



**PERAMALAN CURAH HUJAN MENGGUNAKAN METODE
EXTREME LEARNING MACHINE**

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:

Rich Juniadi D S

NIM: 135150200111004



PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2020

PENGESAHAN

PERAMALAN CURAH HUJAN MENGGUNAKAN METODE *EXTREME LEARNING MACHINE*

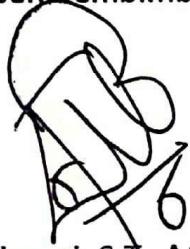
SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Rich Juniadi D S
NIM: 135150200111004

Skripsi ini telah diuji dan dinyatakan lulus pada
3 Januari 2020
Telah diperiksa dan disetujui oleh:

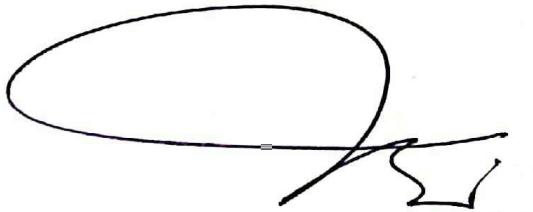
Dosen Pembimbing I



Tibyani, S.T., M.T.

NIP: 19691101 199512 1 002

Dosen Pembimbing II



Ir.Sutrisno, M.T.

NIP: 19570325 198701 1 001

Mengetahui

Ketua Jurusan Teknik Informatika



Tri Astoto Kurniawan, S.T., M.T., Ph.D.

NIP: 19710518 200312 1 001



Universitas Brawijaya
Universitas Brawijaya

Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya

Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini 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 disisipi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 03 Januari 2020

Rich Juniadi D S

NIM: 135150200111004



PRAKATA

Puji dan syukur penulis panjatkan kehadirat Tuhan YME atas anugerah serta limpahan rahmat-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "PERAMALAN CURAH HUJAN MENGGUNAKAN METODE *EXTREME LEARNING MACHINE*". Skripsi ini disusun sebagai syarat memperoleh gelar sarjana pada Program Studi Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.

Dalam proses penyelesaian skripsi ini penulis mendapatkan banyak bantuan, baik bantuan moral maupun materiil dari berbagai pihak. Oleh karena itu, penulis mengucapkan banyak terima kasih kepada:

1. Bapak Tibyani, S.T., M.T selaku Pembimbing I yang telah memberikan bimbingan, arahan, ilmu, dan masukan dalam penyelesaian skripsi ini.
2. Bapak Ir.Sutrisno, M.T. selaku Pembimbing II yang juga telah memberikan bimbingan, arahan, ilmu, dan masukan dalam penyelesaian skripsi ini.
3. Bapak Achmad Arwan, S.Kom., M.Kom selaku dosen pembimbing akademik.
4. Bapak Agus Wahyu Widodo, S.T., M.Cs, selaku Ketua Program Studi Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.
5. Bapak Tri Astoto Kurniawan, S.T., M.T, Ph.D, selaku Ketua Jurusan Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya .
6. Kedua orang tua yaitu ayah saya Pdt R.J Simamora, mamak saya R. br Simanullang dan beserta keluarga besar yang telah mendukung penulis dengan segala usahanya.
7. Teman-teman Program Studi Teknik Informatika yang selalu memberikan semangat, dukungan, dan kebersamaan selama Penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
8. M.Sanzabi, Husin Muhamad, I Putu Bagus, pak Aan, Benjamin Septa Pardamean, Andika Harlan, Galih Ariwanda, Mas Dino dan teman-teman F4.2 serta para sahabat yang tidak ada di Malang.
9. Semua pihak yang tidak dapat disebutkan satu per satu, yang telah membantu dan terlibat baik secara langsung maupun tidak langsung dalam penulisan skripsi ini.

Penulis menyadari bahwa skripsi ini tidak lepas dari kesalahan dan kekurangan. Oleh karena itu, penulis bersedia menerima kritik dan saran yang membangun untuk memperbaiki diri. Penulis berharap semoga skripsi ini dapat memberi manfaat.

Malang, 03 Januari 2020

Penulis
richjuniadi2@gmail.com



ABSTRAK

Rich Juniadi D S, Peramalan Curah Hujan Menggunakan Metode *Extreme Learning Machine*

Pembimbing: Tibyani,, S.T., M.T. dan Ir.Sutrisno, M.T.

Curah hujan adalah ketinggian air hujan yang terdapat dan terkumpul di tempat yang datar, tidak meresap, tidak menguap dan tidak mengalir. Informasi mengenai curah hujan sangat penting terutama dibidang pertanian dan sipil. Pada bidang pertanian informasi curah hujan digunakan untuk menentukan jenis tanaman yang akan ditanam sesuai dengan intensitas curah hujan, memprediksi awal musim tanam dalam kalender tanam untuk meminimalisir resiko penanaman. Pada bidang sipil, digunakan sebagai penentu standar rancang keteknikan dalam melakukan perencanaan bangunan pengendalian bencana banjir. Curah hujan di atas normal akan menimbulkan masalah bencana alam seperti banjir dan tanah longsor. Curah hujan adalah bagian dari elemen cuaca dan salah satu proses meteorologi yang cukup sulit untuk diprediksi. Maka dari itu, diperlukan peramalan curah hujan agar masyarakat dan pemerintah dapat melakukan tindakan pencegahan terhadap masalah yang ada. Proses peramalan terbagi menjadi beberapa proses yang antara lain normalisasi data, peramalan dengan algoritme *Extreme Learning Machine*, denormalisasi data dan hasil *error* dengan MAPE. Berdasarkan hasil pengujian menggunakan data curah hujan daerah Poncokusumo dengan rentang waktu tahun 2002 sampai 2015 diperoleh nilai MAPE terkecil sebesar 3.6852 %, dengan banyak fitur sebanyak 4, banyak *neuron* pada *hidden layer* sebanyak 2, persentase data *training* 80%.

Kata kunci: peramalan, curah hujan, *Extreme Learning Machine*, MAPE



ABSTRACT

Rich Juniasi D S, Rainfall Forecasting Using the Extreme Learning Machine Method

Supervisors: Tibyani., S.T., M.T. dan Ir.Sutrisno, M.T.

Rainfall is the height of rain water that is found and collected in a flat, not absorbed, does not evaporate and does not flow. Information about rainfall is very important especially in agriculture and civil. In agriculture, rainfall information is used to determine the type of plants to be planted in accordance with the intensity of rainfall, predicting the start of the growing season in the planting calendar to minimize the risk of planting. In the civil field, it is used as a determinant of engineering design standards in planning flood disaster control buildings. Above normal rainfall will cause natural disasters such as floods and landslides. Rainfall is part of the weather element and one of the meteorological processes that is quite difficult to predict. Rainfall forecasting is needed so that the community and the government can take preventative measures against the existing problems. The forecasting process is divided into several processes which include data normalization, forecasting with the Extreme Learning Machine algorithm, data denormalization and the results of errors with MAPE. Based on the test results using rainfall data in the Poncokusumo area with a span of years 2002 to 2015 obtained the smallest MAPE value of 3.6852%, with as many features as 4, many neurons in the hidden layer as much as 2, the percentage of training data 90%.

Keywords: forecasting, rainfall, Extreme Learning Machine, MAPE



DAFTAR ISI

PENGESAHAN	ii
PERNYATAAN ORISINALITAS	iii
PRAKATA.....	iv
ABSTRAK.....	v
ABSTRACT.....	vi
DAFTAR ISI.....	vii
DAFTAR TABEL.....	x
DAFTAR GAMBAR.....	xi
DAFTAR LAMPIRAN	xii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Tujuan	2
1.4 Manfaat.....	2
1.5 Batasan Masalah.....	3
1.6 Sistematika Pembahasan	3
BAB 2 LANDASAN KEPUSTAKAAN	5
2.1 Kajian Pustaka	5
2.2 Curah Hujan	7
2.2.1 Hujan	8
2.3 Teori Peramalan.....	8
2.3.1 Peramalan dalam Horison Waktu	9
2.3.2 Evaluasi Kinerja Peramalan	9
2.4 Jaringan Saraf Tiruan (JST)	9
2.4.1 Kemampuan JST.....	10
2.4.2 Model JST.....	10
2.4.3 Fungsi Aktivasi <i>Sigmoid Biner</i>	11
2.5 Normalisasi Data	12
2.6 Denormalisasi Data	12
2.7 Extreme Learning Machine (ELM).....	12



2.7.1 Proses <i>Training</i>	14
2.7.2 Proses <i>Testing</i>	15
BAB 3 METODOLOGI PENELITIAN	16
3.1 Tipe Penelitian	16
3.2 Lokasi Penelitian	16
3.3 Metode Penelitian Secara Umum.....	16
3.4 Pengumpulan Data	17
3.5 Peralatan Pendukung.....	17
3.6 Implementasi Algoritme	18
3.7 Pengujian	18
3.8 Pengambilan Kesimpulan dan Saran	18
BAB 4 Perancangan	19
4.1 Peramalan	19
4.1.1 Definisi Masalah	19
4.1.2 Pengumpulan Data.....	19
4.1.3 Formulasi Metode	20
4.1.4 Arsitektur Perancangan Sistem.....	21
4.2 Diagram Alir Algoritme ELM	22
4.2.2 Normalisasi.....	23
4.2.3 Inisialisasi <i>Input Weight</i> dan Bias	24
4.2.4 Proses <i>Training</i>	25
4.2.5 Menghitung Keluaran <i>Hidden Layer</i> dengan Fungsi Aktivasi	26
4.2.6 Menghitung <i>Output Weight</i>	28
4.2.7 Proses <i>Testing</i>	33
4.2.8 Menghitung Keluaran <i>Output Layer</i>	33
4.2.9 Proses Hitung Nilai <i>Mean Absolute Percentage Error</i> (MAPE).....	34
4.2.10 Proses Denormalisasi	35
4.3 Perhitungan Manual	36
4.3.1 Perhitungan Normalisasi Data	36
4.3.2 Inisialisasi <i>Input Weight</i> dan Bias.....	37

**DAFTAR TABEL**

Table 2.1 Kajian Pustaka	5
Table 2.2 Tingkatan hujan Berdasarkan Intensitasnya	8
Table 4.1 Data Latih dan Data Uji.....	20
Table 4.2 Data <i>Training</i> dan Data <i>Testing</i>	36
Table 4.3 Nilai Maksimal dan Minimal.....	36
Table 4.4 Normalisasi Data	37
Table 4.5 Matriks Nilai <i>Input Weight</i>	37
Table 4.6 Matriks Nilai Bias	37
Table 4.7 Matriks Keluaran <i>Hidden Layer</i>	38
Table 4.8 Matriks Keluaran <i>Hidden Layer</i> Dengan Fungsi Aktivasi	38
Table 4.9 Transpose Matriks Keluaran <i>Hidden Layer</i> dengan Fungsi Aktivasi.....	39
Table 4.10 Perkalian Matriks Hasil <i>Transpose</i> Dengan Keluaran <i>Hidden Layer</i> Dengan Fungsi Aktivasi	39
Table 4.11 Inverse Matriks.....	40
Table 4.12 Matriks <i>Moore-Penrose Generalized Inverse</i>	40
Table 4.13 Nilai <i>Output Weight</i>	40
Table 4.14 Matriks Keluaran <i>Hidden Layer</i>	41
Table 4.15 Matriks Keluaran <i>Hidden Layer</i> Dengan Fungsi Aktivasi	41
Table 4.16 Hasil Keluaran Pada <i>Output Layer</i>	42
Table 4.17 Rancangan Pengujian Jumlah Variabel <i>Lag Time</i>	43
Table 4.18 Rancangan Pengujian Perbandingan Jumlah Data <i>Training</i>	43
Table 4.19 Rancangan Pengujian Jumlah <i>neuron</i> Pada <i>Hidden Layer</i>	44
Table 6.1 Hasil Pengujian Persentase banyak Data <i>Training</i>	56
Table 6.2 Hasil Pengujian Jumlah Variabel <i>lag Time</i> (Fitur).....	58
Table 6.3 Hasil Pengujian Jumlah <i>Neuron</i> pada <i>Hidden Layer</i>	59



DAFTAR GAMBAR

Gambar 2.1 Model Matematis dari JST	11
Gambar 2.2 Fungsi <i>Sigmoid Biner</i>	12
Gambar 2.3 Struktur ELM	13
Gambar 3.1 Diagram Peramalan Curah Hujan dengan Metode ELM	17
Gambar 4.1 Arsitektur Perancangan Sistem ELM	21
Gambar 4.2 Diagram Alir Proses Peramalan Menggunakan ELM	22
Gambar 4.3 Diagram Alir Proses Normalisasi (Lanjutan)	24
Gambar 4.4 Diagram Alir Proses Inisialisasi <i>Input Weight</i> dan Bias (Lanjutan)	25
Gambar 4.5 Diagram Alir Proses <i>Training</i> (Lanjutan)	26
Gambar 4.6 Diagram Alir Proses Menghitung Keluaran <i>Hidden Layer</i> dengan Fungsi Aktivasi (Lanjutan)	27
Gambar 4.7 Diagram Alir Menghitung <i>Output Weight</i>	28
Gambar 4.8 Digram Alir Transpose Matriks Keluaran <i>Hidden layer</i>	29
Gambar 4.9 Diagram Alir Perkalian Matriks	30
Gambar 4.10 Diagram Alir Inverse Matriks Dari Hasil Perkalian	31
Gambar 4.11 Proses <i>Testing</i>	33
Gambar 4.12 Diagram Alir Menghitung Keluaran <i>Output Layer</i>	34
Gambar 4.13 Diagram Alir Proses Hitung Tingkat <i>Error</i> Menggunakan MAPE (Lanjutan)	35
Gambar 4.14 Diagram Alir Proses Denormalisasi	35



DAFTAR LAMPIRAN

LAMPIRAN A DATA CURAH HUJAN DASARIAN DAERAH PONCOKUSUMO

MALANG TAHUN 2000 SAMPAI 2015 65

LAMPIRAN B HASIL PENGUJIAN JUMLAH VARIABEL LAG TIME (FITUR) 67

LAMPIRAN C HASIL PENGUJIAN JUMLAH NEURON PADA HIDDEN LAYER 68



1.1 Latar Belakang

Indonesia adalah negara kepulauan yang berada di daerah iklim tropis dan rentan terhadap perubahan iklim. Perubahan iklim ini mengakibatkan terjadinya perubahan intensitas hujan dengan tidak menentu yang dipengaruhi oleh curah hujan. Curah hujan di atas normal akan menimbulkan masalah bencana alam seperti banjir dan tanah longsor. Seperti bencana banjir besar pada 2 Februari 2016. Adanya curah hujan yang berintensitas tinggi mengakibatkan terjadinya banjir dan tanah longsor di daerah air terjun Coban rondo, Kecamatan Pujon Malang (Liputan6, 2016). Tingginya curah hujan mengakibatkan meluapnya aliran sungai di daerah parang, Kabupaten Kediri. Peristiwa ini merusak jembatan, rumah warga, musola, tandon air bersih, saluran pipa air bersih dan 2 korban tewas (Tribunnews, 2017). Peristiwa yang terjadi di atas karena selain faktor alam juga terjadi karena kurangnya antisipasi dari masing-masing pihak.

Curah hujan adalah ketinggian air hujan yang terdapat dan terkumpul di tempat yang datar, tidak meresap, tidak menguap dan tidak mengalir (Ogi, 2011). Curah hujan adalah bagian dari elemen cuaca dan salah satu proses meteorologi yang cukup sulit untuk diprediksi (Jimoh, 2013). Melalui curah hujan dapat digunakan untuk menentukan suatu sifat hujan, intensitas dan normal curah hujan (BMKG, 2015).

Informasi mengenai curah hujan sangat penting terutama di bidang pertanian dan sipil. Pada bidang pertanian informasi curah hujan digunakan untuk menentukan jenis tanaman yang akan ditanam sesuai dengan intensitas curah hujan (Rani, K. B., & Govardhan, 2013), memprediksi awal musim tanam dalam kalender tanam untuk meminimalisir resiko penanaman (Nhita, 2015). Pada pertanian kentang di wilayah tengger dimana perubahan curah hujan yang tidak tentu menjadi suatu kendala karena memiliki dampak buruk pada produktivitas kentang (Ogi, 2011). Pada bidang sipil, informasi mengenai curah hujan digunakan sebagai penentu standar rancang keteknikan di dalam melakukan perencanaan bangunan pengendalian bencana banjir (Kusumawati & Sulistiowati, 2010). Pentingnya faktor curah hujan tersebut membuat para peneliti mengembangkan metode-metode dalam melakukan prediksi curah hujan dengan tingkat akurasi yang tinggi (Toth, 2005).

Peramalan dapat dilakukan dengan berbagai macam metode dan pemodelan dengan ilmu statistika atau kecerdasan buatan maupun kombinasi dari kedua pemodelan. Pada kecerdasan buatan terdapat metode jaringan saraf tiruan atau *artificial neural network* (ANN). Jaringan saraf tiruan mengadopsi sistem kerja pada otak manusia. Pada penelitian Sun dkk (2008) menjelaskan bahwa metode jaringan saraf tiruan lebih baik dalam melakukan peramalan dibandingkan metode-metode konvensional.

Penelitian sebelumnya melakukan perbandingan hasil peramalan permintaan data produksi kaos dan pin dengan metode *Extreme Learning Machine* (ELM) dengan *Moving Average* (MA) dan *Exponential Smoothing* (ES). Pada penelitian tersebut menunjukkan bahwa metode ELM menghasilkan peramalan dengan tingkat kesalahan *Mean Absolute Percentage Error* (MAPE) yang rendah jika dibandingkan dengan metode ES dan MA (Agustina, 2009), selain itu ELM juga memiliki *learning speed* yang relatif singkat (Agustina, 2009). Hasil keluaran (*output*) dengan metode ELM ditentukan dengan parameter-parameter seperti fungsi aktivasi pada arsitektur jaringan dan jumlah *hidden neuron*. Khotimah (2010) melakukan penelitian dengan membandingkan kinerja dari metode Jaringan saraf tiruan (JST) yaitu ELM dan *Backpropagation*. Pada penelitian tersebut dilakukan uji coba pelatihan dan pengujian dengan melakukan konfigurasi pada parameter *hidden layer* dan jumlah *epoch*. Berdasarkan penelitian tersebut diperoleh nilai kesalahan *Mean Square Error* (MSE) dan *Mean Absolute Percentage Error* (MAPE) dengan metode ELM lebih kecil dibandingkan dengan metode *Backpropagation* (Khotimah, 2010).

Berdasarkan penelitian yang telah dilakukan sebelumnya, peneliti mengusulkan penelitian peramalan curah hujan menggunakan metode *Extreme Learning Machine* (ELM). Diharapkan nantinya penelitian ini akan memperoleh tingkat akurasi yang lebih baik dalam melakukan peramalan curah hujan berdasarkan penelitian yang telah dilakukan sebelumnya.

1.2 Rumusan Masalah

Berdasarkan uraian atau penjelasan yang ada pada latar belakang, dapat dirumuskan bahwa permasalahan yang ada sebagai berikut.

1. Bagaimana pengaruh pengujian jumlah data *training*, variabel *lag time* (fitur) dan jumlah *neuron* terhadap metode *Extreme Learning Machine* dalam peramalan curah hujan?
2. Bagaimana tingkat akurasi metode *Extreme Learning Machine* dalam peramalan curah hujan?

1.3 Tujuan

Berdasarkan uraian yang ada pada rumusan masalah, dapat dirumuskan bahwa tujuan dari penelitian ini sebagai berikut.

1. Melakukan pengujian dan mengetahui pengaruh jumlah data *training*, variabel *lag time* (fitur) dan jumlah *neuron* yang optimal pada metode *Extreme Learning Machine* dalam peramalan curah hujan.
2. Melakukan perhitungan tingkat akurasi metode *Extreme Learning Machine* dalam peramalan curah hujan.

1.4 Manfaat

Memberikan pengetahuan dalam penerapan algoritme *Extreme Learning Machine* informasi, dapat menerapkan algoritme *Extreme Learning Machine*

(*ELM*) untuk peramalan curah hujan, dan melalui penelitian ini dapat memberikan kontribusi pada penelitian selanjutnya.

1.5 Batasan Masalah

Berikut adalah batasan-batasan masalah dari penelitian ini.

1. Data yang digunakan adalah data yang diperoleh dari Stasiun Klimatologi BMKG Karangploso Malang di daerah Poncokusumo dengan rentang waktu tahun 2002 sampai 2015.
2. Parameter yang digunakan adalah berdasarkan data curah hujan perdasarian (data selama 10 hari) dengan satuan millimeter.

1.6 Sistematika Pembahasan

Untuk mencapai tujuan yang diharapkan, maka sistematika pembahasan yang disusun dalam penelitian ini sebagai berikut.

BAB 1 PENDAHULUAN

Bab ini menjelaskan latar belakang dari masalah mengenai *forecasting* (peramalan) dengan metode *Extreme Learning Machine* sesuai rumusan masalah, tujuan, dan manfaat dalam melakukan penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menjelaskan tentang kajian pustaka yang terkait dengan penelitian-penelitian yang telah dilakukan sebelumnya. Menjelaskan dasar teori yang diperlukan dalam penelitian ini seperti peramalan, curah hujan, Jaringan Saraf Tiruan (JST), *Extreme Learning Machine* (*ELM*).

BAB 3 METODOLOGI PENELITIAN

Bab ini membahas tentang metode *Extreme Learning Machine* yang digunakan dalam penelitian ini dan langkah kerja yang akan dilakukan.

BAB 4 PERANCANGAN

Bab ini membahas tentang perancangan sistem yang dilakukan dalam proses peramalan curah hujan menggunakan metode *Extreme Learning Machine*, bagaimana perancangan antarmuka dan perancangan pengujian.

BAB 5 IMPLEMENTASI

Bab ini membahas tentang implementasi metode *Extreme Learning Machine* dalam melakukan komputasi berupa program untuk peramalan curah hujan.

BAB 6 PENGUJIAN DAN PEMBAHASAN

Bab ini membahas tentang evaluasi dan kinerja program dengan melakukan pengujian akurasi dari peramalan curah hujan menggunakan metode *Extreme Learning Machine* serta melakukan analisis terhadap pengujian yang akan dilakukan.



BAB 7 PENUTUP

Bab ini membuat kesimpulan yang diperoleh dari penelitian yang akan dilakukan tentang peramalan curah hujan menggunakan metode *Extreme Learning Machine*. Memberikan saran terhadap pengembangan penelitian ini untuk penelitian selanjutnya.



BAB 2 LANDASAN KEPUSTAKAAN				
Landasan kepustakaan berisi uraian dan pembahasan tentang teori, konsep, model, metode, atau sistem dari pustakailmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian ini.				
2.1 Kajian Pustaka				
Pada bab kajian pustaka membahas tentang penelitian yang telah dilakukan sebelumnya mengenai tentang topik yang terkait dengan penulisan penelitian ini. Kajian pustaka pada penelitian ini dijelaskan pada Tabel 2.1.				
No	Judul	Obyek (input)	Metode (Proses)	Hasil (<i>Output</i>)
1	<i>Sales Forecasting using Extreme Learning Machine with Application in Fashion Retailing. Elsevier Decision Support Systems</i> (Sun, 2008)	Penjualan fashion retail (penjualan eceran)	<i>Extreme Learning Machine</i> (ELM)	Pada penelitian ini metode ELM digunakan untuk menganalisis hubungan antara jumlah penjualan dengan beberapa faktor yang penting dan memengaruhi permintaan. Hasil pada penelitian ini menunjukkan bahwa metode ELM lebih baik dari beberapa metode peramalan penjualan lainnya.
2	<i>Protein Sequence Classification with Improved Extreme Learning Machine Algorithm</i> (Cao & Xiong, 2014)	Protein Sekuensial	<i>Extreme Learning Machine</i> (ELM)	Di dalam penelitian ini metode ELM digunakan untuk melakukan klasifikasi protein sekuensial dan melakukan perbandingan dengan beberapa metode yang dikombinasikan dengan ELM. Pada hasil akhir menunjukkan ELM memiliki <i>learning speed</i> yang lebih baik.
3	<i>Application of Extreme Learning Machine for Estimation of</i>	Kecepatan distribusi angin	<i>Extreme Learning Machine</i> (ELM)	Pada penelitian ini menggunakan metode ELM untuk memperoleh suatu nilai dengan bentuk dan skala

	<i>Wind Speed Distribution</i> (Shamshirband, 2015)			(c) melalui kecepatan angin, penelitian ini menghasilkan efisiensi dan akurasi yang baik. Distribusi kecepatan angin pada penelitian ini adalah hal khusus dalam evaluasi energi angin.
4	Penerapan Metode <i>Extreme Learning Machine</i> untuk Peramalan Permintaan (Agustina, 2009)	Data Produksi kaos dan pin	<i>Extreme Learning Machine</i> (ELM)	Pada penelitian ini metode ELM menghasilkan nilai MAPE yang rendah jika dibandingkan dengan metode peramalan konvensional seperti <i>Exponential Smoothing</i> dan <i>Moving Average</i> . Di dalam penelitian ini, output dihasilkan melalui penentuan parameter seperti fungsi aktivasi dan jumlah neuron tersembunyi
5	Sistem Pendukung Keputusan Peramalan Jumlah Kunjungan Pasien Menggunakan Metode <i>Extreme Learning Machine</i> (Studi Kasus : Poli Gigi RSU Dr. Wahidin Sudiro Husodo Mojokerto) (Fardani, 2015)	Jumlah kunjungan Pasien	<i>Extreme Learning Machine</i> (ELM)	Di dalam Penelitian ini dihasilkan nilai error MSE terbaik sebesar 0.027 pada fungsi aktivasi <i>sigmoid biner</i> dan dengan jumlah hidden layer dengan 7 unit.

