

**PENERAPAN REINFORCEMENT LEARNING BERBASIS
HYBRID REWARD ARCHITECTURE PADA PERMAINAN
PERTARUNGAN**

TUGAS AKHIR

Diajukan sebagai syarat untuk menyelesaikan
Program Studi Strata-1 Informatika

Disusun Oleh:

Oka Hartanto

1114054



**INSTITUT
TEKNOLOGI
HARAPAN
BANGSA**

Veritas vos liberabit

**PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
2020**



LEMBAR PENGESAHAN

**PENERAPAN REINFORCEMENT LEARNING BERBASIS HYBRID REWARD
ARCHITECTURE PADA PERMAINAN PERTARUNGAN**



Disusun oleh:
Nama: Oka Hartanto
NIM : 1114054

Telah Disetujui dan Disahkan
Sebagai laporan Tugas Akhir Program Studi Informatika
Institut Teknologi Harapan Bangsa

Bandung, Juli 2020

Disetujui,

Pembimbing

Ventje Jeremias Lewi Engel, S.T., M.T.

NIK. 161019



PERNYATAAN HASIL KARYA PRIBADI

Saya yang bertanda tangan di bawah ini:

Nama : Oka Hartanto

NIM : 1114054

Dengan ini menyatakan bahwa laporan Tugas Akhir dengan Judul : ”**PENERAPAN REINFORCEMENT LEARNING BERBASIS HYBRID REWARD ARCHITECTURE PADA PERMAINAN PERTARUNGAN**” adalah hasil pekerjaan saya dan seluruh ide, pendapat atau materi dari sumber lain telah dikutip dengan cara penulisan referensi yang sesuai.

Pernyataan ini saya buat dengan sebenar-benarnya dan jika pernyataan ini tidak sesuai dengan kenyataan maka saya bersedia menanggung sanksi yang akan dikenakan pada saya.

Bandung, Juli 2020

Yang membuat pernyataan,

Oka Hartanto

ABSTRAK

Abstrak— Permainan video (Video Game) merupakan suatu media hiburan elektronik yang sekarang semakin berkembang. Saat ini terdapat banyak sekali tipe permainan yang beragam, sehingga masalah yang dihadapi pun berbeda, salah satunya adalah permainan pertarungan. Artificial Intelligence yang diimplementasikan pada permainan pertarungan harus memiliki kemampuan memutuskan aksi yang akan dilakukan dari sekian banyak aksi yang dimiliki oleh sebuah karakter, dan dengan respon waktu yang singkat. Pada umumnya algoritme yang digunakan pada permainan pertarungan ini masih menggunakan AI yang berdasarkan sistem rule-based, di mana tindakan mereka hanya ditentukan oleh berbagai atribut permainan, seperti koordinat karakter atau jumlah kerusakan yang ditimbulkan. Teknik reinforcement learning dengan Hybrid Reward Architecture pada permainan *Ms. Pacman* mampu memberikan skor tertinggi 25304 di atas skor pemain manusia sebesar 15693. Berdasarkan pengujian dan implementasi yang dilakukan, persentase kemenangan dari metode ini menghadapi manusia mencapai 80%.

Kata Kunci: *Reinforcement Learning*, Kecerdasan Buatan, Permainan Pertarungan, *Hybrid Reward Architecture*

ABSTRACT

Abstract — Video game is an electronic entertainment media that is now increasingly growing. Currently there are many different types of games, and the problems it faced are different, one of which is a fighting game. Artificial Intelligence that is implemented in a battle game must have the ability to decide which actions will be carried out from the many actions possessed by a character, and with a short response time. In general, the algorithm used in this battle game still uses AI based on a rule-based system, where their actions are only determined by various game attributes, such as character coordinates or the amount of damage inflicted. Reinforcement learning technique with Hybrid Reward Architecture on Ms. Pacman could achieve highest score of 25304 above human player score of 15693. Based on testing and implementation carried out, the percentage of wins from this method of facing human player reaches 80%.

Keywords: Reinforcement Learning, Artificial Intelligence, Fighting Game, Hybrid Reward Architecture

PEDOMAN PENGGUNAAN TUGAS AKHIR

Laporan tugas akhir yang tidak dipublikasikan terdaftar dan tersedia di Perpustakaan Institut Teknologi Harapan Bangsa, dan terbuka untuk umum dengan ketentuan bahwa hak cipta ada pada pengarang dan pembimbing Tugas Akhir. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan dengan seizin pengarang atau pembimbing Tugas Akhir dan harus disertai dengan ketentuan penulisan ilmiah untuk menyebutkan sumbernya.

Tidak diperkenankan untuk memperbanyak atau menerbitkan sebagian atau seluruh laporan tugas akhir tanpa seizin dari pengarang atau pembimbing Tugas Akhir yang bersangkutan.

KATA PENGANTAR

Terima kasih kepada Tuhan yang Maha Esa karena dengan bimbingan-Nya dan karunia-Nya penulis dapat melaksanakan Tugas Akhir yang berjudul "PENERAPAN REINFORCEMENT LEARNING BERBASIS HYBRID REWARD ARCHITECTURE PADA PERMAINAN PERTARUNGAN". Laporan ini disusun sebagai salah satu syarat kelulusan di Institut Teknologi Harapan Bangsa. Pada kesempatan ini penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Tuhan Yang Maha Esa, karena oleh bimbingan-Nya penulis selalu mendapat pengharapan untuk menyelesaikan tugas akhir ini.
2. Bapak Ventje Jeremias Lewi Engel, S.T., M.T., selaku pembimbing I Tugas Akhir yang senantiasa memberi dukungan, semangat, ilmu-ilmu, saran dan dukungan kepada penulis selama tugas akhir berlangsung dan selama pembuatan laporan tugas akhir ini.
3. Ibu Ken Ratri Retno W, S.Kom., M.T., selaku penguji I Tugas Akhir. Terima kasih atas dukungan, semangat, ilmu-ilmu, dan masukan yang telah diberikan kepada penulis dalam menyelesaikan Laporan Tugas Akhir ini
4. Ibu Ir. Inge Martina, M.T., selaku penguji II dalam Tugas Akhir Terima kasih atas dukungan, semangat, ilmu-ilmu, dan masukan yang telah diberikan kepada penulis dalam menyelesaikan Laporan Tugas Akhir ini.
5. Seluruh dosen dan staff Program Studi Informatika ITHB yang telah membantu dalam menyelesaikan Laporan Tugas Akhir ini.
6. Segenap jajaran staf dan karyawan ITHB yang turut membantu kelancaran dalam menyelesaikan Laporan Tugas Akhir ini.
7. Kedua orang tua tercinta yang selalu menyediakan waktu untuk memberikan doa, semangat dan dukungan yang tak habis-habisnya kepada penulis untuk menyelesaikan Laporan Tugas Akhir ini. Terima kasih untuk nasihat, masukan, perhatian, teguran dan

kasih sayang yang diberikan hingga saat ini.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna karena keterbatasan waktu dan pengetahuan yang dimiliki oleh penulis. Oleh karena itu, kritik dan saran untuk membangun kesempurnaan tugas akhir ini sangat diharapkan. Semoga tugas akhir ini dapat membantu pihak-pihak yang membutuhkannya.

Bandung, Juli 2020

Hormat Saya,

Oka Hartanto

DAFTAR ISI

LEMBAR PENGESAHAN	i
LEMBAR PERNYATAAN HASIL KARYA PRIBADI	ii
ABSTRAK	iii
ABSTRACT	iv
PEDOMAN PENGGUNAAN TUGAS AKHIR	v
KATA PENGANTAR	vi
DAFTAR ISI	x
DAFTAR TABEL	xiv
DAFTAR GAMBAR	xv
I PENDAHULUAN	1-1
1.1 Latar Belakang Masalah	1-1
1.2 Rumusan Masalah	1-3
1.3 Tujuan Penelitian	1-3
1.4 Batasan Masalah	1-3
1.5 Kontribusi Penelitian	1-3
1.6 Metodologi Penelitian	1-3
1.7 Sistematika Penulisan	1-4
II LANDASAN TEORI	2-1
2.1 Tinjauan Pustaka	2-1
2.1.1 <i>Artificial Intelligence pada permainan</i>	2-1
2.1.2 <i>Markov Decision Process (MDP)</i>	2-3
2.1.3 Jaringan Saraf Tiruan	2-3

2.1.4	<i>Hybrid Reward Architecture</i>	2-6
2.1.5	Perhitungan Persentase	2-8
2.2	Tinjauan Studi	2-9
2.3	Tinjauan Objek	2-11
2.3.1	<i>FightingGameICE</i>	2-11
2.3.2	<i>Game Fighting</i>	2-22
III ANALISIS DAN PERANCANGAN SISTEM		3-1
3.1	Analisis Masalah	3-1
3.2	Kerangka Pemikiran	3-2
3.3	Analisis Urutan Proses Global	3-3
3.3.1	Analisis Data Masukan	3-4
3.3.2	<i>Preprocessing</i>	3-5
3.3.3	<i>Hybrid Reward Architecture</i>	3-6
3.3.4	Analisis Manual	3-7
IV IMPLEMENTASI DAN PENGUJIAN		4-1
4.1	Lingkungan Pengembangan	4-1
4.1.1	Spesifikasi Perangkat Keras	4-1
4.1.2	Spesifikasi Perangkat Lunak	4-1
4.2	Implementasi Perangkat Lunak	4-1
4.2.1	Daftar <i>Class</i> dan <i>Method</i>	4-1
4.2.2	Implementasi Metode	4-20
4.3	Skenario Pengujian	4-21
4.3.1	AI vs AI Lawan Latihnya	4-22
4.3.2	Akurasi Serangan	4-37
4.3.3	AI vs Pemain Manusia	4-47
V PENUTUP		5-1
5.1	Kesimpulan	5-1
5.2	Saran	5-1

DAFTAR TABEL

2.1	Tabel Tinjauan Studi	2-9
2.1	Tabel Tinjauan Studi	2-10
2.2	Daftar <i>Method class FrameData</i> dari <i>FigthingICE</i>	2-13
2.3	Daftar <i>Method class CharacterData</i> dari <i>FigthingICE</i>	2-15
3.1	Data Uji yang Digunakan	3-8
3.2	Data Belajar yang Digunakan	3-9
3.3	Inisiasi data	3-9
3.4	Hasil perhitungan yang dimasukkan ke tabel	3-11
3.5	Hasil perhitungan hidden layer yang dimasukkan ke tabel	3-13
3.6	Hasil perhitungan <i>hidden layer 2</i> pada <i>head</i>	3-13
3.7	Tabel indeks Aksi	3-14
3.8	Tabel contoh <i>Batch</i>	3-14
3.9	Tabel contoh prediksi dan nilai <i>Q</i> yang diharapkan	3-15
3.10	Tabel nilai <i>loss function</i>	3-16
4.1	Daftar <i>Class</i>	4-2
4.2	Daftar atribut dari <i>class HRA_3heads</i>	4-2
4.3	Daftar <i>Method class HRA_3heads</i>	4-7
4.4	Daftar atribut dari <i>class HRA</i>	4-11
4.5	Daftar atribut dari <i>class Memory</i>	4-13
4.6	Daftar <i>Method class Memory</i>	4-13
4.7	Daftar atribut dari <i>class ReplayMemory</i>	4-15
4.8	Daftar <i>Method class ReplayMemory</i>	4-16
4.9	Daftar attribut dari <i>class Layer</i>	4-16
4.10	Daftar <i>Method class Layer</i>	4-17
4.11	Daftar atribut dari <i>class NetUtil</i>	4-18
4.12	Daftar <i>Method class NetUtil</i>	4-18
4.13	Daftar atribut dari <i>class Neuron</i>	4-19

4.14 Daftar <i>Method class Neuron</i>	4-20
4.15 Pengujian hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-22
4.16 Pengujian hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-23
4.17 Pengujian hasil latih 50 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-24
4.18 Pengujian hasil latih 50 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-24
4.19 Pengujian hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-25
4.20 Pengujian hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-26
4.21 Pengujian hasil latih 150 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-27
4.22 Pengujian hasil latih 150 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-28
4.23 Pengujian hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-29
4.24 Pengujian hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-30
4.25 Pengujian hasil latih 250 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-31
4.26 Pengujian hasil latih 250 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-31
4.27 Pengujian hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-32
4.28 Pengujian hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-33

4.29 Pengujian hasil latih 350 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-34
4.30 Pengujian hasil latih 350 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-35
4.31 Pengujian akurasi aksi hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-37
4.32 Pengujian akurasi aksi hasil latih 50 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-38
4.33 Pengujian akurasi aksi hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-38
4.34 Pengujian akurasi aksi hasil latih 50 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-39
4.35 Pengujian akurasi aksi hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-40
4.36 Pengujian akurasi aksi hasil latih 150 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-40
4.37 Pengujian akurasi aksi hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-41
4.38 Pengujian akurasi aksi hasil latih 150 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-41
4.39 Pengujian akurasi aksi hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-42
4.40 Pengujian akurasi aksi hasil latih 250 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-43
4.41 Pengujian akurasi aksi hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-43
4.42 Pengujian akurasi aksi hasil latih 250 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-44
4.43 Pengujian akurasi aksi hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-44

4.44 Pengujian akurasi aksi hasil latih 350 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)	4-45
4.45 Pengujian akurasi aksi hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-45
4.46 Pengujian akurasi aksi hasil latih 350 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)	4-46
4.47 Pengujian hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Zen)	4-47
4.48 Pengujian hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Garnet)	4-48
4.49 Pengujian hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Zen)	4-49
4.50 Pengujian hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Garnet)	4-50
4.51 Pengujian hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Zen)	4-51
4.52 Pengujian hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Garnet)	4-51
4.53 Pengujian hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Zen)	4-52
4.54 Pengujian hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Garnet)	4-53
A-1 Data Masukan untuk <i>Input Layer</i>	A-1

DAFTAR GAMBAR

2.1	Contoh dari <i>Reinforcement Learning</i> [11].	2-2
2.2	Contoh JST dengan 3 Lapisan [10]	2-4
2.3	Grafik Fungsi ReLU	2-5
2.4	Tampilan FightingICE	2-12
2.5	Daftar karakter FightingICE. Dari kiri ke kanan: Zen, Garnet, dan Lud	2-12
3.1	Kerangka Pemikiran	3-2
3.2	<i>Flowchart</i> Sistem Pembelajaran AI	3-3
3.3	Struktur MLP	3-8
4.1	Grafik persentase AI 3 unit karakter <i>Zen</i>	4-36
4.2	Grafik persentase AI 3 unit karakter <i>Garnet</i>	4-36
4.3	Grafik persentase AI 1 unit karakter <i>Zen</i>	4-36
4.4	Grafik persentase AI 1 unit karakter <i>Garnet</i>	4-37
4.5	Grafik persentase AI 3 unit karakter <i>Zen</i> terhadap pemain manusia	4-54
4.6	Grafik persentase AI 3 unit karakter <i>Garnet</i> terhadap pemain manusia	4-55

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Permainan video (*Video Game*) merupakan suatu media hiburan elektronik yang sekarang semakin berkembang [1]. Permainan muncul dari keinginan manusia untuk bermain dan dari kemampuan manusia untuk berpura-pura[2]. Untuk waktu yang lama industri permainan video telah berada di garis terdepan teknologi, dengan banyak inovasi dalam perangkat keras komputer, teknik *rendering*, dan kecerdasan buatan baik yang sengaja diciptakan untuk keperluan tertentu atau yang pertama kali digunakan dalam permainan video [1]. Karena perkembangan teknologi ini, evolusi *Artificial Intelligence* pada industri ini terlihat sangat signifikan. Pada awalnya AI dalam permainan video bukanlah sebuah AI, melainkan hanya sebuah algoritme yang memiliki pola tertentu untuk menjadi lawan main dalam permainan video tersebut. Salah satu kesuksesan pertama manusia adalah menciptakan AI pada permainan sederhana, karena dari awal sejarah perkembangan manusia, permainan digunakan sebagai uji coba terhadap kecerdasan [2].

Saat ini terdapat banyak sekali tipe permainan yang beragam, sehingga masalah yang dihadapi pun berbeda, salah satunya adalah *Fighting game*. *Fighting game* merupakan subgenre dari genre *Action game* walaupun tidak adanya aktifitas eksplorasi, tembak-menembak, maupun penyelesaian *puzzle*. Meski demikian, genre ini masih dianggap *Action games* karena genre ini membutuhkan kemampuan fisik yang sangat besar dari seorang pemain, seperti *reaction time* dan *timing* dari pemainnya [3]. *Artificial Intelligence* yang diimplementasikan pada *Fighting game* harus memiliki kemampuan memutuskan aksi yang dilakukan dari sekian banyak aksi yang dimiliki oleh sebuah karakter, dan dengan respon waktu yang singkat.

Pada umumnya algoritme yang digunakan pada permainan *fighting* ini masih menggunakan AI yang berdasarkan sistem *rule-based*, di mana tindakan mereka hanya ditentukan oleh berbagai atribut permainan, seperti koordinat karakter atau jumlah kerusakan yang ditimbulkan. Metode yang digunakan biasanya akan menghasilkan *behaviour* yang statis dan tidak beradaptasi terhadap strategi lawan, sehingga AI dengan kondisi yang seperti itu akan terlalu repetitif dan mudah diprediksi. Jika pemain mampu menghafalkan perilaku dari AI permainan tersebut, pemain akan kehilangan minatnya setelah bermain beberapa kali dikarenakan tantangan yang diberikan menjadi kurang menarik [4].

Beberapa metode yang digunakan untuk membuat AI pada permainan adalah K-NN, MCTS, *Reinforcement Learning* menggunakan *Q-learning* dan *Deep Q Network*. Kekurangan dari implementasi K-NN adalah nilai k yang digunakan berbeda-beda untuk setiap lawan yang memiliki kecenderungan melakukan perilaku yang berbeda dalam setiap permainan, sehingga

dibutuhkan nilai k yang dinamis untuk bisa digunakan dalam memprediksi serangan lawan yang berbeda-beda, sedangkan kelebihannya adalah jika data belajar yang dimiliki cukup tinggi, maka AI dapat menjadi lawan yang cukup tangguh untuk ditandingkan dengan AI kuat lainnya [4].

Sedangkan untuk metode MCTS, metode ini memiliki kelebihan di mana metode ini tidak bergantung pada *rule base* yang sudah ditetapkan sebelumnya atau pola aksi [5]. Penggunaan *Reinforcement Learning* dengan menggunakan *Neural Network* juga dapat menghasilkan yang cukup baik, dan dapat ditingkatkan dengan terus melakukan simulasi *training* sebanyak mungkin [2], walaupun pelatihan yang dilakukan masih sedikit, namun pada [6] agen *reinforcement learning* dapat persentase kemenangan sebesar 62% dengan menggunakan *Q-Learning*.

Penelitian yang berjudul *Hierarchical Reinforcement Learning with Monte Carlo Tree Search in Computer Fighting Game* [7] menggunakan metode reinforcement learning dengan Monte Carlo Tree Search, di mana pada umumnya menggunakan perencanaan MCTS melalui set opsi, pada penelitian ini menggunakan pencarian MCTS sebagai opsinya. Hasil persentasi kemenangan terhadap AI MCTS biasa terhadap 1000 ronde permainan berkisar 66% terhadap lawan tertentu. Jumlah pelatihan hingga ribuan ronde dan lawan berlatih hanya menggunakan pemenang kompetisi AI [4] tanpa mengikuti sertakan runner up lainnya mengakibatkan agen tersebut tidak dapat bekerja dengan baik saat melawan AI yang bukan lawan latihannya.

Pada penelitian yang berjudul *Hybrid Reward Architecture for Reinforcement Learning*, agen AI dibentuk menggunakan *Hybrid Reward Architecture* untuk memainkan permainan *Ms. Pac-Man*, yang mana HRA merupakan *reinforcement learning* berdasarkan *Deep-Q Network*. Yang membedakan antara HRA dengan DQN adalah pembagian fungsi *reward* dari *environment* ke dalam sejumlah n fungsi *reward* yang berbeda [8]. Jumlah fungsi *reward* pada penelitian tersebut adalah 4, dan mampu memberikan skor tertinggi 25304 di atas skor pemain manusia sebesar 15693. Kekurangan dari metode ini adalah membutuhkan jumlah fungsi *reward* yang banyak untuk meningkatkan performa.

Permainan yang digunakan dalam penelitian ini adalah *FightingICE*, sebuah *game fighting engine* yang dikembangkan secara khusus oleh Intelligent Computer Entertainment Lab., dari Ritsumeikan University, di Jepang [4]. FightingICE ini merupakan *fighting game* yang dapat dimainkan oleh 2 pemain, dan terdapat 3 data karakter yang dapat digunakan dengan 56 aksi pada setiap karakter. Aspek penting yang membedakan *platform* FightingICE dari *game fighting* lainnya adalah *platform* ini memiliki delay pada *state variables*. Hal tersebut dilakukan agar pemain AI yang bertarung tidak bereaksi dan memiliki reflek menyerupai manusia super ketika lawan memulai gerakannya, tetapi agar pemain AI dapat melakukan prediksi sehingga lebih menyerupai pemain manusia.

Berdasarkan penelitian [2] [4] [6] [7] [8] pada uraian di atas, saat ini dibutuhkan AI yang dapat beradaptasi dengan kemampuan pemain terutama pada permainan pertarungan,

sehingga penelitian ini akan menggunakan metode *Reinforcement Learning* dengan metode *Hybrid Reward Architecture*. Permainan pertarungan akan menjadi objek penelitian untuk membangun AI, karena masih minimnya penelitian tentang pembuatan AI yang dapat bermain permainan pertarungan[4] dan menggunakan *FightingICE*[9] sebagai sarana pengujian AI tersebut.

1.2 Rumusan Masalah

Berikut ini adalah rumusan masalah yang dibuat berdasarkan latar belakang di atas:

1. Pada ronde pelatihan berapa AI memiliki performa yang lebih baik?
2. Berapa persentase kemenangan yang didapat dengan menggunakan HRA ketika dihadapkan dengan AI lain dan pemain manusia?
3. Apakah akurasi serangan yang mengenai lawan berpengaruh terhadap kemenangan?
4. Apakah jumlah fungsi *reward* berpengaruh terhadap kinerja pelatihan AI?

1.3 Tujuan Penelitian

Berdasarkan batasan masalah di atas, berikut ini adalah tujuan penelitian dari tugas akhir ini:

1. Mengimplementasikan *Reinforcement Learning* pada pembuatan AI adaptif untuk *game fighting*.
2. Menganalisis kinerja *reward function* agen *Reinforcement Learning* ketika menghadapi lawan AI *rule-based* dan manusia.

1.4 Batasan Masalah

Berikut ini adalah batasan masalah dalam pembahasan dan pengembangan yang dilakukan:

1. Permainan pertarungan yang diuji hanya untuk permainan yang memiliki tipe 1 lawan 1.
2. Permainan pertarungan tidak memiliki objek senjata yang berserakan.
3. Dari 3 karakter tersedia hanya 2 yang digunakan untuk pengujian, yaitu karakter *ZEN* dan *GARNET*, dan 1 karakter untuk pelatihan yaitu *ZEN*.

1.5 Kontribusi Penelitian

Penelitian ini memberikan kontribusi di mana pengembangan sistem ini diharapkan dapat membangun sebuah AI adaptif dengan metode *Reinforcement Learning* yang dapat beradaptasi dengan pemain manusia. Dalam penelitian ini juga terdapat pengujian seberapa handal AI yang sudah dilatih dalam menghadapi AI dengan metode lain.

