PENGENDALIAN SUDUT ARAH *MOBILE* ROBOT MENGGUNAKAN JARINGAN SYARAF TIRUAN *BACKPROPAGATION*

Diah Putu Dwijayanti, Sutikno, Sukmawati Nur Endah, Priyo Sidik Sasongko

Jurusan Ilmu Komputer/ Informatika Fakultas Sains dan Matematika, Undip
diahputudwijayanti@gmail.com

Abstrak

Dunia robotika berkembang dengan begitu cepat, sehingga robot dapat diaplikasikan dalam berbagai bidang, misalnya pada bidang industri, bidang kesehatan dan bidang pertanian.Hal penting dalam perencanaan dan pembangunan robot adalah masalah olah gerak.Pengaturan olah gerak pada robot membutuhkan algoritma cerdas agar robot dapat bergerak dengan baik.Penelitian ini membuat sebuah pengendalian sudut arah *mobile* robot menggunakan jaringan syaraf tiruan *backpropagation.*Robot yang digunakan adalah robot lego mindstroms 2.0 beroda empat.Pelatihan data menggunakan *backpropagation* dilakukan pada Matlab menghasilkan nilai bobot.Nilai bobot tersebut digunakan pada pemrograman menggunakan Lejos yang kemudian diterapkan pada robot lego. Setelah program diterapkan, robot dapat bergerak melewati lintasan lurus, lintasan berbelok 30 derajat, lintasan berbelok 45 derajat, lintasan berbelok 60 derajat dan lintasan berbelok 90 derajat tanpa menabrak benda/rintangan dengan rata-rata tingkat keberhasilan 90.67%. Pada tugas akhir ini, algoritma jaringan syaraf tiruan *backpropagation* telah terimplementasi dengan baik pada robot sehingga robot dapat mengendalikan olah geraknya sendiri.

Kata kunci : jaringan syaraf tiruan *backpropagation*, robotika, robot lego, gerak robot

1. PENDAHULUAN

Perkembangan teknologi mikrokontroler memberikan dampak yang besar terhadap dunia robotika sehingga berkembang begitu pesat.Hal ini tentu saja menuntut manusia untuk mengikuti perkembangannya dalam mempelajari lebih lanjut.Secara garis besar robot dibagi menjadi dua berdasarkan tipe pergerakannya, yaitu *mobile* robot dan *stationary* robot.*Mobile* robot adalah robot yang mempunyai kemampuan untuk berpindah lokasi dari satu tempat ke tempat lainnya, sedangkan *stationary* robot adalah robot yang tidak mempunyai kemampuan untuk berpindah posisi.Jenis *mobile* robot bermacam-macam, seperti robot beroda dan robot berkaki [5].

Salah satu hal yang penting dalam perencanaan dan pembangunan pada *mobile* robot adalah mengenai olah gerak.Olah gerak yang dimaksud terdiri dari perencanaan gerak dan pengambilan keputusan pergerakan untuk menghindari tabrakan dalam sebuah lintasan. Agar *mobile* robot bergerak sesuai dengan lintasan yang akan dilalui maka perlu pengaturan sudut arah robot dengan baik. Pengaturan sudut arah robot membutuhkan algoritma cerdas diantaranya yaitu seperti algoritma genetika, logika fuzzy, dan jaringan syaraf tiruan.

Jaringan syaraf tiruan *backpropagation* secara umum digunakan untuk menirukan kinerja otak manusia, yang memiliki kemampuan untuk belajar (beradaptasi dan memahami sesuatu yang baru).Dengan menggunakan metode jaringan syaraf tiruan *backpropagation*, *mobile* robot akan melakukan proses pembelajaran sehingga mampu bergerak maju, ke kiri, ke kanan tanpa tabrakan pada lintasan.

Penelitian pengendalian sudut arah ini menggunakan Matlab untuk melakukan pelatihan jaringan agar mendapatkan bobot/pola terbaik dan menggunakan robot lego mindstorm 2.0 dengan pemrograman *Java Lejos*untuk penerapannya.

