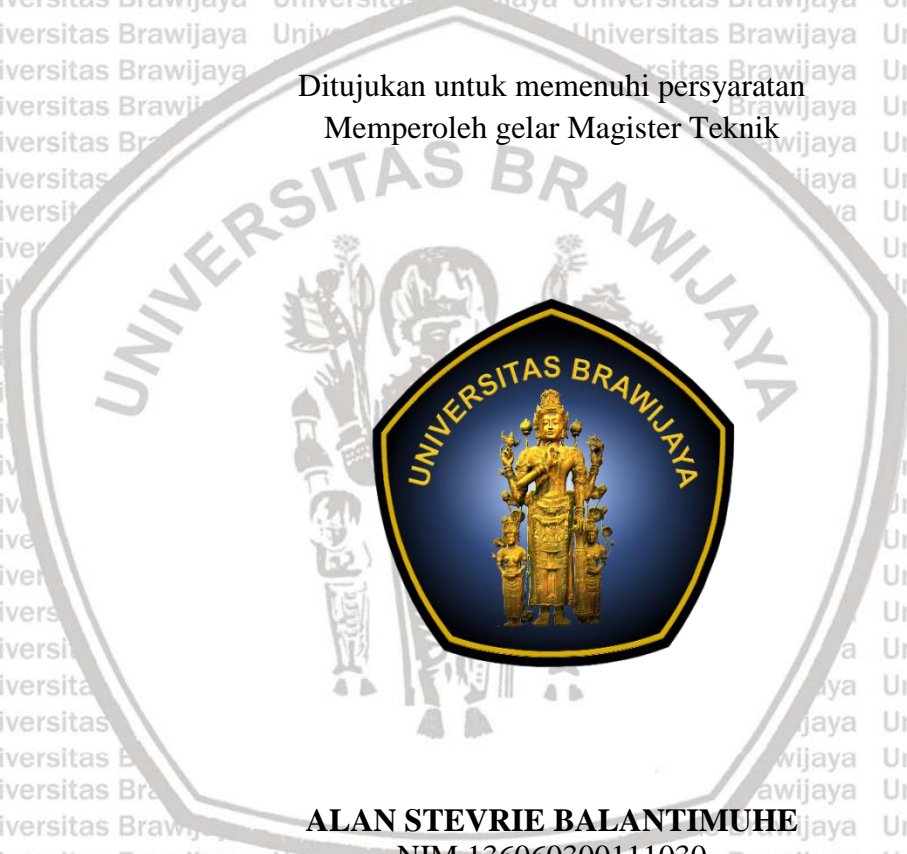


**PENERAPAN KONSOLIDASI BEBAN KERJA KLUSTER WEB  
SERVER SECARA DINAMIS DENGAN MELAKUKAN KLASIFIKASI  
BEBAN KERJA SERVER MENGGUNAKAN PENDEKATAN  
BACKPROPAGATION NEURAL NETWORK**

**TESIS**

**PROGRAM MAGISTER TEKNIK ELEKTRO  
MINAT SISTEM KOMUNIKASI DAN INFORMATIKA**

Ditujukan untuk memenuhi persyaratan  
Memperoleh gelar Magister Teknik



**ALAN STEVRIE BALANTIMUHE**  
NIM 136060300111030

**UNIVERSITAS BRAWIJAYA**  
**FAKULTAS TEKNIK**  
**MALANG**  
**2018**



**TESIS**

**PENERAPAN KONSOLIDASI BEBAN KERJA KLUSTER *WEB SERVER* SECARA  
DINAMIS DENGAN MELAKUKAN KLASIFIKASI BEBAN KERJA *SERVER*  
MENGUNAKAN PENDEKATAN BACKPROPAGATION NEURAL NETWORK**

**Alan Stevrie Balantimuhe**

**NIM. 136060300111030**

telah dipertahankan didepan penguji  
Pada tanggal .....  
dinyatakan telah memenuhi syarat  
untuk memperoleh gelar Magister Teknik

**Komisi Pembimbing,**

Pembimbing I,

Pembimbing II,

Dr. Ir. Sholeh Hadi Pramono, M.S.

Hadi Suyono, S.T., M.T., Ph.D., IPM.

NIP. 19580728 198701 1 001

NIP. 19730520 200801 1 013

Malang, .....

Universitas Brawijaya

Fakultas Teknik, Jurusan Teknik Elektro

Ketua Program Magister Teknik Elektro

Dr. Eng. Panca Mudjirahardjo, ST., MT

NIP. 19700329 200012 1 001



## PERNYATAAN ORISINALITAS PENELITIAN TESIS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas di dalam Naskah Tesis ini adalah asli dari pemikiran saya. Tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu Perguruan Tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah Tesis ini dapat dibuktikan terdapat unsur-unsur jiplakan, saya bersedia Tesis dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, pasal 25 ayat 2 dan pasal 70).

Malang, Juli 2018

Mahasiswa,

Alan Stevrie Balantimuhe

NIM. 136060300111030



**IDENTITAS TIM PENGUJI TESIS**

**JUDUL TESIS** : Penerapan Konsolidasi Beban Kerja Kluster *Web server* Secara Dinamis Dengan Melakukan Klasifikasi Beban Kerja *Server* Menggunakan Pendekatan Backpropagation Neural Network

**Nama Mahasiswa** : Alan Stevie Balantimuhe

**NIM** : 136060300111030

**Program Studi** : Magister Teknik Elektro

**Minat** : Sistem Komunikasi dan Informatika

**KOMISI PEMBIMBING**

**Ketua** : Dr. Ir. Sholeh Hadi Pramono, M.S.

**Anggota** : Hadi Suyono, S.T., M.T., Ph.D., IPM.

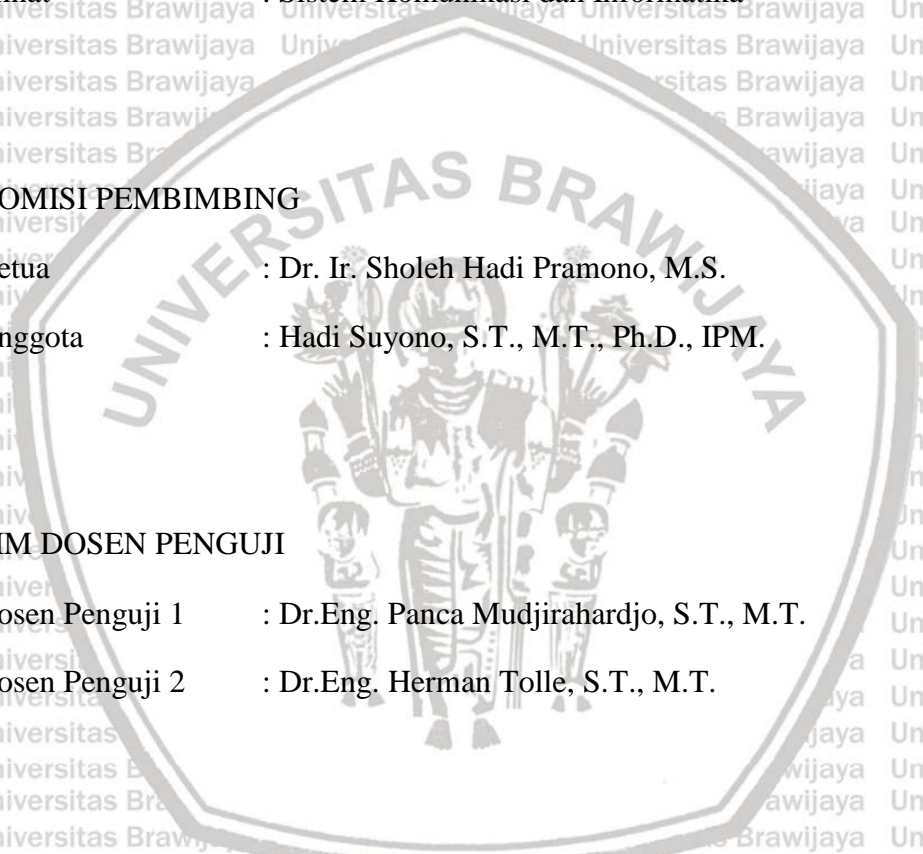
**TIM DOSEN PENGUJI**

**Dosen Penguji 1** : Dr.Eng. Panca Mudjirahardjo, S.T., M.T.

**Dosen Penguji 2** : Dr.Eng. Herman Tolle, S.T., M.T.

**Tanggal Ujian** : 27 Juli 2018

**SK Penguji** : 1625 Tahun 2018



UNIVERSITAS BRAWIJAYA



*Karya ini ku persembahkan untuk  
Seluruh rekan, sahabat, saudara dan keluarga*

## RIWAYAT HIDUP

Alan S Balantimuhe, Kaimana, 22 Juli 1985, anak pertama dari Tephilus Balantimuhe dan Sri Yani Idrawanti Hassanoesi, SDN 1 Sorong Papua dan SLTP Don Bosco Sorong Papua lulus tahun 2000, SMU Kristen 1 Tomohon lulus tahun 2003. Studi S1 Teknik Elektro di Universitas Brawijaya Malang lulus pada tahun 2010. Pengalaman kerja sebagai Koordinator Bidang Infrastruktur TI pada unit TIK Universitas Brawijaya Malang sampai sekarang. Melanjutkan studi program Magister (S2) di Program Magister Teknik Elektro Jurusan Elektro Fakultas Teknik Universitas Brawijaya pada tahun 2013-2018.

Malang, Juli 2018

Penulis

UNIVERSITAS BRAWIJAYA



## UCAPAN TERIMA KASIH

Dalam penyelesaian penelitian tesis ini, penulis banyak mendapatkan bantuan dari berbagai pihak. Untuk itu penulis menyampaikan ucapan terima kasih setulusnya kepada:

1. Dr. Ir. Sholeh Hadi Pramono, M.S. selaku pembimbing utama yang senantiasa memberikan arahan dan garis besar di setiap bimbingan sehingga benar-benar menyalakan semangat penulis dalam penelitian tesis ini.
2. Hadi Suyono, S.T., M.T., Ph.D. selaku pembimbing kedua yang selalu aktif memberikan masukan-masukan teknis sehingga esensi penelitian tesis ini benar-benar muncul ke permukaan.
3. Segenap Sivitas Akademika Fakultas Teknik Jurusan Teknik Elektro Universitas Brawijaya Malang.
4. Seluruh rekan kerja di Unit TIK Universitas Brawijaya yang selalu memberikan support dan semangat.
5. Seluruh keluarga tercinta yang selalu memberikan dukungan dan pengertian dalam proses pengerjaan.

Malang, Juli 2018

Penulis

## RINGKASAN

**Alan S Balantimuhe**, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juli 2018, *Penerapan Konsolidasi Beban Kerja Kluster Web server Secara Dinamis Dengan Melakukan Klasifikasi Beban Kerja Server Menggunakan Pendekatan backpropagation*, Dosen Pembimbing: Sholeh Hadi Pramono dan Hadi Suyono.

Meningkatnya permintaan pengguna aplikasi WWW telah menyebabkan peningkatan yang sepadan dalam penggunaan sumber daya kluster server web. Penelitian ini mengkaji tentang penyediaan sumber daya web server berdasarkan parameter beban kerja server (load average CPU). Data yang digunakan adalah akses terhadap web server yang melayani aplikasi Sistem Informasi Akademik Mahasiswa Universitas Brawijaya (SIAM-UB). Penggunaan sumber daya server secara maksimal (beban puncak) terjadi pada periode registrasi mahasiswa, yaitu lebih dari 65000 mahasiswa akan mengakses server SIAM secara bersamaan. Jumlah permintaan yang dilayani server dalam 1 hari dapat mencapai 1.7juta permintaan. Pada penelitian ini, dilakukan prediksi (klasifikasi) konsolidasi beban kerja CPU dalam kluster web server untuk penyediaan sumber daya server yang optimal. Prediksi konsolidasi beban kerja server diklasifikasikan menjadi 3 kelas, yaitu: Min (0-2), Medium (3-6), Maximum ( $n > 7$ ). Metode *backpropagation neural network* (BNN) digunakan untuk memprediksi kelas konsolidasi beban kerja server berdasarkan parameter input penggunaan CPU, memory, jaringan (throughput) dan jumlah IP akses. Arsitektur BNN dengan 32 input, 2 hidden layer dengan jumlah neuron  $h_1$  512;  $h_2$  32, 3 output, dan learning rate 0.00001, menghasilkan bobot yang mampu melakukan klasifikasi konsolidasi beban kerja CPU dengan tingkat precision 90%, tingkat sensitivity 0.9, dan tingkat akurasi 93%.

Kata kunci : Klasifikasi, prediksi, konsolidasi, *Backpropagation neural network*, load average, cluster web server.



## SUMMARY

**Alan S Balantimuhe**, *Department of Electrical Engineering, Faculty of Engineering, University of Brawijaya, July 2018, Consolidation of Dynamic Web Server Workload using Backpropagation Neural Network Approach: Sholeh Hadi Pramono dan Hadi Suyono.*

*The increasing demand for users of WWW applications has led to a commensurate increase in the use of cluster server resources. This study examines the provision of web server resources based on server workload parameters (load average CPU). Researchers conduct research on the cluster web server that serves Academic Student Information System of Universitas Brawijaya (SIAM-UB) application. Maximum use of server resources (peak load) occurs during the student registration period, more than 65000 students will access the SIAM server simultaneously. The number of requests served by the server in 1 day can reach 1.7 million requests. In this research, the server resources is predicted to determine the optimal CPU load average in web server cluster. The server workload predictions are classified into 3 classes, namely: Min (0-2), Medium (3-6), Maximum ( $n > 7$ ). The backpropagation neural network (BNN) method, is then used to predict the server workload class based on the following parameters such as CPU usage, memory, network (throughput) and number of IP access to SIAM cluster server. The BNN architecture with the 32 inputs, 2 hidden layers including the number of neuron  $h1$  512 and  $h2$  32, 3 outputs, and learning rate of 0.00001, have produced the weights that can perform the classification consolidation work load with the precision level of 90%, the sensitivity level of 0.9, and the accuracy level of 93%.*

**Keywords** : *Classification, Prediction, Consolidation, Backpropagation neural network, load average, cluster web server.*

## KATA PENGANTAR

Puji syukur kepada Tuhan Yesus. Tanpa karunia dan pernyertaan-Nya, tidak mungkin laporan Tesis ini dapat diselesaikan.

Dalam tulisan ini, penulis mengacu pada ketentuan pembuatan tesis yang berlaku pada Program Master dan Doktor Fakultas Teknik Universitas Brawijaya Malang. Dalam melengkapi laporan Tesis ini, digunakan berbagai macam referensi penunjang yang terkait dengan permasalahan yang akan diteliti. Penulis menyadari bahwa proposal ini dapat diselesaikan karena bantuan dan dukungan dari berbagai pihak terutama kedua pembimbing, untuk itu penulis menghaturkan banyak terimakasih, khususnya kepada :

1. Papa, Mama, Papa Bambang, Mama Nia, istriku tercinta Riche Avianty, anakku tersayang Kiel dan Klei, adik-adik terkasih : Amelia Indra Presty, Ariesa Eva Sari, Ariel Rizki Putra Balantimuhe, Samuel Aji Sena, atas doa dan kasih sayang yang telah diberikan selama ini dan juga atas dukungan yang tak henti-hentinya pada penulis untuk menyelesaikan studi di Teknik Elektro UB dengan baik.
2. Bapak Dr. Ir. Sholeh Hadi Pramono, M.S., dan Bapak Hadi Suyono, S.T., M.T., Ph.D., IPM. selaku pembimbing yang telah banyak meluangkan waktunya membimbing, mengarahkan, dan atas dukungan pada penulis untuk menyelesaikan studi di Teknik Elektro UB dengan baik.
3. Bapak Achmad Basuki, ST., M.MG., Ph.D. selaku Kepala UPT-TIK UB yang telah mengizinkan penulis untuk mengambil data dari UB, atas diskusi-diskusi yang bermanfaat dan atas dukungan pada penulis untuk menyelesaikan studi di Teknik Elektro UB dengan baik.
4. Bapak R. Arief Setyawan, ST., M.T. selaku Kepala Divisi Data dan Infrastruktur UPT-TIK UB atas dukungan pada penulis untuk menyelesaikan studi di Teknik Elektro UB dengan baik.
5. Rekan-rekan kerja di Unit TIK Universitas Brawijaya Malang dan khususnya di Bidang Infrastruktur : Ahmad Mukhtarom, Abdurrohman Muhammad, Kharisma Kurnarakarta, Rachmad Tsalaatsa, dan Iwanto, yang telah memberikan dukungan dan semangat kepada penulis.

6. Rekan-rekan mahasiswa Strata Dua (S2) Program Studi Teknik Elektro Universitas Brawijaya Malang, yang telah memberikan dukungan moril kepada penulis.

Akhirnya, semoga penelitian tesis ini mendapatkan perhatian dari semua pihak.

