IMPLEMENTASI ALGORITMA *BACKPROPAGATION NEURAL NETWORK* DAN *ADABOOST* PADA *NETWORK INTRUSION DETECTION SYSTEM*

Nowo Adisuryo[1], Ken Kinanti Purnamasari[2]

[1,2]Teknik Informatika - Universitas Komputer Indonesia
Jl. Dipatiukur 112 – 114 Bandung
E-mail : nowoadisuryo@gmail.com[1,] ken.kinanti@email.unikom.ac.id[2]

ABSTRAK

Network Intrusion Detection System (NIDS) adalah sebuah perangkat lunak atau perangkat keras yang dapat menganalisis lalu lintas yang mengalir ke suatu jaringan. NIDS akan memberi peringatan pada *Network Administrator* jika terdapat percobaan serangan atau penyusupan ke dalam sistem jaringan. Dalam NIDS, algoritma *machine learning* telah banyak diterapkan, contohnya, *Decision Tree*, SVM, *Naive Bayes*, *Artificial Neural Network*, dan lain-lain. Terdapat tantangan dalam penggunaan *machine learning* untuk NIDS yaitu sistem harus memiliki tingkat akurasi yang tinggi, tingkat *error* yang rendah, waktu komputasi yang cepat, dan penggunaan memori yang sedikit dalam mendeteksi serangan pada jaringan [1]. Pada penelitian ini akan digunakan algoritma *backpropagation neural network* (BPNN) dan *adaptive boost* (*AdaBoost*) pada NIDS untuk mengklasifikasikan paket data jaringan menjadi paket data normal atau serangan. Berdasarkan hasil pengujian dengan menggunakan *dataset* Kyoto 2006+ yang terdiri dari 2000 data, metode BPNN memperoleh akurasi yang lebih rendah jika dibandingkan dengan metode BPNN yang dilengkapi dengan *AdaBoost*. Metode BPNN memperoleh akurasi sebesar 99,75% sementara metode BPNN yang dilengkapi dengan *AdaBoost* memperoleh akurasi sebesar 99,8%.

Kata kunci — *backpropagation*, *adaboost*, kyoto *dataset*, sistem deteksi intrusi jaringan

1. PENDAHULUAN

Cepatnya perkembangan dan proliferasi dari *botnet*, serangan skala besar seperti DDoS, *spam email*, dan lain-lain pada jaringan komputer menjadi ancaman yang lebih serius dan berbahaya dari serangan-serangan yang sebelumnya. Berdasarkan ancaman ini maka dari itu *Network based Intrusion Detection System* (NIDS), *Intrusion Prevention System* (IPS), *firewall* menjadi sangat dibutuhkan untuk mempertahankan sistem komputer, jaringan dan informasi yang berharga [2].

NIDS adalah penyebaran dari *real-time monitoring network sensors* pada lokasi vital dalam infrastruktur jaringan. *Network sensors* menganalisa trafik dan mendeteksi aktivitas serangan dan juga aktivitas mencurigakan [3]. Berdasarkan teknik yang digunakan NIDS dapat diklasifikasikan menjadi 2 kategori yang berbeda. Salah satunya adalah *anomaly detection* dan yang lain adalah *signature detection* atau *misuse detection* [4].

Penggunaan machine learning untuk NIDS telah banyak digunakan pada penelitian-penelitian sebelumnya. Seperti penggunaan metode Decision Tree, SVM, Naive Bayes, Artificial Neural Network, dan lain-lain. Terdapat tantangan dalam penggunaan machine learning untuk NIDS yaitu sistem harus memiliki tingkat akurasi yang tinggi, tingkat error yang rendah, waktu komputasi yang cepat, dan penggunaan memori yang sedikit dalam mendeteksi serangan pada jaringan [1].