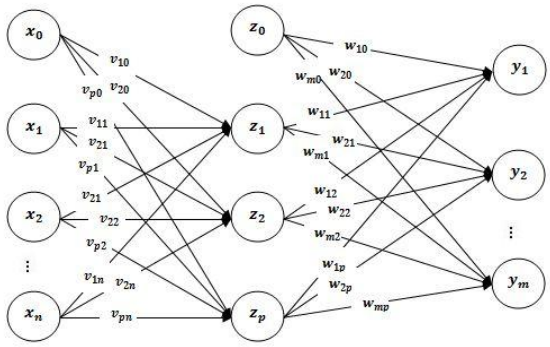
2. TINJAUAN PUSTAKA

Jaringan Syaraf Tiruan *Backpropagation*

Jaringan syaraf tiruan adalah suatu pengolahan informasi yang mempunyai karakteristik unjuk kerja tertentu sebagaimana jaringan syaraf biologis [1]. Jaringan syaraf tiruan dicirikan oleh (1) hubungan pola antar neuron, (2) metode untuk menentukan bobot dan (3) fungsi aktivasinya.

Jaringan syaraf tiruan *backpropagation* merupakan algoritma melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan untuk memberikan respon yang benar terhadap pola masukan yang serupa (tidak sama) dengan pola yang dipakai selama pelatihan [4]. Metode *backpropagation* melibatkan tiga tahapan: feedforward (propagasi maju) pola pelatihan masukan, *backpropagation* (propagasi mundur) terhadap *error*, serta penyesuain bobot [1].

Gambar 1 merupakan gambar arsitektur jaringan syaraftiruan *backpropagation*



Gambar 1. Arsitektur Jaringan Syaraf Tiruan *Backpropagation*

Arsitektur jaringan *backpropagation* pada Gambar 2 meliputi lapisan *input* yaitu x_1, x_2 hingga x_n; lapisan tersembunyi yaitu $z_1, z_2,$ hingga z_p; serta lapisan *output* yaitu y_1, y_2, y_m. Bobot yang menghubungkan lapisan *input* x_1, x_2, x_n dengan lapisan tersembunyi z_1, z_2, z_p adalah v_{11}, v_{21}, v_{p1} dan seterusnya hingga v_{pn}. v_{10}, v_{20}, v_{p0} adalah bobot bias yang menuju ke lapisan tersembunyi. Bobot yang menghubungkan z_1, z_2, z_p dengan lapisan *output* adalah w_{11}, w_{21}, w_{m1} hingga ke w_{mp}. Bobot bias $w_{10}, w_{20}., w_{m0}$ menghubungkan lapisan tersembunyi dengan lapisan *output*.

Algoritma pelatihan *backpropagation* adalah sebagai berikut [1] :

Langkah 0. Menginisialisasi semua bobot (ambil bobot awal dengan nilai random yang cukup kecil)

Langkah 1. Ketika kondisi berhenti salah (belum terpenuhi), melakukan langkah 2-9

Langkah 2. Melakukan langkah 3-8 untuk setiap pasang data pelatihan.

Langkah 3. Tiap-tiap unit *input* x_i ($i = 1,2,3, \dots n$) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua unit pada lapisan yang ada di atasnya (lapisan tersembunyi)

Langkah 4. Tiap-tiap unit tersembunyi z_j ($j = 1,2,3, \dots p$) menjumlahkan sinyal-sinyal *input* terbobot dengan rumus persamaan (1).

$$z\_net_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad \text{..... (1)}$$

Penghitungan dilanjutkan menggunakan fungsi aktivasi untuk menghasilkan sinyal *output* dengan rumus persamaan (2).

$$z_j = f\left(z\_net_j\right) = \frac{1}{1+e^{-z\_net_j}} \quad \text{.....(2)}$$

Kemudian, menghitung fungsi aktivasi untuk menghitung sinyal keluaran, $z_j = f\,(z\_net_j)$ lalu mengirimkan sinyal ini ke semua unit pada layer diatasnya (unit keluaran).

