



UNIVERSITAS INDONESIA
PEMODELAN DAN PEMBELAJARAN MESIN
LAPORAN UAS JARINGAN SYARAF TIRUAN
DENGAN METODE *BACKPROPAGATION*

ABELTUS REFORMA PUTRA – 1806195236
DWI PAMBAGYO MAHARDIKA – 1806147855
PRAMUDITO ANGGRAITO – 1806148006
TEUKU ALIF RAFI - 1806195223

FAKULTAS TEKNIK
TEKNIK ELEKTRO
DEPOK
JUNI 2021

DAFTAR ISI

DAFTAR ISI	i
DAFTAR GAMBAR.....	iii
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	2
1.3. Tujuan Pembelajaran	2
1.4. Batasan Pembelajaran	2
BAB 2 DASAR TEORI	4
2.1. Jaringan Syaraf Tiruan.....	4
2.2. Metode Pembelajaran <i>Backpropagation</i>	5
2.3. Parameter Metode Pembelajaran <i>Backpropagation</i>	7
2.3.1. Inisialisasi bobot	7
2.3.2. <i>Learning Rate</i>	8
2.3.3. Momentum	8
2.4. Algoritma Metode Pembelajaran <i>Backpropagation</i>	8
2.4.1. Inisialisasi Bobot.....	8
2.4.2. Proses <i>Feedforward</i>	9
2.4.3. Proses <i>Backpropagation of Error</i>	9
BAB 3 PEMBAHASAN	11
3.1. Penjelasan Dataset.....	11
3.2. Hasil Percobaan.....	11
3.2.1. Variasi Metode Inisialisasi Bobot.....	11
3.2.2. Variasi Metode Normalisasi.....	13
3.2.3. Variasi Jumlah Epoch.....	14
3.2.4. Variasi Jumlah Hidden Neuron.....	15
3.2.5. Variasi <i>Learning Rate</i>	16

3.2.6. Variasi Koefisien Momentum.....	17
3.3. Analisa Hasil Percobaan	18
3.3.1. Variasi Metode Inisialisasi Bobot.....	18
3.3.2. Variasi Metode Normalisasi.....	19
3.3.3. Variasi Jumlah Epoch.....	21
3.3.4. Variasi Jumlah Hidden Neuron.....	23
3.3.5. Variasi <i>Learning Rate</i>	25
3.3.6. Variasi Koefisien Momentum.....	27
BAB 4 KESIMPULAN	30
DAFTAR REFERENSI.....	31
LAMPIRAN	32

DAFTAR GAMBAR

Gambar 1 Model jaringan tiruan.....	5
Gambar 2 Arsitektur Backpropagation Secara Horizontal.....	6
Gambar 3 Perbandingan Nilai <i>Error</i> dengan Variasi Metode Inisialisasi	18
Gambar 4 Perbandingan <i>Recognition Rate</i> dengan Variasi Metode Inisialisasi	18
Gambar 5 Perbandingan Nilai <i>Error</i> dengan Variasi Metode Normalisasi	20
Gambar 6 Perbandingan <i>Recognition Rate</i> dengan Variasi Metode Normalisasi	20
Gambar 7 Perbandingan Nilai <i>Error</i> dengan Variasi Jumlah Epoch	22
Gambar 8 Perbandingan <i>Recognition Rate</i> dengan Variasi Jumlah Epoch	22
Gambar 9 Perbandingan Nilai <i>Error</i> dengan Variasi Jumlah <i>Hidden Neuron</i>	24
Gambar 10 Perbandingan <i>Recognition Rate</i> dengan Variasi Jumlah <i>Hidden Neuron</i>	24
Gambar 11 Perbandingan Nilai <i>Error</i> terhadap Variasi <i>Learning Rate</i>	26
Gambar 12 Perbandingan Nilai <i>Recognition Rate</i> terhadap Variasi <i>Learning Rate</i>	26
Gambar 13 Perbandingan Nilai <i>Error</i> terhadap Variasi Koefisien Momentum	28
Gambar 14 Perbandingan Nilai <i>Recognition Rate</i> terhadap Variasi Koefisien Momentum....	28

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi yang terjadi selama revolusi industri 5.0 dapat dibilang sangat progresif dan memiliki keterkaitan dalam berbagai aspek kehidupan seperti bidang ekonomi, sosial, maupun untuk hal-hal seperti psikologis dari manusia itu sendiri terhadap bagaimana teknologi bekerja dan berkembang untuk menyelesaikan masalah-masalah yang ada di sekitar kehidupannya. Dalam menyelesaikan masalah yang ada, dibutuhkan ketelitian dan tentunya perlu memerhatikan banyak faktor untuk menyelesaikan masalah tersebut dikarenakan semakin kompleks juga masalah yang kedepannya akan datang dan variable yang harus diperhitungkan terutama dalam industri otomasi maupun industri yang bekerja dalam lingkup pembelajaran mesin. Salah satu metode yang dapat digunakan untuk menyelesaikan masalah yang ada dengan bantuan dari pembelajaran mesin tersebut adalah jaringan saraf tiruan (bisa dibilang sebagai *Artificial Neural Network* atau disingkat ANN). Jaringan saraf tiruan merupakan bidang awalnya merupakan model matematika yang diadaptasikan dan disederhanakan dari struktur sel syaraf oleh Mc. Culloch dan Pitts pada tahun 1943. Lalu model matematika ini dikembangkan karena model matematika terkadang tidak bisa melakukan pengenalan pola yang adaptif juga otomatis, pengembangan tersebut berlangsung hingga saat ini dimana salah satunya yang umum untuk digunakan di masa ini adalah jaringan saraf tiruan propagasi-balik.

Jaringan saraf tiruan propagasi-balik memiliki keunggulan dimana proses pengenalan ataupun proses dari klasifikasi yang dilakukan secara otomatis dan lebih akurat. Namun, tidak dapat dipungkiri bahwa jaringan saraf tiruan propagasi balik tersebut memiliki kekurangan, yaitu ketidakmampuannya algoritma ini untuk mengenali adanya *outlier* pada masalah yang dikaji. Propagasi balik melakukan klasifikasi data yang digunakan atau diberikan berdasarkan kelas yang ada sehingga jika ada sedikit data yang tidak sesuai dengan kelasnya, maka propagasi balik akan tetap menganalisa data tersebut dan mencari kemiripannya untuk dimasukkan kepada kelas yang ada. Dimana aplikasi dari Jaringan saraf tiruan telah digunakan dalam banyak aplikasi termasuk sistem kontrol jaringan saraf tiruan, pemrosesan sinyal, pengenalan penipuan, pengenalan suara, objek dan wajah, saham, mata uang dan melakukan prediksi nilai mata uang, serta laporan keuangan pemasaran untuk mencegah kebangkrutan.

Salah satu aplikasi yang dapat dilakukan adalah dalam bidang psikologi, dimana aplikasi dari jaringan saraf tiruan propagasi balik dapat memprediksi bagaimana manusia berperilaku ataupun menganalisa perasaan yang dapat dirasakan dari manusia itu sendiri. Pada penelitian

ini, kelompok kami menggunakan dataset *balanced scale* dalam dunia psikologi, dimana *balanced scale* adalah tes atau survei dimana untuk setiap kemungkinan respon terdapat respon yang berarti sebaliknya. Skala penilaian dengan empat alternatif sangat buruk, buruk, baik, dan sangat baik adalah contoh, seperti serangkaian pertanyaan survei di mana setengah dari pertanyaan mencirikan sifat tertentu (misalnya, tingkat stres yang dirasakan) dalam satu arah (misalnya, rendah) dan separuh lainnya mencirikan sifat dalam arah yang berlawanan (misalnya, tinggi). Dengan aplikasi dari jaringan saraf tiruan, kita dapat melakukan analisis bagaimana respon yang akan dilakukan manusia dalam kondisi tertentu, karena jaringan saraf tiruan merupakan sistem yang adaptif dengan menggunakan informasi internal maupun eksternal yang ada terkait dengan data yang sehingga diolah dan memprediksi bagaimana manusia melakukan respon terhadap suatu kasus tertentu.

1.2. Rumusan Masalah

Melihat latar belakang permasalahan tersebut, terdapat beberapa permasalahan yang akan diangkat dalam penulisan laporan ini.

- 1) Bagaimana konsep dasar dan cara kerja dari *Artificial Neural Network* (ANN)?
- 2) Bagaimana cara mengimplementasikan metode *backpropagation* dalam ANN?
- 3) Bagaimana metode ANN bisa mengelompokkan dan mengklasifikan data dalam dataset *Balance Scale*?

1.3. Tujuan Pembelajaran

Pembelajaran mengenai jaringan syaraf tiruan dilakukan supaya mahasiswa mampu untuk:

- 1) Memahami dasar dari konsep jaringan syaraf tiruan.
- 2) Memahami algoritma jaringan syaraf tiruan dengan metode pembelajaran *backpropagation*.
- 3) Mampu mengimplementasikan algoritma yang telah dipelajari untuk mengelompokkan sebuah dataset.

1.4. Batasan Pembelajaran

Pada makalah ini terdapat batasan masalah yang mencakup ruang lingkup dalam suatu permasalahan agar pembahasan yang akan dilakukan fokus dan tidak melenceng dari penelitian. Batasan masalah yang dirancang sebagai berikut :

- 1) Menggunakan metode *artificial neural network* dengan algoritma *backpropagation*

- 2) Menggunakan sumber dataset *balance scale* dengan melakukan data *training* dan data *testing*
- 3) Variabel yang akan dilakukan pada percobaan ini yaitu :
 - a) Inisialisasi bobot (random dan nguyen-widrow)
 - b) Normalisasi (zscore, min-max, dan tanpa normalisasi)
 - c) Variasi data *hidden* neuron
 - d) Variasi data *hidden layer*
 - e) Variasi data *Learning Rate*
 - f) Variasi data momentum

BAB 2

DASAR TEORI

2.1. Jaringan Syaraf Tiruan

Jaringan syaraf tiruan atau biasa disebut dengan *Artificial Neural Network* (ANN) adalah salah satu sistem pemrosesan data informasi yang dirancang dengan mengikuti sistem kerja otak manusia saat mengidentifikasi dan menyelesaikan suatu masalah, yaitu melakukan proses belajar melalui perubahan bobot sinapsisnya. Sehingga sistem ini dapat dilakukan untuk memodelkan hubungan antara *input* dengan *output* yang kompleks dan juga membuat pola-pola sistemnya dari data yang didapatkan.

Kelebihan menggunakan sistem jaringan syaraf tiruan adalah belajar adaptive, dimana dapat mengolah informasi walau belum ada kepastian, dapat mengeneralisasi dan mengekstraksi suatu pola data tertentu. Selain itu, jaringan syaraf tiruan juga mampu self-organisation dimana dapat merancang dan membuat suatu pola berdasarkan data yang diterima selama proses pembelajaran, dan memiliki kemampuan real time operation dimana dapat melakukan perhitungan secara paralel sehingga membuat proses perhitungan menjadi lebih singkat [1].

