x4} \cdot x4 + W_{h1x5} \cdot x5 + W_{h1x6} \cdot x6 + W_{h1x7} \cdot x7 + W_{h1x8} \cdot x8 + W_{h1x9} \cdot x9 \quad (1)$$

$$\mathbf{h2} = W_{h2x1} \cdot x1 + W_{h2x2} \cdot x2 + W_{h2x3} \cdot x3 + W_{h2x4} \cdot x4 + W_{h2x5} \cdot x5 + W_{h2x6} \cdot x6 + W_{h2x7} \cdot x7 + W_{h2x8} \cdot x8 + W_{h2x9} \cdot x9 \quad (2)$$

$$\mathbf{A} = F(\mathbf{h1}) = 1 / (1 + \exp -h1.\beta) \quad (3)$$

$$\mathbf{B} = F(\mathbf{h2}) = 1 / (1 + \exp -h2.\beta) \quad (4)$$

Hidden neuron 2

$$\mathbf{j1} = W_{j1h1} \cdot \mathbf{A} + W_{j1h2} \cdot \mathbf{B} \quad (5)$$

$$\mathbf{j2} = W_{j2h1} \cdot \mathbf{A} + W_{j2h2} \cdot \mathbf{B} \quad (6)$$

$$\mathbf{C} = F(\mathbf{j1}) = 1 / (1 + \exp -j1.\beta) \quad (7)$$

$$\mathbf{D} = F(\mathbf{j2}) = 1 / (1 + \exp -j2.\beta) \quad (8)$$

Output

$$\mathbf{L} = W_{Lj1} \cdot \mathbf{C} + W_{Lj2} \cdot \mathbf{D} \quad (9)$$

$$\mathbf{Y} = F(\mathbf{L}) = 1, \text{ if } \mathbf{L} > 0 \quad (10)$$

$$0, \text{ if } \mathbf{L} \leq 0 \quad (11)$$

Setelah mendapatkan hasil keluaran (output), kemudian bila hasilnya tidak mencapai target maka perlu dilakukan proses pembelajaran kembali.

4.5 Proses Perhitungan saat Pembelajaran

Pada bagian ini, hampir sama dengan proses pengujian, namun yang membedakan adalah saat melakukan pembelajaran harus disertakan target yang semestinya (acuan), kemudian proses kalkulasi seperti perhitungan berikut[6].

$$\mathbf{EL} = \mathbf{TL} - \mathbf{YL} \quad (12)$$

\mathbf{TL} adalah nilai target yang ditentukan

\mathbf{YL} adalah nilai keluaran hasil kalkulasi

\mathbf{EL} adalah nilai *error* hasil kalkulasi antara \mathbf{YL} dan \mathbf{TL}

Namun sebelum melakukan perhitungan pada rumusan (12), maka perhitungan sebelumnya adalah sebagai berikut. Proses ini dinamakan sebagai proses "Backward", karena alur perhitungan berawal dari layer keluaran (Output) hingga ke layer masukan[6].

Output neuron

$$\Delta W_{Lj1} = \eta \cdot E_L \cdot Y' \cdot C \longrightarrow W_{Lj1(n+1)} = W_{Lj1} + \Delta W_{Lj1} \quad (13)$$

$$\Delta W_{Lj2} = \eta \cdot E_L \cdot Y' \cdot D \longrightarrow W_{Lj2(n+1)} = W_{Lj2} + \Delta W_{Lj2} \quad (14)$$

