Pembuatan *Survival Action Game* Dengan *Non-Player Character* Berbasis *Neural Network*

Semuel Yootje Daniel Tawas[1], Gregorius Satia Budhi[2], Kristo Radion Purba[3]
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jl. Siwalankerto 121 – 131, Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-mail: samueltawas@yahoo.co.id[1], greg@petra.ac.id[2], kristo@petra.ac.id[3]

ABSTRAK

Game merupakan aplikasi multi media yang membutuhkan beberapa unsur utama yaitu, tujuan, tantangan dan *game play* atau biasa dikenal sebagai aturan main dalam sebuah *game*.

Survival action game akan menerapkan sistem *game play* yang sifatnya *real time*. Sifat *game play* yang *real time* biasanya membutuhkan metode pengambilan keputusan yang cepat sehingga tidak mengganggu sistem *game play* yang sedang berjalan.

Salah satu yang metode yang cepat dalam proses pengambilan keputusan adalah *backpropagation*. Penggunaan metode *backpropagation* dapat memanfaatkan fitur perubahan *data training* sehingga *output* yang dikalkulasi dapat berbeda dari sebelum *data training* dirubah. Untuk bisa diterapkan ke dalam sistem *game*, *backpropagation* perlu menerapkan arsitektur *winner-take-all* sehingga sistem *backpropagation* yang dibuat, dapat menjalankan proses training dengan baik.

Berdasarkan hasil pengujian, sistem *backpropagation* dapat diterapkan dalam sistem *game* yang sifatnya *real time* dan fitur penambahan *data training* dapat memberikan perubahan berupa pengambilan keputusan yang berbeda dari sebelumnya.

Kata Kunci: Jaringan Saraf Tiruan, *Backpropagation*, Kecerdasan buatan, *Survival Action Game*.

ABSTRACT

Game is a multi media application that need some main parts such as, goal, challenge and game play or usually has been know as rules in a game.

Survival action game will apply real time system in its game play. Real time game usually needs fast decision making method for artificial intelligence, so it can't interrupt any system in the game.

Backpropagation is one of the fastest method that can be used to process data for decision making. Backpropagation method can use data training transformation feature so outputs from backpropagate calculation can be different than before. In its application, backpropagation needs to applying winner-take-all architecture so backpropagation system, can run the training process nicely.

Based on testing result, backpropagation system can be applied in real time game and data training addition feature can gives the transformation for making decision that can be different than before.

Keywords: *Neural Network, Backpropagation, Artificial Intelligence, Survival Action Game.*

1. PENDAHULUAN

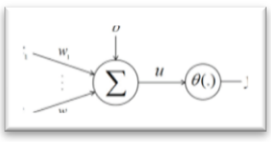
Artificial Intelligence atau kecerdasan buatan adalah kecerdasan yang dibuat dan dimasukan ke dalam suatu mesin atau *computer* supaya bisa melakukan pekerjaan yang bisa dikerjakan oleh manusia. Ada banyak implementasi *AI* dalam kehidupan sehari-hari akan tetapi, dalam penelitian ini akan fokus pada implementasi *AI* dalam *Real Time Survival-Action Game*. Untuk membuat *AI* dalam *Real Time Game* dibutuhkan metode pengambilan keputusan yang sangat cepat. Salah satu metode yang bisa digunakan adalah *Rule Based*. *Rule Based* adalah salah satu metode kecerdasan buatan dimana keputusan atau output yang diambil berdasarkan kondisi yang sedang dihadapi. Tingkat kesulitan dalam pembuatan *AI* dengan metode *Rule Based* tidak begitu tinggi, akan tetapi hasil yang didapat tidak terlalu memuaskan. Salah satu alasannya adalah ada beberapa *game* yang memiliki *AI* yang mudah diprediksi gerakannya karena tindakan yang diambil oleh *AI* tersebut terlihat monoton sehingga pemain atau *gamer* merasa bosan dengan *game* tersebut.

Untuk membuat *AI* yang memiliki gerakan yang sulit diprediksi, perlu dibuat sebuah sistem *AI* yang dapat mempelajari gerakan musuh dan sistem pengambilan keputusan yang sudah disesuaikan dengan gerakan musuh yang sudah disimpan. *Neural Network Backpropagation* adalah sistem yang akan digunakan dalam skripsi ini.