Malang, Juli 2018

Alan Stevie Balantimuhu



DAFTAR ISI

Halaman

KATA PENGANTAR..... i

DAFTAR ISI ..... iii

DAFTAR TABEL..... v

DAFTAR GAMBAR..... vi

BAB 1 PENDAHULUAN..... 1

1.1 Latar Belakang..... 1

1.2 Rumusan Masalah ..... 2

1.3 Batasan Masalah..... 2

1.4 Tujuan Penelitian..... 3

1.5 Manfaat Penelitian..... 3

BAB 2 LANDASAN TEORI..... 5

2.1 Penelitian Relevan ..... 5

2.1.1 Teknik Kosolidasi Statis..... 5

2.1.2 Teknik Konsoldasi Dinamis ..... 5

2.1.3 Teknik Konsolidasi Dinamis dengan Prediksi Beban ..... 6

2.1.4 *Considering Hardware Utilization*..... 7

2.2 Kluster *Computing*..... 9

2.3 *Load Balancing* ..... 10

2.4 CPU *Load average* ..... 11

2.5 Neural Network ..... 11

2.5.1 Analogi Otak Manusia..... 12

2.5.2 *Artificial Neuron* ..... 13

2.5.3 Fungsi Aktivasi..... 14

2.6 *Artificial Neural Network (ANN)*..... 15

2.6.1 Arsitektur ANN ..... 15

2.6.2 Network Umpan-maju *Backpropagation* ..... 19

2.7 *Confusion matrix*..... 22

BAB 3 KERANGKA KONSEP PENELITIAN..... 25

3.1 Kerangka Konsep Penelitian ..... 25

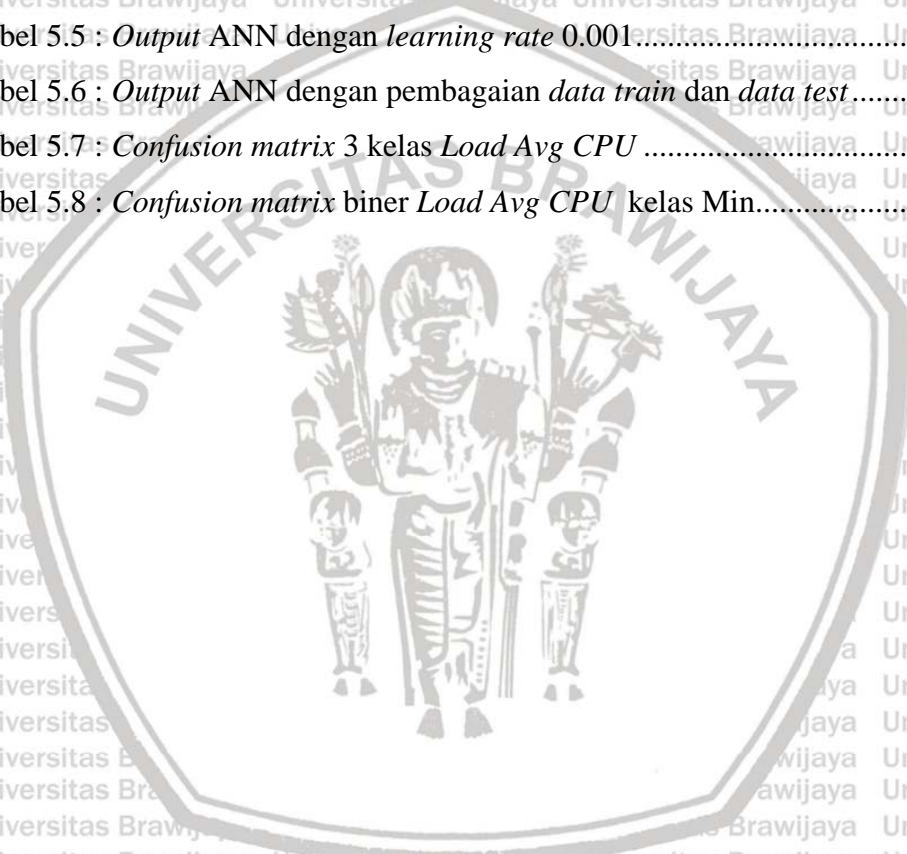
3.2 Data Penelitian..... 25

3.3 Hipotesis..... 26

<b>BAB 4 METODE PENELITIAN</b> .....	27
4.1 Metode Penelitian .....	27
4.2 Pengambilan Data .....	28
4.3 <i>Preprocessing</i> .....	28
4.4 Pelatihan <i>Backpropagation Neural Network</i> .....	29
4.5 Pengujian .....	31
4.6 Evaluasi Hasil .....	31
4.7 Analisis .....	32
<b>BAB 5 HASIL DAN PEMBAHASAN</b> .....	33
5.1 Pengumpulan Data .....	33
5.1.1 Proses Pembelajaran .....	33
5.2 Pengujian .....	33
5.2.1 Pelatihan dengan 1 <i>Hidden layer</i> .....	34
5.2.2 Pelatihan dengan 2 <i>Hidden layer</i> .....	34
5.2.3 Pelatihan berdasarkan <i>learning rate</i> .....	35
5.2.4 Pelatihan Berdasarkan Pembagaaian Data Training dan Data Testing .....	36
5.3 Evaluasi Hasil Pengelompokkan .....	36
5.4 Evaluasi Penentuan Jumlah Server (Max Server = 5) .....	38
<b>BAB 6 KESIMPULAN DAN SARAN</b> .....	39
6.1 Kesimpulan .....	39
6.2 Saran .....	40
<b>DAFTAR PUSTAKA</b> .....	41

DAFTAR TABEL

No.	Judul	Halaman
Tabel 2.1 :	Ringkasan Penelitian .....	8
Tabel 2.2 :	<i>Confusion matrix</i> kelas biner untuk klasifikasi Min. <i>Load Avg CPU</i> .....	23
Tabel 5.1 :	<i>Output</i> ANN dengan 1 <i>hidden layer</i> .....	34
Tabel 5.2 :	<i>Output</i> ANN dengan 2 <i>hidden layer</i> .....	35
Tabel 5.3 :	<i>Output</i> ANN dengan <i>learning rate</i> 0.0001 .....	35
Tabel 5.4 :	<i>Output</i> ANN dengan <i>learning rate</i> 0.000001 .....	35
Tabel 5.5 :	<i>Output</i> ANN dengan <i>learning rate</i> 0.001 .....	36
Tabel 5.6 :	<i>Output</i> ANN dengan pembagaian <i>data train</i> dan <i>data test</i> .....	36
Tabel 5.7 :	<i>Confusion matrix</i> 3 kelas <i>Load Avg CPU</i> .....	37
Tabel 5.8 :	<i>Confusion matrix</i> biner <i>Load Avg CPU</i> kelas Min .....	37



DAFTAR GAMBAR

No.	Judul	Halaman
	<i>Gambar 2.1 : Arsitektur Kluster Computing</i> .....	9
	<i>Gambar 2.2 : Arsitektur Load Balancing</i> .....	10
	<i>Gambar 2.3 : Model Neural Network</i> .....	12
	<i>Gambar 2.4 : Neuron secara biologis</i> .....	13
	<i>Gambar 2.5 : Neuron ANN dan fungsi transfer</i> .....	13
	<i>Gambar 2.6 : Sistem Non-linear multi-dimensi</i> .....	15
	<i>Gambar 2.7 : ANN fully-connected dan bobot matrix</i> .....	15
	<i>Gambar 2.8 : Multilayer perceptron dan bobot matrix</i> .....	16
	<i>Gambar 2.9 : Perceptron</i> .....	17
	<i>Gambar 2.10 : Pembaruan bobot berdasarkan gradient descent</i> .....	19
	<i>Gambar 2.11 : Struktur jaringan single-layer umpan-maju</i> .....	20
	<i>Gambar 2.12 : Jaringan single-layer tidak mampu melakukan klasifikasi fungsi non-linear</i> .....	20
	<i>Gambar 2.13 : Struktur jaringan MLP umpan-maju dengan satu hidden layer</i> .....	21
	<i>Gambar 2.14 : Kemampuan MLP melakukan klasifikasi fungsi non-linear</i> .....	21
	<i>Gambar 2.15 : Pelatihan jaringan dengan algoritme backpropagation</i> .....	22
	<i>Gambar 3.1 : Diagram Alir Metodologi Penelitian</i> .....	25
	<i>Gambar 4.1 : Metodologi Penelitian</i> .....	27
	<i>Gambar 4.2 : Struktur Neural Network</i> .....	29
	<i>Gambar 5.1 : Hasil simulasi jumlah server dengan penerapan konsolidasi beban kerja prediktif</i> .....	38

## BAB 1 PENDAHULUAN

### 1.1 Latar Belakang

Teknologi *World Wide Web* (WWW) berkembang dengan sangat cepat dan memainkan peran yang penting dalam kehidupan sehari-hari. Pertumbuhan jumlah akses www yang menggunakan teknologi ini terus meningkat. Isu penting dengan pertumbuhan yang terus berlanjut ini sangat berkaitan dengan kinerja dari *web server* yang harus menyediakan sumber daya yang dapat diandalkan, diprediksi, terukur dan efisien. Terkait dengan jumlah akses ke *web server* ada kondisi *server* mengalami beban puncak sehingga untuk menangani hal ini, perlu beberapa *server* yang melayani yang disebut dengan nama kluster *server*. Namun pada waktu tertentu ada kondisi beban puncak tidak terjadi sehingga untuk menangani hal ini perlu dilakukan penyesuaian kembali jumlah *server* untuk melayani. Kondisi ini terjadi secara dinamis.

Penelitian-penelitian sebelumnya telah mencoba untuk menyelesaikan permasalahan permintaan jumlah akses yang tinggi menggunakan pendekatan teknik *load balancing* (Setyawan 2014) yaitu dengan mekanisme penjadwalan *round-robin* dan *source hash*, namun demikian pendekatan ini masih belum mempertimbangkan penyediaan sumber daya *server* yang digunakan ketika kondisi dinamis seperti dijelaskan diatas.

Penelitian ini dilakukan untuk melakukan prediksi (kondisi dinamis) kebutuhan sumber daya *server web* dalam sebuah kluster sesuai dengan jumlah request atau beban kerja yang diterima (efisien dan optimal) dengan melakukan klasifikasi beban kerja *server*.

Penelitian ini mencoba untuk melakukan kajian terhadap beban kerja *web server* dan penyediaan sumber daya *web server* itu sendiri. Dari hasil kajian diharapkan dapat mengukur dan memprediksi kinerja beban *server* dan menyediakan sumber daya *web server* yang efisien dan optimal. Dalam melakukan kajian terhadap beban kerja dan penyediaan sumber daya *web server*, peneliti melakukan penelitian terhadap *web server* yang melayani aplikasi Sistem Informasi Akademik Mahasiswa (SIAM) UB.



Dalam melakukan prediksi konsolidasi beban kerja *server* untuk mengatur penyediaan sumber daya *server* yang optimal, dilakukan pengukuran parameter-parameter yang mempengaruhi kinerja dan penggunaan sumber daya kluster *web server*. Kemudian untuk mempermudah penentuan konfigurasi kluster *web server*, semua parameter tersebut akan dikelompokkan menjadi 3 kelas/kelompok beban *server*, yaitu: beban kerja minimum, beban kerja medium, beban kerja maximum. Pada penelitian ini akan digunakan *backpropagation neural network* sebagai *classifier* yang masukannya adalah semua parameter pada setiap kluster *web server* dan diharapkan dapat menghasilkan prediksi kelas beban yang sesuai dan memberikan rekomendasi konfigurasi *server* yang optimal. Dengan membuat 3 kelas atau kelompok beban kerja diharapkan kebutuhan-kebutuhan terkait pengaturan konfigurasi *server* untuk mendapat kinerja yang efisien dan optimal dapat terpenuhi.

## 1.2 Rumusan Masalah

Berdasarkan uraian latar belakang, maka masalah yang menjadi fokus dalam penelitian ini adalah bagaimana merancang dan membuat sistem yang dapat melakukan prediksi beban kerja sebuah kluster *server* dan dapat memberikan rekomendasi jumlah *server* dalam sebuah kluster yang perlu disediakan secara dinamis untuk menerima beban kerja yang dinamis.

Secara lebih detail, rumusan masalah dalam penelitian ini dapat diuraikan sebagai berikut :

1. Bagaimana pengukuran beban kerja dari masing-masing *server* dalam hal ini *virtual machine* (VM) di sebuah kluster *server*?
2. Bagaimana penerapan *backpropagation neural network* untuk mengklasifikasikan konsolidasi beban kerja *server* untuk penyediaan sumber daya *server* yang optimal dalam sebuah kluster *server web*?

## 1.3 Batasan Masalah

Berdasarkan uraian rumusan masalah, maka ruang lingkup dalam penelitian ini dibatasi pada :

1. Tidak membahas algoritme yang digunakan aplikasi SIAM.
2. Studi kasus dilakukan pada kluster *web server* untuk aplikasi SIAM.
3. Penggunaan data yang digunakan dalam penelitian ini adalah data log akses kluster *web server* pada bulan Agustus 2017 dan Januari 2018.



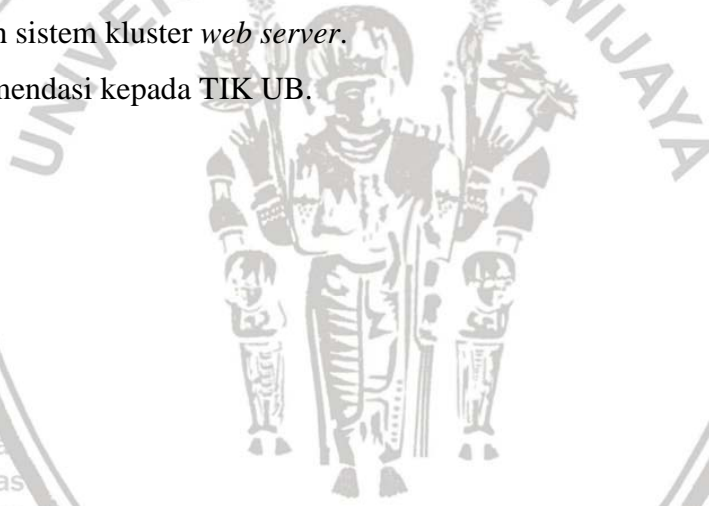
#### 1.4 Tujuan Penelitian

Tujuan dalam penulisan penelitian ini adalah merancang dan membuat model yang dapat menentukan kinerja optimal suatu system kluster *web server* berdasarkan prediksi konsolidasi beban kerja dengan menggunakan pendekatan *backpropagation neural network*.

#### 1.5 Manfaat Penelitian

Manfaat yang ingin dicapai dari penelitian ini adalah:

1. Optimasi kinerja dari sebuah sistem kluster *web server* yang didasarkan pada beban kerja dengan menentukan jumlah *server* yang melayani secara dinamis.
2. Bagi pihak-pihak lain yang ingin merancang dan membuat model berdasarkan *backpropagation neural network* untuk melakukan klasifikasi dalam beban kerja *server* dan menentukan penjadwalan kerja *server* sesuai dengan beban kerja yang diterima dalam sebuah sistem kluster *web server*.
3. Rekomendasi kepada TIK UB.





## BAB 2

### LANDASAN TEORI

#### 2.1 Penelitian Relevan

##### 2.1.1 Teknik Konsolidasi Statis

Dalam konsolidasi *server* statis, pemetaan VM-to-PM tidak diubah untuk waktu yang lama, dan tidak ada migrasi yang dilakukan dengan perubahan beban kerja selama waktu itu (T. C. Ferreto 2011). Keuntungan dari pendekatan ini adalah dalam pemrosesan pekerjaan batch dan aplikasi dengan penggunaan yang konsisten. Kerugian dari konsolidasi statis adalah terjadi *over-provisioning* sumber daya. Dalam konsolidasi statis, sumber daya dialokasikan dengan cara memenuhi tuntutan beban puncak, dan akibatnya sumber daya terbuang sebagian besar ketika *virtual machine* (VM) tidak bekerja pada beban puncaknya mirip dengan kasus *data center* tradisional. Halder dkk. di (K. Halder 2012) memperkenalkan sebuah algoritme menggunakan konsolidasi statis, yang mencoba untuk menghasilkan penempatan awal dan jumlah sumber daya untuk VM mempertimbangkan konsumsi energi. Speitkamp dan Bichler dalam (Bichler 2010) menggunakan formulasi *bin-packing* multidimensi untuk memodelkan masalah konsolidasi *server* statis dan dinamis. Wolke dkk. dalam (A. Wolke 2016) berpendapat metode konsolidasi statis terhadap teknik alokasi sumber daya dinamis, yang dibandingkan langsung dalam percobaan di *data center*.

