BAB 3
ANALISIS DAN PERANCANGAN

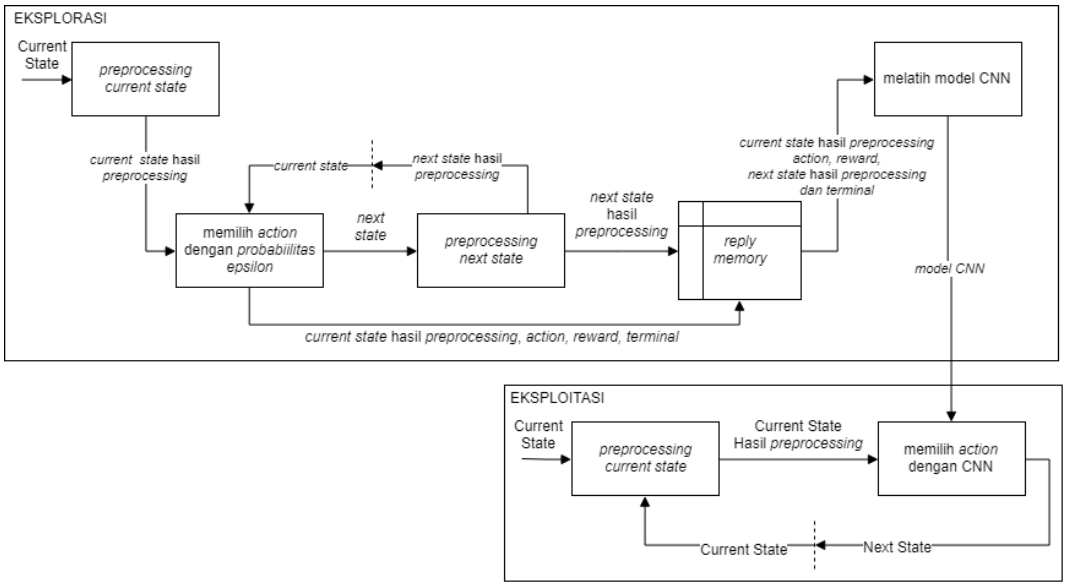
3.1 Analisis Masalah

Flappy bird memiliki komponen-komponen yang dinamis mulai dari posisi burung, pipa dan celah antar pipa yang beragam, sehingga memiliki *state space* yang besar dan Perubahan beberapa *pixel* saja merupakan *state* yang berbeda. Berdasarkan penelitian Moritz [1] penggunaan *Q-Learning* saja memiliki kekurangan yaitu untuk mengetahui *Q-value* suatu *action* pada *state* tertentu *agent* harus mencobanya terlebih dahulu sehingga butuh sangat banyak percobaan untuk mendapatkan kemampuan yang baik pada *agent* dalam memainkan permainan. Maka menambahkan *action value function approximation* untuk memprediksi *Q-value* pada *Q-Lerning* dan diharapkan dapat mengeneralisasi *state* sehingga *agent* bisa memperkecil jumlah percobaan dengan tetap mendapatkan hasil yang baik [3]. Pada penelitian Mnih dan kawan-kawan [4] menjelaskan penggunaan CNN sebagai *action value function approximation* pada *Q-Learning* dengan *input* citra dari keadaan permainan dan mendapatkan hasil yang baik dimana tiga dari tujuh permainan Atarai 2600 berhasil melampaui kemampuan manusia yaitu *breakout* dengan skor 168, *enduro* dengan skor 470 dan *pong* dengan skor 20. Pada penelitian yang dilakukan Ajay Rao dan kawan-kawan [5] yang mengimplementasikan *Q-Learning* dengan CNN sebagai *action value function approximation* pada delapan permainan berbeda, dijelaskan bahwa penggunaan CNN sebagai *action value function approximation* dapat mengeneralisir *state space* yang dinamis dan cocok diterapkan pada kasus *autonomus systems*.

Maka dari itu pada penelitian ini akan mengimplementasikan CNN sebagai *action value function approximation* pada *Q-Learning* untuk memainkan permainan *flappy bird* dan akan dilihat pengaruhnya jumlah percobaan pada proses pembelajaran dan skor yang didapat *agent* dalam memainkan permainan.

3.2 Analisis Proses

Analisis proses adalah tahapan untuk menganalisis metode yang akan digunakan pada sistem. Sistem yang akan dibangun terdapat dua tahap utama yang akan dilakukan, yaitu tahap eksplorasi dan eksploitasi. Satu kali iteras pada tahap ini sama dengan satu *frame* pada permainan. Adapun Gambaran tahapan yang akan dilakukan dapat dilihat pada Gambar 3.1 Tahapan Proses yang Akan Dilakukan.



Gambar 3.1 Tahapan Proses yang Akan Dilakukan

Tahap pertama adalah eksplorasi. Pada tahap ini *agent* akan mempelajari cara memainkan permainan *flappy bird*, tahap ini dimulai dari *agent* mendapat *current state* dari permainan berupa citra keadaan permaian, lalu dilakukan *preprocessing* terhadap citra tersebut. *Preprocessing* terdiri dari pemotongan citra, konversi ke *grayscale*, *resize* dan *thresholding*. Setelah *preprocessing* lalu *agent* akan memilih *action* dengan probabilitas epsilon, setelah *action* didapat dan dieksekusi di permainan *agent* akan mendapat *reward*, *next state* dan *terminal* sebagai umpan balik. Lalu, akan dilakukan *preprocessing* terhadap *next state* yang telah didapat, tahapan *preprocessing* sama dengan yang sebelumnya hanya saja *input*-nya sekarang adalah *next state*. Setelah itu, masukan *current state* hasil *preprocessing*, *action*, *reward*, *next state* dan *terminal* (disebut *experience*) ke dalam *reply memory* dan melatih model CNN dengan data latih (*experience*) yang diambil secara acak dari *reply memory* sebanyak 32 buah (*batch*). Terakhir, lakukan perubahan *next*

state hasil *preprocessing* menjadi *current state* dan proses diulangi dari tahap memilih *action* dengan probabilitas epsilon. Tahap eksplorasi ini memiliki kondisi henti yaitu ketika *agent* menyentuh skor 20 dan saat *agent* sudah memainkan permainan selama 1000000 *frame* [4]. Alasan pengambilan kondisi henti tersebut karena pada skor 20 dianggap sudah cukup besar dan *agent* sudah bisa melewati berbagai macam *state,* untuk nilai lainnya akan diujikan pada bagian pengujian.

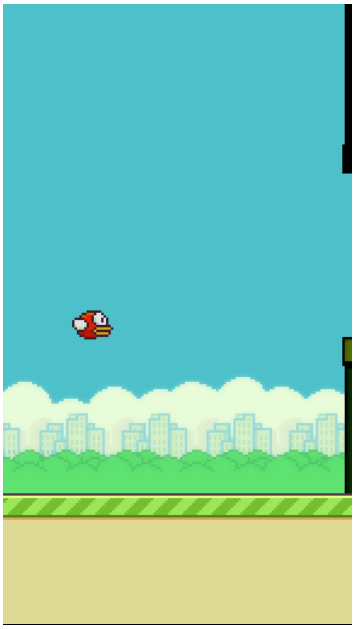
Tahap kedua adalah eksploitasi. Pada tahap ini *agent* akan menggunakan model CNN yang telah dilatih pada tahap eksploitasi. Tahap ini diawal dengan *agent* mendapat *current state* dari permainan, lalu *current state* tersebut dilakukan *preprocessing* yang tahapannya sama dengan yang sebelumnya. Selanjutnya *agent* akan memilih *action* menggunakan model CNN yang telah dilatih pada tahap eksplorasi dengan *input current state* yang telah di *preprocessing*. Output dari model CNN adalah nilai *Q-value* (regresi) dari masing-masing *action* yang bisa dilakukan *agent* (model CNN memiliki 2 output sesuai *action* yang bisa dilakukan, yaitu 0 untuk turun dan 1 untuk naik). Setelah *Q-value* dari setiap *action* didapatkan, lalu akan dipilih *action* dengan *Q-value* terbesar. Lalu *action* dieksekusi dan *agent* mendapat *next state* sebagai umpan balik dari permaian. Proses akan diulang dari tahap *preprocessing current state*.

3.3 Analisis Data Masukan

Adapun data masukan yang digunakan antara lain:

1) *State*

State yang akan diinputkan pada sistem adalah citra dari keadaan (*state*) permainan *flappy bird*. Ada dua jenis *state*, yaitu *current state* yang merupakan keadaan saat ini atau keadaan sebelum *action* dieksekusi dan *next state* adalah keadaan setelah *action* dieksekusi. Contoh *state* dapat dilihat pada Gambar 3.2 *State* Permainan.



Gambar 3.2 *State* Permainan

2) *Action*

Pada permainan *flappy bird* terdapat 2 *action* yang bisa dilakukan *agent* dan setiap *action* diberi nilai *integer*. Berikut adalah *action* yang dapat dilakukan:

1. Nilai 0 untuk turun
2. Nilai 1 untuk naik

3) *Reward*

Reward adalah nilai yang didapat *agent* untuk setiap *action* yang dilakukan pada *state* tertentu. *Reward* yang digunakan pada penelitian ini antara lain:

1. +1 ketika berhasil melewati celah pipa.
2. 0.1 ketika burung hidup.
3. -1 ketika burung mati.

Penetuan *reward* didasarkan pada tujuan dari permainan, pada permainan *flappy bird* pemain harus harus melewati pipa sebanyak banyaknya maka ketika berhasil melewati pipa harus diberi *reward* maksimal, dan ketika mati harus diberi reward *minimal*. Namun ada kondisi dimana burung tetap hidup namun tidak melewati pipa, situasi ini bukan situasi yang buruk namun tidak bisa disebut baik karena belum melakukan tujuan utama permainan sehingga, kondisi ini akan diberi *reward* dengan nilai positif yang kecil.