Berdasarkan penelitian di atas, ditemukan beberapa hasil yang relevan untuk mendukung penulisan penelitian ini seperti penelitian yang dilakukan oleh Sun dkk (2008), Pada penelitian tersebut melakukan peramalan penjualan yang digunakan untuk mengetahui hubungan antara jumlah penjualan dan beberapa faktor yang mempengaruhi jumlah permintaan, pada penelitian ini termasuk faktor desain. Pada penelitian ini menunjukkan bahwa metode ELM lebih baik dibandingkan dengan beberapa metode Jaringan Saraf T iruan yang digunakan untuk peramalan penjualan seperti *Backpropagation* dan *Adaline*.

Pada penelitian kedua dilakukan oleh Cao & Xiong (2014) melakukan penelitian tentang pengelompokan protein sekuensial. Di dalam penelitian ini metode ELM adalah metode tercepat dan pada proses pelatihan dari *dataset* urutan protein memerlukan waktu kurang dari satu detik. Pada metode *Backpropagation* membutuhkan waktu pelatihan lebih dari satu detik dengan kernel linear dan kernel Gaussian, begitu juga dengan kernel Sigmoid lebih dari satu detik. ELM dimodifikasi menjadi VOP-ELM dan OP-ELM menghasilkan waktu pelatihan yang justru meningkat sangat tinggi karena mencari jumlah *hidden neuron* yang optimal.

Penelitian ketiga dilakukan oleh Shamshirband dkk (2015) tentang distribusi angin. Energi angin merupakan salah satu sumber daya energi yang dapat dimanfaatkan oleh negara-negara sebagai dasar energi yang berkelanjutan.

Penelitian ini mencari nilai prediksi pada parameter fungsi distribusi angin.

Parameter yang digunakan yaitu bentuk (*k*) dan skala (*c*) dari fungsi Weibull yang digunakan untuk menghitung potensi tenaga angin. Hasil akhir menunjukkan metode ELM jauh lebih cepat dalam proses *training* dibandingkan dengan algoritme pembelajaran *feedforward* tradisional seperti *Backpropagation*.

Pada penelitian keempat dilakukan oleh Agustina dkk. (2009) dengan melakukan peramalan permintaan. Di dalam penelitian ini *output* dari metode ELM ditentukan melalui parameter seperti fungsi aktivasi dan jumlah neuron tersembunyi. Hasil *output* peramalan pada penelitian ini berdasarkan nilai *error* MAPE dengan tingkat kesalahan yang rendah. Nilai *error* MAPE berada 0.0042% pada produk kaos dan 0.0095% pada produk lainnya. Pada *learning speed* yang dihasilkan juga sangat singkat yaitu berada di rata-rata 0.0059 detik.

Penelitian terakhir dilakukan oleh Fardani (2015) tentang sistem pendukung keputusan peramalan jumlah kunjungan pasien dengan metode ELM. Di dalam penelitian ini evaluasi dilakukan dengan melakukan perbandingan data hasil peramalan dengan data aktual. Pada penelitian ini data yang digunakan adalah 116 data pengujian yang dilakukan uji coba berdasarkan fungsi aktivasi dan jumlah *layer* tersembunyi. Melalui penelitian ini hasil optimal didapatkan dengan nilai *error* MSE sebesar 0.027 dengan fungsi aktivasi *sigmoid biner* dan jumlah *layer* tersembunyi berjumlah 7 unit dengan 500 *epoch*.

2.2 Curah Hujan

Curah hujan merupakan ketinggian air hujan yang terkumpul dalam tempat yang datar, tidak menguap, tidak meresap, dan tidak mengalir (Agustina, 2013). Curah hujan 1 (satu) milimeter artinya dalam luasan satu meter persegi pada tempat yang datar tertampung air setinggi satu milimeter atau tertampung air sebanyak satu liter (Agustina, 2013). Intensitas hujan adalah banyaknya curah hujan persatuan jangka waktu tertentu (Agustina, 2013). Apabila dikatakan intensitasnya besar berarti hujan lebat dan kondisi ini sangat berbahaya karena berdampak dapat menimbulkan banjir, longsor dan efek negatif terhadap tanaman.

2.2.1 Hujan

Hujan merupakan salah satu fenomena alam yang terdapat dalam siklus hidrologi dan sangat dipengaruhi iklim. Keberadaan hujan sangat penting dalam kehidupan, karena hujan dapat mencukupi kebutuhan air yang sangat dibutuhkan oleh semua makhluk hidup (Mulkan, 2010). Hujan adalah hydrometeor yang jatuh berupa partikel-partikel air yang mempunyai diameter 0,5 mm atau lebih.

Hujan yang sampai ke permukaan tanah dapat diukur dengan jalan mengukur tinggi air hujan tersebut dengan berdasarkan volume air hujan per satuan luas (Mulkan, 2010). Hasil dari pengukuran tersebut dinamakan dengan curah hujan.

Curah hujan merupakan salah satu unsur cuaca yang datanya diperoleh dengan cara mengukurnya dengan menggunakan alat penakar hujan, sehingga dapat diketahui jumlahnya dalam satuan millimeter (mm) (Mulkan, 2010). Curah hujan 1 mm adalah jumlah air hujan yang jatuh di permukaan per satuan luas (m^2) dengan catatan tidak ada yang menguap, meresap atau mengalir (Mulkan, 2010). Curah hujan dibatasi sebagai tinggi air hujan yang diterima di permukaan sebelum mengalami aliran permukaan, evaporasi dan peresapan ke dalam tanah (Mulkan, 2010).

Jenis-jenis hujan berdasarkan besarnya curah hujan menurut BMKG dibagi menjadi tiga, yaitu (Mulkan, 2010).

1. Hujan sedang, 20 - 50 mm per hari.
2. Hujan lebat, 50-100 mm per hari.
3. Hujan sangat lebat, di atas 100 mm per hari.

Intensitas curah hujan merupakan ukuran jumlah hujan per satuan waktu tertentu selama hujan berlangsung. Hujan umumnya dibedakan menjadi 5 tingkatan sesuai intensitasnya seperti yang ditunjukkan pada Tabel 2.2.

Table 2.2 Tingkatan hujan Berdasarkan Intensitasnya

Tingkatan	Intensitas (mm/menit)
Sangat lemah	0,02
Lemah	0,02-0,05
Sedang	0,05-0,25
Deras	0,25-1
Sangat deras	>1

(Sumber: Mori, 1997)

2.3 Teori Peramalan

Peramalan adalah proses untuk memperkirakan berapa kebutuhan di masa akan datang yang meliputi kebutuhan dalam ukuran kuantitas, kualitas, waktu, dan lokasi yang dibutuhkan dalam rangka memenuhi permintaan barang atau jasa (Fardani, 2012).

Dalam industri, peramalan berguna untuk (Fardani, 2012).

1. Peramalan Produksi.
2. Peramalan Bahan Baku.

3. Peramalan Anggaran Biaya.
4. Peramalan Pemasaran.

2.3.1 Peramalan dalam Horison Waktu

Dalam hubungannya dengan horizon waktu peramalan, maka kita bisa mengklasifikasikan peramalan tersebut ke dalam 3 kelompok (Asty, 2014)

1. Peramalan jangka panjang, umumnya 2 sampai 10 tahun. Peramalan ini digunakan untuk perencanaan produk dan perencanaan sumber daya.
2. Peramalan jangka menengah, umumnya 1 sampai 24 bulan. Dalam industri peramalan ini lebih mengkhusus dibandingkan peramalan jangka panjang, biasanya digunakan untuk menentukan aliran kas, perencanaan produksi dan penentuan anggaran.
3. Peramalan jangka pendek, umumnya 1 sampai 5 minggu. Peramalan ini digunakan untuk mengambil keputusan dalam hal perlu tidaknya lembur, penjadwalan kerja, dan lain – lain keputusan kontrol jangka pendek.

2.3.2 Evaluasi Kinerja Peramalan

Ukuran akurasi hasil peramalan yang merupakan ukuran kesalahan peramalan merupakan ukuran tentang tingkat perbedaan antara hasil peramalan dengan yang sebenarnya terjadi.

Mean Absolute Percentage Error (MAPE)

Mape merupakan proses evaluasi kinerja suatu peramalan dengan menghitung nilai *error*. proses evaluasi nilai *error* dengan cara menghitung selisih antara nilai ramalan dengan nilai aktual yang selanjutnya diabsolutkan dan lalu diubah dalam bentuk persentase terhadap data asli (Darmayanti, Setiawan & Bachtiar, 2018). Menghitung nilai MAPE dapat dilihat pada Persamaan 2.1.

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_{i,\text{pred}} - y_{i,\text{act}}}{y_{i,\text{act}}} \right| \times 100 \quad (2.1)$$

Keterangan:

y_i, pred : Nilai data hasil peramalan ke- i

y_i, act : Nilai data aktual ke- i

n : Banyaknya data *testing*

i : Indeks data

2.4 Jaringan Saraf Tiruan (JST)

JST adalah paradigma pengolahan informasi yang terinspirasi oleh sistem saraf secara biologis, seperti proses informasi pada otak manusia. Elemen kunci dari paradigma ini adalah struktur dari sistem pengolahan informasi yang terdiri dari sejumlah besar elemen pemrosesan yang saling berhubungan (*neuron*), bekerja serentak untuk menyelesaikan masalah tertentu. Cara kerja JST sama seperti cara kerja manusia, yaitu belajar melalui contoh. Sebuah JST dikonfigurasikan untuk aplikasi tertentu, seperti pengenalan pola atau aplikasi data, melalui proses pembelajaran. Belajar dalam sistem biologis melibatkan

penyesuaian terhadap koneksi *synaptic* yang ada antara *neuron*. Hal ini berlaku juga untuk JST (Khotimah, 2017).

2.4.1 Kemampuan JST

JST mempunyai kemampuan yang baik untuk mendapatkan informasi dari data yang rumit atau tidak tepat, mampu menyelesaikan permasalahan yang tidak terstruktur dan sulit didefinisikan, dapat belajar dari pengalaman, mampu mengakuisisi pengetahuan walaupun tidak ada kepastian, mampu melakukan generalisasi dan ekstrasi dari suatu pola data tertentu, dapat menciptakan suatu pola pengetahuan melalui pengaturan diri atau kemampuan belajar (*self organizing*), mampu memilih suatu input data ke dalam kategori tertentu yang sudah ditetapkan (*klasifikasi*), mampu menggambarkan objek secara keseluruhan maupun hanya diberikan sebagian data dari objek tersebut (*asosiasi*), mempunyai kemampuan mengolah data-data input tanpa harus mempunyai target (*self organizing*) dan mampu menemukan jawaban terbaik sehingga mampu meminimilasi fungsi biaya (*optimasi*) (Khotimah, 2017).

Kelebihan-kelebihan yang diberikan oleh JST antara lain (Khotimah, 2017).

1. *Belajar Adaptive*

Kemampuan untuk mampu mempelajari bagaimana melakukan pekerjaan berdasarkan data yang diberikan untuk pelatihan atau pengalaman awal.

2. *Self-Organization*

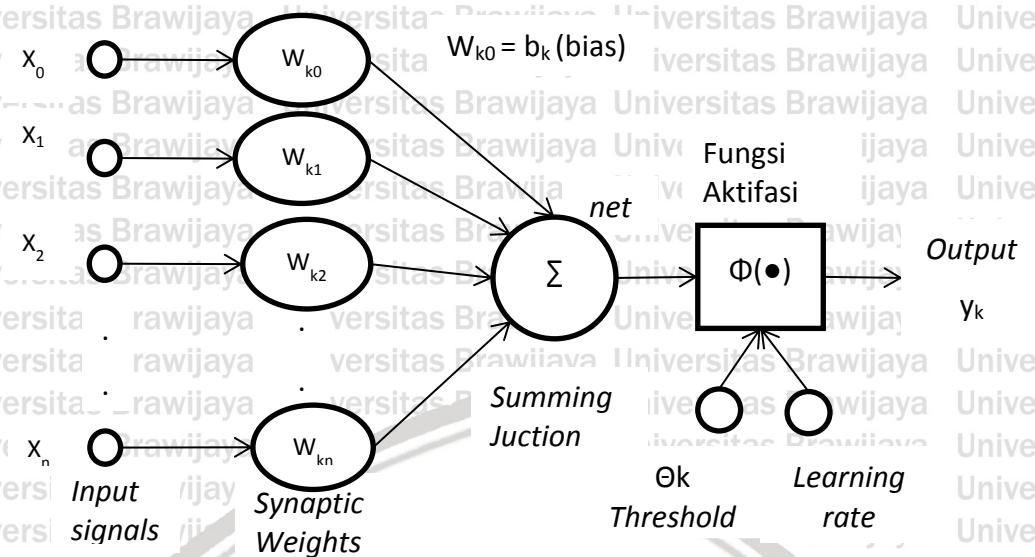
Sebuah JST dapat membuat organisasi sendiri atau representasi dari informasi yang diterimanya selama waktu belajar.

3. *Real Time Operation*

Perhitungan JST dapat dilakukan secara paralel sehingga perangkat keras yang dirancang dan diproduksi secara khusus dapat mengambil keuntungan dari kemampuan ini.

2.4.2 Model JST

Terdapat tiga komponen dasar penting ketika membuat sebuah model fungsional dari *neuron* biologis. Pertama, sinapsis *neuron* dimodelkan sebagai bobot. Kekuatan hubungan antara masukan dan neuron ditentukan oleh nilai bobot. Nilai bobot negatif mencerminkan koneksi hambat, sedangkan nilai-nilai positif menandakan koneksi rangsang sel. Komponen kedua adalah penjumlahan semua masukan yang dimodifikasi oleh masing-masing bobot. Kegiatan ini disebut sebagai kombinasi linear. Komponen ketiga bertindak sebagai fungsi kontrol aktifitas amplitudo *output* dari *neuron* (Khotimah, 2017). Model matematis dari arsitektur jaringan saraf tiruan ditunjukkan pada Gambar 2.1.



Gambar 2.1 Model Matematis dari JST

(Sumber: Khotimah, 2017)

Dari model Gambar 2.1 aktifasi interval *neuron* dapat ditunjukkan dengan Persamaan 2.2 (Khotimah, 2017).

$$net = \sum_{j=1}^i w_{kj}x_j$$

Keterangan

- net* : Matriks jaringan saraf tiruan.
- k* : $[1, 2, \dots, n]$ n adalah jumlah data.
- j* : $[1, 2, \dots, n]$ n adalah jumlah *hidden neuron*.
- w* : Bobot *input*
- x* : *Input*

Setelah *net* melewati fungsi aktivasi tertentu, *output neuron* adalah y_k .

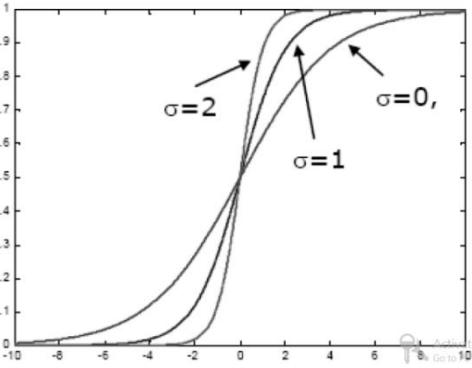
2.4.3 Fungsi Aktivasi *Sigmoid Biner*

Perilaku dari JST ditentukan oleh bobot dan *input-output* fungsi aktifasi yang ditetapkan. Beberapa fungsi aktifasi yang sering digunakan dalam JST seperti fungsi *sin*, fungsi *hardlim* dan fungsi *sigmoid biner*. Pada penelitian ini fungsi yang digunakan adalah fungsi *sigmoid biner* karena fungsi aktivasi ini memperoleh nilai *error* paling kecil dibandingkan dengan fungsi aktivasi lainnya (Cao et al., 2017).

JST yang membutuhkan nilai *output* yang terletak antara 0 sampai 1 sering kali menggunakan fungsi sigmoid biner karena fungsi ini memiliki nilai pada range 0 sampai 1 yang ditunjukkan pada Gambar 2.2. Secara matematis, fungsi sigmoid biner ditunjukkan dengan Persamaan 2.3 (Cao et al., 2017).

$$y = f(x) = \frac{1}{1 + e^{-\sigma x}}$$

(2.3)



Gambar 2.2 Fungsi Sigmoid Biner

2.5 Normalisasi Data

Normalisasi adalah proses penskalaan nilai atribut dari data sehingga bisa jatuh pada *range* tertentu (Patro, 2015). Proses normalisasi dilakukan karena perhitungan yang digunakan menghasilkan keluaran dengan *range* data antara 0 dan 1. Proses perhitungan normalisasi data ditunjukkan pada Persamaan 2.4.

$$X' = \frac{(X - X_{min})}{(X_{max} - X_{min})}$$

Keterangan:

X' = Data hasil normalisasi

X = Data asli

X_{max} = Nilai maksimum data asli

X_{min} = Nilai manimum data asli

(2.4)

2.6 Denormalisasi Data

Data hasil normalisasi kemudian dilakukan denormalisasi. Denormalisasi data adalah proses untuk mengembalikan nilai data menjadi nilai semula atau nilai yang sebenarnya berdasarkan hasil peramalan yang akan dilakukan. Proses perhitungan denormalisasi data ditunjukkan pada Persamaan 2.5.

$$X = (X'(X_{max} - X_{min})) + X_{min}$$

Keterangan:

X' = Data hasil normalisasi

X = Data asli

X_{max} = Nilai maksimum data asli

X_{min} = Nilai manimum data asli

(2.5)

2.7 Extreme Learning Machine (ELM)

Metode *Extreme Learning Machine* (ELM) adalah metode pembelajaran yang termasuk dalam jaringan saraf tiruan. Metode ini pertama kali diperkenalkan oleh Huang (2014). *Extreme Learning Machine* adalah jaringan saraf tiruan *feedforward* sederhana dengan satu lapisan tersembunyi (*hidden layer*) yang lebih dikenal dengan *Single Hidden Layer Feedforward Neural Networks* (SLFNs) (Sun, 2008).



Metode *Extreme Learning Machine* (ELM) dirancang untuk mengurangi kelemahan-kelemahan Jaringan Saraf Tiruan *feedforward* terutama untuk kecepatan *learning*-nya. Pada penelitian Huang (2004) mengemukakan alasan mengapa Jaringan Saraf Tiruan memiliki *learning speed* yang rendah, sebagai berikut.

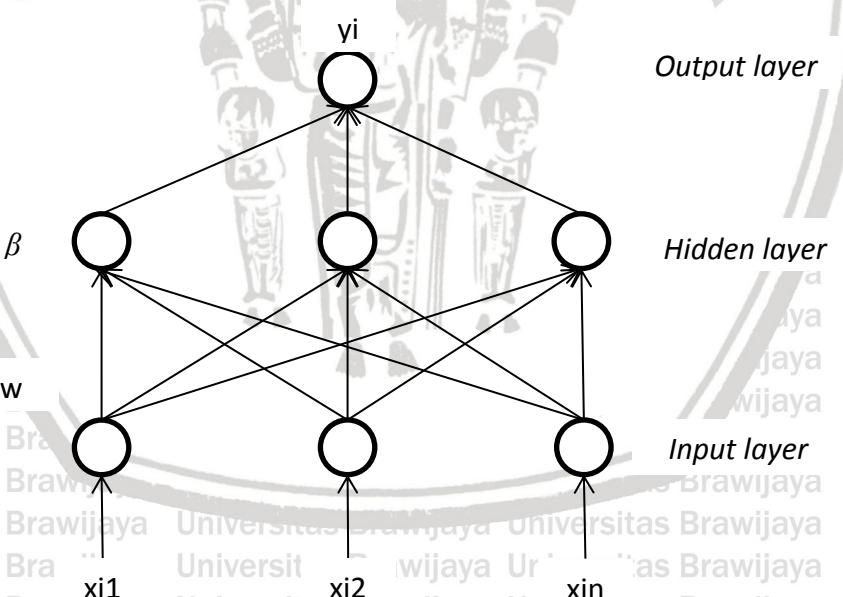
1. JST *feedforward* menggunakan *slow gradient based learning algorithm* untuk melakukan *training*.

2. Parameter pada jaringan JST *feedforward* ditentukan secara iteratif dengan menggunakan metode *training* tersebut.

Secara umum, pada jaringan *feedforward* semua parameter ditentukan secara *manual*. Parameter tersebut adalah *input weight* dan bias. Parameter-parameter tersebut saling berhubungan antara layer yang satu dengan yang lain, hal ini mengakibatkan *learning speed* yang lama (Huang, 2004).

Pada metode *Extreme Learning Machine* (ELM) parameter *input weight* dan *hidden bias* dipilih secara *random* dari nilai tertentu, sehingga ELM memiliki *learning speed* yang cepat dan dapat menghasilkan *good generalization performance*. Melalui pemberian nilai *random* pada *range* tertentu, dapat menghindari hasil prediksi yang tidak stabil (Huang, 2004).

Secara umum model JST pada metode *Extreme Learning Machine* (ELM) sebagai metode pembelajaran dapat dilihat pada Gambar 2.3 (Huang, 2004).



Gambar 2.3 Struktur ELM

(Sumber: Khotimah, 2013)

Keterangan:

x_i : Input jaringan

w : Bobot jaringan

y : Output jaringan

β : Bias jaringan



2.7.1 Proses Training

Tujuan dari proses *training* adalah mendapatkan *output weight* dengan tingkat kesalahan yang rendah. Langkah-langkah yang perlu dilakukan dalam proses *training* pada metode ELM adalah sebagai berikut (Huang, 2004).

1. Melakukan inisialisasi *input weight* dan bias dengan bilangan acak yang kecil dengan *range* 0 sampai dengan 1 (Liang, 2006).
2. Menghitung nilai *output* pada *hidden layer* ditunjukkan pada Persamaan 2.6.

$$H_{init\ ij} = \left(\sum_{i=1}^n x_j w_i^T \right) + b_j \quad (2.6)$$

Keterangan:

H_{init} : Matriks keluaran *hidden layer*.

i : $[1, 2, \dots, n]$ n adalah jumlah data.

j : $[1, 2, \dots, n]$ n adalah jumlah *hidden neuron*.

w^T : Bobot *input transpose*

x : *Input* data

b : Bias

3. Menghitung semua nilai keluaran pada *hidden layer* dengan menggunakan fungsi aktivasi pada Persamaan 2.7. Perhitungan dengan fungsi aktivasi ditunjukkan pada Persamaan 2.7.

$$H(x) = \frac{1}{1 + e^{-x}}$$

Keterangan:

$H(x)$: Matriks *output hidden layer*

x : *Input*

4. Menghitung Matriks *Moore-Penrose Generalized Inverse*. Matriks ini didapatkan dari perkalian matriks *transpose* dari hasil *output hidden layer* dengan fungsi aktivasi. Perhitungan matriks *Moore-Penrose Generalized Inverse* ditunjukkan pada Persamaan 2.8.

$$H^+ = (H(x)^T \cdot H(x))^{-1} H(x)^T$$

(2.7)

Keterangan:

H^+ : Matriks *Moore-Penrose Generalized Inverse*

$H(x)$: Matriks *output hidden layer*

5. Menghitung *output weight* dari *hidden layer* ke *output layer*. Sebelum menghitung *output weight*, terlebih dahulu menghitung matriks *Moore-Penrose Generalized Inverse* dari hasil keluaran *hidden layer* dengan fungsi aktivasi. Kemudian dihitung *output weight* yang ditunjukkan pada Persamaan 2.9.

$$\beta = H^+ T$$

(2.8)

Keterangan:

β : Matriks *output weight*

H^+ : Matriks *Moore-Penrose Generealized Inverse* dari matriks H

T : Matriks Target

(2.9)

6. Menghitung *Output*. Output peramalan diperoleh melalui perkalian matriks *output hidden layer* dengan matriks *output weight* ditunjukkan pada Persamaan 2.10

$$O = H(x)\beta$$

Keterangan:

O : Output Peramalan

$H(x)$: Matriks keluaran *hidden layer*

β : Matriks *output weight*

2.7.2 Proses Testing

Pada proses *testing* dilakukan dengan berdasarkan *input weight*, bias, dan *output weight* yang sesuai dengan hasil perhitungan *training*. Proses *training* (pembelajaran) bertujuan untuk mengembangkan model dari *Extreme Learning Machine* (ELM), sedangkan proses *testing* (pengujian) bertujuan untuk melakukan evaluasi kemampuan dari *Extreme Learning Machine* (ELM) sebagai metode untuk melakukan peramalan. Berikut adalah langkah-langkah dari proses *testing* (Huang, 2004).