1.6 Metodologi Penelitian

Tahapan-tahapan yang akan dilakukan dalam pelaksanaan penelitian ini adalah sebagai berikut:

1. Studi kepustakaan

Tahap pertama penulisan ini adalah studi kepustakaan, yaitu mengumpulkan bahan referensi dari berbagai sumber seperti buku, jurnal, laporan penelitian ataupun situs-situs internet. Materi yang dicari dan dipelajari adalah mengenai pembelajaran mesin, *Reinforcement Learning, Hybrid Reward Architecture*.

2. Analisis dan perancangan

Dalam tahap ini, dilakukan perancangan sistem. Dimulai dari alur bisnis proses sistem, merancang sistem dengan menerapkan metode-metode yang ada.

3. Implementasi dan Pengujian

Program yang telah dibuat selanjutnya akan diimplementasikan dan diujikan menggunakan *game engine FightingICE*

4. Penyusunan Laporan

Dalam tahap ini, dilakukan penyusunan laporan terhadap hasil pengujian yang telah dilakukan, dan melakukan penarikan kesimpulan.

1.7 Sistematika Penulisan

Pada penelitian ini peneliti menyusun berdasarkan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang, identifikasi masalah, tujuan penelitian, manfaat penelitian, batasan masalah, metode penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisi teori dasar yang digunakan pada penyusunan laporan ini yang meliputi penjelasan mengenai *Reinforcement Learning* dan HRA.

BAB III ANALISIS DAN PERANCANGAN

Bab ini berisi perancangan sistem yang meliputi perancangan aplikasi “Penerapan *Reinforcement Learning* berbasis *Hybrid Reward Architecture* pada Permainan *Fighting*”.

BAB IV IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi implementasi dan analisis hasil penelitian terhadap sistem yang dibangun, apakah sesuai dengan tujuan yang diharapkan atau belum.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran dari seluruh kegiatan yang bisa digunakan sebagai masukan untuk pengembangan sistem informasi lebih lanjut yang

nantinya akan dikembangkan.

BAB II

LANDASAN TEORI

Pada bab ini akan dijelaskan mengenai teori dasar dan metode yang digunakan pada penelitian ini.

2.1 Tinjauan Pustaka

Pada bagian ini akan dijelaskan teori-teori terkait yang akan digunakan dalam aplikasi implementasi *Reinforcement Learning* dengan HRA.

2.1.1 *Artificial Intelligence pada permainan*

Artificial Intelligence (AI) dan permainan memiliki sejarah yang panjang bersama, di mana banyak dari penelitian mengenai AI untuk permainan berkaitan dengan membangun agen untuk bermain permainan, dengan atau tanpa komponen pembelajaran [10]. Kebanyakan penelitian mengenai *game-playing* AI berfokus kepada permainan papan klasik, seperti *Checkers* dan Catur. Semakin berkembangnya teknologi, kini muncul juga penelitian AI yang dilakukan pada permainan lain selain permainan papan. Pada permainan modern, AI juga digunakan untuk pembuatan konten dalam permainan, sehingga dapat memberikan pengalaman bermain yang berbeda setiap pemain memulai permainan tersebut. Baru-baru ini, penelitian tentang agen yang memberikan kesan dapat dipercaya dalam permainan telah membuka cakrawala baru dalam AI permainan. Dari kesimpulan di atas, maka ada 3 cara untuk mengimplementasikan AI ke dalam sebuah permainan video, yakni [10] :

1. AI dapat memainkan sebuah permainan.
2. AI dapat membuat konten sebuah permainan.
3. AI dapat memodelkan pemain pada sebuah permainan.

Berdasarkan poin pertama di atas, dan fokus utama dalam penelitian ini untuk membangun sebuah AI yang dapat memainkan sebuah permainan, maka dari itu terdapat metode umum yang dapat digunakan untuk membuat AI yang dapat memainkan permainan video [10]:

2.1.1.1 Pendekatan *Planning-based*

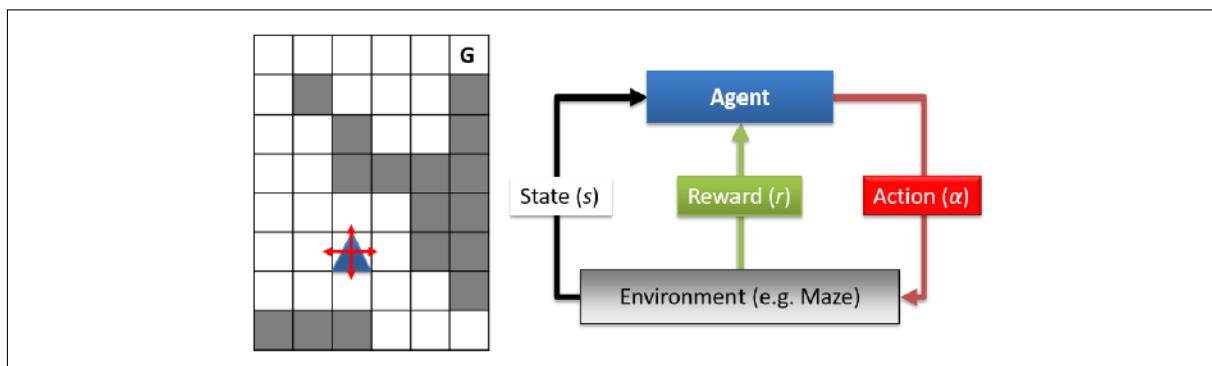
Algoritme ini memilih aksi melalui perencanaan dari serangkaian aksi di masa depan dalam ruang kondisi yang pada umumnya berlaku untuk permainan, dan secara umum tidak membutuhkan waktu pelatihan. Contoh pendekatan ini adalah metode klasik *Tree Search* seperti algoritme *Minimax*, *Alpha-Beta pruning*, *A**, metode *Stochastic Tree Search* seperti algoritme *Monte Carlo Tree Search*, metode *Evolutionary Planning* seperti implementasi algoritme *Physical Traveling Salesman Problem*, dan metode *Planning* dengan Representasi Simbolik.

2.1.1.2 Pendekatan *Reinforcement Learning*

Reinforcement Learning adalah sebuah pendekatan pembelajaran mesin yang terinspirasi oleh psikologi perilaku, khususnya cara manusia dan hewan yang belajar mengambil keputusan melalui hadiah, baik positif atau negatif, yang diterima oleh lingkungan mereka. Algoritme ini adalah setiap algoritme yang menyelesaikan sebuah masalah *reinforcement learning*. Algoritme ini mencakup algoritme-algoritme dari *temporal difference* atau keluarga dari *approximate dynamic programming*, aplikasi dari algoritme evolusioner untuk *reinforcement learning* seperti *neuroevolution* dan *genetic programming*, dan metode lainnya.

Dalam *Reinforcement Learning* [11], sampel dari perilaku baik pada umumnya tidak tersedia, melainkan serupa dengan pembelajaran evolusioner, sinyal pelatihan dari algoritme tersebut diberikan oleh lingkungan berdasarkan dengan bagaimana sebuah agen itu berinteraksi di lingkungan tersebut. Algoritme *Reinforcement learning* dapat diaplikasikan ke dalam permainan ketika adanya waktu untuk belajar. Biasanya waktu belajar ini sangat banyak, dikarenakan kebanyakan metode *reinforcement learning* akan butuh memainkan permainan sebanyak ribuan kali, bahkan jutaan kali, agar algoritme dapat memainkan permainan tersebut dengan baik.

Beberapa algoritme *reinforcement learning* membutuhkan sebuah *forward model* agar ketika algoritme tersebut selesai dilatih, *reinforcement-learned policy* dari model tersebut biasanya dapat dieksekusi dengan sangat cepat. Terdapat dua metode pada *reinforcement learning* yakni metode klasik dan metode evolusioner. Yang membedakan antara metode klasik dan metode evolusioner adalah dari perbedaan antara *ontogenetic*, yang mana agen melakukan pembelajaran semasa hidupnya, dan *phylogenetic*, yang mana agen melakukan pembelajaran diantara masa hidupnya. Contoh dari metode klasik *reinforcement learning* adalah algoritme *Neural Network* dan algoritme *Q-learning*. Contoh dari metode evolusioner adalah *genetic programming*.



Gambar 2.1 Contoh dari *Reinforcement Learning* [11].

Pada titik tertentu pada waktu t , agen berada pada keadaan tertentu dan memutuskan untuk mengambil tindakan a dari semua tindakan yang ada pada kondisi saat ini. Sebagai respon, lingkungan tersebut mengirim hadiah, r . Melalui serangkaian interaksi berkelanjutan antara agen dan lingkungannya, agen secara bertahap belajar untuk memilih tindakan yang

memaksimalkan jumlah hadiahnya.

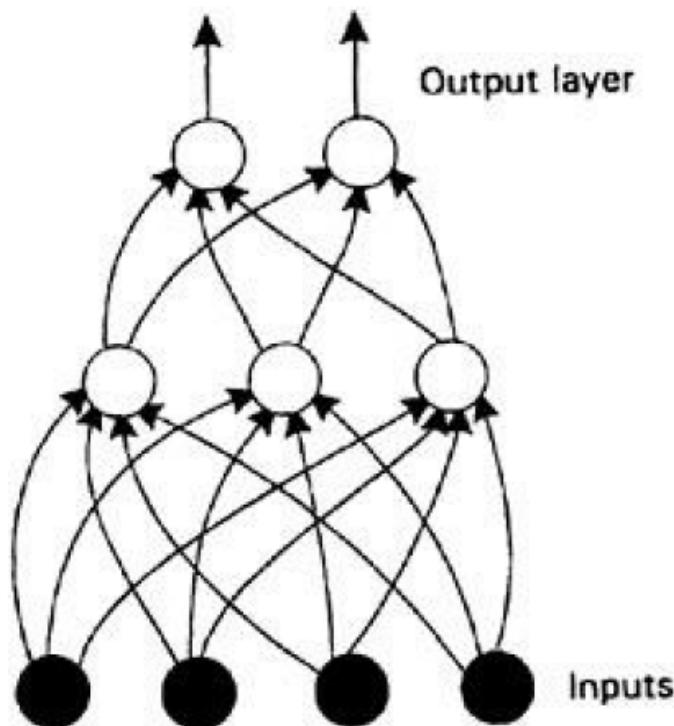
2.1.2 *Markov Decision Process (MDP)*

Secara formal, sasaran dari agen adalah untuk menemukan *policy* (π) untuk memilih tindakan yang memaksimalkan sebuah ukuran dari hadiah jangka panjang seperti hadiah kumulatif yang diharapkan. Sebuah *policy* adalah sebuah strategi yang agen ikuti dalam penentuan aksi, sesuai kondisinya saat ini. Jika fungsi yang memberikan ciri nilai dari setiap tindakan, baik yang sudah ada atau baru saja dipelajari, *policy* optimal (π) dapat diturunkan dari pemilihan aksi dengan nilai tertinggi. Interaksi dengan lingkungan terjadi pada langkah waktu diskrit ($t = \{0, 1, 2, \dots\}$) dan dimodelkan sebagai *Markov Decision Process (MDP)* [11]. MDP didefinisikan sebagai $M = (S, A, P, R, \gamma)$ di mana:

1. S : sebuah kumpulan dari kondisi $\{s_1, \dots, s_n\} \in S$. Keadaan dari lingkungan adalah sebuah fungsi dari informasi agen tentang lingkungan tersebut (contohnya *input* dari agen).
2. A : sebuah kumpulan dari tindakan $\{a_1, \dots, a_n\} \in A$. Aksi mempresentasikan cara yang berbeda - beda yang dapat agen lakukan pada lingkungan tersebut.
3. $P(s, s', a)$: probabilitas transisi dari s menuju s' jika diberikan a . P memberikan probabilitas akhiran pada kondisi s' setelah memilih aksi a pada kondisi s dan mengikuti *Markov property* yang menyatakan bahwa keadaan masa depan dari proses tergantung hanya pada keadaan saat ini, bukan pada urutan peristiwa yang mendahuluinya. Sebagai hasil, *Markov property* dari P memungkinkan memberikan prediksi dinamis 1 langkah.
4. $R(s, s', a)$: Fungsi hadiah pada transisi dari s menuju s' jika diberikan a . Ketika agen pada keadaan s memilih sebuah aksi a dan bergerak menuju kondisi s' , agen akan secara langsung menerima hadiah r dari lingkungan.
5. γ adalah sebuah faktor diskon untuk menyesuaikan kepentingan yang diberikan untuk hadiah langsung.

2.1.3 *Jaringan Saraf Tiruan*

Jaringan Saraf Tiruan (*Neural Network*) adalah salah satu pendekatan untuk perhitungan kepintaran dan pembelajaran mesin yang terinspirasi oleh jaringan saraf pada otak [12]. Sebuah Jaringan Saraf Tiruan adalah sekumpulan dari unit pemrosesan yang saling terhubung (*neuron*) yang dirancang untuk memproses informasi, mengoperasikan, mempelajari, dan melakukan beberapa tugas. Neuron buatan menyerupai neuron biologis di mana memiliki jumlah input x (menyerupai dendrit pada neuron) masing-masing memiliki parameter *weight* w (yang menyerupai kekuatan sinapsis). Dalam sebuah neuron juga memiliki unit pemrosesan yang menggabungkan input dengan masing-masing *weight*-nya dengan rumus *weighted sum* pada rumus 2.1 dan selanjutnya digunakan dalam fungsi aktivasi pada rumus 2.2.



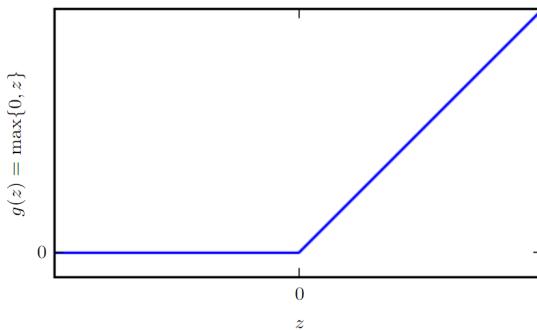
Gambar 2.2 Contoh JST dengan 3 Lapisan [12]

$$a_j = \sum_{i=1}^n (X_i \cdot W_{ji}) \quad (2 . 1)$$

Untuk membentuk sebuah Jaringan Saraf Tiuran, maka beberapa neuron harus terstruktur dan terhubung. Pada bentuknya yang paling sederhana, neuron pada sebuah JST dilapis oleh satu atau lebih lapisan, tetapi tidak terhubung dengan neuron lain pada lapisan yang sama yang disebut dengan *Multi-Layer Perceptron* (MLP). Terdapat lapisan yang dibagi menjadi 3 lapisan yaitu:

1. *Input layer* : terdiri dari node yang berisi data variable x . Semua *node* pada layer ini terhubung ke *node* pada *hidden layer*. Jika tidak terdapat *hidden layer* maka *node* ini dapat langsung terhubung dengan *output layer*.
2. *Hidden layer* : terdiri dari *node* yang menerima data dari *input layer*, dan akan meneruskannya ke *output layer*. Salah satu tujuan dari lapisan ini adalah untuk melakukan fungsi aktivasi sehingga data dapat diteruskan pada output.
3. *Output layer* : terdiri dari *node* yang menerima data dari lapisan sebelumnya.

Setelah diproses menggunakan persamaan pada rumus 2.1, selanjutnya dimasukkan ke dalam fungsi aktivasi. Fungsi aktivasi ini adalah sebuah abstraksi yang menyerupai saklar, di mana fungsi ini menentukan apakah data tersebut diaktifkan pada neuron tersebut atau tidak.



Gambar 2.3 Grafik Fungsi ReLU [14]

Pada umumnya fungsi aktivasi yang paling sering digunakan adalah fungsi *sigmoid*. Fungsi ini membatasi nilai pada interval 0 sampai 1. Namun pada penelitian ini fungsi aktivasi yang digunakan adalah *Rectifier Linear Unit* (ReLU). ReLU dipilih karena komputasinya lebih mudah dikerjakan sehingga pembelajaran agen dapat dilakukan dengan cepat, dan mengurangi kemungkinan gradien yang menghilang [11]. Rumus fungsi ReLU seperti berikut:

$$g_{a_j} = \max(0, a_j) \quad (2 . 2)$$

Sebelum melakukan penyesuaian bobot untuk mencari bobot yang tepat, maka perlu melakukan perhitungan nilai *error* yang nantinya digunakan untuk propagasi mundur. Maka dari itu, perubahan nilai bobot ini bertujuan untuk memperkecil nilai *loss function*. Berikut adalah rumus perhitungan *Squared Loss Function* [14] yang digunakan pada penelitian ini:

$$L(y, f(x)) = \frac{1}{2} [y - f(x)]^2 \quad (2 . 3)$$

Variable y adalah nilai ekspektasi atau nilai yang diharapkan, dan $f(x)$ adalah nilai keluaran dari jaringan saraf tiruan. Setelah menghitung nilai *error*, maka perlu melakukan metode *backpropagation* sebagai proses pembelajaran pada sebuah jaringan saraf tiruan. Metode ini memperbaiki bobot / *weight* yang menghubungkan node secara mundur. Nilai *error* yang digunakan pada metode ini befungsi untuk menghitung nilai gradien dari *loss function* agar dapat mengubah bobot untuk mengurangi *error rate*. Metode optimalisasi yang digunakan pada penelitian ini adalah metode *RMSProp* karena performanya lebih stabil dibandingkan optimalisasi lainnya [13].

Untuk mengubah *weight*, pertama lakukan proses perhitungan δo_i :

$$\delta o_i X_i = (o_i - t_i) * g'(a_j) * X_i \quad (2 . 4)$$

$$\delta h_i = \left(\sum_{i=1}^{80} \delta o_i * w_{ji}^2 \right) * g'(a_j) * X_i \quad (2 . 5)$$

Variabel δo_i merupakan gradien error dari *output*, X_i merupakan variabel *input* dari *neuron*, o_i adalah variabel *output* dari jaringan saraf, sedangkan t_i adalah variabel *output* dari jaringan saraf target, dan $g'(a_j)$ adalah variabel turunan dari fungsi aktivasi. di mana turunan dari fungsi pada persamaan 2.2 adalah 1, lalu dimasukkan ke dalam rumus berikut untuk mencari bias:

$$v_t = \beta 2 * v_{t-1} + (1 - \beta 2) * (\delta o_i)^2 \quad (2 . 6)$$

Variabel v_t adalah *chace* dari gradien. Nilai v pada $t = 0$ diinisialisasi dengan angka 0. Nilai dari variabel $\beta 2$ adalah 0.999, dan variabel δo_i merupakan kumpulan gradien hasil propagasi mundur. Setelah kedua nilai di atas didapat, maka selanjutnya masukkan nilai ke dalam rumus untuk mengubah bobot dengan persamaan:

$$\Delta w_{ji}^l = w_{lama} - \frac{\alpha * \delta o_i}{(\sqrt{v_t} + \epsilon)} \quad (2 . 7)$$

di mana w_{lama} adalah nilai bobot lama, δo_i adalah gradien error hasil propagasi mundur, variabel v_t adalah *cache*, α adalah *learning rate*, dan ϵ bernilai 10^{-8} .

2.1.4 Hybrid Reward Architecture

Hybrid Reward Architecture adalah sebuah metode *reinforcement learning* yang berbasis *Deep Q Network*, di mana DQN ini merupakan terobosan dalam area *reinforcement learning* karena teknik ini menggabungkan antara teknik *reinforcement learning* dengan *deep neural network* [13]. Sebuah fungsi nilai memiliki peranan penting dalam teknik RL, karena fungsi tersebut memprediksi nilai yang diharapkan, yang dikondisikan pada *state* atau

pasangan *state-action*. Setelah nilai fungsi optimal diketahui, maka *optimal policy* bisa didapat dengan bertindak rakus (greedy) sehubungan dengan itu.

Perilaku generalisasi DQN dicapai dengan regularisasi pada model untuk fungsi nilai optimal. Namun jika fungsi nilai optimal sangat kompleks, maka mempelajari representasi dimensi rendah yang akurat dapat menjadi tantangan. Oleh karena itu HRA dibuat untuk mengganti nilai fungsi optimal sebagai target untuk latihan dengan nilai fungsi alternatif yang mudah untuk dipelajari. Strategi utama untuk membuat nilai fungsi yang mudah dipelajari adalah dengan menguraikan *reward function* dari lingkungan menjadi n *reward function* yang berbeda. Masing-masing *reward function* ditugaskan pada agen RL yang terpisah [8]. Semua agen dapat belajar secara paralel pada urutan sampel yang sama dengan menggunakan pembelajaran *off-policy*. Masing-masing agen memberikan nilai aksi dari *state* yang sedang berjalan ke sebuah aggregator, yang nantinya akan disatukan ke dalam sebuah nilai tunggal untuk setiap aksi. Tindakan yang dipilih nantinya berdasarkan dari nilai agregat ini.

Algorithm 1 DQN with Experience Replay

```

1: initialize replay memory D
2: initialize action-value function Q with random weights
3: observe initial state s
4: for t=1,T do
5:   select an action  $a$ 
6:   if probability <  $\epsilon$  then
7:     select a random action
8:   else
9:     select  $a = \text{argmax}_{a'} Q(s, a')$ 
10:  end if
11:  carry out action  $a$ 
12:  observe reward  $r$  and new state  $s'$ 
13:  store transition  $(s, a, r, s')$  in replay memory D
14:  sample random transitions  $(s, a, r, s')$  from replay memory D
15:  calculate target for each minibatch transition
16:  if  $s'$  is terminal state then
17:     $y_j = r_j$ 
18:  else
19:     $y_j = r_j + \gamma \max_{a'} Q(s, a')$ 
20:  end if
21:  train the Q network using  $(y_j - Q(s, a))^2$  as loss
22:  every C steps reset  $Q' = Q$ 
23: end for
```

Reward function pada HRA dapat didefinisikan sebagai berikut [8]:

$$R^{HRA}(s, a) = \sum_{i=1}^n w_i R_i(s, a) \quad (2 . 8)$$

untuk setiap *state* s dan *action* a , di mana w_i adalah bobot dari masing-masing *reward function*. Masing-masing agen i dari HRA dilatih menggunakan *reward function*-nya sendiri $R_i(s, a)$. Karena *reward function* individual, masing-masing agen i memiliki fungsi *Q-value* tersendiri. Fungsi *Q-value* dari HRA didefinisikan sebagai jumlah bobot dari semua agen [8]:

$$Q^{HRA}(s, a; \theta) = \sum_{i=1}^n w_i Q_i(s, a : \theta) \quad (2 . 9)$$

di mana θ adalah parameter *training*, dan Q adalah *Q-value*. Masing-masing *head* dapat dilihat secara alternatif sebagai sebuah agen DQN tunggal, dan semua *head* berbagi beberapa lapisan DQN tingkat rendah.

