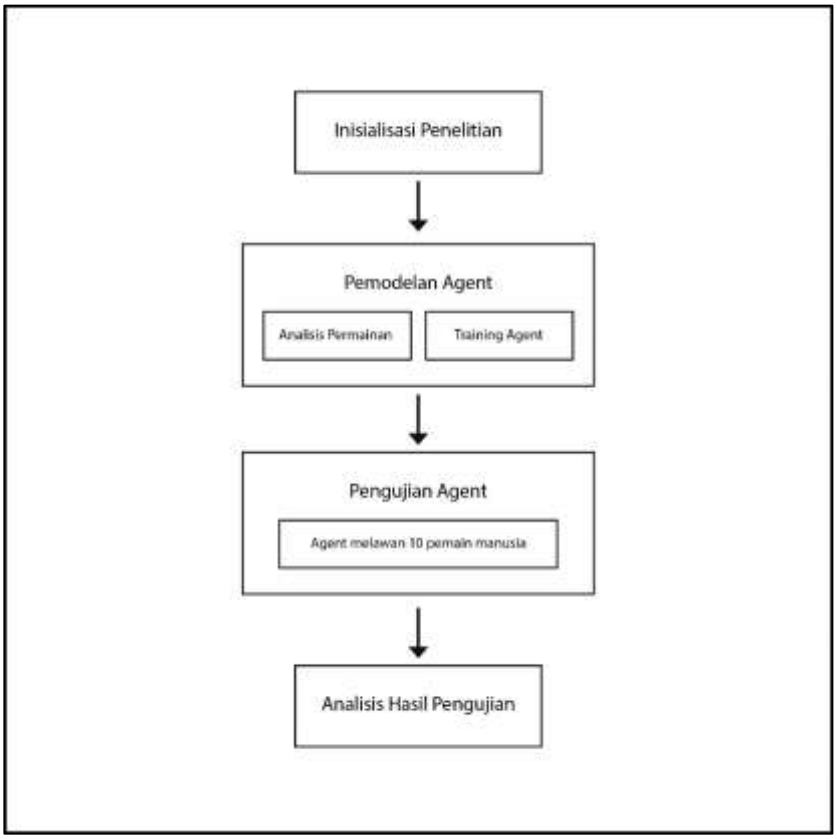
BAB III
ANALISIS DAN PERANCANGAN



Gambar 3.1. Tahapan dalam Penelitian.

Pada Gambar 3.1. merupakan tahap yang akan dilakukan dalam penelitian ini yakni inisialisasi penelitian, pemodelan *agent* menggunakan algoritma *Deep Q-Learning* dan *Actor-Critic*, pengujian *agent*, analisis hasil pengujian.

3.1. Inisialisasi Penelitian

Pada tahap inisialisasi penelitian dilakukan identifikasi masalah dan menentukan solusi yang akan digunakan. Dalam penelitian ini masalah yang dapat diidentifikasi dan dirumuskan adalah bahwa NPC pada *fighting game*

dinilai terlalu sederhana [3]. Untuk solusi masalah diatas maka dibuatlah sebuah *Agent* AI untuk membuat pengalaman bermain menjadi lebih menantang, untuk metode yang digunakan peneliti memilih menggunakan *Deep Q-Learning* dan *Actor-Critic*. Peneliti menggunakan metode didasarkan pada banyaknya penelitian-penelitian terdahulu yang menggunakan metode itu, contohnya adalah penelitian yang dilakukan oleh Inseok oh [5], Yu-Jhe Li [9], Vlad Firoiu [11], dan Matheus R. F. Mendonc [13], maka peneliti ingin mengetahui metode mana yang terbaik untuk *environment game* "Street Fighter II".