Langkah 5. Tiap-tiap unit *output* y_k ($k = 1,2,3, \dots m$) menjumlahkan sinyal-sinyal *input* terbobotmenggunakan rumus persamaan (3).

$$y_{net\ k} = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \quad \text{.....(3)}$$

Penghitungan dilanjutkan menggunakan fungsi aktivasi untuk menghitung sinyal *output* dengan rumus persamaan (4).

$$y_k = f\,(y\_net_k) = \frac{1}{1+e^{-y\_net_k}} \quad \text{.....(4)}$$

Sinyal *output* tersebut dikirim ke semua unit di lapisan atasnya (unit-unit *output*).

Langkah 6. Tiap-tiap unit *output* y_k ($k = 1,2,3. \dots m$) menerima target pola yang berhubungan dengan pola *input* pembelajaran, lalu menghitung informasi *error*nya menggunakan rumus persamaan (6).

$$\delta_k = (t_k - y_k)\, f'(y\_net_k) \quad \text{.....(6)}$$

Kemudian menghitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai w_{jk}) menggunakan rumus persamaan (7).

$$\Delta w_{jk} = \alpha\, \delta_k z_j \quad \text{.....(7)}$$

Selanjutnya menghitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k}) menggunakan rumus persamaan (8).

$$\Delta w_{0k} = \alpha\, \delta_k \quad \text{.....(8)}$$

Informasi *error* δ_k dikirim ke unit-unit yang ada dilapisan bawahnya.

Langkah 7. Menghitung factor δ unit tersembunyi berdasarkan *error* pada tiap-tiap unit tersembunyi z_j ($j = 1,2,3, \dots p$) menggunakan rumus persamaan (9).

(2.3)

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k w_{kj} \quad \text{.....(9)}$$

Kemudian menghitung informasi *error* δ unit tersembunyi dengan persamaan rumus (10).

$$\delta_j = \delta\_net_j f'\,(z\_net_j) \quad \text{.....(10)}$$

Kemudian menghitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij}) dengan persamaan (11).

$$\Delta v_{ij} = \alpha \delta_j x_i \text{ .....(11)}$$

Menghitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{0j})dengan persamaan (12).

$$\Delta v_{0j} = \alpha \delta_j \text{ .....(12)}$$

Langkah 8. Tiap-tiap unit $output y_k$ ($k = 1,2,3, \dots m$) memperbaiki nilai bias dan bobotnya ($j = 0, 1,2, \dots p$) menggunakan rumus persamaan (13).

$$w_{jk}\,(baru) = w_{jk}\,(lama) + \Delta w_{jk} \text{ .....(13)}$$

Tiap-tiap unit tersembunyi z_j ($j = 0,1,2, \dots p$) memperbaiki bias dan bobotnya ($i = 0,1,2, \dots n$)dengan rumus persamaan (14).

$$v_{ij}\,(baru) = v_{ij}\,(lama) + \Delta v_{ij} \text{ .....(14)}$$

Langkah 9. Uji syarat berhenti.

Robotika

Robotika adalah ilmu yang mempelajari mengenai proses perancangan dan pengembangan robot serta membahas mengenai penerapan-penerapan teknologi robotika pada kehidupan manusia. Dalam penerapannya, ilmu robotika erat hubungannya dengan ilmu kecerdasan buatan [5].

Robot Lego Mindstroms adalah robot yang diproduksi oleh Lego untuk dapat dibentuk menjadi berbagai prototipe robot dan dapat diprogram untuk kemampuan tertentu [2].Robot Lego Mindstorms memiliki beberapa komponen. antara lain NXTBrick, sensor (sensor ultrasonik, sensor cahaya, sensor sentuh, sensor suara) dan aktuator.

Sensor Ultrasonik

Sensor ultrasonik yang tersedia dalam LEGO Mindstorms NXT ditunjukkan pada Gambar 2.

Gambar 2. Sensor Ultrasonik

Cara kerja sensor ultrasonik pada Lego Mindstormsmengikuti konsep penggunaan gelombang ultrasonik untuk menentukan jarak.