4) *Terminal*

Terminal adalah status permainan setelah *action* dilakukan, tipe data dari *terminal* adalah *boolean*. *Terminal* bernila *True* jika setelah *action* dilakukan permainan berakhir, dalam *flappy bird* permainan berakhir jika burung menabrak pipa atau tanah. *Terminal* bernilai *false* jika burung masih bertahan hidup setelah *action* dilakukan.

3.4 Analisis *Preprocessing*

Preprocessing adalah suatu proses untuk mengolah data masukan asli (disebut *state*) sebelum data tersebut diolah pada tahap selanjutnya. Masukan data asli adalah citra dari keadaan *state* pada permainan. Citra tersebut akan melalui proses *preprocessing* meliputi pemotongan, konversi citra RGB ke *grayscale* dan *thresholding.*

1) Pemotongan

Pemotongan citra adalah proses dimana membuang sebagian bagian dari citra dengan koordinat tertentu. Pada proses ini citra asal yang memiliki ukuran 288x 512 akan dilakukan pemotongan pada sumbu Y dimana akan dipertahankan *pixel* pada sumbu Y dari koordinat 0 sampai 400 dan sisanya yaitu koordinat 401 sampai 512 sedangkan sumbu X tidak akan dilakukan pemotongan, maka didapatkan:

Ytop = 0 (Batas atas citra yang dipertahankan)
Ybottom = 400 (Batas bawah citra yang dipertahankan)
Xleft = 0 (Batas kiri Citra yang dipertahankan)
Xright = 288 (Batas kanan Citra yang dipertahankan)

Sehingga didapatkan koordinat titik pojok kiri atas citra yang akan dipertahankan yaitu (*Xleft*, *Ytop*), koordinat titik pojok kanan bawah yang akan dipertahankan yaitu (*Xright*, *Ybottom*). Maka citra yang terdapat pada koordinat (0, 0) sampai koordinat (288, 400) akan dipertahankan dan citra dengan koordinat diluar jangkauan tersebut yaitu koordinat (0, 401) sampai koordinat (288, 512) akan dibuang. Ukuran citra setelah pemotongan adalah selisih dari koordinat paling ujung disetiap sumbu.

X’ = *Xright – Xleft*

= 288 – 0

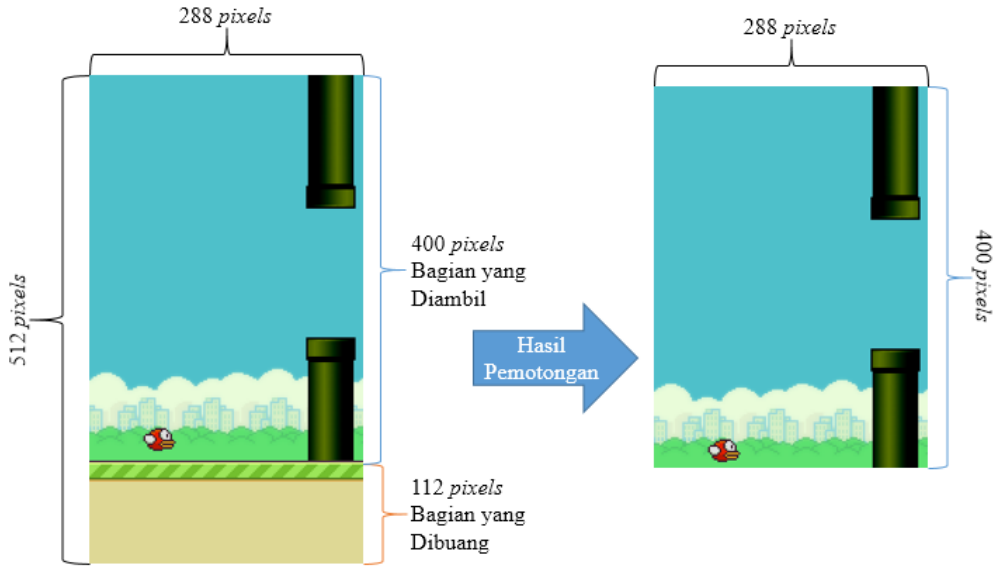
= 288

Y’ = *Ybottom – Ytop*

Y’ = 400-0

Y’ = 400

Adapun ilustrasi pemotongan dapat dilihat pada Gambar 3.3 Proses Pemotongan.



Gambar 3.3 Proses Pemotongan

Setelah melalui proses pemotongan, ukuran citra akan berubah dari asalnya 288 *pixels* x 512 *pixels* menjadi 288 *pixels* x 400 *pixels*.

2) Konversi Citra RGB ke *Grayscale*

Citra yang menjadi masukan awalnya memiliki tiga chanel warna yaitu *red*, *blue* dan *green*. Pada proses konversi citra RGB ke *grayscale* ini, chanel pada citra akan berubah menjadi satu dan ini dapat meringankan proses komputasi pada sistem. Adapun tabel sample nilai RGB dari citra dapat dilihat pada Tabel 3.1 Nilai Pixel RGB dari Citra.

Tabel 3.1 Nilai Pixel RGB dari Citra

x, y	0	1	2	3	.....	286	287
0	[202 192 78]	[202 192 78]	[202 192 78]	[202 192 78]	.....	[202 192 78]	[202 192 78]
1	[202 192 78]	[202 192 78]	[202 192 78]	[202 192 78]	.....	[202 192 78]	[202 192 78]
2	[202 192 78]	[202 192 78]	[202 192 78]	[202 192 78]	.....	[202 192 78]	[202 192 78]
3	[202 192 78]	[202 192 78]	[202 192 78]	[202 192 78]	.....	[202 192 78]	[202 192 78]
.....	.....	.....	.....	.....	.....	.....	.....
398	[112 226 94]	[112 226 94]	[112 226 94]	[112 226 94]	.....	[112 226 94]	[112 226 94]
399	[112 226 94]	[112 226 94]	[112 226 94]	[112 226 94]	.....	[112 226 94]	[112 226 94]

Perhitungan konversi citra RGB ke *grayscale* dapat menggunakan persamaan (2.1), berikut adalah persamaannya.

$$Gray(x,y) = 0.299 * R + 0.587 * G + 0.114 * B \quad \textbf{(2.1)}$$

Berikut adalah contoh perhitungan yang akan menghitung nilai *grayscale* untuk *pixel* ke (0, 0).

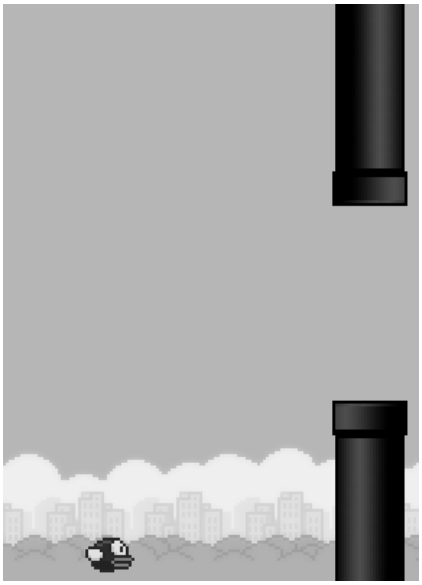
$$Pixel(0, 0) = 0.299 * 202 + 0.587 * 192 + 0.114 * 78$$
$$= 182$$

Lakukan konversi nilai RGB ke *Grayscale* ke seluruh pixel pada citra. Hasil perhitungan secara keseluruhan dapat dilihat pada Tabel 3.2 Nilai *Pixel Grayscale*.

Tabel 3.2 Nilai *Pixel Grayscale*

x, y	0	1	2	3	.....	286	287
0	182	182	182	182	.....	182	182
1	182	182	182	182	.....	182	182
2	182	182	182	182	.....	182	182
3	182	182	182	182	.....	182	182
.....	.....	.....	.....	.....	.....	.....	.....
398	177	177	177	177	.....	177	177
399	177	177	177	177	.....	177	177

Citra hasil konversi ke derajat keabuan dapat dilihat pada Gambar 3.4 Citra *Grayscale*.



Gambar 3.4 Citra *Grayscale*

3) *Resize*

Resize akan merubah ukuran citra sesuai dengan yang dikehendaki. *Resize* dapat memperkecil atau memperbsesar citra, pada penelitian ini citra akan di *resize* menjadi lebih kecil sehingga akan terjadi pengurangan beberapa *pixel*. Ukuran awal citra adalah 288 *pixels* x 512 *pixels*, ukuran tersebut akan di resize menjadi 32 *pixels* x 32 *pixels* [22]. Pada proses penurunan ukuran akan diambil nilai tengah dari suatu kernel untuk mewakili kernel tersebut [9]. Penentuan ukuran kernel diambil dari hasil bagi ukuran citra awal dengan citra hasil. Berikut perhitungan ukuran kernel untuk sumbu x.

kernelX = panjang sumbu X citra awal / panjang *resize* sumbu X
= 288 / 32
= 9

Berikut perhitungan ukuran kernel untuk sumbu y.

kernelY = panjang sumbu Y citra awal / panjang *resize* sumbu Y
= 400 / 32
= 12,5

Setelah hasil perhitungan didapat ukuran kernel sumbu X 9 dan sumbu Y 12,5. Pada nilai pixel citra hasil *resize pixel* ke 0,0 akan mencakup kernel pada pixel ke-0 sampai ke-8 sumbu X dan pixel ke-0 sampai ke-11,5 sumbu Y, maka akan dicari

pixel tengah yang akan menjadi perwakila kernel tersebut, berikut perhitungan penentuan koordinat nilai tengah pada kernel.

sumbuX = XAwalKernel + ((XAkhirKernel – XawalKernel) / 2)
= 0 + ((8 - 0) / 2)
= 0 + (4)
= 4

sumbuY = YAwalKernel + ((YAkhirKernel – YAwalKernel) / 2)
= 0 + ((11,5 – 0) / 2)
= 0 + 5,75
= 5,75

Karena nilai pixel pada pojok kiri atas dan kanan bawah sudah diketahui, tinggal mencari tahu nilai pixel pada pojok kiri dan kanan bawah dari bagian dasar kernel. Berikut perhitungan untuk mencari tahu nilai pojok kiri dan kanan bawah menggunakan interpolasi terhadap sumbu Y menggunaakan persamaan (2.3).

$$\boldsymbol{f(x,y) = \frac{y2-y}{y2-y1}.f(x,y1) + = \frac{y-y1}{y2-y1}.f(x,y2)} \quad \textbf{(2.3)}$$

$Pixel(0, 11.5) = \frac{12-11,5}{12-11}.pixel(0,11) + \frac{11,5-11}{12-11}.pixel(0,12)$
$= \frac{12-11,5}{12-11}.\ 182 + \frac{11,5-11}{12-11}.182$
= 182

$Pixel(8, 11.5) = \frac{12-11,5}{12-11}.pixel(8,11) + \frac{11,5-11}{12-11}.pixel(8,12)$
$= \frac{12-11,5}{12-11}.\ 182 + \frac{11,5-11}{12-11}.182$
= 182

Setelah nilai pojok kiri dan kana bawah diketahui, selanjutnya cari nilai tengah bagian bawah kernel menggunakan interpolasi terhadap sumbu X.