3.2. Pemodelan Agent

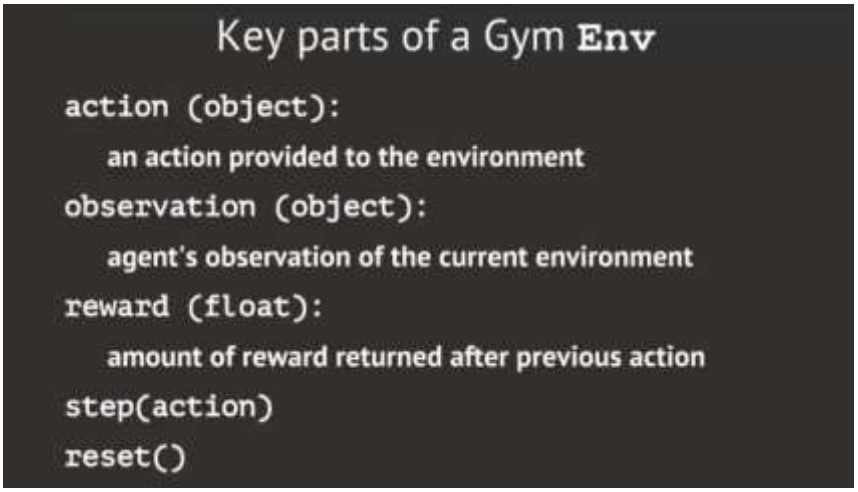
Pada tahap pemodel agent akan dilakukan dua tahapan yang pertama adalah analisis permainan, dimana peneliti akan menganalisis *environment* dari game "Street Fighter II" dan yang kedua adalah tahap training agent, dimana pada tahap ini agent akan dilatih menggunakan metode *Deep Q-Learning* dan *Actor-critic.*

3.2.1. Analisis Permainan

Pada tahap ini akan dilakukan analisis terhadap *environment* dari game "Street Fighter II", serta pendefinisian *state, action,* dan *reward* yang akan digunakan pada penelitian kali ini.

3.2.1.1. *Environment*

Pada penelitian kali *environment* yang akan digunakan adalah OpenAI Gym Retro. OpenAI Gym Retro adalah sebuah platform yang dapat digunakan untuk riset *reinforcement learning* pada *games*. Dalam platform ini terdapat 70 Atari games dan 30 Sega games menjadi lebih dari 1,000 game di berbagai emulator [16]. Dengan menggunakan OpenAI Gym Retro kita dapat mengubah game klasik seperti "Street Fighter II" menjadi sebuah *environment* yang bisa digunakan dalam algoritma *Deep* RL.



Gambar 3.2. Kunci Elemen dalam *Deep* RL [27].

Dalam *Deep* RL, *agent* akan memeriksa sebuah *state* dalam suatu *environtment* dan kemudian mengambil tindakan atau aksi dan menganalisis apakah aksi tersebut dapat memaksimalkan *reward* yang diperoleh.OpenAI Gym Retro menyediakan *environment* ini. OpenAI Retro Gym akan merubah *game* "Street Fighter II" menjadi *environment* yang dapat dianalisis oleh

agent yang akan dikembangkan. Kunci elemen secara abstraksi dapat digambarkan dalam Gambar 3.2.

3.2.1.2. *State, Action, Reward*

Pada penelitian kali ini peneliti mengaplikasikan CNN untuk mengelola citra dari setiap frame pada *game*. CNN akan mengekstraksi informasi dalam citra pada tiap *frame* dari *game.* Hasil dari ekstraksi tersebut adalah berupa informasi-informasi yang berguna seperti posisi pemain lawan dan posisi *agent*, jumlah *Health Point* (HP) lawan dan *agent* jarak antara *agent* dan lawan dan informasi lainnya yang ada pada *state* atau kondisi pada saat itu yang berguna untuk *agent* menentukan aksi apa yang akan ia lakukan. *Action* yang mungkin terjadi dalam *game* "*Street Fighter* II" adalah 144 aksi, dimana 144 ini terdiri dari 8 pergerakan *basic* seperti maju, mundur, merangkak, dll. Lalu 6 gerakan pukulan dan tendangan, serta 3 gerakan spesial. Besaran *reward* yang dapat diperoleh *agent* dapat dilihat pada Tabel 3.1 berikut :