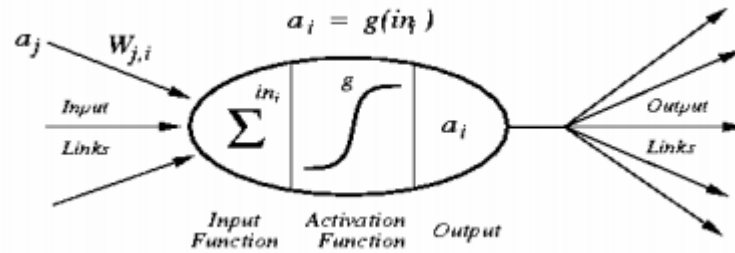
Jaringan saraf tiruan bekerja menggunakan prinsip meniru sistem kerja neuron dengan berdasarkan persamaan matematika yang menggunakan model mendefinisikan fungsi $f : X \rightarrow Y$. Istilah “network” pada ANN (jaringan saraf tiruan) menunjukkan interkoneksi dari beberapa neuron yang berada di lapisan tertentu. Lapisan yang ada di jaringan saraf tiruan dibagi menjadi 3 lapisan, yaitu:

- 1) *Input Layer*. Terdiri dari neuron yang berfungsi untuk mendeteksi data *input* dari variabel X. Neuron pada lapisan ini bisa saling terhubung ke lapisan neuron lainnya, seperti *hidden layer* atau *output layer*
- 2) *Hidden Layer* (Lapisan tersembunyi). Lapisan ini berfungsi untuk menerima data dari *input layer*
- 3) *Output layer*. Lapisan ini berfungsi untuk menerima data dari *hidden layer* atau juga dari *input layer*. Dimana nilai *output* yang dihasilkan merupakan hasil perhitungan matematis nilai X menjadi nilai Y

Secara matematis, fungsi dari jaringan syaraf tiruan adalah :

$$f(x) = K\left(\sum_i w_i g_i(x)\right)$$

Dimana K adalah fungsi aktivasi dan w adalah beban atau *weight data*.



Gambar 1 Model jaringan tiruan

Sistem kerja jaringan syaraf tiruan dimulai dari data *input* yang diterima neuron dan nilai bobot tiap masukan yang ada. Setelah data *input* masuk, maka data tersebut akan melalui tahap penjumlahan (Σ) dan setelah itu data hasil penjumlahan akan dibandingkan dengan suatu nilai ambang. Jika data hasil penjumlahan melebihi nilai ambang, maka aktivitas neuron tidak akan terjadi. Namun jika data hasil penjumlahan di bawah nilai ambang, maka neuron akan teraktivasi. Setelah teraktivasi, neuron akan mengirimkan hasil data berupa data *output* ke semua neuron yang terkait. Dimana hasil perhitungan fungsi aktivasi tersebut digunakan sebagai *input* pada *layer* berikutnya [2].

Metode pembelajaran jaringan syaraf tiruan terdapat 3 tipe, yaitu *Supervised Learning*, *Unsupervised Learning*, dan *Hybrid Learning*. Dimana pada makalah ini yang digunakan adalah metode pembelajaran *backpropagation*.

2.2. Metode Pembelajaran *Backpropagation*

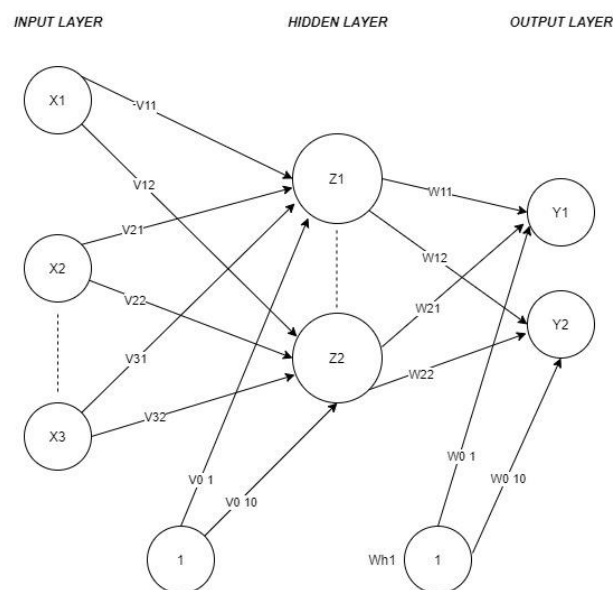
Metode Pembelajaran *Backpropagation* merupakan salah satu jenis dari *Artificial Neural Network* atau Jaringan Saraf Tiruan (JST) yang umum digunakan untuk melakukan fungsi prediksi dalam menyelesaikan suatu masalah yang ada. *Backpropagation* merupakan prosedur pemecahan pembelajaran buat memperkecil taraf *error* menggunakan cara menyesuaikan bobotnya dari disparitas hasil & sasaran yg diinginkan.

Backpropagation juga merupakan sebuah metode sistematis yang dapat digunakan untuk melakukan pembinaan *multilayer* pada Jaringan Saraf Tiruan. *Backpropagation* dapat dikatakan sebagai sebuah prosedur pemecahan untuk pelatihan dari *multilayer*, dimana metode *Backpropagation* mempunyai tiga *layer* utama pada proses pelatihannya, yaitu *input layer*, *hidden layer* & *output layer*, dimana metode *backpropagation* ini adalah merupakan perkembangan menurut *single layer network* (Jaringan Lapisan Tunggal) yang hanya memiliki dua *layer*, yaitu *input layer* dan *output layer*. Pada, *Backpropagation*, terdapat *hidden layer* yang dapat mengakibatkan besarnya nilai *error* yang terjadi menjadi lebih minimum dibanding nilai *error* dalam *single layer network*. Hal itu dikarenakan *hidden layer* dalam

backpropagation berfungsi untuk menjadi node yang dapat melakukan *update* dan penyesuaian nilai dari bobot, oleh karena itu dihasilkan nilai bobot baru yang dapat digunakan sebagai prediksi untuk mendekati target dari nilai ataupun *output* yang diinginkan.

Keuntungan dari metode pembelajaran *backpropagation* diantaranya adalah metode ini menggunakan lebih sedikit memori daripada algoritma lain, dan dapat menghasilkan hasil dengan tingkat kesalahan yang dapat diterima pada kecepatan pemrosesan yang cukup tinggi. Metode ini dipilih karena dapat mengenali pola *input* yang tidak lengkap atau salah, dan mengenali kestabilan pemanggilan pola gambar setiap kali dikembalikan. Dengan menggunakan metode ini, jaringan yang dirancang tidak harus memiliki koneksi khusus untuk melakukan perhitungan yang terbalik dari satu lapisan ke lapisan sebelumnya. Namun, kesalahan pada *output layer* akan dibawa kembali ke *input layer*.

Backpropagation merupakan salah satu metode pembelajaran *Supervised Learning*, dimana *input layer* menerima pola *input* dan melakukan proses perhitungan sesuai dengan bobot awal yang diperoleh secara acak. Jika *output* yang dihasilkan jaringan berbeda dari nilai yang diharapkan, jaringan akan membuat perubahan pada nilai bobot yang ada. Proses tersebut akan mengulang terus hingga *output* yang dihasilkan sesuai dengan target yang diharapkan. Proses tersebut membutuhkan waktu yang lama untuk mencapai nilai ini. Oleh karena itu, proses pembelajaran menjadi terbatas, dan pembelajaran dihentikan jika selisih antara nilai keluaran dan nilai target lebih kecil dari nilai toleransi (tingkat kesalahan atau bisa disebut *error rate*). Besar kecilnya penyesuaian bobot pada tiap siklus pembelajaran ditentukan dari *Learning Rate*. Arsitektur dari metode pembelajaran *backpropagation* dapat dilihat pada gambar dibawah ini.



Gambar 2 Arsitektur *Backpropagation* Secara Horizontal

Pada gambar diatas, dapat dilihat bahwa *input layer* (memiliki simbol ' X_n ') tidak terjadi proses komputasi, namun terjadi pengiriman sinyal *input* dari simbol ' X_1, X_2 dan X_3 ' ke *hidden layer*. Pada *hidden layer* yang dinotasikan dengan simbol Z_n , terdapat nilai bobot (V_{ij}) dan bias (V_{oj}) dimana bobot bias tersebut bertindak sebagai konstanta. Pada *hidden* dan *output layer* terjadi proses komputasi terhadap nilai bobot dan nilai bias dan dihitung pula besarnya *output* dari *hidden layer* dan *output layer* berdasarkan fungsi aktivasi yang digunakan. Dalam algoritma *backpropagation* ini digunakan fungsi aktivasi 'sigmoid biner', hal ini terlihat dari *output* yang diharapkan yang bernilai di antara '0' sampai dengan '1'. Pada *output layer* (dinotasikan dengan simbol Y), data *output* yang dihasilkan dapat dilihat dari variabel Y_n terdapat nilai bobot (W_{ij}) dan nilai bias (W_{oj}).

Aturan dalam melakukan pembelajaran *Backpropagation* disesuaikan dari *delta rule* dengan menambahkan *hidden layer*. Terdapat tiga tahapan utama yang dilakukan dalam metode *Backpropagation*, diantaranya: *feedforward* pola pelatihan masukan, *Backpropagation* terhadap *error*, dan penyesuaian bobot. Tiga tahapan tersebut dibagi dalam dua proses utama, yaitu Pembelajaran (*training*) dan Uji Coba (*testing*). Seperti pada poin penjelasan dibawah ini:

1) Pembelajaran (*training*)

- a) Proses pengolahan data *input* (*feedforward*)
- b) Perhitungan *error* (*backpropagation*)
- c) Perbaharui bobot

2) Uji coba (*testing*)

- a) Pengelompokkan nilai *output* (*Quantizing*)
- b) Perhitungan *Recognition Rate*

2.3. Parameter Metode Pembelajaran *Backpropagation*

Berikut adalah beberapa parameter yang digunakan dalam metode *backpropagation*:

2.3.1. Inisialisasi bobot

Penentuan awal bobot digunakan di tahap awal untuk dapat menghubungkan antar *layer* ANN *backpropagation*. Bobot awal merupakan bobot yang menentukan apakah ANN akan mencapai titik minimum global atau titik minimum local dari fungsi kesalahan.

1) Random

Pada metode random, bobot awal ditentukan secara acak.

2) Nguyen-Widrow

Pada metode Nguyen-Widrow, bobot awal dimodifikasi terlebih dahulu agar bisa mempercepat proses pembelajaran. Algoritmanya adalah sebagai berikut:

- Inisialisasi dengan bilangan acak antara -0,5 sampai 0,5 untuk bias dan bobot antara neuron di lapisan tersembunyi dengan neuron di lapisan keluaran.
- Untuk inisialisasi bias dan bobot antara neuron di lapisan masukan dengan lapisan tersembunyi:
 - Menentukan faktor skala $\beta = 0,7P^{\frac{1}{N}}$, dengan P adalah ukuran lapisan tersembunyi dan N adalah ukuran lapisan masukan.
 - Menginisialisasi bobot v_{ij} dengan bilangan acak antara -0,5 sampai 0,5.
 - Menghitung norma dari vektor bobot dengan rumus:

$$\|v_j\| = \sqrt{\sum_{i=1}^P v_{ij}^2} \quad \text{dan} \quad \|w_k\| = \sqrt{\sum_{j=1}^J w_{jk}^2}$$

- Menyesuaikan nilai bobot dengan rumus:

$$v_{ij} = \frac{\beta v_{ij}}{\|v_j\|} \quad \text{dan} \quad w_{jk} = \frac{\beta w_{jk}}{\|w_k\|}$$

- Menginisialisasi bias dengan bilangan acak antara -0,5 sampai 0,5.