2.1.5 Perhitungan Persentase

Perhitungan persentase adalah perhitungan yang dilakukan untuk menyatakan informasi secara kuantitatif. Persentase ini merupakan bentuk lain dari bilangan pecahan, berasal dari kata Latin *per centum* yang artinya "dari ratusan" [16]. Untuk mencari nilai persentasi dapat dilakukan dengan cara berikut [16]:

$$\text{NilaiPersentase} = \left(\frac{\text{Jumlahdatadicari}}{\text{Jumlahdatakeseluruhan}} \right) \times 100 \quad (2 . 10)$$

2.2 Tinjauan Studi

Pada bagian ini akan dijelaskan mengenai perbandingan dari berbagai penelitian terkait metode penelitian yang akan digunakan. Terdapat beberapa metode lain yang memiliki ruang lingkup yang serupa dengan penelitian ini

Tabel 2.1 Tabel Tinjauan Studi

No	Peneliti	Judul	Tahun	Masalah	Metode/ <i>Win Percentage</i>
1	Ivan Pereira Pinta, Luciano Reis Coutinho	<i>Hierarchical Reinforcement Learning with Monte Carlo Tree Search in Computer Fighting Game</i>	2018	Pembuatan AI untuk permainan fighting yang adaptif masih jarang ditemukan	Menggunakan metode Hierarchical Reinforcement Learning dengan Monte Carlo Tree Search. Persentase kemenangan terhadap 3 AI teratas: 1. <i>GigaThunder</i> - 48.42% 2. <i>FooAI</i> - 65.15% 3. <i>Jaybot</i> - 71.75%

Tabel 2.1 Tabel Tinjauan Studi

No	Peneliti	Judul	Tahun	Masalah	Metode/ <i>Win Percentage</i>
2	Makoto Ishihara, Taichi Miyazaki, Chun Yin Chu, Tomohiro Harada, & Ruck Thawonmas	<i>Applying and Improving Monte-Carlo Tree Search in a Fighting Game AI</i> [5]	2016	Kebanyakan AI pada permainan fighting masih mengandalkan rule base dan bereaksi terhadap situasi apapun dengan menggunakan aksi yang sudah didefinisikan sebelumnya, sehingga mudah untuk diprediksi pergerakannya oleh pemain manusia	Menggunakan metode Monte Carlo Tree Search dengan Roulette Selection. Urutan ranking pada kompetisi tahun 2016: 1. <i>ni1mir4ri</i> - 1st 2. <i>Machete</i> - 2nd 3. <i>RAR(MCTS-Roulette)</i> - 3
3	Harm van Seijen, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes & Jeffrey Tsang	<i>Hybrid Reward Architecture for Reinforcement Learning</i> [8]	2017	Belum adanya penerapan value function yang sederhana yang dapat digunakan untuk pencarian aksi yang optimal	Menggunakan metode DQN dan HRA, AI agen dilatih untuk mencapai skor tertinggi pada permainan <i>Ms. Pac-man</i> . Hasil skor yang didapat adalah 25304.

Pada referensi penelitian pertama, metode utama yang digunakan untuk pembuatan AI adalah *Hierarchical Reinforcement Learning* dengan *Monte Carlo Tree Search*. HRL yang digunakan pada ini menggunakan *Option Framework*, bertujuan untuk mengubah formula *Markov Decision Process* menjadi *Semi-Markov Decision Process* sehingga memungkinkannya agen memilih aksi yang diperluas. Sedangkan MCTS digunakan untuk membuat pohon pencarian berdasarkan hasil dari *Option framework* sebelumnya. Proses ini bertujuan agar MCTS memiliki sedikit aksi yang harus dibentuk pohon pencarinya, sehingga hasil yang dikeluarkan akan lebih presisi [7].

Dalam referensi kedua, menggunakan perpaduan metode *Monte Carlo Tree Search* dengan *Roulette Selection*. *Roulette Selection* digunakan untuk memperbaiki performa MCTS biasa, karena ada beberapa aksi lawan pada proses simulasi MCTS tidak pernah terjadi, sehingga terkadang algoritme MCTS memasukkan aksi lawan yang tidak pernah sama sekali digunakan yang akan membuat prediksi menjadi tidak akurat kedepannya. *Roulette Selection* berguna pada proses simulasi, untuk menentukan probabilitas aksi lawan yang akan digunakan selanjutnya berdasarkan aksi-aksi lawan sebelumnya [5].

Pada referensi ketiga, metode yang digunakan adalah *Deep Q Network* dan *Hybrid Reward Architecture*. Penggunaan HRA pada penelitian ini dengan membuat 4 *head reward function*. Keempat *head reward function* dari agen HRA dan DQN memiliki peran belajar masing - masing sesuai dengan objeknya, *pellet*, buah, *ghost*, dan *blue ghost*. Agen HRA memiliki skor tertinggi dibandingkan dengan pemain manusia, yaitu sebesar 25304 [8].

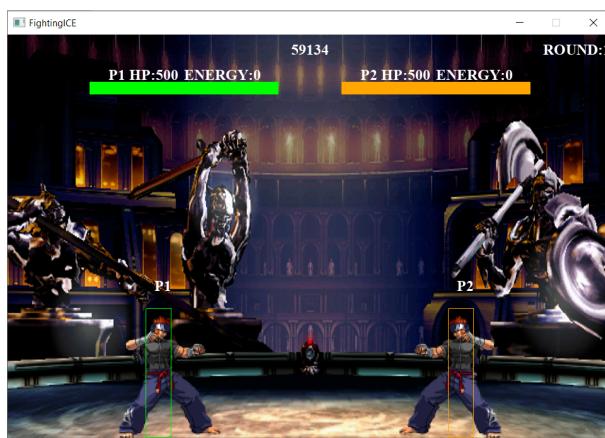
2.3 Tinjauan Objek

Pada bagian ini akan dibahas objek-objek yang terkait dengan penerapan *Reinforcement Learning* berbasis MCTS

2.3.1 FightingGameICE

FightingGameICE dikembangkan secara khusus oleh Intelligent Computer Entertainment Lab., dari Ritsumekan University, di Jepang [9]. FightingICE ini merupakan *fighting game* yang dapat dimainkan oleh 2 pemain, dan terdapat 3 data karakter yang dapat digunakan dengan 40 aksi pada setiap karakter. Aspek penting yang membedakan *platform* FightingICE dari *game fighting* lainnya adalah *platform* ini memiliki delay pada *state variables*. Hal tersebut dilakukan agar pemain AI yang bertarung tidak bereaksi dan memiliki refleks menyerupai manusia super ketika lawan memulai gerakannya, tetapi agar pemain AI dapat melakukan prediksi sehingga lebih menyerupai pemain manusia. Dalam perangkat lunak ini terdapat 2 karakter yang dapat digunakan, yakni ZEN, dan GARNET. Zen merupakan karakter dasar yang kemampuannya dapat digunakan hampir di setiap kondisi, sedangkan Garnet merupakan karakter yang hampir seluruh serangannya menggunakan kaki.

Data masukan yang akan digunakan untuk penelitian menggunakan data yang tersedia pada FightingGameICE. Data yang digunakan adalah *CharacterData* dari setiap *frame* pada



Gambar 2.4 Tampilan FightingGameICE

permainan tersebut, yang akan dibentuk sebagai *state*. Setiap pemain akan diambil data darah, energi, koordinat x, koordinat y, pergerakan terhadap sumbu X, pergerakan terhadap sumbu Y, dan 56 pergerakannya. Selain itu, data proyektil juga akan diambil seperti koordinat x dan y proyektil tersebut, dan nilai kerusakannya.

2.3.1.1 Karakter pada *FightingICE*

Beberapa komponen pada permainan ini berasal dari *The Rumble Fish 2*, terutama pada komponen karakter [9]. Terdapat 3 karakter yang tersedia pada *FightingICE*:

1. Karakter Zen: merupakan karakter utama dalam permainan ini. Karakter ini memiliki serangan umum yang terdiri dari pukulan dan tendangan, baik dalam serangan dasar maupun serangan bertubi - tubi.
2. Karakter Garnet: merupakan karakter kedua dalam permainan ini. Karakter ini memiliki karakteristik serangan tendangan, sehingga baik serangan dasar maupun serangan bertubi - tubinya merupakan serangan dari tendangan kaki.
3. Karakter Lud; merupakan karakter ketiga dalam permainan ini. Karakter ini memiliki karakteristik serangan berlari dan cepat.



Gambar 2.5 Daftar karakter FightingICE. Dari kiri ke kanan: Zen, Garnet, dan Lud

2.3.1.2 Data permainan pada *FightingICE*

Dalam *platform FightingICE*, *state* permainan dibentuk oleh sebuah *class* yang bernama *FrameData*. Di dalam *class* ini terdapat *method* yang dapat digunakan untuk melihat

data permainan seperti:

Tabel 2.2 Daftar *Method class FrameData* dari *FightingICE*

No	Nama <i>Method</i>	Tipe Masukan	Tipe Keluaran	Keterangan
1	getCharacter	boolean	CharacterData	Metode yang digunakan untuk mengeluarkan isi data dari <i>class CharacterData</i> tertentu sesuai masukan.
2	getDistanceX		int	Metode untuk menampilkan jarak horizontal antara P1 dan P2.
3	getDistanceY		int	Metode untuk menampilkan jarak vertikal antara P1 dan P2.
4	getEmptyFlag		boolean	Metode yang mengeluarkan nilai <i>true</i> jika <i>class</i> kosong, selain itu bernilai <i>false</i> jika memiliki data.
5	getFramesNumber		int	Metode yang mengeluarkan jumlah <i>frame</i> sejak awal ronde.

6	getProjectiles		Deque<AttackData>	Metode yang digunakan untuk menampilkan data proyektil dari kedua pemain.
7	getProjectilesByP1		Deque<AttackData>	Metode yang digunakan untuk menampilkan data proyektil P1.
8	getProjectilesByP2		Deque<AttackData>	Metode yang digunakan untuk menampilkan data proyektil P2.
9	getRemainingFramesNumber		int	Metode yang digunakan untuk menampilkan <i>frame</i> tersisa untuk ronde tersebut.
10	getRemainingTimeMilliseconds		int	Metode yang digunakan untuk menampilkan waktu yang tersisa dari sebuah ronde dalam milidetik.
11	getRound		int	Metode yang digunakan untuk menampilkan jumlah ronde saat ini.

Selain itu terdapat juga data karakter yang berada pada *class CharacterData*, yang menyimpan data - data penting dari karakter pada saat permainan berjalan. *Class* tersebut berisi:

Tabel 2.3 Daftar Method class *CharacterData* dari *FightingICE*

No	Nama Method	Tipe Masukan	Tipe Keluaran	Keterangan
1	getAction		Action	Metode yang digunakan untuk menampilkan aksi dari karakter.
2	getAttack		AttackData	Metode untuk menampilkan data serangan yang sedang dilakukan oleh karakter.
3	getBottom		int	Metode untuk menampilkan <i>hit-box</i> koordinat y terbawah dari karakter.
4	getCenterX		int	Metode untuk menampilkan <i>hit-box</i> koordinat x tengah dari karakter.
5	getCenterY		int	Metode untuk menampilkan <i>hit-box</i> koordinat y tengah dari karakter.
6	getEnergy		int	Metode yang digunakan untuk menampilkan jumlah energi karakter.

7	getGraphicAdjustX		int	Metode yang digunakan untuk menampilkan jumlah pergerakan pada arah horizontal yang digunakan untuk menyesuaikan koordinat x ketika menentukan arah karakter.
8	getGraphicSizeX		int	Metode yang digunakan untuk menampilkan lebar dari gambar karakter.
9	getGraphicSizeY		int	Metode yang digunakan untuk menampilkan tinggi dari gambar karakter.
10	getHitCount		int	Metode yang digunakan untuk menampilkan jumlah serangan bertubi - tubi oleh karakter.
11	getHP		int	Metode yang digunakan untuk menampilkan jumlah darah karakter.

12	getLastHitFrame		int	Metode yang digunakan untuk menampilkan angka <i>frame</i> dari <i>frame</i> terakhir yang digunakan oleh serangan karakter yang mengenai sasaran.
13	getLeft		int	Metode untuk menampilkan <i>hit-box</i> koordinat x paling kiri dari karakter.
14	getRemaining Frame		int	Metode untuk menampilkan angka <i>frame</i> yang dibutuhkan karakter untuk kembali ke kondisi normal.
15	getRight		int	Metode untuk menampilkan <i>hit-box</i> koordinat x paling kanan dari karakter.
16	getSpeedX		int	Metode yang digunakan untuk menampilkan kecepatan horizontal karakter.

17	getSpeedY		int	Metode yang digunakan untuk menampilkan kecepatan vertikal karakter.
18	getState		State	Metode yang digunakan untuk menampilkan kondisi karakter (<i>STAND/CROUCH/AIR/DOWN</i>).
19	getTop		int	Metode untuk menampilkan <i>hit-box</i> koordinat y paling atas dari karakter.
20	isControl		boolean	Metode yang digunakan untuk menampilkan <i>flag</i> apakah karakter ini dapat melakukan aksi baru.
21	isFront		boolean	Metode yang digunakan untuk menampilkan arah depan karakter.
22	isHitConfirm		boolean	Metode yang digunakan untuk menampilkan <i>flag</i> apakah serangan mengenai lawan atau tidak.

23	isPlayerNumber		boolean	Metode yang digunakan untuk sisi dari karakter (P1/P2).
24	setAction	Action		Metode yang digunakan untuk menyimpan aksi karakter.
25	setAttack	AttackData		Metode yang digunakan untuk menyimpan data serangan karakter.
26	setBottom	int		Metode yang digunakan untuk menyimpan <i>hit-box</i> koordinat y paling bawah dari karakter.
27	setControl	boolean		Metode yang digunakan untuk menyimpan <i>flag</i> apakah karakter dapat melakukan aksi atau tidak
28	setEnergy	int		Metode yang digunakan untuk menyimpan energi karakter.
29	setFront	int		Metode yang digunakan untuk menyimpan arah depan karakter.

30	setHitConfirm	boolean		Metode yang digunakan untuk menyimpan boolean apakah serangan mengenai lawan atau tidak.
31	setHitCount	int		Metode yang digunakan untuk menyimpan jumlah serangan bertubi - tubi yang dilakukan.
32	setHp	int		Metode untuk menyimpan darah karakter.
33	setLastHitFrame	int		Metode yang digunakan untuk menyimpan angka <i>frame</i> dari <i>frame</i> terakhir yang digunakan oleh aksi yang mengenai lawan.
34	setLeft	int		Metode yang digunakan untuk menyimpan <i>hit-box</i> koordinat x paling kiri dari karakter.

35	setRemainingFrame	int		Metode yang digunakan untuk menyimpan jumlah frame tersisa yang karakter butuhkan untuk kembali ke status normalnya.
36	setRight	int		Metode yang digunakan untuk menyimpan <i>hit-box</i> koordinat x paling kanan dari karakter.
37	setSpeedX	int		Metode yang digunakan untuk menyimpan kecepatan horizontal karakter.
38	setSpeedY	int		Metode yang digunakan untuk menyimpan kecepatan vertikal karakter.
39	setState	int		Metode yang digunakan untuk menyimpan kondisi karakter.
40	setTop	int		Metode yang digunakan untuk menyimpan <i>hit-box</i> koordinat y paling atas dari karakter.

41	setX	int		Metode yang digunakan untuk menyimpan posisi horizontal karakter.
42	setY	int		Metode yang digunakan untuk menyimpan posisi vertikal karakter.

2.3.1.3 Rule-based AI

Rule-based AI adalah sistem kepintaran buatan yang paling sering digunakan dalam aplikasi dunia nyata maupun dalam permainan. Dalam bentuknya yang paling sederhana, *rule-based AI* terdiri dari sekumpulan kondisi *if-then* yang digunakan untuk membuat kesimpulan atau pengambilan keputusan [15]. Objek penelitian yang akan digunakan adalah penggunaan *rule-based AI* sebagai lawan latih dari AI yang akan dikembangkan. *Rule-based AI* ini memiliki *rule* yang mengacu berdasarkan jarak terhadap lawan, jumlah energi musuh, dan energi dari AI itu sendiri. Jika musuhnya terlalu jauh, maka AI akan mendekat dengan cara melompat kedepan. Jika musuh jauh tetapi tidak terlalu jauh, maka AI akan mendekat dengan cara berlari kedepan. Ketika energi telah mencapai *threshold* maka AI akan melakukan aksi sesuai dengan jumlah energi yang dimiliki.

2.3.2 Game Fighting

Game Fighting adalah subgenre dari *Action Game* yang berbeda dengan subgenre lainnya karena tidak adanya eksplorasi, tembak-menembak, maupun penyelesaian puzzle. *Game Fighting* ini masuk ke dalam kualifikasi *Action Game* karena membutuhkan kemampuan fisik pemain, yakni waktu reaksi dan pengukuran waktu (*timing*). Permainan ini mensimulasikan pertarungan tangan kosong, biasanya menggunakan aksi yang berlebihan yang dimodelkan berdasarkan teknik bela diri Asia.

Aksi pemain biasanya terdiri dari serangan tangan kosong dan gerakan bertahan yang bermacam-macam. Biasanya gerakan bertahan hanya menangkis beberapa serangan tertentu saja, dan pemain harus mempelajari kapan dan bagaimana aksi dieksekusi secara efektif melalui uji coba. Setiap serangan yang berhasil diluncurkan mengurangi energi lawan, dan permainan terus berlanjut sampai salah satu energi pemain habis.

Salah satu fitur umum pada permainan ini adalah gerakan *combo*. *Combo* ini diciptakan karena keterbatasannya input tombol pada mesin permainan sehingga sistem *combo* diciptakan

sebagai tantangan ekstra agar pemain dapat mengeksekusi gerakan tertentu yang sangat efektif dengan menggunakan pilihan tombol tertentu. Tingkat efektifitas gerakan ini tergantung dari kesulitan eksekusinya, semakin kompleks *combo*-nya maka semakin besar juga kemungkinan avatar lawan diserang ketika pemain sedang mencoba urutan masukannya.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

Bab ini menjelaskan analisis masalah beserta pendekatan dan alur kerja dari aplikasi yang akan dikembangkan, dimulai dari *preprocessing*, implementasi metode dan hasil yang ditampilkan.

3.1 Analisis Masalah

Penelitian ini akan menggunakan data yang dihasilkan dari *platform FightingICE*. Pendekatan *reinforcement learning* dipilih karena banyaknya implementasi yang berhasil pada sejumlah permainan, bahkan agen AI dapat bersaing dengan manusia. Masalah yang dihadapi ketika membangun AI untuk mempelajari caranya bermain sebuah permainan video adalah ketidaktahuannya agen terhadap hasil yang didapat setelah agen melakukan aksinya. Sebuah lingkungan (*environment*) pada suatu permainan berpengaruh terhadap bagaimana AI memilih aksi yang tepat. Selain faktor lingkungan, banyaknya kemungkinan aksi yang dapat dipilih menyebabkan masalah baru, di mana agen harus dapat memprediksi aksi yang tepat untuk dilakukan, dan bukan melakukan serangkaian aksi secara acak untuk mencapai targetnya. Untuk itu, perlu adanya pembagian nilai *reward* yang jelas agar agen dapat terus belajar dan memaksimalkan nilai *reward* tersebut. Untuk itu, dipilih tipe permainan *Fighting* sebagai objek penelitian, karena permainan *Fighting* memiliki jumlah aksi dan variasi kondisi yang banyak, dan masih relatif sedikit penelitian *Reinforcement Learning* terhadap permainan *Fighting*.

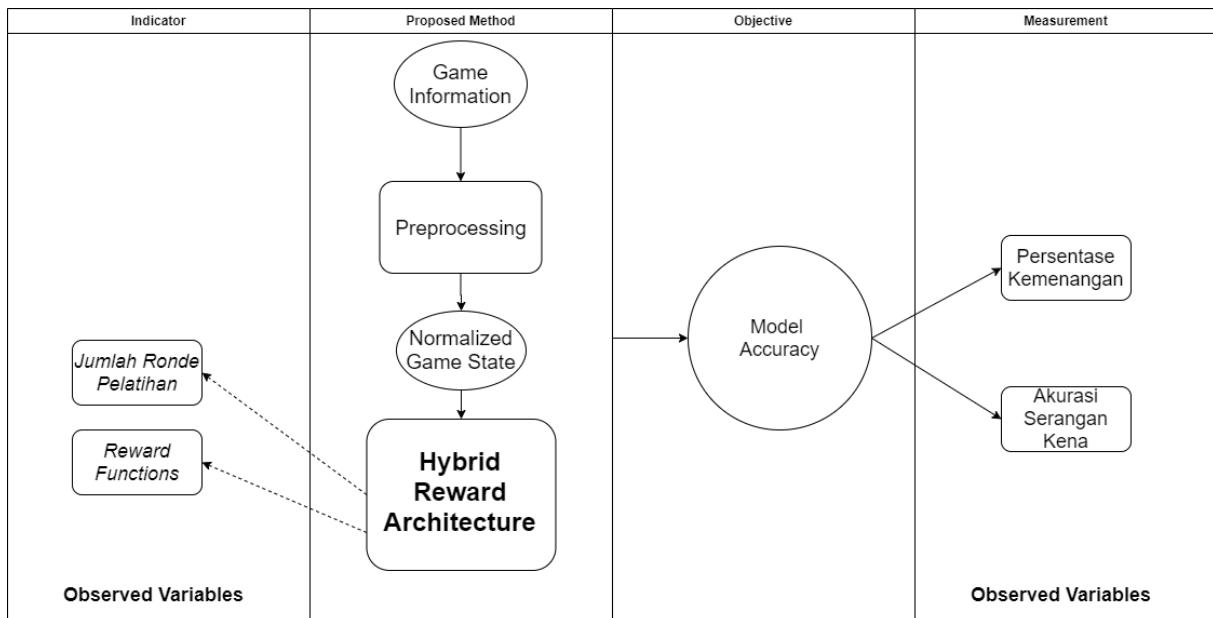
Penelitian ini akan menggunakan metode *Hybrid Reward Architecture* pada pengembangan untuk melakukan estimasi *value function* terhadap sistem *reinforcement learning* untuk tipe permainan *fighting*. HRA ini akan memecah *reward function* menjadi beberapa bagian, dan pada penelitian ini, dipecah menjadi 3 nilai reward function. Metode ini adalah metode modifikasi dari sistem *Deep Q-Network* (DQN), yang merupakan sistem *deep reinforcement learning*, yakni perpaduan antara *deep learning* dan *reinforcement learning*.

Data yang diambil dari *FightingICE* berjumlah 141 unit. *State* permainan dari *FightingICE* dapat direkonstruksi berdasarkan 142 fitur masukan. *FightingICE* juga memiliki 40 aksi untuk kendali AI, masing - masing merepresentasikan aksi yang berbeda - beda. 142 data tersebut dapat dipecah menjadi 3 bagian utama.

Masukan akan diproses dengan melakukan *preprocessing* sebelum digunakan ke dalam sistem pembelajaran dengan melakukan normalisasi data dengan rentang nilai 0 sampai dengan 1. Setelah itu baru dilakukan proses pembelajaran dengan memasukkan data tersebut menjadi sebuah *state* ke dalam sistem *Hybrid Reward Architecture* untuk memprediksi aksi mana yang tepat untuk dilakukan ketika menghadapi AI lawannya.