Awalnya mata kanan akan memancarkan gelombang ultrasonik. Setelah beberapa saat mata kiri akan menerima pantulan gelombang ultrasonik yang dipancarkan sebelumnya. Selisih antara waktu gelombang dikirimkan dan pantulannya diterima akan digunakan untuk menentukan posisi benda terdekat.

NXTBrick

NXTBrick adalah otak dan sumber tenaga dari robot LEGO Mindstorms NXT [2]. Program yang sudah dibuat akan dimasukan dan dijalankan oleh NXTBrick. Robot dapat menerima informasi seperti sentuhan, suara, atau intensitas cahaya sesuai dengan sensor yang digunakan. Informasi tersebut dikirim ke NXTBrick agar dapat diproses lebih lanjut. NXTBrick juga memberikan tenaga yang diperlukan motor agar dapat bergerak. Gambar 3 menunjukkan sebuah NXTBrick.

Gambar 3. Sebuah NXTBrick

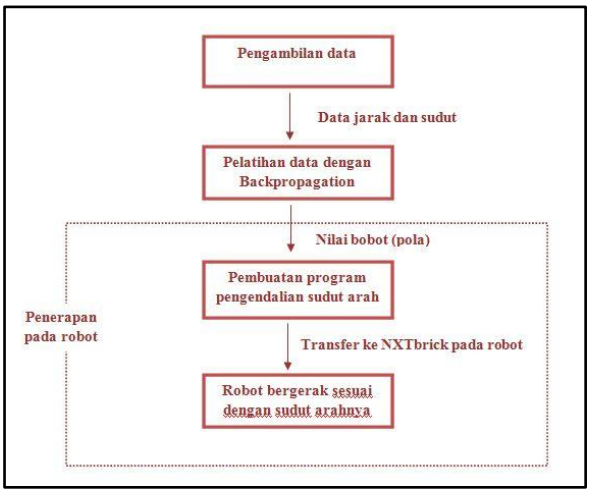
Aktuator

Motor berfungsi untuk menggerakkan bagian robot, seperti memutar roda atau menjadi sendi. Satu *brick* dapat dipasangi hingga 3 motor. Lejos mampu mengelola beberapa jenis aktuator. Jika akan membangun robot mobil dengan NXT, maka dapat menggunakan NXT motor atau motor lain yang dimiliki oleh Lego. Gambar 4 menunjukkan bentuk aktuator.

Gambar 2. Sebuah Aktuator

3. PEMBAHASAN DAN HASIL PENELITIAN

Gambaran umum tahapan - tahapan pengembangan penelitian pengendalian sudut arah *mobile* robot menggunakan *backpropagation* dijelaskan pada Gambar 5.



Gambar 3. Gambaran umum tahapan-tahapan pengembangan penelitian

Robot lego beroda empat diharapkan mampu bergerak mengendalikan sudut arahnya sendiri dengan suatu program yang ditransferkan pada NXTBrick. Olah gerak pada robot ini mengandalkan tiga navigator sensor ultrasonik yang berfungsi menangkap jarak sensor terhadap benda (penghalang) didepannya.

Alur proses pada penelitian ini adalah dimulai dari pengambilan data, pelatihan data pada Matlab dan penerapan pada robot. Data yang diperoleh digunakan untuk pelatihan dan menghasilkan nilai bobot (pola). Nilai bobot tersebut digunakan pada pemrograman Lejos pada PC yang kemudian ditanamkan pada robot. Program pengendalian sudut arah yang diterapkan pada robot, digunakan robot untuk bergerak melewati lintasan tanpa menabrak.

Pengambilan Data

Pengambilan data menggunakan metode *trial and error* yaitu mengambil data secara langsung menggunakan robot lego di lintasan. Hal ini dilakukan untuk mendapatkan data jarak sensor terhadap benda (penghalang) dan sudut arah aktuator robot. Pengambilan data dengan memprediksi gerak robot yang mungkin, yaitu mengambil data ketika robot sejajar dengan lintasan, hampir menabrak lintasan, posisi ketika berbelok ke kiri dan kanan.Data pelatihan berjumlah 180 pasang yang kemudian digunakan untukpelatihan menggunakan jaringan syaraf tiruan *backpropoagation* pada Matlab. Data pelatihan ditunjukkan pada Tabel 1.