$Pixel(4, 11.5) = \frac{8-4}{8-0}.pixel(0,11.5) + \frac{4-0}{8-0}.pixel(8,11.5)$
$= \frac{8-4}{8-0}.\ 182 + \frac{4-0}{8-0}.182$
= 182

Setelah nilai tengah bagian atas dan dasar kernel diketahui, selanjutnya hitung nilai tengah dari kernel tersebut menggunakan interpolasi terhadap sumbu Y, Berikut perhitungannya.

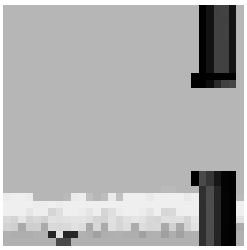
$$Pixel(4, 5.75) = \frac{11,5-5,75}{11.5-0} . pixel(4, 0) + \frac{5,75-0}{11.5-0} . pixel(4, 11.5)$$
$$= \frac{11,5-5,75}{11.5-0} . 182 + \frac{5,75-0}{11.5-0} . 182$$
$$= 182$$

Jadi, nilai pixel yang akan menjadi perwakilan dari kernel dari pixel(0, 0) sampai (8, 11.5) adalah nilai pixel(4, 5.75) yaitu 182 dan menjadi nilai pada citra hasil *resize* di pixel(0,0). Perhitungan tersebut dilakukan terhadap selirih kernel, untukk hasil keseluruhan proses *resize* dapat dilihat pada Tabel 3.3 Nilai Pixel Citra Hasil *Resize*.

Tabel 3.3 Nilai Pixel Citra Hasil *Resize*

x, y	0	1	2	3	.....	30	31
0	182	182	182	182	.....	18	182
1	182	182	182	182	.....	18	182
2	182	182	182	182	.....	18	182
3	182	182	182	182	.....	18	182
.....	.....	.....	.....	.....	.....	.....	.....
30	177	174	174	173	.....	0	177
31	177	177	177	177	.....	0	177

Citra hasil proses *resize* dapat dilihat pada Gambar 3.5 Citra Hasil Proses *Resize*.



Gambar 3.5 Citra Hasil Proses *Resize*

4) *Thresholding*

Thresholding adalah smetode yang membuat nilai citra hanya terdiri dari dua nilai. *Thresholding* memiliki ambang batas p penelitian ini akan menggunakan metode *thresh binary inverted* dengan nilai ambang batas 127, nilai maksimal 255 dan minimal 0, untuk persamaan dapat dilihat pada persamaan (2.2)

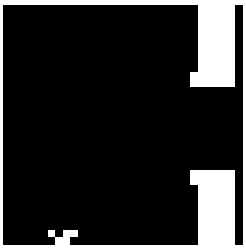
$$\boldsymbol{f(x,y)} = \begin{cases} \mathbf{0}, & \boldsymbol{img(x,y) > 127} \\ \mathbf{255}, & \boldsymbol{img(x,y) \leq 127} \end{cases} \quad \textbf{(2.2)}$$

Setelah dilakukan operasi pada persamaan diatas, didapat kan hasil *thresh binary inverted* seperti pada Tabel 3.4 Nilai *Pixel* Hasil *Tresh Binary Inverted*.

Tabel 3.4 Nilai *Pixel* Hasil *Tresh Binary Inverted*

x, y	0	1	2	3	.....	30	31
0	0	0	0	0	.....	255	0
1	0	0	0	0	.....	255	0
2	0	0	0	0	.....	255	0
3	0	0	0	0	.....	255	0
.....	.....	.....	.....	.....	.....	.....	.....
30	0	0	0	0	.....	255	0
31	0	0	0	0	.....	255	0

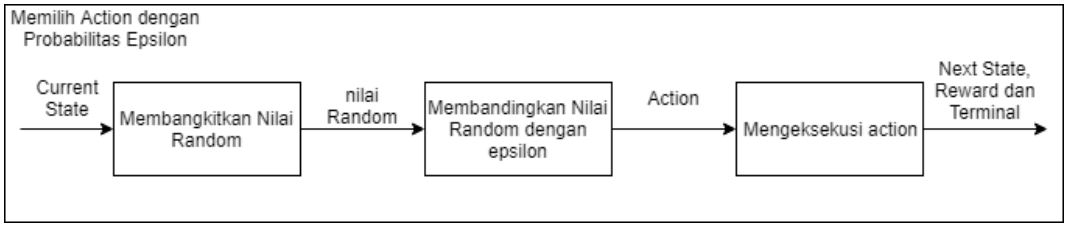
Citra hasil *thresh binary inverted* dapat dilihat pada Gambar 3.6 Citra Hasil *Tresh Binary Inverted*.



Gambar 3.6 Citra Hasil *Tresh Binary Inverted*

3.5 Analisis Pemilihan *Action* dengan Probabilitas Epsilon

Proses pemilihan action dengan probabilitas epsilon terdiri dari beberapa tahapan yang dapat dilihat pada Gambar 3.7 Tahapan Proses Memilih Action dengan Probabilitas Epsilon.



Gambar 3.7 Tahapan Proses Memilih Action dengan Probabilitas Epsilon

Pertama-tama insialisasi nilai awal epsilon yaitu 1.0, akhir epsilon 0.0001 dan nilai awal akan terus berkurang di setiap frame-nya sebesar 0.0000198. Selanjutnya bangkitkan nilai *random* antara 0 sampai 1, lalu bandingkan dengan epsilon jika

nilai *random* lebih kecil dari nilai epsilon, maka *agent* akan memilih *action* secara acak antara 0 atau 1. Jika nilai *random* lebih besar dari nilai epsilon, maka *agent* akan memilih action dengan nilai *Q-value* terbesar yang didapat dari *output feedforward* CNN dengan *input*-an *current state*. Setelah *action* terpilih, *agent* akan mengekseskusi *action* tersebut di permainan dan mendapatkan *feedback* berupa *next state reward* dan *terminal*. Rekaman yang berisi *current state, action, reward, next state* dan *terminal* dari proses ini disebut *experience*.

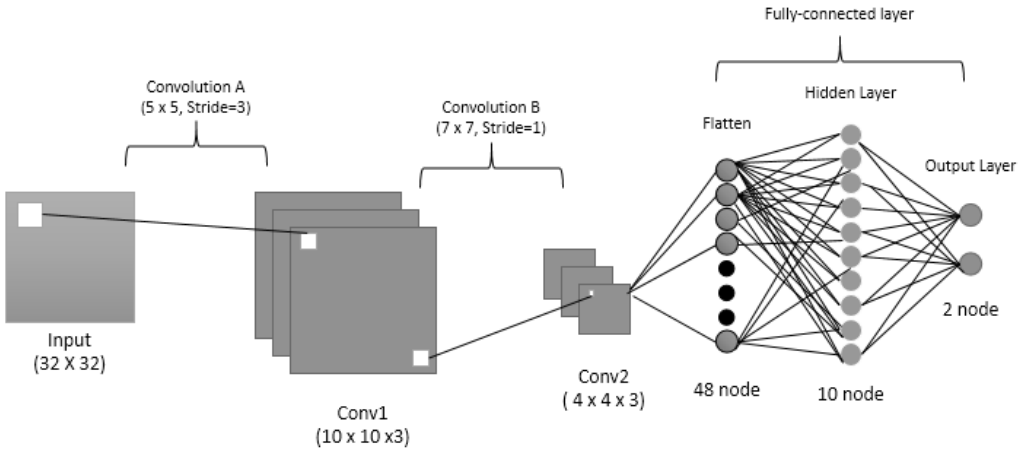
3.6 Analisis *Reply Memory*

Reply memory adalah sebuah wadah yang akan menyimpna data *experience*, satu *experience* terdiri dari *current state, reward, action, next state* dan *terminal* yang dikumpulkan disetiap *framenya* pada proses eksplorasi dengan maksimal jumlah data *experience* sebanyak 1000000. Data yang terdapat pada reply memory akan digunakan sebagai data latih pada proses pelatihan CNN, pengambilan data akan dilakukan secara acak berjumalah 32 buah. Penggunaan *reply memory* bertujuan untuk memutus korelasi antar satu data latih dengan yang lainnya yang dapat menyebabkan *agent* cenderung melakukan satu *action* saja. Selain itu, penggunaan *reply memory* dapat meningkatkan efesiensi *agent* dalam belajar dari pada belajar langsung dari data yang baru saja didapat tanpa menyimpannya terlebih dahulu di *reply memory* [4].