2.3.2. Learning Rate

Learning Rate yang terlalu besar dapat mengakibatkan ketidakstabilan dalam metode *backpropagation*. Nilai laju pembelajaran yang terlalu kecil akan membuat proses konvergensi jaringan menjadi lambat.

2.3.3. Momentum

Momentum dipakai untuk mempercepat proses pembelajaran dengan cara menambahkan arah dari penyesuaian bobot iterasi sebelumnya. Untuk mengatur seberapa besar pengaruh arah dari penyesuaian bobot iterasi sebelumnya, digunakan koefisien momentum (μ).

$$\Delta w_{jk}(t) = \alpha \delta_k z_j + \mu \Delta w_{jk}(t-1)$$

2.4. Algoritma Metode Pembelajaran *Backpropagation*

2.4.1. Inisialisasi Bobot

1) Random

2) Nguyen-Widrow

2.4.2. Proses *Feedforward*

1) Komputasi *input layer*. Untuk setiap *input* x_i , $i = 1, 2, \dots, n$:

- Menerima masukan x_i
- Mengirimkannya ke semua unit *hidden layer*

2) Komputasi *hidden layer*. Untuk setiap unit *hidden* z_j , $j = 1, 2, \dots, p$:

- Menghitung semua sinyal masukan dengan bobotnya

$$z_in_j = v_{oj} + \sum x_i v_{ij}$$

- Menghitung nilai aktivasi setiap unit *hidden* sebagai keluarannya

$$z_j = f(z_in_j)$$

- Mengirim nilai aktivasi sebagai masukan dari *output layer*

3) Komputasi *output layer*. Untuk setiap *output* y_k , $k = 1, 2, \dots, m$:

- Menghitung semua sinyal masukan dengan bobotnya

$$y_in_k = w_{ok} + \sum z_j w_{jk}$$

- Menghitung nilai aktivasi setiap unit *output* sebagai keluaran jaringan

$$y_k = f(y_in_k)$$

- Kuantisasi nilai keluaran

Bertujuan untuk menentukan threshold dari pembelajaran supaya keluaran jaringan bernilai 0 atau 1 jika terdapat keluaran dari neuron yang susah untuk mencapai nilai tersebut.

$$y_k = \begin{cases} 0 & 0 \leq y_k \leq 0,3; y_k = 0 \\ 1 & 0,7 \leq y_k \leq 1; y_k = 1 \\ y_k & \text{lainnya}; y_k = y_k \end{cases}$$

2.4.3. Proses *Backpropagation of Error*

1) Komputasi di *output layer*. Untuk setiap unit *output* y_k , $k = 1, 2, \dots, m$:

- Menerima pola target yang bersesuaian dengan pola masukan
- Menghitung informasi *error*

$$\delta_k = (t_k - y_k) * f'(y_in_k)$$

- Menghitung besar koreksi bobot unit *output*

$$\Delta w_{jk} = \alpha \frac{\partial E(w_{jk})}{\partial w_{jk}} = \alpha \delta_k z_j$$

- Menghitung besar koreksi bias unit *output*

$$\Delta w_{ok} = \alpha \delta_k$$

- Mengirim δ_k ke unit *hidden*

2) Komputasi di *hidden layer*. Untuk setiap unit *hidden* $z_j, j = 1, 2, \dots, p$:

- Menghitung semua koreksi *error*

$$\delta_{in_j} = \sum \delta_k w_{jk}$$

- Menghitung nilai aktivasi koreksi *error*

$$\delta_j = \delta_{in_j} * f'(z_{in_j})$$

- Menghitung koreksi bobot unit *hidden*

$$\Delta v_{ij} = \alpha \delta_j x_i$$

- Menghitung koreksi bias unit *hidden*

$$\Delta v_{oj} = \alpha \delta_j$$

3) Memperbarui Bobot dan Bias

- Untuk setiap unit *output* $y_k, k = 1, 2, \dots, m$:

➤ Update bobot

$$w_{jk}(\text{baru}) = w_{jk}(\text{lama}) + \Delta w_{jk}(\text{baru}) + \mu \Delta w_{jk}(\text{lama})$$

➤ Update bias

$$w_{ok}(\text{baru}) = w_{ok}(\text{lama}) + \Delta w_{ok}(\text{baru}) + \mu \Delta w_{ok}(\text{lama})$$

- Untuk setiap unit *hidden* $z_j, j = 1, 2, \dots, p$:

➤ Update bobot

$$v_{ij}(\text{baru}) = v_{ij}(\text{lama}) + \Delta v_{ij}(\text{baru}) + \mu \Delta v_{ij}(\text{lama})$$

➤ Update bias

$$v_{oj}(\text{baru}) = v_{oj}(\text{lama}) + \Delta v_{oj}(\text{baru}) + \mu \Delta v_{oj}(\text{lama})$$

4) Perhitungan *Error*

Menggunakan fungsi *error* kuadratis

$$E = \frac{1}{2} \sum (t_k - y_k)^2$$

BAB 3

PEMBAHASAN

3.1. Penjelasan Dataset

Pada penelitian ini, kelompok kami menggunakan dataset *balanced scale* dalam dunia psikologi, dimana *balanced scale* adalah tes atau survei dimana untuk setiap kemungkinan respon terdapat respon yang berarti sebaliknya. Skala penilaian dengan empat alternatif sangat buruk, buruk, baik, dan sangat baik adalah contoh, seperti serangkaian pertanyaan survei di mana setengah dari pertanyaan mencirikan sifat tertentu (misalnya, tingkat stres yang dirasakan) dalam satu arah (misalnya, rendah) dan separuh lainnya mencirikan sifat dalam arah yang berlawanan (misalnya, tinggi). Dengan aplikasi dari jaringan saraf tiruan, kita dapat melakukan analisis bagaimana respon yang akan dilakukan manusia dalam kondisi tertentu, karena jaringan saraf tiruan merupakan sistem yang adaptif dengan menggunakan informasi internal maupun eksternal yang ada terkait dengan data yang sehingga diolah dan memprediksi bagaimana manusia melakukan respon terhadap suatu kasus tertentu.

3.2. Hasil Percobaan

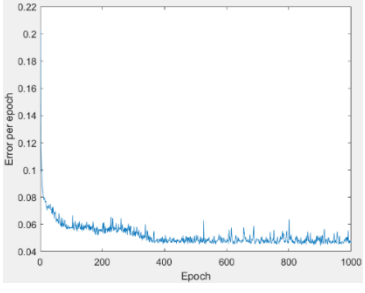
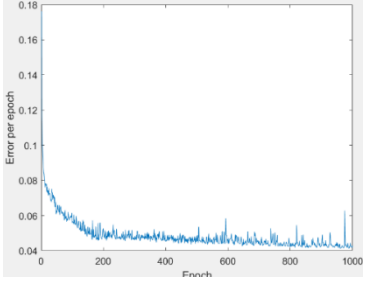
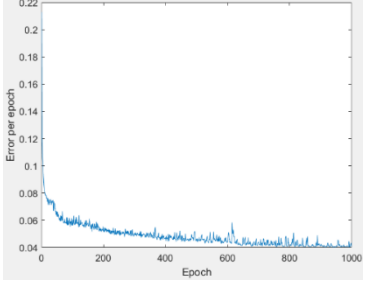
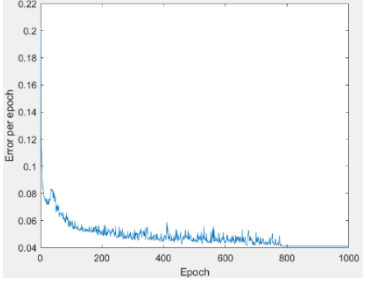
Melalui program jaringan syaraf tiruan dengan metode backpropagation yang diterapkan kepada data *Balanced Scale* didapatkan beberapa hasil pembelajaran. Hasil pembelajaran didapatkan dengan cara memvariasikan beberapa variabel percobaan, seperti inisialisasi bobot, normalisasi, jumlah epoch, jumlah neuron, *Learning Rate*, dan koefisien momentum.

3.2.1. Variasi Metode Inisialisasi Bobot

Metode inisialisasi bobot yang digunakan dalam percobaan ini adalah metode Nguyen-Widrow dan metode Random. Variabel percobaan selain inisialisasi bobot dibuat tetap sebagai berikut:

Normalisasi	Jumlah Epoch	Jumlah Neuron	<i>Learning Rate</i> (α)	Koefisien Momentum (μ)
Tanpa normalisasi	1000	9	0.2	0.5

Hasil percobaan variasi metode inisialisasi bobot adalah sebagai berikut:

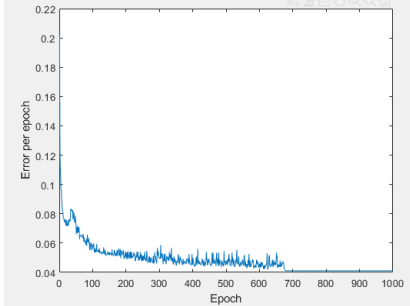
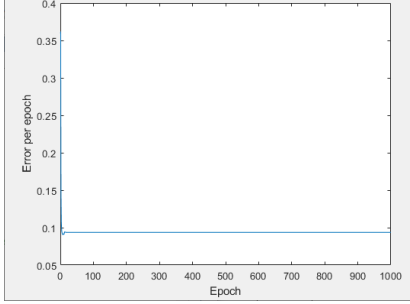
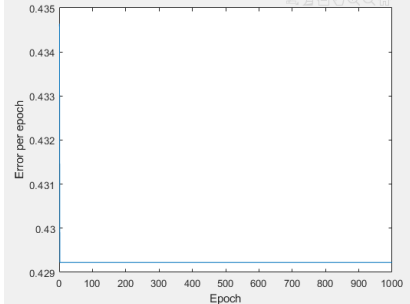
Metode Inisialisasi		Error		Recognition Rate (%)		Grafik Error per Epoch
Input ke Hidden	Hidden ke Output	Trainin g	Testin g	Trainin g	Testing	
Nguyen - Widrow	Nguyen - Widrow	0.0462	0.0655	91.5525	89.304 8	
Nguyen - Widrow	Random	0.0429	0.0622	91.7808	90.374 3	
Random	Nguyen - Widrow	0.0412	0.0661	92.0091	90.374 3	
Random	Random	0.0411	0.0665	91.7808	89.839 6	

3.2.2. Variasi Metode Normalisasi

Metode normalisasi yang digunakan dalam percobaan ini adalah tanpa normalisasi, metode zscore, dan metode min-max. Variabel percobaan selain normalisasi dibuat tetap sebagai berikut:

Inisialisasi Bobot		Jumlah Epoch	Jumlah Neuron	Learning Rate (α)	Koefisien Momentum (μ)
Input ke Hidden	Hidden ke Output				
Random	Random	1000	9	0.2	0.5

Hasil percobaan variasi metode normalisasi adalah sebagai berikut:

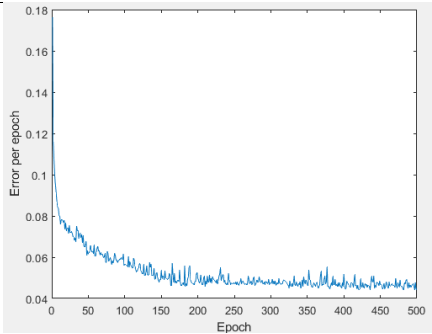
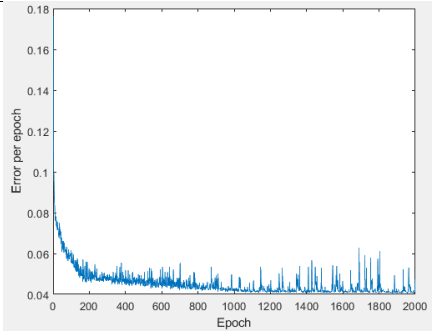
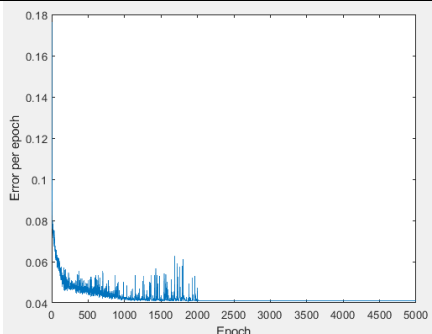
Metode Normalisasi	Error		Recognition Rate (%)		Grafik Error per Epoch
	Training	Testing	Training	Testing	
Tanpa Normalisasi	0.041096	0.067183	91.7808	89.8396	
Zscore	0.093607	0.38503	87.4429	57.2193	
Min-Max	0.42922	0.79679	57.0776	20.3209	

3.2.3. Variasi Jumlah Epoch

Variabel percobaan selain jumlah epoch dibuat tetap sebagai berikut:

Inisialisasi Bobot		Normalisasi	Jumlah Neuron	Learning Rate (α)	Koefisien Momentum (μ)
Input ke Hidden	Hidden ke Output				
Nguyen-Widrow	Random	Tanpa Normalisasi	9	0.2	0.5

Hasil percobaan variasi jumlah epoch adalah sebagai berikut:

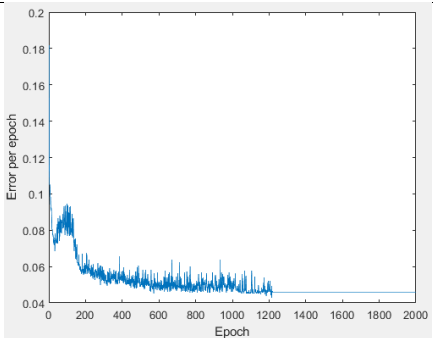
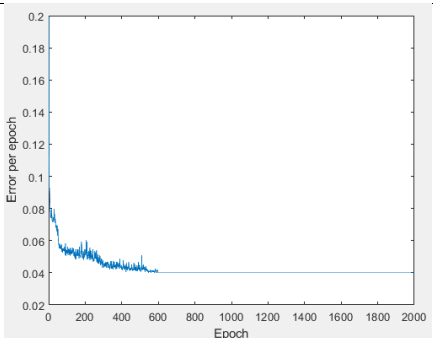
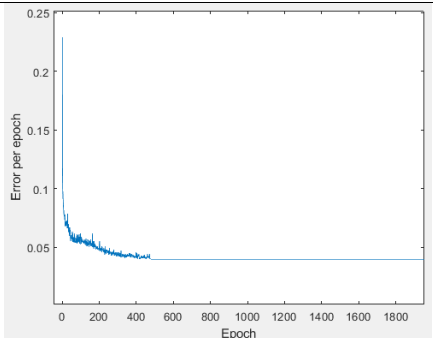
Jumlah Epoch	Error Akhir		Recognition Rate (%)		Grafik Error per Epoch
	Training	Testing	Training	Testing	
500	0.044497	0.28669	91.7808	78.6096	
2000	0.042295	0.2861	91.7808	78.6096	
5000	0.041096	0.28054	91.7808	78.6096	

3.2.4. Variasi Jumlah Hidden Neuron

Variabel percobaan selain jumlah *hidden* neuron dibuat tetap sebagai berikut:

Inisialisasi Bobot		Normalisasi	Jumlah Epoch	<i>Learning Rate</i> (α)	Koefisien Momentum (μ)
<i>Input</i> ke Hidden	Hidden ke <i>Output</i>				
Nguyen-Widrow	Random	Tanpa Normalisasi	2000	0.2	0.5

Hasil percobaan variasi jumlah *hidden* neuron adalah sebagai berikut:

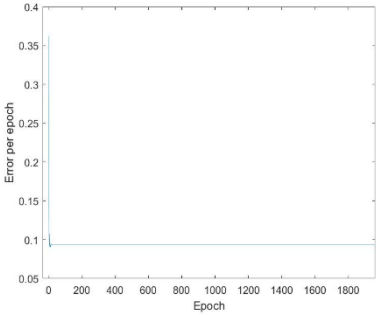
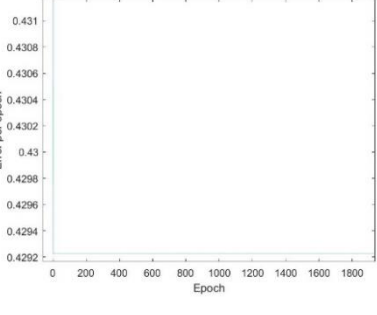
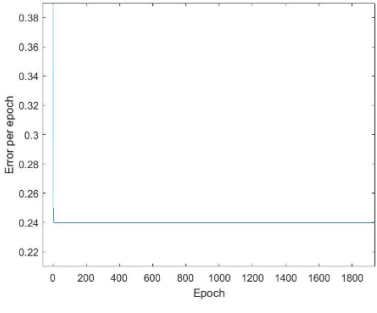
Jumlah Hidden Neuron	<i>Error Akhir</i>		<i>Recognition Rate</i> (%)		Grafik <i>Error per Epoch</i>
	<i>Training</i>	<i>Testing</i>	<i>Training</i>	<i>Testing</i>	
7	0.045662	0.29445	91.3242	73.262	
10	0.039954	0.25282	92.0091	79.6791	
13	0.039954	0.26938	92.0091	79.1444	

3.2.5. Variasi *Learning Rate*

Variabel percobaan selain *Learning Rate* dibuat tetap sebagai berikut:

Inisialisasi Bobot		Normalisasi	Jumlah Neuron	Jumlah Epoch	Koefisien Momentum (μ)
<i>Input ke Hidden</i>	<i>Hidden ke Output</i>				
Random	Random	Z-score	9	2000	0.5

Hasil percobaan variasi *Learning Rate* adalah sebagai berikut:

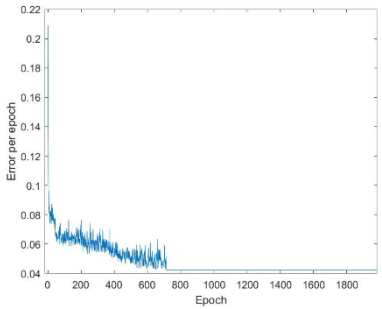
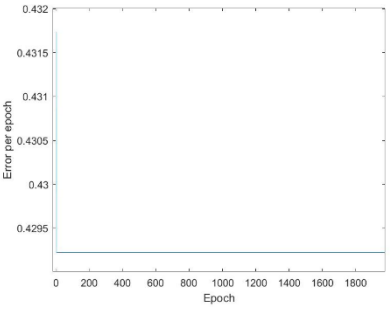
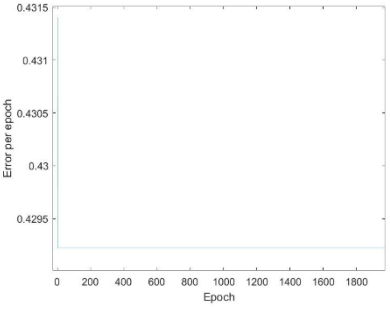
<i>Learning Rate</i> (α)	<i>Error</i>		<i>Recognition Rate</i> (%)		<i>Error per Epoch</i>
	<i>Training</i>	<i>Testing</i>	<i>Training</i>	<i>Testing</i>	
0.2	0.093607	0.38503	87.4429	57.2193	
0.4	0.42922	0.79679	57.0776	20.3209	
0.6	0.23973	0.55375	87.4429	57.2193	

3.2.6. Variasi Koefisien Momentum

Variabel percobaan selain koefisien momentum dibuat tetap sebagai berikut:

Inisialisasi Bobot		Normalisasi	Jumlah Hidden Neuron	Jumlah Epoch	Learning Rate (α)
Input ke Hidden	Hidden ke Output				
Random	Random	Tanpa Normalisasi	9	2000	0.4

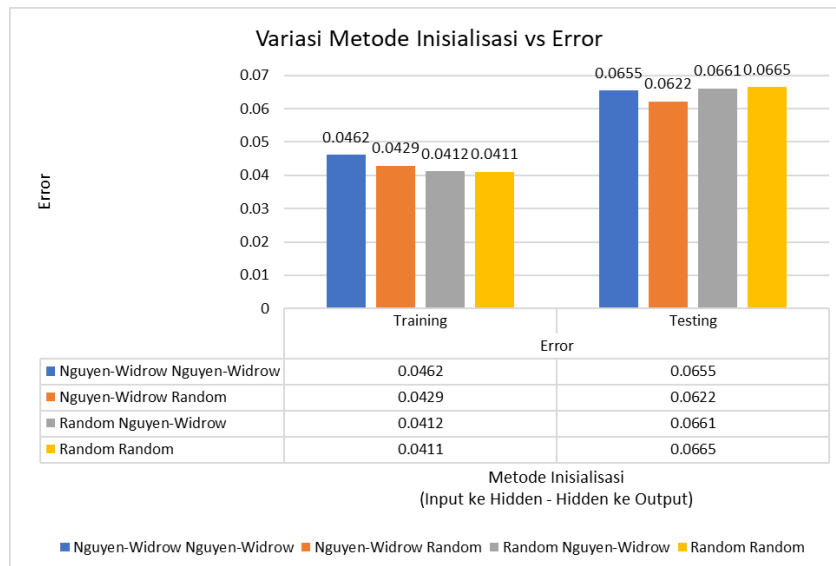
Hasil percobaan variasi koefisien momentum adalah sebagai berikut:

Koefisien Momentum (μ)	Error		Recognition Rate (%)		Error per Epoch
	Training	Testing	Training	Testing	
0.3	0.042237	0.28363	92.0091	75.4011	
0.5	0.42922	0.69323	57.0776	20.3209	
0.7	0.42922	0.79679	57.0776	20.3209	

3.3. Analisa Hasil Percobaan

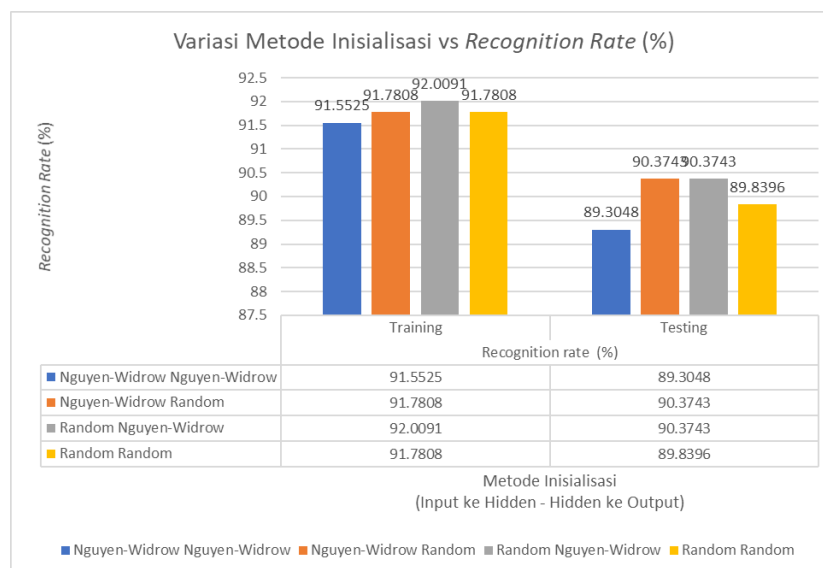
3.3.1. Variasi Metode Inisialisasi Bobot

Berikut ini merupakan grafik perbandingan *error training* dan *testing* serta grafik perbandingan *recognition Rate training* dan *testing* untuk setiap pasangan metode inisialisasi bobot yang digunakan pada percobaan.



