IMPLEMENTASI ALGORITMA NEURAL NETWORK BACKPROPAGATION UNTUK PERILAKU NPC BERKELOMPOK DI GAME ACTION-RPG

Reza Purbayanto

Teknik Informatika Universitas Komputer Indonesia

Jl.Dipatiukur 112-114 Bandung
E-mail : reza.purbayanto@gmail.com

ABSTRAK

Role Playing Game atau yang biasa disingkat RPG adalah salah satu jenis permainan dimana para pemainnya memainkan peran karakter fiksi yang bertujuan untuk merajut sebuah cerita dalam game. Biasanya musuh dalam game *Action*-RPG dibuat berkelompok dengan *job class* yang berbeda. Dari beberapa *job class* tersebut diharapkan dapat saling bekerjasama sesuai situasi dan berdasarkan *job class* nya.

Dibutuhkan algoritma yang dapat memberikan solusi agar sebuah non player character (NPC) dapat melakukan kerjasama. Tidak menutup kemungkinan implementasi *Artificial Neural Network* (ANN) dalam AI *Game*. Backpropagation adalah salah satu metode ANN yang paling banyak digunakan, karena modelnya yang hampir sama dengan sistem pengendalian secara umum. ANN dengan sifat *backpropagation* dipilih dalam game *Action*-RPG karena sifatnya yang terbimbing.

Berdasarkan hasil dari penelitian dapat disimpulkan implementasi algoritma *neural network backpropagation* dapat digunakan untuk memprediksi pola kerjasama yang harus dipilih untuk mengalahkan pemain. Dari *dataset* yang digunakan memiliki akurasi prediksi yang dihasilkan sebesar 96.66% menggunakan parameter *learning rate* 0.4 batas *error* 0.00001 dan menggunakan 8 perceptron pada *hidden layer*. Dengan akurasi yang cukup baik dapat disimpulkan bahwa NPC musuh dapat bekerjasama sesuai dengan pola data latih yang digunakan dan secara objektif perilaku kerjasama dapat terlihat, hal ini membuat permainan lebih baik.

Kata kunci : *action-rpg*, kerjasama, *neural network backpropagation*, permainan

1. PENDAHULUAN

Role Playing Game atau yang biasa disingkat RPG adalah salah satu jenis permainan dimana para pemainnya memainkan peran karakter fiksi yang bertujuan untuk merajut sebuah cerita dalam game. Pemain merancang tokoh utama yang akan ia mainkan sesuai karakteristik yang diinginkannya. Pada *genre Action*-RPG ini semua aksi dilakukan secara *real-time* dan mempunyai kontrol penuh pada karakter sama seperti genre *hack and slash* atau *shooter game* namun para pemainnya merupakan tokoh khayalan. Pada jenis permainan Action-RPG elemen *non player character* (NPC) sangat penting, NPC itu sendiri harus memiliki kecerdasan agar mempunyai sebuah perilaku. Kecerdasan buatan (AI) diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia, hal ini menjadi salah satu cara untuk membantu sebuah mesin untuk memberikan sebuah keputusan untuk setiap kondisi yang ada.

Permasalahan dari penelitian sebelumnya adalah tidak adanya kerjasama pada perilaku NPC. Menurut kamus bahasa Indonesia kata kerjasama memiliki arti kegiatan atau usaha yang dilakukan oleh beberapa orang (lembaga, pemerintah, dan sebagainya) untuk mencapai tujuan bersama [2]. Pada *game action*-RPG terdapat elemen penting yaitu NPC musuh, NPC musuh ini biasanya dibedakan oleh beberapa *job class* seperti *assassin*, *warior*, *archer*, dan NPC musuh ini mempunyai tujuan yang sama yaitu untuk mengalahkan karakter pemain. Biasanya musuh dalam *game action*-RPG dibuat berkelompok dengan *job class* yang berbeda. Dari beberapa *job class* tersebut diharapkan dapat saling bekerjasama seusai situasi dan berdasarkan *job class* nya yang bertujuan agar *player* kalah(mati) atau tidak mendapatkan *experience* (xp).