3.7 Analisis Metode *Q-Learning* yang Dikombinasikan dengan *Convolutional Neural Network* (CNN)

Pada metode pengkombinasian ini CNN akan digunakan sebagai *action value function approximation pada Q-Learning* yang akan memprediksi nilai *Q-value* untuk setiap *action* pada kondisi tertentu (*state* permainan menjadi *inputan*) dan *agent* akan menentukan *action* yang akan diambil dengan membandingkan *Q-value* setiap *action*, *action* dengan *Q-value* terbesar yang akan dipilih. Pada proses pelatihan CNN, *target* yang akan digunakan untuk menghitung nilai *loss* adalah

persamaan *value iteration update* yang terdapat pada metode *Q-learning*. Pengkombinasian *Q-Learning* dengan CNN dapat mengeneralisasi *state* sehinngga dapat mengurangi jumlah percobaan pada proses pembelajaran dengan tetap memberikan hasil yang baik pada *agent*. Adapun arsitektur dari CNN dapat dilihat pada Gambar 3.8 Arsitektur CNN.

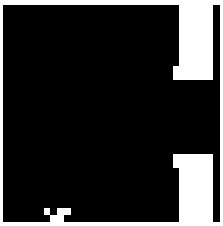


Gambar 3.8 Arsitektur CNN

Pada arsitektur CNN yang digunakan pada penelitian ini memiliki input berukuran 32 x 32, *convolutional layer* A dengan ukuran 5x5 memiliki 3 buah filter, 3 *stride* dengan fungsi aktivasi ReLU dan akan menghasilkan feature map 10 x 10 x 3, selanjutnya *convolutional layer* B dengan ukuran 7x7 memiliki 3 buah filter, 1 *stride* dengan fungsi aktivasi ReLU dan akan meghasilkan feature map 4 x 4 x 3 dan terakhir *Fully-connected layer* dengan input layer 48 node, *hidden layer* 10 node dengan fungsi aktivasi ReLu dan *output layer* dengan 2 buah node (sejumlah *action* yang bisa dilakukan *agent*) dan fungsi aktivasi *linear* [4] [10] [22]. Keluaran yang dihasilkan berisi perkiraan nilai *Q-value* untuk setiap *action* yang bisa dilakukan *agent* (nilai 0 untuk diam dan 1 untuk *flap*). Semua bobot pada setiap layer akan diisi nilai acak dengan rentang nilai -0.5 sampai 0.5 dan untuk nilai biasnya akan diisi 0 [23].

3.7.1. Analisis Pelatihan CNN

Pada proses pelatihan CNN data latih yang akan digunakan adalah sample acak yang diambil dari *reply memory.* Sample tersebut yang berisi *current state* setelah *preprocessing, action, reward, next state* setelah *preprocessing* dan *terminal*. Pengambilan data latih dilakukan secara acak sejumlah 32 buah di *reply memory.* Berikut adalah contoh sample acak dari *reply memory* yang akan digunakan pada proses pelatihan, untuk *current state* dapat dilihat pada Gambar 3.9 *Current State*, untuk *Next State* dapat dilihat pada Gambar 3.10 *Next State*, nilai *action*-nya adalah 0, nilai *reward*-nya adalah -1 dan *terminal*-nya *True.*



Gambar 3.9 *Current State*

Gambar 3.10 *Next State*

Selanjutnya CNN akan dilatih dengan data latih tersebut. Pada proses pelatihan CNN terdapat beberapa tahap antara lain *feedforward* dan *backpropagation.*

3.7.1.1. *Feedforward*

Pada tahap *feedforward* akan dilakukan proses CNN dari awal masukan melewati *convolutional layer* A, *convolutional layer* B dan *fully-connected layer* hingga dihasilkan 2 buah output yang berisi *Q-Value* untuk masing-masing *action.*

1. *Convolutional Layer* A

Pada *convolutional layer* akan dilakukan konvolusi anatara matriks citra masukan dengan matriks-matriks pada filter A. Filter akan digeser hingga keseluruh permukaan dan menghasilkan *feature map* FA. Pada *convolutional layer* A terdapat 3 filter berukuran 5x5 dengan *stride* 3. Citra yang menjadi inputan telah dijelaskan pada bagian analisis pelatihan CNN, yaitu Gambar 3.9 *Current State* dan nilai pixelnya dapat dilihat pada Tabel 3.5 Nilai Pixel Citra Masukan C (*Current Sate*).

Tabel 3.5 Nilai Pixel Citra Masukan C (*Current Sate*)

x, y	0	1	2	3	4	5	6	.....	30	31
0	0	0	0	0	0	0	0	.....	255	0
1	0	0	0	0	0	0	0	.....	255	0
2	0	0	0	0	0	0	0	.....	255	0
3	0	0	0	0	0	0	0	.....	255	0
4	0	0	0	0	0	0	0	.....	255	0
5	0	0	0	0	0	0	0	.....	255	0
6	0	0	0	0	0	0	0	.....	255	0
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
30	0	0	0	0	0	0	255	.....	255	255
31	0	0	0	0	0	0	0	.....	255	255

Bobot filter A1 dapat dilihat pada Tabel 3.6 Bobot *Convolutional Filter* A1 dan untuk bias nya diisi 0.

Tabel 3.6 Bobot *Convolutional Filter* A1

x, y	0	1	2	3	4
0	0,05	0,34	0,38	0,11	0,29
1	0,21	0,27	-0,45	0,24	-0,17
2	0,41	0,22	0,36	-0,18	-0,14
3	0,23	-0,24	-0,45	-0,07	-0,03
4	0,01	-0,14	0,46	0,35	0,22

Proses pada *convolutiona layer* A adalah operasi konvolusi antara citra masukan C dengan Tabel 3.6 Bobot *Convolutional Filter* A1, dimulai dari pojok kiri atas citra lalu bergeser hingga pojok kanan bawah, setiap hasil konvolusi ditambahkan dengan bias. Berikut ini adalah contoh operasi konvolusi menggunakan persamaan (2.5) citra C terhadap filter A1 yang menghasilkan pixel baris satu kolom 1 pada *feature map* FA1.

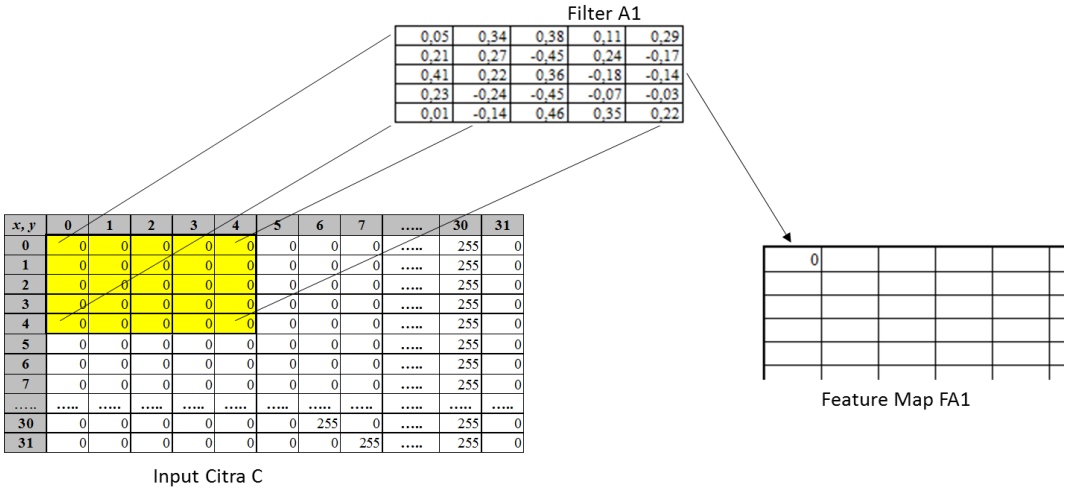
$$\boldsymbol{FA1_{i,j} = \sum_{m=0}^{4} \sum_{n=0}^{4} A1_{m,n}\, C_{i*3+m,j*3+n} + bA_0} \quad \textbf{(2.5)}$$

$$FA1_{0,0} = (A1_{0,0} * C_{0,0} + A1_{0,1} * C_{0,1} + A1_{0,2} * C_{0,2} + \cdots + A1_{4,4} * C_{4,4}) + bA_0$$

= (0,05*0 + 0,34*0 + 0,38*0 + ⋯ + 0,22*0) + 0

= 0

Proses tersebut terus diulang hingga seluruh permukaan citra terkena, untuk visualisasinya dapat dilihat pada Gambar 3.11 Visualisasi 1 Operasi pada *Convolution Layer* A1



Gambar 3.11 Visualisasi 1 Operasi pada *Convolution Layer* A1

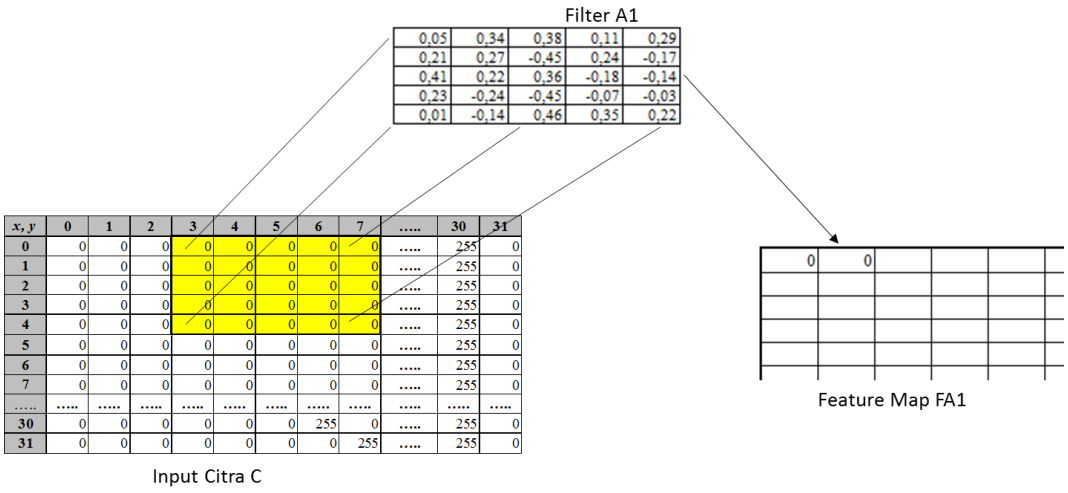
Setelah itu geser filter ke kanan sebanyak nilai *stride* yaitu 3 dan lakukan kembali operasi konvolusi.

$$FA1_{0,1} = (A1_{0,0}.C_{0,3} + A1_{0,1}.C_{0,4} + A1_{0,2}.C_{0,5} + \cdots + A1_{4,4}.C_{4,7}) + bA_0$$

=(0,05*0 + 0,34*0 + 0,38*0 + ⋯ + 0,22*0) + 0

= 0

Berikut adalah visualisasi operasi konvolusi filter A1 yang menghasilkan $FA[1]_{0,1}$ dapat dilihat pada Gambar 3.12 Visualisasi 2 Operasi *Convolution Layer* A1



Gambar 3.12 Visualisasi 2 Operasi *Convolution Layer* A1

Hasil proses keseluruhan pada filter A1 dapat dilihat pada tabel Tabel 3.7 *Feature Map* FA1.