3.2 Kerangka Pemikiran

Penelitian ini bertujuan untuk menguji berapa skor dan persentase kemenangan yang didapat dari agen yang dibuat dengan menggunakan metode *Hybrid Reward Architecture*. Berikut ini adalah kerangka pemikiran dan keterangannya dari metode yang diusulkan untuk melakukan klasifikasi:

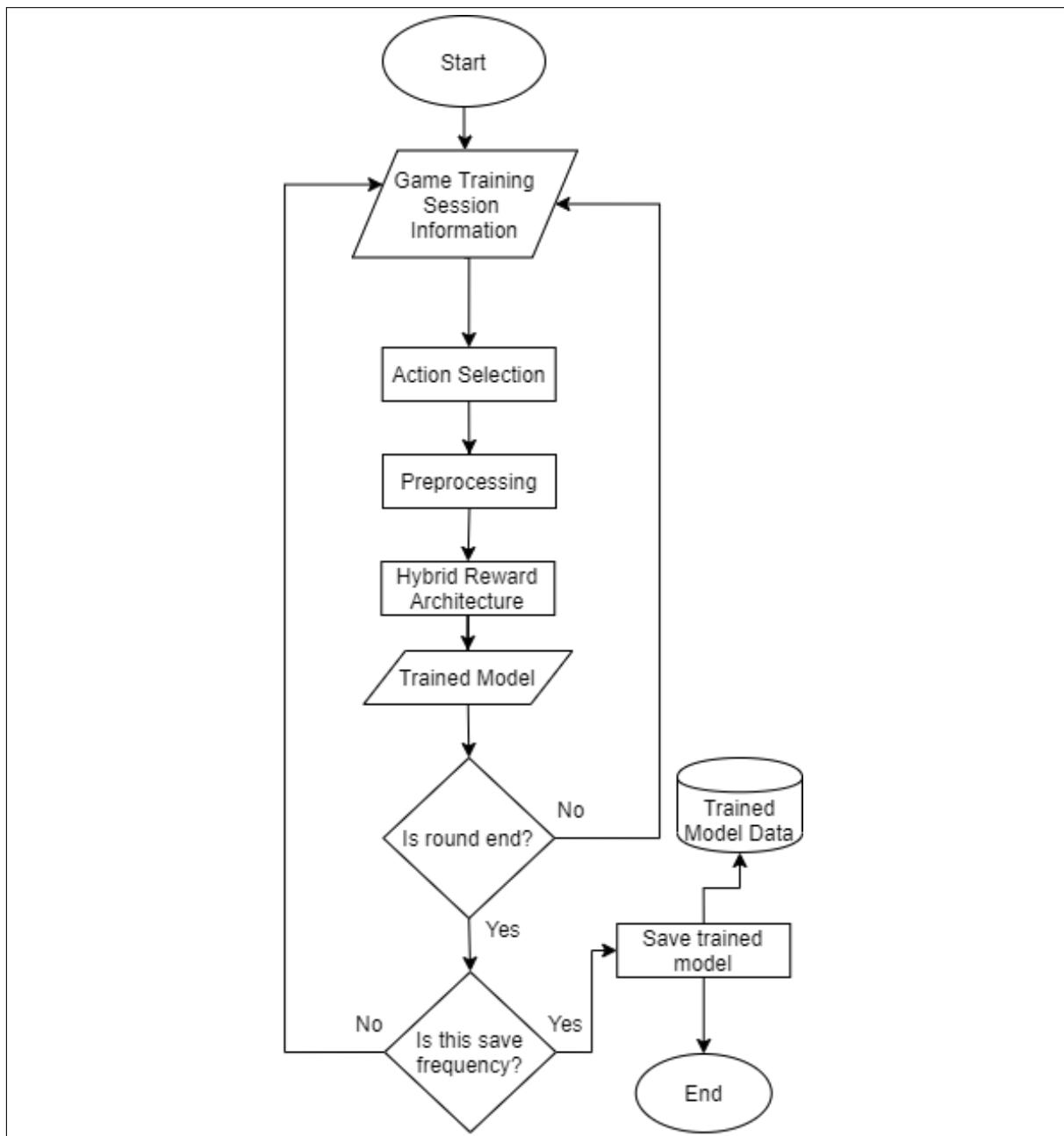


Gambar 3.1 Kerangka Pemikiran

1. *Indicator*: faktor yang dapat mempengaruhi hasil proses dari metode utama dalam penelitian ini. Pada metode utama, *Number of Training* berpengaruh terhadap seberapa bagus performa agen setelah melakukan pembelajaran selama pelatihan sebanyak jumlah tertentu, dan *Reward Function* berpengaruh terhadap seberapa pentingnya aksi yang dilakukan dalam mencapai target.
2. *Proposed Method*: proses yang diusulkan dalam penelitian ini adalah :
 - (a) *Pre-processing* : merupakan tahap untuk mempersiapkan data masukan agar dapat diproses pada proses selanjutnya. Pada tahap ini dilakukan proses normalisasi masukan pada rentang [0,1]
 - (b) *Hybrid Reward Architecture* : Proses pembelajaran agen dengan metode HRA, agar agen dapat memilih 40 aksi yang dapat dilakukan untuk menghadapi lawan.
3. Tujuan dari penelitian ini adalah untuk menguji seberapa besar kemampuan agen yang dihasilkan dari metode-metode yang digunakan dalam sistem ini dan menerapkannya dalam aplikasi.
4. Pengukuran : Cara untuk menghitung kapabilitas AI yang akan digunakan adalah dengan perhitungan *win rate* dan mengukur akurasi serangan.

3.3 Analisis Urutan Proses Global

Alur proses dimulai dari pengambilan informasi permainan dari FightingICE. Karena sifatnya yang merupakan *reinforcement learning* maka sistem akan selalu mempelajari hal baru untuk memprediksi pergerakan AI lawan, baik pada tahapan *training* ataupun *testing*. Pada proses *training* dilakukan proses pencatatan model ke *database*, dan pada tahapan *testing* dilakukan proses pengambilan data model dari *database*. Berikut adalah *flowchart* untuk sistem *reinforcement learning* dalam penelitian ini:



Gambar 3.2 Flowchart Sistem Pembelajaran AI

Pertama proses dimulai dengan mengambil informasi dari sesi pelatihan pada permainan. Dari informasi yang diberikan oleh permainan, akan dibagi menjadi 3 bagian, yaitu infomasi pribadi, infomasi lawan, dan infomasi proyektil. Infomasi pribadi dan infomasi lawan memiliki infomasi - infomasi umum seperti data jumlah darah, jumlah

energi yang didapat, posisi karakter tersebut, dan lainnya. Setelah data tersebut didapat, maka selanjutnya dilakukan pemilihan aksi untuk menentukan *state* dari permainan selanjutnya. Aksi yang dipilih dilakukan secara eksplorasi dan eksloitasi, pemilihan eksplorasi dilakukan secara acak, sedangkan pemilihan eksloitasi dilakukan dengan cara memilih menggunakan jaringan yang sudah dilatih.

Proses selanjutnya adalah *preprocessing* data informasi dengan menormalisasikan nilainya ke dalam rentang 0 dan 1. Data yang sudah diolah dimasukkan ke dalam fungsi HRA untuk melakukan pelatihan. Setelah proses pelatihan akan didapatkan bobot model yang sudah dilatih, yang akan disimpan jika jumlah ronde yang sudah dilakukan sesuai dengan kelipatan frekuensi penyimpanan.

3.3.1 Analisis Data Masukan

Dalam penelitian ini, akan digunakan 142 data masukan yang dapat membentuk *state* permainan dari *FightingICE*. *FightingICE* juga memiliki 40 aksi untuk kendali AI, masing - masing merepresentasikan aksi yang berbeda - beda. 142 data tersebut dapat dipecah menjadi 3 bagian utama. Berikut ini adalah bagian - bagian fitur yang diperlukan untuk sistem *Hybrid Reward Architecture*:

1. *Self Information* yang terdiri dari :

- a HP, fitur ini dipilih karena pentingnya mengetahui jumlah darah karakter pemain digunakan sebagai penentu *defense reward*.
- b Energi fitur ini dipilih agar dapat memanfaatkan serangan energi berdasarkan energi yang dimiliki.
- c Koordinat X, fitur ini dipilih sebagai informasi posisi pemain pada X.
- d Koordinat Y, fitur ini dipilih sebagai informasi posisi pemain pada sumbu Y
- e Kecepatan pada arah X, fitur ini dipilih sebagai informasi berapa kecepatan pemain ke arah X.
- f Kecepatan pada arah Y, fitur ini dipilih sebagai informasi berapa kecepatan pemain ke arah Y.
- g Hit Combo, fitur ini dipilih sebagai informasi berapa jumlah pukulan konsekutif yang dilakukan tanpa interupsi, digunakan sebagai penentu *combo reward*.
- h *Motion*, terdiri dari 56 fitur pergerakan karakter, berupa *one-hot vector* sehingga jika ada salah satu aktif, otomatis bernilai 1, sisanya bernilai 0.

2. *Opponent Information* yang terdiri dari :

- a HP, fitur ini dipilih karena pentingnya mengetahui jumlah darah karakter lawan digunakan sebagai penentu *offense reward*.

- b Energi fitur ini dipilih agar dapat mempertimbangkan serangan energi lawan berdasarkan energi yang dimiliki.
- c Koordinat X, fitur ini dipilih sebagai informasi posisi lawan pada X.
- d Koordinat Y, fitur ini dipilih sebagai informasi posisi lawan pada sumbu Y
- e Kecepatan pada arah X, fitur ini dipilih sebagai informasi berapa kecepatan lawan ke arah X.
- f Kecepatan pada arah Y, fitur ini dipilih sebagai informasi berapa kecepatan lawan ke arah Y.
- g *Remaining Frame*, fitur ini dipilih sebagai informasi berapa jumlah *frame* lawan yang tersisa ketika melakukan aksi.
- h *Motion*, terdiri dari 56 fitur pergerakan karakter, berupa *one-hot vector* sehingga jika ada salah satu aktif, otomatis bernilai 1, sisanya bernilai 0.

3. *Projectile Information* yang terdiri dari:

- a Koordinat X, fitur ini dipilih sebagai informasi posisi proyektil pada X.
- b Koordinat Y, fitur ini dipilih sebagai informasi posisi proyektil pada sumbu Y
- c *Hit Damage*, fitur ini dipilih sebagai informasi berapa kerusakan yang didapat oleh proyektil tersebut.

Karena jumlah maksimal proyektil yang dapat diluncurkan oleh karakter adalah 2, maka informasi proyektil memiliki 4 set pada setiap *state* permainan.

3.3.2 *Preprocessing*

Sebelum melakukan ekstraksi fitur, *preprocessing* akan dilakukan pada data masukan. *Preprocessing* ini dilakukan untuk menormalisasi nilai - nilai masukan, sehingga nilai masukan yang masuk berada antara nilai 0 dan 1. Berdasarkan data dari *Motion.csv* yang diberikan pada dokumentasi *platform FightingICE*, ada beberapa data yang dapat dinormalisasi dengan cara berikut :

1. HP : Karena sifat dari platform ini menganggap *Health Point* pemain sebagai *Damage Receiver*, maka unit ini dinormalisasi dengan cara membagi nilai absolut dengan jumlah maksimum *damage* yakni 500.
2. Energi : Jumlah energi dinormalisasi dengan membagi nilai energi yang dimiliki dengan nilai maksimum energi yang didapat sebesar 300.
3. Koordinat X : Nilai tengah posisi X pemain dibagi dengan jumlah resolusi pixel X sebesar 960.

4. Koordinat Y : Nilai tengah posisi Y pemain dibagi dengan jumlah resolusi pixel X sebesar 640.
5. Kecepatan arah X : Nilai kecepatan arah X dibagi dengan jumlah kecepatan maksimal pada arah X yakni 15.
6. Kecepatan arah Y : Nilai kecepatan arah Y dibagi dengan jumlah kecepatan maksimal pada arah Y yakni 28.
7. *Opp. Remaining Frame* : Jumlah *frame* tersisa dibagi dengan nilai *frame* tertinggi yakni 70.

Untuk masukan *motion* data yang diambil dinormalisasikan dengan bentuk *One-hot vector* sehingga datanya sudah biner. *One-hot vector* ini sendiri dilakukan dengan cara memberi nilai 1 pada aksi yang dilakukan dan nilai 0 pada aksi sisanya yang tidak dilakukan. Teknik ini digunakan karena hanya ada satu aksi yang dapat dilakukan dari 56 aksi.

3.3.3 *Hybrid Reward Architecture*

Setelah 142 data masukan berhasil dinormalisasi, selanjutnya data tersebut dimasukkan ke dalam sistem *Hybrid Reward Architecture* untuk melakukan pembelajaran dengan membentuk *state* permainan.

Proses diawali dengan membaca *State* awal, lalu program akan mengambil sampel *probability* secara acak. Jika nilai *probability* kurang dari nilai ϵ , maka aksi yang dipilih adalah aksi secara acak, sebaliknya jika nilai lebih besar maka aksi yang dipilih adalah berdasarkan perhitungan nilai maksimum dari nilai yang sudah dihitung oleh jaringan.

Aksi tersebut lalu dijalankan, dan dihitung nilai *reward*-nya berdasarkan perhitungan 3.1 sampai dengan 3.3. Transisi setiap *state* disimpan ke dalam *replay memory*, lalu jika jumlah *replay memory* sama dengan jumlah *minibatch*, maka dilakukan proses pengambilan data secara acak pada *replay memory* sejumlah *minibatch*.

Data dari *sampling* tersebut akan diproses untuk mencari nilai yang diharapkan apakah sesuai dengan nilai prediksi pada jaringan. Setelah nilai yang diharapkan didapat, maka dicari *loss* pada *state* tersebut, dan lakukan perbaikan jaringan dengan melakukan *gradient descent step*. Setelah beberapa *step* belajar dilakukan, maka model Q' akan diubah dengan model Q yang sudah diperbaiki sebelumnya.

Head pada penelitian ini dibagi menjadi 3 yaitu:

1. *Offense head* : Peran dari *head* ini untuk mempelajari bagaimana caranya memberi serangan pada lawan secara efektif. Fungsi *Reward* didefinisikan sebagai berikut:

$$Reward_t^{Offense} = HP_t^{Opp} - HP_{t+1}^{Opp} \quad (3 . 1)$$

2. *Defense head* : Peran dari *head* ini untuk mempelajari bagaimana caranya menghindari serangan pada lawan secara efektif. Fungsi *Reward* didefinisikan sebagai berikut:

$$Reward_t^{Defense} = HP_{t+1}^{Self} - HP_t^{Self} \quad (3 . 2)$$

3. *Combo head* : Peran dari *head* ini untuk mempelajari bagaimana caranya memberi *combo* pada lawan secara efektif. *Combo* sendiri aktif jika serangan melebihi dua kali dalam waktu yang berdekatan. Fungsi *Reward* didefinisikan sebagai berikut:

$$Reward_t^{Combo} = HitCount_t^{Self} - 2 \quad (3 . 3)$$

3.3.3.1 Trained Model

Pada proses ini, bobot yang sudah dilatih oleh jaringan saraf tiruan akan disimpan sesuai dengan *layer* dari masing - masing jaringan saraf tiruan. Masing - masing *layer* memiliki dua buah *file* bobot, yaitu bobot sesuai *neuron* pada *layer* tersebut dan bobot bias-nya. Setiap data disimpan dalam folder sesuai dengan nomor frekuensi penyimpanannya.

3.3.4 Analisis Manual

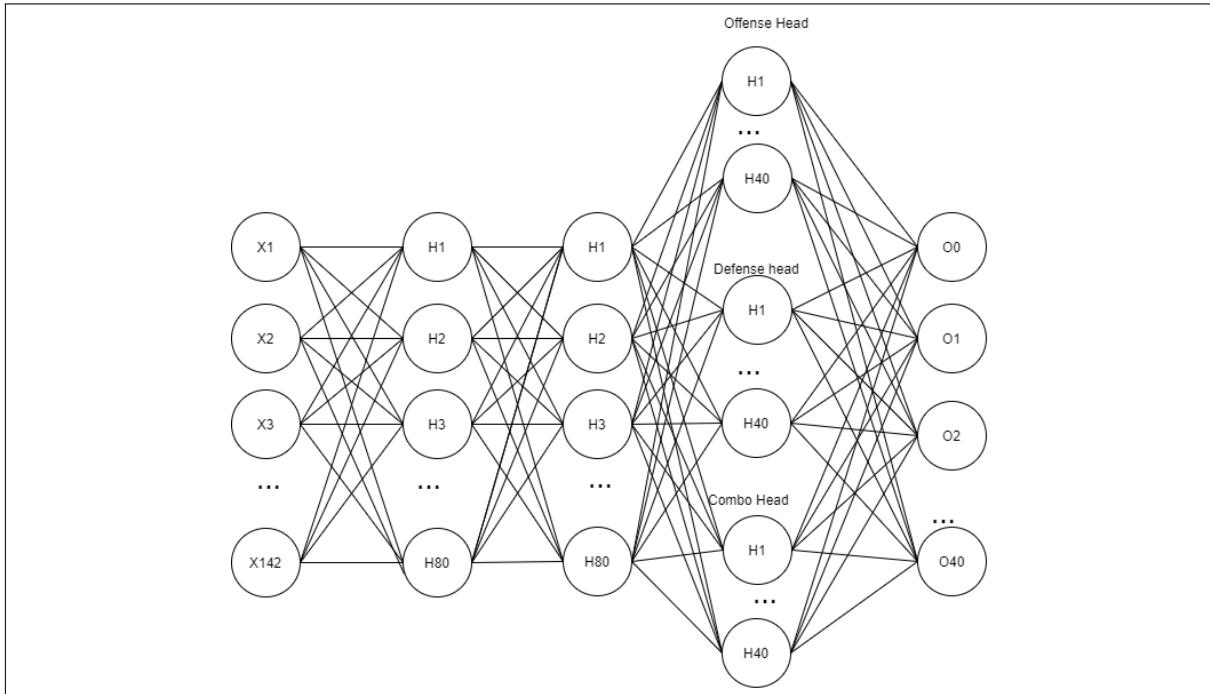
Pada bagian ini terdapat analisis dengan melakukan perhitungan manual metode-metode yang digunakan untuk pembuatan sistem belajar AI permainan *Fighting*.

3.3.4.1 Neural Network

Pada tahap ini, masukan dari hasil preprocessing akan dimasukkan sebagai masukan *state* dari FightingICE. Karena arsitektur HRA menyerupai DQN, maka tahap awal adalah pemasukan masukan terhadap *neural network*. Setiap *neural network* merupakan *Multilayer Perceptron* (MLP) yang pada umumnya terdiri dari tiga *layer*, yaitu *input layer*, *hidden layer*, dan *output layer*. Jumlah *node* pada *input layer* penelitian ini sejumlah vektor dari data masukan. Karena tidak adanya *rules of thumb* yang tepat untuk penentuan seberapa banyak

III. ANALISIS DAN PERANCANGAN SISTEM

hidden layer dan jumlah *neuron*-nya, maka akan digunakan dua *hidden layer* yang masing-masing berisi neuron sebanyak 80 unit, karena pada umumnya jumlah *neuron* yang dibutuhkan untuk *hidden layer* berada antara jumlah masukan dan keluaran, dan karena HRA memiliki basis dari DQN maka jumlah *hidden layer* yang digunakan minimal melebihi satu *hidden layer*. Jumlah *head* ada tiga *head node* dengan masing-masing 40 *node* yang disesuaikan dengan output karena nilainya akan digunakan sebagai penentu pada *output layer*, dan *output layer* dengan jumlah *node* 40 sebagai jumlah aksi yang dapat dipilih oleh AI. Struktur MLP pada sistem HRA ini digambarkan pada gambar dibawah berikut:



Gambar 3.3 Struktur MLP

Tabel 3.1 Data Uji yang Digunakan

State	Input								
	Self HP	Self Energy	Self X Coordinate	Self Y Coordinate	Self X Speed	...	Opponent first Projectile Hit Area Y	Action	
S1	0	0	0.25	0.83984375	1	...	0	?	

III. ANALISIS DAN PERANCANGAN SISTEM

Tabel 3.2 Data Belajar yang Digunakan

State	Input								Opponent first Projectile Hit Area Y
	Self HP	Self Energy	Self X Coordinate	Self Y Coordinate	Self X Speed	Self abs X Speed	...		
S1	0	0	0.25	0.83984375	1	0	...	0	0
S2	0	0	0.25	0.87109375	1	0	...	0	0
S3	0.02	0.03333333	0.630208333	0.83984375	1	0	...	1	1
S4	0.02	0.03	0.520833333	0.83984375	1	0	...	1	1
S5	0.02	0.03	0.40625	0.83984375	0	0.2666666667	...	0	0
S6	0.02	0.03	0.395833333	0.83984375	1	0	...	0	0
S7	0.04	0.04666667	0.386458333	0.68515625	0	0.333333333	...	0	0

Tabel 3.1 adalah kondisi *state* yang akan diujikan, sedangkan Tabel 3.2 adalah data belajar yang digunakan. Masukan adalah data dari FightingICE yang diubah dalam *array 1D*, 1 *node* dalam *input layer* merupakan masukan yang sudah dinormalisasi dari *state* tersebut. Tabel 3.1 menunjukkan nilai dari setiap unit dari *state* untuk data belajar. Sedangkan pada Tabel 3.2 menunjukkan data *testing* yang harus diprediksi untuk memilih aksi yang baik untuk dilakukan oleh AI berdasarkan *state* tersebut. Tahap pertama yang harus dilakukan adalah melakukan inisialisasi beberapa parameter yaitu *Q-Learning mini-batch size* = 32, *Replay memory size* = 100000, *target network update frequency* = 100, *discount factor* = 0.9, *learning rate* = 0.001, *initial exploration* = 0.9, *final exploration* = 0.05, dan seluruh bobot diberi nilai acak, namun pada contoh kasus ini akan diberi nilai 1.

Tabel 3.3 Inisiasi data

	Input Unit	Input	W1i	Hidden Layer (V1)
1	Self HP	0	1	?
2	Self Energy	0	1	
3	Self X Coordinate	0.25	1	
4	Self Y Coordinate	0.83984375	1	
5	Self X Speed	1	1	
6	Self abs X Speed	0	1	
7	Self Y Speed	1	1	
141	Opponent first Projectile Hit Area X	0	1	
142	Opponent first Projectile Hit Area Y	0	1	
	Input Unit	Input	W1i	Hidden Layer (V2)
1	Self HP	0	1	?
2	Self Energy	0	1	
3	Self X Coordinate	0.25	1	
4	Self Y Coordinate	0.83984375	1	
5	Self X Speed	1	1	
6	Self abs X Speed	0	1	
7	Self Y Speed	1	1	
141	Opponent first Projectile Hit Area X	0	1	
142	Opponent first Projectile Hit Area Y	0	1	

	Input Unit	Input	W1i	Hidden Layer (V80)	
1	Self HP		0	1	?
2	Self Energy		0	1	
3	Self X Coordinate		0.25	1	
4	Self Y Coordinate		0.83984375	1	
5	Self X Speed		1	1	
6	Self abs X Speed		0	1	
7	Self Y Speed		1	1	
141	Opponent first Projectile Hit Area X		0	1	
142	Opponent first Projectile Hit Area Y		0	1	

Pada kasus ini, akan dipilih aksi secara acak dengan nilai probabilitas di atas nilai *initial exploration*. Selanjutnya untuk mendapatkan nilai setiap *node* pada *Hidden Layer*, maka dilakukan perhitungan dengan menggunakan perhitungan 2.1

$$\begin{aligned}
 &= \sum_i^{142} (X_i \cdot W_{1i}^1) \\
 &= (0 * 1) + (0 * 1) + (0.25 * 1) + (0.83984375 * 1) + (1 * 1) + (0 * 1) + (1 * 1) + \dots + (0 * 1) \\
 &= 0 + 0 + 0.25 + 0.83984375 + 1 + 0 + 1 + \dots + 1 = 9.365401786 \\
 &= \sum_i^{142} (X_i \cdot W_{2i}^1) \\
 &= (0 * 1) + (0 * 1) + (0.25 * 1) + (0.83984375 * 1) + (1 * 1) + (0 * 1) + (1 * 1) + \dots + (0 * 1) \\
 &= 0 + 0 + 0.25 + 0.83984375 + 1 + 0 + 1 + \dots + 1 = 9.365401786 \\
 &\dots \\
 &\dots \\
 &= \sum_i^{142} (X_i \cdot W_{142i}^1) \\
 &= (0 * 1) + (0 * 1) + (0.25 * 1) + (0.83984375 * 1) + (1 * 1) + (0 * 1) + (1 * 1) + \dots + (0 * 1) \\
 &= 0 + 0 + 0.25 + 0.83984375 + 1 + 0 + 1 + \dots + 1 = 9.365401786
 \end{aligned}$$

Seluruh hasil dari perhitungan *node* 1-142 sama, karena pada contoh ini dipilih bobot sebesar 1 sebagai inisiasi awal. Nilai tersebut lalu dimasukkan ke dalam fungsi aktivasi h, di mana h adalah fungsi ReLU (*rectified linear unit*) sehingga didapatkan hasil berikut:

III. ANALISIS DAN PERANCANGAN SISTEM

$$H_1 = \max(0, 9.365401786) = 9.365401786$$

$$H_2 = \max(0, 9.365401786) = 9.365401786$$

...