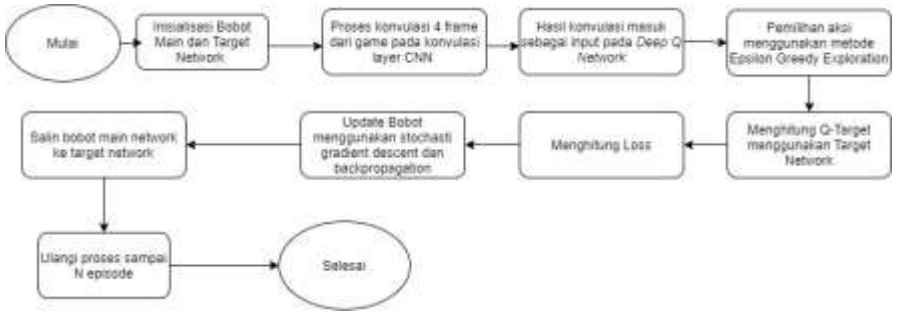
Tabel 3.1 Besaran *Reward*

Aksi	*Reward and Punishment*
Lawan memenangkan pertandingan	-10.0
HP berkurang	-0.5
Memenangkan pertandingan	10.0
Menghasilkan score	0.05

3.2.2. *Training Agent*

Pada tahap ini akan dilakukan *training* pada *agent* dengan menggunakan dua metode yaitu *Deep Q-Learning* dan *Actor-critic*.

3.2.2.1. *Training* Menggunakan *Deep Q-Learning*



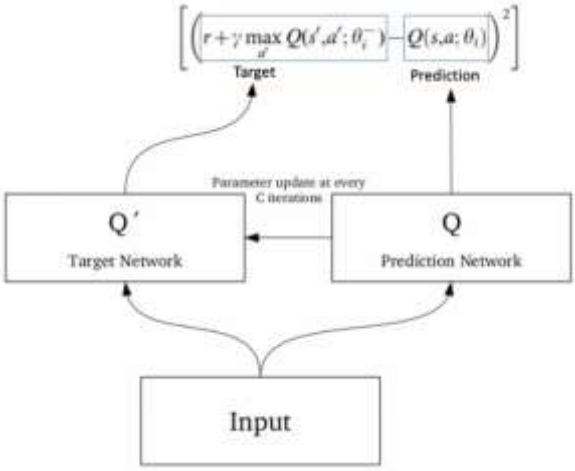
Gambar 3.3. Proses *Training Deep Q Learning*

Pada dasarnya jika kita lihat Gambar 3.3 kita bisa membagi tahap dalam proses *training* menjadi 3 tahap pada pengimplementasian algoritma *Deep Q-Learning* yang akan diimplementasikan pada penelitian kali ini. Tahap-tahap yang akan dilakukan adalah :

1. Inisialisasi *Main* dan Target *Neural Network*

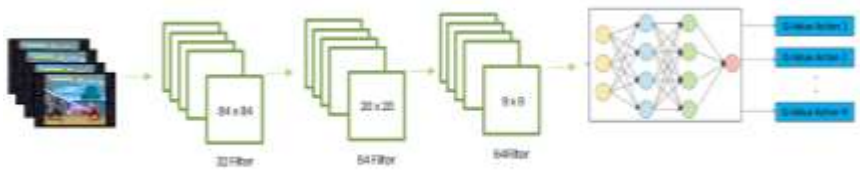
   Perbedaan paling mendasar pada *Deep Q-Learning* dan *Q-Learning* biasa adalah dalam pengimplementasian *Q-Table.* Pada *Deep Q-Learning* penggunaan *Q-Table* akan dengan *neural network.* Dariada melakukan *mapping state-action* dan dipasangkan dengan *q-value,* sebuah *neural network* akan memetakan *state* ke pasangan *action* dan *q-value*. Salah satu hal yang menarik pada *Deep Q-Learning* adalah pada proses pembelajaran