##### 2.1.2 Teknik Konsolidasi Dinamis

Dalam konsolidasi dinamis, algoritme konsolidasi dijalankan sebagai respons terhadap variasi beban kerja atau pada interval waktu tertentu dan dapat memutuskan untuk memigrasikan VM ke *physical machine* (PM) lain (G. D. A. Verma 2009). Sebagian besar studi dilakukan pada metode kedua, yaitu, konsolidasi dinamis. Seperti yang telah disebutkan sebelumnya, algoritme konsolidasi dinamis akan berjalan dalam periode waktu tertentu atau peristiwa tertentu, pilihan yang dapat memengaruhi efisiensi algoritme. Algoritme yang disajikan oleh Lovász et al. dalam (G. Lovász 2013) berjalan setiap 10 menit. Meningkatkan atau menurunkannya jangka waktu dapat mempengaruhi pemanfaatan sumber daya, konsumsi energi, dan kinerja *data center*.

Menjalankan algoritme dalam waktu singkat membuat perubahan *data center* terjadi lebih cepat, dan karenanya, *server* akan hidup dan mati lebih dari sebelumnya, dan akibatnya, masa hidup *server* akan berkurang. Selain itu, lebih banyak *bandwidth* harus dialokasikan untuk migrasi VM, dan lalu lintas jaringan pengguna yang sebenarnya akan menghadapi *bandwidth* yang tersedia lebih rendah. Di sisi lain, interval waktu yang lebih lama juga memiliki dampak negatif pada kinerja *data center*, misalnya, karena tidak bereaksi cukup cepat untuk perubahan beban kerja, *server* mungkin kelebihan beban, dan ini dapat mengurangi kinerja aplikasi dan mungkin melanggar perjanjian tingkat layanan (SLA). Selain itu, periode konsolidasi yang terlalu lama dapat kehilangan peluang penghematan energi, karena selama periode waktu, beberapa *server* berjalan pada mode siaga, dan sampai saat itu, *server* tetap berjalan dengan mengkonsumsi energi (J. J. Prevost 2013) (Wolke 2012) (V. Ebrahimrad 2015). Selain periode waktu, ada cara lain untuk menentukan waktu menjalankan algoritme, yaitu pada peristiwa tertentu.

Deng et al. di (W. Deng 2014) mendefinisikan pemicu untuk menjalankan algoritme. Definisi ini didasarkan pada parameter beban yang terdiri dari beberapa sumber daya multidimensi (yaitu, CPU, *memory*, *disk*, dan jaringan). Ketika parameter beban mencapai nilai tertentu, algoritme konsolidasi berjalan dan mencegah kemungkinan *server* dari kelebihan atau kekurangan sumber daya. Prevost et al. dalam (J. J. Prevost 2013) menyajikan model optimasi *stochastic*, yang menentukan frekuensi pembaruan optimal untuk mengubah pemetaan VM-to-PM.

### 2.1.3 Teknik Konsolidasi Dinamis dengan Prediksi Beban

Salah satu alasan utama konsumsi energi *data center* sangat tinggi adalah karena *server* online tetapi tidak aktif. Untuk menghemat daya, *server* harus dialihkan ke status daya yang lebih rendah saat tidak digunakan. Selain itu, mengalihkan *server* dari keadaan daya ke yang lain menyebabkan *delay* dan kelebihan energi. Oleh karena itu, jika *server* tidak akan diperlukan untuk waktu yang lama, perlu untuk mematikan *server* daripada menyalakannya dan menyebabkan energi dan waktu overhead yang tidak perlu. Faktanya ini meningkatkan kebutuhan teknik prediksi yang dapat digunakan untuk memperkirakan beban kerja *data center* di masa depan. Kita dapat menggunakan pendekatan teknik prediksi untuk memutuskan kapan dan untuk berapa lama *server* perlu dimatikan atau dihidupkan untuk memproses permintaan VM baru. Beberapa penelitian telah menggunakan teknik ini untuk menyediakan algoritme konsolidasi *server* yang efisien dan diinginkan. Xu et al. di (W. Xu 2006) menyajikan tiga algoritme prediksi yang

berbeda: model *autoregressive* (AR) standar, yang melakukan korelasi temporal antara nilai dari parameter waktu saat sekarang dan waktu sebelumnya; model gabungan ANOVA-AR yang menggabungkan metode AR dan menganalisa pola berulang jangka panjang dalam deret waktu; dan model *multi pulse* (MP). MP pertama kali digunakan dalam *speech processing*, yang menganalisis pola jangka panjang dan jangka pendek secara online. Gong et al. dalam (Z. Gong 2010) mengusulkan kerangka kerja *Predictive Elastic Resource Scaling* (PRESS). PRESS mencoba mengalokasikan sumber daya yang cukup untuk VMs dengan cara yang meminimalkan pelanggaran (*Service-level Agreement*) SLA dan penggunaan sumber daya yang berlebihan. PRESS memperhatikan kebutuhan sumberdaya VM yang dinamis dan melakukan prediksi kebutuhan sumber daya ini dalam waktu dekat menggunakan algoritme *Lightweight Signal Processing* dan teknik *statistical learning*. Namun, memprediksi beban kerja *data center* bisa sangat rumit dan menantang karena keragaman dan permintaan klien yang datang bersifat *stochastic*, setiap permintaan datang pada waktu yang berbeda dan meminta jumlah sumber daya yang berbeda (CPU, *memory*, *bandwidth*, dll.).

#### 2.1.4 *Considering Hardware Utilization*

Salah satu parameter yang paling banyak digunakan dalam algoritme konsolidasi *server* dan penyediaan sumber daya adalah pemanfaatan perangkat keras. Berbagai sumber daya perangkat keras (mis., CPU, *memory*, *disk*, dan *bandwidth*) dapat dipertimbangkan dalam pengoptimalan algoritme. Beberapa studi (M. Cardoso 2009), (C. Mastroianni 2011), (P. A. A. Verma 2008) hanya mempertimbangkan CPU untuk algoritme yang diusulkan, sedangkan dalam penelitian lain, jumlah sumber daya yang dipertimbangkan meningkat, yang berpotensi mengarah pada pemetaan yang lebih baik. Song dkk. di (Y. Song 2009) dan Gmach dkk. di (D. Gmach 2009) menganggap CPU dan *memory* sebagai parameter pengoptimalan. Selain itu, Beloglazov dan Buyya di (Buyya 2010) menganggap CPU, *memory*, dan pemanfaatan jaringan sebagai parameter optimasi, dan dalam (W. Deng 2014), Deng et al. juga menggunakan CPU, *memory*, jaringan, dan *disk*.

Dalam studi yang disebutkan, penulis menggunakan pemanfaatan sumber daya untuk memodelkan dan memecahkan masalah optimasi. Fox et al. di (A. Fox 2012) menggunakan 13 metrik yang berbeda untuk memodelkan kinerja VM dan *server*, dan berdasarkan itu, penyediaan sumber daya dilakukan. Selain itu, fungsi obyektif bisa menjadi pekerjaan yang berbeda dengan kerja, tetapi memaksimalkan pemanfaatan sumber daya ini adalah salah satu fungsi obyektif penting di tulis dalam (Y. C. Lee and A. Y.

Zomaya 2012), (S. K. Garg 2014). Namun, tujuannya tidak hanya 100% dari pemanfaatan, dan *server* harus memiliki cadangan sumber daya karena menoleransi ketidakstabilan beban kerja akibat dari jumlah permintaan pengguna yang diterima *server* secara stokastik.

Berikut adalah ringkasan penelitian (Tabel 2.1) yang relevan dalam bentuk tabular:

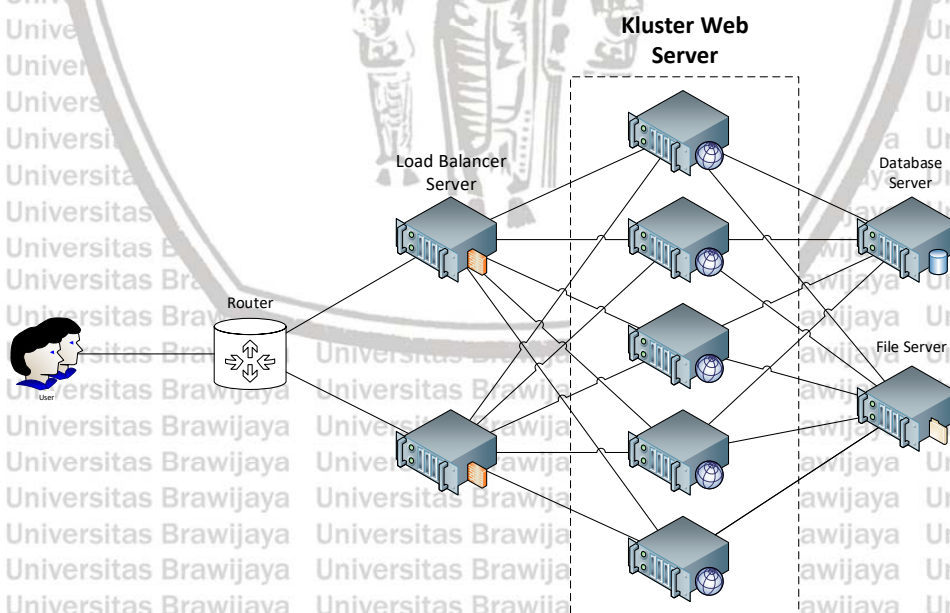
Tabel 2.1 Ringkasan Penelitian

No	Penulis	Judul	Metode	Hasil
1	T. C. Ferreto (2011)	<i>Server consolidation with migration control for virtualized data centers</i>	<i>LP formulation and heuristics</i>	menghindari migrasi VMs dengan kapasitas stabil mengurangi jumlah migrasi dengan meminimalkan penggunaan server fisik.
2	K. Halder (2012)	<i>Risk aware provisioning and resource aggregation based consolidation of virtual machines</i>	<i>Resource Aggregation correlation use matrix and FFD (First-Fit-Decreasing)</i>	Hasil eksperimen menunjukkan bahwa pendekatan mengarah ke sejumlah besar pengurangan dalam jumlah server (hingga 32% dalam pengaturan kami) yang diperlukan untuk meng-host 1000 VM dan dengan demikian memungkinkan kami untuk mematikan server yang tidak perlu.
3	(Bichler 2010)	<i>A Mathematical programming approach for server consolidation problems in virtualized data centers.</i>	<i>NP-hard</i>	penghematan server sebesar 31 persen hanya dapat dicapai dengan memperhatikan siklus dalam beban kerja server.
4	G. D. A. Verma (2009)	<i>Server workload analysis for power minimization using consolidation.</i>	<i>Consolidation methodologies CBP and PCP</i>	Evaluasi eksperimental menunjukkan bahwa PCP mencapai penghematan daya yang unggul, pelanggaran rendah dan keseimbangan beban yang baik di server aktif.
5	W. Xu (2006)	<i>Predictive control for dynamic resource allocation in enterprise data centers.</i>	<i>auto-regressive (AR) model, a combined ANOVA-AR model, multi-pulse model.</i>	hasil simulasi dan emulasi adalah pengontrol prediktif dapat menangani permintaan yang bervariasi waktu dengan cara yang lebih proaktif setelah pola permintaan dipelajari dan prediksi akurat, tetapi juga dapat menghasilkan kinerja yang buruk ketika kesalahan prediksi besar.

No	Penulis	Judul	Metode	Hasil
6	Z. Gong 2010	<i>PRESS: Predictive elastic resource scaling for cloud systems</i>	<i>light-weight signal processing and statistical methods.</i>	Hasilnya menunjukkan bahwa prediksi penggunaan sumber daya PRESS mencapai akurasi tinggi dan alokasinya mencapai profitabilitas penyedia layanan yang lebih baik daripada pendekatan lain di berbagai beban kerja.

## 2.2 Kluster Computing

Kluster *computing* adalah sistem yang memungkinkan komponennya untuk dilihat secara fungsional sebagai entitas tunggal, dari sudut pandang klien untuk layanan yang berjalan dan dari sisi pengelolaan. Sebuah kluster (Gambar 2.1) dapat dilihat sebagai serangkaian proses yang berjalan di beberapa komputer yang berbagi lingkungan yang sama. Ada korelasi yang sangat erat antara pengelompokan dan redundansi. Sebuah kluster menyediakan redundansi untuk suatu sistem dan dapat digunakan untuk sistem ketersediaan tinggi. Kluster *high availability* (juga dikenal sebagai Kluster HA atau *Failover Kluster*) adalah kluster komputer yang diimplementasikan untuk menyediakan ketersediaan layanan yang tinggi. Mereka beroperasi dengan memiliki komputer atau node redundan yang digunakan untuk menyediakan layanan ketika ada sebuah komponen dalam sistem gagal.



Gambar 2.1: Arsitektur Kluster Computing

Ketika beberapa contoh layanan identik tersedia, permintaan klien ke komponen ini dapat diseimbangkan. Cara ini memastikan bahwa semua instance aplikasi memiliki beban



kerja yang sama. Dengan mekanisme penyeimbangan beban yang berjalan di situs, semua instance bersifat *redundant*. Jika salah satu instance gagal, permintaan dapat secara otomatis dikirim ke instance yang bertahan hidup di kluster. Agar ini berfungsi, harus ada setidaknya satu komponen tambahan (komponen yang melebihi kapasitas layanan).

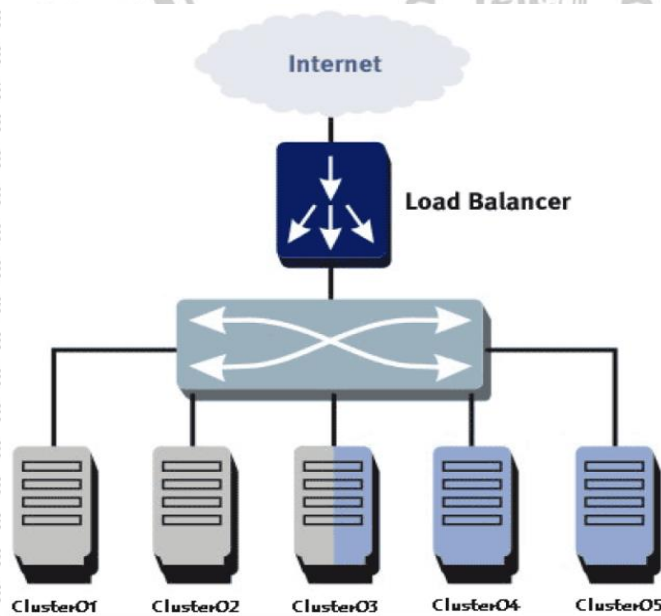
Pendekatan ini lebih murah dan lebih fleksibel daripada pendekatan *Failover* sebagai *Heartbeat*, di mana satu komponen langsung dipasangkan dengan satu komponen cadangan yang mengambil alih jika terjadi kegagalan. Sebuah analogi dalam teknologi RAID *disk controller*, menggunakan analogi RAID1 (*mirror*) dengan pendekatan "live / backup" untuk *Failover*, sedangkan analogi RAID5 sama dengan load balancing *Failover*.

Sistem komputasi kluster dapat digunakan untuk beberapa tujuan, seperti load balancing dan *Failover*.

Gambar 2.1 menunjukkan arsitektur kluster *computing* dengan menggunakan 2 unit perangkat *switch*.

### 2.3 Load Balancing

Dalam jaringan komputer, *load balancing* adalah teknik untuk menyebarkan pekerjaan antara dua atau lebih komputer dalam hal ini sebagai *server*, jaringan, CPU, *hard drive*, atau sumber daya lain, untuk mendapatkan pemanfaatan sumber daya yang lebih optimal, memaksimalkan *throughput*, dan meminimalkan waktu respons. Penggunaan berbagai komponen dan perangkat dengan teknik *load balancing*, dapat meningkatkan keandalan melalui redundansi. Layanan *load balancing* biasanya disediakan oleh program khusus atau perangkat keras. Gambar 2.2 menunjukkan arsitektur *load balancing*.



Gambar 2.2 : Arsitektur *Load Balancing*

## 2.4 CPU Load average

*CPU Load average* adalah jumlah proses yang memanfaatkan sumber daya komputasi (CPU). Ini dihitung berdasarkan rata-rata eksponensial dan diperbarui setiap 1 menit, 5 menit dan 15 menit interval (Walker 2006). Ada berbagai alat analisis kinerja CPU di LINUX OS. Beberapa alat seperti itu yang menyediakan statistik CPU adalah *top* dan *uptime* yang mengambil nilai dari */proc/loadavg*. *Load average* menunjukkan permintaan untuk CPU dan dihitung dengan menjumlahkan jumlah rangkaian yang berjalan dan jumlah yang menunggu untuk dijalankan. Data *load average* didapat menggunakan aplikasi monitoring yaitu cacti (SNMP) berupa format CSV.