1. Menggunakan *input weight* dan bias yang telah dihitung pada proses *training*
2. Melakukan perhitungan semua nilai *output* pada *hidden layer* dengan menggunakan Persamaan 2.6 dan dengan fungsi aktivasi ($H(x)$) pada Persamaan 2.7.
3. Menggunakan hasil perhitungan proses *training* yaitu *output weight* dari *hidden layer* ke *output layer*. Melakukan perhitungan nilai keluaran di *output layer* yang merupakan nilai keluaran (*output*) hasil prediksi yang ditunjukkan pada Persamaan 2.11.

$$y = H(x)\beta$$

Keterangan:

y : Keluaran (*output*) hasil dari peramalan

β : Keluaran (*output*) *weight*

$H(x)$: keluaran (*output*) *hidden layer* dengan fungsi aktivasi

4. Menghitung nilai *error* pada semua layer keluaran (*output layer*) dengan menggunakan Persamaan 2.1.

BAB 3 METODOLOGI PENELITIAN

Pada bab ini menjelaskan tentang langkah-langkah yang digunakan dalam penelitian peramalan curah hujan menggunakan metode *Extreme Learning Machine*. Metodologi penelitian menjelaskan tentang tipe penelitian yang dilakukan, strategi dan rancangan penelitian. Rancangan penelitian yang dilakukan dalam penelitian ini melalui beberapa tahapan yaitu metode secara umum, pengumpulan data, perancangan, implementasi, pengujian, dan pembuatan laporan.

3.1 Tipe Penelitian

Penelitian ini menggunakan tipe non-implementatif analitik. Penelitian nonimplementatif menitikberatkan pada analisis terhadap hubungan antar fenomena yang sedang dikaji untuk kemudian menghasilkan hasil analisis ilmiah sebagai produk atau artefak utamanya. Metode atau teknik yang digunakan untuk menghasilkan produk atau artefak utama yaitu observasi data curah hujan dari Badan Meteorologi Klimatologi dan Geofisika (BMKG) stasiun klimatologi Karangploso Malang.

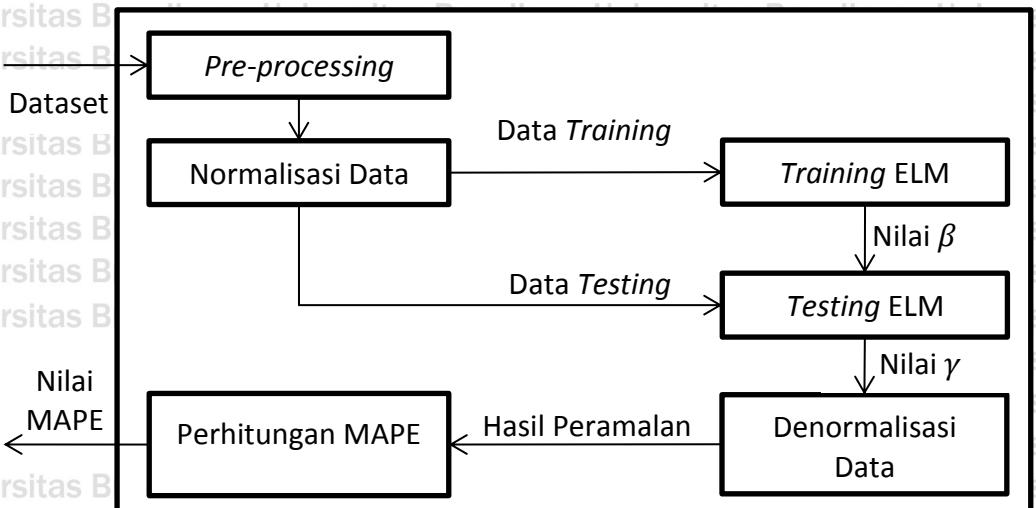
3.2 Lokasi Penelitian

Proses penelitian ini dilakukan di laboratorium riset Fakultas Ilmu Komputer Universitas Brawijaya dan di ruang F 4.2 Fakultas Ilmu Komputer Universitas Brawijaya. Melalui lokasi penelitian memberikan informasi dan kebutuhan lainnya dalam penelitian peramalan curah hujan.

3.3 Metode Penelitian Secara Umum

Peramalan curah hujan dilakukan dengan metode *Extreme Learning machine (ELM)*. Data yang digunakan adalah data hasil observasi Badan Meteorologi Klimatologi dan Geofisika (BMKG) stasiun klimatologi Karangploso Malang pada daerah Poncokusumo. Data yang digunakan pada penelitian ini adalah data curah hujan dalam bentuk *time series* dan dalam dasarian (10 hari).

Data pada penelitian ini akan dibagi menjadi beberapa fitur dan target, setelah itu dilakukan normalisasi data. Data kemudian dibagi menjadi data *training* dan data *testing* untuk proses *training* dan *testing* pada algoritme ELM. Proses *training* akan menghasilkan nilai *output weight* (β) yang akan digunakan pada proses *testing*. Pada proses *testing* menghasilkan nilai peramalan curah hujan (\hat{y}). Hasil dari proses *testing* selanjutnya dilakukan denormalisasi mengembalikan data kedalam bentuk sebelum normalisasi. Data hasil peramalan kemudian dilakukan proses perhitungan *Mean Absolute Percentage Error* (MAPE) untuk mengetahui kemampuan algoritme ELM dalam peramalan curah hujan melalui nilai selisih *error* dari data hasil peramalan dan data aktual. Diagram metode ELM dalam penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Peramalan Curah Hujan dengan Metode ELM

3.4 Pengumpulan Data

Kebutuhan data diperlukan untuk mendukung objek penelitian dalam melakukan peramalan curah hujan. Data curah hujan yang digunakan pada penelitian ini merupakan data sekunder berdasarkan pengamatan dari Badan Meteorologi Klimatologi dan Geofisika (BMKG) stasiun klimatologi Karangploso Malang. Data yang akan digunakan adalah data curah hujan dasarian (kumpulan data selama 10 hari) dengan menggunakan data curah hujan di wilayah Poncokusumo dengan rentang tahun 2002– 2015.

Pada penelitian ini menggunakan data curah hujan berdasarkan data runut waktu. Berikut adalah spesifikasi data yang digunakan pada penelitian ini:

1. Data curah hujan berasal dari daerah Poncokusumo dari wilayah di Malang Jawa Timur.
2. Rentang waktu data yang digunakan selama 14 tahun, mulai dari tahun 2002 sampai 2015.
3. Data curah hujan dengan waktu perdasarian selama 10 hari.
4. Data curah hujan menggunakan satuan millimeter per dasarian.

3.5 Peralatan Pendukung

Pada bagian peralatan pendukung adalah kebutuhan perangkat keras dan lunak sebagai pendukung dalam melakukan penelitian. Perangkat keras yang digunakan adalah laptop prosesor Intel[®] Celeron[™], CPU @ 1.80 Ghz dan memory 2 GB. Perangkat lunak yang digunakan adalah sistem operasi Windows 8.1, Netbeans 7.3.1, Java Development Kit (JDK) versi 1.7.0 untuk menyusun *source code* dan pengujian, Microsoft Excel sebagai penyimpanan data curah hujan.

3.6 Implementasi Algoritme

Pada Implementasi sistem merupakan tahapan melakukan implementasi sesuai dengan metode yang ditetapkan. Pada penelitian ini Implementasi algoritme *Extreme Learning Machine* dalam peramalan curah hujan dengan menggunakan pemrograman Java. Dalam implementasi algoritme akan dilakukan evaluasi tingkat hasil peramalan dengan menghitung nilai *error*. Evaluasi implementasi algoritme menggunakan MAPE yang akan menghasilkan nilai *error* dari data hasil peramalan terhadap data aktual.

3.7 Pengujian

Pada tahap ini dilakukan pengujian sistem terhadap sistem yang akan dibuat agar dapat menunjukkan bahwa algoritme *Extreme Learning Machine* (ELM) bisa bekerja dengan baik atau maksimal dan memiliki tingkat akurasi yang optimal. Berikut adalah beberapa uji coba yang dilakukan untuk evaluasi sistem in.

1. Pengujian jumlah *neuron*.
2. Pengujian variasi jumlah data *training* dan data *testing*.
3. Pengujian variabel *lag time* (fitur).

3.8 Pengambilan Kesimpulan dan Saran

Kesimpulan dilakukan setelah semua tahapan implementasi, dan pengujian metode yang diterapkan telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis penelitian dengan metode *Extreme Learning Machine* (ELM) yang akan diterapkan. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan pada penelitian yang akan datang.

Universitas Brawijaya
Universitas Brawijaya

Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya

Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya

BAB 4 PERANCANGAN

Pada bab ini menjelaskan perancangan peramalan curah hujan dengan metode *Extreme Learning Machine* (ELM), diagram alir sistem, perhitungan manual, perancangan antarmuka dan skenario pengujian.

4.1 Peramalan

Pada bab ini menjelaskan tahapan dari sistem peramalan, tahapan peramalan yang dilakukan pada penelitian ini yaitu definisi masalah, pengumpulan data, analisis data, formulasi metode, dan arsitektur perancangan.

4.1.1 Definisi Masalah

Pada tahun 2014 Indonesia mengalami bencana alam sebanyak 257 bencana. Bencana alam tersebut meliputi banjir sebanyak 86 kasus, tanah longsor sebanyak 111 kasus, banjir disertai longsor sebanyak 7 kasus, puting beliung sebanyak 5 kasus dan letusan gunung berapi sebanyak 1 kasus. Melalui kasus-kasus bencana yang telah disebutkan sebanyak 90% termasuk jenis bencana geologi yang disebabkan oleh pola curah hujan yang tidak stabil (BNPN, 2014). Berdasarkan bencana tersebut, salah satu solusi yang dapat diberikan adalah dengan melakukan peramalan (*forecasting*) yang dapat dikategorikan dalam jangka panjang maupun jangka pendek sehingga dapat dilakukan antisipasi terhadap kasus akibat dari pola curah hujan yang tidak stabil.

4.1.2 Pengumpulan Data

Pengumpulan data diperoleh dari Badan Meteorologi Klimatologi dan Geofisika (BMKG) di Karang Ploso Malang. Data yang akan digunakan adalah data curah hujan dasarian (kumpulan data selama 10 hari) dengan menggunakan data curah hujan di wilayah Poncokusumo dengan rentang tahun 2002 – 2015. Data dasarian wilayah Poncokusumo pada tahun 2002-2015 ditunjukkan pada Tabel 4.1.

Pada penelitian ini menggunakan data curah hujan bardasarkan data runut waktu. Berikut adalah spesifikasi data yang digunakan pada penilitian ini.

1. Data curah hujan berasal dari daerah Poncokusumo dari wilayah di Malang, Jawa Timur.
2. Rentang waktu data yang digunakan selama 14 tahun, mulai dari tahun 2002 sampai 2015.
3. Data curah hujan dengan waktu perdasarian selama 10 hari, maka dalam waktu sebulan memiliki 3 dasarian.
4. Data curah hujan menggunakan satuan millimeter per dasarian.

4.1.3 Formulasi Metode

Permasalahan yang akan diselesaikan adalah peramalan curah hujan di wilayah Poncokusumo, Malang. Hasil peramalan kemudian dievaluasi tingkat nilai *error*-nya untuk mengetahui kualitas peramalan. Semakin kecil nilai *error* semakin baik pula kualitas hasil peramalan. Solusi untuk permasalahan peramalan curah hujan yaitu dengan membuat perhitungan atau komputasi menggunakan metode *Extreme Learning Machine* (ELM). Masukan untuk komputasi ini antara lain data curah hujan dan parameter perhitungan seperti jumlah *neuron* pada *hidden layer*, fungsi aktivasi, data *training*. Selanjutnya masuk pada proses perhitungan peramalan menggunakan metode ELM. Inisialisasi data *training* dan data *testing* harus dinormalisasi terlebih dahulu sebelum memulai perhitungan metode ELM, normalisasi yang digunakan adalah *MinMax Normalization*. Selanjutnya masuk pada proses evaluasi menghitung nilai *error* menggunakan metode *Mean Absolute Percentage Error* (*MAPE*). Keluaran dalam komputasi ini antara lain hasil peramalan dan nilai *Mean Absolute Percentage Error* (*MAPE*).

Adapun sampel data peramalan curah hujan yang digunakan pada penelitian ini adalah data curah hujan di Daerah Poncokusumo Malang pada dasarian ke-2 bulan Januari dari tahun 2002-2015 yang ditunjukkan pada Lampiran A. Selanjutnya menentukan data latih dan data uji. Pada studi kasus ini, digunakan 10 data untuk perhitungan ELM dimana 8 data untuk data *training* dan 2 data untuk data *testing*. Inisialisasi data *training* dan data *testing* curah hujan di wilayah Poncokusumo pada dasarian ke-2 bulan Januari tahun 2002-2015 dengan menggunakan 4 *lag time*. 4 variabel *lag time* terdiri dari x_1 , x_2 , x_3 dan x_4 yang berfungsi sebagai pelatihan data peramalan. Sementara itu nilai *T* sebagai data *target* (aktual) dalam proses peramalan yang akan dilakukan. Pola deret yang akan digunakan dalam proses peramalan ditunjukkan pada Tabel 4.1

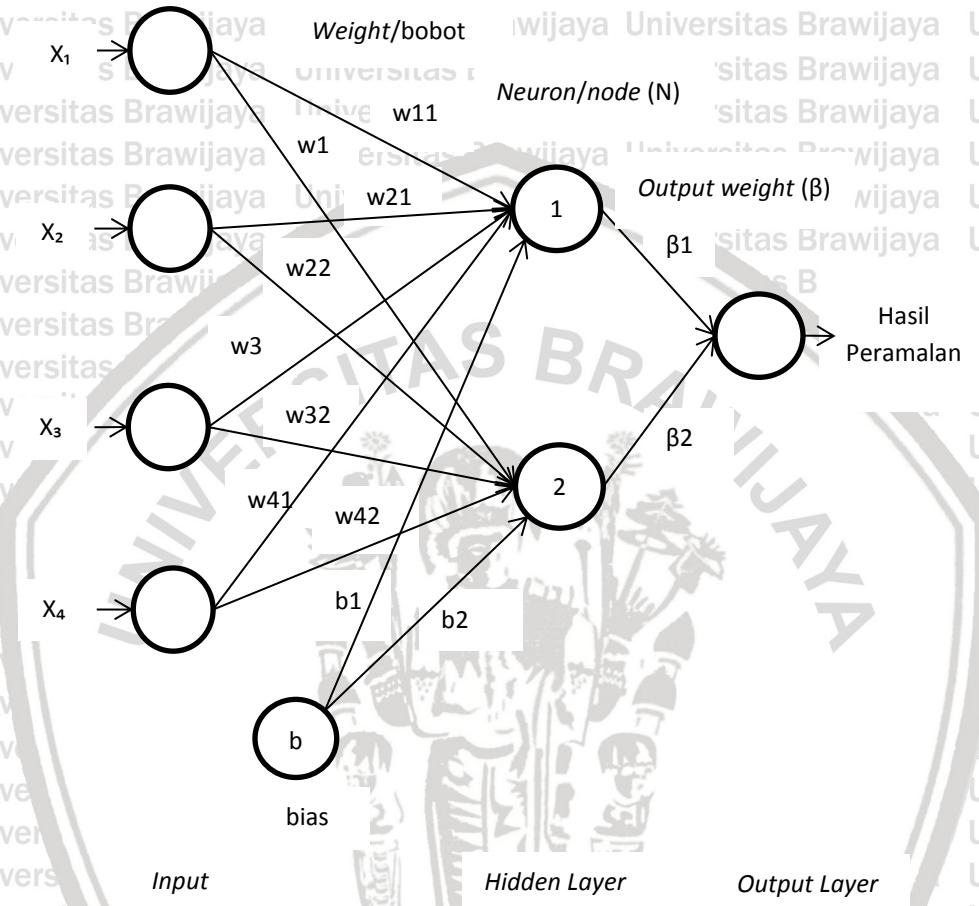
Table 4.1 Data Latih dan Data Uji

Data ke-	X ₁	X ₂	X ₃	X ₄	T	Jenis Data
1	101	51	232	88	77	Data Training
2	51	232	88	77	7	
3	232	88	77	7	48	
4	88	77	7	48	223	
5	77	7	48	223	80	
6	7	48	223	80	51	
7	48	223	80	51	112	
8	223	80	51	112	53	
9	80	51	112	53	77	
10	51	112	53	77	85	Data Testing



4.1.4 Arsitektur Perancangan Sistem

Arsitektur perancangan sistem merupakan gambaran secara umum proses pada sistem peramalan curah hujan menggunakan metode *Extreme Learning Machine*. Berikut gambaran arsitektur perancangan sistem yang ditunjukkan pada Gambar 4.1



Gambar 4.1 Arsitektur Perancangan Sistem ELM

Keterangan:

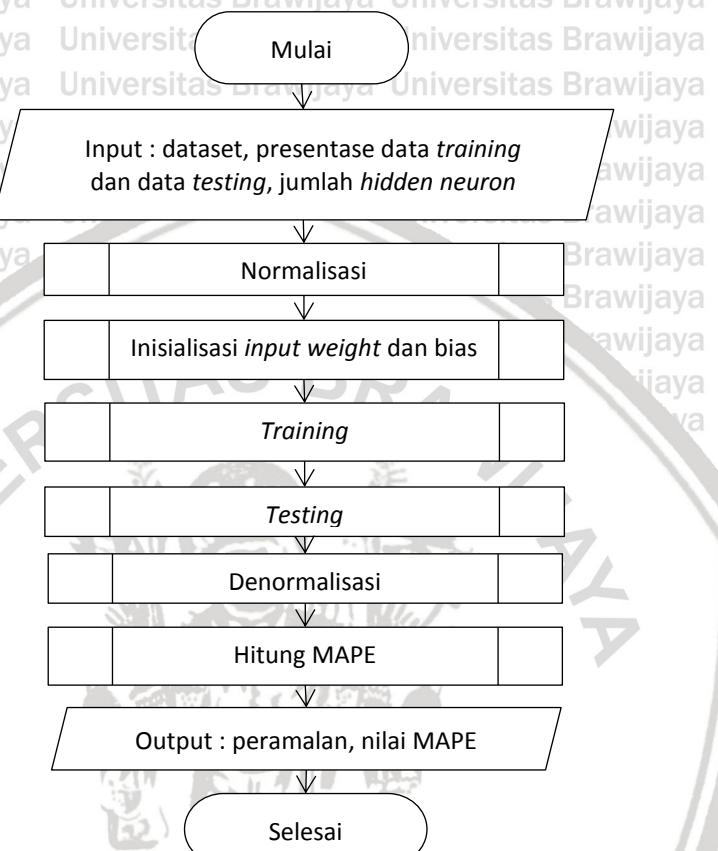
- x : Input
- w : Bobot input
- b : Bias
- β : Output weight

Banyaknya *neuron* pada *input layer* disesuaikan dengan masukkan pada sistem ini yang terdiri dari 4 input data yaitu 4 variabel *lag time* X_1, X_2, X_3 dan X_4 . Setiap *neuron* pada *input layer* terhubung dengan *hidden neuron* pada *hidden layer*. *Neuron* tersebut dihubungkan oleh bobot yang disebut *input weight* dengan nilai yang berbeda. Kemudian setiap *neuron* pada *hidden layer* terhubung dengan *output layer* yang dihubungkan oleh *output weight*. Semua *hidden neuron* terhubung pada satu *output*.



4.2 Diagram Alir Algoritme ELM

Diagram alir sistem merupakan gambaran dari alur proses sistem akan bekerja. Diagram alir sistem pada peramalan curah hujan ini meliputi *input* data, normalisasi, *training*, *testing*, denormalisasi dan evaluasi. Diagram alir sistem ditunjukkan pada Gambar 4.2.



Gambar 4.2 Diagram Alir Proses Peramalan Menggunakan ELM

Langkah-langkah proses peramalan curah hujan menggunakan metode ELM berdasarkan Gambar 4.2 sebagai berikut.

1. Sistem menerima masukan berupa data curah hujan, jumlah *neuron* pada *hidden layer*, fungsi aktivasi yang digunakan.
2. Inisialisasi *input weight* dan bias ditunjukkan pada Gambar 4.4.
3. Menghitung normalisasi data menggunakan Persamaan 2.4. Diagram alir untuk menghitung normalisasi data ditunjukkan pada Gambar 4.3.
4. Melakukan proses *training* menggunakan data yang telah dinormalisasi. Diagram alir untuk proses *training* ditunjukkan pada Gambar 4.5.
5. Melakukan proses *testing* peramalan. Diagram alir untuk proses *testing* ditunjukkan pada Gambar 4.11.
6. Menghitung nilai MAPE untuk mengevaluasi hasil peramalan proses *testing* menggunakan Persamaan 2.1. Diagram alir untuk proses menghitung nilai MAPE ditunjukkan pada Gambar 4.13.

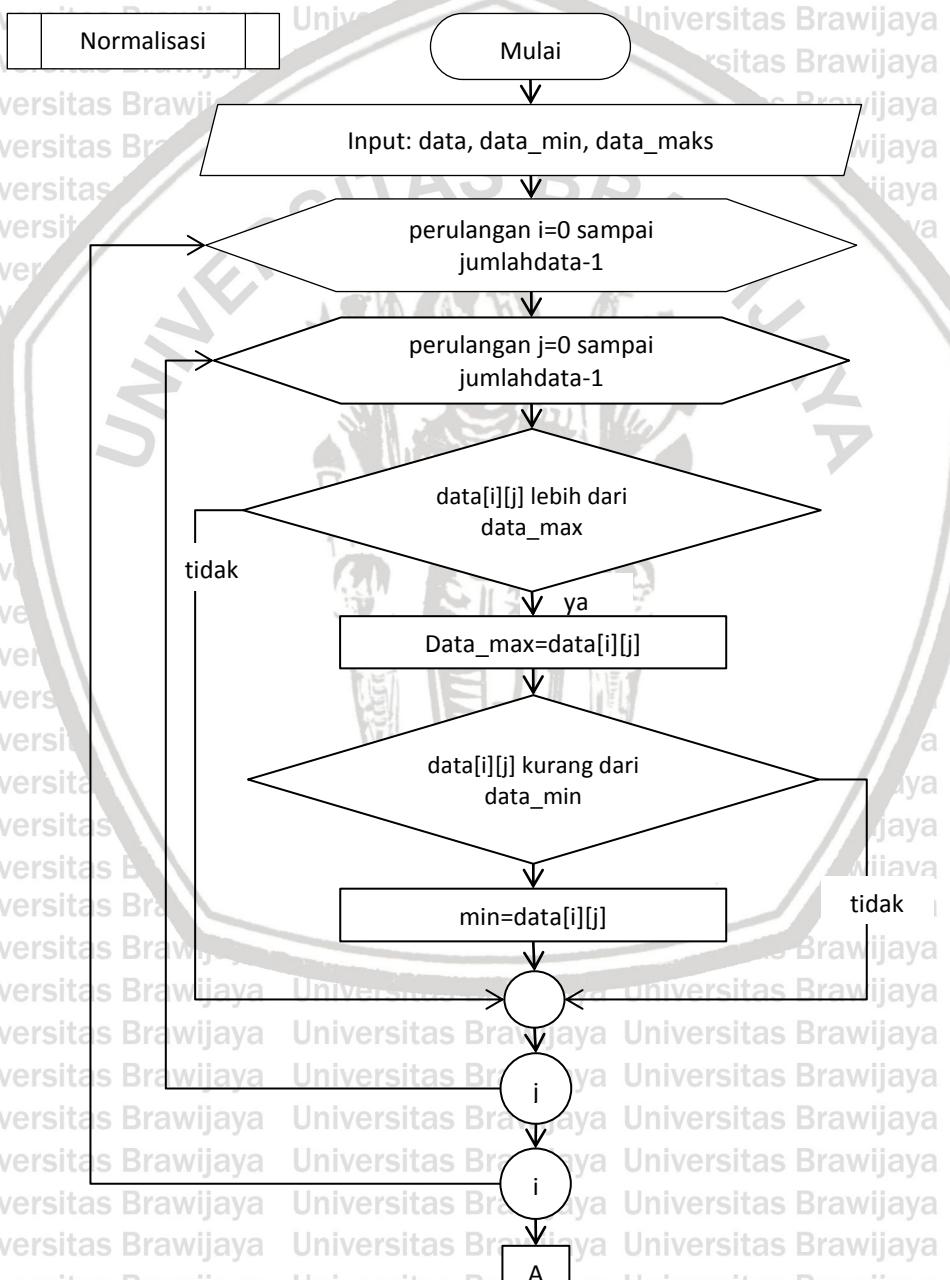


7. Menghitung denormalisasi untuk hasil akhir proses *testing* menggunakan Persamaan 2.5. Diagram alir untuk proses denormalisasi ditunjukkan pada Gambar 4.14

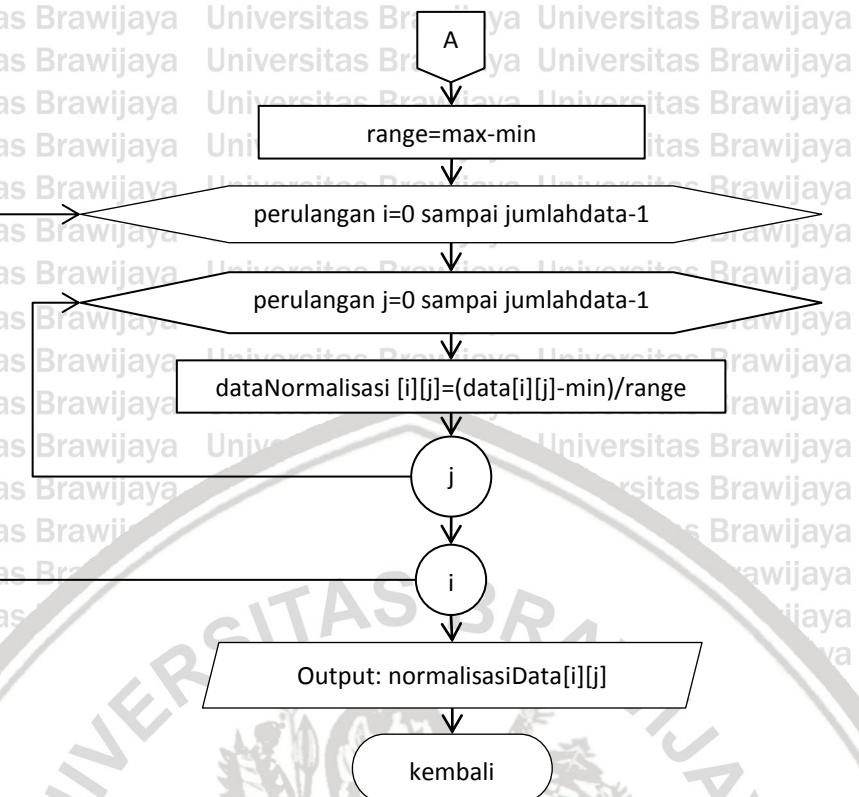
8. Keluaran sistem yaitu hasil peramalan setelah denormalisasi dan nilai MAPE.