Penelitian yang dilakukan oleh Harshal A. Sonawane dan Tareek M. Pattewar [1] menggunakan metode *Backpropagation Neural Network* (BPNN) dan *Principal Component Analysis* (PCA) pada IDS. Penelitian ini membandingkan antara penggunaan BPNN dan PCA dengan penggunaan BPNN saja. Hasil dari penelitian ini, penggunaan BPNN dan PCA mendapatkan akurasi sebesar 86,90% dan penggunaan BPNN saja mendapatkan akurasi sebesar 90,20%. Penelitian yang dilakukan oleh Indraneel Mukhopadhyay dan kawan-kawan [5] menggunakan pendekatan BPNN pada IDS, dari penelitian ini menghasilkan akurasi sebesar 95,6%. Penelitian yang dilakukan oleh Gang Wang dan kawan-kawan [6] menggunakan metode *Artificial Neural Networks* (ANN) dan *fuzzy clustering* pada IDS, dalam penelitian ini juga membandingkan antara metode lain yaitu, *Decision Tree*, *Naive Bayes*, BPNN, FC-ANN (*Fuzzy Clustering – Artificial Neural Networks*). Penelitian ini menghasilkan akurasi sebesar 96,71% untuk FC-ANN, 96,65% untuk BPNN, 96,11% untuk *Naive Bayes*, 96,75% untuk *Decision Tree*. Penelitian yang dilakukan oleh Weiming Hu dan kawan-kawan [7] menggunakan algoritma *AdaBoost* dan sebagai *weak classifier* menggunakan algoritma *Decision Stumps*. Penelitian ini menghasilkan akurasi sebesar 90,04% - 90,88%. Penelitian [1], [5], [6], [7] menggunakan *dataset* KDD CUP 1999 sebagai data latih dan uji. *Dataset* KDD CUP 1999 telah banyak dikritik karena sudah tidak lagi mewakili kondisi

jaringan terkini [2]. Secara teori akurasi dari algoritma learning dapat ditingkatkan dengan menggunakan *Boosting* [8]. Salah satu metode boosting yaitu *Adaptive Boosting* (*AdaBoost*). Metode *AdaBoost* dapat meningkatkan akurasi pada kasus *binary classification*. Untuk melakukan boosting dengan *AdaBoost* diperlukan algoritma dasar sebagai algoritma *weak learner* yang *output* dari algoritma tersebut akan diintegrasikan dengan *AdaBoost* untuk membuat *strong learner* [9]. Penelitian yang dilakukan oleh Jianfang Cao dan kawan-kawan [9] telah mengimplementasikan BPNN dan *AdaBoost* untuk mengklasifikasikan semantik emosi yang terkandung dalam gambar-gambar pemandangan. Hasil dari penelitian tersebut algoritma BPNN dan *AdaBoost* mendapatkan tingkat akurasi yang lebih tinggi sekitar 15% dibandingkan dengan metode-metode yang telah dikembangkan sebelumnya. Peforma dari BPNN dan *AdaBoost* pada kasus NIDS belum diketahui, maka dari itu pada penelitian ini akan menggunakan algoritma BPNN dan *AdaBoost* untuk kasus NIDS.

2. ISI PENELITIAN

2.1 Preprocessing

Preprocessing terdiri dari *encoding* dan normalisasi *min-max.*

2.1.1 *Encoding*

Dalam *supervised learning* umumnya terdapat berbagai macam bentuk data. Data tersebut bisa dalam bentuk angka atau kata. Jika data tersebut berbentuk angka, maka algoritma dapat langsung memprosesnya. Jika data tersebut berbentuk kata, maka data tersebut harus ditransformasikan menjadi bentuk numerik sehingga algoritma dapat memprosesnya.

Pada *label encoding* cara mentransformasikan data berbentuk kata menjadi bentuk numerik adalah sebagai berikut. Misalkan terdapat sebuah fitur dengan data berbentuk kategori, {‘audi’,’bmw’,’ford’,’toyota’}, data tersebut akan ditransformasi menjadi bentuk numerik dengan cara mengganti data indeks ke 0, ..., n, n adalah panjang *array*, dengan indeksnya, {‘0’,’1’,’2’,’3’} [10].

2.1.2 Normalisasi *Min-Max*

Normalisasi *min-max* menjalankan transformasi linier pada data asli. Misalkan min_A dan max_A adalah nilai minimum dan maksimum dari sebuah atribut, *A*. Normalisasi *min-max* memetakan sebuah nilai , x_i, dari *A* ke x'_i dalam rentang [new_min_A, new_max_A] dengan menghitung

$$x'_i = \frac{x_i - min_j}{max_j - min_j}(new_max_j - new_min_j) + new_min_j. \quad (2.1)$$

Normalisasi *min-max* mempertahankan hubungan diantara nilai-nilai data asli. Normalisasi ini akan menemukan sebuah *error* “di-luar-rentang” jika sebuah kasus masukan di masa depan berada di luar rentang data asli untuk *A* [11].