Tabel 3.7 *Feature Map* FA1

x, y	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	234,6	538,05
1	0	0	0	0	0	0	0	0	234,6	538,05
2	0	0	0	0	0	0	0	48,45	137,7	538,05
3	0	0	0	0	0	0	0	30,6	257,55	293,25
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	56,1	226,95	173,4
7	0	0	0	0	0	0	0	-79,05	160,65	313,65
8	0	0	0	0	0	0	0	0	234,6	538,05
9	0	38,25	7,65	58,65	0	0	0	0	234,6	538,05

Langkah selanjutnya adalah jalankan fungsi aktivasi pada *feature map* FA1 , setiap pixel pada FA1 akan dimasukan ke dalam fungsi aktivasi ReLU yang akan mengubah setiap nilai negative menjadi 0, berikut perhitungannya menggunakan persamaan (2.7).

$$f(x) = max(0,x) \quad (2.7)$$

$RA1_{0,0} = max(0, FA1_{0,0})$

$= max(0, 0)$

$= 0$

Hasil dari fungsi aktivasi dapat dilihat pada Tabel 3.8 *Feature Map* ReLU RA1.

Tabel 3.8 *Feature Map* ReLU RA1

x, y	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	234,6	538,05
1	0	0	0	0	0	0	0	0	234,6	538,05
2	0	0	0	0	0	0	0	48,45	137,7	538,05
3	0	0	0	0	0	0	0	30,6	257,55	293,25
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	56,1	226,95	173,4
7	0	0	0	0	0	0	0	0	160,65	313,65
8	0	0	0	0	0	0	0	0	234,6	538,05
9	0	38,25	7,65	58,65	0	0	0	0	234,6	538,05

Selanjutnya proses konvolusi dilakukan terhadap citra C dengan bobot pada filter A2. Bobot filter A2 dapat dilihat pada Tabel 3.9 Bobot *Convolutional Filter* A2 dan biasnya diisi 0.

Tabel 3.9 Bobot *Convolutional Filter* A2

x, y	0	1	2	3	4
0	0,07	0,21	0,34	-0,31	-0,43
1	-0,35	0,14	-0,48	-0,27	-0,25
2	0,05	0,15	-0,11	0,05	0,16
3	-0,18	0,13	-0,45	-0,23	0,35
4	-0,27	0,29	0,39	0,22	0,15

Proses ini akan menghasilkan *feature map* FA2. Berikut hasil proses konvolusi dapat dilihat pada Tabel 3.10 *Feature Map* FA2.

Tabel 3.10 *Feature Map* FA2

x, y	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	-221,85	-155,55
1	0	0	0	0	0	0	0	0	-221,85	-155,55
2	0	0	0	0	0	0	0	127,5	-114,75	-155,55
3	0	0	0	0	0	0	0	-173,4	-267,75	-165,75
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	38,25	267,75	160,65
7	0	0	0	0	0	0	0	-22,95	-45,9	-234,6
8	0	0	0	0	0	0	0	0	-221,85	-155,55
9	0	-20,4	-45,9	-45,9	0	0	0	0	-221,85	-155,55

Setelah itu jalankan fungsi aktivasi ReLU pada *feature map* FA2. Hasil dari fungsi aktivasi ReLU dapat dilihat pada Tabel 3.11 *Feature Map* ReLU RA2.

Tabel 3.11 *Feature Map* ReLU RA2

x, y	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	127,5	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	38,25	267,75	160,65
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Terakhir, lakukan proses konvolusi terhadap citra C dengan bobot pada filter terakhir A3. Bobot filter A3 dapat dilihat Tabel 3.12 Bobot *Convolutional Filter* A3 dan biasnya diisi 0.

Tabel 3.12 Bobot *Convolutional Filter* A3

x, y	0	1	2	3	4
0	0,22	-0,22	0,24	-0,03	0,19
1	-0,26	-0,04	-0,41	0,25	0,36
2	-0,17	0,48	-0,31	0,3	-0,44
3	-0,46	-0,19	-0,21	0,46	0,22
4	0,36	0,39	0,34	0,02	0,47

Proses ini menghasilkan *feature map* FA3 yang dapat dilihat pada Tabel 3.13 *Feature Map* FA3.

Tabel 3.13 *Feature Map* FA3

x, y	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	369,75	193,8
1	0	0	0	0	0	0	0	0	369,75	193,8
2	0	0	0	0	0	0	0	175,95	420,75	193,8
3	0	0	0	0	0	0	0	140,25	86,7	-63,75
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	119,85	311,1	283,05
7	0	0	0	0	0	0	0	-20,4	379,95	140,25
8	0	0	0	0	0	0	0	0	369,75	193,8
9	0	237,15	132,6	-117,3	0	0	0	0	369,75	193,8

Setelah itu jalankan fungsi aktivasi ReLU pada *feature map* FA3. Hasil dari fungsi aktivasi dapat dilihat pada Tabel 3.14 *Feature Map* ReLU RA3.

Tabel 3.14 *Feature Map* ReLU RA3

x, y	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	369,75	193,8
1	0	0	0	0	0	0	0	0	369,75	193,8
2	0	0	0	0	0	0	0	175,95	420,75	193,8
3	0	0	0	0	0	0	0	140,25	86,7	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	119,85	311,1	283,05
7	0	0	0	0	0	0	0	0	379,95	140,25
8	0	0	0	0	0	0	0	0	369,75	193,8
9	0	237,15	132,6	0	0	0	0	0	369,75	193,8

Hasil dari *convolutional layer* A, yaitu *Feature Map* RA1, RA2 dan RA3 akan digunakan pada layer selanjutnya, yaitu *convolutional layer* B.

2. *Convolutional Layer* B

Pada *convolutional layer* B akan dilakukan proses yang sama seperti pada *convolutinal layer* A hanya saja *inputannya* rangkap tiga, yaitu *feature map* RA1, RA2 dan RA3. Pada *convolutional layer* B terdapat 3 filter berukuran 7x7 dengan *stride* 1. Pertama-tama akan dilakukan proses konvolusi antara *feature map* RA1, RA2 dan RA3 dengan bobot pada filter B1. Bobot pada filter B1 dapat dilihat pada Tabel 3.15 Bobot *Convolutional Filter* B1 dan biasnya diisi 0.

Tabel 3.15 Bobot *Convolutional Filter* B1

x, y	0	1	2	3	4	5	6
0	0,31	-0,14	0,44	0,25	0,44	-0,36	0,05
1	-0,47	0,09	0,28	0,05	0,4	-0,06	0,09
2	-0,29	0,4	0,25	0,36	0,33	-0,2	0,36
3	-0,1	0,34	-0,1	-0,49	-0,37	-0,18	-0,19
4	-0,01	-0,01	0,21	-0,26	0,43	-0,45	-0,26
5	0,19	0,34	0,1	0,31	0,33	0,17	0,45
6	-0,43	0,06	-0,39	0,03	0,21	0,22	-0,02

Proses konvolusi akan dimulai dari pojok kiri atas lalu bergeser hingga pojok kanan bawah, setiap hasil konvolusi ditambahkan dengan bias. Berikut perhitungannya.

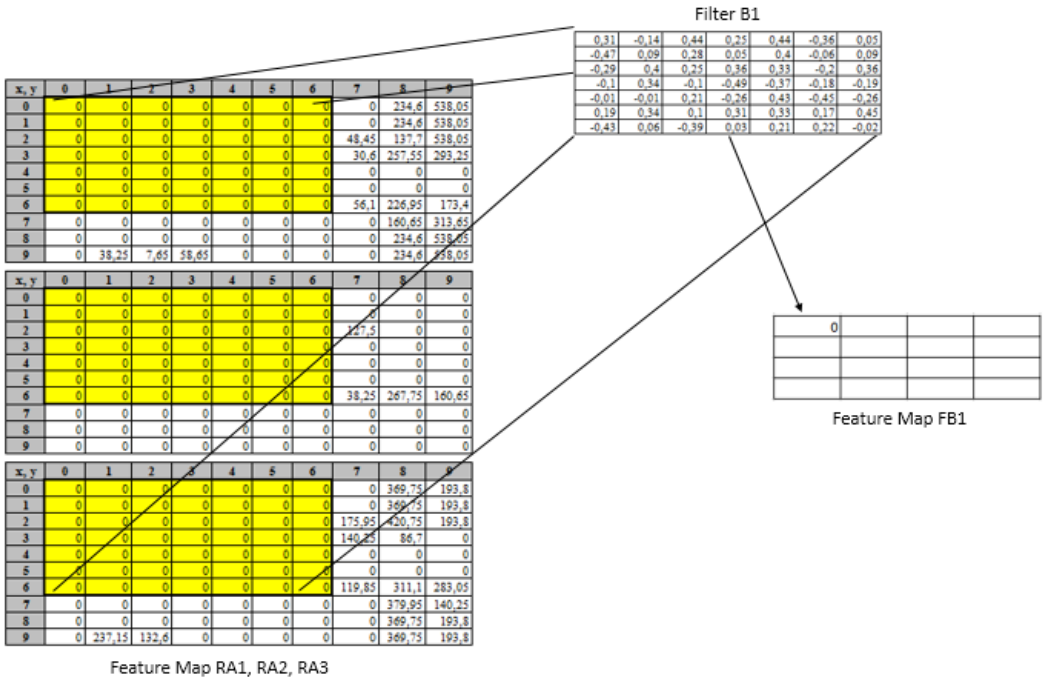
$$\boldsymbol{FB1_{i,j} = \sum_{C=1}^{3} \sum_{m=0}^{6} \sum_{n=0}^{6} B1_{m,n} \cdot RAc_{i*1+m,j*1+n} + bB_0} \quad \textbf{(2.5)}$$

$$FB1_{0,0} = (B1_{0,0}.RA1_{0,0} + B1_{0,1}.RA1_{0,1} + B3_{0,2}.RA1_{0,2} + \cdots + B1_{6,6}.RA3_{6,6}) + bB_0$$

= (0,31*0 + -0,14*0 + 0,44*0 + ... + -0,02*0) + 0

= 0

Berikut adalah visualisasi operasi konvolusi filter B1 yang menghasilkan $FB1_{0,0}$ dapat dilihat pada Gambar 3.13 Visualisasi 1 operasi *Convolution Layer* B1.