Gambar 3 Perbandingan Nilai *Error* dengan Variasi Metode Inisialisasi

Dari grafik *error* ini terlihat bahwa besar *error* tersebut memiliki nilai yang saling mendekati dimana *error training* berada di sekitar 0.04 dan *error testing* berada di sekitar 0.06 untuk setiap pasangan metode inisialisasi bobot. Namun bisa dilihat bahwa *error* paling kecil adalah untuk pasangan metode inisialisasi Nguyen-Widrow (*input ke hidden*) dan Random (*hidden ke output*).



Gambar 4 Perbandingan *Recognition Rate* dengan Variasi Metode Inisialisasi

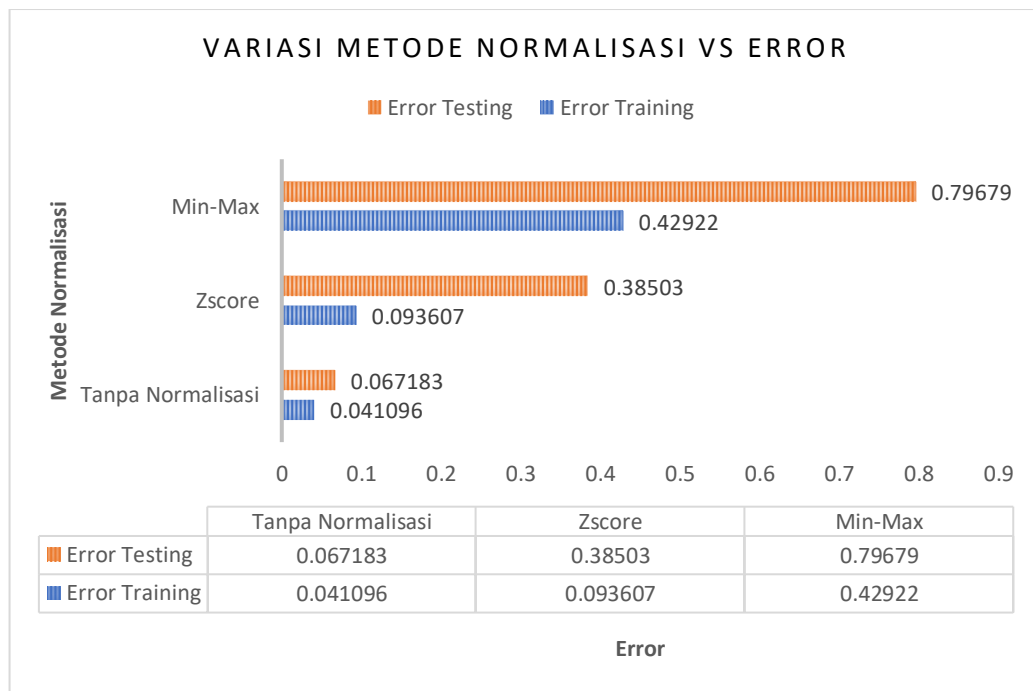
Dari grafik *recognition Rate* ini terlihat bahwa besar ketetapan tersebut memiliki nilai yang saling mendekati dimana *recognition Rate training* berada di sekitar 91% dan *recognition Rate testing* berada di sekitar 90% untuk setiap pasangan metode inisialisasi bobot. Namun bisa dilihat bahwa *recognition Rate* paling besar adalah untuk pasangan metode inisialisasi Nguyen-Widrow (*input* ke *hidden*) dan Random (*hidden* ke *output*). Pada grafik *error* per epoch di bagian 3.2.1. terlihat bahwa konvergensi paling cepat dicapai untuk pasangan metode inisialisasi random – random. Hal ini terlihat juga bahwa *error*nya paling kecil untuk *training*nya. Namun grafik *error* per epoch tersebut hanya menggambarkan untuk *training*nya sehingga sesuai dengan hasil *error* akhir *training*nya adalah yang paling kecil.

3.3.2. Variasi Metode Normalisasi

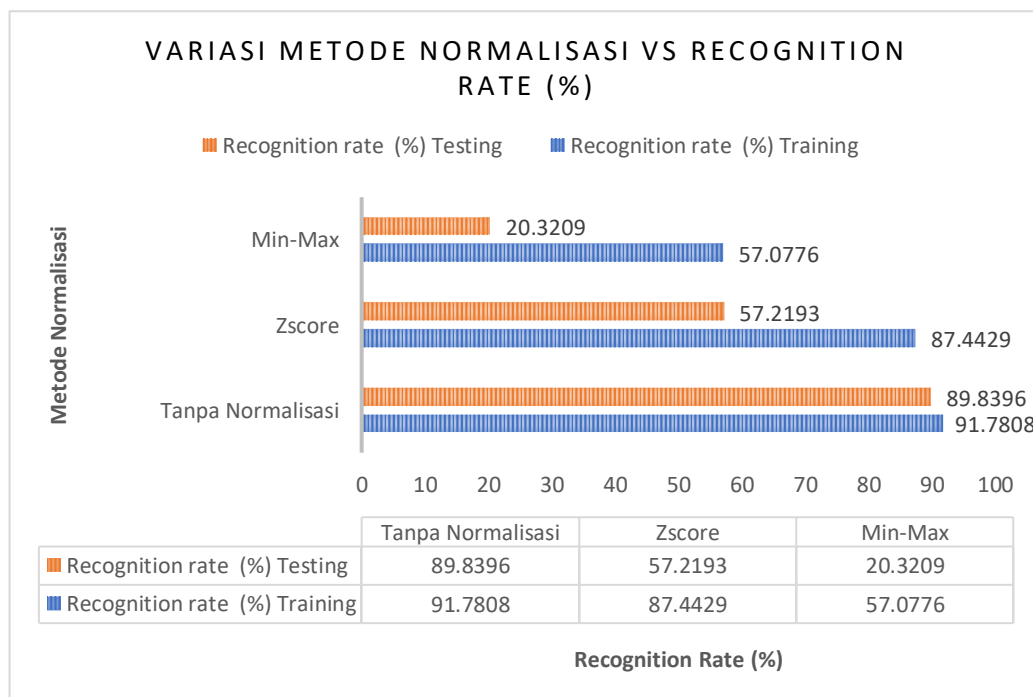
Pada percobaan ini, Jaringan Saraf Tiruan divariasikan berdasarkan metode normalisasi data yang digunakan, diantaranya: tanpa normalisasi, Zscore, dan Min-Max. Dimana pada percobaan kali ini, digunakan beberapa parameter tetap, yaitu:

- 1) Metode Inisialisasi Bobot yang digunakan:
 - a) *Input layer* ke *hidden layer*: Random
 - b) *hidden layer* ke *output layer*: Random
- 2) Jumlah Epoch Maksimum: 1000
- 3) Jumlah Neuron / *hidden layer*: 9
- 4) *Learning Rate* (α): 0.2
- 5) Koefisien Momentum (μ): 0.5

Berikut adalah hasil dari nilai *error* dan nilai *recognition Rate* berdasarkan variasi metode normalisasi yang digunakan melalui pengolahan data pada excel, dilihat di gambar 5 dan gambar 6.



Gambar 5 Perbandingan Nilai *Error* dengan Variasi Metode Normalisasi



Gambar 6 Perbandingan *Recognition Rate* dengan Variasi Metode Normalisasi

Dari grafik yang dihasilkan melalui pengolahan data pada excel, didapatkan keterangan bahwa untuk parameter-parameter yang sudah ditetapkan sebelumnya, nilai *error* dan *recognition Rate* paling baik terdapat pada metode tanpa menggunakan normalisasi, dengan nilai *error* sekitar 0,04 dan nilai dari *recognition Rate* sekitar 90% untuk data hasil olahan dalam proses *training* maupun *testing*. Dari kondisi ini kita dapat menyimpulkan bahwa data *balanced scale* memiliki akurasi lebih baik untuk diprediksi

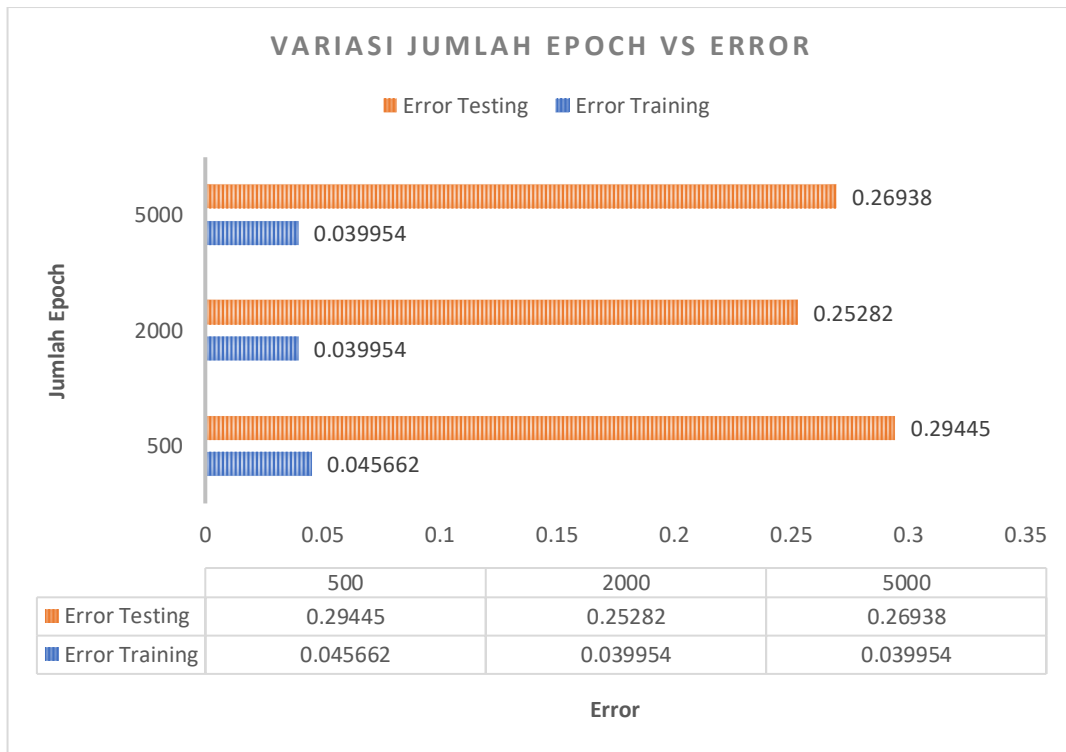
apabila data yang diolah tidak perlu dinormalisasi, dikarenakan data real yang tercantum pada dataset sudah memiliki tingkat akurasi yang baik sehingga prediksi dari nilai kedepannya akan sangat mendekati nilai real yang sudah ada.