Hidden neuron 2

$$\Delta W_{j1h1} = \eta \cdot \delta_{j1} \cdot A \longrightarrow \delta_{j1} = E_{j1} \cdot C' \quad (15)$$

$$E_{j1} = E_L \cdot Y' \cdot W_{Lj1} \longrightarrow \Delta W_{j1h1} = \eta \cdot E_L \cdot Y' \cdot W_{Lj1} \cdot C' \cdot A \quad (16)$$

$$W_{j1h1(n+1)} = W_{j1h1} + \Delta W_{j1h1} \quad (17)$$

$$\Delta W_{j1h2} = \eta \cdot \delta_{j1} \cdot B \longrightarrow \Delta W_{j1h2} = \eta \cdot E_L \cdot Y' \cdot W_{Lj1} \cdot C' \cdot B \quad (18)$$

$$W_{j1h2(n+1)} = W_{j1h2} + \Delta W_{j1h2} \quad (19)$$

$$\Delta W_{j2h1} = \eta \cdot \delta_{j2} \cdot A \longrightarrow \delta_{j2} = E_{j2} \cdot D' \quad (20)$$

$$E_{j2} = E_L \cdot Y' \cdot W_{Lj2} \longrightarrow \Delta W_{j2h1} = \eta \cdot E_L \cdot Y' \cdot W_{Lj2} \cdot D' \cdot A \quad (21)$$

$$W_{j2h1(n+1)} = W_{j2h1} + \Delta W_{j2h1} \quad (22)$$

$$\Delta W_{j2h2} = \eta \cdot \delta_{j2} \cdot B \longrightarrow \Delta W_{j2h2} = \eta \cdot E_L \cdot Y' \cdot W_{Lj2} \cdot D' \cdot B \quad (23)$$

$$W_{j2h2(n+1)} = W_{j2h2} + \Delta W_{j2h2} \quad (24)$$

Hidden neuron 1

$$\Delta W_{h1x1} = \eta \cdot \delta_{h1} \cdot x1 \longrightarrow \delta_{h1} = E_{h1} \cdot A' \quad (25)$$

$$E_{h1} = E_{j1} \cdot C' \cdot W_{j1h1} + E_{j2} \cdot D' \cdot W_{j2h1} \quad (25)$$

$$\Delta W_{h1x1} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h1} + E_{j2} \cdot D' \cdot W_{j2h1}) \cdot A' \cdot x1 \quad (25)$$

$$W_{h1x1(n+1)} = W_{h1x1} + \Delta W_{h1x1} \quad (26)$$

$$\Delta W_{h1x2} = \eta \cdot \delta_{h1} \cdot x2 \longrightarrow \Delta W_{h1x2} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h1} + E_{j2} \cdot D' \cdot W_{j2h1}) \cdot A' \cdot x2 \quad (27)$$

$$W_{h1x2(n+1)} = W_{h1x2} + \Delta W_{h1x2} \quad (28)$$

$$\Delta W_{h1x3} = \eta \cdot \delta_{h1} \cdot x3 \longrightarrow \Delta W_{h1x3} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h1} + E_{j2} \cdot D' \cdot W_{j2h1}) \cdot A' \cdot x3 \quad (29)$$

$$W_{h1x3(n+1)} = W_{h1x3} + \Delta W_{h1x3} \quad (30)$$

$$\Delta W_{h1x4} = \eta \cdot \delta_{h1} \cdot x4 \longrightarrow \Delta W_{h1x4} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h1} + E_{j2} \cdot D' \cdot W_{j2h1}) \cdot A' \cdot x4 \quad (31)$$

$$W_{h1x4(n+1)} = W_{h1x4} + \Delta W_{h1x4} \quad (32)$$

$$\Delta W_{h1x5} = \eta \cdot \delta_{h1} \cdot x5 \longrightarrow \Delta W_{h1x5} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h1} + E_{j2} \cdot D' \cdot W_{j2h1}) \cdot A' \cdot x5 \quad (33)$$

$$W_{h1x5(n+1)} = W_{h1x5} + \Delta W_{h1x5} \quad (34)$$

$$\Delta W_{h1x6} = \eta \cdot \delta_{h1} \cdot x6 \longrightarrow \Delta W_{h1x6} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h1} + E_{j2} \cdot D' \cdot W_{j2h1}) \cdot A' \cdot x6 \quad (35)$$