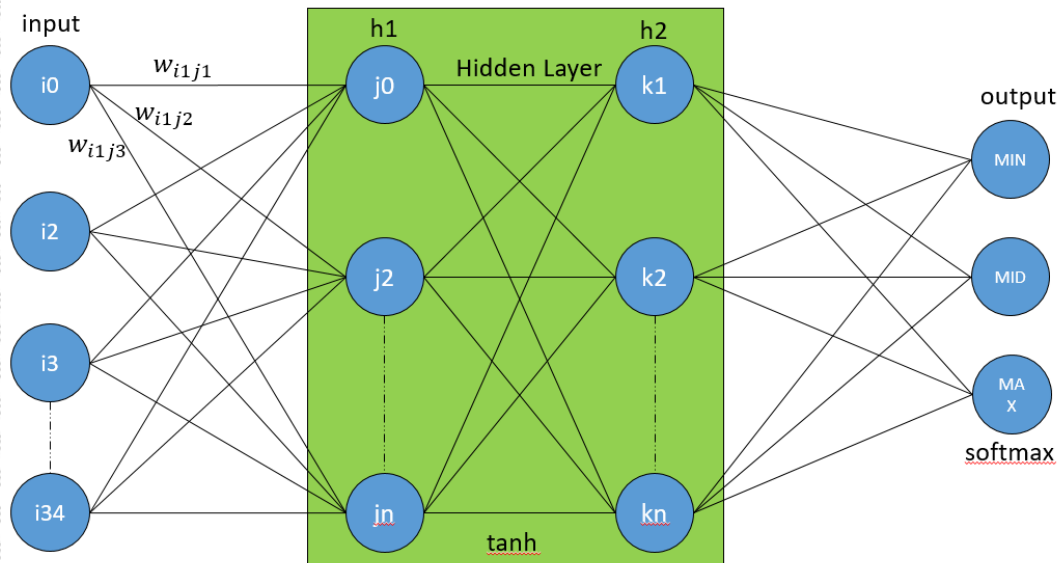
Ketika ada proses yang diantrikan untuk menghitung sumber daya, CPU dikatakan dalam saturasi. *Load average* harus antara jumlah inti CPU dan jika rata-rata beban lebih tinggi dari jumlah maksimum inti CPU, CPU dalam keadaan saturate karena ada proses yang diantrikan. Jumlah CPU yang dialokasi untuk Setiap *server* SIAM adalah 8 inti. Dalam penelitian ini *CPU load average* digunakan sebagai data *output* untuk melakukan pembagian klasifikasi. Penelitian ini menggunakan pendekatan total *CPU Load average* sebagai parameter untuk klasifikasi. Klasifikasi dibagi menjadi 3 kelas, yaitu nilai kelas minimum 0-2 (min. saturate), nilai kelas medium 3-6, dan nilai kelas maksimum >7 (maks. saturate), diharapkan sistem yang dibuat dapat melakukan klasifikasi berdasarkan parameter-parameter masukkan.

## 2.5 Neural Network

*Neural Networks* (NN) adalah model yang distimulasi berdasarkan struktur saraf otak. Otak pada dasarnya belajar dari pengalaman. Bertentangan dengan model matematika tradisional, yang diprogramkan, NN mempelajari hubungan antara *input* dan *output* yang dipilih. Dengan NN, seseorang akan memiliki akses ke lingkungan pemodelan yang kuat yang memungkinkan satu tes dan mengeksplorasi model simulasi lebih cepat dan lebih mudah daripada sebelumnya. Pelatihan model didasarkan pada data digital. *Input* dan *output* data akan diperkenalkan ke jaringan dengan menggunakan program jaringan saraf. Ketika pelatihan sudah siap, model siap untuk memprediksi kinerja data. Dalam bab berikut akan dijelaskan secara singkat perbandingan antara otak dan NN untuk mendapatkan pemahaman tentang NN.

*Neural network* adalah elemen pemrosesan yang saling berhubungan yang berfungsi untuk memecahkan masalah tertentu. *Neural network* biasanya terdiri dari 3 lapisan yang berbeda. Lapisan masukan, *hidden layer* dan lapisan keluaran. Lapisan-

lapisan ini saling berhubungan dengan sejumlah simpul yang berisi fungsi aktivasi di *hidden layer*. Data masukan dalam bentuk pola matriks disajikan ke dalam *neural network* melalui lapisan *input* yang kemudian dihubungkan ke *hidden layer*, di mana data masukan tersebut diproses untuk menentukan bobot koneksi. Demikian pula *hidden layer* terhubung ke lapisan *output* dan lagi diproses menggunakan bobot koneksi yang sama, yang merupakan *output* dari *neural network*. Gambar 2.3 merupakan model *neural network* menggunakan tiga lapisan,



Gambar 2.3 : Model Neural Network

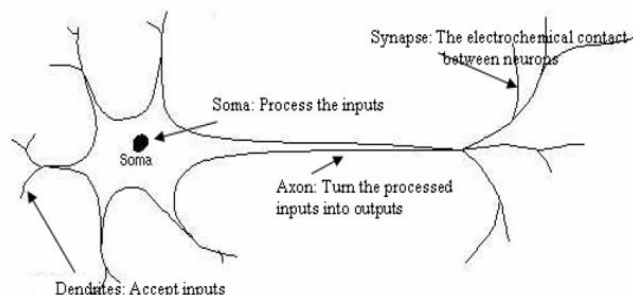
### 2.5.1 Analogi Otak Manusia

Komponen paling dasar dari ANN dimodelkan berdasarkan struktur otak. Beberapa bagian struktur ANN memang relatif tidak mirip dengan jaringan saraf otak dan beberapa bagian tidak mempunyai hubungan dengan jaringan saraf di otak secara biologis. Namun, ada bagian-bagian pada ANN memiliki kesamaan yang kuat dengan otak biologis dan karena itu banyak terminologi dipinjam dari neuroscience.

Otak pada dasarnya terdiri dari sejumlah besar (sekitar 10 miliar) *neuron*, yang saling terhubung secara besar-besaran. *Neuron* adalah unsur paling dasar dari otak manusia. *Neuron* adalah sel-sel tertentu, yang mengontrol kemampuan seperti identifikasi, berpikir, dan menerapkan pengalaman sebelumnya untuk setiap tindakan. *Neuron-neuron* ini saling berhubungan dengan hingga 200000 *neuron* lain. Kemampuan otak ini berasal dari jumlah komponen-komponen ini dan banyak koneksi di antara mereka.

Semua *neuron* alami memiliki empat elemen dasar yaitu. dendrit, soma, akson, dan sinapsis. Dendrit menerima masukan dari sumber lain, yang dikombinasikan dalam

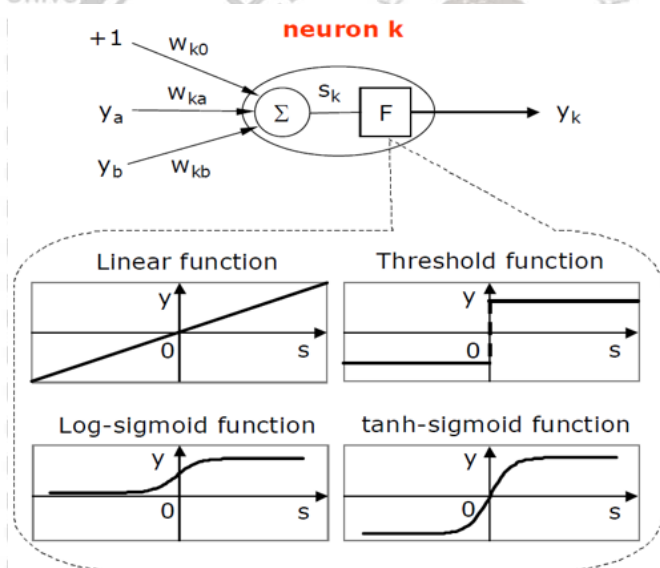
beberapa cara di soma, melakukan operasi nonlinear pada hasil akson dan kemudian *output* akan dihasilkan pada tahap akhir yaitu sinapsis. Gambar 2.4 menunjukkan *neuron* biologis yang disederhanakan dan hubungan antara keempat komponen ini.



Gambar 2.4 : *Neuron* secara biologis  
Sumber: (Kaiadi 1990)

### 2.5.2 Artificial Neuron

Unit dasar dari *neuron* buatan disimulasikan dari model biologis tetapi jauh lebih sederhana. Gambar 2.5 menunjukkan dasar-dasar dari *neuron* buatan.



Gambar 2.5 : *Neuron* ANN dan fungsi transfer  
Sumber: (Kaiadi 1990)

*Input* diwakili oleh simbol matematika ( $y_a$ ), ( $y_b$ ). Setiap *input* adalah dikalikan dengan bobot sambungan yang ditentukan, yang diwakili oleh simbol matematika ( $w_{kj}$ ).

Selain itu ada *input* tambahan yang sama dengan +1 untuk setiap *neuron* dan bobotnya yang sesuai dengan simbol matematika ( $w_{k0}$ ) disebut bias yang memperkenalkan off-set ke fungsi transfer (lihat bagian 2), yang memungkinkan *neuron* untuk memiliki *output*, bahkan jika *output* sama dengan nol. Pada langkah berikutnya beberapa operasi matematika (penjumlahan) dilakukan dan produk operasi ini diberi makan melalui fungsi

transfer nonlinier untuk menghasilkan *output*. Konstruksi semua jaringan ANN adalah sama tetapi beberapa dasar mungkin berbeda dalam yang berbeda.

### 2.5.3 Fungsi Aktivasi

Fungsi aktivasi yang paling banyak digunakan adalah fungsi linear, fungsi *threshold*, fungsi *Log-sigmoid* dan fungsi *tanh-sigmoid*. Penggunaan Fungsi transfer dapat saling berbeda tergantung pada aplikasi; dalam penelitian ini fungsi transfer yang digunakan adalah *tanh*. Pada gambar 2.5, sumbu  $x(s)$  dari grafik menunjukkan hasil penjumlahan bobot yang dihasilkan dan sumbu  $y$  mewakili *output*.

Fungsi transfer linear (Persamaan 2.1): ( $c$  konstan)

$$y = c \cdot s \quad (2.1)$$

Fungsi *threshold* (Persamaan 2.2): Ini hanya menghitung 1 dan 0. Jenis fungsi aktivasi ini disebut juga model *McCulloch-Pitts*. Model *McCulloch-Pitts* adalah *neuron* buatan yang sangat sederhana. *Input* bisa berupa nol atau satu. Dan *outputnya* nol atau satu.

$$y = \begin{cases} +1, & \text{jika } s \geq 0 \\ -1, & \text{jika } s \leq 0 \end{cases} \quad (2.2)$$

Fungsi aktivasi *sigmoid* logistik (Persamaan 2.3): Ketika menggunakan fungsi aktivasi *sigmoid*, *neuron* buatan akan terlihat lebih seperti alami. Hal yang baik tentang fungsi ini adalah ia membuat transisi yang halus (lihat gambar 2). Nilai *output* yang diberikan dalam rentang  $[0,1]$ :

$$y = \frac{1}{1 + e^{-s}} \quad (2.3)$$

Fungsi aktivasi *tanh* (Persamaan 2.4): Fungsi aktivasi *tangents hyperbolic* hampir sama seperti *Sigmoid*, namun fungsi ini dapat memberikan *output* bernilai negatif. Nilai *output* yang diberikan pada pada rentang  $[-1, 1]$ :

$$y = \frac{e^s - e^{-s}}{e^s + e^{-s}} \quad (2.4)$$

Fungsi aktivasi *softmax* (Persamaan 2.5) biasanya digunakan pada lapisan jaringan terakhir, mengubah nilai *arbitrary real* menjadi *posterior probability* kelas  $c_k$  dalam rentang  $(0; 1)$ :

$$p(c_k|x) = \frac{e^{a_k}}{\sum_{i=1}^m e^{a_i}} \quad (2.5)$$

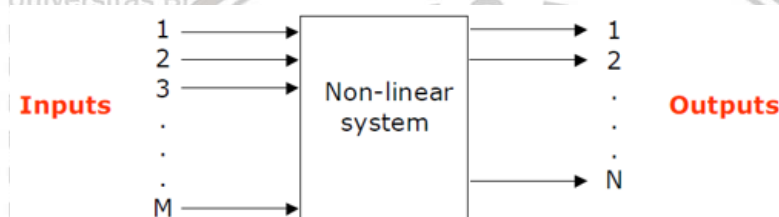
di mana  $m$  sesuai dengan jumlah node *output* (kelas) dan  $a_k$  adalah nilai aktivasi dari node  $k$ :

$$a_i = \sum_{j=0}^a w_{ij} h_j(x) \quad (2.6)$$

diberikan pada node  $i$  bobot  $w_{ij}$  dan *output* dari layer sebelumnya  $h_j(x)$ .

## 2.6 Artificial Neural Network (ANN)

ANN merupakan interkoneksi *neuron* buatan. ANN mempelajari hubungan antara *input* dan *output* yang dipilih dari pengalaman sebelumnya. ANN juga melakukan tugasnya secara bersamaan (yaitu pemrosesan paralel), yang membuat ANN menjadi sangat cepat. Sebuah ANN yang khas dapat mengidentifikasi dan mempelajari hubungan antara *input* dan *output* dari sistem multi-dimensi non-linear (Gambar 2.6).



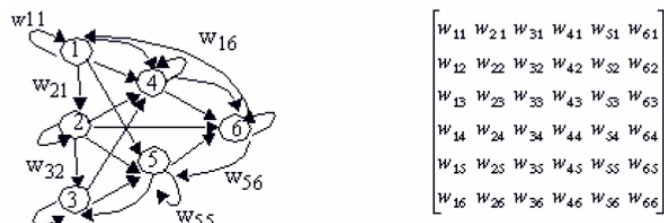
Gambar 2.6 : Sistem *non-linear* multi-dimensi

### 2.6.1 Arsitektur ANN

Pada umumnya arsitektur ANN diklasifikasikan menjadi dua kelompok berdasarkan pada struktur koneksi, arsitektur network umpan maju (hanya diizinkan untuk melewati satu arah, yaitu dari *input* ke *output*) dan jaringan berulang (umpan balik).

#### 2.6.1.1 Jaringan Umpan Balik

Ketika semua unit di semua lapisan saling terhubung ke semua unit di semua lapisan disebut *fully-connected* yang mana dalam banyak kasus yang paling umum di gunakan pada *recurrent networks*. Gambar 2.7 menyajikan ANN *fully-connected* dengan bobot matriks.



Gambar 2.7 : ANN *fully-connected* dan bobot matrix

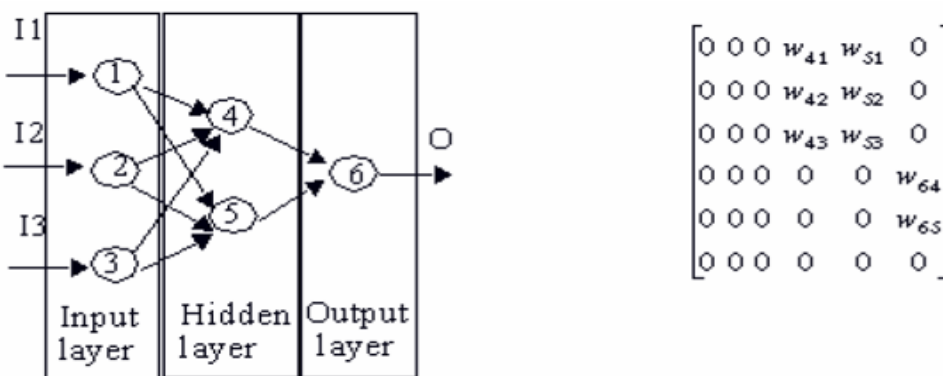
Sumber: (Kaiadi 1990)

Dalam *recurrent network* beberapa koneksi mungkin tidak ada, tetapi ada koneksi yang melakukan umpan balik. “Masukan pada *recurrent network* pada waktu  $t$ , akan mempengaruhi nilai *output* jaringan untuk tahap proses di waktu berikutnya yang nilainya lebih besar daripada  $t$ ”. Oleh karena itu, *recurrent network* perlu dioperasikan dari waktu ke waktu (pengulangan) sampai didapat hasil *output* yang optimal.

### 2.6.1.2 Jaringan Umpan Maju

ANN didefinisikan sebagai umpan maju, jika matriks interkoneksi dibatasi hanya pada satu arah (tidak ada umpan balik atau koneksi sendiri). Jaringan umpan maju berlapis, yang disebut *Multilayer perceptron* (MLP). *Perceptron* dalam hal ini terdiri dari satu lapisan *neuron* buatan nonlinier tanpa koneksi umpan balik.

Arsitektur ANN, pada dasarnya terdiri dari satu set unit yang merupakan *input layer*, satu (atau lebih) *hidden layer* (s) dan *output layer*. *Hidden layer* tidak memiliki kontak langsung dengan lingkungan, oleh karena itu disebut *hidden layer*. Tidak ada proses komputasi yang dilakukan pada *input layer* dan karena itu komponennya disebut *node input*. Gambar 2.8 menampilkan network umpan-maju dan matriks bobotnya.