metode ini akan menggunakan 2 *neural network,* yaitu *main network* dan *target network*. Jaringan ini akan memiliki arsitektur yang sama akan tetapi memiliki bobot yang berbeda. Pada setiap C iterasi bobot dari *main network* akan disalin menuju *target network.* Penggunaan 2 *neural network* ini sangan membantu dalam membuat proses pembelajaran lebih stabil dan membuat algoritma berjalan lebih efisien. Penggambaran proses persalinan bobot dari *main network* ke *target network* dapat dilihat pada Gambar 3.3. Pada tahap pertama ini akan dilakukan penginisialisasian *main network* dan *target network*.



Gambar 3.4. Proses salin *weight*.

2. Memilih *Action* yang tepat menggunakan strategi *Epsilon-Greedy Exploration*



Gambar 3.5. Proses *Deep Q-Learning* [28].

Pada Gambar 3.5 bisa dilihat kalau pada *Deep Q-Learning* input pada *neural network* adalah frame-frame gambar dari "Street Fighter II" yang berasal dari *environment* OpenAI Gym Retro yang telah diproses oleh konvulosional *layer* pada CNN dan kemudian masuk sebagai input pada *neural network*. Lalu dari *state* tersebut *neural network* akan menghasilkan sekumpulan *Q-Value* dari tiap aksi yang kemungkinan dapat dilakukan pada *state* tersebut. Lalu *agent* akan memilih *action* mana yang akan ia lakukan untuk *state* tersebut. Pada dasarnya terdapat dua metode dalam memilih *action* mana yang akan dipilih oleh *agent*. Metode yang pertama dengan pendekatan Eksplorasi, dimana membiarkan *agent* untuk melakukan eksplorasi terhadap tiap-tiap *action* dan diharapkan *agent* dapat menemukan informasi lebih detail tentang *action* yang akan ia ambil. Pendekatan ini memungkinkan *agent* untuk mengambil keputusan yang lebih tepat di masa depan. Pendekatan yang kedua adalah ekploitasi, dimana

agent akan memilih secara *greedy*, dimana ia akan memilih *action* dengan *Q-Value* paling besar. Metode yang ketiga adalah menggunakan *Epsilon-Greedy Exploration*, dimana metode ini akan menyeimbangkan antara ekplorasi dan eksploitasi.

$$\text{Action pada waktu (t)} = \begin{cases} \max Q_t\,(ekploitasi) & : probabilitas\ 1-\in \\ random\ action\ (eksplorasi) & : probabilitas\ \in \end{cases}$$

Metode ini menggunakan nilai epsilon, dimana pada awal training nilai epsilon ($\in$) akan diset menjadi 1 dan membuat *agent* akan cenderung memilih pendekatan ekplorasi pada awal-awal *training*. Ini berguna agar *agent* dapat mempelajari *environment* yang ada. Lalu seiring berjalannya waktu nilai epsilon ($\in$) akan diset turun, sehingga ketika *agent* telah lebih paham akan *environment* maka pemilihan action akan diubah ke pendekatan eksploitasi.

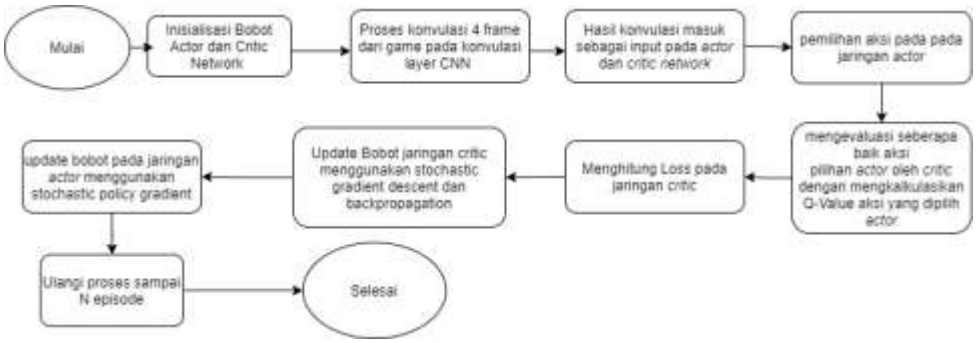
3. Update bobot *network* menggunakan persamaan Bellman