$$W_{h1x6(n+1)} = W_{h1x6} + \Delta W_{h1x6} \quad (36)$$

$$\Delta W_{h1x7} = \eta \cdot \delta_{h1} \cdot x7 \longrightarrow \Delta W_{h1x7} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h1} + E_{j2} \cdot D' \cdot W_{j2h1}) \cdot A' \cdot x7 \quad (37)$$

$$W_{h1x7(n+1)} = W_{h1x7} + \Delta W_{h1x7} \quad (38)$$

$$\Delta W_{h1x8} = \eta \cdot \delta_{h1} \cdot x8 \longrightarrow \Delta W_{h1x8} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h1} + E_{j2} \cdot D' \cdot W_{j2h1}) \cdot A' \cdot x8 \quad (39)$$

$$W_{h1x8(n+1)} = W_{h1x8} + \Delta W_{h1x8} \quad (40)$$

$$\Delta W_{h1x9} = \eta \cdot \delta_{h1} \cdot x9 \longrightarrow \Delta W_{h1x9} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h1} + E_{j2} \cdot D' \cdot W_{j2h1}) \cdot A' \cdot x9 \quad (41)$$

$$W_{h1x9(n+1)} = W_{h1x9} + \Delta W_{h1x9} \quad (42)$$

$$\Delta W_{h2x1} = \eta \cdot \delta_{h2} \cdot x1 \longrightarrow \delta_{h2} = E_{h2} \cdot B' \quad (43)$$

$$E_{h2} = E_{j1} \cdot C' \cdot W_{j1h2} + E_{j2} \cdot D' \cdot W_{j2h2} \quad (44)$$

$$\Delta W_{h2x1} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h2} + E_{j2} \cdot D' \cdot W_{j2h2}) \cdot B' \cdot x1 \quad (45)$$

$$W_{h2x1(n+1)} = W_{h2x1} + \Delta W_{h2x1} \quad (46)$$

$$\Delta W_{h2x2} = \eta \cdot \delta_{h2} \cdot x2 \longrightarrow \Delta W_{h2x2} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h2} + E_{j2} \cdot D' \cdot W_{j2h2}) \cdot B' \cdot x2 \quad (47)$$

$$W_{h2x2(n+1)} = W_{h2x2} + \Delta W_{h2x2} \quad (48)$$

$$\Delta W_{h2x3} = \eta \cdot \delta_{h2} \cdot x3 \longrightarrow \Delta W_{h2x3} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h2} + E_{j2} \cdot D' \cdot W_{j2h2}) \cdot B' \cdot x3 \quad (49)$$

$$W_{h2x3(n+1)} = W_{h2x3} + \Delta W_{h2x3} \quad (50)$$

$$\Delta W_{h2x4} = \eta \cdot \delta_{h2} \cdot x4 \longrightarrow \Delta W_{h2x4} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h2} + E_{j2} \cdot D' \cdot W_{j2h2}) \cdot B' \cdot x4 \quad (51)$$

$$W_{h2x4(n+1)} = W_{h2x4} + \Delta W_{h2x4} \quad (52)$$

$$\Delta W_{h2x5} = \eta \cdot \delta_{h2} \cdot x5 \longrightarrow \Delta W_{h2x5} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h2} + E_{j2} \cdot D' \cdot W_{j2h2}) \cdot B' \cdot x5 \quad (53)$$

$$W_{h2x5(n+1)} = W_{h2x5} + \Delta W_{h2x5} \quad (54)$$

$$\Delta W_{h2x6} = \eta \cdot \delta_{h2} \cdot x6 \longrightarrow \Delta W_{h2x6} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h2} + E_{j2} \cdot D' \cdot W_{j2h2}) \cdot B' \cdot x6 \quad (55)$$