2.2 *Backpropagation*

Backpropagation adalah salah satu algoritma *supervised learning* yang digunakan dalam *artificial neural networks*. *Backpropagation* mencari kombinasi bobot untuk meminimalkan kesalahan *output* untuk dianggap menjadi solusi yang benar [12]. Secara sederhana *Backpropagation* dilakukan dalam dua tahap, yaitu:

1. *Feedfoward*

   Pola yang akan dilatih diset ke setiap unit di *input layer*, lalu *output* yang dihasilkan ditransmisikan ke *layer* selanjutnya terus sampai *output layer*.
2. *Backpropagation*

   Berdasarkan *output* yang diharapkan, setiap bobot disesuaikan agar menghasilkan *error* yang minimal mulai dari bobot yang terhubung ke *output neuron*, lalu terus mundur sampai ke *input layer.*

Salah satu *activation function* yang populer untuk jaringan *backpropagation* adalah *binary sigmoid* yang didefinisikan sebagai (2.2). Untuk tahap *backpropagation* menggunakan fungsi turunannya yaitu (2.3).

$$f(X) = \frac{1}{1+e^{-X}} \quad (2.2)$$

$$f'(X) = f(X)(1 - f(X)) \quad (2.3)$$

Berikut tahapan pada proses pelatihan algoritma *backpropagation*.

Feedfoward:

Langkah 1 : Inisialisasi bobot dan parameter

Langkah 2 : Setiap unit *input* (X_i, i = 1, 2, 3, ..., n) mengirimkan sinyal *input* ke setiap *input* yang berada pada *hidden layer.*

Langkah 3 : Masing-masing unit di lapisan *hidden layer* (Z_j, j = 1, 2, 3, ..., p) dikalikan dengan bobotnya dan jumlahkan serta tambahkan dengan biasnya.

$$z_in_j = V_{0,j} + \sum_{i=1}^{n} X_i V_{i,j} \quad (2.4)$$

menghitung sinyal *hidden* dengan menggunakan fungsi aktivasi.

$$Z_j = f(z_in_j) \quad (2.5)$$

Langkah 3 : Masing-masing unit *output* (y_k, k = 1, 2, 3, ..., m) dikalikan bobot dan dijumlahkan dengan biasnya.

$$y_in_k = W_{0,k} + \sum_{j=1}^{p} Z_j W_{j,k} \quad (2.6)$$

menghitung sinyal *output* dengan menggunakan fungsi aktivasi.

$$y_k = f(y_in_k) \quad (2.7)$$

Backward:

Langkah 4 : Masing-masing unit *output* (y_k, k = 1, 2, 3, ..., m) menerima target pola t_k sesuai dengan pola *input* saat pelatihan dan kemudian informasi *error* lapisan *output* (δ_k) dihitung.

$$\delta_k = (t_k - y_k) * ((y_k) * (1 - y_k)) \quad (2.8)$$

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai $W_{j,k}$):

$$\Delta W_{j,k} = \alpha\delta_k Z_j \quad (2.9)$$

Hitung koreksi bias untuk memperbaiki nilai $W_{0,k}$:

$$\Delta W_{0,k} = \alpha\delta_k \quad (2.10)$$

Langkah 5 : Pada setiap *hidden layer* (Z_j, j = 1, 2, 3, …, p) menjumlahkan delta *input* (dari unit-unit yang berada pada lapisan sebelumnya)

$$\delta_\text{in}_j = \sum_{k=1}^{m} \delta_k W_{j,k} \quad (2.11)$$

$$\delta_j = (\delta_\text{in}_j)(z_in_j)(1 - z_in_j) \quad (2.12)$$

Kemudian hitung koreksi bobot untuk memperbaiki bobot V.

$$\Delta V_{i,j} = \alpha\delta_k X_i \quad (2.13)$$

Hitung koreksi bobot bias $V_{0,j}$.

$$\Delta V_{0,j} = \alpha\delta_j \quad (2.14)$$

Langkah 6 : masing-masing unit *output* (y_k, k = 1, 2, 3, ..., m) dilakukan untuk memperbaiki bias dan bobotnya (j = 0, 1, 2, 3, ..., p) sehingga menghasilkan bobot dan bias baru.

$$W_{j,k}(baru) = W_{j,k}(lama) + \Delta W_{j,k} \quad (2.15)$$

Demikian juga untuk bobot pada *input layer* ke *hidden layer*, diperbaiki dengan menggunakan koreksi bobot yang sudah dihitung sebelumnya.

$$V_{i,j}(baru) = V_{i,j}(lama) + \Delta V_{i,j} \quad (2.16)$$

2.3 *Backpropagation* dan *Adaptive Boost*

Merujuk kepada algoritma *AdaBoost*, efisiensi dan akurasi dari algoritma *AdaBoost* yang berdasarkan dari sebuah seri *weak classifiers* sebagian besar bergantung kepada akurasi dari *weak classifier* itu sendiri. Pada penelitian ini *Backpropagation Neural Network* dipilih sebagai *weak classifier* dan hasil akhir dari *AdaBoost classifier* diperlukan suatu metode untuk mengkombinasikan hasil *weak classifier* tersebut menjadi sebuah *AdaBoost classifier* yang efisien. Pada penelitian ini akan menggunakan metode *weighted voting method* untuk mengkombinasikan *output* dari setiap *weak classifier* untuk mendapatkan *final outputs* [13]. Algoritma untuk membangun *hybrid model* dari *AdaBoost* dan *Backpropagation* dapat dilihat sebagai berikut.