Gambar 3.13 Visualisasi 1 operasi *Convolution Layer* B1

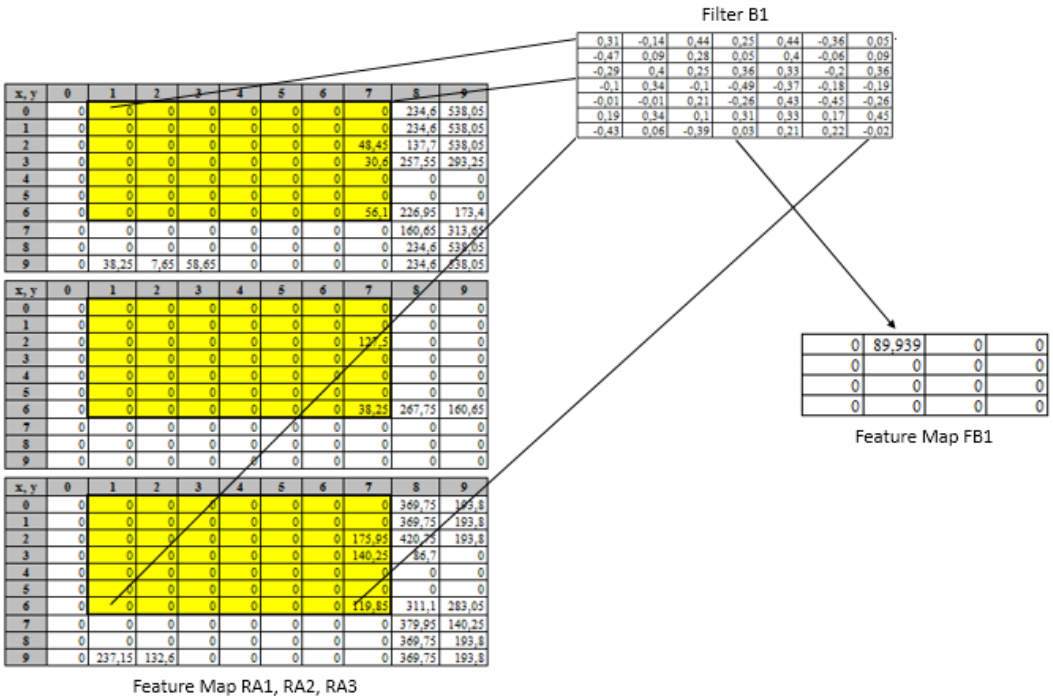
Setelah itu geser filter ke kanan sebanyak nilai *stride* yaitu 1 dan lakukan kembali operasi konvolusi.

$$FB1_{0,1} = (B1_{0,0}.RA1_{0,1} + B1_{0,1}.RA1_{0,2} + B3_{0,2}.RA1_{0,3} + \cdots + B1_{6,6}.RA3_{6,7}) + bB_0$$

= (0,31*0 + -0,14*0 + 0,44*0 + ... + -0,02*119,85) + 0

= 89,9385

Berikut adalah visualisasi operasi konvolusi filter B1 yang menghasilkan $FB1_{0,1}$ dapat dilihat pada Gambar 3.14 Visualisasi 2 operasi *Convolution Layer* B1.



Gambar 3.14 Visualisasi 2 operasi *Convolution Layer* B1

Hasil proses keseluruhan pada filter B1 dapat dilihat Tabel 3.16 *Feature Map* FB1.

Tabel 3.16 *Feature Map* FB1

x, y	0	1	2	3
0	0	89,9385	150,119	145,554
1	0	189,567	537,336	680,468
2	0	-22,721	-152,75	47,991
3	-36,414	-165,04	-173,43	-211,75

Langkah selanjutnya jalankan fungsi aktivasi ReLU pada *feature map* FA1. Berikut perhitungannya menggunakan persamaan (2.7).

$$f(x) = max(0, x) \quad (2.7)$$

$RB1_{0,0} = max(0, FB1_{0,0})$

$= max(0, 0)$

$= 0$

Hasil keseluruhan dari fungsi aktivasi ReLU menghasilkan *feature map* RB1 yang dapat dilihat pada Tabel 3.17 *Feature Map* ReLU RB1.

Tabel 3.17 *Feature Map* ReLU RB1

x, y	0	1	2	3
0	0	89,9385	150,1185	145,554
1	0	189,567	537,336	680,4675
2	0	0	0	47,991
3	0	0	0	0

Selanjutnya proses konvolusi dilakukan terhadap *feature map* RA1, RA2 dan RA3 dengan bobot pada filter B2. Bobot filter B2 dapat dilihat pada Tabel 3.18 Bobot *Convolutional Filter* B2 dan biasnya diisi 0.

Tabel 3.18 Bobot *Convolutional Filter* B2

x, y	0	1	2	3	4	5	6
0	-0,39	0,41	-0,16	-0,18	0,4	0,41	0,06
1	0,33	0,21	0,12	0,34	0	0,07	0,13
2	0,46	-0,04	0,4	-0,2	-0,44	0,36	-0,12
3	-0,34	-0,08	-0,11	-0,16	0,1	-0,14	0,4
4	-0,39	0,26	-0,22	0,1	-0,05	-0,17	-0,39
5	0,14	-0,16	-0,22	0,26	0,4	0,42	-0,13
6	-0,26	-0,43	0,12	0,4	0,01	0,22	0,33

Proses ini akan menghasilkan feature map FB2. Berikut hasil proses konvolusi dapat dilihat pada Tabel 3.19 *Feature Map* FB2.

Tabel 3.19 *Feature Map* FB2

x, y	0	1	2	3
0	0	96,798	601,316	856,775
1	0	-2,601	317,297	1052,1
2	0	-40,214	12,9795	630,003
3	-78,132	-28,943	231,387	631,508

Setelah itu jalankan fungsi aktivasi ReLU pada *feature map* FB2. Hasil dari fungsi aktivasi dapat dilihat pada Tabel 3.20 *Feature Map* ReLU RB2.

Tabel 3.20 *Feature Map* ReLU RB2

x, y	0	1	2	3
0	0	96,798	601,3155	856,7745
1	0	0	317,2965	1052,1045
2	0	0	12,9795	630,003
3	0	0	231,387	631,5075

Terakhir, lakukan proses konvolusi terhadap RA1, RA2 dan RA3 dengan filter terakhir B3 untuk menghasilkan *feature map* FB3. Bobot filter B3 dapat dilihat pada Tabel 3.21 Bobot *Convolutional* B3 dan biasnya diisi 0.

Tabel 3.21 Bobot *Convolutional* B3

x, y	0	1	2	3	4	5	6
0	0,36	-0,32	0,39	-0,26	-0,31	0	-0,19
1	0,03	0,25	0,08	0,26	0,17	0,1	-0,41
2	-0,37	-0,16	-0,37	0,07	0,06	0,41	-0,4
3	-0,29	0,15	0,04	0,32	-0,2	-0,15	-0,3
4	0,13	0,21	0,37	-0,17	-0,49	-0,44	0,34
5	0,49	-0,48	-0,11	-0,03	0,41	0,29	-0,43
6	0,46	-0,24	0,03	0,29	0,4	0,43	-0,48

Proses ini akan menghasilkan feature map FB3. Berikut hasil proses konvolusi dapat dilihat pada Tabel 3.22 *Feature Map* FB3.

Tabel 3.22 *Feature Map* FB3

x, y	0	1	2	3
0	0	-294,83	-865,29	-459,15
1	0	-304,73	-920,12	-218,61
2	0	-64,082	-572,99	-684,42
3	-44,88	-1,938	-654,99	-744,93

Setelah itu jalankan fungsi aktivasi ReLU pada *feature map* FB3. Hasil dari fungsi aktivasi dapat dilihat pada Tabel 3.23 *Feature Map* ReLU RB3.

Tabel 3.23 *Feature Map* ReLU RB3

x, y	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

Feature Map RB1, RB2 dan RB3 akan digunakan pada layer selanjutnya, yaitu *fully-connected layer*. *Fully-connected layer* adalah sebuah *neural network* yang memiliki *input layer*, *hidden layer* dan *output layer*. Berikut adalah operasi-operasi yang terjadi di masing-masing layer.

3. *Input Layer* (*Flatten*)

Pada *input layer* akan terjadi penggabungan dari matriks *feature map* RB1, RB2 dan RB3. Matriks akan diratakan menjadi sebuah vektor dengan panjang sejumlah keseluruhan pixel dari RB1, RB2 dan RB3. Total pixel dari ketiga *feature map* adalah 48, sehingga *input layer* X memiliki 48 node. Berikut adalah vektor dari *input layer* X.