1.1 Action-RPG

Action-Role Playing Game atau ARPG adalah bentuk sebuah subgenre yang didefinisikan dari *role-playing video game* yang menggabungkan unsur-unsur dari tindakan atau action-adventure game, menekankan tindakan real-time di mana pemain memiliki kontrol langsung pada karakter, bukan turn-based atau berbasis menu tempur. Permainan ini sering menggunakan sistem tempur yang mirip dengan *hack and slash* atau *game shooter*.

1.2 Jaringan Syaraf Tiruan

Jaringan syaraf adalah merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses

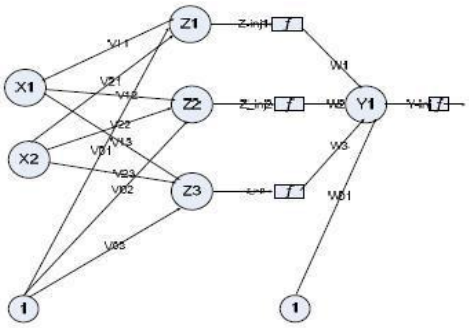
pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran.

1.3 Supervised Learning

Metode pembelajaran pada jaringan syaraf disebut terawasi jika keluaran yang diharapkan telah diketahui sebelumnya, misalkan seperti pada proses AND. Pada proses pembelajaran, satu pola masukan akan diberikan ke satu neuron pada lapisan masukan . Pola ini akan dirambatkan di sepanjang jaringan syaraf hingga sampai ke neuron pada lapisan keluaran. Lapisan keluaran ini akan membangkitkan pola keluaran yang nantinya akan dicocokkan dengan pola keluaran targetnya. Apabila terjadi perbedaan antara pola keluaran hasil pembelajaran dengan pola target, maka akan muncul error. Apabila nilai error masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi.

1.4 Backpropagation

Salah satu arsitektur jaringan saraf tiruan yang sangat populer adalah *multilayer feedforward networks*. Secara umum, jaringan seperti ini terdiri dari sejumlah unit neuron sebagai lapisan masukan, satu atau lebih lapisan simpul-simpul neuron komputasi lapisan tersembunyi, dan sebuah lapisan simpul-simpul neuron komputasi keluaran. Sinyal masukan dipropagasikan ke arah depan (arah lapisan keluaran), lapisan demi lapisan. Jenis jaringan ini adalah hasil generalisasi dari arsitektur *perceptron* satu lapisan, jadi biasa disebut sebagai *multilayer perceptron* (MLPs). *Error back propagation* adalah algoritma MLPs yang menggunakan prinsip pembelajaran terawasi. Propagasi balik (ke arah lapisan masukan) terjadi setelah jaringan menghasilkan keluaran yang mengandung *error*. Pada fase ini seluruh bobot *synaptic* (yang tidak memiliki aktivasi nol) dalam jaringan akan disesuaikan untuk mengkoreksi/memperkecil *error* yang terjadi (*error correction rule*). Untuk pelatihan jaringan, pasangan fase propagasi ke depan dan balik dilakukan secara berulang untuk satu set data latihan, kemudian diulangi untuk sejumlah *epoch* (satu sesi lewatan untuk seluruh data latihan dalam sebuah proses pelatihan jaringan) sampai *error* yang terjadi mencapai batas kecil toleransi tertentu atau nol. Jaringan *backpropagation* dapat dilihat pada Gambar 1.1



Gambar 1. Arsitektur Multi Layer Perceptron

1.5 Tahapan Algoritma Neural Network Backpropagation

Langkah 0 :

Pemberian inisialisasi penimbang (diberi nilai kecil secara acak)

Langkah 1 :

Jika kondisi berhenti tidak tercapai, Ulangi langkah 2 hingga 9 sampai kondisi akhir iterasi dipenuhi(kondisi berhenti jika perulangan mencapai maksimal perulangan atau *mean square error (MSE)* sudah lebih kecil dari rasio pembelajaran)

Langkah 2 :

Untuk masing-masing pasangan data pelatihan (training data) lakukan langkah 3 hingga 8 Propagasi maju (Feedforward)