Langkah 1 : Pilih sampel-sampel untuk pelatihan secara acak dari *dataset* dengan menggunakan metode 10-*fold cross-validation*. Setiap grup sampel pelatihan akan diberi bobot yang sama.

T = $\{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\}$ selama $x_i \in X$, $y_i \in Y = \{-1, +1\}$.

Bobot-bobot dari distribusi ini pada sampel-sampel pelatihan pada putaran *t* akan dilambangkan dengan $D_t(i)$. Pada awalnya, semua bobot akan diberi nilai yang sama.

$$D1(i) = \frac{1}{n}, i = 1, 2, \ldots, n, \quad (2.17)$$

dimana *n* merupakan jumlah total sampel pada *dataset* pelatihan.

Langkah 2 : Parameter-parameter dari BPNN, dimana termasuk struktur dari *network*, *neurons* pada setiap *layer*, target yang diharapkan, akan diatur menurut atribut-atribut dari *input* dan *output* sampel-sampel pelatihan. Setelah itu inisialisasi bobot-bobot dan bias dari BPNN.

Langkah 3 : Latih menggunakan BPNN dengan sampel-sampel yang telah diberi bobot.

Langkah 4 : Ketika *epochs* pelatihan sampai kepada putaran *t*, sebuah *weak classifier* $h_t(x)$ akan didapatkan dengan bobot-bobot distribusi D_t dan setelah itu hitung *error* terbobot ε_t dari model:

$$\varepsilon_t = \sum_{i=1}^{n} D_t(i) \text{ x } I(h_t(x_i)) \quad (2.18)$$

dimana $I(h_t(x_i))$ adalah fungsi indikator. Ekspresi matematika dari fungsi indikator dapat dilihat sebagai berikut.

$$I(h_t(x_i)) = \begin{cases} 1, jika\ h_t(x_i) \neq y_i \\ 0, jika\ h_t(x_i) = y_i. \end{cases} \quad (2.19)$$

Langkah 5 : Hitung bobot dari *weak classifier*

$$a_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right). \quad (2.20)$$

Jika *error* prediksi dari sampel-sampel kurang dari 0.5, bobot dari *weak classifier* akan lebih besar daripada nol, dengan berkurangnya *error* prediksi akan meningkatkan bobot dari *weak classifier*, yang berarti bobot *voting* yang lebih besar pada *weak classifier* ini pada *final output.*

Langkah 6 : Perbaharui bobot-bobot pada setiap sampel pada *dataset* pelatihan untuk iterasi selanjutnya.

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \exp(-a_t \text{ x } y_i \text{ x } h_t(x_i)) \quad (2.21)$$

dimana Z_t adalah nilai normalisasi konstan untuk $D_{t+1}(i)$ yang ekspresi matematikanya dapat dilihat sebagai berikut.

$$Z_t = \sum_{i=1}^{n} D_t(i) \text{ x } \exp(-a_t \text{ x } y_i \text{ x } h_t(x_i)). \quad (2.22)$$

Dengan dilakukannya seperti ini, bobot-bobot pada data pelatihan yang mana diklasifikasikan dengan salah oleh *weak classifier* saat ini akan meningkat nilainya atau akan menurun nilainya jika sampel-sampel dikasifikasikan dengan benar.

Langkah 7 : Selama iterasi pelatihan terus berjalan, sebuah seri dari *weak classifiers* akan didapatkan, setelah itu kombinasikan *weak classifiers* tersebut selama proses pelatihan dengan persamaan berikut.

$$f(x) = \sum_{t=1}^{T} a_t \text{ x } h_t(x). \quad (2.23)$$

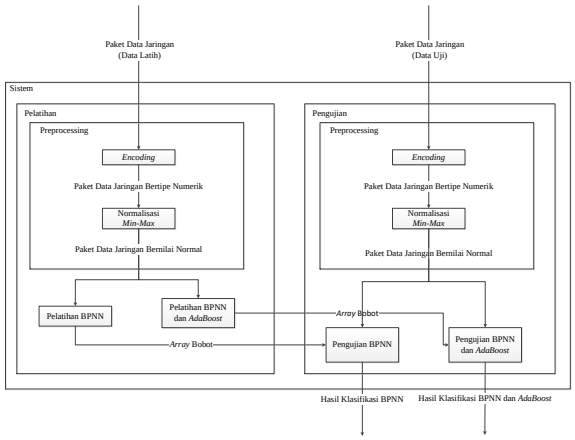
Sebuah seri dari *weak classifiers* dengan keberagaman bobot dikumpulkan dengan perulangan dari langkah 2 sampai langkah 5. Pada akhirnya, sebuah *AdaBoost model* berdasarkan *weak classifier* yang telah dilatih akan dibangun dengan persamaan berikut.