Tabel 3.24 *Input Layer X*

Xi	**Nilai**	***Xi***	**Nilai**	***Xi***	**Nilai**
0	0	**16**	0	**32**	0
1	89,9385	**17**	96,798	**33**	0
2	150,119	**18**	601,316	**34**	0
3	145,554	**19**	856,775	**35**	0
4	0	**20**	0	**36**	0
5	189,567	**21**	0	**37**	0
6	537,336	**22**	317,297	**38**	0
7	680,468	**23**	1052,1	**39**	0
8	0	**24**	0	**40**	0
9	0	**25**	0	**41**	0
10	0	**26**	12,9795	**42**	0
11	47,991	**27**	630,003	**43**	0
12	0	**28**	0	**44**	0
13	0	**29**	0	**45**	0
14	0	**30**	231,387	**46**	0
15	0	**31**	631,508	**47**	0

Selanjutnya nilai dari *input layer* X akan digunakan pada *hidden layer* Z.

4. *Hidden Layer*

Setiap node pada *input layer* X akan mengirimkan sinyal pada setiap node di *hidden layer* Z. Bobot pada *hidden layer* Z dapat dilihat pada Tabel 3.25 Bobot *Hidden Layer* V dan untuk nilai biasnya semua bernilai 0.

Tabel 3.25 Bobot *Hidden Layer* V

x, y	0	1	2	3	4	5	6	7	8	9
0	-0,14	0,13	0,15	-0,33	0,35	-0,34	-0,41	0,35	0,19	-0,26
1	-0,48	0,25	-0,28	0,35	0,38	0,39	-0,3	-0,41	-0,35	-0,06
2	-0,3	-0,42	0,47	0,07	0,21	0,25	-0,28	0,16	0,09	0,35
3	0,04	0,36	-0,02	0,47	0,15	0,16	0,26	0,45	-0,42	-0,12
4	0,24	0,2	0,34	-0,12	0,37	0,44	0,04	-0,16	-0,21	0,18
5	-0,06	0,48	-0,19	0,49	-0,08	0,36	0,23	-0,19	0,19	-0,23
6	0,03	0,22	-0,27	-0,33	0,1	-0,17	-0,09	0	-0,28	-0,19
.....	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**
44	0,24	0,21	-0,48	-0,17	-0,06	-0,41	0,3	0,3	-0,44	0,05
45	0,1	-0,19	-0,49	-0,42	0,27	-0,38	-0,08	0,14	0,01	-0,13
46	0,13	-0,31	-0,3	0,42	-0,08	-0,35	-0,1	0,29	0,19	-0,25
47	0,29	-0,08	0,12	-0,35	0,24	-0,15	0	0,49	-0,19	-0,42

Proses perhitungan pada *hidden layer* Z menggunakan persamaan (2.9). Berikut adalah contoh perhitungannya.

$$\boldsymbol{z\_in_i} = \sum_{j=0}^{n=47} \boldsymbol{X_j} * \boldsymbol{V_{j,i}} + \boldsymbol{bV_i} \quad \textbf{(2.9)}$$

$z\_in_0 = (X_0*V_{0,0} + X_1*V_{1,0} + X_2*V_{2,0} + X_3*V_{3,0} + ... + X_{47}*V_{47,0}) + bV_0$

$= (0*-0,14 + 89,9385*-0,48 + 150,119*-0,3 +145,554*0,04 + ... + 0*0,29) + 0$

$= 1098,70932$

Hidden layer Z memiliki 10 node sehingga pada proses perhitungan akan menghasilkan 10 nilai z_in. Hasil perhitungan *z_in* keseluruhan dapat dilihat pada Tabel 3.26 Nilai *z_in*.

Tabel 3.26 Nilai *z_in*

i	***z_in***
0	1098,70932
1	1207,1009
2	-375,77795
3	629,970105
4	103,58559
5	-458,21715
6	101,162325
7	661,017885
8	-407,46603
9	63,602865

Selanjutnya jalankan fungsi aktivasi ReLU pada setiap nilai *z_in* dan menghasilkan nilai *Z*. Berikut adalah contoh perhitungannya menggunakan persamaan (2.7).

$$f(x) = max(0, x) \quad (2.7)$$

$$Z_1 = max(0,\ z\_in_1)$$

$$= max(0,\ 1098{,}70932)$$

$$= 1098{,}70932$$

Hasil dari fungsi aktivasi dapat dilihat pada Tabel 3.27 Nilai Z.

Tabel 3.27 Nilai Z

i	*Z*
0	1098,70932
1	1207,1009
2	0
3	629,970105
4	103,58559
5	0
6	101,162325
7	661,017885
8	0
9	63,602865

5. *Output Layer*

Ini adalah tahap terakhir dari *feedforward*. Hasil perhitungan di *hidden layer* Z akan digunakan pada perhitungan di *output layer Y*. *Output layer Y* memiliki 20 bobot dapat dilihat pada Tabel 3.28 Bobot *Output Layer* W dan biasnya bernilai 0.

Tabel 3.28 Bobot *Output Layer* W

x,y	0	1
0	0,21	0,27
1	-0,35	-0,09
2	-0,09	0,27
3	-0,15	-0,21
4	0,61	0,32
5	0,32	-0,19
6	0,11	0,37
7	0,36	-0,16
8	0,35	0,34
9	-0,43	0,04

Proses perhitungan pada *output layer Y* menggunakan persamaan (2.9). Berikut adalah contoh perhitungannya.

$$y\_in_i = \sum_{j=0}^{n=9} Z_j * W_{j,i} + bW_i \quad \textbf{(2.9)}$$

y_in_0 = (Z_0*$W_{0,0}$ + Z_1*$W_{1,0}$ + Z_2*$W_{2,0}$ + ... + Z_9*$W_{9,0}$) + bW_0

= (1098,70932*0,21 + 1207,1009*-0,35 + 0*-0,09 +...+ 63,6202865*-0,43)+0

= -1,3196

Hasil perhitungan *y_in* keseluruhan dapat dilihat pada Tabel 3.29 Nilai *y_in*.

Tabel 3.29 Nilai *y_in*

i	*y_in*
0	-1,3195995
1	23,07741585

Selanjutnya jalankan fungsi aktivasi *linear* pada setiap nilai *y_in* untuk menghasilkan nilai keluaran *Y*, dengan persamaan (2.8).

$$\varphi(x) = x \quad \textbf{(2.8)}$$

$Y = \varphi(y\_in_0)$

= -1,3196

Hasil perhitungan Y dapat dilihat pada Tabel 3.30 Nilai *Output* Y.

Tabel 3.30 Nilai *Output* Y

i	*Y*
0	-1,3195995
1	23,07741585

3.7.1.2. *Backpropagation*

Pada tahap ini akan dilakukan beberapa proses perhitungan. Pertama akan dihitung gradien dari *loss function* terhadap semua bobot di setiap layer dengan mencari turunan parsial dari fungsi tersebut dan selanjutnya akan dilakukan penyesuaian bobot dengan menggunakan Adam *Oprimizer* dengan $\alpha = 0{,}001$, $\beta_1 = 0{,}9$ $\beta_2 = 0{,}999$ dan $\varepsilon = 10^{-8}$.

1. Perhitungan Nilai *Loss*

Perhitungan nilai *Loss* menggunakan sebuah fungsi *Loss*. Fungsi tersebut akan mengukur seberapa bagus peforma dari sebuah model dalam melakukan

prediksinya terhadap target. Pada metode CNN yang dikombinasikan dengan *Q-Learning*, penentuan target berdasarkan *terminal* [4]. Terminal adalah status yang menandakan apakah pada *next state* permainan berakhir atau tidak, bernilai *True* jika permainan berakhir dan *False* jika sebaliknya. Adapun penentuan nilai target berdasarkan persamaan (2.10).

$$Set\ Y' = \begin{cases} r\ ,\ terminal = True \\ r + \gamma\ max_a Q(S_{t+1}, a'; \theta)\ ,\ terminal = False \end{cases} \quad \textbf{(2.10)}$$

Dimana r adalah *reward* yang didapatkan, γ adalah *discount rate* dengan nilai 0,99 dan $max_a Q(S_{t+1}, a'; \theta)$ adalah *output* dengan nilai maksimal dari *feedforward* CNN dengan *input next state*.

Karena status *terminal* data latih bernilai *True* (sudah dijelaskan pada bagian awal sub bab pelatihan CNN) maka nilai target adalah *reward* yang didapat, yaitu -1. Nilai target tersebut ditempatkan pada *action* yang diambil, pada data latih ini *action* yang diambil adalah 0 dan untuk *action* selain 0 akan diberi nilai 0 baik pada vektor target maupun vektor hasil prediksi [5]. Nilai vektor target *Y'* dapat dilihat pada Tabel 3.31 Nilai Target Y dan nilai vektor perdiksi *Y* dapat dilihat pada Tabel 3.32 Nilai Prediksi Y.

Tabel 3.31 Nilai Target Y'

Y'	Nilai
0	-1
1	0

Tabel 3.32 Nilai Prediksi Y

Y	Nilai
0	-1,3195995
1	0

Pada kasus ini, CNN digunakan untuk regresi dan *loss function* yang biasa digunakan adalah *mean squared error* menggunakan persamaan (2.11)

$$L = \frac{1}{n}\sum_{i=1}^{n} (Y_{i-1} - Y'_{i-1})^2 \quad \textbf{(2.11)}$$

dimana L adalah nilai loss, *Y* adalah hasil prediksi dan *Y'* adalah target prediksi.

$$L = \frac{1}{2}((Y_0 - Y'_0)^2 + (Y_1 - Y'_1))^2$$

$$= ½\ ((-1{,}3195995 - (-1))^2 + (0 - 0)^2)$$

= 0,051072

2. Perhitungan Gradien Terhadap Bobot W

Setelah perhitungan nilai loss, selanjutnya akan dilakukan perhitungan gradien terhadap bobot $W_{i,j}$. Perhitungan akan menggunakan rumus *chain rule* berdasarkan persamaan. Berikut adalah perhitungan gradien pada bobot $W_{i,j}$ berdasarkan persamaan (2.12).

$$\frac{\partial L}{\partial W_{j,i}} = \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y_{in_i}}\frac{\partial y_{in_i}}{\partial W_{j,i}} \quad \textbf{(2.12)}$$

$$\frac{\partial L}{\partial W_{j,i}} = 2(\mathrm{Y_i} - \mathrm{Y'_i}) * 1 * Z_j$$

$$\frac{\partial L}{\partial W_{0,0}} = 2\big((-1{,}3195995) - (-1)\big) * 1 * 1098{,}70932$$

$$= -702{,}2938986$$

Hasil dari perhitungan $\frac{\partial L}{\partial W}$ keseluruhan dapat dilihat pada Tabel 3.33 Matriks Gradien $\boldsymbol{\frac{\partial L}{\partial W}}$.