2. LANDASAN TEORI

2.1 Backpropagation

Backpropagation adalah struktur *neural network* yang memiliki 3 *layer* yaitu *input layer*, *hidden layer* dan *output layer*. Dimana tiap *layer* memiliki jumlah *node* sesuai dengan kebutuhan penggunannya. Jika diperiksa lebih dalam lagi tiap *node* memiliki beberapa *input signal*, *weight* dan *output signal* [2]. Dari komponen tersebut dapat diambil nilainya untuk melakukan proses perhitungan yang ada pada Gambar 1.

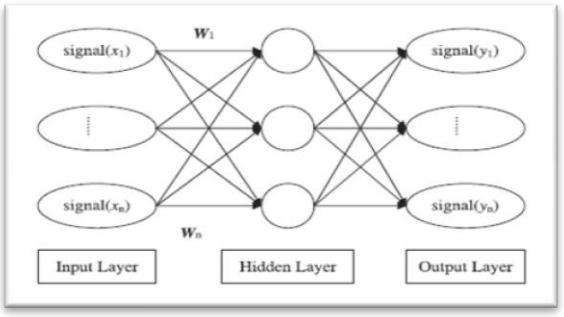


Gambar 1. Struktur node/neuron dalam backpropagation

Rumus yang digunakan dalam tiap *node* sesuai dengan persamaan 1.

$$y = \theta(\sum_{i=1}^{m} x_i w_i + b) \quad (1)$$

Dari hasil perhitungan tiap *node* akan menjadi inputan bagi *node-node* yang ada pada layer berikutnya. Contoh, hasil perhitungan dari input layer akan menjadi input bagi node yang ada pada hidden layer, dan output dari tiap node yang ada pada hidden layer akan menjadi input bagi node-node yang ada pada output layer [1]. Untuk lebih jelas lagi, ada pada Gambar 2**.**



Gambar 2. Struktur backprop

Setelah mendapatkan *output*-nya akan diperiksa apakah *output* yang diperoleh dari hasil perhitungan *feed-forward* sesuai dengan *training set* atau tidak. Jika sesuai, maka tidak akan terjadi perubahan nilai *weight* pada struktur *backprop*, tetapi jika hasilnya tidak sesuai dengan *output* yang ada pada *training set* maka akan dilakukan perubahan nilai weight atau biasa disebut *Weight Training* [3]. Dalam proses *training*, dibutuhkan *squared difference* yaitu nilai yang diperoleh dari selisih antara *desire output* dengan *actual output* [4]. Setelah itu, hasil *squared difference* diproses lebih lanjut dalam perhitungan mencari *gradient error*-nya. Rumus *gradient error* ada pada persamaan 2.

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p) \quad (2)$$

Tahap selanjutnya, menghitung *weight corrections*. Dimana rumus *weight corrections* ada pada persamaan 3.

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p) \quad (3)$$

Dari hasil persamaan 3, dapat digunakan untuk mengubah *weight* lama menjadi *weight* yang baru. Cara mengubah *weight* lama adalah menjumlahkan *weight* lama dengan *weight corrections* yang sudah dihitung [5]. Untuk lebih jelas lagi, lihat persamaan 4.

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \quad (4)$$

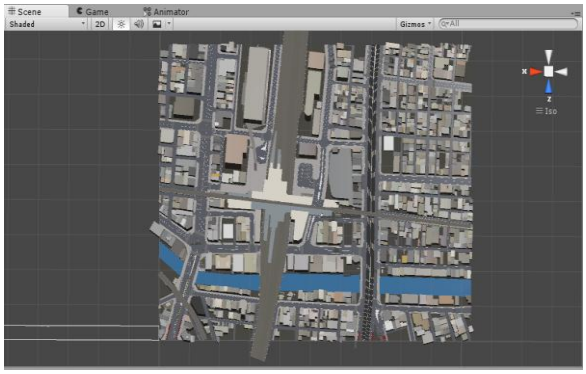
Proses *training* akan terus berlanjut hingga *actual output* sesuai dengan *desire output* yang ada pada *data training* yang diberikan [6].