$$H(x) = \text{sign}(f(x)). \quad (2.24)$$

2.4 Pembahasan

Sistem meliputi dua tahap yaitu proses pelatihan dan proses pengujian. Proses pelatihan dilakukan sampai mendapatkan bobot yang diinginkan terpenuhi. Jika kondisi yang diinginkan sudah terpenuhi maka selanjutnya bisa dilanjutkan ke proses pengujian. Pada

proses pengujian, akan dimasukan data uji untuk mengklasifikasikan hasilnya akan dibandingkan dari bobot pelatihan yang dilakukan sebelumnya. alur proses sistem dapat dilihat pada Gambar 2.1.



Gambar 2.1 Alur Proses Sistem

Alur proses sistem pada Gambar 2.1 dimulai dengan memasukkan paket data jaringan yang disimpan dalam bentuk *comma separated values* (.csv) pada sistem. Pada tahap *preprocessing* dibagi menjadi dua proses yaitu proses *encoding* dan normalisasi data. Proses *encoding* yaitu mentransformasikan data berbentuk *string* menjadi bentuk numerik untuk dapat diproses oleh algoritma. Proses normalisasi data yaitu menormalisasi data agar data tersebut nilainya berada pada rentang -1,0 sampai 1,0 [11]. Pada tahap klasifikasi untuk pelatihan dan pengujian menggunakan ANN dengan arsitektur *Multi Layer Fowards* dan ANN integrasi dengan *AdaBoost*. *Neuron input* ANN berupa paket data jaringan yang sudah ternormalisasi. Kemudian proses pelatihan pada ANN dan ANN integrasi dengan *AdaBoost* menggunakan algoritma *backpropagation* yang hasil dari pelatihan ini menghasilkan bobot-bobot yang digunakan untuk tahapan pada proses pengujian. Setelah proses pengujian maka akan didapatkan hasil klasifikasi bahwa paket data jaringan termasuk normal atau serangan.

2.4.1 *Preprocessing*

Tahap *preprocessing* terbagi menjadi dua proses yaitu proses *encoding* dan normalisasi terhadap *dataset* paket data jaringan. Teknik normalisasi yang digunakan yaitu normalisasi *min-max*. Hasil dari *preprocessing* akan menghasilkan *dataset* yang telah ter-*encoding* dan ternormalisasi yang akan berguna sebagai data masukan bagi setiap tahap klasifikasi.

2.4.1.1 *Encoding*

Proses *encoding* data dilakukan untuk mengubah data bertipe *string* menjadi data bertipe numerik, agar selanjutnya data tersebut dapat digunakan untuk setiap tahap dalam klasifikasi.

Teknik men-*encode* data akan dilakukan dengan cara *label encoding*. Blok diagram untuk tahapan *encoding* dapat dilihat pada Gambar 2.2.

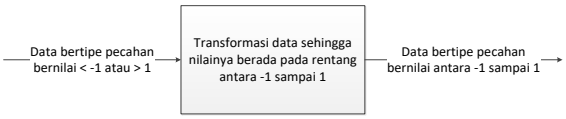


Gambar 2.2 Blok Diagram Proses *Encoding*

Dapat dilihat pada Gambar 2.2 data yang menjadi masukan adalah data bertipe *string*, lalu data tersebut akan melalui proses *encoding* sehingga data tersebut akan bertipe numerik setelah melalui proses *encoding*.

2.4.1.2 Normalisasi *Min-Max*

Normalisasi *min-max* merupakan proses normalisasi data bertipe numerik sehingga data dapat berada dalam rentang nilai antara -1,0 sampai 1,0 [11]. Data yang akan dinormalisasikan terdapat pada semua fitur. Normalisasi data menggunakan persamaan (2.19). Blok diagram untuk tahapan normalisasi data dapat dilihat pada Gambar 2.3.