Gambar 2.8 : *Multilayer perceptron* dan bobot matrix  
Sumber: (Kaiadi 1990)

### 2.6.1.3 Pelatihan Artificial Neural Network

Proses pembelajaran disebut pelatihan dan dilakukan sesuai dengan aturan pembelajaran, yang diklasifikasikan ke dalam dua jenis berikut utama:

- *Supervised learning*: di mana kedua *input* dan *output* diketahui, ini berarti jaringan dapat menentukan prediksi dari *input* yang diberikan.
- *Unsupervised learning*: di mana *output* tidak diketahui dan *neuron* harus menemukan cara untuk mengelolanya, metode ini tidak dipertimbangkan dalam penelitian ini.

#### 2.6.1.4 Learning Rate

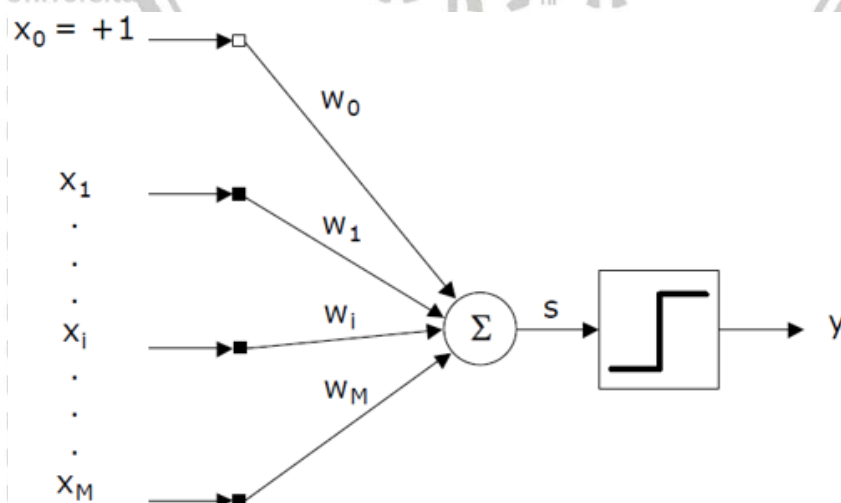
Sebagian besar fungsi pembelajaran memiliki beberapa istilah untuk *learning rate*, atau belajar konstan. Biasanya istilah ini positif dan antara nol dan satu. Dengan memilih tingkat yang lebih rendah, lebih banyak waktu dihabiskan untuk melatih ANN tetapi hasilnya akan lebih stabil. Dengan *learning rate* yang lebih cepat, pelatihan membutuhkan waktu lebih sedikit tetapi keakuratan hasilnya akan lebih buruk. Beberapa faktor lain dapat berperan dalam menentukan berapa lama waktu yang dibutuhkan untuk melatih jaringan seperti: Kompleksitas jaringan (jumlah *neuron* dan *hidden layer*), ukuran data, arsitektur dan jenis aturan pembelajaran.

#### 2.6.1.5 Aturan Pembelajaran

Sebagian besar aturan pembelajaran adalah semacam variasi dari aturan pembelajaran paling terkenal dan tertua, aturan Hebb. Hebb menerbitkan bukunya *The Organization of Behaviour* pada tahun 1949, di mana ia mengusulkan bahwa pembelajaran fisiologis dilakukan melalui modifikasi sinaptik di otak, dan dengan demikian, ia memperkenalkan aturan pertama yang diketahui untuk pembelajaran self-organizing. Beberapa hukum pembelajaran utama disajikan sebagai contoh. Aturan *perceptron* dan delta atau LMS (*Least Mean Square error*) aturan, baik menggunakan pelatihan yang diawasi, dan bekerja secara iteratif untuk memperbarui bobot mereka.

#### 2.6.1.6 Aturan Pembelajaran Perceptron

*Perceptron* terdiri dari *neuron* buatan nonlinear, yang memiliki fungsi aktivasi *threshold*. Gambar 2.9 menunjukkan *perceptron* dengan satu *neuron*.



Gambar 2.9 : *Perceptron*

Sumber: (Kaiadi 1990)



Persamaan (2.7) menyatakan proses matematis dalam *perceptron*:

$$\begin{cases} s = \sum_{i=0}^M w_i \cdot x_i \\ y = F(s) \end{cases} \quad (2.7)$$

*Error* didefinisikan sebagai perbedaan antara *output* yang diinginkan ( $d$ ) dan *output* aktual;

persamaan (2.8) menyatakan fungsi *error*:

$$e(n) = d(n) - y(n) \quad (2.8)$$

Di mana  $n$  adalah nomor iterasi dan  $e$  adalah *error*.

Pada awal pelatihan, bobot diberi nilai acak, dan dengan memulai proses, *error* akan dihasilkan. *Perceptron* memiliki algoritme untuk menghasilkan kumpulan bobot baru dengan bantuan *error* yang dihitung. Persamaan (2.9) menyatakan fungsi bobot:

$$w(n+1) = w(n) + \eta \cdot e(n) \cdot x(n) \quad (2.9)$$

#### 2.6.1.6.1 Aturan Delta

Aturan delta adalah salah satu yang paling umum digunakan. Aturan ini mengubah bobot dalam cara yang meminimalkan *Least Mean Square* jaringan. Aturan ini juga disebut

Aturan Pembelajaran *Least Mean Square* (LMS). Persamaan (2.10) menyatakan fungsi *error*:

$$E(w) = \frac{1}{2} e^2(n) \quad (2.10)$$

Fungsi *threshold* digantikan oleh fungsi transfer linear, dan oleh karena itu *error* dapat dinyatakan sebagai

$$e(n) = d(n) - x^T(n) \cdot w(n) \quad (2.11)$$

Dimana  $x$  dan  $w$  adalah *input* dan vektor bobot.

“Asumsi utamanya adalah bahwa bobot optimal dapat ditemukan dalam arah gradien kemiringan fungsi *error* sehubungan dengan bobot (metode penurunan paling curam), yaitu [2]“

$$\Delta w(n) = -\eta \cdot \frac{\partial E(w)}{\partial w(n)} \quad (2.12)$$

Persamaan differensiasi (2.8)

$$\frac{\partial E(w)}{\partial w(n)} = e(n) \cdot \frac{\partial e(n)}{\partial w(n)} \quad (2.13)$$

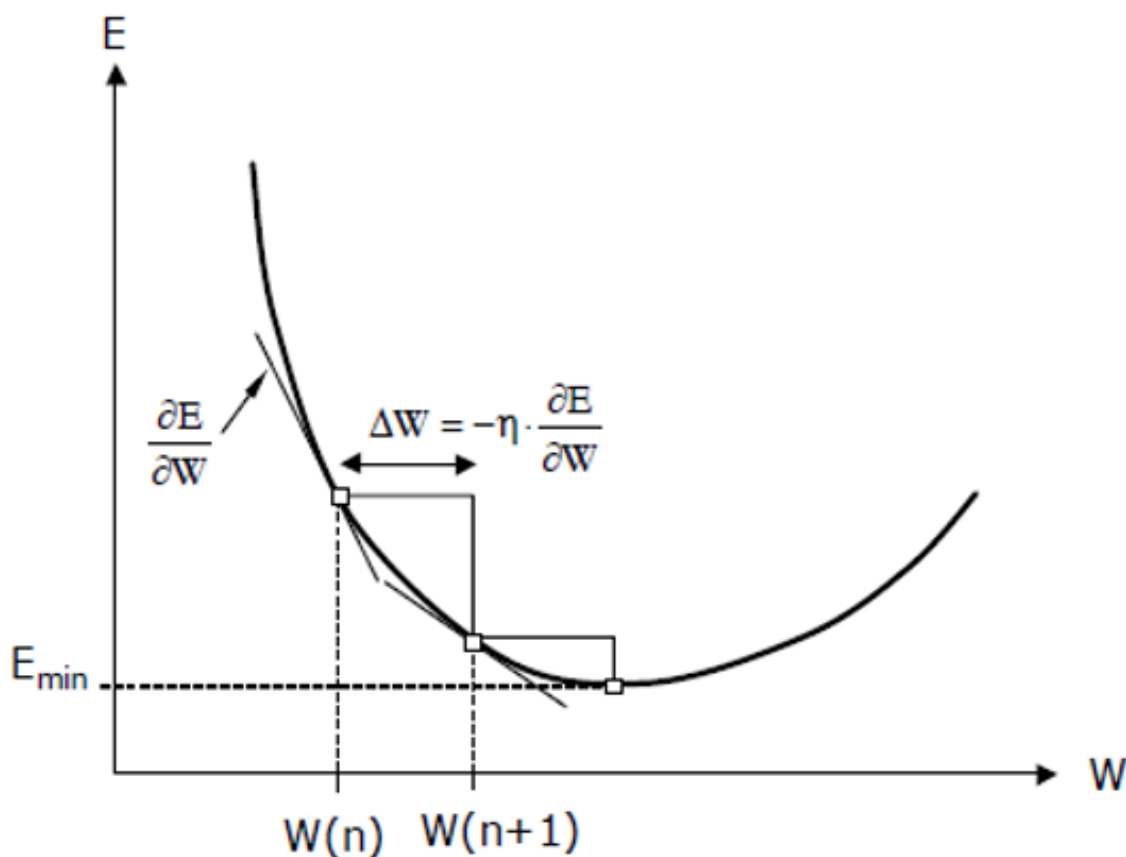
Memasukkan persamaan (2.11) ke dalam persamaan (2.13)

$$\frac{\partial e(n)}{\partial w(n)} = \frac{\partial (d - x^T \cdot w)}{\partial w(n)} = -x(n) \quad (2.14)$$

Dan akhirnya, vektor bobot baru sama dengan:

$$w(n+1) = w(n) + \Delta w(n) = w(n) + \eta \cdot e(n) \cdot x(n) \quad (2.15)$$

Ketika, di antara dua iterasi berulang, *error* tidak terjadi dari jumlah yang diberikan, pelatihan harus diakhiri dan bobot akan ditetapkan. Gambar 2.10 menunjukkan aturan pembelajaran *gradient descent learning*:



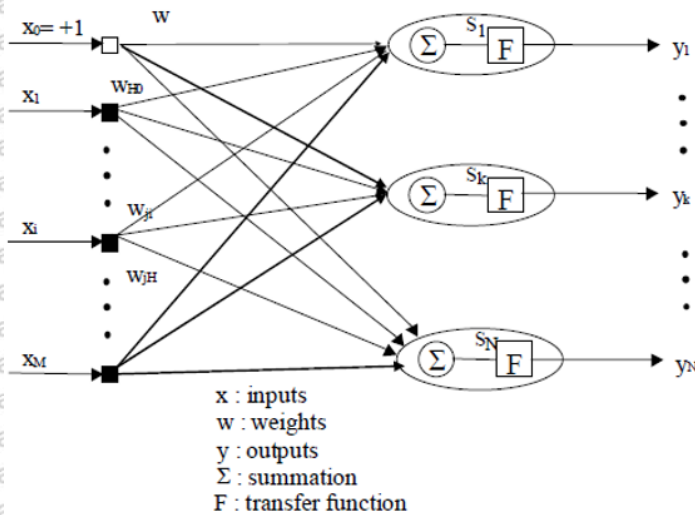
Gambar 2.10 : Pembaruan bobot berdasarkan *gradient descent*  
Sumber: (Kaiadi 1990)

## 2.6.2 Network Umpan-maju *Backpropagation*

ANN mempunyai sangat banyak tipe network, tetapi yang difokuskan dalam penelitian ini, adalah *Network* umpan-maju *Multilayer perceptron* (MLP). Untuk lebih memahami dan mengetahui keunggulan jaringan umpan-maju *Multilayer perceptron*, jaringan umpan-maju lapisan-tunggal telah didiskusikan terlebih dahulu.

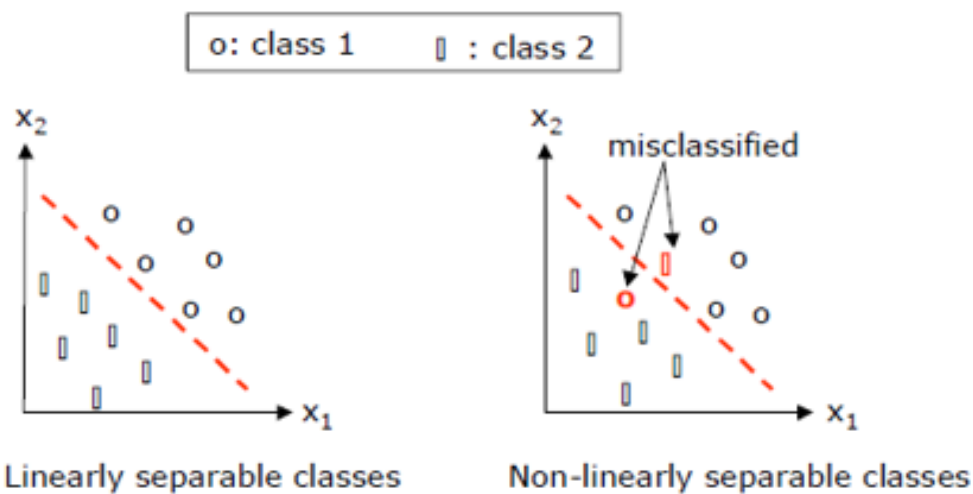
### 2.6.2.1 Jaringan *Single-layer* Umpan-maju

Jaringan *single-layer* hanya terdiri dari satu lapisan *neuron*, sebenarnya ada dua, tetapi karena hanya yang terakhir dari lapisan yang membuat perhitungan yang sebenarnya, itu disebut jaringan *single-layer*. Alasan mengapa ini disebut sistem umpan-maju karena informasi hanya menyebar dari *input* ke *output* (lihat gambar 2.11).



Gambar 2.11: Struktur jaringan *single-layer* umpan-maju

Masalah utama dengan jaringan *single-layer* adalah ketidakmampuan untuk melakukan klasifikasi fungsi non-linear (lihat gambar 2.12).



Gambar 2.12: Jaringan *single-layer* tidak mampu melakukan klasifikasi fungsi non-linear

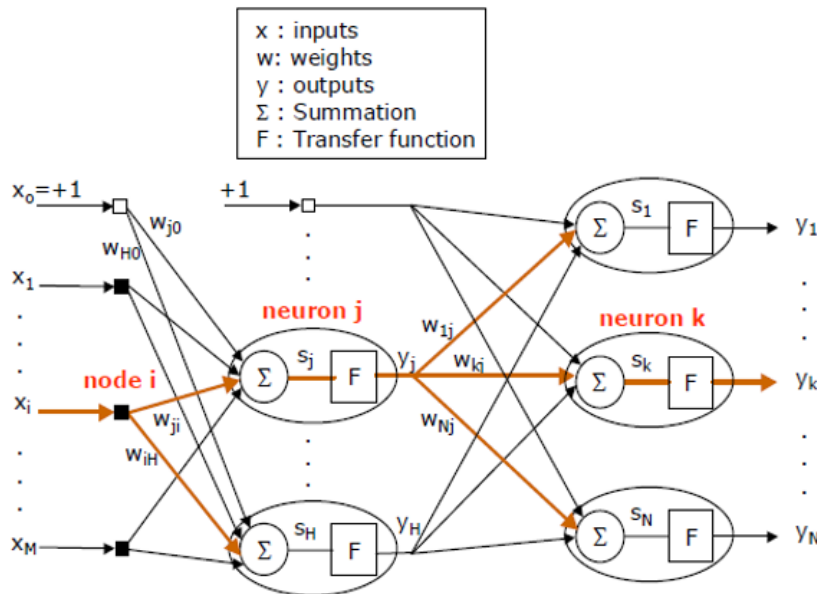
Sumber: (Kaiadi 1990)

Kerumitan *perceptrons single-layer* tidak cukup untuk menangani masalah kategorisasi yang lebih besar. Kedalaman logis dari masalah semacam ini terlalu besar untuk satu *perceptron* untuk ditangani. Untuk mengatasi masalah ini, dalam jaringan dibuat beberapa layer. Ini biasanya disebut sebagai MLP (*Multi Layer Perceptrons*).

### 2.6.2.2 Jaringan *Multi-layer* Umpan-maju

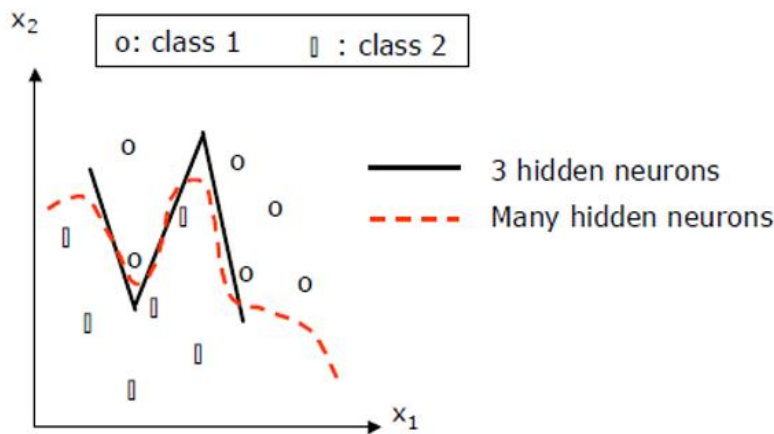
MLP tampak pada gambar 2.13. Pada jenis jaringan ini, satu atau lebih *hidden layer* berada di antara *input layer* dan *output layer* dan mereka akan membantu jaringan untuk memecahkan masalah yang lebih kompleks. Setelah jumlah *input* dan *output* telah diputuskan, jumlah *neuron* dalam *input* dan *output layer* ditetapkan. Jumlah optimal dari

neuron tersembunyi di *hidden layer*, bagaimanapun, harus dipilih. Menemukan jumlah yang optimal dari *hidden layer* adalah dengan menggunakan *trial* dan *error*, tetapi di sini adalah aturan praktis untuk memulai dengan nilai awal yang baik dan itu dimulai dengan jumlah parameter *input* dan dengan demikian meningkatkan jumlah *hidden layer*.