3.3.3. Variasi Jumlah Epoch

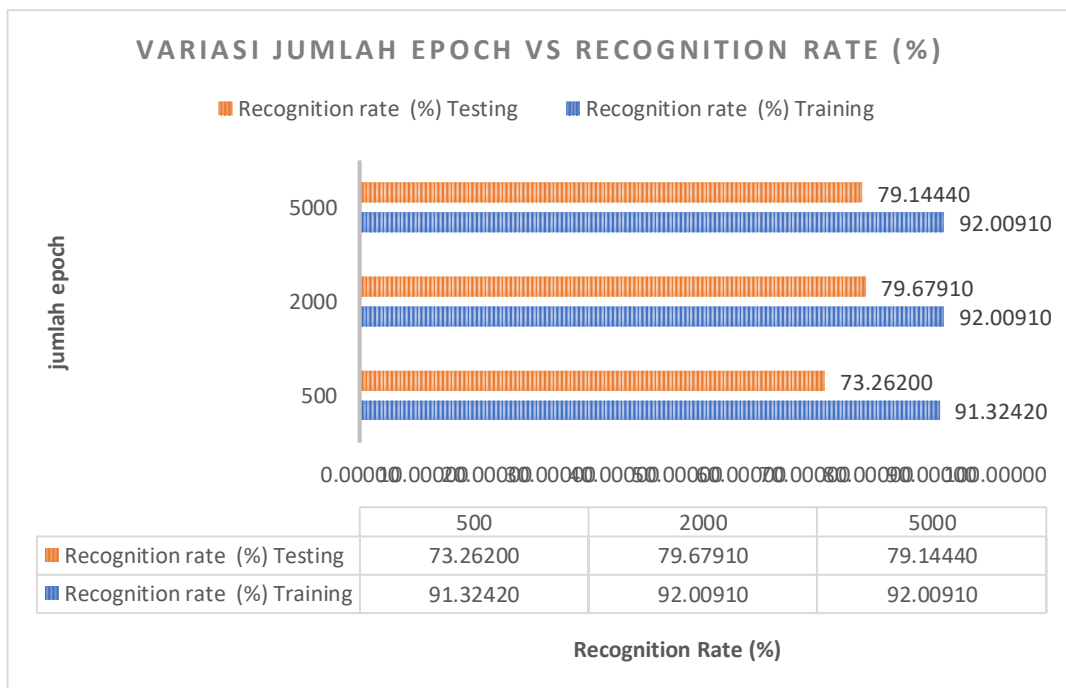
Pada percobaan ini, data Jaringan Saraf Tiruan akan divariasikan berdasarkan jumlah epoch data yang digunakan untuk mengetahui pengaruh jumlah epoch terhadap percobaan yang dilakukan, dimana jumlah epoch yang divariasikan pada percobaan sebesar: 500, 2000, dan 5000. Pada percobaan kali ini, digunakan beberapa parameter tetap, yaitu:

- 1) Metode Inisialisasi Bobot yang digunakan:
 - a) *Input layer* ke *hidden layer*: Nguyen-Widrow
 - b) *hidden layer* ke *output layer*: Random
- 2) Metode normalisasi: Tanpa normalisasi
- 3) Jumlah Neuron / *hidden layer*: 9
- 4) *Learning Rate* (α): 0.2
- 5) Koefisien Momentum (μ): 0.5

Berikut adalah hasil dari nilai *error* dan nilai *recognition Rate* berdasarkan variasi jumlah epoch yang digunakan melalui pengolahan data pada excel, dilihat di gambar 7 dan gambar 8.



Gambar 7 Perbandingan Nilai *Error* dengan Variasi Jumlah Epoch



Gambar 8 Perbandingan *Recognition Rate* dengan Variasi Jumlah Epoch

Berdasarkan dari grafik yang dihasilkan melalui pengolahan data pada excel, didapatkan bahwa nilai *error* dan *recognition Rate* mengalami nilai fluktuatif untuk variasi jumlah epoch yang berbeda. Hal ini disebabkan karena terjadinya *overfitting* pada variasi jumlah epoch. Untuk nilai *error* dan *recognition Rate* paling baik terdapat pada jumlah epoch sebesar 5000, dengan nilai *error* 0.041096 dan nilai dari *recognition Rate*

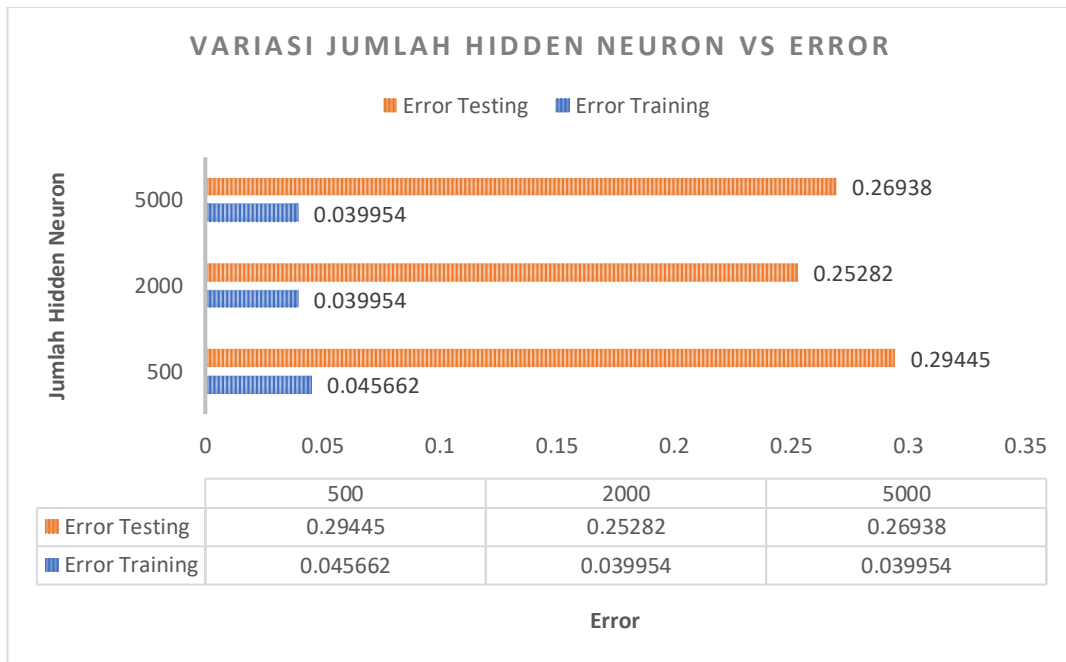
91.7808% untuk data hasil olahan dalam proses *training* maupun *testing*. Dari kondisi ini kita dapat menyimpulkan bahwa data *balanced scale* memiliki akurasi lebih baik untuk diprediksi apabila data yang diolah menggunakan jumlah epoch yang semakin besar, dimana pada percobaan ini jumlah epoch yang lebih baik digunakan sebesar 5000. Sehingga banyaknya epoch akan cenderung linear dengan banyaknya variance dalam dataset.

3.3.4. Variasi Jumlah Hidden Neuron

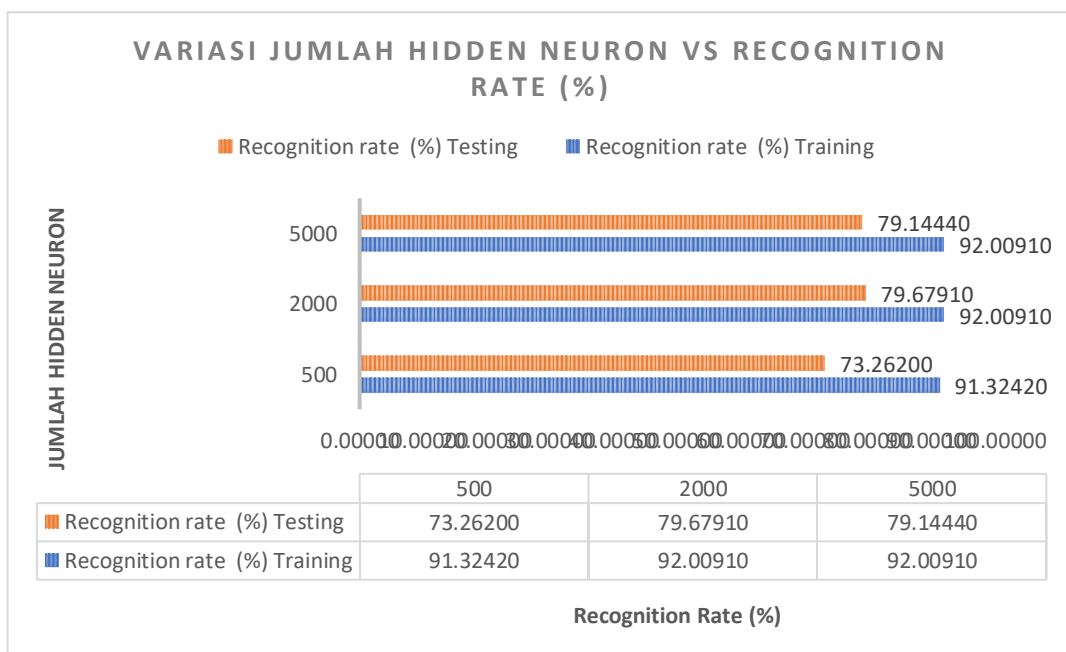
Pada percobaan ini, data Jaringan Saraf Tiruan akan divariasikan berdasarkan jumlah *hidden* neuron data yang digunakan untuk mengetahui pengaruh jumlah *hidden* neuron terhadap percobaan yang dilakukan, dimana jumlah *hidden* neuron yang divariasikan pada percobaan sebesar: 7, 10, dan 15. Pada percobaan kali ini, digunakan beberapa parameter tetap, yaitu:

- 1) Metode Inisialisasi Bobot yang digunakan:
 - a) *Input layer* ke *hidden layer*: Nguyen-Widrow
 - b) *hidden layer* ke *output layer*: Random
- 2) Metode normalisasi: Tanpa normalisasi
- 3) Jumlah epoch : 2000
- 4) *Learning Rate* (α): 0.2
- 5) Koefisien Momentum (μ): 0.5

Berikut adalah hasil dari nilai *error* dan nilai *recognition Rate* berdasarkan variasi jumlah *hidden* neuron yang digunakan melalui pengolahan data pada excel, dilihat di gambar 9 dan gambar 10.