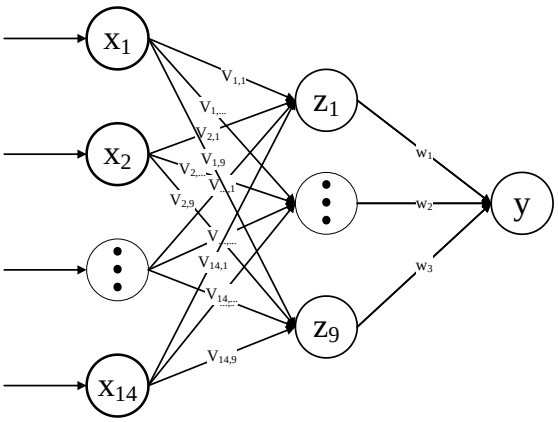


Gambar 2.3 Blok Diagram Proses Normalisasi *Min-Max*

Dapat dilihat pada Gambar 2.3 semua data bertipe pecahan yang nilainya < -1 atau > 1 akan ditransformasikan sehingga data tersebut nilainya akan berada pada rentang antara -1 sampai 1.

2.4.2 *Backpropagation*

Proses yang berjalan pada pelatihan dan pengujian ini menggunakan algoritma *backpropagation*. Jumlah *neuron* pada arsitektur *backpropagation* yang digunakan pada penelitian ini dapat dilihat pada Gambar 2.4.



Gambar 2.4 Arsitektur *Backpropagation*

Pada Gambar 2.4 terdapat *neuron* pada *input layer* (X) dengan jumlah 14 *neuron*, hal ini ditentukan berdasarkan jumlah fitur yang digunakan pada penelitian ini yaitu 14 fitur, X_j dimana j = 1, 2, ..., 14, akan berisi data pada fitur ke-j. Sedangkan pada *hidden layer* (Z) terdapat 9 *neuron*, jumlah ini didasarkan pada perbandingan menggunakan 5 dan 13 *neuron*. *Neuron* pada *output layer* berjumlah 1 *neuron* yang merupakan kelas normal atau serangan. Diantara *neuron input*, *hidden*, dan *output* terdapat bobot penghubung yaitu bobot antara *input-hidden layer* (V) dan *hidden-ouput layer* (W) serta bobot biasnya (V_0) dan (W_0).

2.4.2.1 Pelatihan *Backpropagation*

Proses yang dilakukan pada tahap pelatihan dengan menggunakan *backprpagation* adalah tahap inisialisasi bobot, proses *feedfoward*, *backpropagate*, dan *update* bobot.



Gambar 2.5 Blok Diagram Pelatihan BPNN

Dapat dilihat pada blok diagram pelatihan BPNN, masukannya berupa paket data jaringan bernilai normal, tahapan pertama adalah inisialisasi *neural networks* dan parameter pelatihan, tahapan kedua adalah inisialisasi bobot-bobot dimana pada penelitian ini nilai untuk bobot-bobot berada pada rentang -0,5 sampai 0,5, tahapan ketiga adalah pelatihan dengan BPNN dimana akan dilakukan tahapan *feedfoward* dan *backward* sampai kondisi pelatihan berhenti terpenuhi, setelah dilakukan pelatihan maka akan didapatkan bobot-bobot hasil pelatihan.

2.4.2.2 Pengujian *Backpropagation*

Tahap pengujian dengan menggunakan algoritma *backpropagation* menggunakan proses *feedfoward*. Bobot yang dihasilkan pada tahap pelatihan akan digunakan untuk proses pengujian.



Gambar 2.6 Blok Diagram Proses Pengujian dengan BPNN

Dapat dilihat pada Gambar 2.6 masukan dari sistem berupa paket data jaringan bernilai normal, lalu sistem akan mengambil bobot hasil pelatihan sebelumnya, setelah itu akan dilakukan klasifikasi dengan BPNN, dan proses selanjutnya sistem akan menghitung akurasi dari klasifikasi tersebut. Keluaran dari proses ini adalah hasil klasifikasi dan tingkat akurasi dari algoritma BPNN.

2.4.3 *Backpropagation* dan *Adaptive Boost*

Proses yang berjalan pada pelatihan dan pengujian ini menggunakan algoritma *backpropagation* sebagai *weak classifier* dan *AdaBoost* untuk membangun *strong classifier*. Jumlah *neuron* pada arsitektur *backpropagation* yang digunakan pada penelitian ini dapat dilihat pada Gambar 2.4. Jumlah *weak classifier* yang digunakan akan disesuaikan selama proses pelatihan, parameter jumlah *neuron* pada *input layer*, *hidden layer*, dan *output layer* akan disesuaikan berdasarkan arsitektur *backpropagation* pada Gambar 2.4.