Langkah 3 :

Masing-masing unit masukan (X_i, I = 1,…,n) menerima sinyal masukan X_i dan sinyal tersebut disebarkan ke unit-unit bagian berikutnya (unit-unit lapisan tersembunyi)

Langkah 4 :

Masing-masing unit dilapisan tersembunyi dikalikan dengan faktor penimbang dan dijumlahkan serta ditambahkan dengan biasnya :

$$Z_{in_j} = V_{oj} + \sum_{I=1}^{n} X_i V_{ij} \qquad (1)$$

Kemudian menghitung sesuai dengan fungsi aktifasi yang digunakan :

$$Z_j = f(Z_{in_j}) \qquad (2)$$

Bila yang digunakan adalah fungsi sigmoid maka bentuk fungsi tersebut adalah :

$$Z_j = \frac{1}{1+exp^{-z_in_j}} \qquad (3)$$

Masing-masing unit keluaran (Y_k, k=1,2,3,..m) dikalikan dengan faktor penimbang dan dijumlahkan :

$$Y_{in_k} = W_{ok} + \sum_{j=1}^{p} Z_j W_{jk} \qquad (4)$$

Menghitung kembali sesuai dengan fungsi aktifasi

$$y_k = f(y_{in_k}) \qquad (5)$$

Langkah 6 :

Masing-masing unit keluaran (Y_k, k=1,…,m) menerima pola target sesuai dengan pola masukan saat pelatihan dan dihitung galatnya :

$$\delta_k = y_k(1 - y_k) \qquad (6)$$

Menghitung perbaikan faktor penimbang (kemudian untuk memperbaiki w_{jk}

$$\Delta W_{kj} = \alpha . \delta_k . Z_j \qquad (7)$$

Menghitung perbaikan koreksi :

$\Delta W_{0j} = \alpha . \partial_k$ (8)

Dan menggunakan nilai δ_k pada semua unit lapisan sebelumnya.

Langkah 7 :

Masing-masing penimbang yang menghubungkan unit-unit l;apisan keluaran dengan unit-unit pada lapisan tersembunyi $(Z_j, j = 1, \dots, p)$ dikalikan delta dan dijumlahkan sebagai masukan ke unit-unit lapisan berikutnya.

$\delta_in_j = \sum_{k=1}^{n} \delta_k W_{jk}$ (9)

Selanjutnya dikalikan dengan turunan dari fungsi aktifasinya untuk menghitung galat.

$\delta_j = \delta_in_j f'(y_in_j)$ (10)

Kemudian menghitung perbaikan penimbang (digunakan untuk memperbaiki V_{ij})

$\Delta V_{ij} = \alpha \delta_j X_i$ (11)

Kemudian menghitung perbaikan bias (untuk memperbaiki V_{oj})

$\Delta V_{oj} = \alpha \delta_j$ (12)

Memperbaiki penimbang dan bias

Langkah 8 :

Masing-masing keluaran unit (y_k, k=1,…,m) diperbaiki bias dan penimbangnya (j=0,…,p)

$W_{jk}(baru) = W_{jk}(lama) \Delta W_{jk}$ (13)

Masing-masing unit tersembunyi (Z_j, j:1,…,p) diperbaiki bias dan penimbangnya (j=0,…,n)

$V_{jk}(baru) = V_{jk}(lama) \Delta V_{jk}$ (14)

Langkah 9 :

Uji kondisi pemberhentian (akhir iterasi) pengujian pemberhentian dilakukan dengan menghitung nilai *error*. Jika mencari nilai *error* yang tidak memiliki nilai target maka rumus mencari nilai *error* adalah sebagai berikut :

$1 - (y)^2$ (15)

Jika sudah memiliki nilai target maka rumus mencari nilai *error* adalah sebagai berikut :

$\frac{1}{2} \sum k (Tk - Yk)^2$ (16)