...

$$H_{142} = \max(0, 9.365401786) = 9.365401786$$

Tabel 3.4 Hasil perhitungan yang dimasukkan ke tabel

	Input Unit	Input	W1i	Hidden Layer (V1)
1	Self HP	0	1	9.365401786
2	Self Energy	0	1	
3	Self X Coordinate	0.25	1	
4	Self Y Coordinate	0.83984375	1	
5	Self X Speed	1	1	
6	Self abs X Speed	0	1	
7	Self Y Speed	1	1	
141	Opponent first Projectile Hit Area X	0	1	
142	Opponent first Projectile Hit Area Y	0	1	

	Input Unit	Input	W1i	Hidden Layer (V2)
1	Self HP	0	1	9.365401786
2	Self Energy	0	1	
3	Self X Coordinate	0.25	1	
4	Self Y Coordinate	0.83984375	1	
5	Self X Speed	1	1	
6	Self abs X Speed	0	1	
7	Self Y Speed	1	1	
141	Opponent first Projectile Hit Area X	0	1	
142	Opponent first Projectile Hit Area Y	0	1	

...

	Input Unit	Input	W1i	Hidden Layer (V80)
1	Self HP	0	1	9.365401786
2	Self Energy	0	1	
3	Self X Coordinate	0.25	1	
4	Self Y Coordinate	0.83984375	1	
5	Self X Speed	1	1	
6	Self abs X Speed	0	1	
7	Self Y Speed	1	1	
141	Opponent first Projectile Hit Area X	0	1	
142	Opponent first Projectile Hit Area Y	0	1	

$$\begin{aligned}
 H2_0 &= h\left(\sum_i^{142} (X_i \cdot W_{0i}^2)\right) \\
 &= h((9.365401786 * 1) + (9.365401786 * 1) + (9.365401786 * 1) + (9.365401786 * 1) + \\
 &\quad (9.365401786 * 1) + (9.365401786 * 1) + (9.365401786 * 1) + \dots + (9.365401786 * 1)) \\
 &= h(749.2321429) = 749.2321429 \\
 H2_1 &= h\left(\sum_i^{142} (X_i \cdot W_{1i}^2)\right) \\
 &= h((9.365401786 * 1) + (9.365401786 * 1) + (9.365401786 * 1) + (9.365401786 * 1) + \\
 &\quad (9.365401786 * 1) + (9.365401786 * 1) + (9.365401786 * 1) + \dots + (9.365401786 * 1)) \\
 &= h(749.2321429) = 749.2321429 \\
 &\dots \\
 &\dots \\
 H2_{142} &= h\left(\sum_i^8 (X_i \cdot W_{142i}^2)\right) \\
 &= h((9.365401786 * 1) + (9.365401786 * 1) + (9.365401786 * 1) + (9.365401786 * 1) + \\
 &\quad (9.365401786 * 1) + (9.365401786 * 1) + (9.365401786 * 1) + \dots + (9.365401786 * 1)) \\
 &= h(749.2321429) = 749.2321429
 \end{aligned}$$

Langkah di atas dilakukan untuk meneruskan hasil pada *hidden layer* pertama menuju *hidden layer* selanjutnya dengan cara yang sama seperti sebelumnya. Namun pada tahap tersebut perhitungan langsung digabungkan dengan fungsi aktivasi untuk mempersingkat perhitungan.

III. ANALISIS DAN PERANCANGAN SISTEM

Tabel 3.5 Hasil perhitungan hidden layer yang dimasukkan ke tabel

	Input	W1i	H2(1)
1	9.365402	1	749.2321429
2	9.365402	1	
3	9.365402	1	
4	9.365402	1	
5	9.365402	1	
6	9.365402	1	
7	9.365402	1	
80	9.365402	1	

	Input	W1i	H2(2)
1	9.365402	1	749.2321429
2	9.365402	1	
3	9.365402	1	
4	9.365402	1	
5	9.365402	1	
6	9.365402	1	
7	9.365402	1	
80	9.365402	1	

...

	Input	W1i	H2(80)
1	9.365402	1	749.2321429
2	9.365402	1	
3	9.365402	1	
4	9.365402	1	
5	9.365402	1	
6	9.365402	1	
7	9.365402	1	
80	9.365402	1	

Setelah hasil pada *Hidden Layer 2* didapat, selanjutnya akan diteruskan secara linear pada tiap *head*. Dengan menggunakan perhitungan 2.13 setiap nilai Q pada masing-masing *head* akan dijumlahkan sebagai *output*.

Tabel 3.6 Hasil perhitungan *hidden layer 2* pada *head*

	Input	W1i	Head(1)		Input	W1i	Head _i (2)		Input	W1i	Head _i (40)
1	749.2321	1	59938.57		1	749.2321	1	59938.57		1	59938.57
2	749.2321	1			2	749.2321	1			2	1
3	749.2321	1			3	749.2321	1			3	1
4	749.2321	1			4	749.2321	1			4	1
5	749.2321	1			5	749.2321	1			5	1
6	749.2321	1			6	749.2321	1			6	1
7	749.2321	1			7	749.2321	1			7	1
8	749.2321	1			8	749.2321	1			8	1
80	749.2321	1			80	749.2321	1			80	1

...

Tabel 3.7 Tabel indeks Aksi

i	Action	i	Action	i	Action	i	Action
0	FORWARD_WALK	10	THROW_B	20	STAND_FB	30	STAND_F_D_DFB
1	DASH	11	STAND_A	21	CROUCH_FA	31	STAND_D_DB_BA
2	BACK_STEP	12	STAND_B	22	CROUCH_FB	32	STAND_D_DB_BB
3	JUMP	13	CROUCH_A	23	AIR_FA	33	AIR_D_DF_FA
4	FOR_JUMP	14	CROUCH_B	24	AIR_FB	34	AIR_D_DF_FB
5	BACK_JUMP	15	AIR_A	25	AIR_UA	35	AIR_F_D_DFA
6	STAND_GUARD	16	AIR_B	26	AIR_UB	36	AIR_F_D_DFB
7	CROUCH_GUARD	17	AIR_DA	27	STAND_D_DF_FA	37	AIR_D_DB_BA
8	AIR_GUARD	18	AIR_DB	28	STAND_D_DF_FB	38	AIR_D_DB_BB
9	THROW_A	19	STAND_FA	29	STAND_F_D_DFA	39	STAND_D_DF_FC

Nilai yang didapat setelah menjumlahkan masing-masing *head* akan berbeda, namun dalam kasus ini, karena semua *weight* pada masing-masing *layer* yang diinisiasi dengan nilai 1, maka nilai *Q* yang didapat akan sama. Setelah *output* didapat, selanjutnya adalah memilih nilai *Q* terbesar yang indeksnya akan diambil sebagai pilihan aksi dari 40 aksi yang dapat dilakukan seperti pada Tabel 3.7.

Tabel 3.8 Tabel contoh *Batch*

State	Action	Next State	off.reward
S1	39	S1'	0
S2	31	S2'	0
S3	4	S3'	0
S4	9	S4'	0
S5	8	S5'	0
S6	30	S6'	0
S7	24	S7'	10
S32	18	S32'	10

Setelah aksi secara acak dilakukan, *reward* terhadap masing-masing *head* akan dibentuk berdasarkan perhitungan 2.12. Lalu algoritma akan membaca *State* selanjutnya dan menyimpannya ke dalam *replay memory*. Jika jumlah data tersimpan pada memori lebih besar dari *mini-batch*, maka proses belajar dilakukan mengambil secara acak sampel data dari memori tersebut sebanyak *mini-batch*. Sampel data pada memori ini merupakan data-data *state* dan *action* sebelumnya dan yang sekarang sedang dijalani. Sebagai contoh pada Tabel 3.8 digunakan data acak sebanyak 32 data yang berhasil dihimpun sebelumnya.

Menggunakan cara yang sama seperti memilih nilai *Q* maksimum pada contoh sebelumnya, nilai *Q* untuk *state* selanjutnya diambil. Lalu nilai *Q* maksimum akan digunakan untuk mencari nilai *Q* yang diharapkan, dengan menggunakan perhitungan 2.11. Sebagai contoh, terdapat nilai *Q* maksimum dari *S'* adalah 0.1349, dan total *reward* pada aksi sekarang adalah 10, maka perhitungan dapat dilakukan sebagai berikut:

$$\begin{aligned}
 ExpectedValue &= r + \gamma \max_{a'} Q(s', a'; \theta^-) \\
 &= 10 + (0.9 * 0.1349) \\
 &= 10.12141
 \end{aligned}$$

Setelah mendapatkan nilai yang diharapkan, selanjutnya dilakukan perhitungan error sehingga dapat diketahui perbedaan antara nilai ekspektasi dan nilai yang dihasilkan dari jaringan. Nilai *loss function* dicari dengan menggunakan perhitungan 2.3. Perhitungan dilakukan menggunakan *mean squared error*. Variabel y' adalah nilai ekspektasi / nilai yang diharapkan. Variabel y adalah nilai keluaran yang dihasilkan oleh jaringan. Berikut contoh perhitungan *loss* dengan tabel nilai Q yang sudah diprediksi dan nilai Q yang diharapkan:

Tabel 3.9 Tabel contoh prediksi dan nilai Q yang diharapkan

State	expect	predict	State	expect	predict	State	expect	predict
1	0.1095	-0.0778	12	0.119	0.0238	23	0.1061	-0.1337
2	0.1258	-0.1204	13	0.1075	0.0442	24	0.1205	0.0976
3	0.1093	0.0845	14	0.1256	-0.1068	25	0.1123	0.0792
4	0.1091	0.122	15	0.1186	-0.0731	26	0.1196	0.0501
5	0.1212	-0.1266	16	5.1303	-0.1497	27	10.1285	0.1162
6	0.1184	-0.0302	17	0.1082	-0.0014	28	0.1191	0.0423
7	10.1214	-0.0829	18	0.1139	-0.0501	29	0.119	0.0611
8	0.113	0.1191	19	0.1121	-0.032	30	0.127	0.0611
9	0.1054	-0.0774	20	0.1172	-0.0326	31	0.1131	0.0029
10	0.1191	0.0362	21	0.1271	0.0089	32	10.1104	0.1006
11	0.1258	0.0099	22	0.1148	-0.1164			

Berikut adalah contoh perhitungan *loss function*:

$$\begin{aligned}
 E &= \frac{1}{2} (y'_i - y_i)^2 \\
 &= \frac{1}{2} * (0.1095 - (-0.0778))^2 = 0.0175
 \end{aligned}$$

Berikut tabel hasil yang didapat setelah menghitung *loss function* :

Tabel 3.10 Tabel nilai *loss function*

State	Loss	State	Loss	State	Loss	State	Loss
1	0.017540645	9	0.016708	17	0.006006	25	0.000548
2	0.03030722	10	0.003436	18	0.013448	26	0.002415
3	0.00030752	11	0.006716	19	0.010382	27	50.12308
4	0.000083205	12	0.004532	20	0.01122	28	0.002949
5	0.03070242	13	0.002003	21	0.006986	29	0.001676
6	0.01104098	14	0.027005	22	0.026727	30	0.002171
7	52.06386925	15	0.018374	23	0.028752	31	0.006072
8	0.000018605	16	13.9392	24	0.000262	32	50.09805

Setelah itu lakukan *gradient descent step* untuk mengurangi nilai rata-rata *loss*.

Karena *function activation* yang digunakan adalah *ReLU*, maka fungsi turunan dari *ReLU* adalah 1. Karena output adalah hasil linear dari *head* maka perhitungan langsung melalui *head*. Untuk mengubah *weight* lakukan proses perhitungan 2.4

$$\delta_{o_i X_i} = (59938.57 - 0.1095) * 1 * 749.2321 = 44907818.631$$

Setelah $\delta_{o_i X_i}$ ditemukan, selanjutnya dimasukkan ke dalam perhitungan 2.6 untuk menemukan v_t dari turunan tersebut:

$$v_t = 0.999 * 0 + (1 - 0.999) * (44907818.631)^2 = 2016712174211.142$$

Lalu gunakan nilai tersebut pada perhitungan 2.7 untuk mencari bobot baru:

$$\begin{aligned} \Delta w_{ji}^1 &= 1 - \frac{0.001 * 44907818.631}{(\sqrt{2016712174211.142} + 0.00000001)} \\ &= 0.0316228 \end{aligned}$$

Begitu pula pada hidden layer, gunakan *chain rule* dengan perhitungan 2.5 untuk mencari turunan untuk *hidden layer*, setelah itu proses mencari bias dan bobot baru sama seperti sebelumnya.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

Bab ini menjelaskan mengenai lingkungan pengembangan aplikasi, meliputi perangkat keras dan perangkat lunak yang digunakan untuk membuat penelitian ini, implementasi dan pengujian dari aplikasi yang dibuat.

4.1 Lingkungan Pengembangan

Dalam lingkungan pengembangan dijelaskan mengenai perangkat keras dan perangkat lunak yang digunakan dalam proses pengembangan aplikasi ini.

4.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk melakukan pengembangan aplikasi adalah sebagai berikut :

1. Laptop dengan processor Intel Core i5-8300H CPU @2.30GHz
2. RAM DDR4 8 GB
3. VGA NVIDIA GeForce GTX 1050Ti 4 GB
4. SSD dengan kapasitas 500GB
5. SSHD dengan kapasitas 1TB

4.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk melakukan pengembangan aplikasi adalah sebagai berikut :

1. Sistem Operasi Windows 10 Pro 64-bit
2. Eclipse IDE 4.12.0
3. FightingICE 4.40

4.2 Implementasi Perangkat Lunak

Dalam subbab ini, akan dijelaskan daftar *class* dan *method* yang digunakan, serta implementasi metode.

4.2.1 Daftar *Class* dan *Method*

Bagian ini menjelaskan tentang daftar *class* dan *method* yang digunakan dalam implementasi aplikasi pada penelitian ini. Berikut adalah daftar *class* dan *method* yang dibuat

dalam proses pembuatan aplikasi penerapan AI pada permainan pertarungan.

Tabel 4.1 Daftar *Class*

No	Nama Kelas	Deskripsi
1	HRA_3heads	Kelas utama yang memanggil prosedur dan fungsi lain
2	Actions	Kelas yang berisi daftar-daftar aksi yang dapat dilakukan
3	HRA	Kelas yang berisi parameter untuk metode HRA
4	Memory	Kelas yang digunakan untuk pembuatan memori
5	ReplayMemory	Kelas yang berisi metode untuk menyimpan memori
6	Layer	Kelas yang digunakan untuk membentuk pembuatan <i>layer</i> untuk jaringan saraf tiruan
7	NetUtil	Kelas yang berisi metode pembantu untuk jaringan saraf tiruan
8	Neuron	Kelas yang digunakan untuk membantu pembuatan <i>neuron</i> untuk jaringan saraf tiruan

4.2.1.1 Class *HRA_3heads*

Class ini merupakan *Class* utama yang memanggil prosedur dan fungsi lain. Pada *Class* ini terdapat *Method* sebagai berikut:

Tabel 4.2 Daftar atribut dari *class* HRA_3heads

No	Nama Atribut	Tipe Atribut	Keterangan
1	inputKey	Key	Atribut yang digunakan oleh AI yang nantinya berisi kunci masukan untuk mengendalikan AI.

2	commandCenter	CommandCenter	Atribut dari <i>class CommandCenter</i> yang mengonversikan aksi yang diterima dari AI sebagai kunci masukan dan mengaturnya setelah dikonversi.
3	frameData	FrameData	Atribut yang digunakan untuk mengambil informasi dari permainan yang berubah - ubah.
4	frameDataNon Delay	FrameData	Atribut yang digunakan untuk mengambil informasi dari permainan yang berubah - ubah yang tidak memiliki <i>delay</i> .
5	simulator	Simulator	Atribut yang menyimulasikan perkembangan pertarungan berdasarkan masukan dari FrameData.
6	player	boolean	Atribut untuk menentukan pemain utama
7	gameData	GameData	Atribut yang digunakan untuk mengambil informasi dari permainan yang tidak berubah - ubah.
8	currentFrameNum	int	Atribut untuk menyimpan nomor <i>frame</i>
9	steps_done	int	Atribut untuk menyimpan jumlah pemilihan aksi yang sudah dilakukan.
10	currentRound Number	int	Atribut untuk menyimpan ronde untuk saat ini.
11	numberOfLearn InaRound	int	Atribut untuk menyimpan jumlah pembelajaran yang sudah dilakukan pada ronde tersebut.

12	model_count_in_a_round	int	Atribut untuk menyimpan berapa banyak penggunaan model (eksplorasi) untuk menentukan aksi yang akan dipilih.
13	reward_count_in_a_round	int	Atribut untuk menyimpan berapa banyak total <i>reward</i> yang berhasil dibuat dalam ronde tersebut.
14	total_loss_in_a_round	double	Atribut untuk menyimpan berapa banyak <i>loss</i> keseluruhan pada ronde tersebut.
15	total_offense_loss_in_a_round	double	Atribut untuk menyimpan berapa banyak <i>loss</i> penyerangan pada ronde tersebut.
16	total_defense_loss_in_a_round	double	Atribut untuk menyimpan berapa banyak <i>loss</i> pertahanan pada ronde tersebut.
17	total_combo_loss_in_a_round	double	Atribut untuk menyimpan berapa banyak <i>loss</i> serangan bertubi - tubi pada ronde tersebut.
18	total_offense_Q_in_a_round	double	Atribut untuk menyimpan total nilai Q penyerangan pada ronde tersebut.
19	total_defense_Q_in_a_round	double	Atribut untuk menyimpan total nilai Q pertahanan pada ronde tersebut.
20	total_combo_Q_in_a_round	double	Atribut untuk menyimpan total nilai Q serangan bertubi - tubi pada ronde tersebut.
21	total_offense_reward_in_a_round	double	Atribut untuk menyimpan total <i>reward</i> penyerangan yang didapat pada ronde tersebut.

IV. IMPLEMENTASI DAN PENGUJIAN

22	total_defense_reward_in_a_round	double	Atribut untuk menyimpan total <i>reward</i> pertahanan yang didapat pada ronde tersebut.
23	total_combo_reward_in_a_round	double	Atribut untuk menyimpan total <i>reward</i> serangan bertubi - tubi yang didapat pada ronde tersebut.
24	win	int	Atribut untuk menyimpan informasi kemenangan, 1 untuk menang dan 0 jika kalah.
25	offenseReward	dobule	Atribut untuk menyimpan nilai <i>reward</i> penyerangan.
26	defenseReward	double	Atribut untuk menyimpan nilai <i>reward</i> pertahanan.
27	comboReward	double	Atribut untuk menyimpan nilai <i>reward</i> serangan bertubi - tubi.
28	mainLayer	layer[]	Atribut objek <i>main layer</i> jaringan saraf tiruan.
29	offLayer	layer[]	Atribut objek <i>offense layer</i> jaringan saraf tiruan.
30	defLayer	layer[]	Atribut objek <i>defense layer</i> jaringan saraf tiruan.
31	comLayer	layer[]	Atribut objek <i>combo layer</i> jaringan saraf tiruan.
32	targetLayer	layer[]	Atribut objek target dari <i>main layer</i> jaringan saraf tiruan.
33	targetoffLayer	layer[]	Atribut objek target dari <i>offense layer</i> jaringan saraf tiruan.

34	targetDefLayer	layer[]	Atribut objek target dari <i>defense layer</i> jaringan saraf tiruan.
35	targetComLayer	layer[]	Atribut objek <i>combo layer</i> jaringan saraf tiruan.
36	mylastHP	int	Atribut untuk menyimpan darah pemain pada <i>state</i> sebelumnya.
37	oppLastHP	int	Atribut untuk menyimpan darah lawan pada <i>state</i> sebelumnya
38	replayMemory	ReplayMemory	Atribut inisialisasi objek untuk ReplayMemory
39	actionList	Action[]	Atribut yang akan diisi oleh kumpulan aksi.
40	action	int	Atribut yang menyimpan <i>index</i> dari sebuah aksi.
41	isGameJustStarted	boolean	Atribut untuk menentukan apakah awal permainan atau bukan.
42	avgloss	double	Atribut nilai rata - rata dari <i>minibatch loss</i>
43	avgoffloss	double	Atribut nilai rata - rata dari <i>minibatch loss</i> untuk penyerangan
44	avgdefloss	double	Atribut nilai rata - rata dari <i>minibatch loss</i> untuk pertahanan
45	avgcomloss	double	Atribut nilai rata - rata dari <i>minibatch loss</i> untuk penyerangan serangan bertubi - tubi.
46	my	CharInfo	Atribut inisialisasi objek yang berisi data karakter pemain.

IV. IMPLEMENTASI DAN PENGUJIAN

47	opp	CharInfo	Atribut inisialisasi objek yang berisi data karakter lawan.
48	state	ArrayList<Double>	Atribut objek dari <i>state</i> pada permainan.
49	generator	Random	Atribut objek untuk menghasilkan nilai acak
50	seed	long	Atribut untuk menentukan <i>seed</i> pada inisialisasi bobot secara acak
51	trainedFolder	String	Atribut yang berisi nomor <i>folder</i> yang akan digunakan.

Tabel 4.3 Daftar *Method class* HRA_3heads

No	Nama <i>Method</i>	Tipe Masukan	Tipe Keluaran	Keterangan
1	close			Metode yang digunakan untuk menyelesaikan AI pada akhir dari permainan.
2	getInformation	FrameData, boolean, FrameData		Metode yang digunakan untuk mendapatkan informasi dari status permainan dari masing-masing <i>frame</i> .
3	initialize	GameData, boolean	integer	Metode untuk menginisialisasi AI dan akan dijalankan hanya sekali pada saat awal permainan.

4	input		Key	Metode untuk mengambil input dari AI.
5	processing			Metode yang digunakan untuk mengolah data dari AI, dieksekusi setiap <i>frame</i> .
6	roundEnd	integer, integer, integer		Metode yang digunakan untuk memberikan hasil dari masing-masing ronde.
7	forward	ArrayList<Double>		Metode untuk perhitungan maju ANN.
8	forwardTarget	ArrayList<Double>		Metode untuk perhitungan maju ANN target.
9	isWin			Metode untuk mengecek apakah AI menang atau tidak
10	getObservation		ArrayList<Double>	Metode untuk mengambil data permainan dari setiap <i>frame</i> .
11	ableAction		boolean	Metode untuk mengecek apakah AI dapat bergerak.

12	selectAction	ArrayList<Double>	integer	Metode untuk memilih nomor aksi.
13	setLastHP			Metode untuk mengambil data darah pemain tiap <i>frame</i> .
14	runAction	integer		Metode untuk menjalankan aksi.
15	setLastCombo			Metode untuk mengambil data combo permainan dari setiap <i>frame</i> .
16	makeReward			Metode untuk membuat <i>reward</i> untuk agen AI.
17	pushData	ArrayList<Double>, double, double, double		Metode untuk memasukkan data <i>state</i> kedalam memori.
18	learn			Metode untuk melakukan pembelajaran AI.
19	saveNetwork			Metode untuk menyimpan bobot ANN.
20	loadWeight			Metode untuk memuat bobot untuk ANN.