2.4.3.1 Pelatihan Backpropagation dan AdaBoost

Proses yang dilakukan pada tahap pelatihan dengan menggunakan *backprpagation* dan *AdaBoost* adalah tahap inisialisasi bobot sampel ($D_{i,1}$) untuk setiap sampel, dimana i adalah indeks baris, pelatihan *backpropagation* menggunakan sampel terbobot, perhitungan *error weak classifier* (ε), perhitungan bobot *weak classifier* (α), dan memperbaharui matriks bobot D.



Gambar 2.7 Blok Diagram Proses Pelatihan BPNN dan *AdaBoost*

Dapat dilihat pada Gambar 2.7, masukannya berupa paket data jaringan bernilai normal, tahapan pertama adalah inisialisasi *neural networks* dan parameter pelatihan, tahapan kedua adalah inisialisasi bobot sampel, bobot *neural netowork* dimana pada penelitian ini nilai untuk bobot-bobot *neural network* berada pada rentang -0,5 sampai 0,5, tahapan ketiga adalah pelatihan dengan BPNN dan *AdaBoost* dimana akan dilakukan tahapan *feedfoward*, *backward*, dapatkan *weak classifier*, hitung *error weak classifier*, dan perbaharui bobot sampel, tahapan ini dilakukan sampai kondisi pelatihan berhenti terpenuhi, setelah dilakukan pelatihan maka akan didapatkan bobot-bobot hasil pelatihan.

2.4.3.2 Pengujian Backpropagation dan AdaBoost

Tahap pengujian BPNN dan *AdaBoost* menggunakan proses *feedfoward* pada BPNN untuk setiap putaran c. Setelah nilai putaran c = C, maka akan dilakukan proses *weighted voting* dari hasil klasifikasi *weak classifiers* untuk mendapatkan klasifikasi dari BPNN dan *AdaBoost.*



Gambar 2.8 Blok Diagram Proses Pengujian BPNN dan *AdaBoost*

Dapat dilihat pada Gambar 2.8 masukan dari sistem berupa paket data jaringan bernilai normal, lalu sistem akan mengambil bobot hasil pelatihan sebelumnya, setelah itu akan dilakukan klasifikasi dengan BPNN dan *AdaBoost*, dan proses selanjutnya sistem akan menghitung akurasi dari klasifikasi tersebut. Keluaran dari proses ini adalah hasil klasifikasi dan tingkat akurasi dari algoritma BPNN dan *AdaBoost*.

2.4.4 Hasil Pengujian

Pengujian ini dilakukan untuk mengklasifikasi 2 kelas yaitu normal dan serangan, dengan menggunakan total 2000 data diuji dengan metode 10-*fold cross validation.*

2.4.4.1 Skenario Pengujian 1

Pada skenario pengujian 1, pengujian metode BPNN dan BPNN dengan *AdaBoost* dilakukan dengan mengubah nilai jumlah *hidden layer* menjadi nilai 1, 3, 6, 8, 10, sedangkan variabel lainnya diisi dengan nilai yang sama. Berikut adalah nilai variabel lainnya yang digunakan pada skenario pengujian 1, nilai *learning rate* adalah 0.1, *Epoch* 1000, Target *Error* 0.05, *Weak Classifier* 3.

Dari skenario pengujian 1 didapatkan hasil secara keseluruhan yang dapat dilihat pada Tabel 2.1.

Tabel 2.1 Data Hasil Skenario Pengujian 1

Neuron	Akurasi BPNN	MSE BPNN	Akurasi BPNN dengan *AdaBoost*	MSE BPNN dengan *AdaBoost*
1	93,15%	0,490	93,2%	0,108
3	99,25%	0,490	99,4%	0,128
6	99,2%	0,543	99,5%	0,139
8	99,4%	0,494	99,65%	0,136
10	99,25%	0.511	99,5%	0.131

Dari data hasil skenario pengujian 1 ini dapat diambil kesimpulan bahwa jumlah *neuron* yang memiliki akurasi paling tinggi baik pada BPNN atau BPNN dengan *AdaBoost* pada kasus ini adalah 8 *neuron* dengan akurasi sebesar 99.4% untuk BPNN dan 99,65% untuk BPNN dengan *AdaBoost*.

2.4.4.2 Skenario Pengujian 2

Pada skenario pengujian 2, pengujian metode BPNN dan BPNN dengan *AdaBoost* dilakukan dengan mengubah nilai jumlah *learning rate* menjadi nilai 0.05, 0.1, 0.15, 0.2, 0.25, sedangkan variabel lainnya diisi dengan nilai yang sama. Berikut adalah nilai variabel lainnya yang digunakan pada skenario pengujian 2.Jumlah *neuron* pada *hidden layer* adalah 8, *Epoch* 1000, Target *Error* 0,05, *Weak Classifier* 3.