2.2 Game Survival

Game Survival adalah salah satu *genre game*, dimana pemain harus bertahan dari setiap tantangan yang diberikan dalam *game*. Tantangan yang diberikan bisa berupa jumlah musuh yang terus bertambah dan terus berdatangan menyerang *player*, kondisi *health point* yang tidak mengalami *refreshing*, serta semakin lama *player* bertahan maka semakin kuat musuh yang harus dihadapi. *Game Survival* bisa menjadi *game* yang seru apabila, *programmer* atau *game* disainer sukses memiliki dan membuat 2 komponen utama yaitu *Design Gameplay* dan *Latar Belakang Cerita*. Contoh *game survival* ada pada Gambar 3.

Gambar 3. Contoh Game Survival Action, *Dead Island*.

Berikut tampilan *game survival* yang akan digunakan sebagai tempat implementasi *backpropagation*. Tampilan *game* ada pada Gambar 4 dan Gambar 5.



Gambar 4. Lokasi atau *environtment* yang akan digunakan

Berdasarkan Gambar 4, dapat terlihat bahwa *map* atau *arena* yang digunakan bersifat *low-poly*. *Mesh* yang *low-poly* dapat membantu optimasi dalam *game.*

Gambar 5. *Game* yang digunakan dalam implementasi *backpropagation*

Pada Gambar 5, telah ditunjukkan tampilan sementara *game* yang akan digunakan dalam implementasi *backpropagation*. Implementasi *backpropagation* diterapkan pada *non-player character* dalam *survival action game*.

3. DESAIN SISTEM

Dalam bagian desain sistem, akan membahas mengenai desain sistem *game survival* yang akan dibuat dan desain sistem *backpropagation*. Hubungan antara *backpropagation* dan *game survival* adalah *backpropagation* akan dipakai untuk *unit AI* dalam pengambilan keputusan di dalam *game survival*.

Sistem yang dibuat terdiri dari 3 bagian yang memiliki fungsi yang berbeda-beda. Sistem yang pertama adalah sistem *game survival*, dimana bagian sistem ini berfungsi sebagai tempat implementasi *backpropagation*. Sistem yang kedua adalah sistem *unit backpropagation* yang berperan untuk menggerakan *ai* dalam sistem pertama. Sedangkan sistem yang ketiga adalah sistem *perekaman gerakan player*. Sistem yang ketiga dibuat untuk mengubah dan menambah *data set* pada *backpropagation* sehingga *unit backpropagation* dapat memberi *input* yang cukup bervariasi.

4. PENGUJIAN SISTEM

Pada bab ini akan membahas mengenai pengujian terhadap sistem yang telah dibuat. Pengujian akan dilakukan pada dua sistem dalam satu aplikasi yaitu sistem *game survival action* dan sistem *backpropagation*. Pengujian yang dilakukan dalam sistem *game* adalah interaksi karakter *player* dalam *game*, sedangkan pengujian yang dilakukan dalam sistem *backpropagation* adalah hasil *training* setelah melakukan proses perekaman gerakan *player* atau penambahan *data training set* yang baru.

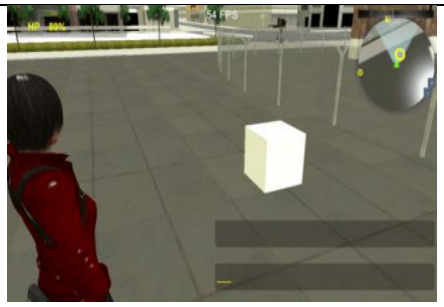
4.1. Pengujian Sistem Game

Pada subab ini akan membahas mengenai pengujian terhadap beberapa sistem *gameplay survival action game* yang telah dibuat. Pengujian yang dilakukan ada beberapa hal seperti sistem *quest*, sistem *inventory* dan beberapa interaksi yang dapat dilakukan *player* terhadap *ai* dalam *survival action game*.