4.2.2 Normalisasi

Normalisasi data merupakan tahapan transformasi data dimana nilai data skala yang sama pada rentang 0 sampai dengan 1. Proses normalisasi data menggunakan metode *Min-Max Normalization*. Diagram alir proses normalisasi data ditunjukkan pada Gambar 4.3.



Gambar 4.3 Diagram Alir Proses Normalisasi



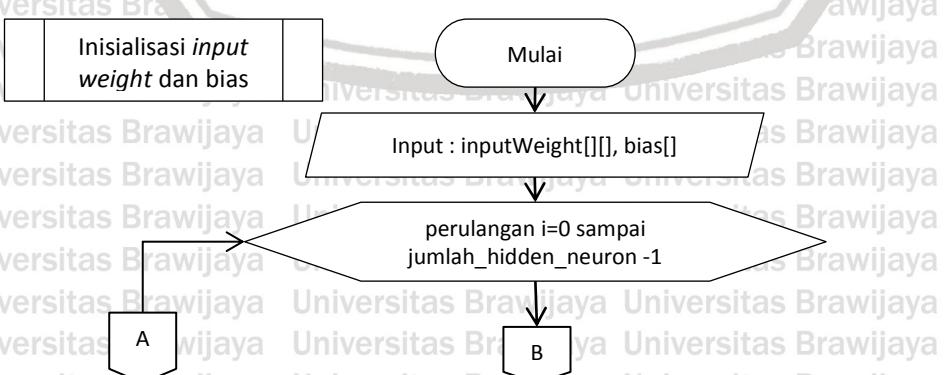
Gambar 4.3 Diagram Alir Proses Normalisasi (Lanjutan)

Langkah-langkah normalisasi data curah hujan menggunakan *metode Min-Max Normalization* berdasarkan Gambar 4.3 sebagai berikut:

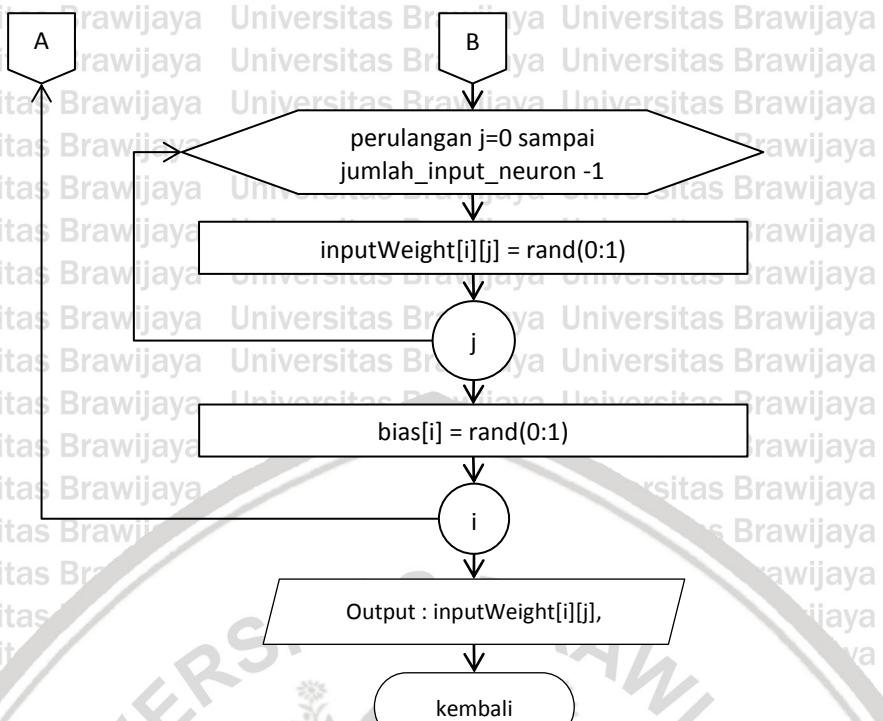
1. Sistem menerima masukan berupa data curah hujan.
2. Mencari nilai maksimal (*max*) dan minimal (*min*) dari seluruh data.
3. Menghitung nilai normalisasi menggunakan Persamaan 2.4 untuk setiap data.
4. Keluaran sistem adalah data curah hujan yang telah dinormalisasi.

4.2.3 Inisialisasi *Input Weight* dan Bias

Diagram alir tahapan inisialisasi *input weight* dan bias ditunjukkan pada Gambar 4.4.



Gambar 4.4 Diagram Alir Proses Inisialisasi *Input Weight* dan Bias



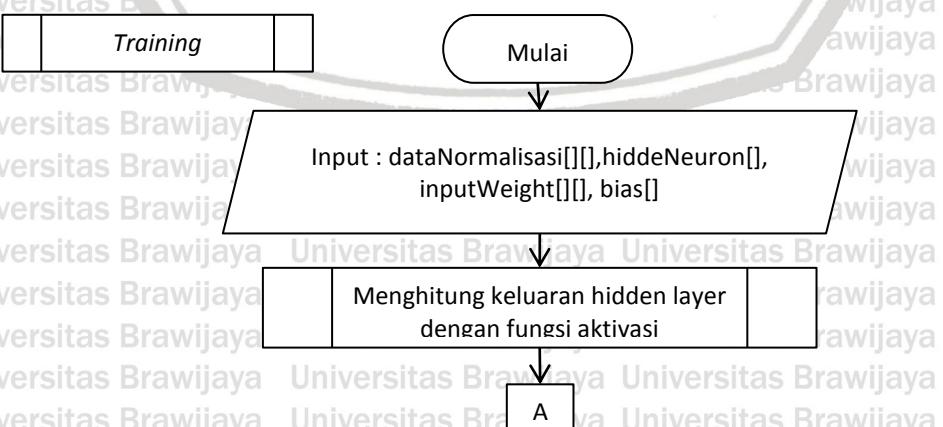
Gambar 4.4 Diagram Alir Proses Inisialisasi *Input Weight* dan *Bias* (Lanjutan)

Langkah-langkah proses inisialisasi *input weight* dan *bias* berdasarkan pada Gambar 4.4 sebagai berikut.

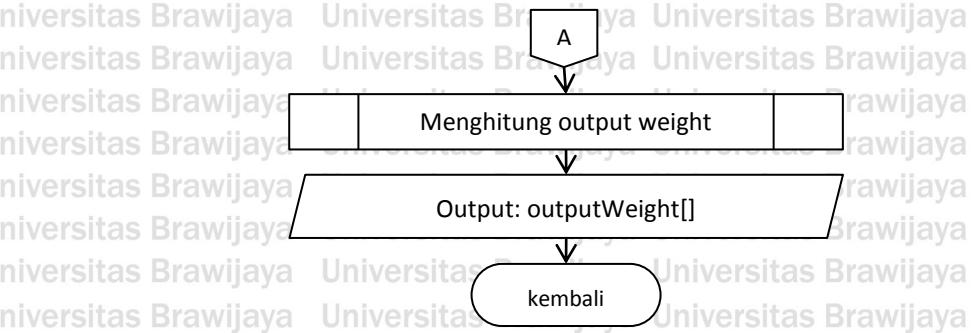
1. Melakukan perulangan sebanyak jumlah *hidden neuron* dan jumlah *input neuron*.
2. Menghitung *input weight* dan *bias* secara *random* dengan *range* antara 0 dengan 1.

4.2.4 Proses Training

Training dilakukan untuk memperoleh *input weight*, *bias*, dan *output weight* dengan tingkat kesalahan yang rendah untuk digunakan pada proses *testing*. Diagram alir proses *training* dengan metode ELM ditunjukkan pada Gambar 4.5.



Gambar 4.5 Diagram Alir Proses *Training*



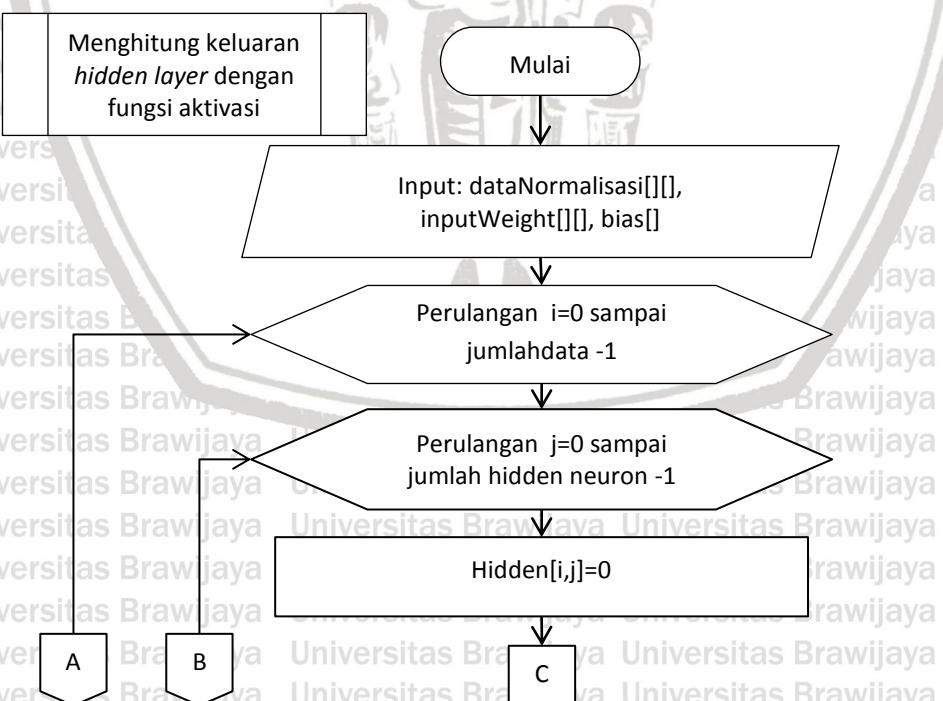
Gambar 4.5 Diagram Alir Proses Training (Lanjutan)

Langkah-langkah diagram alir proses *training* metode ELM berdasarkan Gambar 4.5 sebagai berikut.

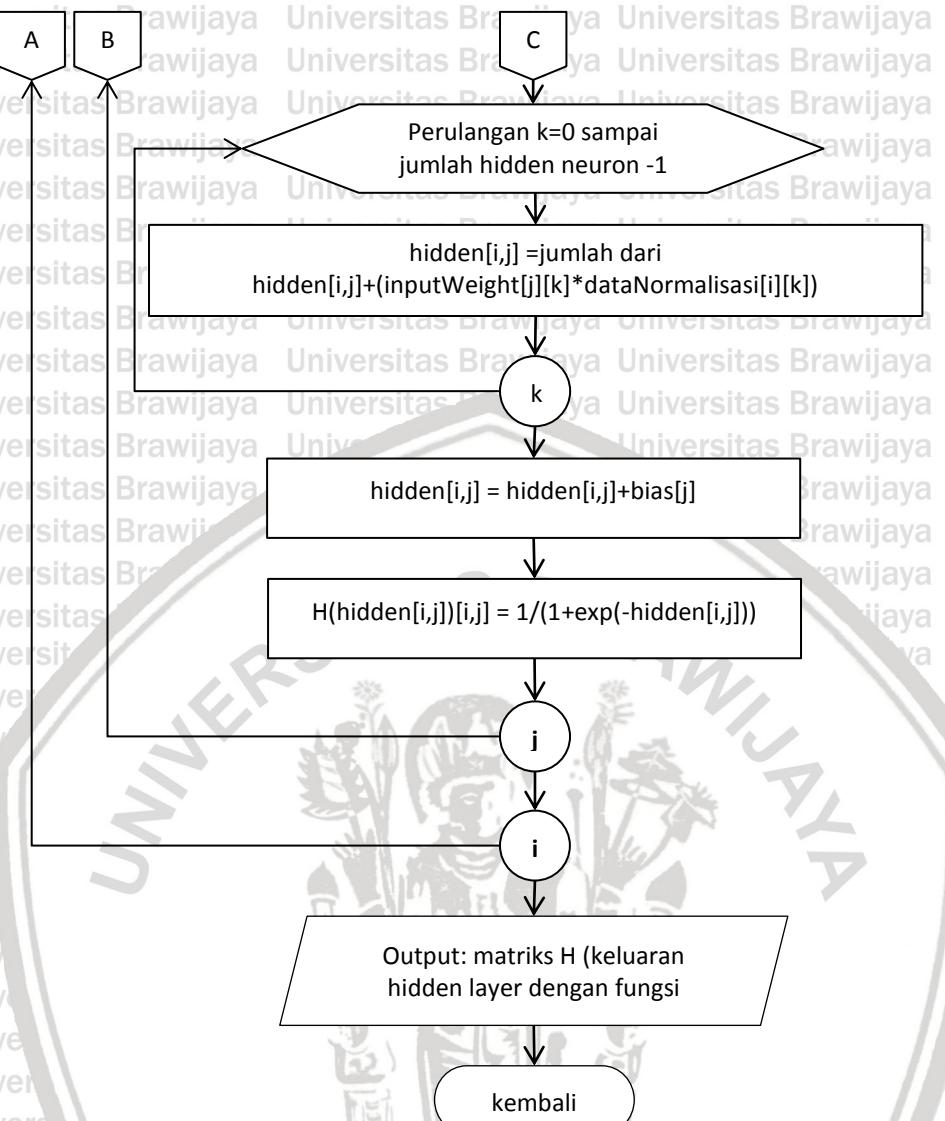
1. Sistem menerima masukan yaitu data *training*, jumlah *hidden neuron* dan fungsi aktivasi.
2. Menghitung keluaran *hidden layer* dengan fungsi aktivasi menggunakan Persamaan 2.6. Diagram alir untuk proses menghitung keluaran *hidden layer* dengan fungsi aktivasi ditunjukkan pada Gambar 4.6.
3. Menghitung *output weight* menggunakan Persamaan 2.9. Diagram alir untuk proses menghitung *output weight* ditunjukkan pada Gambar 4.7.
4. Keluaran sistem yaitu matriks *output weight*.

4.2.5 Menghitung Keluaran *Hidden Layer* dengan Fungsi Aktivasi

Diagram alir tahapan keluaran *hidden layer* dengan fungsi aktivasi ditunjukkan pada Gambar 4.6.



Gambar 4.6 Diagram Alir Proses Menghitung Keluaran *Hidden Layer* dengan Fungsi Aktivasi



Gambar 4.6 Diagram Alir Proses Menghitung Keluaran *Hidden Layer* dengan Fungsi Aktivasi (Lanjutan)

Langkah-langkah diagram alir proses menghitung keluaran *hidden layer* dengan fungsi aktivasi berdasarkan Gambar 4.6 sebagai berikut.

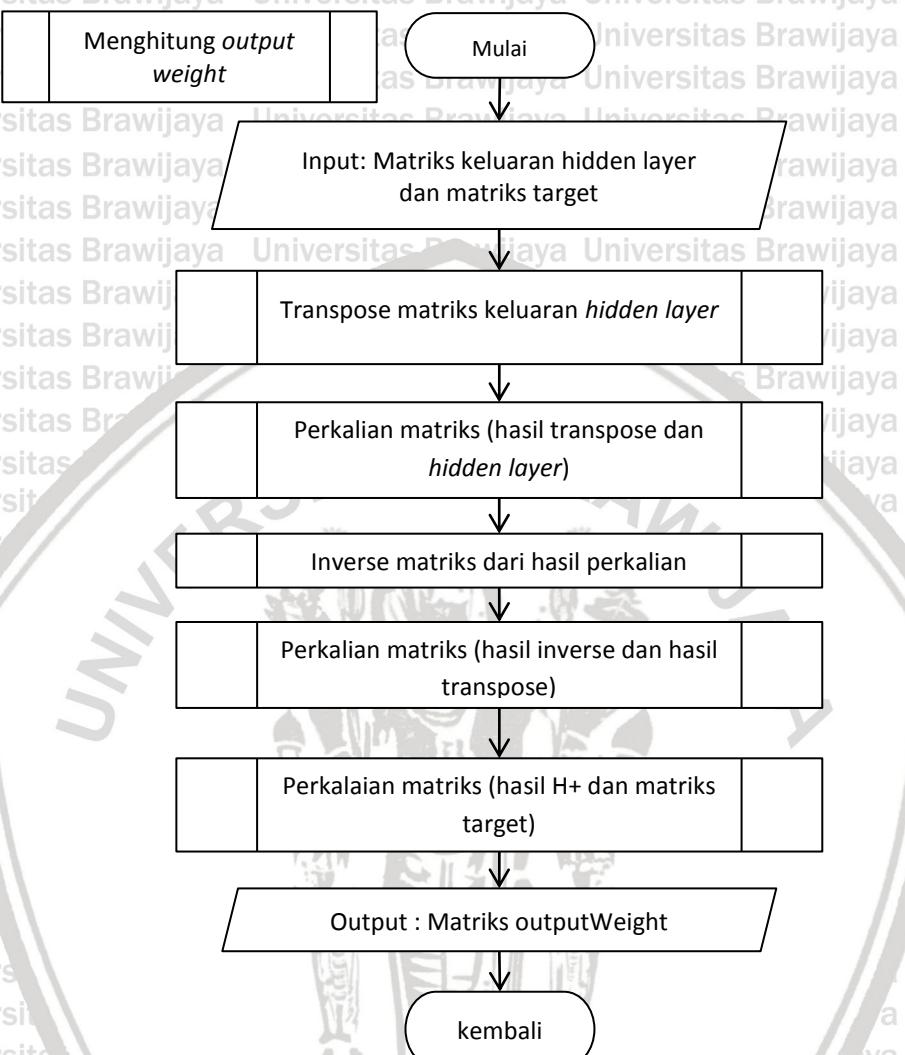
1. Sistem menerima masukan yaitu data yang telah dinormalisasi apabila pada proses *training* menggunakan data *training*, apabila pada proses *testing* menggunakan data *testing*. Sehingga data menyesuaikan pada saat proses ini dipanggil.
2. Menghitung keluaran *hidden layer* menggunakan Persamaan 2.6.
3. Menghitung keluaran *hidden layer* dengan fungsi aktivasi menggunakan Persamaan 2.7.
4. Keluaran sistem yaitu matriks keluaran *hidden layer* dengan fungsi aktivasi.



4.2.6 Menghitung Output Weight

Diagram alir tahapan menghitung *output weight* ditunjukkan pada Gambar 4.7.

4.7.



Gambar 4.7 Diagram Alir Menghitung *Output Weight*

Langkah-langkah diagram alir proses menghitung *output weight* berdasarkan

Gambar 4.7 sebagai berikut.

1. Sistem menerima masukan yaitu matriks keluaran *hidden layer* dan matriks *target*.

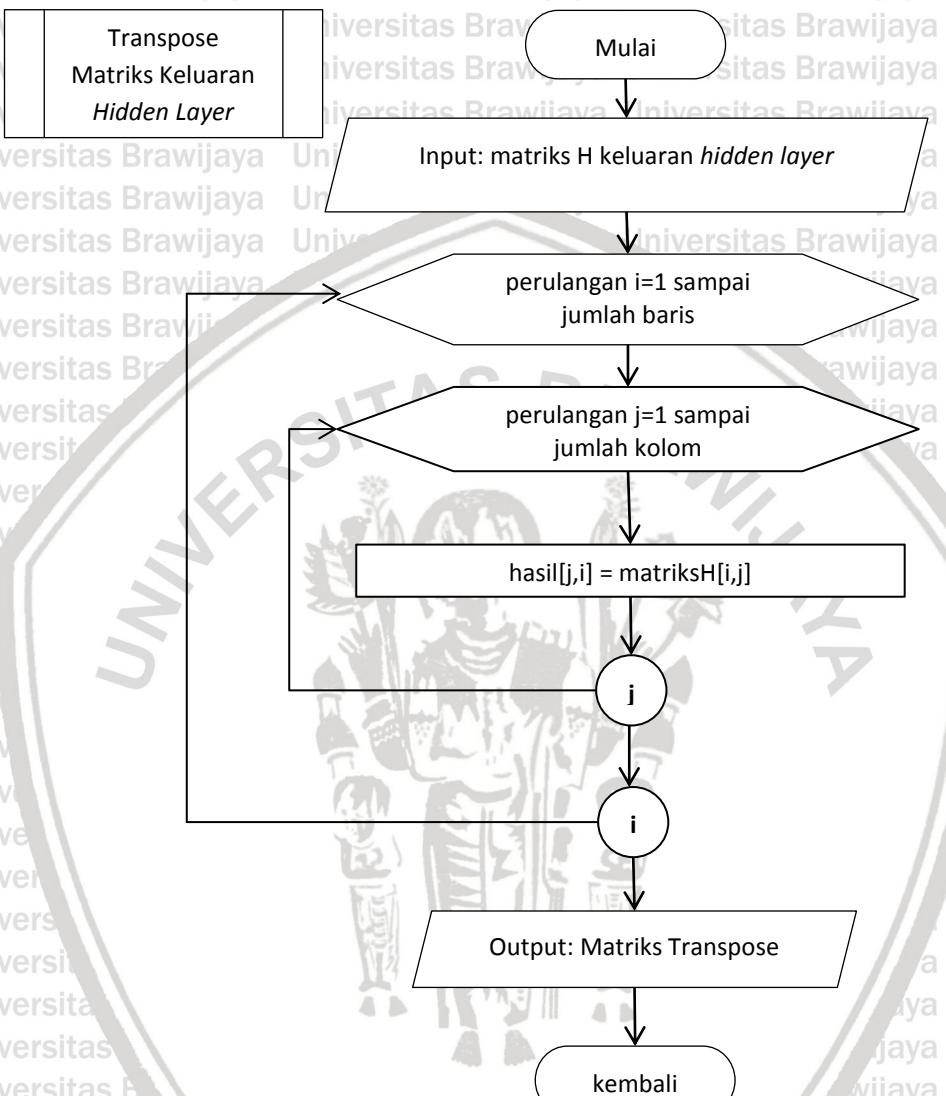
2. Menghitung matriks *Moore-Penrose Generalized Inverse* dari matriks keluaran *hidden layer* dengan melakukan *transpose* pada matriks keluaran *hidden layer*, kemudian melakukan perhitungan perkalian matriks hasil *transpose* dengan matriks keluaran *hidden layer*. Hasil dari perkalian matriks tersebut dilakukan *inverse*, kemudian melakukan perkalian matriks hasil *inverse* dengan matriks hasil *transpose* menjadi matriks *H+*. Terakhir



melakukan perkalian matriks H^+ dengan matriks *target* untuk menghasilkan matriks *output weight*.