Tabel 1. Data Pelatihan

No	Left distance (x1)	Right distance (x2)	Frontal distance (x3)	Angle (T)
1	33	36	33	25
2	34	40	33	25
3	33	225	34	25
4	33	51	34	25
5	34	180	33	25
…	…..			
180	50	60	199	0

Pelatihan Data

Sebelum dilakukan proses pelatihan, data yang diperoleh sebanyak 180 pasang data ditransformasikan terlebih dahulu pada interval [0.1, 0.9] menggunakan persamaan (14). Penggunaan p sebagai data minimun dan q sebagai data maksimum.

$$x^b = \frac{0.8\,(x-p)}{q-p} + 0.1 \quad \text{.....(14)}$$

Data yang sudah ditransformasi, dapat dilakukan pelatihan data menggunakan algoritma *backpropagation* dengan bantuan Matlab.Pelatihan pada Matlab dilakukan secara berkelompok (*batch training*) yaitu semua pola data dimasukkan dulu diletakkan pada sebuah matriks, kemudian bobot diubah dan diproses. Pada saat membentuk jaringan *backpropagation*, Matlab akan memberi nilai bobot dan bias awal dengan bilangan acak kecil. Bobot dan bias tersebut akan berubah setiap kali pembentukan jaringan. Bobot berubah setelah semua pola masukan data diberikan pada jaringan. Nilai *error* dan suku perubahan bobot yang terjadi pada setiap pola masukan dijumlahkan untuk menghasilkan bobot baru. Hal tersebut terus dilakukan hingga mencapai *epoch* yang ditentukan atau hingga nilai *MSE* yang ditentukan. Kemudian diperolehnya nilai bobot terbaik hasil dari pelatihan. dan dapat digunakan untuk pemrograman selanjutnya pada *Java Lejos*.

Bobot hasil pelatihan terbaik dari pelatihan data digunakan di dalam robot untuk penghitungan jaringan syaraf tiruan *backpropagation*nya adalah sebagai berikut :

Bobot antara lapisan masukan jaringan dan lapisan tersembunyi jaringan sebagai berikut:

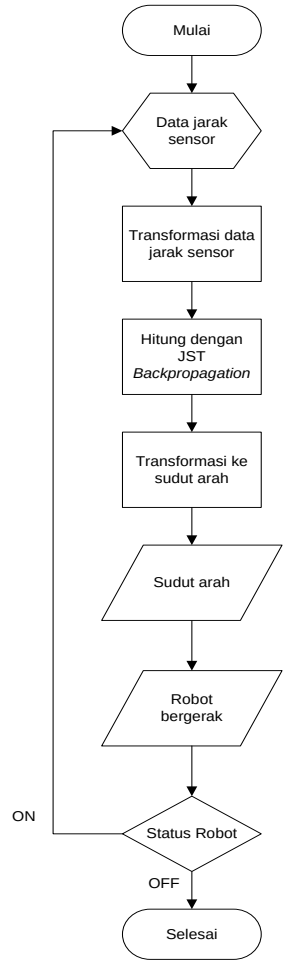
$$V = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & z_2 & z_3 \end{array} \\ \begin{bmatrix} 0.07 & 0.56 & 0.22 \\ 1.05 & 0.53 & 4.29 \\ 7.30 & 1.41 & 1.68 \\ 0.48 & 0.79 & 1.30 \end{bmatrix} \end{array}$$

Bobot antara lapisan tersembunyi jaringan dan lapisan keluaran jaringan sebagai berikut:

$$W = \begin{array}{c} \\ z_0 \\ z_1 \\ z_2 \\ z_3 \end{array} \begin{array}{c} y_1 \\ \left[\begin{array}{c} 0.08 \\ 4.36 \\ -1.17 \\ -2.93 \end{array}\right. \end{array}$$