21	selectMaxAction	ArrayList<Double>	integer	Metode untuk memilih aksi langsung dari ANN.
22	saveRoundResult			Metode untuk menyimpan hasil ronde permainan.
24	sumGradient	integer, integer, Layer	double	Metode untuk menghitung jumlah gradien ANN
25	backwardOffense	double, ArrayList<Object>, int		Metode untuk perhitungan mundur ANN pada agen <i>Offense</i> .
26	backwardDefense	double, ArrayList<Object>, int		Metode untuk perhitungan mundur ANN pada agen <i>Defense</i> .
27	backwardCombo	double, ArrayList<Object>, int		Metode untuk perhitungan mundur ANN pada agen <i>Combo</i> .
28	backwardMain	double, ArrayList<Object>, int		Metode untuk perhitungan mundur ANN pada MainLayer.

28	rmsPropOptimize			Metode untuk perhitungan perubahan bobot ANN pada secara keseluruhan.
29	getAttack Observation			Metode untuk mencatat aksi atau serangan yang dilakukan.

4.2.1.2 Class Actions

Class ini berisi atribut yang di dalamnya daftar aksi yang dapat dilakukan oleh AI, berdasarkan dari *framework FightingICE*. Tipe atribut yang terdapat pada *class* ini adalah *array* dari *Action*.

4.2.1.3 Class HRA

Class ini berisi atribut yang di dalamnya terdapat parameter yang dibutuhkan untuk *Method* HRA. Berikut adalah isi dari atribut yang ada pada *class* ini.

Tabel 4.4 Daftar atribut dari *class* HRA

No	Nama Atribut	Tipe Atribut	Keterangan
1	Memory_Size	int	Atribut yang berisi jumlah kapasitas memori penyimpanan untuk <i>state</i> . Memiliki nilai 100000 sebagai nilai maksimum yang dapat ditampung.
2	Input_Size	int	Atribut yang berisi jumlah masukan. Memiliki nilai 142 berdasarkan jumlah masukan yang diambil dari permainan.
3	Hidden_Size	int	Atribut yang berisi jumlah <i>neuron</i> pada <i>hidden layer</i> . Memiliki nilai 80.

4	Output_Size	int	Atribut yang berisi jumlah <i>neuron</i> pada <i>hidden layer</i> . Memiliki nilai 40 sesuai keluaran aksi yang dapat dilakukan secara langsung.
5	Batch_Size	int	Jumlah kumpulan transisi yang diambil untuk dipelajari, berjumlah 32.
6	Epsilon_Start	double	Nilai awalan dari <i>epsilon greedy</i> , bernilai 0.9.
7	Epsilon_End	double	Nilai akhir dari <i>epsilon greedy</i> , bernilai 0.05.
8	Epsilon_Decay	int	Jumlah pembelajaran yang dilakukan untuk mengurangi nilai <i>epsilon greedy</i> dari nilai awal menuju nilai akhir. Bernilai 10000.
9	Gamma	double	Nilai faktor diskon untuk <i>reward</i> , bernilai 0.9
10	Learn_Rate	double	Nilai untuk optimisasi jaringan saraf tiruan, memiliki nilai 0.001
11	Save_Frequency	int	Jumlah frekuensi untuk menyimpan bobot dari model yang sudah dilatih, bernilai 50.
12	Train_Mode	int	Atribut untuk menentukan mode pelatihan, 0 untuk mode percobaan, 1 untuk mode pelatihan
13	Load_Weight	int	Atribut untuk menentukan pemuat bobot, 0 untuk tidak memuat bobot, 1 untuk memuat bobot

4.2.1.4 Class *Memory*

Class ini berisi fungsi dan *Method* objek untuk pembuatan memori. Atribut dan *Method* yang terdapat pada *class* ini adalah sebagai berikut.

Tabel 4.5 Daftar atribut dari *class Memory*

No	Nama Atribut	Tipe Atribut	Keterangan
1	state	ArrayList<Double>	Atribut untuk menyimpan <i>state</i> dari permainan.
2	actionIndex	int	Atribut untuk menyimpan indeks dari aksi permainan
3	next_state	ArrayList<Double>	Atribut untuk menyimpan <i>state</i> selanjutnya dari permainan.
4	offenseReward	double	Atribut untuk menyimpan <i>reward</i> dari penyerangan pada <i>state</i> tersebut.
5	defenseReward	double	Atribut untuk menyimpan <i>reward</i> dari pertahanan pada <i>state</i> tersebut.
6	comboReward	double	Atribut untuk menyimpan <i>reward</i> dari serangan bertubi - tubi pada <i>state</i> tersebut.

Tabel 4.6 Daftar *Method* *class Memory*

No	Nama <i>Method</i>	Tipe Masukan	Tipe Keluaran	Keterangan
1	Memory	ArrayList<Double>, integer, ArrayList<Double>, integer, integer, integer		Metode yang digunakan untuk melakukan inisialisasi sebuah memori.
2	getState		ArrayList<Double>	Metode yang digunakan untuk menampilkan state.

IV. IMPLEMENTASI DAN PENGUJIAN

3	setState	ArrayList<Double>		Metode untuk mengisi state.
4	getActionIndex		int	Metode yang digunakan untuk menampilkan actionIndex.
5	setActionIndex	int		Metode untuk mengisi actionIndex.
6	getNext_State		ArrayList<Double>	Metode yang digunakan untuk menampilkan state.
7	setNext_State	ArrayList<Double>		Metode untuk mengisi next_state.
8	getOffenseReward		double	Metode yang digunakan untuk menampilkan offenseReward.
9	setOffenseReward	double		Metode untuk mengisi offenseReward.
10	getDefenseReward		double	Metode yang digunakan untuk menampilkan defenseReward.
11	setDefenseReward	double		Metode untuk mengisi defenseReward.

12	getComboReward		double	Metode yang digunakan untuk menampilkan comboReward.
13	setComboReward	double		Metode untuk mengisi comboReward.

4.2.1.5 Class *ReplayMemory*

Class ini berisi fungsi dan *Method* untuk membuat dan menyimpan sebuah kumpulan memori. Atribut dan *Method* yang terdapat pada *Class* ini adalah.

Tabel 4.7 Daftar atribut dari *class ReplayMemory*

No	Nama Atribut	Tipe Atribut	Keterangan
1	capacity	int	Atribut untuk menyimpan kapasitas dari <i>ReplayMemory</i> .
2	memory	ArrayList<Memory>	Atribut untuk menyimpan beberapa objek dari <i>memoryFragments</i> .
3	memoryFragments	Memory	Atribut untuk menyimpan objek <i>Memory</i> .
4	copyMemory	ArrayList<Memory>	Atribut untuk menyimpan duplikat dari <i>memory</i> .
5	batchMemory	ArrayList<Memory>	Atribut untuk menyimpan <i>memory</i> sejumlah dari <i>batch</i> yang sudah ditentukan.

Tabel 4.8 Daftar Method class *ReplayMemory*

No	Nama Method	Tipe Masukan	Tipe Keluaran	Keterangan
1	ReplayMemory	integer		Metode yang digunakan untuk melakukan inisialisasi sebuah ReplayMemori.
2	push	ArrayList<Double>, int, ArrayList<Double>, double, double, double		Metode yang digunakan untuk memasukkan data <i>state</i> kedalam memori.
3	sample	int	ArrayList<memory>	Metode untuk mengambil sampel dari ReplayMemory.

4.2.1.6 Class *Layer*

Class ini memiliki *Method* dan fungsi untuk membangun sebuah *Layer* untuk Jaringan Saraf Tiruan. Berikut adalah tabel atribut dan *Method* yang terdapat pada *class* ini.

Tabel 4.9 Daftar attribut dari *class Layer*

No	Nama Atribut	Tipe Atribut	Keterangan
1	neurons	Neuron[]	Atribut untuk menyimpan kumpulan objek neuron.
2	neuron	Neuron	Atribut inisialisasi objek Neuron.
3	ran	Random	Atribut untuk menghasilkan nilai acak.
4	gaussian	double	Atribut yang berisi nilai acak yang terdistribusi secara normal.

5	reluweight	double	Atribut yang berisi bobot untuk fungsi aktivasi ReLU.
5	ni	double	Atribut yang berisi jumlah masukan dari sebuah neuron.

Tabel 4.10 Daftar *Method class Layer*

No	Nama <i>Method</i>	Tipe Masukan	Tipe Keluaran	Keterangan
1	Layer	ArrayList<Double>		Metode yang digunakan untuk melakukan inisialisasi <i>input layer</i> .
2	Layer	int, int, long		Metode yang digunakan untuk melakukan inisialisasi <i>hidden layer</i> dan <i>output layer</i> .
3	Layer	Layer		Metode untuk melakukan pengklonaan sebuah <i>layer</i> .
4	reluUniform	int	double	Metode yang digunakan untuk melakukan inisialisasi bobot untuk fungsi aktivasi ReLU.

4.2.1.7 Class *NetUtil*

Class ini memiliki *Method* dan fungsi pembantu untuk Jaringan Saraf Tiruan. Berikut ini adalah tabel yang berisi atribut dan *Method* pada *class* ini.

Tabel 4.11 Daftar atribut dari *class* *NetUtil*

No	Nama Atribut	Tipe Atribut	Keterangan
1	beta1	double	Nilai pengurangan eksponensial untuk momen pertama, bernilai 0.9.
2	beta2	double	Nilai pengurangan eksponensial untuk momen kedua, bernilai 0.99.
3	epsilon	double	Nilai epsilon, bernilai 0.000000010.

Tabel 4.12 Daftar *Method* *class* *NetUtil*

No	Nama <i>Method</i>	Tipe Masukan	Tipe Keluaran	Keterangan
1	Relu	double	double	Metode yang digunakan untuk melakukan fungsi aktivasi ReLU.
2	ReluDerivative	double	double	Metode yang digunakan untuk mencari fungsi turunan dari ReLU.
3	squaredLoss	double, double	double	Metode untuk menghitung <i>Squared Loss</i>
4	squaredError Derivative	double, double	double	Metode untuk menghitung nilai turunan dari <i>Squared Loss</i> .

5	updateBiasSecond Moment	double, double	double	Metode untuk menghitung <i>cache</i> dari gradien.
6	weightUpdater	double, double	double	Metode untuk menghitung perubahan bobot yang dibutuh.

4.2.1.8 Class *Neuron*

Class ini memiliki *Method* dan fungsi untuk membangun *neuron* untuk Jaringan Saraf Tiruan. Berikut adalah daftar tabel yang berisi atribut, metode, dan fungsi pada *class* ini.

Tabel 4.13 Daftar atribut dari *class Neuron*

No	Nama Atribut	Tipe Atribut	Keterangan
1	weights	double[]	Atribut yang menyimpan nilai bobot untuk <i>neuron</i> .
2	cache_weights	double[]	Atribut yang menyimpan nilai sementara hasil perubahan bobot.
3	gradient	double	Atribut yang menyimpan nilai gradien keluaran dan turunan fungsi aktivasi.
4	value	double	Atribut yang menyimpan nilai masukan neuron.
5	bias	double	Atribut yang menyimpan bias.
6	biasnSecond MomentEstimate	double	Atribut yang menyimpan bias yang dikoreksi.
7	dB	double	Atribut yang menyimpan nilai gradien bias keluaran dan turunan fungsi aktivasi.

8	accumulated WeightsGrad	double[]	Atribut yang menyimpan akumulasi gradien total.
9	secondMoment Estimate	double[]	Atribut yang menyimpan gradien yang dikoreksi.

Tabel 4.14 Daftar *Method class Neuron*

No	Nama <i>Method</i>	Tipe Masukan	Tipe Keluaran	Keterangan
1	Neuron	double[], double[], double, double[]		Metode yang digunakan untuk melakukan inisialisasi <i>neuron</i> pada <i>hidden layer</i> dan <i>output layer</i> .
2	Neuron	double		Metode yang digunakan untuk melakukan insialisasi <i>neuron</i> untuk masukan.

4.2.2 Implementasi Metode

Pada tahap ini akan dibahas mengenai langkah-langkah implementasi *Method HRA* pada penerapan AI untuk permainan pertarungan.

1. Masukan *State* Permainan.

Pada tahap ini setiap *State* permainan akan diambil per-frame, pemilihan aksi secara acak, lalu menyimpan data darah pada frame yang sedang berjalan.

2. Normalisasi *State* Permainan.

Pada tahap ini dilakukan normalisasi *State* permainan pada rentang 0 dan 1 agar dapat digunakan sebagai masukan untuk ANN.

3. Pemilihan Aksi.

Langkah selanjutnya adalah pemilihan aksi secara acak, lalu observasi hasil yang didapatkan setelah aksi dijalankan.

4. Perhitungan *Reward*.

Pada tahap ini, dilakukan perhitungan reward berdasarkan aksi yang telah dilakukan.

5. Masukan *State* Permainan selanjutnya.

Setelah pemilihan aksi secara acak maka selanjutnya akan ada *State* permainan yang terbentuk sehingga dilakukan kembali proses pengambilan *State* permainan pada *frame* saat ini.

6. Penyimpanan Transisi.

Game state pada awal permainan, aksi yang dilakukan, *reward* yang didapat dan *State* permainan hasil dari aksi akan disimpan ke dalam memori yang sudah ditentukan kapasitasnya sebagai transisi.

7. Pengambilan *minibatch* dari transisi.

Setelah jumlah transisi yang didapat sebanyak 32 transisi, maka selanjutnya akan dipilih secara acak 32 transisi untuk digunakan sebagai data latih.

8. Pelatihan Jaringan Saraf Tiruan.

Dalam tahapan ini transisi yang sudah dipilih secara acak akan digunakan sebagai data latih. Penghitungan *loss* dilakukan dengan *Method squared loss* sesuai dengan perhitungan 2.3. Lalu ubah bobot pada jaringan saraf tiruan target setiap 100 langkah dengan menggunakan bobot dari jaringan saraf tiruan yang sudah dilatih.

4.3 Skenario Pengujian

Bagian ini akan menjelaskan pengujian yang akan dilakukan dengan aplikasi yang telah dibuat. Hasil akhir dari pengujian ini adalah untuk melihat berapa rasio kemenangan yang didapat jika ditandingkan dengan AI lain dan juga hasil akhir setelah diuji terhadap lawan manusia. Skenario pengujian yang dilakukan adalah pengujian persentase kemenangan dari tiap bobot berdasarkan lama waktu pelatihan, dan menguji tingkat akurasi serangan terhadap AI lawan. Proses pada pengujian juga akan menggunakan karakter yang berbeda untuk menguji performa AI terhadap karakter tersebut. Terdapat 3 Karakter yang tersedia pada *FightingICE*, namun pengujian yang dilakukan hanya menggunakan karakter *ZEN* dan *GARNET*. Karakter *LUD* tidak dapat digunakan karena pada saat pengujian pertama, baik AI HRA maupun AI yang bersifat *rule-based* tidak dapat menggerakkan karakter tersebut sehingga dapat disimpulkan bahwa karakter tersebut tidak dapat digunakan.

Masing - masing pengujian akan dilakukan sebanyak 10 ronde. AI yang digunakan sebagai lawan tanding merupakan AI latih bersifat *rule-based*. Setiap *Rule* yang digunakan oleh AI ini berdasarkan jarak terhadap lawan, jumlah energi musuh, dan energi dari AI itu sendiri. Jika musuhnya terlalu jauh, maka AI akan mendekat dengan cara melompat ke depan. Jika musuh jauh tetapi tidak terlalu jauh, maka AI akan mendekat dengan cara berlari ke depan. Ketika energi telah mencapai *threshold* maka AI akan melakukan aksi sesuai dengan jumlah energi yang dimiliki.

AI ini memiliki satu aturan bertahan hidup yang sangat penting, ketika energi lawan mencapai nilai 300, maka AI akan mencoba untuk menghindari serangan bola energi yang dapat musuh luncurkan dengan energi sebanyak 300 poin. Jika ketika kondisi tadi tidak terpenuhi maka AI akan melakukan serangan tendangan yang dapat membantu dengan 2 cara, yaitu serangan acak, sehingga musuh tidak dapat memprediksi secara akurat pergerakan AI, dan AI tidak akan pernah berdiri dan diam, AI akan terus melakukan tendangan yang akan meningkatkan kemungkinannya untuk memenangi pertarungan.

4.3.1 AI vs AI Lawan Latihnya

Skenario pengujian ini dilakukan untuk melihat seberapa besar kemenangan yang didapat dengan melawan musuh latihnya. Terdapat 2 karakter yang akan digunakan dalam pengujian pada *FightingICE*, yaitu *ZEN* dan *GARNET*. Berikut ini adalah tabel hasil pengujian dengan jumlah pelatihan sebanyak 50 ronde.

Tabel 4.15 Pengujian hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Kalah	Menang	51
2	Kalah	Menang	51
3	Kalah	Menang	51
4	Kalah	Menang	51
5	Kalah	Menang	51
6	Kalah	Menang	51
7	Kalah	Menang	51
8	Kalah	Menang	51
9	Kalah	Menang	51
10	Kalah	Menang	51
Rata - rata selisih darah			51
Persentase Kemenangan	0 %	100 %	

Persentase Seri	0 %	0 %	
Persentase Kalah	100 %	0 %	

Tabel 4.16 Pengujian hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Menang	Kalah	201
2	Menang	Kalah	329
3	Menang	Kalah	329
4	Menang	Kalah	329
5	Menang	Kalah	329
6	Menang	Kalah	329
7	Menang	Kalah	329
8	Menang	Kalah	329
9	Menang	Kalah	329
10	Menang	Kalah	329
Rata - rata selisih darah			316.2
Persentase Kemenangan	100 %	0 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	0 %	100 %	

Pada Tabel 4.15, AI yang sudah dilatih sebanyak 50 ronde tidak dapat bertahan melawan AI latihnya dengan menggunakan karakter *ZEN*, akan tetapi pada pengujian dengan menggunakan karakter *GARNET*, AI dapat melawan AI latihnya. Sehingga total persentase

IV. IMPLEMENTASI DAN PENGUJIAN

kemenangan yang didapat oleh AI dengan waktu latih sebanyak 50 ronde adalah 50%.

Tabel 4.17 Pengujian hasil latih 50 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Kalah	Menang	157
2	Kalah	Menang	85
3	Kalah	Menang	157
4	Kalah	Menang	85
5	Kalah	Menang	157
6	Kalah	Menang	85
7	Kalah	Menang	157
8	Kalah	Menang	85
9	Kalah	Menang	157
10	Kalah	Menang	85
Rata - rata selisih darah			121
Persentase Kemenangan	0 %	100 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	100 %	0 %	

Tabel 4.18 Pengujian hasil latih 50 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Kalah	Menang	229
2	Kalah	Menang	24

IV. IMPLEMENTASI DAN PENGUJIAN

3	Menang	Kalah	13
4	Kalah	Menang	124
5	Kalah	Menang	35
6	Kalah	Menang	229
7	Kalah	Menang	24
8	Menang	Kalah	13
9	Kalah	Menang	124
10	Kalah	Menang	35
Rata - rata selisih darah			85
Persentase Kemenangan	20 %	80 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	80 %	20 %	

Pengujian yang dilakukan pada Tabel 4.17 adalah AI dengan nilai fungsi 1 unit. Sama seperti pengujian pada Tabel 4.15, AI tidak dapat bertahan melawan AI latihnya dengan menggunakan karakter *ZEN*, namun pada pengujian dengan karakter *GARNET*, persentase kemenangan diraih sebesar 20%. Total persentase kemenangan yang didapat oleh AI ini sebesar 10%.

Selanjutnya yang akan diuji adalah AI yang sudah dilatih sebanyak 150 ronde dengan pengujian yang sama seperti AI pada 50 ronde.

Tabel 4.19 Pengujian hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Kalah	Menang	15
2	Kalah	Menang	131

IV. IMPLEMENTASI DAN PENGUJIAN

3	Kalah	Menang	15
4	Kalah	Menang	131
5	Kalah	Menang	15
6	Kalah	Menang	131
7	Kalah	Menang	15
8	Kalah	Menang	131
9	Kalah	Menang	15
10	Kalah	Menang	131
Rata - rata selisih darah			73
Persentase Kemenangan	0 %	100 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	100 %	0 %	

Tabel 4.20 Pengujian hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Menang	Kalah	127
2	Menang	Kalah	127
3	Menang	Kalah	127
4	Menang	Kalah	127
5	Menang	Kalah	127
6	Menang	Kalah	127

7	Menang	Kalah	127
8	Menang	Kalah	127
9	Menang	Kalah	127
10	Menang	Kalah	127
Rata - rata selisih darah			127
Persentase Kemenangan	100 %	0 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	0 %	100 %	

Pada Tabel 4.19, dan 4.20 terlihat tidak ada perubahan persentase kemenangan, namun adanya perubahan selisih darah terutama pada pengujian karakter *ZEN* menunjukkan adanya perubahan bobot. Maka dari itu, proses pelatihan akan dilakukan kembali dengan menambah pelatihan sebanyak 100 ronde permainan.

Tabel 4.21 Pengujian hasil latih 150 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Kalah	Menang	155
2	Kalah	Menang	142
3	Kalah	Menang	142
4	Kalah	Menang	142
5	Kalah	Menang	142
6	Kalah	Menang	142
7	Kalah	Menang	142
8	Kalah	Menang	142

IV. IMPLEMENTASI DAN PENGUJIAN

9	Kalah	Menang	142
10	Kalah	Menang	142
Rata - rata selisih darah			143.3
Persentase Kemenangan	0 %	100 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	100 %	0 %	

Tabel 4.22 Pengujian hasil latih 150 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Kalah	Menang	5
2	Kalah	Menang	5
3	Kalah	Menang	5
4	Kalah	Menang	5
5	Kalah	Menang	5
6	Kalah	Menang	5
7	Kalah	Menang	5
8	Kalah	Menang	5
9	Kalah	Menang	5
10	Kalah	Menang	5
Rata - rata selisih darah			5
Persentase Kemenangan	0 %	100 %	

Persentase Seri	0 %	0 %	
Persentase Kalah	100 %	0 %	

Pada Tabel 4.21 dan Tabel 4.22 terdapat perubahan persentase pada AI dengan 1 unit nilai fungsi yang memiliki nilai total persentase kemenangannya 0%. Hal ini terjadi karena adanya perubahan bobot pada jaringan saraf tiruan yang berusaha mencari nilai fungsi Q terbaik, namun karena nilai fungsi *reward* pada AI ini tidak optimal, maka penurunan performa tersebut dapat terjadi dan pada iterasi pelatihan berikutnya performa seharusnya dapat meningkat kembali.

Tabel 4.23 Pengujian hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Kalah	Menang	20
2	Menang	Kalah	60
3	Kalah	Menang	20
4	Menang	Kalah	60
5	Kalah	Menang	20
6	Menang	Kalah	60
7	Menang	Kalah	60
8	Menang	Kalah	60
9	Kalah	Menang	20
10	Menang	Kalah	60
Rata - rata selisih darah			40
Persentase Kemenangan	60 %	40 %	
Persentase Seri	0 %	0 %	

Persentase Kalah	40 %	60 %	
------------------	------	------	--

Tabel 4.24 Pengujian hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Menang	Kalah	101
2	Menang	Kalah	101
3	Menang	Kalah	101
4	Menang	Kalah	101
5	Menang	Kalah	101
6	Menang	Kalah	101
7	Menang	Kalah	101
8	Menang	Kalah	101
9	Menang	Kalah	101
10	Menang	Kalah	101
Rata - rata selisih darah			101
Persentase Kemenangan	100 %	0 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	0 %	100 %	

Pada pengujian yang dilakukan oleh AI yang telah dilatih sebanyak 250 ronde, persentase kemenangan menggunakan karakter *ZEN* meningkat menjadi 60%, dan persentase kemenangan dengan karakter *GARNET* tidak berubah. Selisih darah pada saat kondisi kalah mengalami pengurangan dari pengujian sebelumnya. Sehingga total persentase kemenangan yang didapat adalah 80%.

IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.25 Pengujian hasil latih 250 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Kalah	Menang	5
2	Kalah	Menang	5
3	Kalah	Menang	5
4	Kalah	Menang	5
5	Kalah	Menang	5
6	Kalah	Menang	5
7	Kalah	Menang	5
8	Kalah	Menang	5
9	Kalah	Menang	5
10	Kalah	Menang	5
Rata - rata selisih darah			5
Persentase Kemenangan	0 %	100 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	100 %	0 %	

Tabel 4.26 Pengujian hasil latih 250 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Menang	Kalah	118
2	Menang	Kalah	58
3	Menang	Kalah	118

IV. IMPLEMENTASI DAN PENGUJIAN

4	Menang	Kalah	58
5	Menang	Kalah	118
6	Menang	Kalah	58
7	Menang	Kalah	118
8	Menang	Kalah	58
9	Menang	Kalah	118
10	Menang	Kalah	58
Rata - rata selisih darah			88
Persentase Kemenangan	100 %	0 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	0 %	100 %	

Pengujian pada Tabel 4.25 dan 4.26 menunjukkan bahwa terjadinya peningkatan performa pada AI dengan 1 unit fungsi nilai ketika dilatih sebanyak 250 ronde. Walaupun persentase kemenangan pada karakter *ZEN* adalah 0%, AI dapat mengurangi selisih darah pada saat pertarungan. Hal yang berbeda terjadi pada penggunaan karakter *GARNET*, dengan persentase kemenangan sebesar 100%. Sehingga total persentase yang didapat adalah 50%.

Tabel 4.27 Pengujian hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Menang	Menang	25
2	Menang	Kalah	40
3	Menang	Menang	70
4	Menang	Kalah	25

IV. IMPLEMENTASI DAN PENGUJIAN

5	Menang	Menang	70
6	Menang	Kalah	20
7	Menang	Kalah	30
8	Menang	Kalah	115
9	Menang	Menang	60
10	Menang	Kalah	20
Rata - rata selisih darah			47.5
Persentase Kemenangan	100 %	0 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	0 %	100 %	

Tabel 4.28 Pengujian hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Menang	Kalah	292
2	Menang	Kalah	45
3	Menang	Kalah	45
4	Menang	Kalah	45
5	Menang	Kalah	45
6	Menang	Kalah	45
7	Menang	Kalah	45
8	Menang	Kalah	45

9	Menang	Kalah	45
10	Menang	Kalah	45
Rata - rata selisih darah			69.7
Persentase Kemenangan	100 %	0 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	0 %	100 %	

Pada pengujian Tabel 4.27 menunjukkan persentase kemenangan pada karakter *ZEN* setelah dilatih sebanyak 350 ronde adalah 100%, sehingga dapat disimpulkan AI berhasil meningkatkan persentase kemenangannya sebanyak 40%. Total persentase yang didapat oleh AI dengan 3 unit fungsi ini adalah 100%.

Tabel 4.29 Pengujian hasil latih 350 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	AI HRA	AI Latih	Selisih Darah
1	Menang	Kalah	5
2	Menang	Kalah	86
3	Menang	Kalah	5
4	Menang	Kalah	86
5	Menang	Kalah	5
6	Menang	Kalah	86
7	Menang	Kalah	5
8	Menang	Kalah	86
9	Menang	Kalah	5
10	Menang	Kalah	86

IV. IMPLEMENTASI DAN PENGUJIAN

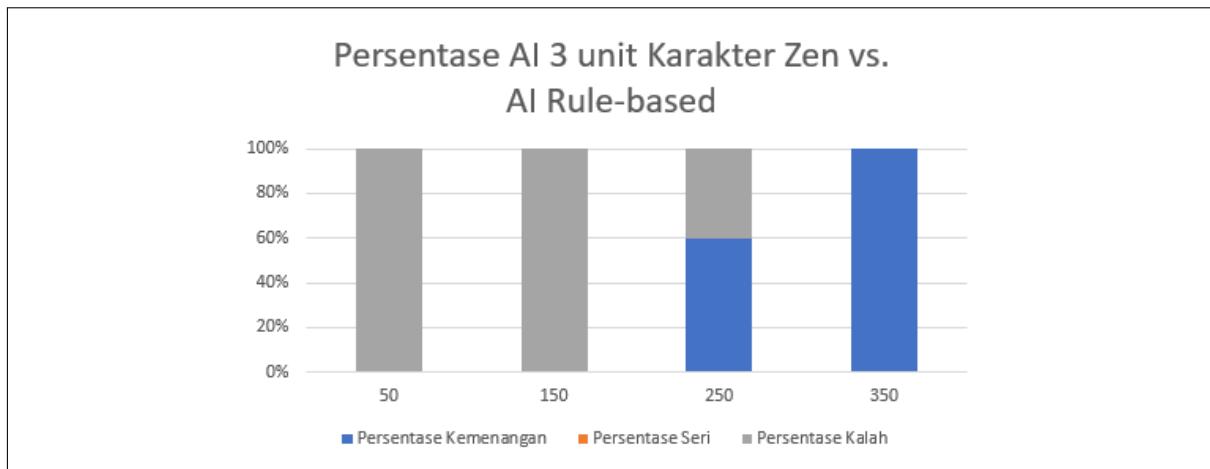
Rata - rata selisih darah			45.5
Persentase Kemenangan	100 %	0 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	0 %	100 %	

Tabel 4.30 Pengujian hasil latih 350 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

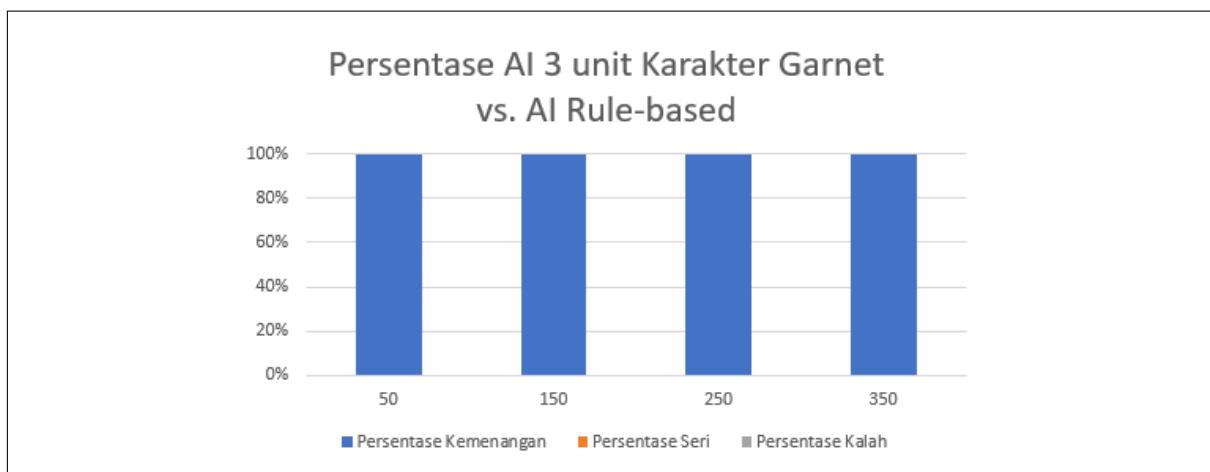
Pengujian	AI HRA	AI Latih	Selisih Darah
1	Menang	Kalah	10
2	Menang	Kalah	10
3	Menang	Kalah	10
4	Menang	Kalah	10
5	Menang	Kalah	10
6	Menang	Kalah	10
7	Menang	Kalah	10
8	Menang	Kalah	10
9	Menang	Kalah	10
10	Menang	Kalah	27
Rata - rata selisih darah			11.7
Persentase Kemenangan	100 %	0 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	0 %	100 %	

IV. IMPLEMENTASI DAN PENGUJIAN

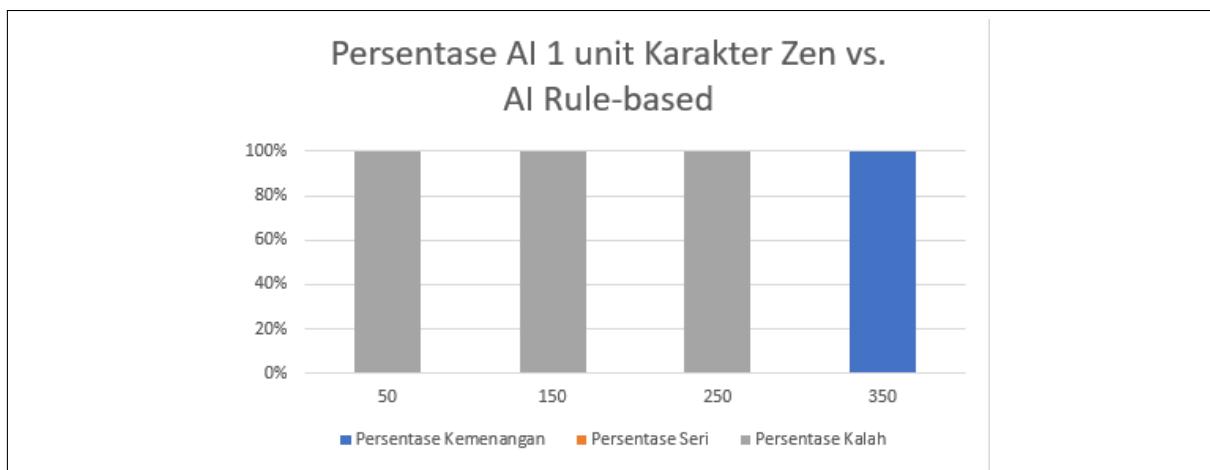
Pada Tabel 4.29, AI dengan 1 unit fungsi baru dapat memenangi permainan pada pelatihan sebanyak 350 ronde, dengan kemenangan mutlak sebanyak 10 kali dari 10 percobaan. Sehingga total persentase kemenangan yang didapat oleh AI ini sebanyak 100%.



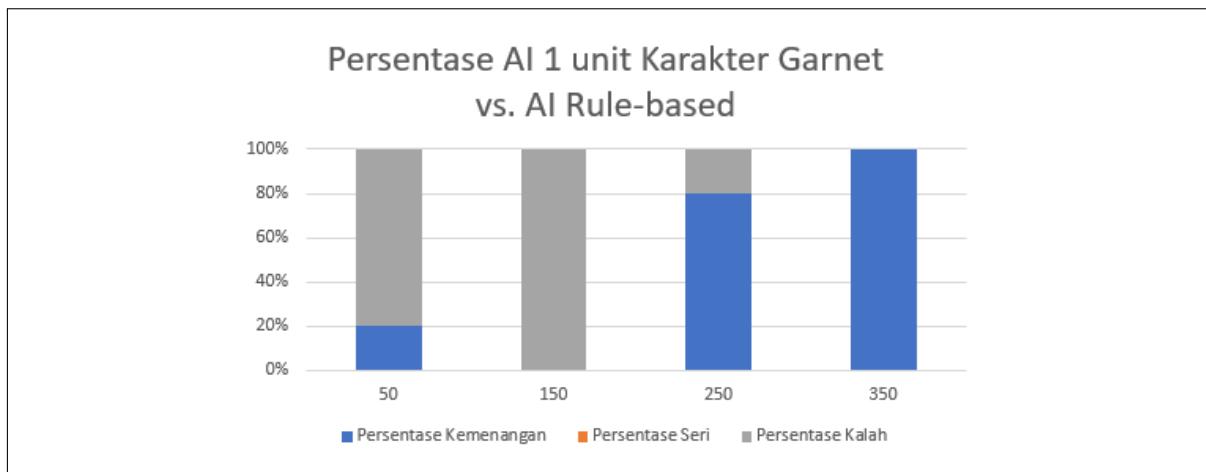
Gambar 4.1 Grafik persentase AI 3 unit karakter *Zen*



Gambar 4.2 Grafik persentase AI 3 unit karakter *Garnet*



Gambar 4.3 Grafik persentase AI 1 unit karakter *Zen*

**Gambar 4.4** Grafik persentase AI 1 unit karakter *Garnet*

Dari pengujian di atas, dapat disimpulkan bahwa pembagian nilai fungsi ke dalam beberapa bagian dapat membantu agar AI dapat mempelajari permainan dengan cepat. Oleh karena itu, pada pengujian dengan melawan pemain manusia selanjutnya akan menggunakan AI dengan 3 unit fungsi dikarenakan performanya yang cukup baik melawan *rule-based* AI dengan persentase kemenangan total 100%.

4.3.2 Akurasi Serangan

Dalam skenario pengujian ini akan dilakukan pengecekan terhadap aksi yang dilakukan oleh AI, apakah akurasi aksi berpengaruh terhadap kemenangan atau tidak. Jumlah perhitungan akurasi aksi yang terkena adalah jumlah pukulan yang kena dibagi dengan jumlah pukulan yang dilakukan. Untuk mencari perhitungan berapa banyak pukulan yang dilakukan, akan digunakan fungsi pada *FightingICE* untuk mencari serangan yang dilakukan yaitu *isHitConfirm* pada masing - masing ronde lalu disimpan datanya setiap akhir ronde. Berikut adalah pengujian akurasi serangan terhadap AI lawan latihnya.

Tabel 4.31 Pengujian akurasi aksi hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	77	43	34	55.84%	Kalah
2	77	43	34	55.84%	Kalah
3	77	43	34	55.84%	Kalah
4	77	43	34	55.84%	Kalah
5	77	43	34	55.84%	Kalah
6	77	43	34	55.84%	Kalah

IV. IMPLEMENTASI DAN PENGUJIAN

7	77	43	34	55.84%	Kalah
8	77	43	34	55.84%	Kalah
9	77	43	34	55.84%	Kalah
10	77	43	34	55.84%	Kalah

Tabel 4.32 Pengujian akurasi aksi hasil latih 50 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	25	20	5	80%	Kalah
2	29	23	6	79.31%	Kalah
3	32	23	9	71.88%	Kalah
4	21	17	4	80.95%	Kalah
5	25	20	5	80%	Kalah
6	29	23	6	79.31%	Kalah
7	32	23	9	71.88%	Kalah
8	21	17	4	80.95%	Kalah
9	25	20	5	80%	Kalah
10	29	23	6	79.31%	Kalah

Tabel 4.33 Pengujian akurasi aksi hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	100	59	41	59%	Menang
2	127	78	49	61.42%	Menang
3	127	78	49	61.42%	Menang

IV. IMPLEMENTASI DAN PENGUJIAN

4	127	78	49	61.42%	Menang
5	127	78	49	61.42%	Menang
6	127	78	49	61.42%	Menang
7	127	78	49	61.42%	Menang
8	127	78	49	61.42%	Menang
9	127	78	49	61.42%	Menang
10	127	78	49	61.42%	Menang

Tabel 4.34 Pengujian akurasi aksi hasil latih 50 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	51	27	24	52.94%	Kalah
2	59	28	31	47.46%	Kalah
3	34	21	13	61.76%	Menang
4	34	21	13	61.76%	Kalah
5	34	21	13	61.76%	Kalah
6	34	21	13	61.76%	Kalah
7	34	21	13	61.76%	Kalah
8	34	21	13	61.76%	Menang
9	34	21	13	61.76%	Kalah
10	34	21	13	61.76%	Kalah

IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.35 Pengujian akurasi aksi hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	53	37	16	69.81%	Kalah
2	83	54	29	65.06%	Kalah
3	53	37	16	69.81%	Kalah
4	83	54	29	65.06%	Kalah
5	53	37	16	69.81%	Kalah
6	83	54	29	65.06%	Kalah
7	53	37	16	69.81%	Kalah
8	83	54	29	65.06%	Kalah
9	53	37	16	69.81%	Kalah
10	83	54	29	65.06%	Kalah

Tabel 4.36 Pengujian akurasi aksi hasil latih 150 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	63	27	36	42.86%	Kalah
2	63	27	36	42.86%	Kalah
3	63	27	36	42.86%	Kalah
4	63	27	36	42.86%	Kalah
5	63	27	36	42.86%	Kalah
6	63	27	36	42.86%	Kalah
7	63	27	36	42.86%	Kalah
8	63	27	36	42.86%	Kalah

IV. IMPLEMENTASI DAN PENGUJIAN

9	63	27	36	42.86%	Kalah
10	63	27	36	42.86%	Kalah

Tabel 4.37 Pengujian akurasi aksi hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	174	25	149	14.368%	Menang
2	187	56	131	29.947%	Menang
3	162	33	129	20.37%	Menang
4	131	28	103	21.374%	Menang
5	174	25	149	14.368%	Menang
6	174	25	149	14.368%	Menang
7	174	25	149	14.368%	Menang
8	174	25	149	14.368%	Menang
9	174	25	149	14.368%	Menang
10	174	25	149	14.368%	Menang

Tabel 4.38 Pengujian akurasi aksi hasil latih 150 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	138	86	52	62.32%	Kalah
2	101	74	27	73.27%	Kalah
3	133	88	45	66.17%	Kalah
4	92	66	26	71.74%	Kalah
5	133	88	45	66.17%	Kalah

IV. IMPLEMENTASI DAN PENGUJIAN

6	92	66	26	71.74%	Kalah
7	133	88	45	66.17%	Kalah
8	92	66	26	71.74%	Kalah
9	133	88	45	66.17%	Kalah
10	92	66	26	71.74%	Kalah

Tabel 4.39 Pengujian akurasi aksi hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	77	47	30	61.04%	Kalah
2	68	44	24	64.7%	Menang
3	77	47	30	61.04%	Kalah
4	68	44	24	64.7%	Menang
5	77	47	30	61.04%	Kalah
6	68	44	24	64.7%	Menang
7	77	47	30	61.04%	Menang
8	68	44	24	64.7%	Menang
9	77	47	30	61.04%	Kalah
10	68	44	24	64.7%	Menang

IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.40 Pengujian akurasi aksi hasil latih 250 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	85	75	10	88.24%	Kalah
2	85	75	10	88.24%	Kalah
3	85	75	10	88.24%	Kalah
4	85	75	10	88.24%	Kalah
5	85	75	10	88.24%	Kalah
6	85	75	10	88.24%	Kalah
7	85	75	10	88.24%	Kalah
8	85	75	10	88.24%	Kalah
9	85	75	10	88.24%	Kalah
10	85	75	10	88.24%	Kalah

Tabel 4.41 Pengujian akurasi aksi hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	114	36	78	31.58%	Menang
2	114	36	78	31.58%	Menang
3	114	36	78	31.58%	Menang
4	114	36	78	31.58%	Menang
5	114	36	78	31.58%	Menang
6	114	36	78	31.58%	Menang
7	114	36	78	31.58%	Menang
8	114	36	78	31.58%	Menang

IV. IMPLEMENTASI DAN PENGUJIAN

9	114	36	78	31.58%	Menang
10	114	36	78	31.58%	Menang

Tabel 4.42 Pengujian akurasi aksi hasil latih 250 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	122	76	46	62.3%	Menang
2	276	11	265	3.9855%	Menang
3	123	76	47	61.97%	Menang
4	276	11	265	3.9855%	Menang
5	123	76	47	61.97%	Menang
6	276	11	265	3.9855%	Menang
7	123	76	47	61.97%	Menang
8	276	11	265	3.9855%	Menang
9	123	76	47	61.97%	Menang
10	276	11	265	3.9855%	Menang

Tabel 4.43 Pengujian akurasi aksi hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	23	17	6	73.91%	Menang
2	28	22	6	78.57%	Menang
3	32	28	4	87.5%	Menang
4	23	17	6	73.91%	Menang
5	32	28	4	87.5%	Menang

IV. IMPLEMENTASI DAN PENGUJIAN

6	45	34	11	75.56%	Menang
7	56	42	14	75%	Menang
8	27	22	5	81.48%	Menang
9	42	28	14	66.67%	Menang
10	26	18	8	69.23%	Menang

Tabel 4.44 Pengujian akurasi aksi hasil latih 350 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Zen)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	78	71	7	91.03%	Menang
2	73	66	7	90.41%	Menang
3	78	71	7	91.03%	Menang
4	73	66	7	90.41%	Menang
5	78	71	7	91.03%	Menang
6	73	66	7	90.41%	Menang
7	78	71	7	91.03%	Menang
8	73	66	7	90.41%	Menang
9	78	71	7	91.03%	Menang
10	73	66	7	90.41%	Menang

Tabel 4.45 Pengujian akurasi aksi hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	85	35	50	41.18%	Menang
2	182	5	177	2.747%	Menang

IV. IMPLEMENTASI DAN PENGUJIAN

3	182	5	177	2.747%	Menang
4	182	5	177	2.747%	Menang
5	182	5	177	2.747%	Menang
6	182	5	177	2.747%	Menang
7	182	5	177	2.747%	Menang
8	182	5	177	2.747%	Menang
9	182	5	177	2.747%	Menang
10	182	5	177	2.747%	Menang

Tabel 4.46 Pengujian akurasi aksi hasil latih 350 ronde AI dengan 1 unit fungsi nilai terhadap lawan latih (Karakter Garnet)

Pengujian	Jumlah Pukulan	Pukulan Kena (<i>Hit</i>)	Pukulan Meleset (<i>Missed</i>)	Akurasi	Hasil
1	216	11	205	5.093%	Menang
2	216	11	205	5.093%	Menang
3	216	11	205	5.093%	Menang
4	216	11	205	5.093%	Menang
5	216	11	205	5.093%	Menang
6	216	11	205	5.093%	Menang
7	216	11	205	5.093%	Menang
8	216	11	205	5.093%	Menang
9	216	11	205	5.093%	Menang
10	202	13	189	6.436%	Menang

Pada Tabel 4.31 dan Tabel 4.32 dapat dilihat bahwa akurasi tertinggi memang didapat oleh karakter *Zen* yang digunakan oleh AI dengan 1 unit fungsi, namun jumlah dari serangan yang dilakukan oleh AI 1 unit fungsi tersebut lebih rendah daripada AI dengan 3 unit fungsi. Hal ini jelas terlihat perbandingannya pada tabel sebelumnya yang menunjukkan bahwa AI

dengan 1 unit fungsi belum mampu dapat melawan AI *rule-based* lawan latihnya. Hal sama pun terlihat pada penggunaan karakter *Garnet*, AI dengan 3 unit lebih unggul dengan jumlah pukulan terhadap lawannya yang lebih banyak.

Namun secara keseluruhan ketika AI semakin melakukan pembelajaran, nilai akurasi dan jumlah serangan terhadap lawan terus berubah cenderung naik terutama pada karakter *Zen*, namun hal tersebut menjadi tidak relevan ketika AI dapat mempelajari untuk melakukan serangan yang memiliki tingkat serangan yang lebih tinggi dibandingkan dengan serangan lainnya. Hal ini dapat dibuktikan dengan kemampuan AI dengan 3 unit fungsi *reward* mampu mengatasi AI *rule-based* lawannya dengan lebih baik daripada AI dengan 1 unit fungsi *reward*.

4.3.3 AI vs Pemain Manusia

Skenario pengujian pada bagian ini dilakukan untuk melihat persentase kemenangan yang didapat dengan melawan pemain manusia. Ada 2 karakter yang diujikan juga, yaitu *ZEN*, dan *GARNET*. Pada Pengujian ini pemain manusia yang ditandingkan melawan AI adalah pemain pemula dan menengah. Pada pengujian AI yang sudah dilatih 50 dan 150 ronde, pemain manusia yang menjadi lawannya adalah pemain pemula. Pada pengujian AI yang sudah dilatih sebanyak 250 kali, digunakan pemain manusia yang sudah paham mekanisme dari permainan ini, dan pada pengujian AI yang sudah dilatih sebanyak 350 ronde, lawan manusia yang digunakan adalah pemain yang mahir dalam memainkan permainan ini. Berikut ini adalah tabel hasil pengujian dengan jumlah pelatihan sebanyak 50 ronde.