$$W_{h2x6(n+1)} = W_{h2x6} + \Delta W_{h2x6} \quad (56)$$

$$\Delta W_{h2x7} = \eta \cdot \delta_{h2} \cdot x7 \longrightarrow \Delta W_{h2x7} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h2} + E_{j2} \cdot D' \cdot W_{j2h2}) \cdot B' \cdot x7 \quad (57)$$

$$W_{h2x7(n+1)} = W_{h2x7} + \Delta W_{h2x7} \quad (58)$$

$$\Delta W_{h2x8} = \eta \cdot \delta_{h2} \cdot x8 \longrightarrow \Delta W_{h2x8} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h2} + E_{j2} \cdot D' \cdot W_{j2h2}) \cdot B' \cdot x8 \quad (59)$$

$$W_{h2x8(n+1)} = W_{h2x8} + \Delta W_{h2x8} \quad (60)$$

$$\Delta W_{h2x9} = \eta \cdot \delta_{h2} \cdot x9 \longrightarrow \Delta W_{h2x9} = \eta \cdot (E_{j1} \cdot C' \cdot W_{j1h2} + E_{j2} \cdot D' \cdot W_{j2h2}) \cdot B' \cdot x9 \quad (61)$$

$$W_{h2x9(n+1)} = W_{h2x9} + \Delta W_{h2x9} \quad (62)$$

Proses selanjutnya adalah proses pembelajaran yang dilakukan secara berulang-ulang hingga nilai *error* yang dihasilkan mencapai nilai yang paling kecil.

Perhitungan diatas hanya sebagai simulasi saja, karena pengambilan metoda yang digunakan tidak dilakukan secara keseluruhan, namun dengan penjabaran diatas sudah mencakup cara penggunaan metoda yang lain.

Simulasi perhitungan ini diperoleh dari metoda yang paling mudah yaitu menggunakan metoda Back Propagation dengan memakai *Gradient Descendent* sebagai metoda untuk melakukan proses *update* pada nilai *weight* dan *bias*. Penggunaan metoda lain dapat digunakan dengan prinsip pemrosesan yang telah dilakukan diatas, sehingga proses untuk dapat mencapai hasil yang bervariasi dapat dengan mudah untuk diperoleh.

Pada bab selanjutnya akan dijelaskan proses pengujian dengan data yang sesungguhnya yang kemudian akan dijadikan sebagai hasil dari penggunaan metoda MLP-NN pada pendeteksi gangguan dengan berdasarkan pada anomali yang terjadi pada suatu jaringan.

Bab 5

Proses Pengujian serta Analisa Multilayer Perceptron Neural Network pada suatu Jaringan

Pada bab ini dilakukan proses pengujian dengan simulasi penggunaan Sistem UNIX sebagai sistem operasi yang dipakai pada jaringan yang akan diuji. Adapun hasil yang didapat dari setiap *USER*, menunjukkan bahwa pada lingkungan aplikasi Sistem Operasi UNIX memiliki empat atribut seperti *Command*, *Host*, *Time* serta waktu Eksekusi atau *Execution Time* yang memberikan kesimpulan secara mendasar untuk dijadikan sebagai acuan dalam pengambilan data yang akan dianalisa. Pertama adalah seperti hal dibawah ini[5] :

- Bahwa setiap komputer pengguna memiliki perintah *UNIX* yang berbeda-beda (unik).
- Proses Pengerjaan yang berdasarkan pada sistem penjadwalan.
- Proses *logs* terjadi pada *Host* yang tetap (Regular Host).
- Proses berjalannya suatu perintah biasanya tidak memakan waktu lama.