2. ISI PENELITIAN

Mount & Blade adalah *game* dengan jenis *action*-rpg yang dikembangkan oleh *Taleworlds Entertaiment* dimana game tersebut menggunakan perangkat *desktop* atau *game console* untuk memainkannya. Pada game tersebut dimana pemain yang memainkan karakter utama dengan aksi secara langsung dan setiap karakter pemain maupun NPC memiliki karakteristik yang berbeda-beda dan dapat berkembang mengikuti jalannya permainan. Adapun aturan dalam simulasi permainan adalah sebagai berikut :

1. Pemain harus mengalahkan semua NPC *enemy* yang ada, jika semua NPC *enemy* mati atau melarikan diri atau karakter utama mati maka permainan selesai.
2. NPC *enemy* dapat melarikan diri jika keluar dari arena permainan.
3. NPC enemy memiliki 3 macam job class (Warior, Assassin, Archer).
4. Titik *respawn* NPC *enemy* dan karakter utama ditentukan oleh sistem.
5. Enemy dengan *job class warior* dan *assassin* memiliki jarak serang dekat.
6. Enemy dengan *job class archer* memiliki jarak serang jauh.
7. Terdapat 4 attribut utama pada setiap karakter yaitu *strength*, *agility*, *speed* dan *attack*.
8. Setiap poin *strength* menambah 1 poin *attack*.
9. Setiap poin *agility* menambah 0,5% *speed*.

Ukuran arena permainan yaitu 300x300 pixel dengan skala 1:3779.

2.1 Penentuan Input & Keputusan

Pada permainan ini sebuah keputusan yang diambil dipengaruhi oleh nilai dari karakter yang ada pada permainan sehingga data masukan untuk arsitektur jaringan yang digunakan adalah sebagai berikut :

1. Input 1 (x1) : nilai HP(*health point*) *npc warrior*
2. Input 2 (x2) : nilai HP(*health point*) *npc assassin*
3. Input 3 (x3) : nilai HP(*health point*) *npc archer*
4. Input 2 (x4) : nilai HP(*health point*) *player*
5. Input 3 (x5) : nilai jarak *npc warrior* menuju *player*
6. Input 4 (x6) : nilai jarak *npc assassin* menuju *player*
7. Input 5 (x7) : nilai jarak *npc archer* menuju *player*

Penentuan keputusan yang akan di buat berdasarkan referensi taktik roma. Adapun beberapa taktik yang digunakan oleh roma saat berperang seperti berikut:

a. Pertarungan akan di mulai oleh prajurit jarak jauh (Pemanah), mereka mulai menyerang saat musuh berada pada jarak 18 meter.
b. Ketika musuh sudah mencapai 2 meter maka prajurit dengan gladius (senjata jarak dekat) mulai menyerang dan prajurit dengan jarak jauh mundur.
c. Bila musuh akan menggunakan javelin(senjata jarak jauh) maka akan dibentuk formasi bertahan dengan barisan prajurit yang lebih solid biasanya membawa sebuah perisai untuk melindungi pasukan di belakangnya.
d. Prajurit bertahan juga melindungi prajurit yang lebih lemah saat musuh yang lebih kuat menyerang.

Maka dari itu dibuatlah beberapa kondisi keputusan menyerupai taktik di atas dalam pengambilan keputusan yang ada sebagai berikut:

Tabel 1. Keputusan

Keputusan	Warior	Assasin	Archer
1	Escape	Escape	Escape
2	Escape	Escape	Attack
3	Attack	Attack	Hold
4	Attack	Hold	Hold
5	Hold	Attack	Hold
6	Hold	Hold	Attack

Berikut adalah contoh data pelatihan:

Tabel 2. Data Pelatihan

No	Data Input							Target Output
	X1	X2	X3	X4	X5	X6	X7	O1
1	100	100	100	100	33	31	35	0.92
2	100	100	100	100	25	24	28	0.92
3	100	100	100	100	19	18	22	0.92
4	100	100	100	100	13	11	15	0.92
5	100	100	100	95	6	6	10	0.75
6	100	70	100	90	2	1	8	0.75
7	100	70	100	90	2	1	8	0.75
8	100	40	100	80	3	2	7	0.75
9	100	30	100	85	3	2	7	0.58
10	100	30	100	85	2	1	8	0.58
11	80	30	100	70	1	3	9	0.58
12	60	30	100	50	2	2	10	0.58
13	40	30	100	45	2	3	5	0.42
14	40	30	70	45	3	3	1	0.42
15	40	30	70	40	3	2	2	0.42
16	40	30	40	40	2	3	1	0.42
17	40	0	100	40	1	0	11	0.25
18	0	0	100	40	1	0	9	0.25
19	0	0	100	40	0	0	6	0.25
20	0	0	100	40	0	0	2	0.25
21	0	0	70	40	0	0	1	0.08
22	0	0	40	40	0	0	1	0.08
23	0	0	10	40	0	0	1	0.08
24	0	0	0	40	0	0	0	0.08