3. Menghitung *output weight* menggunakan Persamaan 2.9.

4. Keluaran sistem yaitu matriks *output weight*.



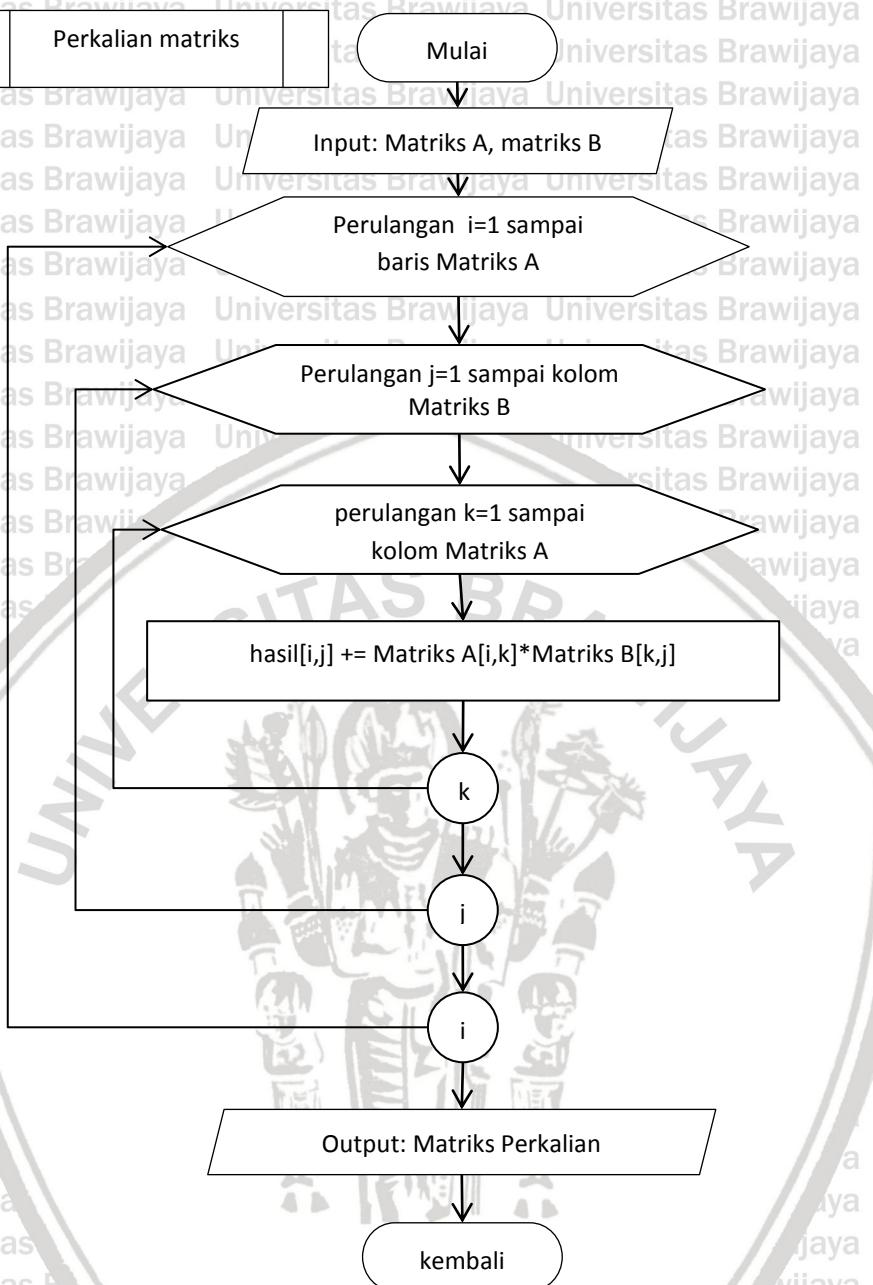
Gambar 4.8 Digram Alir Transpose Matriks Keluaran *Hidden layer*

Langkah-langkah diagram alir proses *transpose* matriks keluaran *hidden layer* berdasarkan Gambar 4.8 sebagai berikut.

1. Sistem menerima masukan yaitu matriks keluaran *hidden layer*.

2. Proses *transpose* matriks mengubah ordo matriks yang [baris,kolom] menjadi [kolom,baris].

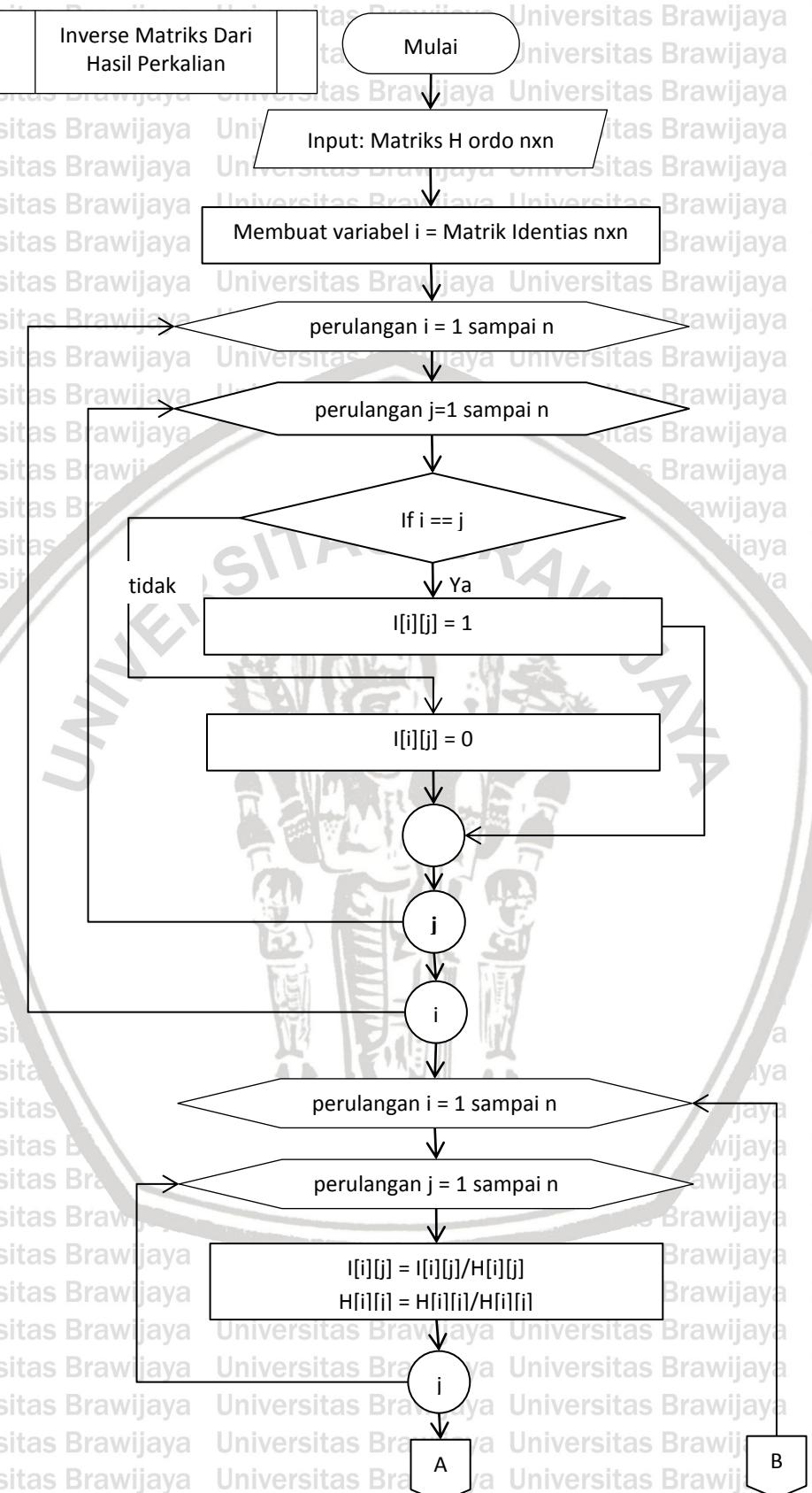
3. Keluaran sistem yaitu matriks hasil *transpose*.



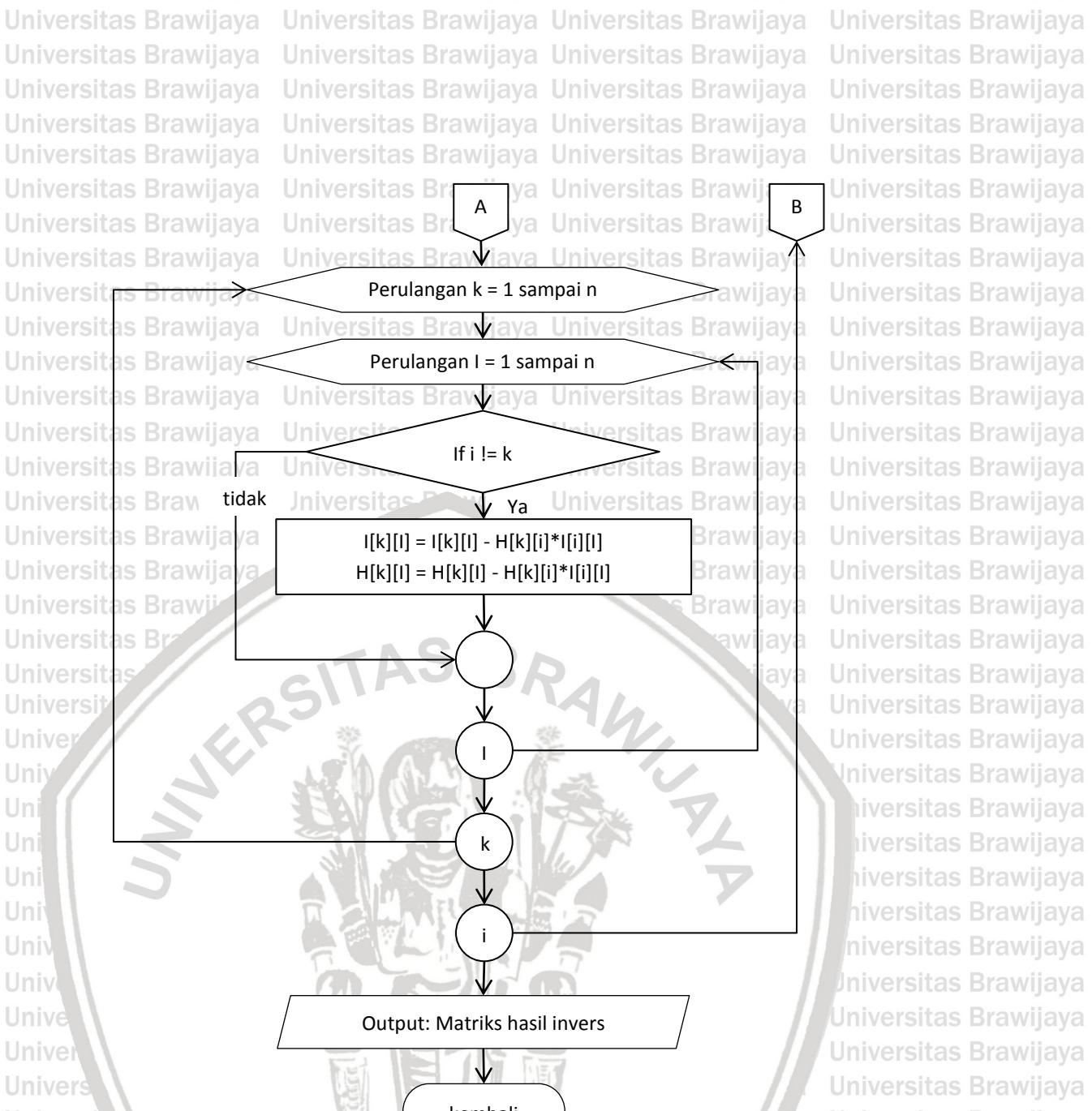
Gambar 4.9 Diagram Alir Perkalian Matriks

Langkah-langkah diagram alir proses perkalian matriks berdasarkan gambar 4.9 sebagai berikut.

1. Sistem menerima masukan yaitu 2 matriks yang akan dikalikan yaitu matriks A dan matriks B.
2. Proses perkalian matriks dilakukan dengan cara mengalikan elemen kolom dari matriks A dengan elemen baris dari matriks B.
3. Keluaran sistem yaitu matriks hasil perkalian.



Gambar 4.10 Diagram Alir Inverse Matriks Dari Hasil Perkalian



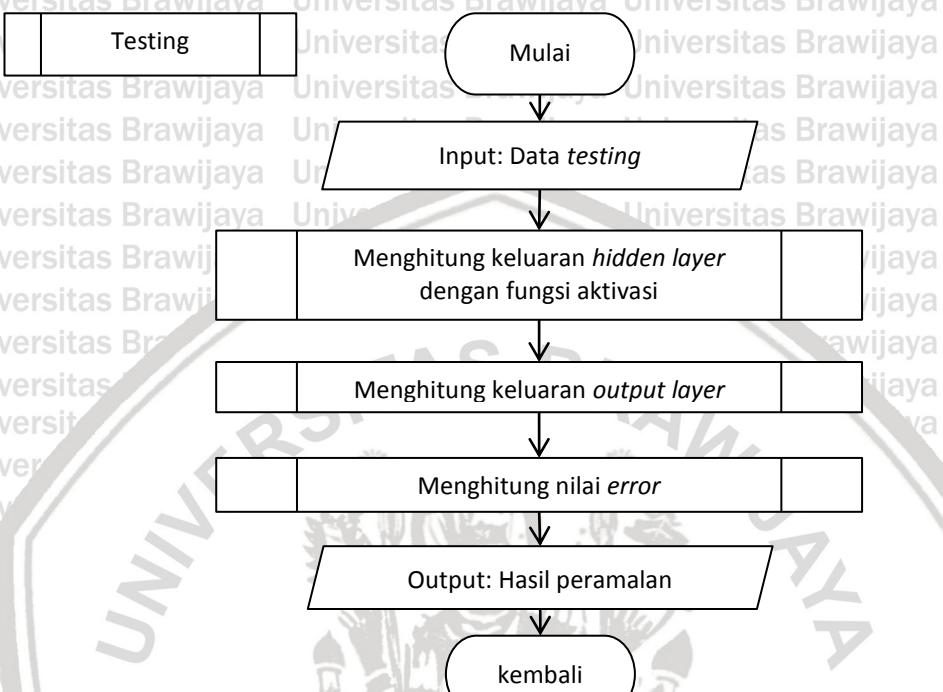
Gambar 4.10 Diagram Alir Inverse Matriks Dari Hasil Perkalian (Lanjutan)

Langkah-langkah diagram alir proses *inverse* matriks dari hasil perkalian berdasarkan Gambar 4.10 sebagai berikut.

1. Sistem menerima masukan yaitu matriks hasil perkalian dari matriks transpose keluaran *hidden layer* dengan matriks keluaran *hidden layer*.
2. Inisialisasi matriks identitas dengan ordo sama dengan matriks hasil perkalian. Pada perhitungan *inverse*, matriks identitas digunakan dalam Operasi Baris Elementer (OBE).
3. Proses *inverse* matriks dilakukan dengan cara mengubah matriks hasil perkalian (H) menjadi matriks identitas, sehingga matriks identitas (I) menjadi matriks baru.
4. Keluaran sistem yaitu matriks identitas yang menjadi matriks baru inilah yang merupakan *inverse* dari matriks hasil perkalian.

4.2.7 Proses Testing

Proses *testing* dilakukan berdasarkan *input weight*, bias dan *output weight* yang sesuai dari perhitungan *training*. Proses *testing* dilakukan sesuai dengan urutan proses *training*. Diagram alir proses *testing* dengan metode ELM ditunjukkan pada Gambar 4.11.



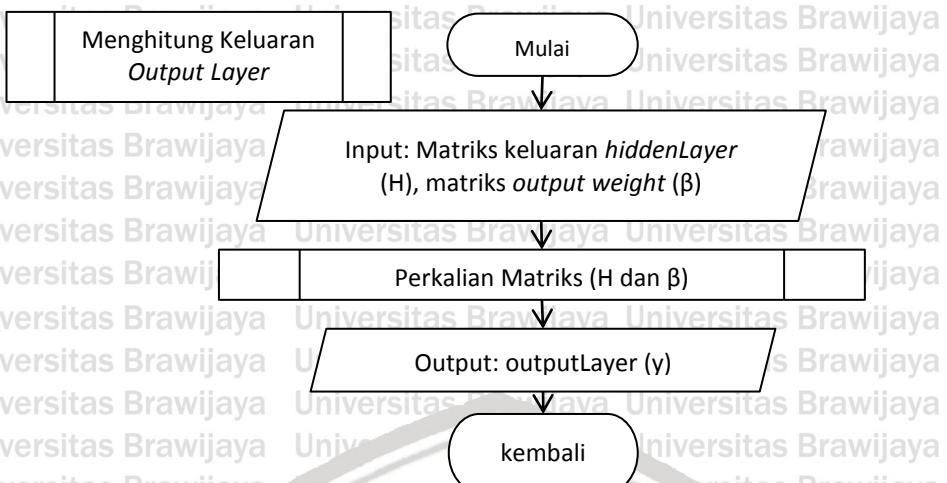
Gambar 4.11 Proses Testing

Langkah-langkah diagram alir proses *testing* metode ELM berdasarkan Gambar 4.11 sebagai berikut.

1. Sistem menerima masukan yaitu data *testing*. Menggunakan nilai *input weight*, bias, serta *output weight* dari proses *training*.
2. Menghitung keluaran *hidden layer* dengan fungsi aktivasi menggunakan Persamaan 2.7. Diagram alir untuk proses menghitung keluaran *hidden layer* dengan fungsi aktivasi ditunjukkan pada Gambar 4.6.
3. Menghitung keluaran *output layer* menggunakan Persamaan 2.11. Diagram alir untuk proses menghitung keluaran *output layer* ditunjukkan pada Gambar 4.12.
4. Menghitung nilai *error* menggunakan Persamaan 2.1. Diagram alir proses menghitung nilai *error* ditunjukkan pada Gambar 4.13.
5. Keluaran sistem yaitu hasil peramalan dan nilai *error output layer* dengan *target*.

4.2.8 Menghitung Keluaran *Output Layer*

Diagram alir tahapan menghitung keluaran *output layer* ditunjukkan pada Gambar 4.12.



Gambar 4.12 Diagram Alir Menghitung Keluaran *Output Layer*

Langkah-langkah diagram alir proses menghitung keluaran di *output layer*

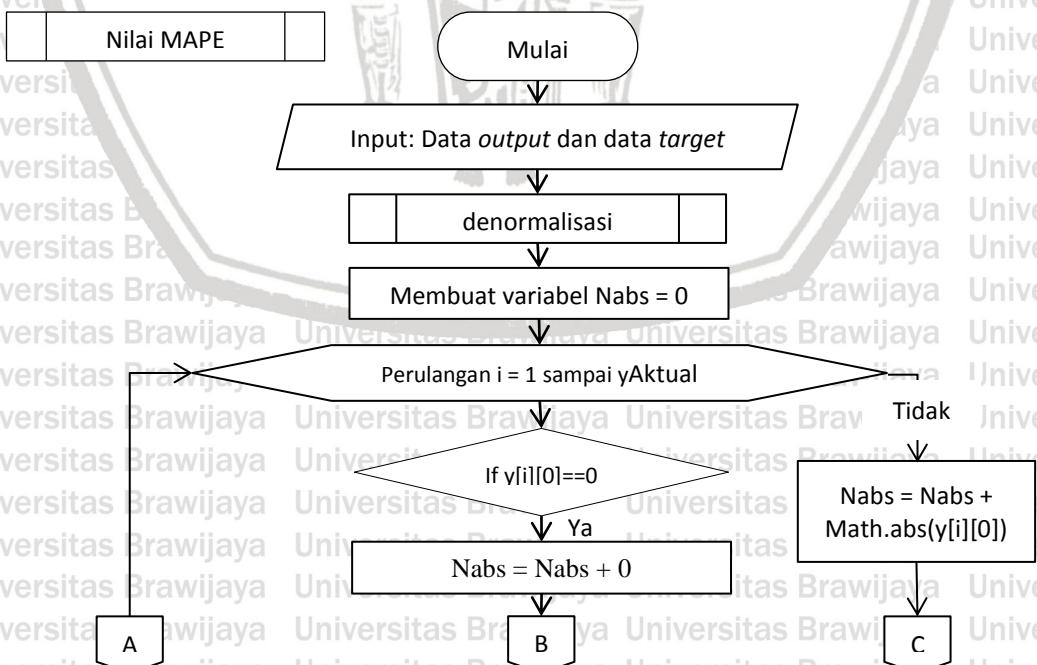
berdasarkan Gambar 4.12 sebagai berikut.

1. Sistem menerima masukan yaitu matriks keluaran di *hidden layer* dan matriks *output weight* dari proses *training*.
2. Menghitung keluaran di *output layer* dengan menggunakan Persamaan 2.11, yaitu dengan perkalian matriks keluaran *hidden layer* dengan matriks *output weight* dari proses *training*.
3. Keluaran sistem yaitu keluaran di *output layer* yang merupakan *output* hasil proses *testing*.

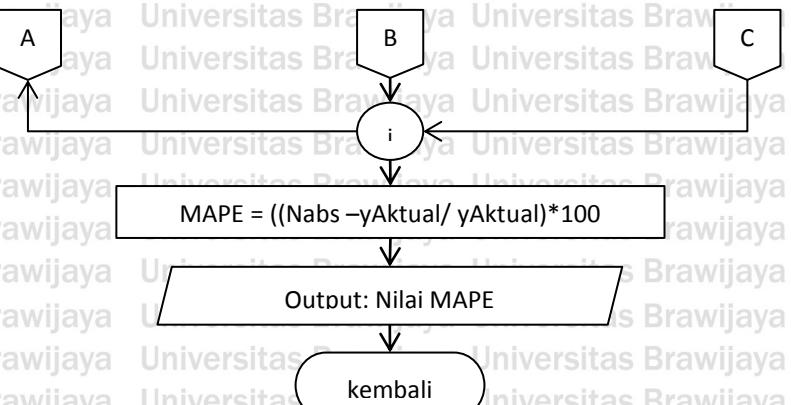
4.2.9 Proses Hitung Nilai Mean Absolute Percentage Error (MAPE)

Menghitung nilai *error rate* menggunakan *Mean Absolute Percentage Error* (MAPE)

bertujuan untuk mengevaluasi metode peramalan yang digunakan yaitu ELM. Diagram alir proses menghitung *error rate* menggunakan MAPE ditunjukkan pada Gambar 4.13.



Gambar 4.13 Diagram Alir Proses Hitung Tingkat Error Menggunakan MAPE



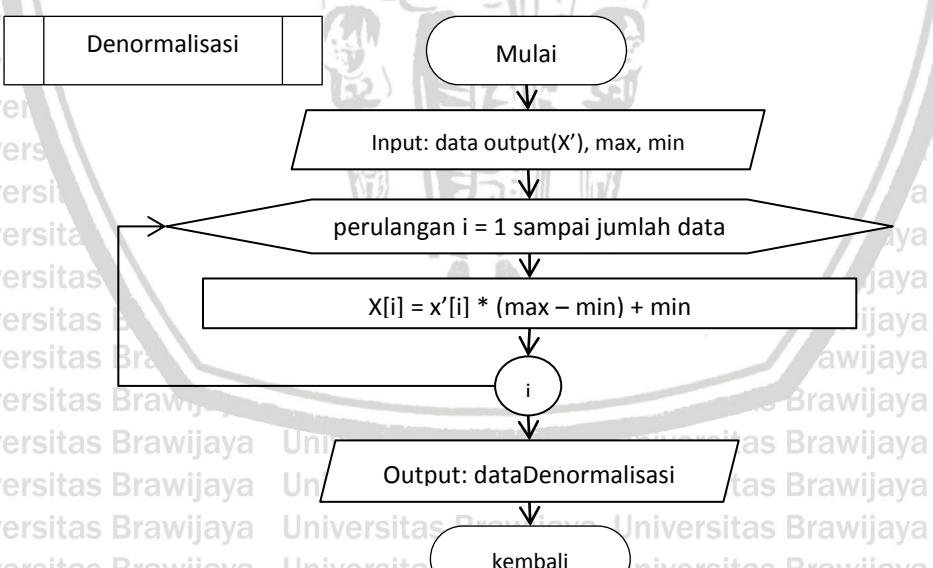
Gambar 4.13 Diagram Alir Proses Hitung Tingkat Error Menggunakan MAPE (Lanjutan)

Langkah-langkah proses menghitung tingkat *error* menggunakan MAPE berdasarkan Gambar 4.13 sebagai berikut.

1. Sistem menerima masukan berupa data *output*, data *target* dan jumlah data.
2. Menghitung hasil MAPE akhir dengan menggunakan Persamaan 2.1.
3. Keluaran sistem berupa nilai MAPE yang menunjukkan tingkat *error* dari hasil perhitungan ELM.

4.2.10 Proses Denormalisasi

Denormalisasi data digunakan untuk mendapatkan nilai sebenarnya dari *output* berdasarkan hasil *output* peramalan. Diagram alir poses denormalisasi data ditunjukkan pada Gambar 4.14.