Penerapan pada Robot

Proses penerapan robotdimulai dengan memperoleh data jarak sensorultrasonik kemudian data jarak sensor tersebut ditransformasikan kedalam interval [0.1, 0.9] menggunakan persamaan (14). Data jarak yang telah ditransformasi diolah dengan jaringan syaraf tiruan *backpropagation* hingga menghasilkan keluaran jaringan. Hasil keluaran jaringan itu diretransformasi ke dalam bentuk sudut arah. Sudut arah tersebut digunakan robot untuk menggerakan aktuator (motor) robot untuk bergerak pada lintasan. Proses itulah dilakukan berulang selama ada *input* masuk berupa data jarak sensor. Alur proses penerapan robot ditunjukkan pada Gambar 6, simbol-simbol yang digunakan dalam pembuatan *flowchart* ini mengacu pada referensi [3].



Gambar 4. *Flowchart* Proses Implementasi pada Robot

Pengujian

Hasil pengujian pada pelatihan data dengan menggunakan berbagai parameter-parameter jaringan adalah sebagai berikut :

1. Laju pembelajaran (*learning rate*)

Setelah dilakukan pengujian pada proses pelatihan data dengan membedakan variabel laju pembelajaran (*learning rate)* 0.1, 0.2, 0.3 dan 0.4, diperoleh hasil pelatihan pada Tabel 2.

Tabel 2. Hasil Pelatihan Melihat Pengaruh *learning rate*

No	lr	*Goal target*	MSE	*Epoch*
1	0.1	0.02	0.02	202884
2	0.2	0.02	0.02	101447
3	0.3	0.02	0.02	67635
4	0.4	0.02	0.02	50729

Tabel 2 menunjukkan bahwa pengaruh *learning rate* (lr) dalam pelatihan data bahwa semakin besar nilai *learning rate* (lr) akan berimplikasi pada semakin besarnya langkah pembelajaran. Apabila *learning rate* (lr) kecil, maka algoritma akan konvergen dalam jangka waktu yang lebih lama, iterasinya (*epoch*) juga semakin banyak.

2. *Goal target*

Setelah dilakukan pengujian pada proses pelatihan data dengan membandingkan *goal target.* Iterasi berhenti ketika nilai fungsi kinerja (MSE) kurang dari atau sama dengan *goal target* (kinerja tujuan). Berikut hasil yang diperoleh pada Tabel 3.

Tabel 3. Hasil Pelatihan Melihat Pengaruh *Goal Target*

No	*Goal target*	lr	MSE	*Epoch*
1	0.01	0.02	0.01	202884
2	0.02	0.02	0.02	101447
3	0.03	0.02	0.03	13192

Tabel 3 menunjukan bahwa *goal target* mempengaruhi banyaknya iterasi. Semakin kecil MSEnya, hasil pelatihan (bobot) akan semakin kecil selisih *error*-nya (semakin baik karena mendekati sempurna).

3. Iterasi (*epoch*)

Setelah dilakukan pengujian dengan membandingkan *epoch* (iterasi) dalam pelatihan, berikut hasil yang diperoleh pada Tabel 4.

Tabel 4. Hasil Pelatihan Melihat PengaruhIterasi

No	*Epoch*	lr	*Time*	MSE
1	300000	0.02	1:13:48	0.0078
2	30000	0.02	03:41	0.0284
3	3000	0.02	00:22	0.0442

Tabel 4 menunjukkan bahwa semakin besar jumlah iterasi maka MSE yang diperoleh semakin kecil dan waktu yang digunakan semakin panjang.

4. Bobot/pola terbaik

Setelah dilakukan pengujian pada data pelatihan dengan memasukkan variabel/ parameter jaringan syaraf tiruan yang berbeda-beda guna mendapatkan hasil terbaik, maka diperoleh hasil pengujian data pelatihan pada Tabel 5.