2.2 Nilai Treshold Keputusan

Pada nilai output terdapat nilai untuk menentukan keputusan *enemy*. Nilai normalisasi keputusan didapatkan menggunakan persamaan. Nilai yang ada dapat dilihat pada tabel 3.6

Tabel 3. Treshold nilai keputusan

Warior	Assasin	Archer	Nilai Keputusan	Nilai normalisasi keputusan	Treshold	Klasifikasi Kerjasana
Escape	Escape	Escape	1	0.16	0-0.16	All Retreat
Escape	Escape	Attack	2	0.33	0.17-0.33	Defend Retreat
Attack	Attack	Hold	3	0.5	0.34-0.5	Full Attack
Attack	Hold	Hold	4	0.66	0.51-0.66	Warrior Attack
Hold	Attack	Hold	5	0.83	0.67-0.83	Assassin Attack
Hold	Hold	Attack	6	1	0.84-1	Archer Attack

2.3 Pelatihan & Pengujian Neural Network Backpropagation

Berikut adalah nilai bobot yang digunakan :

Tabel 4. Nilai random pada jaringan input

Input	Hidden 1	Hidden 2	Hidden 3
1	0.7195	0.6621	0.96
2	0.4957	0.6251	0.3561
3	0.7862	0.81	0.2136
4	0.1623	0.2011	0.1116
5	0.5151	0.4327	0.6204
6	0.1756	0.4083	0.564
7	0.4051	0.3564	0.294
bias	0.5469	0.8274	0.9169

Tabel 5. Nilai random pada bobot jaringan hidden

Hidden	Output
1	0.6807
2	0.9666
3	0.647
bias	0.2964

Berikut adalah nilai input yang digunakan:

Tabel 6. Nilai Input

Input	Keterangan	Nilai
x1	Nilai HP Warior	100
x2	Nilai HP Assassin	30
x3	Nilai HP Archer	100
x4	Nilai HP Player	85
x5	Nilai jarak warior ke player	2
x6	Nilai jarak assassin ke player	1

x7	Nilai jarak archer ke player	8

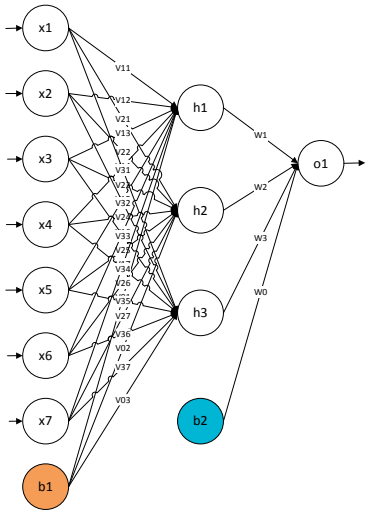
Berikut adalah nilai input yang telah di normalisasi:

Tabel 7. Nilai normalisasi input

Input	Nilai Normalisasi
x1	1
x2	0.3
x3	1
x4	0.85
x5	0.067
x6	0.033
x7	0.026

Parameter *neural network backpropagation* yang digunakan adalah sebagai berikut:
Learning Rate : 0.5
Minimal Error : 0.0001

Dari nilai input tersebut arsitektur yang digunakan untuk *neural network backpropagation* adalah sebagai berikut:



Gambar 2. Arsitektur *neural network*

Dari hasil perhitungan yang dilakukan menggunakan parameter dan nilai input di atas didapatkan hasil 0.065639486582918 dari *error output* yang dihasilkan. Karena error yang dihasilkan masih terlalu besar dari target *error* yang ditetapkan sebesar 0,0001 dan nilai output yang dihasilkan jauh dari nilai target yang ditetapkan maka akan dilakukan proses pembelajaran dengan menggunakan metode *backpropagation* untuk merubah bobot jaringan yang ada, selanjutnya dilakukan proses perhitungan *backpropagation* untuk menghasilkan nilai output mendekati atau sama dengan nilai target yang ditetapkan dan untuk meminimalkan nilai *error* dari *output* yang dihasilkan.