   Setelah melakukan pemilihan *action*, maka waktunya *agent* untuk melakukan *action* tersebut dan mendapatkan *reward* dan berpindah ke *state* selanjutnya. Lalu melakukan pembobotan ulang pada *main network* menggunakan sisi kanan persamaan bellman untuk menghitung *loss*.

$$Loss = \left(R_t + \gamma\, max\left(Q(s', a')\right) - Q(s, a)\right)^2 \qquad (3.2.1)$$

Output yang didapatkan dari *network* (Q(s,a)) akan dikurangkan dengan target *Q-Value* yang didapatkan dari sisi kanan persamaan bellman. Dari pengurangan ini maka akan didapatkan *loss* dan kemudian menggunakan *loss* ini bobot dari *network* akan diupdate menggunakan *back propagation*. Lalu setiap K waktu menyalin bobot dari *main network* ke *target network*.

3.2.2.2.*Training* Menggunakan *Actor-Critic*



Gambar 3.6. Proses *training Actor-Critic*.

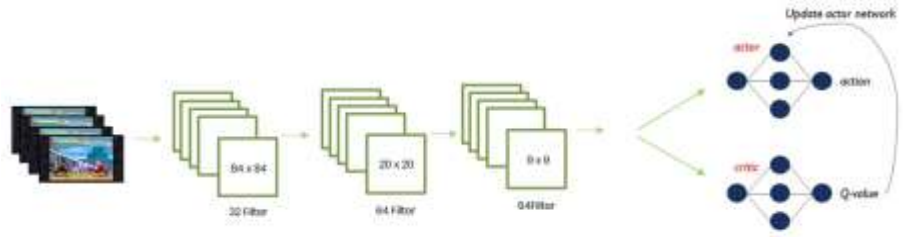
Pada dasarnya jika kita lihat Gambar 3.6. kita bisa membagi tahap dalam proses *training* menjadi 3 tahap, tahapan-tahapan implementasi yang akan dilakukan adalah :

1. Inisialisasi *Actor* dan *Critic Network*

   Pada langkah pertama adalah inisialisasi untuk bobot jaringan *actor* dan *critic*. Pada dasarnya kedua jaringan ini memiliki arsitektur yang sama hanya saja memiliki kegunaan yang berbeda. Jaringan *actor* akan digunakan menentukan *action* apa yang akan dilakukan oleh *agent*,

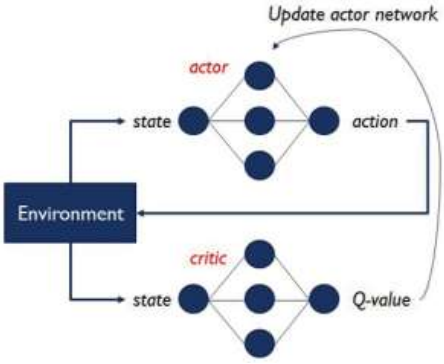
setelah itu *action* akan dievaluasi oleh jaringan *critic* dengan menghitung sebaik apa *value* dari *action* tersebut.

2. Pemilihan *action* dan evaluasi *action*



Gambar 3.7. Proses *Actor-Critic* [29].