Gambar 2.13 : Struktur jaringan MLP umpan-maju dengan satu *hidden layer*

Menggunakan jaringan umpan maju *multi-layer* akan memecahkan masalah mengklasifikasikan fungsi non-linear (Gambar 2.14).

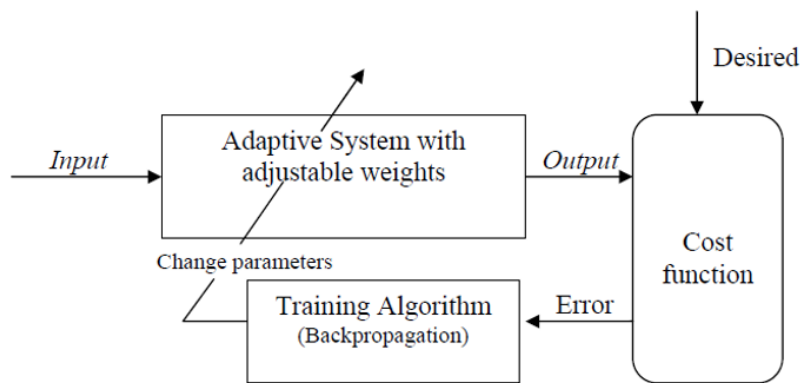


Gambar 2.14 : Kemampuan MLP melakukan klasifikasi fungsi non-linear

### 2.6.2.3 Algoritme Backpropagation

*Backpropagation* adalah dasar untuk melatih *supervised neural network*. Data yang digunakan sebagai *input* dikirimkan melalui jaringan, melewati layer demi layer, dan menghasilkan satu set *output* yang sudah ditentukan (*desired*). Selama data diteruskan melalui jaringan nilai bobot jaringan ditetapkan. Nilai *output* yang diperoleh dibandingkan

dengan nilai *output* yang diinginkan (*desired*), disebut sebagai *backward pass*; perbedaan antara *output* yang (*desired*) dan *output* yang dihitung (*error*) digunakan untuk menyesuaikan kembali bobot pada jaringan untuk mengurangi tingkat *error* (lihat Gambar 2.15). nilai *desired* adalah Proses ini dilakukan berulang, yang berlanjut sampai tingkat *error* yang dapat diterima akan diperoleh. Setiap kali jaringan memproses seluruh rangkaian data (baik yang maju maupun mundur), disebut *epoch*. Jaringan dengan cara ini dilatih dan *error* berkurang disetiap *epoch* sampai didapat tingkat *error* yang dapat diterima. Metode ini disebut pelatihan *error back-propagation*.



Gambar 2.15 : Pelatihan jaringan dengan algoritme *backpropagation*  
Sumber: (Kaiadi 1990)

#### 2.6.2.4 Fungsi loss

Fungsi *loss* digunakan untuk mengukur ketepatan hasil *output* yang diharapkan (prediksi). *Output* dari fungsi *loss* adalah nilai *real* yang disebut juga sebagai *cost*. Fungsi *loss* menghasilkan probabilitas, penelitian ini menghitung *error* hasil klasifikasi menggunakan fungsi *loss cross-entropy* (Persamaan 2.16):

$$L = - \sum_{i=1}^m y_i \log p_i \quad (2.16)$$

di mana  $m$  adalah jumlah kelas yang mungkin (*node*) di lapisan *output*,  $y$  vektor target dan  $p$  probabilitas *aposterior* untuk setiap kelas yang diprediksi oleh jaringan. Hasil dari perhitungan turunan fungsi *loss* akan digunakan dalam tahap pelatihan *backward*.

#### 2.7 Confusion matrix

*Confusion matrix* adalah sebuah tabel yang menyatakan jumlah data uji yang benar diklasifikasikan dan jumlah data uji yang salah diklasifikasikan. *Confusion matrix* akan digunakan sebagai alat untuk evaluasi atau mengukur kinerja dari sistem atau metode yang digunakan dalam penelitian ini adalah ANN *Backpropagation*. Contoh: *confusion matrix* untuk kelas biner (minimum *load avg.*) ditunjukkan pada tabel 2.1:

Tabel 1.2 : *Confusion matrix* kelas biner untuk klasifikasi *Min. Load Avg CPU*.

n=180		Kelas Prediksi ( <i>Predicted</i> )	
		Min. <i>Load Avg.</i> (P)	non-Min. <i>Load Avg.</i> (N)
Kelas Sebenarnya ( <i>Real</i> )	Min. <i>Load Avg.</i> (P)	TP	FN
	non-Min <i>Load Avg.</i> (N)	FP	TN

Keterangan untuk tabel 2.2 dinyatakan sebagai berikut:

- Kondisi *Positive* (P), yaitu jumlah kondisi *positive* dalam data.
- Kondisi *Negative* (N), yaitu jumlah kondisi *negative* dalam data.
- *True Positive* (TP), yaitu jumlah *load average CPU* dari kelas *Min. Load Avg* yang benar dan diklasifikasikan sebagai kelas *Min. Load Avg*.
- *True Negative* (TN), yaitu jumlah *load average CPU* dari kelas *Non Min. Load Avg* yang benar diklasifikasikan sebagai kelas *Non Min. Load Avg*.
- *False Positive* (FP), yaitu jumlah *load average CPU* dari kelas *Non Min. Load Avg* yang salah diklasifikasikan sebagai kelas *Min. Load Avg*.
- *False Negative* (FN), yaitu jumlah *load average CPU server* dari kelas *Min. Load Avg* yang salah diklasifikasikan sebagai kelas *Non. Min. Load Avg*.

Parameter-parameter yang akan dievaluasi atau diukur adalah sebagai berikut:

*Positive Predictive Value* (PPV) atau *precision* adalah tingkat ketepatan sistem atau metode melakukan klasifikasi sesuai dengan nilai klasifikasi sebenarnya. Untuk menghitung *precision* menggunakan persamaan 2.17.

$$PPV = \frac{\sum TP}{\sum \text{Predicted Kondisi Positive}} \times 100\% \quad (2.17)$$

*True Positive Rate* (TPR) atau *recall* atau *sensitivity* adalah tingkat keberhasilan sistem untuk selalu melakukan klasifikasi dengan benar. Untuk menghitung *sensitivity* menggunakan Persamaan 2.18

$$TPR = \frac{\sum TP}{\sum \text{Real Kondisi Positive}} \quad (2.18)$$

*Accuracy* adalah tingkat kedekatan antara nilai prediksi dengan nilai sebenarnya. Untuk menghitung *Accuracy* (ACC) dengan menggunakan Persamaan 2.19.

$$ACC = \frac{\sum TP + \sum TN}{\sum \text{Total Data}} \times 100\% \quad (2.19)$$

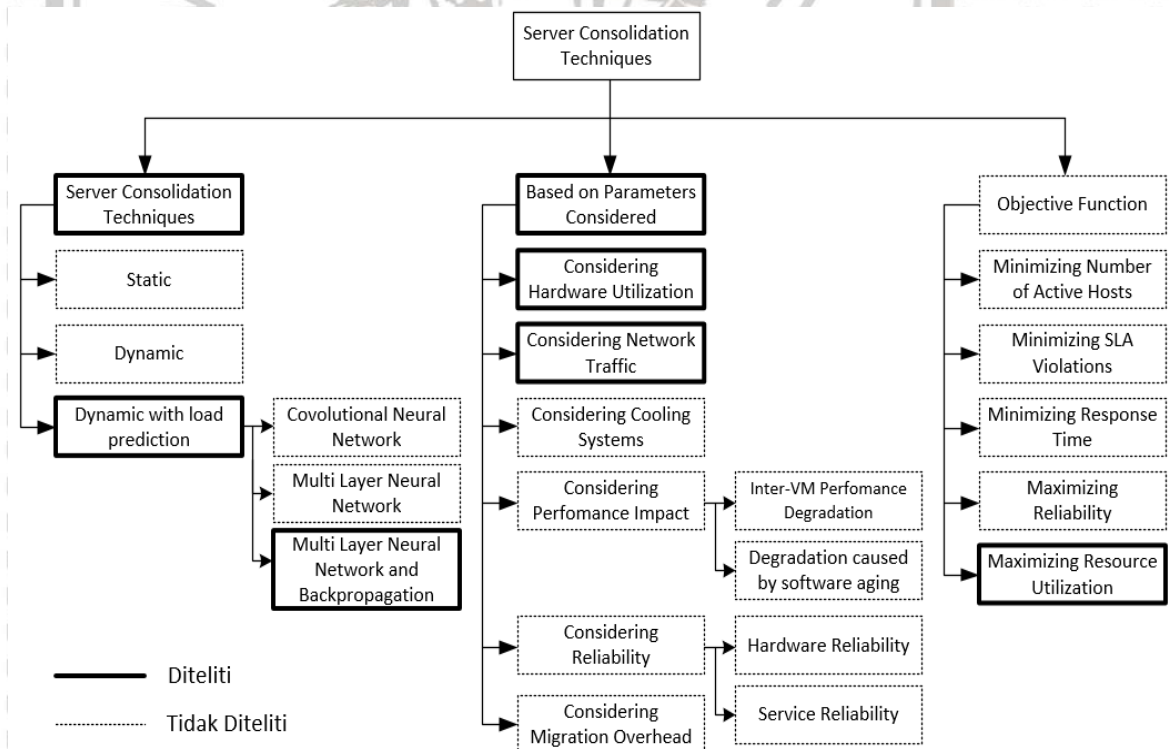


## BAB 3

### KERANGKA KONSEP PENELITIAN

#### 3.1 Kerangka Konsep Penelitian

Kajian yang dilakukan dalam penelitian ini adalah bidang ilmu teknik konsolidasi *server*. Penelitian ini khususnya mengkaji teknik konsolidasi *server* secara dinamis dengan melakukan prediksi beban kerja *server*. Untuk bagian prediksi beban kerja *server* dilakukan menggunakan pendekatan *supervised learning* ANN MLP (klasifikasi) menggunakan algoritme *backpropagation*. Parameter-parameter yang menjadi pertimbangan dalam kajian ini adalah utilisasi *hardware* (CPU, *Memory*, *Load average*), trafik jaringan dan jumlah *request* client yang terjadi pada sebuah kluster *web server*. Adapun kerangka penelitian yang digunakan dalam penelitian ini dapat dilihat pada diagram alir (Gambar 3.1):



Gambar 3.1: Diagram Alir Metodologi Penelitian

#### 3.2 Data Penelitian

Data penelitian yang digunakan adalah data penggunaan *hardware server* (CPU, *Memory*, *Load average*) dan jumlah *request* client sebuah kluster *server* selama bulan



agustus 2017 dan januari 2018. Ketersediaan akan sumber daya kluster *web server* untuk aplikasi SIAM cukup besar yaitu menggunakan 6 unit *server virtual machine* dengan spesifikasi masing-masing *server* yaitu CPU: 8 core, *Memory*: 16GB, *Hard Disk*: 50GB. Penggunaan sumber daya *server* secara maksimal (beban puncak) terjadi pada periode registrasi mahasiswa (KRS), lebih dari 65000 mahasiswa akan mengakses *server* SIAM secara bersamaan. Jumlah permintaan yang dilayani *server* dalam 1 hari dapat mencapai 1.7juta permintaan.

Peningkatan jumlah permintaan secara drastis menyebabkan *server* tidak responsif sehingga terjadi kelambatan akses, kegagalan koneksi dan lain sebagainya. Jumlah permintaan yang tinggi ke *web server* tidak terjadi secara kontinu dalam satuan waktu, ada saat dimana jumlah permintaan menurun. Dari data yang didapat dalam 1 hari puncak jumlah permintaan terjadi dengan rentang waktu jam 07.00-13.00 per hari. Sumber-sumber data yang digunakan dalam penelitian ini yaitu sebagai berikut:

1. Data utilisasi *server* diambil dari aplikasi monitoring (*web* aplikasi) yang beralamat <http://sysmon.ub.ac.id/cacti>
2. Data jumlah *request* client ke *server* menggunakan data *log access* masing-masing *server* dalam kluster.

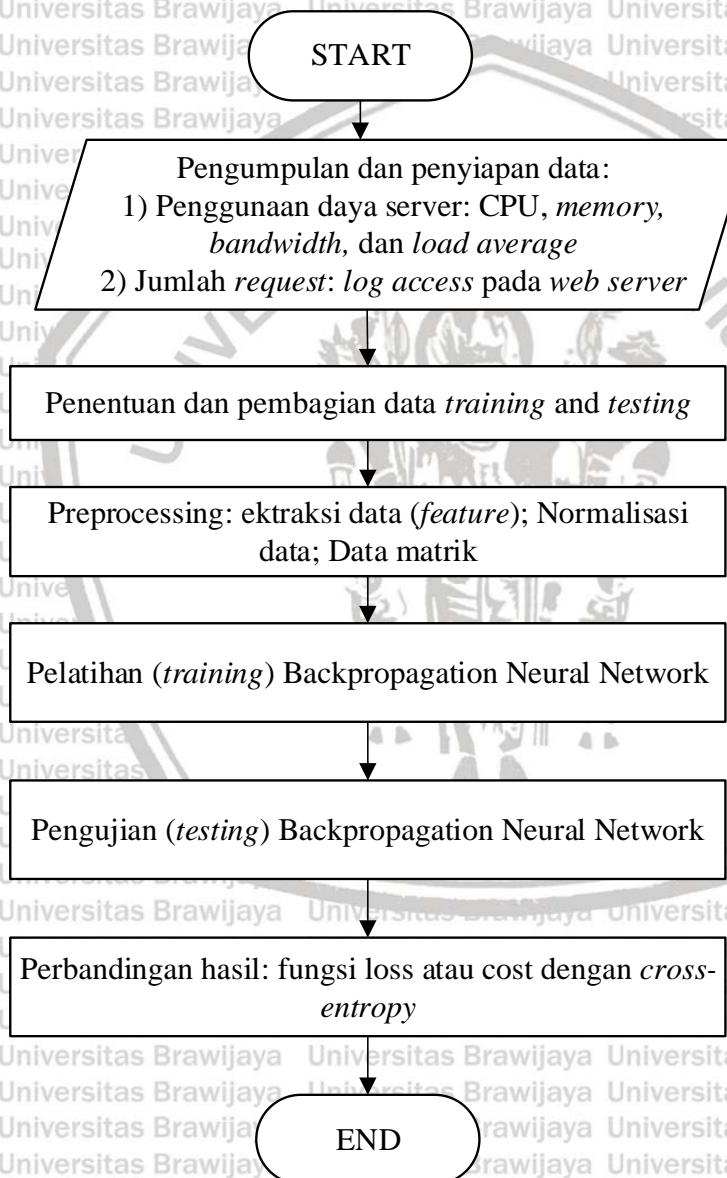
### 3.3 Hipotesis

Hipotesis yang digunakan sebagai dasar pada penelitian ini adalah teknik konsolidasi *server* secara dinamis memiliki tingkat penggunaan *hardware* yang lebih optimal jika dibandingkan dengan teknik konsolidasi *server* secara statis. Untuk menentukan konfigurasi *server* yang optimal, semua parameter yang dapat diukur dari kluster *web server* harus diperhatikan secara bersamaan. Untuk mempermudah penentuan konfigurasi kluster *web server*, semua parameter tersebut akan dikelompokkan menjadi 3 kelas/kelompok beban *server*. Pada penelitian ini akan digunakan *backpropagation neural network* sebagai *classifier* yang masukannya adalah semua parameter pada setiap kluster *web server* dan diharapkan dapat menghasilkan prediksi kelas beban yang sesuai dan memberikan rekomendasi konfigurasi *server* yang optimal.

## BAB 4 METODE PENELITIAN

### 4.1 Metode Penelitian

Metodologi penelitian yang digunakan pada penelitian ini secara rinci dapat dilihat dalam Gambar 4.1.



Gambar 4.1 : Metodologi Penelitian

Secara keseluruhan proses pengelompokan dilakukan dengan menggunakan bantuan bahasa pemrograman python. Versi python yang digunakan adalah python versi 3.

## 4.2 Pengambilan Data

Data-data yang digunakan dalam penyusunan penelitian ini adalah data primer dan data sekunder.