2.4 Pengujian K-Fold Cross Validation

Berdasarkan hasil skenario pengujian 1 sampai 10 yaitu pengujian data uji sebanyak 330 yang didapat dari 10 kali permainan yang dilakukan oleh pemain.

Tabel 8. Jumlah Data Pengujian

Data	Jumlah Data
A1	33
A2	33
A3	33
A4	33
A5	33
A6	33
A7	33
A8	33
A9	33
A10	33

Tabel 9. Skenario Pengujian

Skenario pengujian	Data pengujian	Data *training*
Pengujian 1	A1	A2, A3, A4, A5, A6, A7, A8, A9, A10
Pengujian 2	A2	A1, A3, A4, A5, A6, A7, A8, A9, A10
Pengujian 3	A3	A1, A2, A4, A5, A6, A7, A8, A9, A10
Pengujian 4	A4	A1, A2,A3, A5, A6, A7, A8, A9, A10
Pengujian 5	A5	A1, A2, A3, A4, A6, A7, A8, A9, A10
Pengujian 6	A6	A1, A2, A3, A4, A5, A7, A8, A9, A10
Pengujian 7	A7	A1, A2, A3, A4, A5, A6, A8, A9, A10
Pengujian 8	A8	A1, A2,A3, A4, A5, A6, A7, A9, A10
Pengujian 9	A9	A1, A2, A3, A4, A5, A6, A7, A8,A10
Pengujian 10	A10	A1, A2, A3, A4, A5, A6, A7, A8, A9

Scenario pengujian ke-1 adalah pengujian menggunakan parameter jumlah perceptron pada hidden layer dari 1-10. Berikut adalah hasil dari pengujian yang didapatkan

Tabel 10. Perbandingan nilai akurasi *hidden layer*

Jumlah *hidden layer*	Akurasi

1 *hidden layer*	55.75%
2 *hidden layer*	74.84%
3 *hidden layer*	94.54%
4 *hidden layer*	94.54%
5 *hidden layer*	93.93%
6 *hidden layer*	95.45%
7 *hidden layer*	95.15%
8 *hidden layer*	96.06%
9 *hidden layer*	95.45%
10 *hidden layer*	95.75%

Scenario pengujian ke-2 adalah pengujian dengan menggunakan parameter learning rate yang berbeda. Parameter learning rate yang digunakan pada pengujian ini yaitu bernilai dari 0.1 – 1. Berikut adalah hasil pengujian yang didapatkan

Tabel 11. Perbandingan nilai akurasi *learning rate*

Learning Rate	Akurasi
0.1	94.84%
0.2	94.54%
0.3	95.75%
0.4	96.36%
0.5	96.06%
0.6	95.75%
0.7	95.75%
0.8	95.45%
0.9	95.15%
1	94.84%

Scenario pengujian ke-3 adalah pengujian dengan menggunakan parameter nilai error yang berbeda. Parameter nilai error yang digunakan pada pengujian ini yaitu bernilai dari 0.1 – 0.00001. Berikut adalah hasil pengujian yang didapatkan

Tabel 12. Perbandingan nilai akurasi nilai *error*

Nilai *Error*	Akurasi
0.1	94.84 %
0.01	95.15 %
0.001	96.06 %
0.0001	96.36 %
0.00001	96.66 %

Berdasarkan hasil skenario pengujian 1 sampai 10 yaitu pengujian data uji sebanyak 330 yang didapat dari 10 kali permainan yang dilakukan oleh pemain, dapat ditarik kesimpulan bahwa jumlah perceptron pada hidden layer mempengaruhi tingkat error yang dihasilkan pada arsitektur jaringan yang digunakan, pada tabel 4.53 dapat dilihat hasil pengujian cross validation yang dilakukan, pada pengujian yang dilakukan menggunakan kombinasi parameter 8 perceptron pada hidden layer, learning rate 0.4 dan nilai error 0.00001 memiliki akurasi 96.66%, nilai akurasi tersebut menandakan besarnya tingkat kerjasama yang benar dari prediksi kerjasama yang di hasilkan oleh model yang digunakan.