4.1.1. Pengujian Sistem Quest dalam Game

Sistem *quest* adalah sistem yang mengatur tugas atau misi yang harus dilakukan *player* untuk menyelesaikan *survival action game*. *Quest* yang ada dalam *game*, didapat dari permintaan *non-player character*.Sistem *quest* yang diuji adalah pembacaan *trigger* dalam sistem.

Tabel Pengujian 1. Sistem *quest* dalam *game*

Kondisi	***Output* dalam *game***
Player belum mencapai *target* dalam *quest*	 *Output* untuk *quest* yang butuh mendekati objek tertentu.
	 *Output* untuk *quest* yang butuh membunuh musuh dalam *game*.  Dari sisi *UI current quest* masih ada dalam *quest list* dalam *game*.
Player sudah mencapai target dalam *quest*	 *Output* akan ditampilkan sehingga *player* dapat mengetahui bahwa *current quest* sudah selesai.  Dari sisi *UI current quest* sudah di-*destroy* dari dalam *quest list* dalam *game*.

Berdasarkan Tabel Pengujian 1, sistem *quest* dalam *survival action game* dapat menyesuaikan kesamaan objek dalam sistem dengan objek dalam *scene game* sehingga *player* tidak merasakan kebingungan dalam menjalankan *quest* dalam *game*. Selain menyesuaikan sistem *quest* dengan *scene*, dalam *game survival* ini juga menyesuaikan posisi *quest* yang aktif dengan posisi *icon quest* dalam *mini map* dalam *game*. *Mini map* dibuat agar *player* tidak bingung dalam mencari arah atau jalan menuju tempat tujuan atau *quest* yang sedang aktif.

4.1.2. Pengujian Sistem Inventory dalam Game

Sistem *inventory* adalah sistem yang mengatur objek-objek dalam *game* sehingga dapat dipakai oleh *player* dalam memenuhi kebutuhan dalam *game* yang dimainkan. Kesesuaian *ui inventory*

atau sistem *inventory* dengan objek yang digunakan atau dimiliki oleh *player* dalam *scene game* adalah bagian yang perlu diuji.

Tabel Pengujian 2. Sistem Inventory

Kondisi	***Output* dalam *game***
Objek dalam *inventory* belum dipakai atau diaktifkan	 Tampilan *ui inventory*  Tampilan *player* sebelum mengaktifkan objek yang tersedia dalam *inventory*
Objek dalam *inventory* sudah dipakai atau diaktifkan	 Tampilan *player* ketika mengaktifkan objek dalam *inventory*

Seperti yang dilihat pada Tabel Pengujian 2, Sistem *inventory* dapat menyesuaikan objek dalam *ui* dengan objek dalam *scene* sehingga *player* dapat dengan mudah menggunakan fungsi-fungsi yang ada dalam sistem *inventory*.

4.1.3. Pengujian Interaksi Player terhadap AI

Subab ini akan membahas mengenai pengujian yang dilakukan pada interaksi *player* terhadap *ai*. Interaksi *player* perlu diuji agar sistem perekaman gerakan *player* dapat dijalankan dengan baik. Selain itu, perlu adanya kesesuaian antara kondisi yang sedang dihadapi *player* dengan hasil perekaman yang disimpan oleh sistem *backpropagation*.

Karakter *player* dan karakter *non-player* yang dipakai dalam pengujian ini telah dilengkapi animasi yang sudah disesuaikan nilai parameternya dengan *desire output* yang ada pada sistem *backpropagation*.

Tabel Pengujian 3. Interaksi *player* dengan *ai*

Kondisi	***Output* dalam *game***
Player berinteraksi tanpa berhadapan dengan *ai* atau musuh. Jarak antara *player* dengan *enemy* berada diluar jarak pandang *enemy*.	 *Output* bernilai *null*. Jika *null* maka *output* dalam *game* akan ditampilkan dengan *string* "*Action for Nothing*".
Player berinteraksi dengan sedang berhadapan dengan *enemy* Jarak antara *player* dengan *enemy* berada dalam jangkauan jarak pandang *enemy* Hasil perekaman diambil berdasarkan *enemy* yang memiliki jarak terdekat dengan *player*.	 *Output* untuk *dodge*  *Output* untuk *Melee Attack*  *Output* untuk *Range Attack*