Hasil pengujian yang tersaji pata Tabel 5 dapat diketahui bahwa hasil pelatihan terbaik diperoleh dari Pelatihan 12 yaitu dengan MSE bernilai 0.01, *learning rate* 0.04, iterasi sebanyak 118518 dan waktu pelatihan selama 17 menit 57 detik. Dengan nilai *learning rate* relatifkecil dan nilai MSE akhir paling kecil, maka dapat disimpulkan bobot hasil Pelatihan 12 adalah yang terbaik dan hasil pelatihannya akan digunakan pada penerapan gerak robot pada robot lego.

Bobot hasil pelatihan terbaik dari pelatihan 12 digunakan di dalam robot untuk penghitungan jaringan syaraf tiruan *backpropagation*nya adalah sebagai berikut :

Bobot antara lapisan masukan jaringan dan lapisan tersembunyi jaringan sebagai berikut:

$$V = \begin{matrix} \\ x_0 \\ x_1 \\ x_2 \\ x_3 \end{matrix} \begin{bmatrix} z_1 & z_2 & z_3 \\ 0.07 & 0.56 & 0.22 \\ 1.05 & 0.53 & 4.29 \\ 7.30 & 1.41 & 1.68 \\ 0.48 & 0.79 & 1.30 \end{bmatrix}$$

Bobot antara lapisan tersembunyi jaringan dan lapisan keluaran jaringan sebagai berikut:

$$W = \begin{matrix} \\ z_0 \\ z_1 \\ z_2 \\ z_3 \end{matrix} \begin{bmatrix} y_1 \\ 0.08 \\ 4.36 \\ -1.17 \\ -2.93 \end{bmatrix}$$

Setelah bobot ditanamkan pada robot melalui pemrograman *Lejos*, robot dapat diujikan pada lintasan dengan kecepatan dan lintasan berbeda.Pengujian dilakukan lima kali pada masing-masing parameter agar memperoleh keakuratan analisa pada hasil. Dari pengujian yang dilakukan maka didapatlah hasil pengujian pada Tabel 5 untuk lintasan lurus, Tabel 6 pada lintasan berbelok 30 derajat, Tabel 7 pada lintasan berbelok 45 derajat, Tabel 8 pada lintasan berbelok 60 derajat dan Tabel 9 pada lintasan berbelok 90 derajat.

Tabel 5. Lintasan Lurus

No	Kecepatan	Pengujian 1	Pengujian 2	Pengujian 3	Pengujian 4	Pengujian 5
1	10	berhasil	Berhasil	berhasil	berhasil	Berhasil
2	55	berhasil	Berhasil	berhasil	berhasil	Berhasil
3	100	berhasil	Berhasil	berhasil	berhasil	Berhasil

Tingkat keberhasilan$\frac{15}{15}$x100 = 100 %

Tabel 6. Lintasan Berbelok 30 Derajat

No	Kecepatan	Pengujian 1	Pengujian 2	Pengujian 3	Pengujian 4	Pengujian 5
1	10	Berhasil	Berhasil	berhasil	berhasil	Berhasil
2	55	Berhasil	tidak berhasil	berhasil	berhasil	Berhasil
3	100	tidak berhasil	Berhasil	berhasil	berhasil	Berhasil

Tingkat keberhasilan$\frac{13}{15}$x 100 = 93.3 %

Tabel 7. Lintasan Berbelok 45 Derajat

No	Kecepatan	Pengujian 1	Pengujian 2	Pengujian 3	Pengujian 4	Pengujian 5
1	10	berhasil	berhasil	berhasil	berhasil	Berhasil
2	55	berhasil	berhasil	berhasil	berhasil	Berhasil
3	100	tidak berhasil	tidak berhasil	berhasil	berhasil	Berhasil

Tingkat keberhasilan $\frac{13}{15}$x 100 = 93.3%

Tabel 8. Lintasan Berbelok 60 Derajat

No	Kecepatan	Pengujian 1	Pengujian 2	Pengujian 3	Pengujian 4	Pengujian 5
1	10	berhasil	berhasil	berhasil	berhasil	berhasil
2	55	tidak berhasil	berhasil	berhasil	berhasil	berhasil
3	100	berhasil	tidak berhasil	tidak berhasil	berhasil	berhasil