Gambar 4.14 Diagram Alir Proses Denormalisasi

Langkah-langkah diagram alir proses denormalisasi data berdasarkan Gambar 4.14 sebagai berikut.

1. Sistem menerima masukan yaitu matriks keluaran di *output layer*, nilai maksimal, dan nilai minimal.

2. Menghitung nilai denormalisasi dengan menggunakan Persamaan 2.5 untuk setiap data.

3. Keluaran sistem yaitu data yang telah di denormalisasi.

4.3 Perhitungan Manual

Pada penelitian ini menggunakan sampel data sebanyak 10 record dengan perbandingan data *training* dan data *testing* yaitu 80:20 sehingga didapat 8 data *training* dan 2 data *testing*. Pada metode ELM contoh perhitungan manual menggunakan 2 *hidden nodes*. Data yang digunakan ditunjukkan pada Tabel 4.2.

Table 4.2 Data Training dan Data Testing

Data ke-	X ₁	X ₂	X ₃	X ₄	T	Jenis Data
1	101	51	232	88	77	Data training
2	51	232	88	77	7	
3	232	88	77	7	48	
4	88	77	7	48	223	
5	77	7	48	223	80	
6	7	48	223	80	51	
7	48	223	80	51	112	
8	223	80	51	112	53	
9	80	51	112	53	77	Data testing
10	51	112	53	77	85	

Pada contoh perhitungan manual ini menggunakan fungsi aktivasi *sigmoid biner* dan jumlah *neuron* pada *hidden layer* sebanyak 2. Selanjutnya akan dibahas perhitungan manual menggunakan metode ELM.

4.3.1 Perhitungan Normalisasi Data

Langkah-langkah normalisasi data menggunakan *Min-Max Normalization* dihitung berdasarkan Persamaan 2.5 adalah sebagai berikut.

Langkah 1: menghitung nilai maksimal dan minimal dari semua data. Nilai maksimal dan minimal ditunjukkan pada Tabel 4.3.

Table 4.3 Nilai Maksimal dan Minimal

Max	Min
232	7

Langkah 2: menghitung nilai normalisasi data menggunakan *Min-Max Normalization*. Berikut contoh perhitungan nilai normalisasi:

$$X'_{1,1} = \frac{(x_{1,1} - X_{min})}{(X_{max} - X_{min})} = \frac{(101 - 7)}{(232 - 7)} = 0,41777778$$

Hasil perhitungan normalisasi secara keseluruhan ditunjukkan pada Tabel 4.4.

Data ke-	X_1	X_2	X_3	X_4	T
1	0,417777778	0,195555556	1	0,36	0,311111111
2	0,195555556	1	0,36	0,311111111	0
3	1	0,36	0,311111111	0	0,182222222
4	0,36	0,311111111	0	0,182222222	0,96
5	0,311111111	0	0,182222222	0,96	0,324444444
6	0	0,182222222	0,96	0,324444444	0,195555556
7	0,182222222	0,96	0,324444444	0,195555556	0,466666667
8	0,96	0,324444444	0,195555556	0,466666667	0,204444444
9	0,324444444	0,195555556	0,466666667	0,204444444	0,311111111
10	0,195555556	0,466666667	0,204444444	0,311111111	0,346666667

4.3.2 Inisialisasi Input Weight dan Bias

Jumlah *neuron* pada *hidden layer* yang digunakan dalam perhitungan manual ini sebanyak 2. Menentukan *input weight* dan bias didapatkan secara *random* dalam bentuk matriks dengan ordo sesuai dengan banyaknya *input neuron* dan *hidden neuron*. Dalam perhitungan manual ini, jumlah *input neuron* sebanyak 4 sehingga *input weight* memiliki ordo 2x4. Nilai *input weight* dan bias memiliki *range* [0,1]. Matriks *input weight* ditunjukkan pada Tabel 4.5.

Table 4.5 Matriks Nilai *Input Weight*

W	1	2	3	4
1	0,07	0,93	0,08	0,24
2	0,95	0,99	0,42	0,51

Sedangkan jumlah bias sama dengan jumlah *neuron* pada *hidden layer* yaitu sebanyak 2. Matriks bias ditunjukkan pada Tabel 4.6.

Table 4.6 Matriks Nilai Bias

b	
1	2
0,1	0,1

4.3.3 Perhitungan Proses Training

Pada proses *training* data yang digunakan sebanyak 8, jumlah *neuron* pada *hidden layer* yaitu 2, dan menggunakan fungsi aktivasi *sigmoid*. Langkah-langkah perhitungan manual proses *training* metode ELM adalah sebagai berikut.

Langkah 1: Menghitung keluaran di *hidden layer* dengan fungsi aktivasi. Perhitungan keluaran *hidden layer* ditunjukkan pada Persamaan 2.7. Kemudian setelah dilakukan perhitungan dilanjutkan dengan menghitung fungsi aktivasi. Fungsi aktivasi yang digunakan adalah fungsi aktivasi *sigmoid*. Berikut ini contoh perhitungan keluaran *hidden layer*:

$$H_{init\ ij} = \left(\sum_{i=1}^n w_i x_j \right) + b_j$$

$$H_{init\ 1,1} = ((0,07 * 0,41777778) + (0,93 * 0,19555556) + (0,08 * 0,1) + (0,24 * 0,36)) + 0,1 = 0,477511111$$

Hasil dari keluaran *hidden layer* secara keseluruhan ditunjukkan pada Tabel 4.7.

Table 4.7 Matriks Keluaran *Hidden Layer*

$H_{init\ ij}$	1	2
1	0,477511111	1,294088889
2	1,147155556	1,585644444
3	0,529688889	1,537066667
4	0,458266667	0,842933333
5	0,366755556	0,961688889
6	0,424133333	0,849066667
7	1,078444444	1,459511111
8	0,596577778	1,653333333

Berikut ini contoh perhitungan keluaran *hidden layer* dengan fungsi aktivasi.

$$H(x)_{1,1} = \frac{1}{1 + e^{-x}} = \frac{1}{1 + e^{-0,477511111}} = 0,617159988$$

Hasil keluaran *hidden layer* dengan fungsi aktivasi secara keseluruhan ditunjukkan pada Tabel 4.8 sebagai berikut:

Table 4.8 Matriks Keluaran *Hidden Layer* Dengan Fungsi Aktivasi

$H(x)$	1	2
1	0,617159988	0,784838472
2	0,758990984	0,830002424
3	0,629410547	0,8230379
4	0,612602901	0,69908265
5	0,590674774	0,723459821
6	0,604471896	0,700371318
7	0,746199496	0,811457889
8	0,644872966	0,839341051

Langkah 2: Menghitung *output weight* dari *hidden layer* ke *output layer*. Untuk menghitung *output weight*, pertama harus menghitung matriks *Moore-Penrose Generalized Inverse* dari hasil keluaran *hidden layer* dengan fungsi aktivasi. Pada perhitungan *inverse* matriks hasil perkalian menggunakan perhitungan Operasi Baris Elementari (OBE). Setelah itu baru dihitung *output weight* seperti yang ditunjukkan pada Persamaan 2.10.

Langkah 2.1: Menghitung matriks *Moore-Penrose Generalized Inverse* dari hasil keluaran *hidden layer* dengan fungsi aktivasi yaitu $H^+ = (H(x)^T H(x))^{-1} H(x)^T$. Pertama mencari matriks *transpose* dari matriks H . Hasil *transpose* ditunjukkan pada Tabel 4.9

Table 4.9 Transpose Matriks Keluaran Hidden Layer dengan Fungsi Aktivasi

$(H(x))^T$	1	2	3	...	7	8
1	0,617159	0,758990	0,62941	...	0,746199	0,644872
2	0,784838	0,830002	0,8230379	...	0,81145	0,839341

Kemudian menghitung perkalian matriks dari hasil *transpose* dengan keluaran *hidden layer* dengan fungsi aktivasi. Berikut ini contoh perhitungan perkalian matriks.

$$(H(x)^T H(x))_{1,1} = (0,617159 * 0,617159) + (0,758990 * 0,758990) + (0,629410 * 0,629410) + (0,612602 * 0,612602) + (0,590674 * 0,590674) + (0,604471 * 0,604471) + (0,746199 * 0,746199) + (0,644872 * 0,644872) = 3,389333666$$

Hasil perkalian matriks secara keseluruhan ditunjukkan pada Tabel 4.10.

Table 4.10 Perkalian Matriks Hasil *Transpose* Dengan Keluaran Hidden Layer Dengan Fungsi Aktivasi

$(H(x)^T H(x))$	1	2
1	3,415351508	4,058086119
2	4,058086119	4,84785479

Setelah menghitung perkalian matriks dari hasil *transpose* dengan keluaran *hidden layer* dengan fungsi aktivasi kemudian menghitung *inverse* matriks hasil perhitungan tersebut. Berikut ini contoh perhitungan *inverse* matriks menggunakan perhitungan Operasi Baris Elementari (OBE).

1. Bentuk matriks identitas (I), Baris 1 ($R1$), dan Baris 2 ($R2$)

$$[H|I] = \begin{bmatrix} 3,415351508 & 4,058086119 \\ 4,058086119 & 4,84785479 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2. $R1 / 3,415351508 \rightarrow R1$

$$[H|I] = \begin{bmatrix} 1 & 1,188189886 \\ 4,058086119 & 4,84785479 \end{bmatrix} \begin{bmatrix} 0,292795631 & 0 \\ 0 & 1 \end{bmatrix}$$

3. $R2 - 4,058086119 * R1 \rightarrow R2$

$$[H|I] = \begin{bmatrix} 1 & 1,188189886 \\ 0 & 0,026077906 \end{bmatrix} \begin{bmatrix} 0,292795631 & 0 \\ -1,188189886 & 1 \end{bmatrix}$$

4. $R2 / 0,026077906 \rightarrow R2$

$$[H|I] = \begin{bmatrix} 1 & 1,188189886 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0,292795631 & 0 \\ -45,56308649 & 38,3466372 \end{bmatrix}$$

$$5. R1 - 1,188189886 * R2 \rightarrow R1$$

$$[H|I] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 54,43039418 & -45,56308649 \\ -45,56308649 & 38,3466372 \end{bmatrix}$$

Hasil *inverse* matriks secara keseluruhan ditunjukkan pada Tabel 4.11.

Table 4.11 Inverse Matriks

$(H(x)^T H(x))^{-1}$	1	2
1	54,43039418	-45,56308649
2	-45,56308649	38,3466372

Terakhir menghitung matriks *Moore-Penrose Generalized Inverse* (H^+) dengan cara perkalian matriks antara *inverse* matriks (hasil perkalian matriks *transpose* keluaran *hidden layer* dengan matriks keluaran *hidden layer*) dengan matriks *transpose* keluaran *hidden layer*. Berikut ini contoh perhitungan perkalian matriks antara *inverse* matriks dengan *transpose* keluaran *hidden layer*.

$$H^+ = (H(x)^T H(x))^{-1} H(x)^T$$

$$H^+_{1,1} = (54,43039418 * 0,617159) + (-45,56308649 * 0,784838) \\ = -2,167401763$$

Hasil matriks *Moore-Penrose Generalized Inverse* secara keseluruhan ditunjukkan pada Tabel 4.12 sebagai berikut.

Table 4.12 Matriks Moore-Penrose Generalized Inverse

H^+	1	2	3	4	5	6	7	8
1	-2,167	3,494	-3,241	1,491	-0,812	0,991	3,643	-3,142
2	1,976	-2,754	2,882	-1,104	0,829	-0,684	-2,882	2,803

Langkah 2.2: Menghitung *output weight* dengan cara perkalian matriks *Moore-Penrose Generalized Inverse* dengan matriks *target*. Berikut ini contoh perhitungan perkalian matriks *Moore-Penrose Generalized Inverse* dengan *target*.

$$\beta = H^+ T$$

$$\beta = (-2,167 * 0,311111) + (3,494 * 0) + (-3,241 * 0,182222) + (1,491 * 0,96) + (-0,812 * 0,324444) + (0,991 * 0,195555) + (3,643 * 0,466666) + (-3,142 * 0,204444) = 1,155246$$

Hasil akhir *output weight* secara keseluruhan ditunjukkan pada Tabel 4.13.

Table 4.13 Nilai Output Weight

β
1,155246
-0,55712

4.3.4 Perhitungan Proses Testing

Perhitungan proses *testing* data yang digunakan sebanyak 2, jumlah *neuron* pada *hidden layer* yaitu 2, dan menggunakan fungsi aktivasi *sigmoid*. Langkah-langkah perhitungan manual proses *testing* metode ELM adalah sebagai berikut.

Langkah 1: Menghitung keluaran di *hidden layer* dengan fungsi aktivasi.

Perhitungan keluaran *hidden layer* ditunjukkan pada Persamaan 2.7. Kemudian setelah dilakukan perhitungan dilanjutkan dengan menghitung fungsi aktivasi.

Berikut ini contoh perhitungan keluaran *hidden layer*.

$$H_{init\ ij} = \left(\sum_{i=1}^n w_i x_j \right) + b_j$$

$$\begin{aligned} H_{init\ 1,1} &= ((0,07 * 0,324444444) + (0,93 * 0,195555556) \\ &\quad + (0,08 * 0,466666667) + (0,24 * 0,204444444)) + 0,1 \\ &= 0,390977778 \end{aligned}$$

Hasil dari keluaran *hidden layer* secara keseluruhan ditunjukkan pada Tabel 4.14.

Table 4.14 Matriks Keluaran Hidden Layer

$H_{init\ ij}$	1	2
1	0,390977778	0,902088889
2	0,638711111	0,992311111

Berikut ini adalah contoh perhitungan dari hasil keluaran *hidden layer* dengan fungsi aktivasi.

$$H(x)_{1,1} = \frac{1}{1 + e^{-x}} = \frac{1}{1 + e^{-0,390977778}} = 0,596518057$$

Hasil keluaran *hidden layer* dengan fungsi aktivasi secara keseluruhan ditunjukkan pada Tabel 4.15 sebagai berikut.

Table 4.15 Matriks Keluaran Hidden Layer Dengan Fungsi Aktivasi

$H(x)$	1	2
1	0,596518057	0,711378581
2	0,654462047	0,729544168

Langkah 2: Menghitung keluaran di *output layer*. Perhitungan keluaran di *output layer* ditunjukkan pada Persamaan 2.12 dengan menggunakan *output weight* yang telah dihitung pada proses *training*. Berikut ini contoh perhitungan keluaran *output layer*.

$$y = H(x)\beta$$

$$y = (0,596518057 * 1,155246) + (0,711378581 * -0,55712) = 0,292798812$$

Hasil keluaran *output layer* secara keseluruhan ditunjukkan pada Tabel 4.16

Universitas Brawijaya
Universitas Brawijaya

Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya

Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya
Universitas Brawijaya

Table 4.16 Hasil Keluaran Pada Output Layer

Y
0,292798812
0,349617887

4.3.5 Perhitungan Denormalisasi Data

Denormalisasi data adalah proses untuk mengembalikan nilai data menjadi nilai semula atau nilai yang sebenarnya berdasarkan hasil peramalan yang akan dilakukan. Proses perhitungan denormalisasi menggunakan Persamaan 2.5.

Berikut ini adalah perhitungan denormalisasi data.

$$X_1 = (0,292798812(232 - 7)) + 7 = 72,87973271$$

Hasil perhitungan denormalisasi secara keseluruhan ditunjukkan pada Tabel 4.17.

Table 4.177 Hasil Denormalisasi Data

Denormalisasi
72,87973271
85,6640245

4.3.6 Perhitungan Nilai MAPE

Mean Absolute Percentage Error (MAPE) adalah metode untuk mengevaluasi hasil peramalan. Pada perhitungan nilai MAPE menggunakan Persamaan 2.2.

Berikut ini adalah perhitungan hasil nilai MAPE.

$$MAPE = \frac{1}{2} \left(\frac{|72,87973271 - 77|}{77} \right) + \left(\frac{|85,6640245 - 85|}{85} \right) \times 100 = 3,0661089\%$$

4.4 Perancangan Skenario Pengujian

Pengujian pada penelitian ini terdiri dari pengujian yang terkait algoritme *Extreme Learning Machine* (ELM) yang digunakan untuk peramalan curah hujan dan pengujian terkait tingkat error. Skenario pengujian yang dilakukan antara lain sebagai berikut.

1. Pengujian jumlah variabel *lag time* (fitur)

2. Pengujian perbandingan jumlah data *training*

3. Pengujian jumlah *neuron* pada *hidden layer*

4.4.1 Pengujian Jumlah Variabel *Lag Time* (Fitur)

Pengujian variabel *lag time* ini dilakukan untuk mengetahui pengaruh dari perbandingan jumlah variabel *lag time* terhadap nilai MAPE yang dihasilkan.

Pengujian perbandingan jumlah variabel *lag time* dengan 3 jenis variabel *lag time* yaitu 3 variabel *lag time*, 4 variabel *lag time*, dan 5 variabel *lag time*.

Table 4.18 Rancangan Pengujian Jumlah Variabel *Lag Time*

Percobaan ke-i	Nilai MAPE Pada Perbandingan Jumlah variabel <i>lag time</i>									
	2	3	4	5	6	7	8	9	10	
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
Rata-rata										
Nilai MAPE										

4.4.2 Pengujian Perbandingan Jumlah Data *Training*

Pengujian perbandingan jumlah data *training* ini dilakukan untuk mengetahui pengaruh dari perbandingan jumlah data *training* terhadap nilai MAPE yang dihasilkan. Pengujian perbandingan jumlah data *training* dilakukan dengan 5 jenis perbandingan data yaitu 90%, 80%, 70%, 60%, dan 50%. Rancangan pengujian perbandingan jumlah data *training* ditunjukkan pada Tabel 4.18.

Table 4.19 Rancangan Pengujian Perbandingan Jumlah Data *Training*

Percobaan ke-i	Nilai MAPE Pada Perbandingan Jumlah Data <i>Training</i>				
	90%	80%	70%	60%	50%
1					
2					
3					
4					

5									
6									
7									
8									
9									
10									
Rata-rata									
Nilai MAPE									

4.4.3 Pengujian Jumlah *Neuron* Pada *Hidden Layer*

Pengujian jumlah *neuron* pada *hidden layer* bertujuan untuk mendapatkan hasil pengenalan yang lebih baik. Jumlah *neuron* pada *hidden layer* yang diuji pada penelitian ini adalah 2 sampai 10 (Chao, 2017). Rancangan pengujian jumlah *neuron* pada *Hidden Layer* ditunjukkan pada Tabel 4.19.

Table 4.20 Rancangan Pengujian Jumlah *neuron* Pada *Hidden Layer*

Percobaan ke-i	Nilai MAPE Pada Jumlah <i>Hidden Neuron</i>									
	2	3	4	5	6	7	8	9	10	
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
Rata-rata										
Nilai MAPE										



BAB 5 IMPLEMENTASI

5.1 Implementasi Algoritme *Extreme Learning Machine*

Pada bab ini menjelaskan hasil implementasi dari perancangan pada bab sebelumnya. Hasil perancangan algoritme *Extreme Learning Machine* akan diterapkan pada sistem dengan kode program sesuai dengan bab sebelumnya.

Pada implementasi ini proses implementasi hasil perancangan algoritme ELM menjadi implementasi beberapa kode program yaitu kode program normalisasi data, kode program *training* dan *testing* yang akan dibagi ke beberapa sub proses, kode program denormalisasi dan kode program untuk menghitung hasil nilai MAPE.

5.1.1 Implementasi Normalisasi Data

Proses normalisasi data merupakan proses untuk menyamakan *range* nilai data pada setiap fitur. Tahap awal dari proses normalisasi adalah mencari nilai *min* dan *max* dari setiap data fitur. Hasil nilai *min* dan *max* selanjutnya digunakan untuk menormalisasi nilai keseluruhan data. Implementasi proses normalisasi data ditunjukkan pada Kode Program 5.1.

Normalisasi Data	
1 public double getMinMatrik(Matrik x) { 2 double min = 999999999; 3 for (int i = 0; i < x.getNBaris(); i++) { 4 for (int j = 0; j < x.getNKolom(); j++) { 5 if (x.getItem(i, j) < min) { 6 min = getItem(i, j); 7 } 8 } 9 } 10 return min; 11 } 12 13 public double getMaxMatrik(Matrik y) { 14 double maks = -9999; 15 for (int i = 0; i < y.getNBaris(); i++) { 16 for (int j = 0; j < y.getNKolom(); j++) { 17 if (y.getItem(i, j) > maks) { 18 maks = getItem(i, j); 19 } 20 } 21 } 22 return maks; 23 } 24 25 public Matrik normalisasiMatrik(Matrik latih, Matrik 26 master) { 27 double dtTemp; 28 Matrik dtNorm = new Matrik(latih.getNBaris(), 29 latih.getNKolom()); 30 for (int i = 0; i < latih.getNBaris(); i++) { 31 for (int j = 0; j < latih.getNKolom(); j++) { 32 dtTemp = (latih.getItem(i, j) -	

```
33     master.getMinMatrik(master)) /  
34     (master.getMaxMatrik(master))  
35     master.getMinMatrik(master));  
36     dtNorm.setItem(i, j, dtTemp);  
37 }  
38     return dtNorm;  
39 }  
40 }
```

Kode Program 5.1 Implementasi Normalisasi Data

Berikut ini adalah penjelasan dari implementasi kode program 5.1.

1. Baris 1 melakukan deklarasi fungsi *getMinMatrik* dengan parameter nilai Matrik *x*.
2. Baris 2 melakukan inisialisasi variabel *min* untuk nilai minimum.
3. Baris 3 melakukan perulangan dengan parameter variabel *i* lebih kecil dari *getNBaris*.
4. Baris 4 melakukan perulangan dengan parameter variabel *j* lebih kecil dari *getNKolom*.
5. Baris 5-6 menjelaskan kondisi jika variabel *getItem* lebih kecil dari variabel *min* dan inisialisasi variabel *min* dengan nilai *getItem*.
6. Baris 10 menjelaskan pengembalian nilai variabel *min*.
7. Baris 13-14 melakukan deklarasi fungsi *getmaxMatrik* dengan parameter Matrik *y* dan inisialisasi variabel *maks*.
8. Baris 15-22 menjelaskan perulangan dengan parameter variabel *i* lebih kecil dari *getNBaris* dan variabel *j* lebih kecil dari *getNKolom* dengan kondisi jika variabel *getItem* lebih besar dari *maks* maka nilai *maks* adalah *getItem* dengan pengembalian nilai *maks*.
9. Baris 25-27 melakukan deklarasi variabel *normalisasiMatrik* pada kelas *matrik* dengan parameter *latih* dan *master* pada kelas *Matrik* dan pembuatan variabel *dtTemp*.
10. Baris 28-31 Melakukan deklarasi objek dengan variabel *dtNorm* pada kelas *Matrik* dengan Parameter *getNBaris* dan *getNKolom* dan melakukan perulangan dengan parameter variabel *i* lebih kecil dari *getNBaris* dan variabel *j* lebih kecil dari *getNKolom*.
11. Baris 32-39 Melakukan inisialisasi varabel *dtTemp* dengan nilai selisih data *main* dan *max* dengan parameter *master* kemudian melakukan pengembalian nilai dari variabel *dtNorm*.

5.1.2 Implementasi Input Weight dan Bias

Pada tahap ini implementasi bertujuan untuk melakukan inisialisasi *input weight* (bobot) dan *hidden node* (bias) yang dilakukan dengan *random*. Implementasi *input weight* (bobot) dan *hidden node* (bias) ditunjukkan pada Kode Program 5.2.

<i>Input Weight dan Bias</i>	
1	public Matrik setBiasHNode(int jmlN){
2	Matrik x = new Matrik(1, jmlN);
3	double tmp;
4	for (int j = 0; j < x.getNKolom(); j++) {
5	tmp = Math.random() * 1;
6	x.setItem(0, j, tmp);
7	}
8	return x;
9	}
10	
11	public Matrik setRandomBobot(Matrik x) {
12	double tmp;
13	for (int i = 0; i < x.getNBaris(); i++) {
14	for (int j = 0; j < x.getNKolom(); j++) {
15	tmp = (Math.random() * 1) - 0.2;
16	x.setItem(i, j, tmp);
17	}
18	}
19	return x;
20	}

Kode Program 5.2 Implementasi *Input Weight* dan *Bias*

Berikut ini adalah penjelasan dari implementasi Kode Program 5.2.

1. Baris 1 melakukan deklarasi fungsi *setBiasHNode* dari kelas *Matrik* dengan parameter *jmlN*.
2. Baris 2-3 membuat objek baru dengan variabel *x* dari kelas *Matrik* dengan parameter *jmlN* dan inisialisasi variabel *tmp*.
3. Baris 4-6 melakukan perulangan dengan parameter variabel *j* lebih kecil dari *getNKolom* dan Inisialisasi variabel *tmp* dengan nilai *random* kemudian melakukan pengembalian dengan variabel *x*.
4. Baris 11 melakukan deklarasi fungsi *setRandomBobot* dari kelas *Matrik* dengan paremeter *a*.
5. Baris 12-14 melakukan inisialisasi variabel *tmp* dan membuat perulangan dengan parameter variabel *i* lebih kecil dari *getNBaris* dan variabel *j* lebih kecil dari *getNKolom*
6. Baris 15-19 melakukan inisialisasi variabel *tmp* dengan nilai *random* kemudian melakukan pengembalian dengan varabel *x*.