Berdasarkan Tabel Pengujian 3, Interaksi *player* dapat terbaca dengan baik oleh sistem perekaman gerakan dan *ai* dapat memberikan respon balik atau reaksi yang sesuai dengan aksi yang diberikan oleh *player*. Reaksi yang dimaksudkan adalah ketika *player* menyerang dan mengenai *enemy* maka setidaknya

enemy memberikan reaksi seperti memainkan animasi *hit*, animasi serang balik atau animasi *dodge*.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian pada bab sebelumnya, dapat diambil kesimpulan sebagai berikut:

- *Data training* yang sifatnya ambigu dapat mengganggu proses *training backpropagate*. Dampak dari *data training* yang ambigu adalah sistem *backprop* akan sulit mencapai standar yang diinginkan oleh *developer* karena *actual output* yang sama dengan *desire output* akan mengalami perubahan secara berkelanjutan.
- Sistem *backpropagation* yang menggunakan metode *non-real time training* dalam *game* memiliki kelebihan dan kekurangan. Kelebihannya adalah sistem *backpropagation* tidak akan menampilkan *output* yang tidak sesuai dengan *data training* karena setelah proses *training* benar-benar selesai dan memiliki kualitas yang baik maka, *unit-unit ai* yang menggunakan sistem *backpropagation* dapat mengirimkan *signal input* dan menerima *signal output*. Sedangkan kekurangannya adalah butuh waktu untuk menunggu sistem *backpropagation* dalam menyelesaikan proses *training*.
- Sistem *backpropagation* yang memiliki jumlah *data training* yang cukup banyak dapat mempengaruhi kinerja sistem *training backpropagate*. Semakin banyak *data training*-nya maka sistem *backprop* akan semakin lambat dalam melakukan proses *training*.
- Jumlah *data training* yang optimal untuk implementasi *survival action game* adalah sebanyak 50 data. Alasan memilih 50 data adalah sistem *AI* dalam *game* membutuhkan gerakan *AI* yang bervariasi, untuk itu dibutuhkan memori yang cukup untuk dapat menyimpan variasi gerakan yang diinginkan.
- Jumlah *hidden layer* dan jumlah *neuron* dalam *hidden layer* yang optimal untuk implementasi dalam *survival action game* adalah 1 *hidden layer* dan 15 *neuron*.

5.2 Saran

Beberapa hal yang dapat dijadikan saran dalam proses pengembangan selanjutnya antara lain:

- Dapat menambahkan fungsi seleksi terhadap *data training* yang sifatnya *ambigue*.
- Menambahkan fitur untuk menampilkan struktur *backprop* dalam sistem *game*.
- Menambahkan fitur seleksi *node input* dalam proses perekaman gerakan *player*, sehingga pada saat *player* memberi *gerakan* yang sifatnya tidak *adaptive* dapat dikelola dengan baik oleh sistem *backpropagation*.

6. DAFTAR PUSTAKA

[1] Dai, Qun. 2012. *Back-propagation with diversive curiosity: An automatic conversion from search stagnation to exploration*. *Jurnal Applied Soft Computing, Vol. 13*, pp. 483-495.

[2] Hrasko, Rafael., Pacheco, Andr´e G. C., Krohling, Renato A. 2015. *Time Series Prediction using Restricted Boltzmann Machines and Backpropagation*. *Procedia Computer Science, Vol. 55*, pp. 990-999.

[3] Jones, M. T. 2005. *AI Application Programming Second Edition*. Boston: Charles River Media.

[4] Khadse, C. B., Chaudhari, M. A., Borghate, V. B. 2016. *Conjugate gradient back-propagation based artificial neural network for real time power quality assessment*. *International Journal of Electrical Power & Energy Systems*, Vol. 82, pp. 197–206.

[5] Murru, N., Rossini, R. 2016. *A Bayesian approach for initialization of weights in backpropagation neural net with application to character recognition. Neurocomputing, Vol. 38*, pp. 4967–4971.

[6] Singh, Bikesh K., Verma, K., Thoke, A. S. 2015. *Adaptive gradient descent backpropagation for classification of breast tumors in ultrasound imaging*. *Procedia Computer Science, Vol. 46*, pp. 1601-1609.