Setelah menginisialisasi bobot dari jaringan *actor* dan *critic*. Pada Gambar 3.7 bisa dilihat kalau pada *actor-critic* input pada *neural network* adalah frame-frame gambar dari "Street Fighter II" yang berasal dari *environment* OpenAI Gym Retro yang telah diproses oleh konvulosional *layer* pada CNN dan kemudian masuk sebagai input pada kedua jaringan tersebut. Jika digambarkan prosesnya maka akan seperti pada Gambar 3.8.



Gambar 3.8. Arsitektur *Actor* dan *Critic*.

Pada setiap satuan waktu t, *state* saat itu (S_t) akan masuk sebagai input dari kedua jaringan tersebut. Jaringan *actor* akan memilih *action* yang pas untuk dieksekusi pada S_t (A_t), dan akan menerima *state* selanjutnya (S_{t+1}) dan *reward* untuk *action* tersebut (R_t). Lalu setelah itu jaringan pada *critic* akan mengevaluasi *action* yang diambil jaringan *actor* dengan mengkomputasikan *value* dari *action* tersebut ($V(S_t)$). Lalu setelah itu dilakukan perhitungan V_{target} menggunakan estimasi TD sebagai berikut :

$$V_{target} = R_t + \gamma \mathrm{Vt}(S_{t+1}) \qquad (3.2.2)$$

Dimana R_t adalah *reward* yang didapat pada *state* S_t dan γ adalah *discount rate* yang digunakan dan $\mathrm{Vt}(S_{t+1})$ adalah *value* maksimum dari *state* S_{t+1} dimana cara mengkalkulasikannya adalah memasukkan *state* S_{t+1} sebagai input pada jaringan *critic* dan mengambil nilai *value* hasil yang paling besar.

3. Perhituingan *loss* dan update bobot

Lalu setelah melakukan perhitungan V_{target} dilakukan perhitungan *loss* untuk jaringan *critic* dengan menggunakan formulasi seperti ini :

$$Loss = (V_{target} - \mathrm{V}(\mathrm{St}))^2 \qquad (3.2.3)$$

Setelah dilakukan perhitungan *loss* untuk jaringan *critic*, maka dilakukan update bobot untuk jaringan *critic* menggunakan *stochastic gradient descent* dan *back propagation*. Setelah itu akan dilakukan

pengupdatean bobot untuk jaringan *actor* menggunakan *stochastic policy gradient* yang mempunyai formulasi formulasi seperti ini :

$$\Delta\theta = \beta\ [\nabla\theta\ (log\ \pi w(s,a))Q\pi w\ (s,a)] \qquad (3.2.4)$$

Dimana $\Delta\theta$ adalah perubahan bobot pada jaringan actor, $log\ \pi w(s,a)$ adalah nilai yang didapatkan dari jaringan *actor* pada *state* s untuk aksi a, dan $Q\pi w\ (s,a)$ adalah Q-Value yang didapat dari jaringan *critic* dan β adalah *learning rate* dari jaringan *actor* .

Ketiga proses diatas akan dilakukan terus menerus sebanyak N episode.

3.3. Pengujian *Agent*

Pengujian *agent* untuk dikatakan manakah *agent* yang terbaik dan mana *agent* yang lebih menantang adalah dengan melawankan *agent* dengan 5 pemain manusia yang sudah pernah memainkan permainan bergenre *fighting game*. Akan ada 3 *agent* yang akan diukur diuji yaitu *agent* default dari *game* "Street Fighter II", *agent* yang dilatih menggunakan *Deep Q Learning*, dan *agent* yang dilatih menggunakan *Actor-Critic*. Komponen yang akan dijadikan perbandingan adalah *winrate* dan tingkat menantang. Rumus untuk menghitung *winrate* yang digunakan adalah :

$$Winrate\ = \frac{\text{Jumlah } agent \text{ memenangkan pertandingan}}{\text{Jumlah bermain}} * 100\% \qquad (3.2.5)$$

Lalu untuk pengukuran tingkat menantang, peneliti akan menyakan kepada pemain manusia, diantara 3 *agent* tersebut mana yang lebih menantang atau lebih susah untuk dilawan.