### a. Data Primer

Data primer adalah data yang didapatkan dari *web server* SIAM Universitas Brawijaya selama bulan agustus 2017 dan bulan januari 2018 dengan interval 2 jam. Kegunaan dari masing-masing data ini sbb:

1. Data *log access web server* digunakan untuk mengukur jumlah *request* yang dilayani oleh sistem kluster.
2. Data SNMP adalah hasil pengukuran penggunaan sumber daya CPU, *Memory* dan trafik jaringan (*throughput*).

### b. Data Sekunder

Data sekunder adalah data yang diperoleh dari hasil studi literatur (buku, jurnal-jurnal, dan internet). Data sekunder yang diperlukan dalam kajian ini adalah:

1. Fungsi aktivasi pada *hidden layer* adalah *tanh*.
2. Fungsi aktivasi pada *output layer* adalah *softmax*.
3. Fungsi *Loss* atau *Cost* menggunakan *Cross-entropy*.
4. Memperbarui parameter bobot dan bias pada saat proses *backpropagation* menggunakan *mini-batch gradient descent*.

## 4.3 Preprocessing

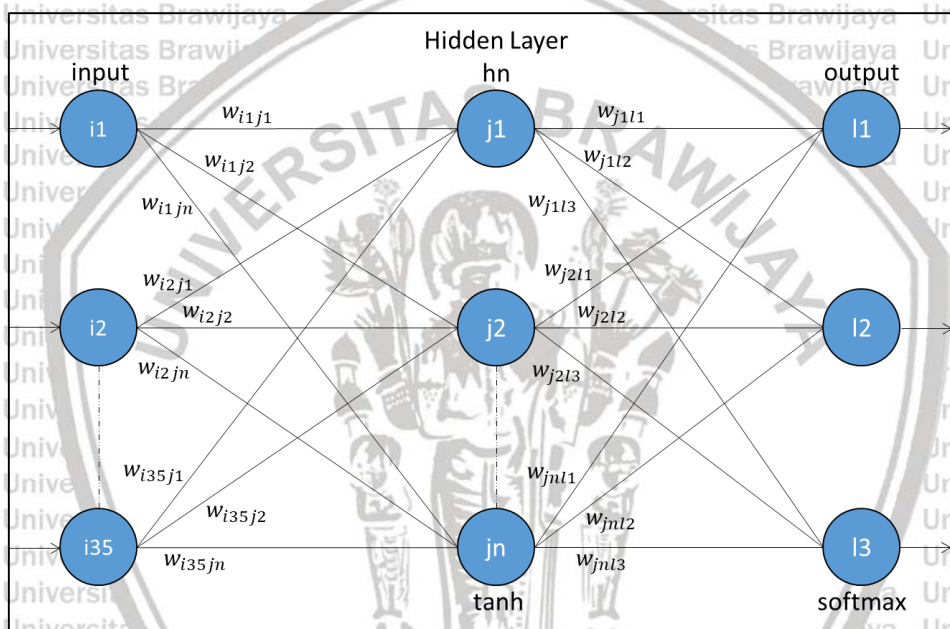
Terdapat beberapa tahapan pada preprocessing data, yang secara terurut adalah sebagai berikut:

1. Eksktraksi data dengan memecah seluruh bagian data mentah berdasarkan tanda baca (karakter selain alfabet) menjadi vektor string dan dikelompokkan berdasarkan ciri (*feature*).
2. Melakukan normalisasi data (*feature scalling*) untuk membuat data numeric pada dataset memiliki rentang nilai (*scale*) yang sama sehingga tidak ada satu variabel data yang mendominasi variabel data lainnya. Rentang nilai yang dihasilkan dari proses normalisasi ini adalah  $0 - 1$ , kecuali untuk nilai jumlah IP akses tetap menggunakan nilai awal.
3. Data yang dihasilkan dari normalisasi (*feature scalling*) dikonversi menjadi data matriks yang akan digunakan untuk *pelatihan backpropagation neural network*. Data yang

dihasilkan ini kemudian dibagi menjadi data latih dan data validasi, yaitu 70% data latih dan 30% data validasi.

#### 4.4 Pelatihan *Backpropagation Neural Network*

*Neural Network* (NN) terdiri dari elemen-elemen *neuron* yang beroperasi secara paralel yang saling terkoneksi dengan nilai bobot tertentu (Gambar 4.2). Nilai-nilai bobot koneksi antar *neuron-neuron* ini akan disesuaikan atau dilatih untuk melakukan fungsi tertentu dalam hal ini untuk melakukan klasifikasi beban kerja sebuah kluster *server*. Klasifikasi ini terdiri dari 3 kelas atau kelompok beban yang akan digunakan untuk menentukan jumlah *server* yang akan melayani *request* dari client. Pada tahap pelatihan dimulai dengan memasukkan data latih ke dalam jaringan.



Gambar 4.2 : Struktur *Neural Network*

Dengan menggunakan data latih ini jaringan akan mengubah-ubah bobot yang menjadi penghubung antar node. Pada setiap *epoch* dan iterasi dilakukan evaluasi terhadap *output* NN. Tahap ini berlangsung pada beberapa iterasi dan berhenti setelah jaringan menemukan bobot yang sesuai dan nilai *error* yang diinginkan telah tercapai atau jumlah iterasi telah mencapai nilai yang ditetapkan. Selanjutnya bobot ini menjadi dasar pengetahuan pada tahap pengujian.

Algoritma *backpropagation* yang dilakukan adalah sebagai berikut:

- Menentukan jumlah *neuron* dan *hidden layer*.
- Menentukan jumlah *epoch*.
- Menentukan *learning rate*.
- Untuk proses *training*, dilakukan langkah e + g.

e. Fase propagasi maju (*forward*):

i. Jumlahkan semua sinyal *input* ( $i_n$ ) dan bobot ( $w_{ij}$ ) yang masuk ke setiap *hidden unit* ( $J_n$ ) pada *hidden layer* ( $h_n$ ), menggunakan Persamaan 4.1

$$h_{in_n} = b_{j0} + \sum_{i=1}^n i_i w_{ij} \quad (4.1)$$

ii. Hitung keluaran setiap *hidden unit* ( $J_n$ ) pada *hidden layer* ( $h_n$ ) menggunakan fungsi aktivasi *tanh*, menggunakan Persamaan 4.2

$$h_{out_n} = f(h_{in_n}) = \frac{e^s - e^{-s}}{e^s + e^{-s}} \quad (4.2)$$

iii. Jumlahkan semua sinyal yang masuk ke *output layer* pada setiap unit ( $l_n$ ), dengan menggunakan Persamaan 4.3

$$o_{in_n} = b_{l0} + \sum_{i=1}^n h_{out_n} w_{jl} \quad (4.3)$$

iv. Hitung keluaran dari setiap *output unit* ( $l_n$ ) di *output layer*, menggunakan fungsi aktivasi *softmax*, menggunakan Persamaan 4.4

$$o_{out_n} = \frac{e^{o_{in_n}}}{\sum_{a=1}^n e^{o_{in_a}}} \quad (4.4)$$

v. Hitung *error* menggunakan fungsi *loss cross-entropy* di setiap *epoch*, menggunakan Persamaan 4.5.

$$error = -\left(\frac{1}{n}\right) \left( \sum_{i=1}^3 y_i \times \log(o_{out_i}) + ((1 - y_i)) \times \log((1 - o_{out_i})) \right) \quad (4.5)$$

f. Fase propagasi mundur (*backward*):

i. Hitung faktor *error* pada *output layer* dengan melakukan penurunan partial Persamaan 4.5 terhadap  $o_{out_n}$ , menggunakan Persamaan 4.6

$$\frac{\partial E_n}{\partial o_{out_n}} = \frac{\partial \left( -1 \times ((y_n \times \log(o_{out_n})) + (1 - y_n) \times \log(1 - o_{out_n})) \right)}{\partial o_{out_n}} \quad (4.6)$$

ii. Hitung perubahan bobotnya, menggunakan Persamaan 4.7

$$\Delta w_{jl} = \alpha \partial_k z_j \quad (4.7)$$

iii. Hitung penjumlahan *error*-nya, menggunakan Persamaan 4.8

$$\partial_{net_j} = \sum_{k=1}^m \partial_k w_{kj} \quad (4.8)$$

iv. Hitung faktor *error* pada *hidden layer*, menggunakan Persamaan 4.9

$$\partial_j = \partial_{net_j} z_j (1 - z_j) \quad (4.9)$$

v. Hitung perubahan bobot, menggunakan Persamaan 4.10

$$\Delta v_{ji} = \alpha \partial_j i_i \quad (4.10)$$

g. Perubahan bobot,

i. Ubah bobot yang menuju *output layer*, menggunakan Persamaan 4.11

$$w_{kj}(t+1) = w_{kj} + \Delta w_{kj} \quad (4.11)$$

ii. Ubah bobot yang menuju *hidden layer*, menggunakan Persamaan 4.12.

$$v_{ji}(t+1) = v_{ji}(t) + \Delta v_{ji} \quad (4.12)$$

#### 4.5 Pengujian

Pada tahap ini dilakukan uji coba untuk mendapatkan struktur NN yang optimal dengan mengukur nilai *error (loss)* terkecil, tingkat *accuracy (%)* tertinggi dan waktu *training (s)*. Tahapan dan parameter-parameter yang dilakukan pada proses pengujian ini adalah sebagai berikut:

- a. Menentukan jumlah *hidden layer*, ujicoba pertama menggunakan 1 layer dan ujicoba ke-dua menggunakan 2 layer. Parameter jumlah unit *neuron* yang diuji adalah 8, 16, 32, 64, 128, 256, dan 512. Hasil (jumlah *neuron*) dari ujicoba pertama akan digunakan sebagai referensi untuk ujicoba ke-dua.
- b. Menentukan nilai *learning rate*, sebagai berikut 0.00001, 0.0001, 0.001, dan 0.01.
- c. Melakukan ujicoba akurasi klasifikasi menggunakan data latih, data uji secara bersamaan dan parameter-parameter yang sudah ditentukan.
- d. Setiap proses latih (*epoch*) langsung dilakukan uji menggunakan data uji, proses ini tidak mempengaruhi nilai bobot.

#### 4.6 Evaluasi Hasil

Melakukan perbandingan dari hasil pengujian, untuk melakukan evaluasi tingkat kinerja sistem klasifikasi yang optimal dengan menggunakan *confusion matrix*. Parameter-parameter yang diukur adalah tingkat *Positive Predictive Value (PPV)* atau *precision*, *True Positive Rate (TPR)* atau *sensitivity* dan *Accuracy (ACC)*.

#### 4.7 Analisis

Pada tahap ini dilakukan analisis terhadap sistem yang telah dibuat dengan cara membandingkan hasil perancangan dan pembuatan dengan parameter-parameter hasil pengujian. Hasil analisis ini akan digunakan sebagai bahan dalam pengambilan kesimpulan dan saran.



## BAB 5

### HASIL DAN PEMBAHASAN

#### 5.1 Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah data penggunaan sumber daya *server* dan jumlah IP akses (*request*) pada kluster *server* SIAM pada bulan agustus 2017 dan januari 2018. Fokus penelitian ini adalah melakukan klasifikasi beban kerja *server-server* dalam kluster yang akan digunakan untuk melakukan konsolidasi beban kerja kluster *server* SIAM dalam penentuan jumlah *server* yang akan melayani. Parameter-parameter yang digunakan adalah parameter *input*: penggunaan CPU, penggunaan *Memory*, penggunaan Jaringan (*Troughput*), jumlah alamat IP akses (*request*), dan parameter *output*: *load average* CPU.

##### 5.1.1 Proses Pembelajaran

Proses pembelajaran dilakukan dengan menggunakan script *learn.py*, dibuat dengan menggunakan bahasa pemrograman python yang telah menyediakan fungsi-fungsi pembelajaran dan pengujian pada ANN dengan algoritma *backpropagation*. Proses pembelajaran dilakukan untuk mencari konfigurasi terbaik dengan cara sebagai berikut:

- a. Menentukan jumlah *hidden layer*, ujicoba pertama menggunakan 1 *layer* dan ujicoba ke-dua menggunakan 2 *layer*.
- b. Menentukan jumlah unit *neuron* pada *hidden layer*. Parameter jumlah unit yang diuji adalah 8, 16, 32, 64, 128, 256, 512.
- c. Menentukan nilai *learning rate*, parameter *learning rate* sebagai berikut 0.00001, 0.0001, 0.001, 0.01, 0.1, dan 1
- d. Melakukan ujicoba akurasi klasifikasi menggunakan data latih, data uji secara bersamaan dan parameter-parameter yang sudah ditentukan.

Data sumber daya *server server* dan jumlah IP akses (*request*) yang digunakan pada proses pembelajaran diambil pada bulan agustus 2017 dan Januari 2018.

#### 5.2 Pengujian

Pengujian dilakukan melalui dua tahap yaitu, pengujian terhadap data yang dilatih/*training* dan pengujian data baru yang belum pernah dilatih/*train*. Selama proses



pembelajaran berlangsung, *error* dari tiap pola dapat ditampilkan, *error* dari tiap pola yang diajarkan pada jaringan yang disimpan dari iterasi 1 sampai dengan 500. Penyebab lamanya proses pembelajaran pada struktur jaringan yang digunakan adalah pola yang diajarkan sangat banyak, sehingga pengulangan dari satu iterasi ke iterasi dipengaruhi oleh banyaknya pola dan struktur data pada pola tersebut.

Pada proses pembelajaran, jaringan melakukan proses *backward* atau arus balik dengan cara mengubah nilai bobot jaringan, sebelum target *error* terpenuhi, pembelajaran akan terus dilanjutkan sampai tercapai target *error*. Semakin kecil target *error* yang ditetapkan semakin akurat jaringan mengenali dan menghitung bentuk pola baru yang diujikan kepadanya.

### 5.2.1 Pelatihan dengan 1 *Hidden layer*

Hasil pelatihan data sumber daya *server* dan jumlah IP akses (*request*), dengan menggunakan struktur jaringan 3 *layer* terdiri dari 32 unit *input*, 1 *hidden layer* dan 3 unit *output* dengan 500 *epoch*, seperti tabel 5.1. Dengan menggunakan struktur jaringan 1 *hidden layer* didapat nilai *loss (error)*, yaitu: avg. loss=0.8564, min. loss=0.7485, max. loss=1.002. Struktur ANN yang dipilih untuk melanjutkan proses pelatihan dengan 2 *Hidden layer* adalah menggunakan jumlah neuron= 512 karena mempunyai nilai *loss (error)*= 0.7485 dan nilai *accuracy* 88.83% seperti yang ditunjukkan pada tabel 5.1 no. 7.

Tabel 5.1 : *Output ANN dengan 1 hidden layer*

No	Jumlah <i>neuron</i> H1	Jumlah Iterasi	Loss ( <i>error</i> )	<i>Accuracy</i> (%)	<i>Accuracy training</i> (%)	Waktu <i>Training</i> (s)
1	8	500	1.002	34.64	53.34	45
2	16	500	0.9968	33.52	43.83	38
3	32	500	0.8770	48.04	75.71	37
4	64	500	0.8210	76.54	91.04	36
5	128	500	0.7970	83.24	94.98	38
6	256	500	0.7528	83.80	93.62	42
7	512	500	0.7485	88.83	97.01	42

### 5.2.2 Pelatihan dengan 2 *Hidden layer*

Hasil pelatihan data sumber daya *server* dan jumlah IP akses (*request*), dengan menggunakan struktur jaringan 5 *layer* terdiri dari 32 unit *input*, 2 *hidden layer* dan 3 unit *output* dengan 500 iterasi (*epoch*), seperti tabel 5.2. Dengan menggunakan struktur jaringan 2 *hidden layer* didapat nilai *loss* semakin kecil, yaitu: avg. loss=0.6655, min. loss=0.6537, dan max. loss=0.6733. Struktur ANN yang dipilih untuk melanjutkan proses pelatihan dengan *learning rate* adalah menggunakan jumlah neuron H1= 512, H2= 32

karena memiliki nilai loss (error) paling rendah yaitu 0.6537 dan nilai *accuracy* 90.50% seperti yang ditunjukkan pada tabel 5.2 no. 2.

Tabel 5.2 : *Output ANN dengan 2 hidden layer*

No.	Jumlah <i>neuron</i> H1	Jumlah <i>neuron</i> H2	Jumlah Iterasi	Loss ( <i>error</i> )	<i>Accuracy</i> (%)	<i>Accuracy Training</i> (%)	Waktu <i>training</i> (s)
1	512	16	500	0.6724	90.50	97.15	62
2	512	32	500	0.6537	90.50	98.92	59
3	512	64	500	0.6733	90.50	97.56	49
4	512	128	500	0.6638	90.50	97.96	66
5	512	256	500	0.6667	90.50	97.96	71
6	512	512	500	0.6631	90.50	99.05	143

### 5.2.3 Pelatihan berdasarkan *learning rate*

Hasil pengujian menunjukkan bahwa nilai *learning rate* awal mencapai nilai loss paling optimal. Dalam penelitian ini semakin besar nilai *learning rate*, nilai loss yang dicapai semakin besar. Jaringan optimal saat nilai *learning rate* awal 0.00001 yaitu kemampuan jaringan dalam mengenali data baru adalah yang paling tinggi (90.50%) dengan nilai loss paling kecil (0.6537).