Tingkat keberhasilan $\frac{12}{15}$x 100 = 86.7 %

Tabel 9. Lintasan Berbelok 90 Derajat

No	Kecepatan	Pengujian 1	Pengujian 2	Pengujian 3	Pengujian 4	Pengujian 5
1	10	berhasil	berhasil	tidak berhasil	berhasil	tidak berhasil
2	55	tidak berhasil	tidak berhasil	berhasil	berhasil	berhasil
3	100	berhasil	tidak berhasil	tidak berhasil	tidak berhasil	berhasil

Tingkat keberhasilan $\frac{11}{15}$x 100 = 80 %

Tingkat keberhasilan rata-rata gerak robot melewati lintasan secara keseluruhan adalah $\frac{100\% +93.3\%+93.3\%+86.7\%+80\%}{5}$ = 90.67%

Menganalisa pengujian gerak robot, terlihat bahwa pergerakan robot terhadap lintasan hampir mendekati sempurna. Berikut detail analisisnya :

1. Sensor ultrasonik LEGO Mindstorms NXT tidak mampu membaca jarak yang lebih kecil dari 3 cm, terjadinya kekacauan untuk perhitungan pada jarak lebih besar dari 255 cm, dan untuk benda yang berada pada jarak 25 cm – 50 cm, sensor ultrasonik mempunyai probabilitas besar untuk membaca jarak tersebut menjadi 48 cm, sehingga sudut arah yang dihasilkan kurang relevan.
2. Sensor ultrasonik yang merupakan sensor utama untuk navigasi dan penghindaran halangan sering mengalami gangguan (*noise*), sehingga sudut arah yang dikendalikan kurang relevan.
3. Deteksi jarak sudut memerlukan waktu sekitar 2 hingga 3 detik, sehingga faktor kecepatan mempengaruhi pengendalian sudut arah dan pergerakan robotnya. Semakin tinggi kecepatan gerak robot, kinerja pengendalian sudut arah menurun.

4. KESIMPULAN

Penelitian ini menunjukkan bahwa jaringan syaraf tiruan b*ackpropagation* dapat diterapkan pada robot edukasi lego mindstorms 2.0 untuk mengatur gerak robot melalui pengendalian sudut arah, dengan tingkat keberhasilan ujicoba gerak robot pada pengendalian sudut arahnya melewati berbagai lintasan mencapai rata-rata prosentase 90.67%.

Sedangkan pada proses pelatihan data, pelatihan dilakukan berulang dengan pembeda pada parameter-parameter jaringan untuk mendapatkan bobot/model terbaik sehingga dapat memperoleh nilai *error* paling rendah dalam penerapannya pada robot. Hasil pelatihan terbaik diperoleh pada *learning rate* 0.4, MSE 0.01, iterasi sebanyak 118518 dan waktu pelatihan 17 menit 57 detik. Hasil ini memuaskan karena menghasilkan MSE yang paling kecil, *learning rate* yang relatif kecil dengan iterasi dan waktu yang efisien.

5. DAFTAR PUSTAKA

[1] Fausett, L., 1994,"*Fundamentals of Neural Networks: Architecture, Algorithms, and Applications"*, Prentice Hall Inc

[2] Jatmiko W and Dkk., 2010, "*Robot Lego Mindstrom : Teori dan Praktek"*, Cetakan Pertama. Depok, Fasilkom Universitas Indonesia

[3] Ladjamudin A. B., 2006,"*Rekayasa Perangkat Lunak*", Cetakan Pertama, Yogyakarta, Graha Ilmu

[4] Siang J. J., 2005,"*Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan Matlab"*, Edisi Pertama, Yogyakarta, Penerbit Andi

[5] Siegwart R, Nourbakhsh I. R, dan Scaramuzza D., 2004,*"Introduction to Autonomous Mobile Robots"*, 2nd ed., London, The MIT Press