2.5 Pengujian Kuesioner

Pengujian dilakukan dengan memperlihatkan game sebelum dan sesudah diimplementasikan algoritma neural network backpropagation setelah itu diberikan kuisioner pada responden tersebut. Kuisioner terdiri dari 3 pertanyaan dan menggunakan skala jawaban 1 sampai 3 yang diajukan terhadap responden, sedangkan hasilnya dilampirkan. Berikut adalah pertanyaan dari kuisioner tersebut:

a) Apakah prilaku kerjasama antara npc musuh terlihat setelah diimplementasikan algoritma backpropagation pada game tersebut?

b) Apakah setelah diimplementasikan algoritma backpropagation game menjadi lebih sulit?

c) Apakah dengan adanya prilaku kerjasama pada npc musuh game menjadi lebih baik?

Berikut adalah kategori jawaban kuesioner:

Tabel 13. Kategori Jawaban Kuesioner

Nilai Jawaban	Keterangan
3	Ya
2	Sama Saja
1	Tidak

Tabel 14. Skor Kategori Jawaban

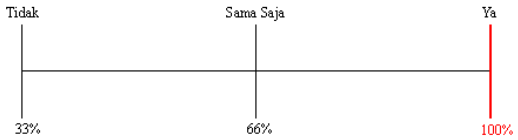
Angka	Keterangan
0%-33%	Tidak
34%-66%	Sama Saja
67%-100%	Ya

Dari pertanyaan no 1 yang diajukan kepada responden menghasilkan jawaban sebagai berikut :

Tabel 15. Pengolahan pertanyaan pertama

Nilai Jawaban	Keterangan	Responden	Total Skor Responden
2	Ya	10	30
1	Sama Saja	0	0
0	Tidak	0	0
	Jumlah	10	30

Jumlah skor ideal untuk seluruh item 3 x 10 = 30 (Ya). Jumlah skor rendah 1 x 10 = 10 (Tidak). Jadi berdasarkan data itu maka tingkat apakah terlihat perilaku kerjasama pada game yang dibuat adalah (30:30) x 100% = 100%.



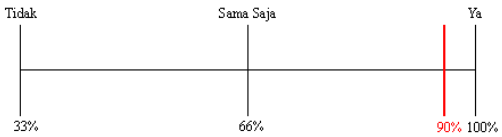
Gambar 3. Skala pengolahan pertanyaan pertama

Dari pertanyaan no 2 yang diajukan kepada responden menghasilkan jawaban sebagai berikut :

Tabel 16. Pengolahan pertanyaan kedua

Kategori Jawaban	Keterangan	Responden	Total Skor Responden
3	Ya	8	24
2	Sama Saja	1	2
1	Tidak	1	1
	Jumlah	10	27

Jumlah skor ideal untuk seluruh item 3 x 10 = 30 (Ya). Jumlah skor rendah 1 x 10 = 10 (Tidak). Jadi berdasarkan data tersebut maka tingkat apakah game menjadi lebih sulit adalah (27:30) x 90% = 100%.



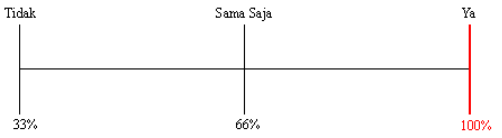
Gambar 4. Skala pengolahan pertanyaan kedua

Dari pertanyaan no 3 yang diajukan kepada responden menghasilkan jawaban sebagai berikut :

Tabel 17. Pengolahan pertanyaan ketiga

Kategori Jawaban	Keterangan	Responden	Total Skor Responden
3	Ya	10	30
2	Sama Saja	0	0
1	Tidak	0	0
	Jumlah	10	30

Jumlah skor ideal untuk seluruh item 3 x 10 = 30 (Ya). Jumlah skor rendah 1 x 10 = 10 (Tidak). Jadi berdasarkan data tersebut maka tingkat apakah game lebih baik dengan adanya perilaku kerjasama pada game yang dibuat adalah (30:30) x 100% = 100%.