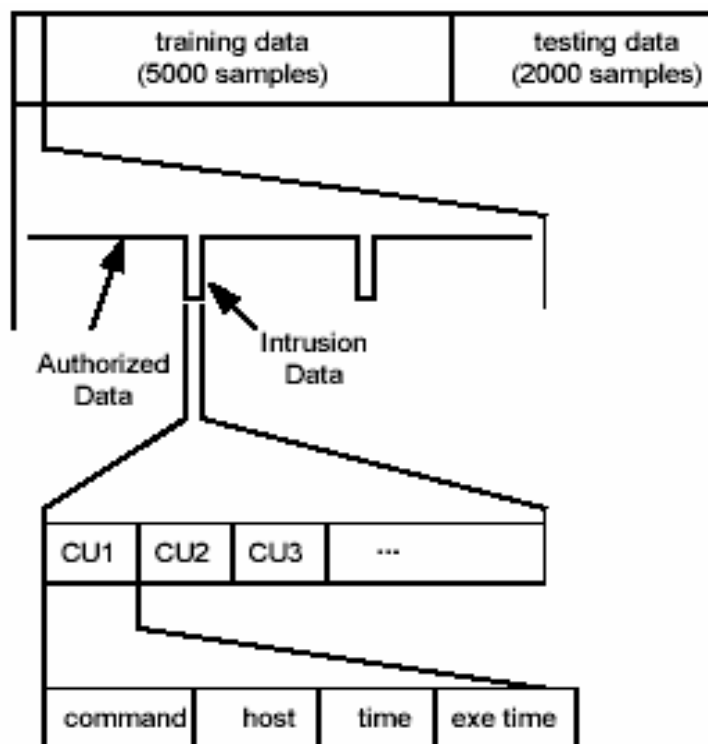
Kesimpulan yang kedua adalah profil yang dimiliki oleh pengguna biasanya selalu ada saat proses perintah dilakukan.

Untuk mempermudah dalam pengujian pada metoda Back Propagation ini, pengambilan data tidak mengikutsertakan profil pengguna sebagai bahan kajian pada sistem pendeteksian ini.

Bentuk data yang diberikan sebagai masukan pada jaringan MLP-NN ini terbagi menjadi dua bagian, pertama adalah data yang digunakan sebagai pembelajaran, dengan setiap masukan diberikan acuan yang sesuai dengan kondisi masing-masing. Adapun pola pembagian yang terjadi saat proses pembelajaran yaitu 90% sebagai kondisi normal dan 10% sebagai kondisi yang mengalami gangguan. Kedua, Proses pengujian ini menggunakan lima metoda penggunaan *back Propagation* untuk mendapatkan hasil yang maksimal dalam pendeteksian kondisi gangguan pada suatu jaringan. Pola pemabagian data untuk pengujian ini adalah 98% sebagai kondisi normal dan 2% sebagai kondisi mengalami gangguan.

Secara keseluruhan pengambilan data ini berjumlah 7000 data, dengan pembagian 5000 data sebagai pembelajaran dan 2000 data selanjutnya adalah data pengujian.

Adapun kondisi masukan yang terjadi jika dilihat secara *logic* menunjukkan bahwa sisi positif dari suatu data masukan adalah penunjuk sebagai kondisi normal sedangkan sisi negatifnya adalah gambaran bila terjadi suatu gangguan. Berikut ini adalah penjelasan secara gambar [5].



Gambar 12. Metoda Masukan untuk indikasi suatu jaringan

Dalam melakukan pengujian serta pembelajaran diharapkan kemampuan *Neural network* dapat dengan baik saat melakukan pendeteksian, untuk mendukung hal tersebut maka diberikan tiga jenis *File* yang terdiri dari beberapa unit perintah yang masing-masing *File* memiliki jumlah yang berbeda. *File* 1 terdiri dari 5 unit perintah, *File* 2 terdiri dari 6 unit perintah dan *File* 3 terdiri dari 7 unit perintah. Hasil kalkulasi secara total dari *File* yang ada, Tabel berikut merupakan keterangannya.

Tabel 3. Total Elemen Pembelajaran serta Pengujian Sistem MLP

	File1	File2	File3
Train Data	100,000	120,000	140,000
Test Data	40,000	48,000	56,000
Total	140,000	168,000	196,000

Berikut ini adalah hasil kalkulasi dengan metoda perhitungan baik saat melakukan pembelajaran serta saat pengujian dengan merubah pola arsitektur secara bergantian[5].