#### 5.2.3.1 *Learning rate 0.0001*

Hasil pelatihan data sumber daya *server* dan jumlah IP akses (*request*), dengan menggunakan struktur jaringan 5 *layer* terdiri dari 32 unit *input*, 2 *hidden layer* dan 3 unit *output* dengan 500 iterasi (*epoch*) dan *learning rate* 0.0001, seperti tabel 5.3.

Tabel 5.3 : *Output ANN dengan learning rate 0.0001*

No.	Jumlah <i>neuron</i> H1	Jumlah <i>neuron</i> H2	Jumlah Iterasi	Loss ( <i>error</i> )	<i>Accuracy</i> (%)	<i>Accuracy Training</i> (%)	Waktu <i>training</i> (s)
1	512	32	500	0.7509	90.50	98.37	42
2	512	64	500	0.7726	90.50	98.78	43
3	512	128	500	0.8040	90.50	98.64	52
4	512	256	500	0.8336	90.50	98.37	65
5	512	512	500	0.8518	90.50	98.24	123

#### 5.2.3.2 *Learning rate 0.000001*

Hasil pelatihan data sumber daya *server* dan jumlah IP akses (*request*), dengan menggunakan struktur jaringan 5 *layer* terdiri dari 32 unit *input*, 2 *hidden layer* dan 3 unit *output* dengan 500 iterasi (*epoch*) dan *learning rate* 0.000001, seperti pada tabel 5.4.

Tabel 5.4 : *Output ANN dengan learning rate 0.000001*

No.	Jumlah <i>neuron</i> H1	Jumlah <i>neuron</i> H2	Jumlah Iterasi	Loss ( <i>error</i> )	<i>Accuracy</i> (%)	<i>Accuracy Trainig</i> (%)	Waktu <i>training</i> (s)
1	512	32	500	0.9919	59.78	66.21	43
2	512	64	500	0.9393	61.45	66.49	47
3	512	128	500	0.9773	62.01	63.91	54
4	512	256	500	0.9731	60.34	63.64	72
5	512	512	500	0.9388	58.10	62.96	129

### 5.2.3.3 Learning rate 0.001

Hasil pelatihan data sumber daya *server* dan jumlah IP akses (*request*), dengan menggunakan struktur jaringan 5 *layer* terdiri dari 32 unit *input*, 2 *hidden layer* dan 3 unit *output* dengan 500 iterasi (*epoch*) dan *learning rate* 0.001, seperti pada tabel 5.5.

Tabel 5.5 : *Output ANN dengan learning rate 0.001*

No.	Jumlah neuron H1	Jumlah neuron H2	Jumlah Iterasi	Loss ( <i>error</i> )	Accuracy (%)	Accuracy Training (%)	Waktu training (s)
1	512	32	500	1.137	90.50	99.46	43
2	512	64	500	1.173	90.50	99.32	47
3	512	128	500	1.245	90.50	99.19	54
4	512	256	500	1.276	90.50	99.32	72
5	512	512	500	1.310	90.50	99.46	129

### 5.2.4 Pelatihan Berdasarkan Pembagian Data Training dan Data Testing

Hasil pengujian (tabel 5.6) menunjukkan bahwa menggunakan pembagian data train 90% dan data test 10% mendapatkan nilai loss paling optimal. Dalam penelitian ini semakin besar pembagian data train dan data test, nilai loss yang dicapai semakin besar tetapi masih dapat di toleransi. Jaringan optimal saat menggunakan data train 90% dan data test 10% yaitu kemampuan jaringan dalam mengenali data baru adalah yang paling tinggi (90.50%) dengan nilai loss paling kecil (0.63).

Tabel 5.6 : *Output ANN dengan pembagian data train dan data test*

No.	Data Train (%)	Data Test (%)	Jumlah Iterasi	Loss ( <i>error</i> )	Accuracy (%)	Accuracy Training (%)	Waktu training (s)
1	90	10	500	0.63	90.50	96.72	43
2	80	20	500	0.66	90.50	96.59	47
3	70	30	500	0.75	90.50	95.03	54
4	60	40	500	0.75	90.50	95.31	72

### 5.3 Evaluasi Hasil Pengelompokkan

*Confusion matrix* digunakan sebagai alat untuk melakukan evaluasi atau mengukur kinerja dari sistem atau metode klasifikasi yang digunakan dalam penelitian ini (*ANN Backpropagation*). Parameter-parameter yang digunakan untuk mengukur kinerja dari sistem adalah *PPV* atau *precision*, *TPR* atau *sensitivity*, *Accuracy* (*ACC*). Bobot yang dihasilkan dari arsitektur *ANN* yang optimal akan digunakan untuk membentuk tabel *confusion matrix* 3-kelas (tabel 5.6), dari tabel tersebut akan dibentuk menjadi tabel *confusion matrix* biner (tabel 5.7). Kemudian dari tabel *confusion biner* dengan menggunakan persamaan (2.17), (2.18) dan (2.19), dilakukan proses perhitungan *PPV* atau *precision*, *TPR* atau *sensitivity* dan *Accuracy* (*ACC*) untuk 180 data uji.

Tabel 5.7 : *Confusion matrix* 3 kelas *Load Avg CPU*

n=180		Predicted Class		
		Min.	Med.	Max
Real Class	Min.	54	3	3
	Med.	3	54	3
	Max.	3	3	54

Tabel 5.8 : *Confusion matrix* biner *Load Avg CPU* kelas Min.

n=180		Predicted Class		
		Min. (P)	non-Min. (N)	
Real Class	Min. (P)	54 (TP)	6 (FN)	60
	non-Min. (N)	6 (FP)	114 (TN)	120
		60	120	

Nilai *PPV* atau *precision* adalah,

$$PPV = \frac{\sum TP}{\sum \text{Predicted Kondisi Possitive}} \times 100\%$$

$$PPV = \frac{54}{60} \times 100\%$$

$$PPV = 90\%$$

Nilai *TPR* atau *sensitivity* adalah,

$$TPR = \frac{\sum TP}{\sum \text{Real Kondisi Possitive}}$$

$$TPR = \frac{54}{60}$$

$$TPR = 0.9$$

Nilai *Accuracy* (*ACC*) adalah,

$$ACC = \frac{\sum TP + \sum TN}{\sum \text{Total Data}} \times 100\%$$

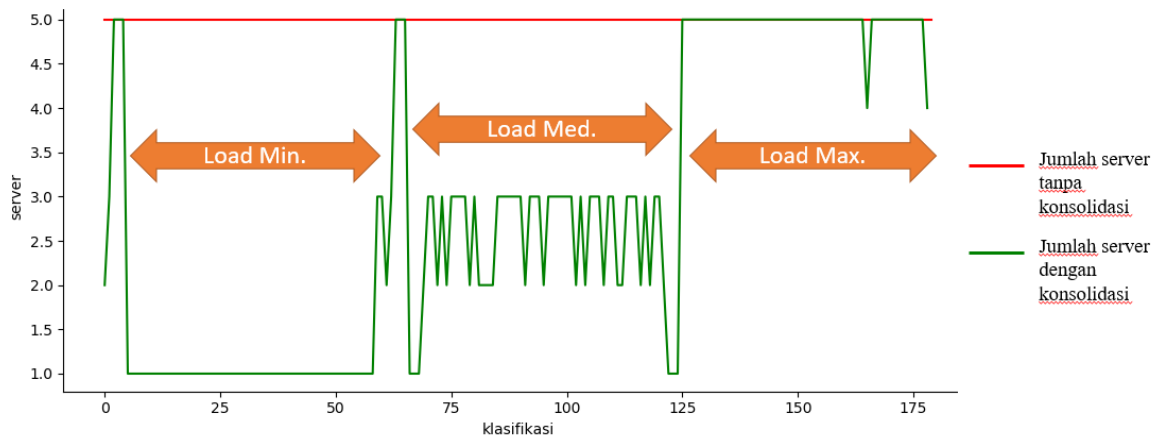
$$ACC = \frac{54 + 114}{180} \times 100\%$$

$$ACC = 93\%$$

#### 5.4 Evaluasi Penentuan Jumlah Server (Max Server = 5)

Hasil simulasi penerapan konsolidasi beban kerja server dengan melakukan prediksi sebagai berikut, jumlah server yang melayani pada saat beban Min. adalah 1 server (effisien 80%), beban Med. adalah 2-3 server (effisien 60%), dan pada saat beban kerja Max. adalah 4-5 server (optimal 100%), (Gambar 5.1).

Hasil simulasi menunjukkan dengan penerapan konsolidasi beban kerja penggunaan server menjadi lebih efisien dan optimal.



Gambar 5.1 : Hasil simulasi jumlah server dengan penerapan konsolidasi beban kerja prediktif

## BAB 6

### KESIMPULAN DAN SARAN

#### 6.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, berikut ini merupakan beberapa hal yang dapat disimpulkan:

1. Hasil Pengukuran beban kerja kluster *server* diklasifikasikan menjadi 3 kelas, yaitu: Minimum (0-2), Medium (3-6), Maksimum ( $n > 7$ ). Metode yang digunakan untuk memprediksi kelas konsolidasi beban kerja *server* adalah *artificial neural network backpropagation* dengan berdasarkan parameter input yaitu: penggunaan CPU, *Memory*, jaringan (*throughput*) dan jumlah IP akses. Pengujian Metode ANN dilakukan menggunakan 1 dan 2 *hidden layer*; jumlah neuron *hidden layer* : 8, 16, 32, 64, 128, 256, 512; *learning rate* : 0.0001, 0.00001, 0.000001, dan 0.001.
2. Hasil pengujian ANN Backpropagation didapat arsitektur yang optimal adalah menggunakan 32 input, 2 *hidden layer* dengan jumlah neuron masing-masing H1 : 512; H2 : 32, 3 output, dan *learning rate* 0.00001. Arsitektur ini menghasilkan prediksi (klasifikasi) dengan tingkat *precision* : 90%, tingkat sensitivitas : 0.9, dan tingkat akurasi : 93%.
3. Terjadi tingkat kesalahan akurasi dalam klasifikasi sebesar 7%, hal ini dapat menyebabkan kesalahan dalam penentuan jumlah *server*. Dampak dari hal ini adalah dapat terjadi *bottleneck* atau *over-provisioning (server)* pada kluster web server dalam melayani jumlah permintaan.
4. Hasil simulasi menggunakan konsolidasi beban kerja server menghasilkan efisiensi penggunaan server (maks = 5) dibandingkan dengan tanpa menggunakan konsolidasi beban kerja. Jumlah server yang digunakan pada saat beban kerja minimum = 1-2 (efisiensi 80%), beban kerja medium = 2-3 (efisiensi 40%). Pada saat beban kerja maksimum jumlah server yang digunakan secara penuh.

## 6.2 Saran

Klasifikasi beban kerja *server* dapat dilakukan dengan optimal menggunakan pendekatan ANN *backpropagation*, namun masih terdapat beberapa peluang penelitian lanjutan yang dapat dikembangkan antara lain:

1. Menambahkan sistem kontrol yang akan menggunakan nilai bobot dari proses learning untuk mengatur kinerja dari *server* dalam kluster.
2. Masih dapat dilakukan pengembangan arsitektur ANN *backpropagation* sampai tahap prediksi dengan menambahkan parameter seperti: *request time client* ke *server*, *request time server* ke *upstream server*, besar data (*bytes*) dikirim *server* ke *client*.



## DAFTAR PUSTAKA

- A. Fox, A. Turner, and H. S. Kim. "Resource contention-aware virtual machine management for enterprise applications." *Proc. IEEE GLOBECOM*, 2012: pp. 1641–1646.
- A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari. "Server workload analysis for power minimization using consolidation." *Proc. USENIX Annu. Tech. Conf.*, 2009: pp. 28–28.
- A. Verma, P. Ahuja, and A. Neogi. "pMapper: Power and migration cost aware application placement in virtualized systems." *Springer-Verlag*, 2008: pp. 243–264.
- A. Wolke, M. Bichler, and T. Setzer. "Planning vs. dynamic control: Resource allocation in corporate clouds." *IEEE Trans. Cloud Comput.*, 2016: vol. 4, no. 3, pp. 322–335.
- Bichler, B. Speitkamp and M. "A mathematical programming approach for server consolidation problems in virtualized data centers." *IEEE Trans. Serv. Comput.*, 2010: vol. 3, no. 4, pp. 266–278.
- Buyya, A. Beloglazov and R. "Energy efficient resource management in virtualized cloud data centers." *Proc. IEEE 10th Int. Conf. Cluster, Cloud Grid Comput.*, 2010: pp. 826–831.
- C. Mastroianni, M. Meo, and G. Papuzzo. "Self-economy in cloud data centers: Statistical assignment and migration of virtual machines." *Springer-Verlag*, 2011: pp. 407–418.
- D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. "Resource pool management: Reactive versus proactive or let's be friends." *Comput. Netw.*, 2009: vol. 53, no. 17, pp. 2905–2922.
- G. Lovász, F. Niedermeier, and H. de Meer. "Performance tradeoffs of energy-aware virtual machine consolidation." *Cluster Comput.*, 2013: vol. 16, no. 3, pp. 481–496.
- Ida Wahyuni, Nakhel Rifqi Adam, Wayan Firdaus Mahmudy, Atiek Iriany. *Modeling Backpropagation Neural Network for Rainfall Prediction in Tengger East Java*. SIET, 2017.
- J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi. "Optimal update frequency model for physical machine state change and virtual machine placement in the cloud." *Proc. IEEE 8th Int. Conf. SoSE*, 2013: pp. 159–164.
- K. Halder, U. Bellur, and P. Kulkarni. "Risk aware provisioning and resource aggregation based consolidation of virtual machines." *Proc. IEEE 5th Int. Conf. CLOUD*, 2012: pp. 598–605.
- Kaiadi, Mehrzad. *Artificial Neural Networks Modelling for Monitoring and Performance Analysis of a Heat and Power Plant*. Lund Sweden: ISRN LUTMDN, 1990.
- M. Aldinucci, M. Torquati, M. Vanneschi, P. Zuccato. "The VirtualLinux Storage Abstraction Layer for Efficient Virtual Clustering." *Proceedings of the 16th Euromicro Conference on Parallel*, 2008: pp. 619–627.
- M. Cardosa, M. R. Korupolu, and A. Singh. "Shares and utilities based power consolidation in virtualized server environments." *Proc. IFIP/IEEE Int. Symp. IM*, 2009: pp. 327–334.
- S. K. Garg, A. N. Toosi, S. K. Gopalayengar, and R. Buyya. "SLAbased virtual machine management for heterogeneous workloads in a cloud datacenter." *J. Netw. Comput. Appl.*, 2014: vol. 45, pp. 108–120.
- Setyawan, Raden Arief. "Analisis Implementasi Load Balancing dengan Metode Source Hash Scheduling pada Protocol SSL." *EECCIS*, 2014: Vol. 8, No. 2.



- T. C. Ferreto, M. A. Netto, R. N. Calheiros, and C. A. De Rose. "Server consolidation with migration control for virtualized data centers." *Future Gener. Comput. Syst.*, 2011: vol. 27, no. 8, pp. 1027–1034.
- V. Ebrahimirad, M. Goudarzi, and A. Rajabi. "Energy-Aware Scheduling for Precedence-Constrained Parallel Virtual Machines in Virtualized Data Centers." *J. Grid Comput.*, 2015: vol. 13, no. 2, pp. 233–253.
- W. Deng, F. Liu, H. Jin, X. Liao, and H. Liu. "Reliability-aware server consolidation for balancing energy–lifetime tradeoff in virtualized cloud datacenters." *Int. J. Commun. Syst.*, 2014: vol. 27, no. 4, pp. 623–642.
- W. Xu, X. Zhu, S. Singhal, and Z. Wang. "Predictive control for dynamic resource allocation in enterprise data centers." *Proc. IEEE/IFIP 10th NOMS*, 2006: pp. 115–126.
- Walker, Ray. "Examining Load Average." 1 December 2006: 5.
- Wolke, T. Setzer and A. "Virtual machine re-assignment considering migration overhead." *Proc. IEEE NOMS*, 2012: pp. 631–634.
- Y. C. Lee and A. Y. Zomaya. "Energy efficient utilization of resources in cloud computing systems." *J. Supercomput.*, 2012: vol. 60, no. 2, pp. 268–280.
- Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun. "Multi-tiered on-demand resource scheduling for VM-based data center." *Proc. IEEE 9th Int. Symp. Cluster Comput. Grid*, 2009: pp. 148–155.
- Z. Gong, X. Gu, and J. Wilkes. "PRESS: Predictive elastic resource scaling for cloud systems." *Proc. IEEE Int. CNSM*, 2010: pp. 9–16.