Tabel 4.47 Pengujian hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Zen)

Pengujian	AI HRA	Pemain manusia	Selisih Darah
1	Menang	Kalah	242
2	Menang	Kalah	35
3	Menang	Kalah	55
4	Kalah	Menang	22
5	Kalah	Menang	90
6	Menang	Kalah	206
7	Menang	Kalah	48
8	Kalah	Menang	17

IV. IMPLEMENTASI DAN PENGUJIAN

9	Menang	Kalah	55
10	Kalah	Menang	5
Rata - rata selisih darah			77.5
Persentase Kemenangan	60 %	40 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	40 %	60 %	

Tabel 4.48 Pengujian hasil latih 50 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Garnet)

Pengujian	AI HRA	Pemain manusia	Selisih Darah
1	Kalah	Menang	70
2	Kalah	Menang	5
3	Menang	Kalah	80
4	Kalah	Menang	40
5	Kalah	Menang	14
6	Menang	Kalah	130
7	Menang	Kalah	325
8	Kalah	Menang	110
9	Menang	Kalah	135
10	Kalah	Menang	33
Rata - rata selisih darah			94.2
Persentase Kemenangan	40 %	60 %	

Persentase Seri	0 %	0 %	
Persentase Kalah	60 %	40 %	

Dari hasil Tabel 4.27, AI dengan karakter *ZEN* mampu menang terhadap permainan manusia dengan kemenangan 6:4. Namun dengan menggunakan karakter *GARNET*, kemenangan yang didapat belum maksimal. Total persentase kemenangan yang didapat oleh AI dengan pelatihan sebanyak 50 ronde adalah 50%.

Tabel 4.49 Pengujian hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Zen)

Pengujian	AI HRA	Pemain manusia	Selisih Darah
1	Menang	Kalah	81
2	Menang	Kalah	45
3	Menang	Kalah	75
4	Kalah	Menang	6
5	Menang	Kalah	121
6	Menang	Kalah	74
7	Menang	Kalah	102
8	Kalah	Menang	34
9	Menang	Kalah	19
10	Kalah	Menang	89
Rata - rata selisih darah			64.6
Persentase Kemenangan	70 %	30 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	30 %	70 %	

IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.50 Pengujian hasil latih 150 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Garnet)

Pengujian	AI HRA	Pemain manusia	Selisih Darah
1	Kalah	Menang	112
2	Kalah	Menang	94
3	Kalah	Menang	85
4	Menang	Kalah	130
5	Menang	Kalah	74
6	Menang	Kalah	34
7	Menang	Kalah	225
8	Menang	Kalah	178
9	Menang	Kalah	68
10	Kalah	Menang	28
Rata - rata selisih darah			122.8
Persentase Kemenangan	60 %	40 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	40 %	60 %	

Pada Tabel 4.29 dapat dilihat bahwa performa AI terhadap lawan manusia cukup meningkat dengan persentase kemenangan mencapai 70%. Hal yang sama juga didapatkan pada Tabel 4.30 dengan kemenangan yang didapatkan oleh AI sebesar 60% ketika menggunakan karakter *GARNET*. Total keseluruhan persentase kemenangan yang didapat oleh AI yang dilatih sebanyak 150 ronde adalah 65%, sehingga dapat dilihat bahwa persentase kemenangan terhadap lawan manusia meningkat sebanyak 15% dari AI yang dilatih sebanyak 50 ronde.

IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.51 Pengujian hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Zen)

Pengujian	AI HRA	Pemain manusia	Selisih Darah
1	Menang	Kalah	73
2	Kalah	Menang	60
3	Menang	Kalah	194
4	Kalah	Menang	49
5	Menang	Kalah	196
6	Menang	Kalah	121
7	Menang	Kalah	170
8	Menang	Kalah	105
9	Menang	Kalah	2
10	Menang	Kalaah	234
Rata - rata selisih darah			120.4
Persentase Kemenangan	80 %	20 %	
Persentase Seri	0 %	0 %	
Persentase Kalah	20 %	80 %	

Tabel 4.52 Pengujian hasil latih 250 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Garnet)

Pengujian	AI HRA	Pemain manusia	Selisih Darah
1	Kalah	Menang	20
2	Menang	Kalah	5
3	Menang	Kalah	265

IV. IMPLEMENTASI DAN PENGUJIAN

4	Menang	Kalah	27
5	Menang	Kalah	95
6	Menang	Kalah	15
7	Kalah	Menang	10
8	Seri	Seri	0
9	Menang	Kalah	105
10	Menang	Kalah	20
Rata - rata selisih darah			56.2
Persentase Kemenangan	70 %	20 %	
Persentase Seri	10 %	10 %	
Persentase Kalah	20 %	70 %	

Pada pengujian AI dengan pelatihan sebanyak 250 ronde, kemenangan yang diraih oleh karakter *ZEN* meningkat sebesar 10%, sehingga persentase kemenangan AI dengan karakter *ZEN* adalah 80%. Hal yang sama juga didapatkan ketika pengujian dilakukan dengan karakter *GARNET* yang kini persentase kemenangannya mencapai 70%. Total persentase yang didapat dengan melawan pemain manusia naik sebesar 10%, dengan nilai total sebesar 75%.

Tabel 4.53 Pengujian hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Zen)

Pengujian	AI HRA	Pemain manusia	Selisih Darah
1	Menang	Kalah	234
2	Seri	Seri	0
3	Menang	Kalah	121
4	Menang	Kalah	71

IV. IMPLEMENTASI DAN PENGUJIAN

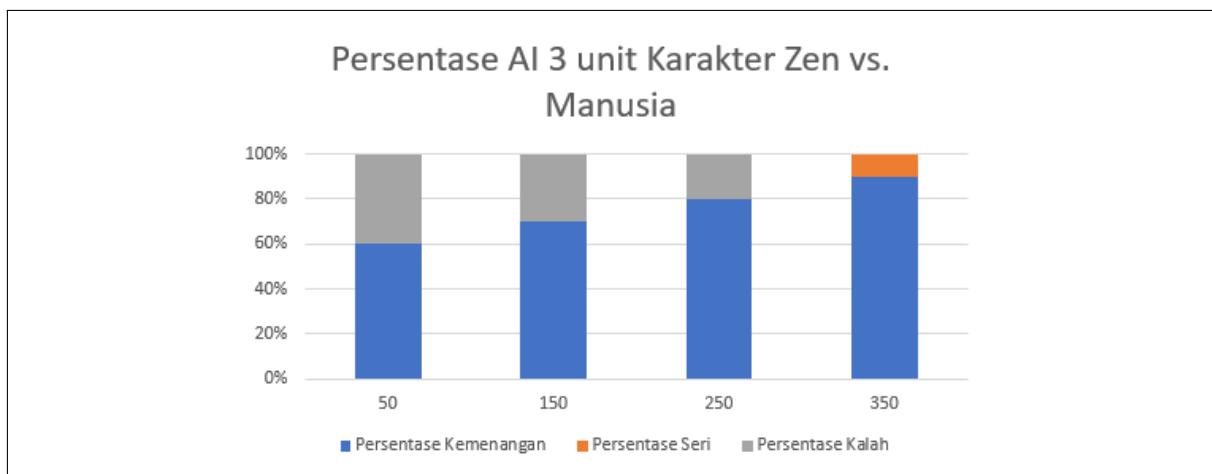
5	Menang	Kalah	12
6	Menang	Kalah	73
7	Menang	Kalah	60
8	Menang	Kalah	90
9	Menang	Kalah	47
10	Menang	Kalaah	220
Rata - rata selisih darah			92.8
Persentase Kemenangan	90 %	0 %	
Persentase Seri	10 %	10 %	
Persentase Kalah	0 %	90 %	

Tabel 4.54 Pengujian hasil latih 350 ronde AI dengan 3 unit fungsi nilai terhadap pemain manusia (Karakter Garnet)

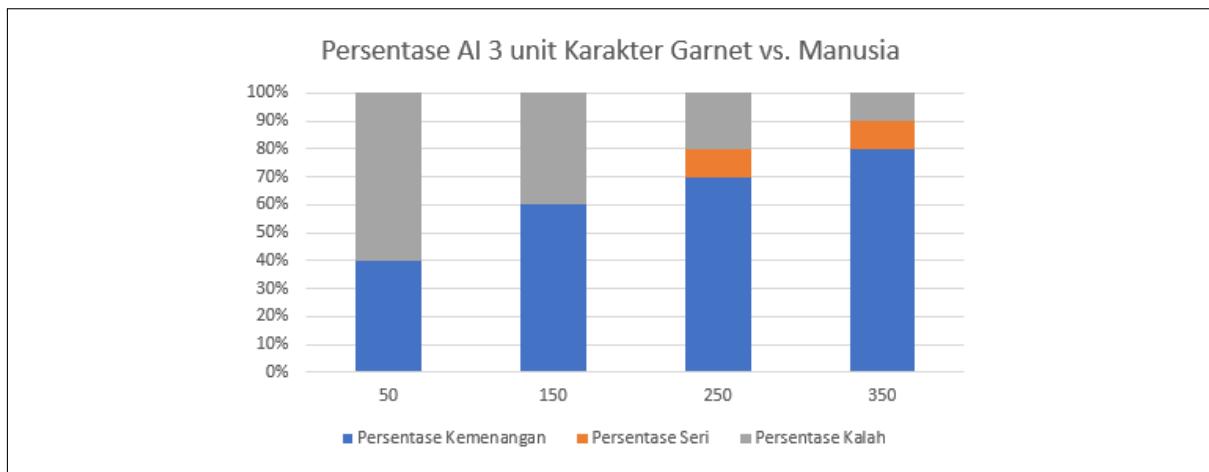
Pengujian	AI HRA	Pemain manusia	Selisih Darah
1	Kalah	Menang	128
2	Seri	Seri	0
3	Menang	Kalah	149
4	Menang	Kalah	65
5	Menang	Kalah	61
6	Menang	Kalah	113
7	Menang	Kalah	82
8	Menang	Kalah	67

9	Menang	Kalah	142
10	Menang	Kalah	52
Rata - rata selisih darah			85.9
Persentase Kemenangan	80 %	10 %	
Persentase Seri	10 %	10 %	
Persentase Kalah	10 %	80 %	

Pada Tabel 4.53 dapat dilihat bahwa AI memiliki persentase kemenangan sebesar 90% terhadap lawan manusia. Hal ini membuktikan bahwa adanya peningkatan sebanyak 5% dari hasil Tabel 4.51 dengan menggunakan karakter *ZEN*. Pengujian AI melawan manusia dengan menggunakan karakter *GARNET* juga mengalami kenaikan persentase sebesar 10%, sehingga kini kemenangannya dengan karakter ini sebesar 80%. Total persentase kemenangan yang diraih pada pengujian AI yang sudah dilatih sebanyak 350 ronde ini adalah 80%.



Gambar 4.5 Grafik persentase AI 3 unit karakter *Zen* terhadap pemain manusia



Gambar 4.6 Grafik persentase AI 3 unit karakter *Garnet* terhadap pemain manusia

Dari semua pengujian di atas, dapat disimpulkan bahwa AI yang memiliki 3 unit fungsi yang berbeda mampu mempelajari permainan lebih baik daripada AI dengan 1 unit fungsi. Jumlah pelatihan pun mempengaruhi seberapa baik AI dalam mempelajari permainan, semakin banyak jumlah pelatihan maka performanya pun ikut meningkat. AI dengan 3 unit fungsi yang sudah dilatih sebanyak 350 ronde dapat mengatasi permainan lawan *rule-based* AI dan juga pemain manusia.

BAB V

PENUTUP

Bab ini berisi tentang kesimpulan dan saran dari aplikasi pembuatan kecerdasan buatan pada permainan pertarungan yang telah dikembangkan.

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian serta pembuatan kecerdasan buatan pada permainan pertarungan adalah :

1. Pembagian nilai fungsi optimal berpengaruh terhadap kinerja pembelajaran AI, dengan pembagian 1 nilai fungsi optimal, dapat dilihat pada Tabel 4.25 AI masih tidak dapat bersaing dengan *rule-based* AI meskipun sudah dilatih sebanyak 250 ronde, sedangkan pada Tabel 4.23, AI dengan 3 pembagian dapat bersaing dengan AI lawannya. Hal ini dikarenakan 3 nilai fungsi tersebut dapat mempercepat proses pembelajaran agen dengan masing - masing nilai fungsi tersebut belajar secara individu sesuai dengan *reward*-nya masing - masing.
2. Jumlah aksi sia - sia pada pengujian Tabel 4.31 hingga Tabel 4.46 terlihat tidak terlalu berpengaruh terhadap kemenangan AI, sebab ada beberapa serangan yang cukup dilakukan sedikit namun memiliki dampak yang lebih besar ketimbang melakukan banyak serangan namun tidak memiliki dampak yang tinggi. Namun dapat dilihat bahwa setiap bertambahnya ronde pelatihan, jumlah serangan yang dilakukan dan akurasinya juga semakin meningkat meskipun tidak terlalu signifikan.
3. Jumlah ronde pelatihan AI mempengaruhi performa AI dalam mempelajari permainan lawan, AI 3 unit fungsi dengan pelatihan sebanyak 350 ronde dipilih karena performanya pada saat pengujian mampu mengalahkan *rule-based* AI dapat dilihat pada Gambar 4.1 dan 4.2 dengan total persentasi kemenangannya sebesar 100%.
4. Pada Gambar 4.5 dan 4.6 dapat dilihat persentase kemenangan AI 3 unit meningkat, dan hasil total persentasi yang didapat dengan melawan pemain manusia sebesar 80%, ini menunjukkan selain dapat bersaing melawan AI lain, AI juga mampu menghadapi lawan manusia.

5.2 Saran

Dari kesimpulan yang didapat, terdapat saran yang dapat dijadikan masukan dalam pengembangan aplikasi di masa yang akan datang:

1. Pemilihan metode optimisasi perubahan bobot yang lebih baik, yang mungkin dapat mengurangi waktu pelatihan seperti *Adaptive moment Estimation*, *Adagrad* atau lainnya.

V. PENUTUP

2. Penambahan unit nilai fungsi lainnya dengan penerapan *reward* yang tepat dapat meningkatkan performa AI pada proses pelatihan [9].

DAFTAR REFERENSI

- [1] M. Bedder, D. Kudenko, and S. Lucas, "Challenge for Implementing Monte Carlo Tree Search into Commercial Games", The EPSRC Centre for Doctoral Training in Intelligent Games and Game Intelligence, 2017.
- [2] D. Nguyen, "Supervised and Reinforcement Learning for Fighting Game AIs using Deep Convolutional Neural Network", Japan Advanced Institute of Science and Technology, 2017.
- [3] E. Adams, *Fundamentals of game design*, 2nd ed. California: New Riders, 2009.
- [4] K. Yamamoto, S. Mizuno, Chun Yin Chu and R. Thawonmas, "Deduction of fighting-game countermeasures using the k-nearest neighbor algorithm and a game simulator", *2014 IEEE Conference on Computational Intelligence and Games*, 2014. Available: 10.1109/cig.2014.6932915.
- [5] M. Ishihara, T. Miyazaki, C. Chu, T. Harada and R. Thawonmas, "Applying and Improving Monte-Carlo Tree Search in a Fighting Game AI", *Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology - ACE2016*, pp. 1-6, 2016.
- [6] L. Oses, "Reinforcement Learning in Videogames", Polytechnic University of Catalonia (UPC) Final Project, 2017.
- [7] I. Pinto and L. Coutinho, "Hierarchical Reinforcement Learning With Monte Carlo Tree Search in Computer Fighting Game", *IEEE Transactions on Games*, vol. 11, no. 3, pp. 290-295, 2018. Available: 10.1109/tg.2018.2846028.
- [8] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes and J. Tsang, "Hybrid Reward Architecture for Reinforcement Learning", in *Advances in Neural Information Processing Systems*, pp. 5392-5402, 2017.
- [9] Intelligent Computer Entertainment Lab., Ritsumeikan University, "FightingGameICE", *Official FightingGameICE homepage*, 2018. Available: <http://www.ice.ci.ritsumei.ac.jp/ftgaic/index.htm>.
- [10] G. Yannakakis and J. Togelius, *Artificial intelligence and games*, 1st ed. Springer, 2018.
- [11] R. Sutton and A. Barto, *Reinforcement learning: An Introduction*, 2nd ed. USA: The MIT Press, 2018.
- [12] G. Kevin, *An introduction to neural networks*. London: UCL Press, 1997.
- [13] V. Mnih et al., "Human-level control through deep reinforcement learning", *Nature*, vol. 518, no. 7540, pp. 529-533, 2015. Available: 10.1038/nature14236.

DAFTAR REFERENSI

- [14] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*. USA: The MIT Press, 2016.
- [15] D. Bourg and G. Seemann, *AI for game developers*. Germany: O'Reilly, 2004.
- [16] J. Bennett and W. Briggs, *Using & understanding mathematics*, 6th ed. USA: Pearson, 2016.

LAMPIRAN

LAMPIRAN A

DATA MASUKAN

Tabel A-1 Data Masukan untuk *Input Layer*

No	Nama Masukan	No	Nama Masukan
1	Self HP	22	Self Motion AIR_FB = 13
2	Self Energy	23	Self Motion AIR_GUARD = 14
3	Self X Coordinate	24	Self Motion AIR_GUARD_RECov = 15
4	Self Y Coordinate	25	Self Motion AIR_RECov = 16
5	Self X Speed	26	Self Motion AIR_UA = 17
6	Self abs X Speed	27	Self Motion AIR_UB = 18
7	Self Y Speed	28	Self Motion BACK_JUMP = 19
8	Self abs Y Speed	29	Self Motion BACK_STEP = 20
9	Self Hit Count	30	Self Motion CHANGE_DOWN = 21
10	Self Motion AIR = 1	31	Self Motion CROUCH = 22
11	Self Motion AIR_A = 2	32	Self Motion CROUCH_A = 23
12	Self Motion AIR_B = 3	33	Self Motion CROUCH_B = 24
13	Self Motion AIR_D_DB_BA = 4	34	Self Motion CROUCH_FA = 25
14	Self Motion AIR_D_DB_BB = 5	35	Self Motion CROUCH_FB = 26
15	Self Motion AIR_D_DF_FA = 6	36	Self Motion CROUCH_GUARD = 27
16	Self Motion AIR_D_DF_FB = 7	37	Self Motion CROUCH_GUARD_RECov = 28
17	Self Motion AIR_DA = 8	38	Self Motion CROUCH_RECov = 29
18	Self Motion AIR_DB = 9	39	Self Motion DASH = 30
19	Self Motion AIR_F_D_DFA = 10	40	Self Motion DOWN = 31
20	Self Motion AIR_F_D_DFB = 11	41	Self Motion FOR_JUMP = 32
21	Self Motion AIR_FA = 12	42	Self Motion FORWARD_WALK = 33

A. DATA MASUKAN

43	Self Motion JUMP = 34	68	Opponent X Coordinate
44	Self Motion LANDING = 35	69	Opponent Y Coordinate
45	Self Motion NEUTRAL = 36	70	Opponent X Speed
46	Self Motion RISE = 37	71	Opponent abs X Speed
47	Self Motion STAND = 38	72	Opponent Y Speed
48	Self Motion STAND_A = 39	73	Opponent abs Y Speed
49	Self Motion STAND_B = 40	74	Opponent Motion AIR = 1
50	Self Motion STAND_D_DB_BA = 41	75	Opponent Motion AIR_A = 2
51	Self Motion STAND_D_DB_BB = 42	76	Opponent Motion AIR_B = 3
52	Self Motion STAND_D_DF_FA = 43	77	Opponent Motion AIR_D_DB_BA = 4
53	Self Motion STAND_D_DF_FB = 44	78	Opponent Motion AIR_D_DB_BB = 5
54	Self Motion STAND_D_DF_FC = 45	79	Opponent Motion AIR_D_DF_FA = 6
55	Self Motion STAND_F_D_DFA = 46	80	Opponent Motion AIR_D_DF_FB = 7
56	Self Motion STAND_F_D_DFB = 47	81	Opponent Motion AIR_DA = 8
57	Self Motion STAND_FA = 48	82	Opponent Motion AIR_DB = 9
58	Self Motion STAND_FB = 49	83	Opponent Motion AIR_F_D_DFA = 10
59	Self Motion STAND_GUARD = 50	84	Opponent Motion AIR_F_D_DFB = 11
60	Self Motion STAND_GUARD_RECov = 51	85	Opponent Motion AIR_FA = 12
61	Self Motion STAND_RECov = 52	86	Opponent Motion AIR_FB = 13
62	Self Motion THROW_A = 53	87	Opponent Motion AIR_GUARD = 14
63	Self Motion THROW_B = 54	88	Opponent Motion AIR_GUARD_RECov = 15
64	Self Motion THROW_HIT = 55	89	Opponent Motion AIR_RECov = 16
65	Self Motion THROW_SUFFER = 56	90	Opponent Motion AIR_UA = 17
66	Opponent HP	91	Opponent Motion AIR_UB = 18
67	Opponent Energy	92	Opponent Motion BACK_JUMP = 19
93	Opponent Motion BACK_STEP = 20	112	Opponent Motion STAND_A = 39

A. DATA MASUKAN

94	Opponent Motion CHANGE_DOWN = 21	113	Opponent Motion STAND_B = 40
95	Opponent Motion CROUCH = 22	114	Opponent Motion STAND_D_DB_BA = 41
96	Opponent Motion CROUCH_A = 23	115	Opponent Motion STAND_D_DB_BB = 42
97	Opponent Motion CROUCH_B = 24	116	Opponent Motion STAND_D_DF_FA = 43
98	Opponent Motion CROUCH_FA = 25	117	Opponent Motion STAND_D_DF_FB = 44
99	Opponent Motion CROUCH_FB = 26	118	Opponent Motion STAND_D_DF_FC = 45
100	Opponent Motion CROUCH_GUARD = 27	119	Opponent Motion STAND_F_D_DFA = 46
101	Opponent Motion CROUCH_GUARD_RECov = 28	120	Opponent Motion STAND_F_D_DFB = 47
102	Opponent Motion CROUCH_RECov = 29	121	Opponent Motion STAND_FA = 48
103	Opponent Motion DASH = 30	122	Opponent Motion STAND_FB = 49
104	Opponent Motion DOWN = 31	123	Opponent Motion STAND_GUARD = 50
105	Opponent Motion FOR_JUMP = 32	124	Opponent Motion STAND_GUARD_RECov = 51
106	Opponent Motion FORWARD_WALK = 33	125	Opponent Motion STAND_RECov = 52
107	Opponent Motion JUMP = 34	126	Opponent Motion THROW_A = 53
108	Opponent Motion LANDING = 35	127	Opponent Motion THROW_B = 54
109	Opponent Motion NEUTRAL = 36	128	Opponent Motion THROW_HIT = 55
110	Opponent Motion RISE = 37	129	Opponent Motion THROW_SUFFER = 56
111	Opponent Motion STAND = 38	130	Opponent Remaining Frame
131	Self first Projectile Hit Damage	137	Opponent first Projectile Hit Damage

A. DATA MASUKAN

132	Self first Projectile Hit Area X	138	Opponent first Projectile Hit Area X
133	Self first Projectile Hit Area Y	139	Opponent first Projectile Hit Area Y
134	Self second Projectile Hit Damage	140	Opponent first Projectile Hit Damage
135	Self second Projectile Hit Area X	141	Opponent first Projectile Hit Area X
136	Self second Projectile Hit Area Y	142	Opponent first Projectile Hit Area Y