Dari skenario pengujian 2 didapatkan hasil secara keseluruhan yang dapat dilihat pada Tabel 2.2.

Tabel 2.2 Data Hasil Skenario Pengujian 2

Learning Rate	Akurasi BPNN	MSE BPNN	Akurasi BPNN dengan *AdaBoost*	MSE BPNN dengan *AdaBoost*
0,05	99,15%	0.490	99,5%	0.136
0,1	99,25%	0.490	99,4%	0.128
0,15	93,3%	0.543	93,3%	0.107
0,2	99,65%	0.494	99,55%	0.144
0,25	99,75%	0.518	99,8%	0.112

Dari data hasil skenario pengujian 2 ini dapat diambil kesimpulan bahwa *learning rate* yang memiliki akurasi paling tinggi baik pada BPNN atau BPNN dengan *AdaBoost* pada kasus ini adalah 0,25 dengan akurasi sebesar 99,75% untuk BPNN dan 99,8% untuk BPNN dengan *AdaBoost*.

3. PENUTUP

Berdasarkan hasil pengujian yang telah dilakukan, maka diperoleh kesimpulan bahwa metode BPNN dan BPNN dengan *AdaBoost* yang diimplementasikan kedalam sistem pengklasifikasi paket data jaringan memiliki akurasi tertinggi sebesar adalah 99,75% untuk BPNN dan 99,8% untuk BPNN dengan *AdaBoost*. Akurasi ini didapatkan dari susunan variabel yang memiliki jumlah *neuron* sebanyak 8 *neuron*, *learning rate* bernilai 0,25, batas *epoch* sebesar 1000, target *error* bernilai 0,05, dan jumlah *weak classifier* adalah 3.

Adapun saran yang diajukan yaitu penambahan *dataset* yang digunakan dalam penelitian, karena degan semakin beragam dataset yang digunakan maka semakin baik juga metode BPNN dan BPNN dengan *AdaBoost* dalam klasifikasinya.

DAFTAR PUSTAKA

[1] Harshal A. Sonawane dan Tareek M. Pattewar, "A Comparative Performance Evaluation of Intrusion Detection based on Neural Network and PCA," *IEEE ICCSP 2015 conference*, 2015.

[2] Jungsuk Song, Hiroki Takakura, Yasuo Okabe, Masashi Eto, Daisuke Inoue, dan Koji Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation," *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security - BADGERS*, 2011.

[3] Gert Schauwers dan Gert DeLaet, *Network Security Fundamentals*. California: Cisco Press, 2004.
[4] Nutan Farah Haq, Abdur Rahman Onik, Avishek Khan Hridoy, Musharrat Rafni, Faisal Muhammad Shah, dan Dewan Farid, "Application of Machine Learning Approaches in Intrusion Detection System: A Survey," *International Journal of Advanced Research in Artificial Intelligence*, vol. 4, no. 3, hal. 9–18, 2015.
[5] Indraneel Mukhopadhyay, Mohuya Chakraborty, S. Chakrabarti, dan Tanusree Chatterjee, "Back propagation neural network approach to Intrusion Detection System," *2011 International Conference on Recent Trends in Information Systems*, 2011.
[6] Gang Wang, Jinxing Hao, Jian Ma, dan Lihua Huang, "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering," *Expert Systems with Applications*, 2010.
[7] Weiming Hu, Wei Hu, dan Stephen J. Maybank, "AdaBoost-based algorithm for network intrusion detection," *IEEE Transactions on Systems, Man, and Cybernetics*, 2008.
[8] Holger Schwenk dan Yoshua Bengio, "Boosting Neural Networks," *Neural Computation*, 2000.
[9] Jianfang Cao, Junjie Chen, dan Haifang Li, "An AdaBoost-backpropagation neural network for automated image sentiment classification," *Scientific World Journal*, 2014.
[10] Prateek Joshi, *Python Machine Learning Cookbook*. Birmingham: Packt Publishing, 2016.
[11] Jiawei Han, Micheline Kamber, dan Jian Pei, *Data Mining: Concepts and Techniques*. Massachusetts: Morgan Kaufmann, 2012.
[12] Raul Rojas, "Neural Networks A Systematic Introduction," *Neural Networks*, 1996.
[13] Feng Shen, Xingchao Zhao, Dao Lan, dan Limei Ou, "A Hybrid Model of AdaBoost and Back-Propagation Neural Network for Credit Scoring," *Proceedings of the Eleventh International Conference on Management Science and Engineering Management*, 2018.
[14] S. Sahu dan B. M. Mehtre, "Network Intrusion Detection System Using J48 Decision Tree," *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, 2015.