Gambar 9 Perbandingan Nilai *Error* dengan Variasi Jumlah *Hidden Neuron*



Gambar 10 Perbandingan *Recognition Rate* dengan Variasi Jumlah *Hidden Neuron*

Berdasarkan dari grafik yang dihasilkan melalui pengolahan data pada excel, didapatkan bahwa nilai *error* dan *recognition Rate* mengalami nilai fluktuatif untuk variasi jumlah *hidden neuron* yang berbeda. Hal ini disebabkan karena terjadinya *overfitting* pada variasi jumlah *hidden neuron*. Untuk nilai *error* dan *recognition Rate* paling baik terdapat pada jumlah *hidden neuron* sebesar 13, dengan nilai *error* 0.26938 dan nilai dari *recognition Rate* 79.1444% untuk data hasil olahan dalam proses *testing*. Dari kondisi ini kita dapat menyimpulkan bahwa data *balanced scale* memiliki akurasi

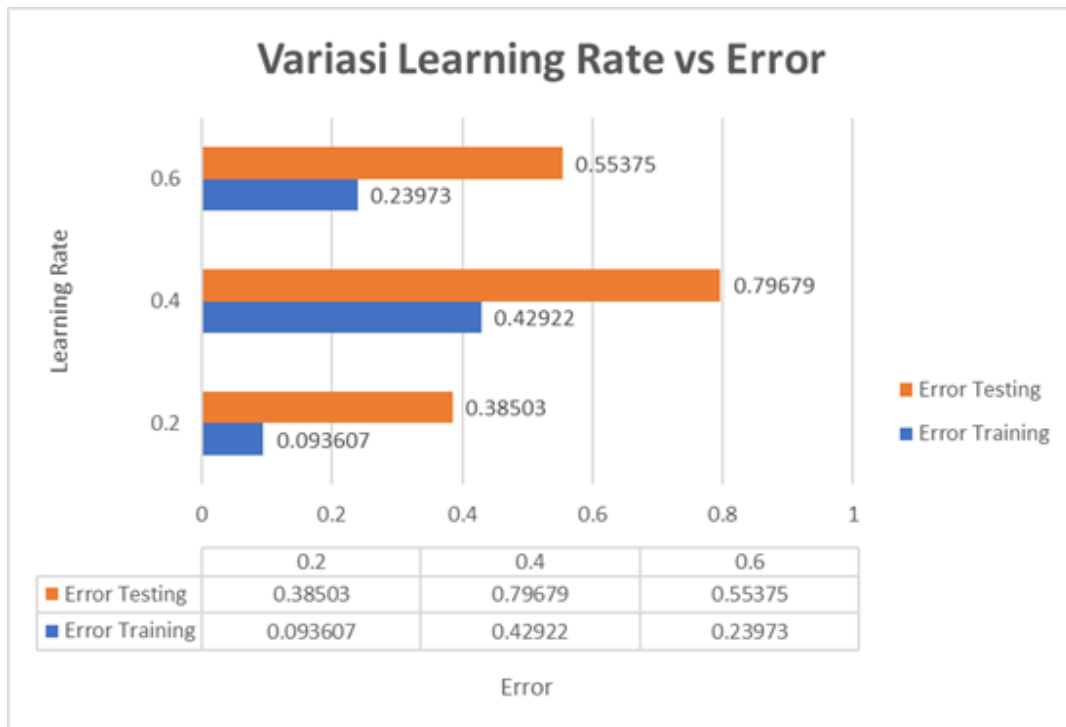
lebih baik untuk diprediksi apabila data yang diolah menggunakan *hidden* neuron yang semakin besar, dimana pada percobaan ini jumlah *hidden* neuron yang lebih baik digunakan sebesar 13. Sehingga banyaknya *hidden* neuron akan cenderung linear dengan banyaknya variance dalam dataset.

3.3.5. Variasi *Learning Rate*

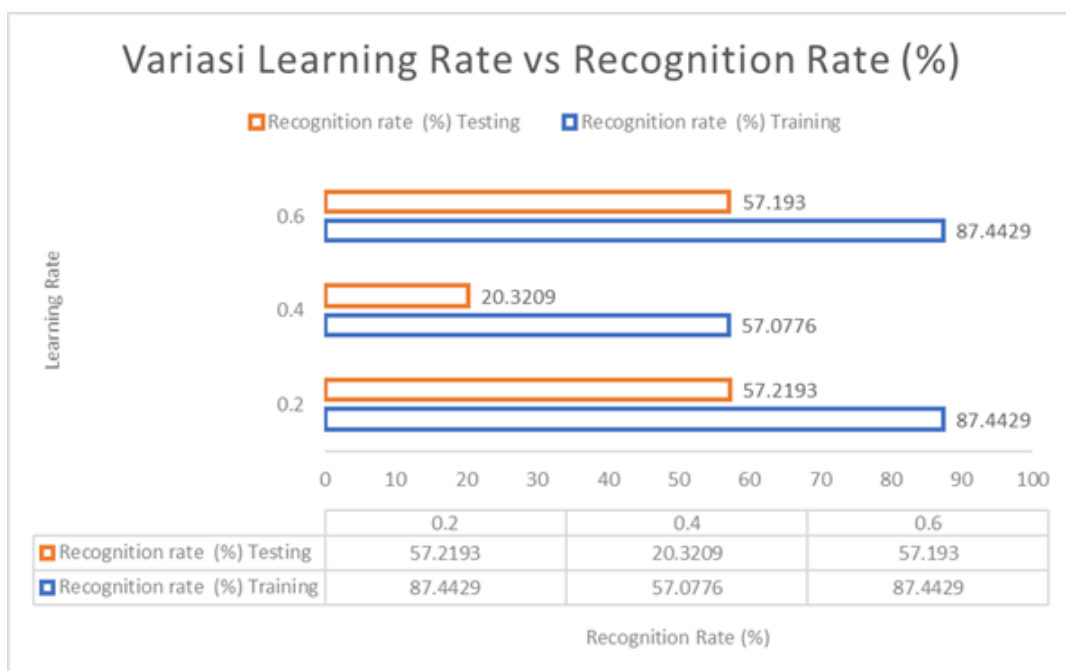
Pada percobaan ini, data Jaringan Saraf Tiruan akan divariasikan berdasarkan nilai *Learning Rate* sehingga diketahui pengaruh nilai *Learning Rate* terhadap percobaan yang dilakukan, dimana nilai *Learning Rate* yang divariasikan pada percobaan sebesar: 0.2, 0.4, dan 0.6. Pada percobaan kali ini, digunakan beberapa parameter tetap, yaitu:

- 1) Metode Inisialisasi Bobot yang digunakan:
 - a) *Input layer* ke *hidden layer*: Random
 - b) *hidden layer* ke *output layer*: Random
- 2) Metode normalisasi: Zscore
- 3) Jumlah Neuron: 9
- 4) Jumlah epoch: 2000
- 5) Koefisien Momentum (μ): 0.5

Berikut adalah hasil dari nilai *error* dan nilai *recognition Rate* berdasarkan variasi *Learning Rate* yang digunakan melalui pengolahan data pada excel, dilihat di gambar 11 dan gambar 12.



Gambar 11 Perbandingan Nilai *Error* terhadap Variasi *Learning Rate*



Gambar 12 Perbandingan Nilai *Recognition Rate* terhadap Variasi *Learning Rate*

Berdasarkan dari grafik yang dihasilkan melalui pengolahan data pada excel, didapatkan bahwa nilai *error* dan *recognition Rate* mengalami nilai fluktuatif untuk variasi *Learning Rate* yang berbeda. Hal ini disebabkan karena terjadinya *overfitting* pada variasi jumlah *hidden neuron*. Untuk nilai *error* dan *recognition Rate* paling baik terdapat pada *Learning Rate* sebesar 0.2, dengan nilai *error* 0.38503 dan nilai dari *recognition Rate* 57.2193% untuk data hasil olahan dalam proses *testing*. Dari kondisi ini kita dapat

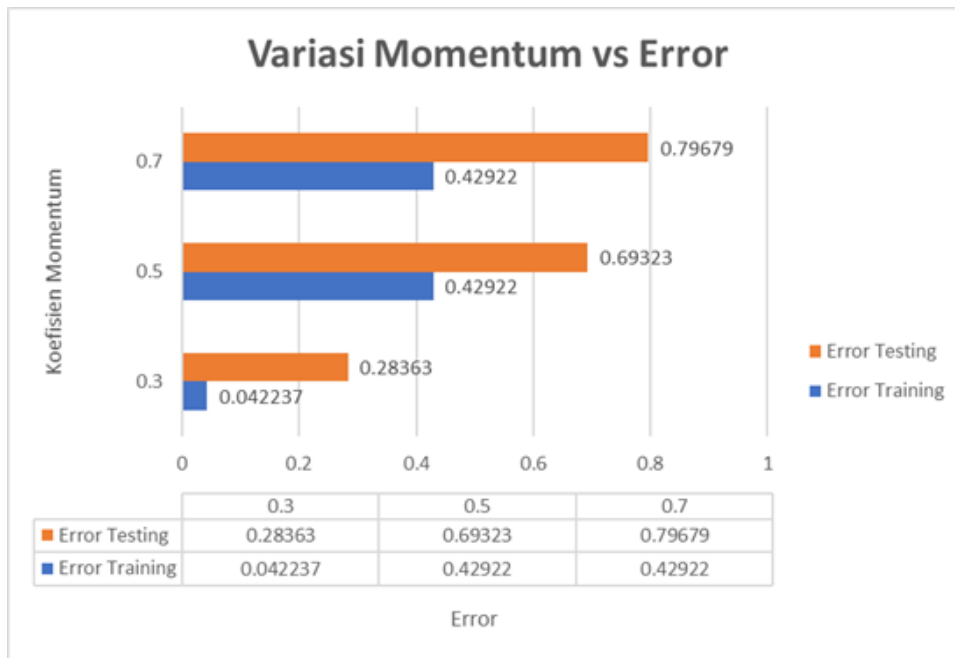
menyimpulkan bahwa data *balanced scale* memiliki akurasi lebih baik untuk diprediksi apabila data yang diolah menggunakan *Learning Rate* yang semakin kecil, dimana pada percobaan ini *Learning Rate* yang lebih baik digunakan sebesar 0.2. Sehingga nilai *Learning Rate* akan cenderung berbanding terbalik dengan banyaknya variance dalam dataset.

3.3.6. Variasi Koefisien Momentum

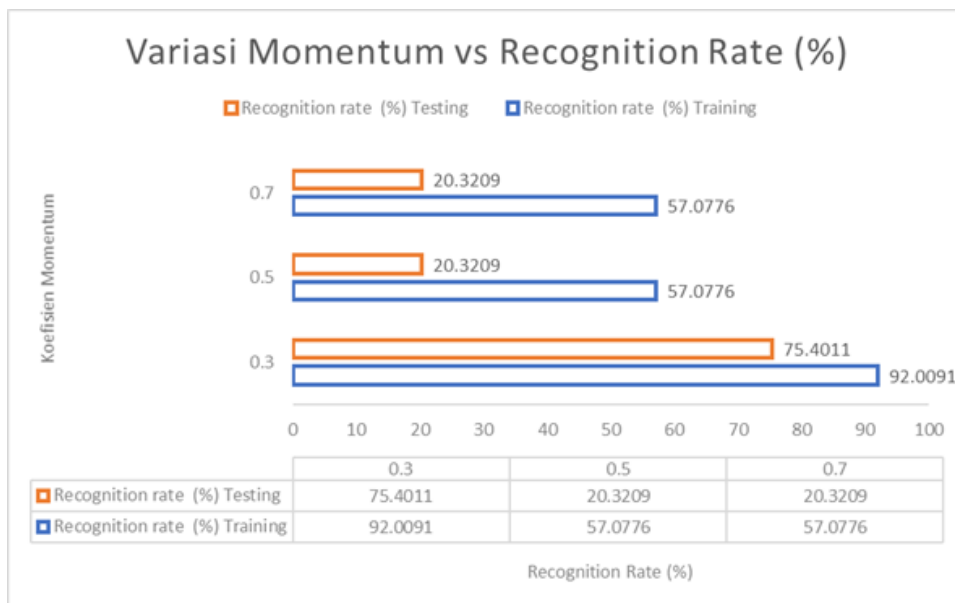
Pada percobaan ini, data Jaringan Saraf Tiruan akan divariasikan berdasarkan nilai koefisien momentum sehingga diketahui pengaruh nilai koefisien momentum terhadap percobaan yang dilakukan, dimana nilai *Learning Rate* yang divariasikan pada percobaan sebesar: 0.3, 0.5, dan 0.7. Pada percobaan kali ini, digunakan beberapa parameter tetap, yaitu:

- 1) Metode Inisialisasi Bobot yang digunakan:
 - a) *Input layer* ke *hidden layer*: Random
 - b) *hidden layer* ke *output layer*: Random
- 2) Metode normalisasi: Tanpa Normalisasi
- 3) Jumlah Neuron: 9
- 4) Jumlah epoch: 2000
- 5) *Learning Rate*: 0.4

Berikut adalah hasil dari nilai *error* dan nilai *recognition Rate* berdasarkan variasi koefisien momentum yang digunakan melalui pengolahan data pada excel, dilihat di gambar 13 dan gambar 14.