Tabel 4. Kondisi Hasil untuk File 1

Gradient Descent (GD)	
Topology: {20,10,1}	Nbr Epoch = 1000
$\alpha = 0.01$	Nbr Flops = 5 GFlops
MSE = $4.5 \exp(-5)$	Error Percentage = 0
GD with Momentum (GDM)	
Topology: {20,10,1}	Nbr Epoch = 1000
$\alpha = 0.05, \mu = 0.75$	Nbr Flops = 5 GFlops
MSE = 60	Error Percentage = 100
Variable Learning Rate GD with Momentum (GDX)	
Topology: {20,10,1}	Epoch = 1000
$\alpha = 0.05, \mu = 0.75,$ $\beta = 0.7 \& 1.05$	Nbr Flops = 8.4 GFlops
MSE = $4.5 \exp(-4)$	Error Percentage = 0
Conjugate Gradient Descent (CGP)	
Topology: {20,6,1}	Nbr Epoch = 35
	Nbr Flops = 0.23 GFlops
MSE = $\exp(-5)$	Error Percentage = 0
Quasi-Newton (BFGS)	
Topology: {20,6,1}	Nbr Epoch = 20
	Nbr Flops = 0.2 GFlops
MSE = $1.2 \exp(-6)$	Error Percentage = 0

Tabel 5. Kondisi hasil untuk *File 2*

Gradient Descent (GD)	
Topology: {24,10,1}	Nbr Epoch = 1000
$\alpha = 0.01$	Nbr Flops = 5.8 GFlops
MSE = $1.1 \exp(-4)$	Error Percentage = 0
GD with Momentum (GDM)	
Topology: {24,10,1}	Nbr Epoch = 1000
$\alpha = 0.05, \mu = 0.75$	Nbr Flops = 5.8 GFlops
MSE = $2.28 \exp(-4)$	Error Percentage = 0
Variable Learning Rate GD with Momentum (GDX)	
Topology: {24,10,1}	Epoch = 1000
$\alpha = 0.05, \mu = 0.75,$ $\beta = 0.7 \& 1.05$	Nbr Flops = 1.5 GFlops
MSE = $5.1 \exp(-4)$	Error Percentage = 0
Conjugate Gradient Descent (CGP)	
Topology: {24,6,1}	Nbr Epoch = 1000
	Nbr Flops = 1.5GFlops
MSE = $\exp(-5)$	Error Percentage = 0
Quasi-Newton (BFGS)	
Topology: {24,6,1}	Nbr Epoch = 45
	Nbr Flops = 0.6 GFlops
MSE = $\exp(-5)$	Error Percentage = 0

Tabel 6. Kondisi Hasil untuk *File 3*

Gradient Descent (GD)	
Topology: {28,16,1}	Nbr Epoch = 1000
$\alpha = 0.01$	Nbr Flops = 21 GFlops
MSE = $2.3 \exp(-5)$	Error Percentage = 0
GD with Momentum (GDM)	
Topology: {28,18,1}	Nbr Epoch = 1000
$\alpha = 0.05, \mu = 0.75$	Nbr Flops = 15 GFlops
MSE = $2.3 \exp(-4)$	Error Percentage = 0
Variable Learning Rate GD with Momentum (GDX)	
Topology: {28,12,1}	Nbr Epoch = 1000
$\alpha = 0.05, \mu = 0.75,$ $\beta = 0.7 \& 1.05$	Nbr Flops = 7.5 GFlops
MSE = $\exp(-3)$	Error Percentage = 0
Conjugate Gradient Descent (CGP)	
Topology: {28,10,1}	Nbr Epoch = 45
	Nbr Flops = 0.6 GFlops
MSE = $1.9 \exp(-3)$	Error Percentage = 0
Quasi-Newton (BFGS)	
Topology: {28,8,1}	Nbr Epoch = 30
	Nbr Flops = 0.6 GFlops
MSE = $\exp(-5)$	Error Percentage = 0