5.1.3 Implementasi Keluaran *Hidden Layer* dengan Fungsi Aktivasi

Pada tahap ini akan mencari nilai dari keluaran *hidden layer* pada proses *training* maupun pada proses *testing* dan dengan menggunakan fungsi aktivasi *sigmoid biner*. Pada proses *training* akan menggunakan data *training* sedangkan pada proses *testing* menggunakan data *testing*. Implementasi keluaran *hidden layer* dengan fungsi aktivasi ditunjukkan pada Kode Program 5.3.

Keluaran *Hidden Layer* dengan Fungsi Aktivasi

1	public Matrik setHInit(Matrik hi, Matrik nbias) {
2	Matrik tmp = new Matrik();



```
3     double a;
4     tmp = hi;
5     for (int i = 0; i < hi.getNBaris(); i++) {
6         for (int j = 0; j < hi.getNKolom(); j++) {
7             a = hi.getItem(i, j) + nbias.getItem(0, j);
8             tmp.setItem(i, j, a);
9         }
10    }
11    return tmp;
12 }
```

Kode Program 5.3 Implementasi keluaran *hidden layer*

Berikut ini adalah penjelasan dari implementasi Kode Program 5.3.

1. Baris 1 menjelaskan deklarasi fungsi *setHInit* dari kelas *Matrik* dengan parameter *hi* dan *hbias* dari kelas *Matrik*.
2. Baris 2-4 membuat objek *tmp* dari kelas *Matrik* dan inisialisasi variable *a* dengan tipe data *double*, inisialisasi varibel *tmp* dengan nilai *hi*.
3. Baris 5-6 melakukan perulangan dengan parameter variabel *i* lebih kecil dari *getNBaris* dan variabel *j* lebih kecil dari *getNKolom*.
4. Baris 7-11 menjelaskan inisialisasi variable *a* dengan nilai hasil perkalian matrik dan penjumlahan dengan bias kemudian mengembalikan variabel *tmp*.

Keluaran *Hidden Layer* dengan Fungsi Aktivasi

```
1 public Matrik setFinalH(Matrik hInit) {
2     Matrik tmp = new Matrik();
3     double a;
4     tmp = hInit;
5     for (int i = 0; i < tmp.getNBaris(); i++) {
6         for (int j = 0; j < tmp.getNKolom(); j++) {
7             a = 1 / (1 + Math.exp(-hInit.getItem(i,
8                 j)));
9             tmp.setItem(i, j, a);
10        }
11    }
12    return tmp;
}
```

Kode Program 5.4 Implementasi dengan Fungsi Aktivasi

Berikut ini adalah penjelasan dari implementasi Kode Program 5.4.

1. Baris 1 menjelaskan deklarasi fungsi *setFinalH* dari kelas *Matrik* dengan parameter *hInit* dari kelas *Matrik*.
2. Baris 2-4 membuat objek *tmp* dari kelas *Matrik* dan inisialisasi variable *a* dengan tipe data *double*, inisialisasi varibel *tmp* dengan nilai *hInit*.
3. Baris 5-6 melakukan perulangan dengan parameter variabel *i* lebih kecil dari *getNBaris* dan variabel *j* lebih kecil dari *getNKolom*.
4. Baris 7-11 menjelaskan inisialisasi variable *a* dengan nilai hasil perhitungan dengan fungsi aktivasi *sigmoid biner* kemudian mengembalikan variabel *tmp*.

5.1.4 Implementasi Output Weight

Pada tahap ini akan dilakukan implementasi mencari output weight yang merupakan bagian dari implementasi *Moore-Penrose Generalized Inverse*. Pada implementasi *Moore-Penrose Generalized Inverse* terdapat sub proses yaitu proses transpose nilai H , proses perkalian matriks $H^T H$, proses inverse hasil perkalian matriks $H^T H$ dan proses perkalian matriks inverse $H^T H$ dengan H^T . Terakhir akan dilakukan implementasi output weight dari perkalian matriks hasil dari implementasi *Moore-Penrose Generalized Inverse* dengan matriks data *target training*. Implementasi transpose nilai H ditunjukkan pada Kode Program 5.3.

Transpose Matriks H

```
1 public Matrik transposeMatrik(Matrik x) {  
2     Matrik transpose = new Matrik(x.getNKolom(),  
3         x.getNBaris());  
4     for (int c = 0; c < x.getNBaris(); c++) {  
5         for (int d = 0; d < x.getNKolom(); d++) {  
6             transpose.setItem(d, c, x.getItem(c, d));  
7         }  
8     }  
9     return transpose;  
}
```

Kode Program 5.5 Implementasi Transpose Matriks H

Berikut ini adalah penjelasan dari implementasi Kode Program 5.5.

1. Baris 1 melakukan deklarasi fungsi *transpose* Matrik dari kelas *Matrik* dengan parameter *Matrik x*.
2. Baris 2 membuat objek *tr* dari kelas *Matrik* dengan parameter *kolom* dan *baris*.
3. Baris 3-8 melakukan perulangan dengan parameter variabel *c* lebih kecil dari *getNBaris* dan variabel *d* lebih kecil dari *getNKolom* dengan pengembalian nilai *transpose*.

Perkalian Matriks $H^T H$

```
1 public Matrik kaliMatrik(Matrik a, Matrik b) {  
2     Matrik Mkali = new Matrik(a.getNBaris(),  
3         b.getNKolom());  
4     double tmp;  
5     for (int i = 0; i < Mkali.getNBaris(); i++) {  
6         for (int j = 0; j < Mkali.getNKolom(); j++) {  
7             tmp = 0;  
8             for (int k = 0; k < b.getNBaris(); k++) {  
9                 tmp += a.getItem(i, k) * b.getItem(k, j);  
10            }  
11            Mkali.setItem(i, j, tmp);  
12        }  
13    }  
14    return Mkali;  
15 }
```

Kode Program 5.6 Implementasi Perkalian Matriks $H^T H$



Berikut ini adalah penjelasan dari implementasi Kode Program 5.6.

1. Baris 1 melakukan deklarasi fungsi *kaliMatrik* dari kelas *Matrik* dengan parameter variabel *a* dan *b* dari kelas *Matrik*.
2. Baris 2-4 membuat objek *Mkali* dari kelas *Matrik* dengan parameter *baris* dan *kolom*, inisialisasi varibel *tmp* dengan tipe data *double*.
3. Baris 5-6 melakukan perulangan dengan parameter variabel *i* lebih kecil dari *getNBaris* dan variabel *j* lebih kecil dari *getNKolom* pada objek *Mkali*.
4. Baris 7-14 Menjelaskan inisialisasi variable *tmp* dengan nilai hasil perkalian matrik dan kemudian mengembalikan variabel *Mkali*.

Inverse $H^T H$

```
1  public double[][] invert(double a[][]) {  
2      int n = a.length;  
3      double x[][] = new double[n][n];  
4      double b[][] = new double[n][n];  
5      int index[] = new int[n];  
6      for (int i = 0; i < n; ++i) {  
7          b[i][i] = 1;  
8      }  
9      for (int i = 0; i < n - 1; ++i) {  
10         for (int j = i + 1; j < n; ++j) {  
11             for (int k = 0; k < n; ++k) {  
12                 b[index[j]][k] -= a[index[j]][i]  
13                 b[index[i]][k];  
14             }  
15         }  
16         for (int i = 0; i < n; ++i) {  
17             x[n - 1][i] = b[index[n - 1]][i] / a[index[n -  
18             1]][n - 1];  
19             for (int j = n - 2; j >= 0; --j) {  
20                 x[j][i] = b[index[j]][i];  
21                 for (int k = j + 1; k < n; ++k) {  
22                     x[j][i] -= a[index[j]][k] * x[k][i];  
23                 }  
24                 x[j][i] /= a[index[j]][j];  
25             }  
26         }  
27     }  
28     return x;  
29 }  
30 public void gaussian(double a[][], int index[]) {  
31     int n = index.length;  
32     double c[] = new double[n];  
33     for (int i = 0; i < n; ++i) {  
34         index[i] = i;  
35     }  
36     for (int i = 0; i < n; ++i) {  
37         double c1 = 0;  
38         for (int j = 0; j < n; ++j) {  
39             double c0 = Math.abs(a[i][j]);  
40             if (c0 > c1) {  
41                 c1 = c0;  
42             }  
43         }
```

```
44 }  
45 }  
46 c[i] = c1;  
47 }  
48 int k = 0;  
49 for (int j = 0; j < n - 1; ++j) {  
50     double pi1 = 0;  
51     for (int i = j; i < n; ++i) {  
52         double pi0 = Math.abs(a[index[i]][j]);  
53         pi0 /= c[index[i]];  
54         if (pi0 > pi1) {  
55             pi1 = pi0;  
56             k = i;  
57         }  
58     }  
59     int itmp = index[j];  
60     index[j] = index[k];  
61     index[k] = itmp;  
62     for (int i = j + 1; i < n; ++i) {  
63         double pj = a[index[i]][j] / a[index[j]][j];  
64         a[index[i]][j] = pj;  
65         for (int l = j + 1; l < n; ++l) {  
66             a[index[i]][l] -= pj * a[index[j]][l];  
67         }  
68     }  
69 }  
70 }  
71 }  
72 }
```

Kode Program 5.7 Implementasi Matriks Inverse $H^T H$

Berikut ini adalah penjelasan dari implementasi Kode Program 5.7.

1. Baris 1 melakukan deklarasi fungsi *invert* dengan parameter matrik *a* tipe data *double*.
2. Baris 2-4 membuat variabel *n* tipe data integer dengan nilai panjang matrik *a* dan membuat objek matrik *x* dan *b* dengan tipe data *double*.
3. Baris 5-7 membuat objek *matrik index* dengan parameter *n* dan membuat perulangan dengan parameter variabel *n*, inisialisasi matrik *b* dengan nilai 1.
4. Baris 9-13 membuat perulangan dengan parameter *n-1*, membuat perulangan nilai *n* terhadap variabel *j* dan variabel *k* dan inisialisasi matrik *b* dengan index *i,j* dan *k*.
5. Baris 17-28 membuat perulangan dengan parameter *n*, inisialisasi matrik *x* dan inisialisasi matrik *x* dengan index *matrik a* kemudian pengembalian variabel *x*.
6. Baris 30 menjelaskan deklarasi *method void Gaussian* dengan parameter matrik *a*.
7. Baris 31-32 membuat inisialisasi variabel *n* tipe data *integer* dengan nilai *index.length* dan membuat objek *c* dengan variabel *n* tipe data *double*.

8. Baris 34-36 membuat perulangan dengan parameter index *i* lebih kecil dari variabel *n*.
9. Baris 38-47 membuat perulangan dengan parameter variabel *i* dan *j*. inisialisasi variabel *c1* dan *c0* kemudian melakukan pengembalian nilai dengan matrik *c*.
10. Baris 48-58 menjelaskan inisialisasi variabel *k* dengan nilai 0 dan membuat perulangan dengan parameter *i* dan *j*, inisialisasi variabel *pi0* , *pi1* dan *k*.
11. Baris 59-61 membuat inisialisasi variabel *tmp* dengan matrik index *j*. inisialisasi variabel matrik index *j* dengan matrik index *k* dan insialisasi matrik index *k* dengan variabel *itmp*.
12. Baris 62- 72 membuat perulangan dengan parameter variabel *l* dan inisialisasi variabel matrik *a* dengan nilai *pj*.

<i>Output Weigth</i>	
1	public void setDtBeta() {
2	dtBeta = new Matrik();
3	dtBeta = dtBeta.kaliMatrik(getDtHPlus(),
4	getDtTargetNorm());
5	}

Kode Program 5.8 Implementasi *Output Weigth*

Berikut ini adalah penjelasan dari implementasi Kode Program 5.8.

1. Baris 1 menjelaskan deklarasi method *void* dengan variabel *setDtBeta*.
2. Baris 2 membuat inisialisasi objek variabel *dtBeta* dengan parameter perkalian matrik *getDtHPlus* dan *getDtTargetNorm*.

5.1.5 Implementasi Keluaran *Output Layer*

Pada tahap *testing* akan dilakukan implementasi untuk mencari nilai hasil peramalan sebelum di denormalisasi , dengan data *testing*. Proses ini dibagi menjadi implementasi *Hinit* dan *H* pada data *testing* kemudian mencari hasil peramalan sebelum di denormalisasi. Implementasi mencari *Hinit testing* ditunjukkan pada Kode Program 5.9.

<i>Hinit Testing</i>	
1	public void setDtHTesting() {
2	dtHTesting = new Matrik();
3	dtHTesting = m.setH(getDtNormTes(), getDtWT());
4	}
5	
6	7 public void setDtHInitTesting() {
7	dtHInitTesting = new Matrik();
8	dtHInitTesting = dtHInitTesting.setHInit(getDtHTes(),
9	getDtBias());
10	}
11	

Kode Program 5.9 Implementasi *Hinit Testing*

Berikut ini adalah penjelasan dari implementasi kode program 5.9.

1. Baris 1 menjelaskan deklarasi *method void* dengan variabel *setDtHTesting*.

2. Baris 2 membuat objek variabel *dtHTesting* dari kelas Matrik.

3. Baris 3 menjelaskan inisialisasi variabel *dtHTesting* dengan perkalian matriks pada fungsi *setH*.

4. Baris 7 menjelaskan deklarasi *method void* dengan variabel *setDtHInitTesting*.

5. Baris 8 membuat objek variabel *dtHInitTesting* dari kelas Matrik.

6. Baris 9-10 menjelaskan inisialisasi variabel *dtHInitTesting* dengan perkalian matriks pada fungsi *setHInit*.

<i>H Testing</i>	
1	public void setDtFinalHTesting() {
2	dtFinalHTesting = new Matrik();
3	dtFinalHTesting = dtFinalHTes.setFinalH(getDtHInitTes());
4	}

Kode Program 5.10 Implementasi *H Testing*

Berikut ini adalah penjelasan dari implementasi Kode Program 5.10.

1. Baris 1 melakukan deklarasi *method void* dengan variabel *setDtFinalHTesting*.

2. Baris 2 membuat objek variabel *dtFinalHTesting* dari kelas Matrik.

3. Baris 3-4 menjelaskan inisialisasi variabel *dtFinalHTesting* dengan perkalian matriks pada fungsi *setFinalH*.

Hasil Peramalan	
1	public void setDtHasilPeramalan() {
2	dtHasilPeramalan = new Matrik();
3	dtHasilPeramalan
4	dtHasilPeramalan.kaliMatrik(getDtFinalHTes(),
5	getDtBeta());

Kode Program 5.11 Implementasi Hasil Peramalan

Berikut ini adalah penjelasan dari implementasi kode program 5.11.

1. Baris 1 menjelaskan deklarasi *method void* dengan variabel *setDtHasilPeramalan*.

2. Baris 2-3 membuat objek dengan variabel *dtHasilPeramalan* dari kelas Matrik.

3. Baris 4-5 menjelaskan inisialisasi variabel *dtHasilPeramalan* dengan perkalian matriks pada fungsi *kaliMatrik* dengan parameter *getDtFinalHtes* dan *getDtBeta*.

5.1.6 Implementasi Denormalisasi Data

Pada implementasi ini dilakukan proses pengembalian nilai hasil peramalan ke nilai sebelum normalisasi. Implementasi proses denormalisasi data ditunjukkan pada Kode Program 5.12.

Denormalisasi Data	
1	public void setDenormalisasi() {
2	dtDenormalisasi=new
3	Matrik(getDtHasilPeramalan().getNBaris(), Brawijaya
4	getDtHasilPeramalan().getNKolom());
5	double denormalisasi, min, max;
6	min = dtPilih.getMinMatrik(getDtPilih());
7	max = dtPilih.getMaxMatrik(getDtPilih());
8	for (int i=0; i < dtDenormalisasi.getNBaris();i++) {
9	for (int j=0; j<dtDenormalisasi.getNKolom();j++) {
10	denormalisasi = getDtHasilPeramalan().getItem(i,
11	j) * (max - min) + min;
12	dtDenormalisasi.setItem(i, j, denormalisasi);
13	}
14	}
15	}
16	}

Kode Program 5.12 Implementasi Denormalisasi Data

Berikut ini adalah penjelasan dari implementasi Kode Program 5.12.

1. Baris 1 melakukan deklarasi *method void* dengan variabel *setDenormalisasi*.
2. Baris 2-4 melakukan inisialisasi objek dengan variabel *dtDenormalisasi* pada kelas matrik dengan parameter data hasil peramalan, jumlah baris dan kolom.
3. Baris 5-7 membuat variabel denormalisasi, *min* dan *max* dengan tipe data double, inisialisasi variabel *min* dan *max* dengan nilai fungsi *getMinMatrik* dan *getMaxMatrik* dengan parameter *getDtPilih*.
4. Baris 8-9 membuat perulangan dengan parameter variabel *i* lebih kecil dari baris data denormalisasi dan variabel *j* lebih kecil dari kolom data denormalisasi.
5. Baris 10-11 menjelaskan inisialisasi variabel denormalisasi dengan perkalian denormalisasi pada data hasil peramalan.
6. Baris 12 menjelaskan variabel *dtDenormalisasi* pada fungsi *setItem* dengan parameter *i, j* dan denormalisasi.

5.1.7 Implementasi Menghitung Mean Absolute Percentage Error (MAPE)

Pada tahap ini melakukan implementasi proses evaluasi terhadap data yang telah di denormalisasi pada implementasi sebelumnya. Pada proses ini akan memakai proses evaluasi *Mean Absolute Percentage Error* sebagai metode untuk evaluasi. Implementasi proses denormalisasi data ditunjukkan pada Kode Program 5.13.



Proses MAPE	
1	public void setMape() {
2	dtMape = new Matrik(getDtDenormalisasi().getNBaris(),
3	getDtDenormalisasi().getNKolom());
4	for (int i = 0; i < dtMape.getNBaris(); i++) {
5	for (int j = 0; j < dtMape.getNKolom(); j++) {
6	double tmp = Math.abs(((getDtDenormalisasi()).getItem(i,
7	j) - getDtTargetTes().getItem(i, j))
8	/ getDtTargetTes().getItem(i, j) * 100));
9	dtMape.setItem(i, j, tmp);
10	}
11	}
12	}
13	public void setError() {
14	error = 0;
15	for (int i = 0; i < getMape().getNBaris(); i++) {
16	for (int j=0; j < getMape().getNKolom(); j++) {
17	error += getMape().getItem(i, j);
18	}
19	}
20	error /=getMape().getNBaris();
21	}

Kode Program 5.13 Implementasi Menghitung *Mean Absolute Percentage Error*

Berikut ini adalah penjelasan dari implementasi Kode Program 5.13.

1. Baris 1 melakukan deklarasi *method void* dengan variabel *setMape*.
2. Baris 2-3 melakukan inisialisasi objek dengan variabel *dtMape* pada kelas matrik dengan parameter data denormalisasi, jumlah baris dan kolom.
3. Baris 4-5 membuat perulangan dengan parameter variabel *i* lebih kecil dari baris *dtMape* dan variabel *j* lebih kecil dari kolom *dtMape*.
4. Baris 6-10 membuat inisialisasi variabel *tmp* tipe data double dengan persamaan MAPE.
5. Baris 13 melakukan deklarasi *method void* dengan variabel *setError*.
6. Baris 14 menjelaskan inisialisasi variabel dengan nilai 0.
7. Baris 15-16 membuat perulangan dengan parameter variabel *i* lebih kecil dari baris nilai MAPE dan variabel *j* lebih kecil dari kolom MAPE.
8. Baris 17 menjelaskan inisialisasi variabel *error* dengan perkalian matrik data MAPE.
9. Baris 18 menjelaskan inisialisasi variabel *error* dengan fungsi pengambilan jumlah baris pada hasil MAPE.

BAB 6 PENGUJIAN DAN PEMBAHASAN

Pada bab ini menjelaskan pengujian dan pembahasan dari hasil implementasi algoritme *Extreme Learning Machine* (ELM) dalam peramalan curah hujan. Hasil pengujian dilakukan berdasarkan nilai MAPE sesuai dengan hasil implementasi.

Pengujian yang akan dilakukan pada bab ini yaitu pengujian jumlah *neuron*, pengujian variasi jumlah data *training* dan data *testing* kemudian pengujian variabel *lag time* (fitur). Pada masing-masing pengujian akan dilakukan 10 kali percobaan dengan hasil masing-masing MAPE dan akan dihitung rata-rata MAPE.

Kemudian pada masing-masing hasil pengujian akan dilakukan pembahasan.

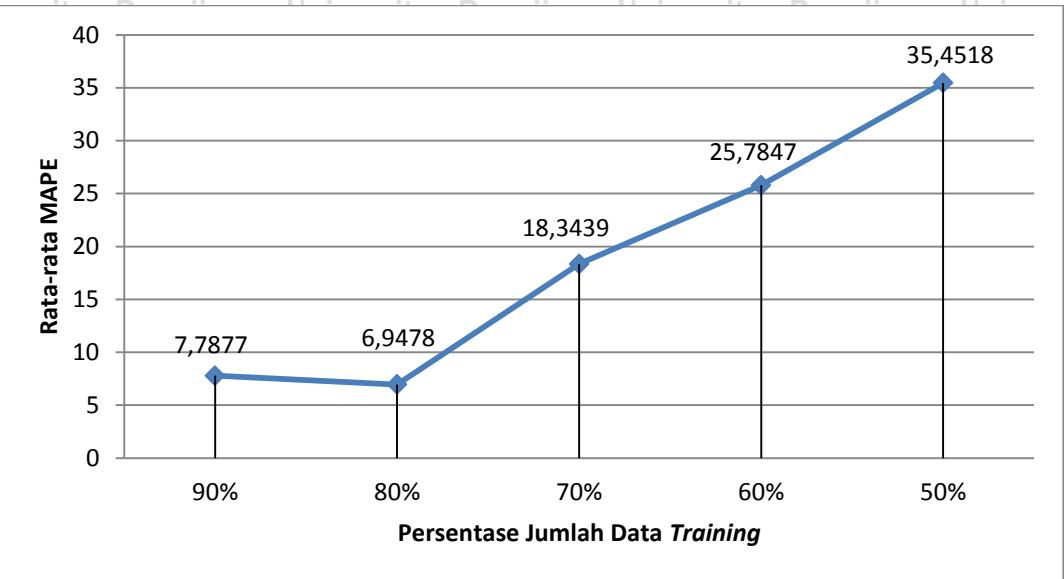
6.1 Pengujian Perbandingan Persentase Jumlah Data *Training*

Pada tahap ini dilakukan pengujian dengan perbandingan persentase antara banyak data *training* dan data *testing*. Melalui pengujian ini diharapkan dapat mengetahui nilai perbandingan data yang menghasilkan nilai rata-rata MAPE terkecil dari 10 percobaan dimasing-masing jumlah persentase data. Pengujian pada penelitian ini melakukan perbandingan banyak data dilakukan sebanyak 5 variasi yaitu dengan perbandingan 90%, 80%, 70%, 60% dan 50%. Pada masing-masing variasi persentase data *training* akan dilakukan pengujian dengan parameter yaitu variabel *lag time* (fitur) sebesar 4, banyak *neuron* pada *hidden layer* yaitu 2. Hasil pengujian perbandingan persentase jumlah data *training* pada Tabel 6.1.

Table 6.1 Hasil Pengujian Persentase banyak Data *Training*

Percobaan ke-i	Nilai MAPE Persentase Jumlah Data <i>Training</i>				
	90%	80%	70%	60%	50%
1	14,4221	6,5783	18,6277	24,7561	30,1235
2	4,1272	3,9812	12,2763	28,3832	32,8781
3	3,7123	14,4511	23,4091	20,1712	38,5612
4	13,1725	7,1901	26,0786	18,9715	41,3721
5	8,1021	6,3914	18,6401	34,7326	43,8763
6	16,0716	4,8168	14,7202	35,0423	30,2556
7	7,2052	3,2152	15,5821	28,0217	37,0783
8	4,0112	5,7907	13,1654	23,2566	35,8241
9	3,6021	13,5865	23,2742	21,7523	34,5214
10	3,4504	3,4766	17,6653	22,7592	30,0272
Rata-rata	7,7877	6,9478	18,3439	25,7847	35,4518
Nilai MAPE					

Melalui hasil pengujian pada Tabel 6.1 dapat ditampilkan rata-rata nilai MAPE yang diperoleh dari 10 percobaan pada masing-masing persentase data yang ditampilkan dalam bentuk diagram garis. Berikut diagram garis untuk rata-rata nilai MAPE hasil pengujian persentase banyak data *training* dapat dilihat pada gambar 6.1.