Gambar 13 Perbandingan Nilai *Error* terhadap Variasi Koefisien Momentum



Gambar 14 Perbandingan Nilai *Recognition Rate* terhadap Variasi Koefisien Momentum

Berdasarkan dari grafik yang dihasilkan melalui pengolahan data pada excel, didapatkan bahwa nilai *error* dan *recognition Rate* mengalami nilai fluktuatif untuk variasi *Learning Rate* yang berbeda. Hal ini disebabkan karena terjadinya *overfitting* pada variasi jumlah *hidden* neuron. Untuk nilai *error* dan *recognition Rate* paling baik terdapat pada koefisien momentum sebesar 0.3, dengan nilai *error* 0.38503 dan nilai dari *recognition Rate* 75.4011% untuk data hasil olahan dalam proses *testing*. Dari kondisi ini kita dapat menyimpulkan bahwa data *balanced scale* memiliki akurasi lebih baik untuk diprediksi apabila data yang diolah menggunakan koefisien momentum yang

semakin kecil, dimana pada percobaan ini koefisien momentum yang lebih baik digunakan sebesar 0.3. Sehingga nilai koefisien momentum akan cenderung berbanding terbalik dengan banyaknya variance dalam dataset.

BAB 4

KESIMPULAN

Berdasarkan hasil percobaan yang dilakukan untuk setiap perubahan variasi variabel jaringan syaraf tiruan dapat disimpulkan bahwa:

- 1) *Artificial neural network* atau jaringan syaraf tiruan adalah sistem pemrosesan data informasi yang dirancang dengan mengikuti sistem kerja otak manusia saat mengidentifikasi dan menyelesaikan suatu masalah, yaitu melakukan proses belajar melalui perubahan bobot sinapsisnya.
- 2) *Backpropagation* merupakan metode pembelajaran yang digunakan jaringan syaraf tiruan dengan mengubah bobot berdasarkan bobot sebelumnya.
- 3) Beberapa variabel atau parameter yang dapat mempengaruhi pembelajaran di dalam jaringan syaraf tiruan antara lain adalah metode inisialisasi bobot, metode normalisasi, jumlah epoch, jumlah *hidden* neuron, *Learning Rate* (α), dan koefisien momentum (μ).
- 4) Untuk dataset yang digunakan dalam percobaan ini jaringan syaraf tiruan akan memiliki tingkat ketepatan yang lebih tinggi dan *error* yang lebih kecil jika data tidak dinormalisasi.

DAFTAR REFERENSI

- [1] Uity, "Kelebihan dan Kekurangan Jaringan Syaraf Tiruan," 28 January 2020. [Online]. Available: <https://lancangkuning.com/post/14778/kelebihan-dan-kekurangan-jaringan-syaraf-tiruan.html>. [Accessed 19 Juny 2021].
- [2] D. Suhartono, "Dasar Pemahaman Neural Network," 26 July 2012. [Online]. Available: <http://socs.binus.ac.id/2012/07/26/konsep-neural-network/>. [Accessed 19 Juny 2021].

LAMPIRAN

Pembagian Tugas

Nama	Tugas
Abeltus Reforma Putra	Variasi learning rate dan momentum, merancang laporan
Dwi Pambagyo Mahardika	Variasi metode inisialisasi bobot, merancang algoritma dan program
Pramudito Anggraito	Variasi banyaknya jumlah epoch dan jumlah hidden neuron, menyiapkan dataset untuk diproses
Teuku Alif Rafi Akbar	Variasi metode normalisasi data, merancang PPT

Kode Matlab *Backpropagation* untuk *Balance Scale* Dataset

```
clear; clc;

% Import Data
data = readtable('balance-scale.xlsx');

% Data Training
input = data{1:438,1:4};
target = data{1:438,5:7};
[input_baris,input_kolom] = size(input);
[target_baris,target_kolom] = size(target);
banyak_data = input_baris;
input = normalize(input); %normalisasi data dengan zscore
                             %kalau ga di normalisasi jadiin
input = input

% Data Testing
input_test = data{439:625,1:4};
target_test = data{439:625,5:7};
banyak_test = length(input_test);
input_test = normalize(input_test); %normalisasi data
dengan zscore

% Inisialisasi ANN
banyak_input = input_kolom; %jumlah unit input layer
banyak_hidden = 9; %jumlah unit hidden layer
banyak_output= target_kolom; %jumlah unit output layer
```

```

alpha = 0.2; %learning rate
miu = 0.5; %koefisien momentum
rng(24,"philox");

% Inisialisasi Bobot: Input layer ke hidden layer (Nguyen-
Widrow)
beta = 0.7*banyak_hidden^(1/banyak_input);
v_ij = rand(banyak_input,banyak_hidden) - 0.5;
for i = 1:banyak_hidden
    norma(i) = sqrt(sum(v_ij(:,i).^2));
    v_ij(:,i) = (beta*v_ij(:,i))/norma(i);
end
v_0j = (rand(1,banyak_hidden) - 0.5)*beta;

% Inisialisasi Bobot: Input layer ke hidden layer (Random)
%v_ij = rand(banyak_input,banyak_hidden) - 0.5;
%v_0j = rand(1,banyak_hidden) - 0.5;

% Inisialisasi Bobot: Hidden layer ke output layer (Nguyen-
Widrow)
%beta = 0.7*banyak_output^(1/banyak_hidden);
%w_jk = rand(banyak_hidden,banyak_output) - 0.5;
%for i = 1:banyak_output
    %norma(i) = sqrt(sum(w_jk(:,i).^2));
    %w_jk(:,i) = (beta*w_jk(:,i))/norma(i);
%end
%w_0k = (rand(1,banyak_output) - 0.5)*beta;

% Inisialisasi Bobot: Hidden layer ke output layer (Random)
w_jk = rand(banyak_hidden,banyak_output) - 0.5;
w_0k = rand(1,banyak_output) - 0.5;

% Training
banyak_epoch = 1000;
error_min = 0.01;
flag = 0;
epoch_iter = 1;
delta_wjk_old = 0;
delta_w0k_old = 0;
delta_vij_old = 0;
delta_v0j_old = 0;

while flag == 0 && epoch_iter <= banyak_epoch
    train_true = 0;
    train_false = 0;
    for n=1:banyak_data

        % Feedforward
        xi = input(n,:);
        ti = target(n,:);

        % Input layer ke hidden layer

```

```

        z_inj = xi*v_ij + v_0j; % hitung sinyal input dengan
bobot
        for j=1:banyak_hidden;
            zj(1,j) = 1/(1+exp(-z_inj(1,j))); % hitung nilai
aktivasi (Sigmoid) setiap unit hidden sebagai hasil unit hidden
            end

            % Hidden layer ke output layer
            y_inj = zj*w_jk + w_0k; % hitung sinyal input (hasil
hidden zj) dengan bobot
            for k=1:banyak_output
                yk(1,k) = 1/(1+exp(-y_inj(1,k))); % hitung nilai
aktivasi (Sigmoid) setiap unit output sebagai hasil jaringan
                if yk(1,k) >= 0.7 %kuantisasi hasil aktivasi
                    yk(1,k) = 1;
                end
                if yk(1,k) <= 0.3
                    yk(1,k) = 0;
                end
            end
        end

        % Simpan error
        error(1,n) = 0.5*sum((yk - ti).^2); %kuadratik

        % Kalkulasi recognition rate train
        [value_train, index_train] = max(yk);
        y_train = zeros(size(ti));
        y_train(1,index_train) = 1;
        if y_train == ti
            train_true = train_true + 1;
        else
            train_false = train_false + 1;
        end

        % Backpropagation
        % Output layer ke hidden layer
        dok = (yk - ti).*(yk).*(1-yk);
        delta_wjk = alpha*zj'*dok + miu*delta_wjk_old; %
modifikasi bobot dengan momentum
        delta_w0k = alpha*dok + miu*delta_w0k_old; %
modifikasi bias dengan momentum
        delta_wjk_old = delta_wjk;
        delta_w0k_old = delta_w0k;

        % Hidden layer ke input layer
        doinj = dok*w_jk';
        doj = doinj.*zj.*(1-zj); % hitung sinyal error
        delta_vij = alpha*xi'*doj + miu*delta_vij_old; %
modifikasi bobot (momentum)
        delta_v0j = alpha*doj + miu*delta_v0j_old; %
modifikasi bias (momentum)
        delta_vij_old = delta_vij;

```

```

    delta_v0j_old = delta_v0j;

    % Update bobot dan bias (new) dengan momentum
    w_jk = w_jk - delta_wjk;
    w_0k = w_0k - delta_w0k;
    v_ij = v_ij - delta_vij;
    v_0j = v_0j - delta_v0j;
end
epoch_error(1,epoch_iter) = sum(error)/banyak_data;

if epoch_error(1,epoch_iter) < error_min
    flag = 1;
end

epoch_iter = epoch_iter + 1;
end

epoch_iter = epoch_iter - 1;
figure;
plot(epoch_error);
ylabel('Error per epoch');
xlabel('Epoch')

disp("Error per epoch = " + min(epoch_error) + "");
disp("Error akhir = " + epoch_error(1,epoch_iter) + "");

recog_rate_train = (train_true/banyak_data)*100;
disp("Recognition rate train = " + recog_rate_train + " %");

% Testing
test_true = 0;
test_false = 0;
for n=1:banyak_test

    % Feedforward
    xi_test = input_test(n,:);
    ti_test = target_test(n,:);

    % Input layer ke hidden layer
    z_inputj_test = xi_test*v_ij + v_0j;
    for j=1:banyak_hidden
        zj_test(1,j) = 1/(1+exp(-z_inputj_test(1,j)));
    end

    % Hidden layer ke output layer
    y_inputk_test = zj_test*w_jk + w_0k;
    for k=1:banyak_output
        yk_test(1,k) = 1/(1+exp(-y_inputk_test(1,k)));
        if yk_test(1,k) >= 0.7
            yk_test(1,k) = 1;
        end
        if yk_test(1,k) <= 0.3
            yk_test(1,k) = 0;
        end
    end
end

```

```

        end
    end

    % Simpan error
    error_test(1,n) = 0.5*sum((yk_test - ti_test).^2); %
kuadratik

    % Kalkulasi recognition rate test
    [value_test, index_test] = max(yk_test);
    y_test = zeros(size(ti_test));
    y_test(1,index_test) = 1;
    if y_test == ti_test
        test_true = test_true + 1;
    else
        test_false = test_false + 1;
    end
end

avgerrorestest = sum(error_test)/banyak_test;
disp("Error average test = "+ avgerrorestest + "");

recog_rate_test = (test_true/banyak_test)*100;
disp("Recognition rate test = "+ recog_rate_test + " %");

```