3. Penggunaan metoda GD, GDM dan GDM, tidak dapat mencapai nilai MSE hingga sama dengan $\exp(-5)$, ketiga metoda ini hanya dapat mencapai nilai $MSE = \exp(-3)$, meskipun masih dapat diklasifikasikan untuk dapat mendeteksi suatu gangguan.
4. Hasil terbaik dari penggunaan beberapa metoda *Back Propagation* adalah metoda CGP dan BFGS, dimana iterasi pembelajaran (number epoch) yang dihasilkan saat mencapai nilai konvergen adalah paling kecil dibandingkan metoda-metoda lain. Selain itu, dari beberapa metoda lain kedua metoda ini merupakan rancangan yang paling sederhana, yang terlihat pada pola [Masukan, hidden neuron, keluaran]. Meskipun hasil akhir yang terbaik dari tingkat sederhana rancangannya adalah metoda BFGS.
5. Hasil akhir dari penggunaan jumlah data masukan yang bervariasi, yang paling baik adalah saat menggunakan 6 jumlah unit perintah, dimana hasil MSE yang diberikan mencapai nilai *error* yang paling rendah.
6. Penggunaan jumlah neuron sangat tergantung dari jumlah unit perintah yang diberikan, hal ini terlihat bahwa hasil terbaik untuk metoda GD, GDM dan GDH adalah rancangan [24, 10, 1], sedangkan pola [24, 6, 10] dicapai oleh metoda CGP dan BFGS.

Bab 6

Kesimpulan

Penggunaan Metoda *Multilayer Perceptron Neural Network* (MLP-NN) untuk Sistem Pendeteksian Gangguan yang terjadi pada Anomali suatu Jaringan adalah sebagai berikut :

1. Metoda MLP-NN sangat membantu dalam melakukan pendeteksian suatu gangguan secara efisiensi bila dibandingkan dengan metoda konvensional yang tidak mampu melakukan pembelajaran untuk mengetahui sesuatu yang baru.
2. Kapasitas penggunaan komputer yang memang sangat mempengaruhi dalam penggunaan metoda ini, menyebabkan persyaratan sistem (System Requirement) yang tinggi untuk dapat melakukan pencapaian hasil yang maksimal.
3. Metoda ini hanya dapat menjaga sistem keamanan secara eksternal sehingga pada prakteknya tetap membutuhkan perangkat lain yang sangat mendukung keamanan secara keseluruhan dalam hal ini misalnya penggunaan *Firewall* yang dipasang terintegrasi dengan sistem ini.
4. Penggunaan Metoda Back Propagation sangat mempengaruhi tingkat sensitifitas pendeteksian, sehingga perlu pengkajian secara mendalam dan bervariasi dalam memberikan pola pembelajaran terhadap mekanisme MLP-NN ini.
5. Pola kebijakan yang baik juga sangat mendukung sehingga terciptanya Sistem Keamanan yang terpadu.

Daftar Pustaka

- [1] Brenton, Chris, Mastering Network Security. Sybex: San Fransisco, 1999.
- [2] <http://www.cas.mcmaster.ca/~wmfarmer/SE-4C03-01/papers/Bielewicz-IDS.pdf>
- [3] Salameh, A. Walid, AStudies in Informatics and Control, Vol. 13, No. 2, June 2004 135
- [4] <http://mow.ecn.perdue/~terran/facts/research.html,1998>.
- [5] <http://www.cs.ucdavis.edu/~vemuri/bp-intrusion%20detection.pdf>
- [6] S. Haykin, Neural Networks – A Comprehensive Foundation, 2nd Edition, Prentice Hall, 2000.
- [7] <http://www.cs.rpi.edu/~szymansk/papers/annie02.pdf>