Gambar 6.1 Hasil Pengujian Persentase Jumlah Data *training*

Melalui hasil pengujian pada bagian Tabel 6.1 dan Gambar 6.1 dapat dilihat bahwa nilai dari rata-rata MAPE terkecil terlihat pada persentase sebesar 80% dengan nilai sebesar 6,9478% sedangkan nilai rata-rata MAPE terbesar pada persentase 50% dengan nilai 35,4518%. Melalui pengujian tersebut disimpulkan bahwa persentase data *training* berpengaruh pada nilai rata-rata MAPE. Semakin besar data *training* maka nilai rata-rata MAPE semakin kecil sebaliknya semakin kecil data *training* terjadi kenaikan nilai rata-rata MAPE. Hal ini terjadi karena semakin banyak proses pembelajaran dengan menggunakan data *training* maka pengetahuan pembelajaran algoritme juga semakin baik, sebaliknya semakin sedikit menyebabkan algoritme kurang mampu mempelajari data yang diujikan dengan pengetahuan pembelajaran sedikit.

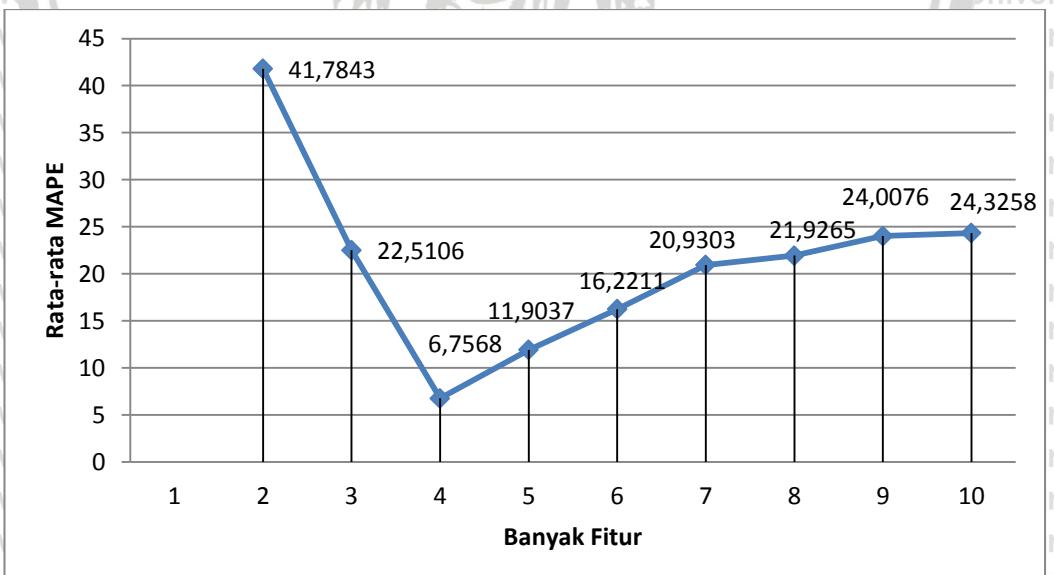
6.2 Pengujian Jumlah Variabel *Lag Time* (Fitur)

Pada pengujian jumlah fitur akan dilakukan pengujian pada jumlah variabel *lag time* (fitur) untuk mengetahui jumlah fitur yang paling optimal (nilai rata-rata MAPE paling kecil). Jumlah parameter yang digunakan sebagai pengujian banyak fitur adalah 2 sampai 10. Setiap parameter nilai yang di uji dilakukan sebanyak 10 kali percobaan. Pada masing-masing jumlah fitur yang akan diuji dilakukan pengujian dengan parameter yaitu persentase data *training* yaitu sebesar 80%, banyak *neuron* pada *hidden layer* yaitu 2. Hasil pengujian jumlah variabel *lag time* ditunjukkan pada Tabel 6.2 dan hasil keseluruhan secara detail terdapat pada Lampiran B..

**Table 6.2 Hasil Pengujian Jumlah Variabel lag Time (Fitur)**

Percobaan ke-i	Nilai MAPE Pada Jumlah Hidden Neuron					
	2	3	4	5	...	10
1	41,2351	23,9272	6,1721	11,8121	...	21,4437
2	42,2123	25,1281	3,6852	12,2672	...	21,7218
3	40,1264	20,1827	8,2923	9,7681	...	24,5062
4	42,5223	21,0272	8,1271	12,7872	...	26,2461
5	41,2548	23,9021	4,7562	8,7826	...	24,7402
6	41,0251	21,7127	6,8512	14,7181	...	26,3891
7	43,0238	24,6261	3,8831	12,2621	...	25,2728
8	42,1187	22,0652	8,1245	8,9903	...	26,3712
9	43,1522	21,2621	8,9821	14,8276	...	23,5218
10	41,1724	21,2728	8,6946	12,8213	...	23,0451
Rata-rata Nilai MAPE	41,7843	22,5106	6,7568	11,9037	...	24,3258

Berdasarkan hasil pengujian pada Tabel 6.2 diperoleh hasil nilai dari rata-rata MAPE yang kemudian ditampilkan dalam bentuk diagram garis. Gambar diagram garis hasil pengujian jumlah variabel lag time (fitur) ditunjukkan pada Gambar 6.2.

**Gambar 6.2 Hasil Pengujian Banyak variabel lag time (fitur)**

Melalui hasil pengujian pada Tabel 6.2 dan Gambar 6.2 dapat diperhatikan bahwa jumlah fitur sebanyak 4 menghasilkan nilai dari rata-rata MAPE yang terkecil dengan nilai 6,7568%. Nilai dengan rata-rata MAPE terbesar ada pada

jumlah fitur sebanyak 2 dengan nilai 41.7843%. Berdasarkan pengujian tersebut dapat di perhatikan bahwa nilai rata-rata MAPE pada fitur sebanyak 2 memiliki nilai paling besar karena proses pembelajaran yang kurang dari jumlah fitur yang digunakan. Pada jumlah fitur 4 sampai 10 terjadi kenaikan karena perbedaan data time series (terdapat perbedaan 2 digit pada data curah hujan) yang digunakan dan data yang digunakan hanya berjumlah 14 tahun (hanya 14 data time series curah hujan). Hal tersebut membuktikan bahwa pengujian banyak variabel *lag time* (fitur) dapat mempengaruhi nilai rata-rata MAPE.

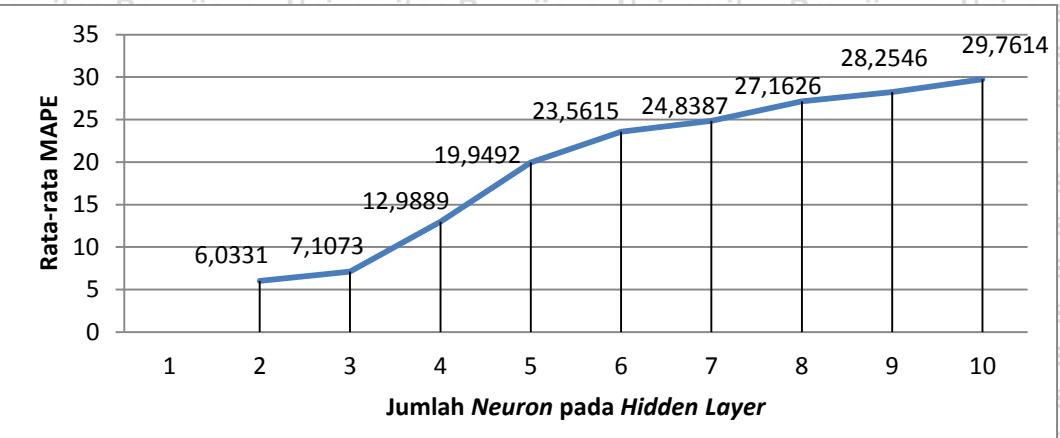
6.3 Pengujian Jumlah *Neuron* pada *Hidden Layer*

Pada pengujian ini dilakukan untuk mengetahui banyak jumlah *neuron* yang paling optimal dalam peramalan curah hujan menggunakan algoritme ELM. Banyak jumlah parameter yang di gunakan yaitu 2 sampai 10 *neuron* pada *hidden layer*. Pada pengujian ini menggunakan persentase data training dan testing yaitu sebesar 80% sedangkan untuk jumlah *lag time* (fitur) yaitu sebanyak 4 fitur. Hasil pengujian jumlah *neuron* pada *hidden layer* ditunjukkan pada Tabel 6.3 dan tabel hasil pengujian jumlah *neuron* secara detail ditunjukkan pada Lampiran C.

Table 6.3 Hasil Pengujian Jumlah *Neuron* pada *Hidden Layer*

Percobaan ke- <i>i</i>	Nilai MAPE Pada Jumlah <i>Hidden Neuron</i>					
	2	3	4	5	...	10
1	4,3421	3,7209	14,1601	25,2122	...	32,1672
2	4,1267	8,6545	13,0278	21,0728	...	23,1872
3	3,7633	6,7612	11,1205	22,5101	...	25,6987
4	6,3718	3,8912	14,4602	21,2624	...	26,1762
5	8,5812	3,9122	10,1812	15,0202	...	35,1282
6	4,4021	8,6701	15,1081	18,1223	...	32,1282
7	4,8301	8,7892	18,0117	19,0924	...	28,7812
8	8,4521	12,5575	12,7026	18,0233	...	31,6183
9	6,5601	5,6208	13,0125	19,0875	...	30,7434
10	8,9012	8,4953	8,1044	20,0883	...	31,9853
Rata-rata Nilai MAPE	6.0331	7,1073	12,9889	19,9492	...	29,7614

Melalui hasil pengujian jumlah *neuron* pada Tabel 6.3 kemudian dilakukan analisis pengaruh jumlah *neuron* terhadap rata-rata MAPE dengan menampilkan menjadi grafik dalam bentuk diagram garis. Gambar grafik diagram garis hasil pengujian jumlah *neuron* pada *hidden layer* ditunjukkan pada Gambar 6.3.



Gambar 6.1 Hasil Pengujian Jumlah *Neuron* pada *Hidden layer*

Melalui Tabel 6.3 dan Gambar 6.3 menunjukkan bahwa nilai rata-rata MAPE terkecil terletak pada jumlah *neuron* sebanyak 2 dengan nilai sebesar 6,0331% dan nilai rata-rata MAPE terbesar adalah sebanyak 10 *neuron* dengan nilai sebesar 29,7614%. Hal tersebut disebabkan karena semakin besar jumlah *neuron* pada *hidden layer* maka dapat meningkatkan hasil ketepatan nilai peramalan mendekati nilai aktual, tetapi jumlah *neuron* yang semakin besar juga membuat ruang varabel *lag time* (fitur) yang terbentuk semakin besar sehingga membuat pencarian ruang fitur semakin luas sehingga membuat terjadi kenaikan rata-rata MAPE pada jumlah neuron. Hal ini juga membuktikan bahwa diperlukan pengujian parameter yang optimal (optimasi) jumlah *neuron* pada *hidden layer* yang semakin besar tidak selalu akan menghasilkan tingkat kesalahan semakin kecil.

BAB 7 PENUTUP

Pada bab ini akan menjelaskan bagian penutup dari penelitian peramalan curah hujan dengan metode *Extreme Learning Machine* (ELM). Pada bab ini dibagi menjadi dua bagian yaitu kesimpulan dan saran. Pada bagian kesimpulan akan menjelaskan kesimpulan dari hasil dan pengujian peramalan curah hujan dengan algoritme ELM. Pada bagian saran menjelaskan tentang hal-hal yang perlu diperhatikan dalam kekurangan penelitian ini dan memberikan saran untuk penelitian selanjutnya yang membutuhkan refrensi.

7.1 Kesimpulan

Melalui hasil implementasi dan pembahasan pengujian pada bab sebelumnya pada peramalan curah hujan menggunakan algoritme *Extreme Learning Machine* (ELM) dapat disimpulkan beberapa hal sebagai berikut.

1. Pada peramalan curah hujan menggunakan metode *Extreme Learning Machine* (ELM) menggunakan pengujian banyak variabel *lag time* (fitur), persentase banyak data *training* dan data *testing* dan banyak *neuron* pada *hidden layer*. Melalui hasil pengujian diperoleh rata-rata MAPE terkecil pada pengujian banyak variabel *lag time* (fitur) menghasilkan nilai sebanyak 4 fitur dengan nilai rata-rata MAPE sebesar 6,7568%. Pada pengujian persentase banyak data *training* menghasilkan nilai rata-rata MAPE sebesar 6,9478% dengan menggunakan perbandingan data *training* sebesar 80%. Pada pengujian jumlah *neuron* pada *hidden layer* menghasilkan rata-rata MAPE sebesar 6,0331% dengan menggunakan 2 *neuron* pada *hidden layer*. Melalui hasil nilai rata-rata MAPE pada masing-masing pengujian dapat disimpulkan bahwa pengujian variabel *lag time* (fitur), pengujian perbandingan data *training* dan data *testing* serta pengujian *neuron* pada *hidden layer* berpengaruh dalam peramalan curah hujan.
2. Metode evaluasi yang digunakan dalam peramalan curah hujan menggunakan metode *Extreme Learning Machine* (ELM) yaitu *Mean Absolute Percentage* (MAPE). Melalui hasil pengujian yang telah dilakukan menghasilkan nilai *error rate* terkecil sebesar 3,6852% dengan menggunakan 2 *neuron* pada *hidden layer*. Parameter yang digunakan pada pengujian peramalan curah hujan menggunakan metode *Extreme Learning Machine* (ELM) yaitu perbandingan data *training* yaitu sebesar 80%, nilai variabel *lag time* (fitur) senanyak 4 fitur dan *neuron* yang digunakan pada *hidden layer* yaitu sebanyak 2 *neuron*.

7.2 Saran

Saran yang digunakan untuk penelitian selanjutnya sebagai berikut.

1. Pemilihan parameter yang digunakan dalam penelitian ini hanya berdasarkan data *time series* (data tahun-tahun sebelumnya), untuk



penelitian selanjutnya dapat menggunakan faktor lain seperti kelembapan udara, suhu dan tekanan udara.

2. Peramalan curah hujan menggunakan metode *Extreme Learning Machine* (*ELM*) dalam penentuan bobot (*weight*) dan bias (*hidden node*) pada penelitian ini masih menggunakan hasil *random* sebagai penentuan nilai bobot dan bias dalam perhitungannya, penitikan selanjutnya dapat melakukan optimasi terhadap nilai bobot dan bias yang akan digunakan untuk mencari nilai *random* bobot dan bias yang optimal dengan *error* terkecil. Optimasi yang dapat digunakan dalam penelitian selanjutnya yaitu *Swarm Intelligence* seperti *Ant Colony*, *Bee Colony* dan *Particle Swarm Optimization*.



DAFTAR REFERENSI

- Agustina, Irwin D., Anggraeni, W., & Mukhlason, A., 2009. Penerapan Metode Extremem Learning Machine Untuk Peramalan Permintaan. Surabaya : Institut Teknologi Sepuluh Novermber.
- Badan Meteorologi Klimatologi dan Geofisika, 2015. *Analisis Hujan November 2015 dan Prakiraan Hujan Januari-Maret 2016*. Malang: Stasiun Klimatologi Karangploso.
- Cao, W., Gao, J., Ming, Z. & Cai, S., 2017. Some Tricks in Parameter Selection for Extreme Learning Machine.
- Cao, J., & Xiong, L., 2014. Protein Sequence Classification with Improved Extreme Learning Machine Algorithms. China.
- Chai, T., & Draxler, R., R., 2014. Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avolding RMSE in the literature. *Geosci. Model Dev.*, 1247-1250.
- Ertuğrul, Ö.F. & Kaya, Y., 2014. A Detailed Analysis on Extreme Learning Machine and Novel Approaches Based on ELM. *American Journal of Computer Science and Engineering*, [online] 1(5), pp.43–50.
- Fardani, D., P., 2015. Sistem Pendukung Keputusan Peramalan Jumlah Kunjungan Pasien Menggunakan Metode Extreme learning Machine (Studi Kasus : Poli Gigi RSU Dr. Wahidin Sudiro Husodo Mojokerto). Vol. 1, no. 1, April 2015. Universitas Airlangga.
- Jimoh, R., G., Olagunju, M., Folorunso, I., O., & Asiribo, M., A., 2013. *Modeling Rainfall Prediction using Fuzzy Logic*. International Journal of Innovative Research in Computer and Communication Engineering, 2320 - 9798.
- Huang, G., B., Zhu, Q., Y., & Siew, C., K., 2006. Extreme Learning Machine : Theory and applications. Elsevier science : Neurocomputing.
- Huang, G., B., Zhu, Q., Y., & Siew, C., K., 2006. Extreme Learning Machine : A New Learning Scheme of Feedforward Neural Networks. Elsevier science : Neurocomputing.
- Khotimah, Bain K., Sari, Eka Mala R., & Yulianarta, H., 2010. Kinerja Metode Extreme Learning Machine (ELM) Pada Sistem Peramalan. *Journal Simantec*, Vol 1, p.186.
- Kusumastuti, D. I. & Sulistiowati., 2015. *Analisa Karakteristik Curah Hujan dan Kurva Instensitas Durasi Frekuensi (IDF) di Provinsi Lampung*. Rekayasa, Volume 14.
- Liang, N., Y.,Huang, G., B., Saratchandran, P., & Sundararajan, N., 2006. A Fast and Accurate Online Sequential Learning Algorithm for Feedforward networks. *IEEE Transactions On Neural Networks*, Vol 17(6), p.1411-1423.

- Nhita, F., A., Wisesty, U. N. & Ummah, I., 2015. *Planting Calender Forecasting System Using Evolving Neural Network*. Far East Journal of Electronics and Communications, Volume 14, pp. 81-92.
- Ogi., 2011. Analisis Variabilitas Curah Hujan Dan Suhu Di Bali. *2016 1st International Conference on Information Technology, Information Systems and Electrical Engineering*.
- Patro, S.G.K., & Sahu, K.K., 2015. *Normalization : A Preprocessing Stage*. Department of CSE & IT, VSSUT, Burla, Odisha, India.
- Rani, K., B., & Govardhan, A., 2013. *Rainfall Prediction using Data Mining Techniques – A Survey*. Department of Computer Science & Engineering, 3903.
- Sun, Z., L, Chai, T., M., Au, K., F., & Yu, Y., 2008. Sales Forecasting Using Extreme Learning Machine With Application In Fashion Retailing. Elsevier Decision Support Systems 46 (2008) 411-419.
- Shamshirband, S., Mohammadi, K., Tog, C., W., Petkovic, D., Porcu, E., Mostafaeipour, A., Ch, S., & Sedaghat, A., 2015. Application of extreme learning machine for estimation of wind speed distribution. Springer-Verlag Berlin Heidelberg.
- Toth, E., Brath, A., & Montanari, A., 2005. *Comparison of Short-Term Rainfall Prediction Models for Real-Time Flood Forecasting*. Journal of Hydrology, 132 – 147.

LAMPIRAN A DATA CURAH HUJAN DASARIAN DAERAH PONCOKUSUMO MALANG TAHUN 2000 SAMPAI 2015

Tahun Dasarian	Jan			Feb			Mar			Apr			May			Jun		
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
2002	69	101	224	148	167	109	145	113	202	112	145	50	0	39	0	0	0	0
2003	259	51	286	89	149	93	170	29	63	141	48	0	101	27	0	0	0	0
2004	161	232	232	68	235	158	97	252	99	10	23	0	55	11	40	23	0	0
2005	83	88	114	56	136	187	204	99	78	106	57	0	0	0	0	0	32	77
2006	279	77	132	89	162	118	122	134	97	56	112	42	67	52	82	6	0	0
2007	10	7	51	94	203	61	46	125	205	59	64	28	0	18	25	11	0	13
2008	149	48	51	80	18	29	103	161	188	36	0	9	48	0	0	6	0	0
2009	75	223	255	57	257	183	113	78	65	169	205	90	13	74	67	0	0	0
2010	148	80	217	219	146	81	114	47	119	168	194	223	81	46	29	29	34	57
2011	55	51	91	164	53	19	105	80	116	172	64	78	95	32	23	0	0	0
2012	214	112	76	114	115	78	210	94	39	33	56	18	28	30	0	0	0	0
2013	59	53	226	193	221	46	172	34	49	63	122	89	0	25	78	193	0	14
2014	251	77	71	73	23	59	18	199	60	22	135	72	19	7	5	0	22	17
2015	15	85	125	123	36	155	61	83	134	74	83	19	35	6	5	6	0	0

**LAMPIRAN A DATA CURAH HUJAN DASARIAN DAERAH PONCOKUSUMO MALANG TAHUN 2000
SAMPAI 2015 (LANJUTAN)**

Tahun	Jul			Aug			Sep			Oct			Nov			Dec		
Dasar	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
2002	0	0	0	0	0	0	0	0	0	0	0	92	78	194	100	261	345	
2003	0	0	0	0	0	0	0	0	0	11	0	40	28	208	180	177	105	198
2004	0	0	0	0	0	0	0	22	0	0	0	59	98	26	241	197	152	244
2005	23	2	0	7	0	0	0	27	0	0	81	42	26	119	59	134	137	194
2006	0	0	0	0	0	0	0	0	0	0	0	22	21	31	94	96	167	
2007	0	0	0	0	0	0	0	0	0	0	0	49	134	0	10	166	157	254
2008	0	0	0	0	0	0	0	0	0	47	51	83	102	181	66	240	164	36
2009	0	0	0	0	0	0	0	0	0	7	0	0	16	66	28	19	0	57
2010	16	53	42	0	0	53	59	115	153	42	7	184	179	60	128	126	119	40
2011	0	0	0	0	0	0	0	0	0	0	0	55	102	124	57	92	85	252
2012	0	0	0	0	0	0	0	0	0	4	38	17	5	113	94	105	98	143
2013	8	28	12	0	0	0	0	0	0	0	27	22	23	51	64	30	279	141
2014	0	24	0	0	0	0	0	0	0	0	27	22	102	124	57	138	97	228
2015	0	0	0	0	0	0	0	0	0	0	0	0	7	10	74	130	123	42

LAMPIRAN B HASIL PENGUJIAN JUMLAH VARIABEL LAG TIME (FITUR)

Jumlah Banyak fitur	MAPE pada percobaan ke-										Rata-rata MAPE
	1	2	3	4	5	6	7	8	9	10	
2	41.2351	42.2123	40.1264	42.5223	41.2548	41.0251	43.0238	42.1187	43.1522	41.1724	41.7843
3	23.9272	25.1281	20.1827	21.0272	23.9021	21.7127	24.6261	22.0652	21.2621	21.2728	22.5106
4	6.1721	3.6852	8.2923	8.1271	4.7562	6.8512	3.8831	8.1245	8.9821	8.6946	6.7568
5	11.8121	12.2672	9.7681	12.7872	8.7826	14.7181	12.2621	8.9903	14.8276	12.8213	11.9037
6	14.2882	14.2821	18.8251	19.3754	17.9872	18.2532	18.8671	10.3881	14.1922	15.7525	16.2211
7	23.4542	22.7731	19.8521	20.3315	22.8573	18.1721	19.1232	19.7217	22.2541	20.7633	20.9303
8	23.8921	24.0282	20.3731	22.5681	25.2204	21.2729	18.7312	22.7322	18.7621	21.6845	21.9265
9	26.2763	25.7613	22.5512	22.6788	24.0524	22.3722	24.3822	22.2273	24.8821	24.8921	24.0076
10	21.4437	21.7218	24.5062	26.2461	24.7402	26.3891	25.2728	26.3712	23.5218	23.0451	24.3258

LAMPIRAN C HASIL PENGUJIAN JUMLAH NEURON PADA HIDDEN LAYER

Jumlah Neuron Pada Hidden Layer	MAPE pada percobaan ke-										Rata-rata MAPE
	1	2	3	4	5	6	7	8	9	10	
2	4.3421	4.1267	3.7633	6.3718	8.5812	4.4021	4.8301	8.4521	6.5601	8.9012	6.0331
3	3.7209	8.6545	6.7612	3.8912	3.9122	8.6701	8.7892	12.5575	5.6208	8.4953	7.1073
4	14.1601	13.0278	11.1205	14.4602	10.1812	15.1081	18.0117	12.7026	13.0125	8.1044	12.9889
5	25.2122	21.0728	22.5101	21.2624	15.0202	18.1223	19.0924	18.0233	19.0875	20.0883	19.9492
6	23.1125	23.1014	25.7812	22.1782	28.5633	22.1282	27.1043	20.5221	22.9163	20.2072	23.5615
7	24.5702	22.0278	28.0632	21.7932	32.1812	23.52382	22.1101	22.4183	22.0237	29.6752	24.8387
8	28.1862	27.2111	33.6251	21.0921	25.5283	23.7523	25.2723	29.0782	29.7571	28.1231	27.1626
9	27.3208	25.0822	29.5212	33.0221	33.5792	27.1245	23.6233	33.0628	25.9881	24.2221	28.2546
10	32.1672	23.1872	25.6987	26.1762	35.1282	32.1282	28.7812	31.6183	30.7434	31.9853	29.7614