Gambar 5. Skala pengolahan pertanyaan ketiga

Berdasarkah hasil pengujian beta dapat disimpulkan bahwa terlihatnya perilaku kerjasama pada game yang telah diimplementasikan dengan algoritma *neural network backpropagation*, juga dengan adanya perilaku kerjasama pada npc musuh membuat pemain lebih sulit untung mengalahkan npc musuh dan membuat game menjadi lebih baik.

3. PENUTUP

3.1 Kesimpulan

Berdasarkan hasil yang didapat dalam penelitian dan penyusunan skripsi ini serta disesuaikan dengan tujuan, maka dapat diperoleh kesimpulan bahwa :

Implementasi algoritma *neural network backpropagation* untuk perilaku npc berkelompok di game *action*-rpg dapat terlihat prilaku kerjasama antar npc musuh yang digunakan untuk memprediksi keputusan yang harus diambil untuk memilih pola kerjasama yang sesuai dengan data latih yang digunakan. Dari hasil penelitian yang dilakukan didapatkah hasil dengan parameter yang digunakan 8 perceptron pada *hidden layer*, *learning rate* 0.4 , nilai batas error 0.00001 memiliki nilai akurasi yang paling tinggi dibandingkan model arsitektur jaringan lainnya. Nilai akurasi ini memperlihatkan bawah data yang diuji memiliki nilai yang paling mendekati nilai dari data *output* pelatihan dan memiliki tingkat akurasi prediksi pengklasifikasian keputusan sebesar 96.66% juga dalam hasil kuisioner kepada 10 responden yang telah dilakukan 100% responden menyatakan kerjasama dalam simulasi game yang telah diimplementasikan algoritma BPNN dapat terlihat. Dengan akurasi yang cukup baik dapat disimpulkan bahwa npc musuh dapat bekerjasama sesuai dengan pola data latih yang digunakan dan secara objektif perilaku kerjasama dapat terlihat, hal ini membuat permainan lebih baik dan menantang.

3.2 Saran

Berdasarkan hasil dari penelitian yang telah dilakukan, maka dapat diberikan saran-saran untuk pengembangan selanjutnya yaitu :

1. Untuk pengembangan lebih lanjut parameter input yang digunakan dapat lebih beragam agar dapat menyesuaikan perkembangan game bergenre *action*-rpg saat ini.
2. Implementasi neural network pada npc musuh sebaiknya tidak sebatas untuk pertarungan saja.

DAFTAR PUSTAKA

[1] R. P. Kristo, N. H. Rini dan M. M.Azis, “Implementasi Logika Fuzzy Untuk Mengatur Perilaku Musuh dalam Game,” *Jurnal EECCIS*, vol. 7, no. 1, pp. 15-20, Juni 2013.T. Callahan, J. Hauser, and J. Wawrzynek, “The Garp Architecture and C Compiler,” Computer, vol. 33, no. 4, pp. 62-69, 2000.

[2] Kemdikbud, "Kamus Besar Bahasa Indonesia (KBBI)," [Online]. Available: http://kbbi.web.id/kerja. [Accessed 5 Januari 2017].
[3] J. A. Freeman and D. M. Skapura, Neural Networks, Houston: Addison-Wesley Publishing Company, 1991.
[4] J. Heaton, "Heaton Research," April 2012. [Online]. Available: http://www.heatonresearch.com/book/introduction-neural-network-math.html. [Accessed November 2016].
[5] J. Schneider, "Cross Validation," 1997. [Online]. Available: https://www.cs.cmu.edu/~schneide/tut5/node42.html.
[6] T. Entertainment, "Mount & Blade : Warband," Paradox Interactive, 2010.