Tabel 3.33 Matriks Gradien $\boldsymbol{\frac{\partial L}{\partial W}}$

j, i	**0**	**1**
0	-702,2938986	0
1	-771,577685	0
2	0	0
3	-402,6762611	0
4	-66,21180554	0
5	0	0
6	-64,66285698	0
7	-422,5219711	0
8	0	0
9	-40,65488771	0

Setelah gradien $\frac{\partial L}{\partial W}$ diketahui, selanjutnya lakukan perhitungan nilai bobot W, baru.menggunakan Adam *Oprimizer* dengan $\alpha = 0{,}001$, $\beta_1 = 0{,}9$ $\beta_2 = 0{,}999$ dan $\varepsilon = 10^{-8}$. Berikut tahapan-tahapan perhitungannya, dimulai dari menghitung nilai m_t menggunakan persamaan (2.17).

$$\boldsymbol{m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \frac{\partial L}{\partial W_{0,0}}} \quad \textbf{(2.17)}$$

$m_1 = 0{,}9 * 0 + (1 - 0{,}9) * -702{,}2938986$

$m_1 = -70{,}22938986$

Selanjutnya, hitung nilai v_1 menggunakan persamaan (2.18).

$$\boldsymbol{v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * \left(\frac{\partial L}{\partial W_{0,0}}\right)^2} \quad \textbf{(2.18)}$$

v_1 = 0,999 * 0 + (1 – 0,999) * -702,2938986^2

v_1 = 493,2167201

Lalu hitung nilai $\hat{m}_1$ menggunakan persamaan (2.19).

$$\boldsymbol{\hat{m}_t = m_t / (1 - \beta_1^t)} \quad \textbf{(2.19)}$$

$\hat{m}_1$ = -70,22938986 / (1 – 0,9^1)

$\hat{m}_1$ = -702,2938986

Setelah itu lakukan perhitungan nilai $\hat{\boldsymbol{v}}_1$ menggunakan persamaan (2.20).

$$\boldsymbol{\hat{v}_t = \mathbf{v_t} / \left(1 - \beta_2^t\right)} \quad \textbf{(2.20)}$$

$\hat{\boldsymbol{v}}_1$ = 493,2167201 / (1 – 0,999^1)

$\hat{\boldsymbol{v}}_1$ = 493216,7201

Terakhir, lakukan perhitungan bobot baru menggunakan persamaan (2.21).

$$\boldsymbol{W'_{0,0} = W_{0,0} - \alpha * \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)} \quad \textbf{(2.21)}$$

$W'_{0,0} = W_{0,0} - \alpha * \hat{m}_1 / (\sqrt{\hat{v}_1} + \varepsilon)$

$W'_{0,0}$ = 0,21 - 0,001 * -702,2938986 / ($\sqrt{493216,7201}$ + 10^{-8})

$W'_{0,0}$ = 0,211

Hasil perhitungan *W'* keseluruhan dapat dilihat pada Tabel 3.34 Bobot Baru *W*

Tabel 3.34 Bobot Baru *W'*

x, y	0	1
0	0,211	0,27
1	-0,349	-0,09
2	-0,09	0,27
3	-0,149	-0,21
4	0,611	0,32
5	0,32	-0,19
6	0,111	0,37
7	0,361	-0,16
8	0,35	0,34
9	-0,429	0,04

3. Perhitungan Gradien Terhadap Bobot V

Setelah menghitung gradien pada bobot W, selanjutnya akan dilakukan perhitungan terhadap bobot V. Perhitungan gradien pada bobot V berdasarkan persamaan (2.13) dapat dilihat di bawah ini.

$$\frac{\partial L}{\partial V_{k,j}} = \sum_{i}^{m} \frac{\partial L}{\partial Y_i} \frac{\partial Y_i}{\partial y_{in_i}} \frac{\partial y_{in_i}}{\partial Z_j} \frac{\partial Z_j}{\partial z\_in_j} \frac{\partial z\_in_j}{\partial V_{k,j}} \quad \textbf{(2.13)}$$

$$\frac{\partial L}{\partial V_{k,j}} = \sum_{i=0}^{1} -(Y_i - Y'_i) * 1 * W_{j,i} * \begin{cases} 1 & z\_in_j \geq 0 \\ 0 & z\_in_j < 0 \end{cases} * X_k$$

$$\frac{\partial L}{\partial V_{0,0}} = \sum_{i=0}^{1} 2(\mathrm{Y_i} - \mathrm{Y'_i}) * 1 * W_{0,i} * \begin{cases} 1 & z\_in_0 \geq 0 \\ 0 & z\_in_0 < 0 \end{cases} * X_0$$

$$= \big(2\big((-1{,}3195995) - (-1)\big) * 1 * 0{,}21 * 1 * 0\big) + (2(0-0) * 1 * 0{,}27 * 1 * 0)$$

$$= 0$$

Hasil perhitungan $\frac{\partial L}{\partial V}$ keseluruhan dapat dilihat pada Tabel 3.35 Matriks Gradien $\frac{\partial L}{\partial V}$.

Tabel 3.35 Matriks Gradien $\frac{\partial L}{\partial V}$

j, i	0	1	2	3	4	.....	9
0	0	0	0	0	0	.....	0
1	-12,07260584	20,12100974	0	8,623289889	-35,06804555	.....	24,72009768
2	-20,15067497	33,58445828	0	14,39333926	-58,532913	.....	41,26090589
3	-19,53797396	32,56328994	0	13,95569569	-56,75316246	.....	40,00632764
4	0	0	0	0	0	.....	0
5	-25,44591773	42,40986289	0	18,17565552	-73,91433247	.....	52,10354584
.....	.....	.....	.....	.....	.....	.....	.....
44	0	0	0	0	0	.....	0
45	0	0	0	0	0	.....	0
46	0	0	0	0	0	.....	0
47	0	0	0	0	0	.....	0

Setelah melakukan perhitungan gradien $\frac{\partial L}{\partial V}$, selanjutnya akan dilakukan perhitungan nilai bobot V baru. Perhitungan bobot baru menggunakan Adam *Oprimizer*. Pertama-tama hitung dahulu m_1 menggunakan persamaan (2.17).

$$\boldsymbol{m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \frac{\partial L}{\partial V_{0,0}}} \quad \textbf{(2.17)}$$

$m_1 = 0{,}9 * 0 + (1 - 0{,}9) * 0$

$m_1 = 0$

Selanjutnya, hitung nilai v_1 menggunakan persamaan (2.18).

$$\boldsymbol{v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * \left(\frac{\partial L}{\partial V_{0,0}}\right)^2} \quad \textbf{(2.18)}$$

$v_1 = 0{,}999 * 0 + (1 - 0{,}999) * 0^2$

$v_1 = 0$

Lalu hitung nilai $\hat{m}_1$ menggunakan persamaan (2.19).

$$\hat{m}_t = m_t / (1 - \beta_1^t) \quad \textbf{(2.19)}$$

$\hat{m}_1 = 0 / (1 - 0{,}9^1)$

$\hat{m}_1 = 0$

Setelah itu lakukan perhitungan nilai $\hat{v}_1$ menggunakan persamaan (2.20).

$$\hat{v}_t = v_t / \left(1 - \beta_2^t\right) \quad \textbf{(2.20)}$$

$\hat{v}_1 = 0 / (1 - 0{,}999^1)$

$\hat{v}_1 = 0$

Terakhir, lakukan perhitungan bobot baru menggunakan persamaan (2.21).

$$V'_{0,0} = V_{0,0} - \alpha * \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) \quad \textbf{(2.21)}$$

$V'_{o,0} = V_{0,0} - \alpha * \hat{m}_1 / (\sqrt{\hat{v}_1} + \varepsilon)$

$V'_{0,0} = -0{,}14 - 0{,}001 * 0 / (\sqrt{0} + 10^{-8})$

$V'_{0,0} = -0{,}14$

Hasil dari perhitungan bobot baru $\hat{v}$ keseluruhan dapat dilihat pada Tabel 3.28 Bobot *Output Layer* .

Tabel 3.36 Bobot Baru V

x, y	0	1	2	3	4	5	6	7	8	9
0	-0,14	0,13	0,15	-0,33	0,35	-0,34	-0,41	0,35	0,19	-0,26
1	-0,479	0,249	-0,28	0,349	0,381	0,39	-0,299	-0,409	-0,35	-0,061
2	-0,299	-0,421	0,47	0,069	0,211	0,25	-0,279	0,161	0,09	0,349
3	0,041	0,359	-0,02	0,469	0,151	0,16	0,261	0,451	-0,42	-0,121
4	0,24	0,2	0,34	-0,12	0,37	0,44	0,04	-0,16	-0,21	0,18
5	-0,059	0,479	-0,19	0,489	-0,079	0,36	0,231	-0,189	0,19	-0,231
.....	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**
44	0,24	0,21	-0,48	-0,17	-0,06	-0,41	0,3	0,3	-0,44	0,05
45	0,1	-0,19	-0,49	-0,42	0,27	-0,38	-0,08	0,14	0,01	-0,13
46	0,13	-0,31	-0,3	0,42	-0,08	-0,35	-0,1	0,29	0,19	-0,25
47	0,29	-0,08	0,12	-0,35	0,24	-0,15	0	0,49	-0,19	-0,42

4. Perhitungan Gradien Terhadap Bobot Filter B

Selanjutnya akan dilakukan perhitungan gradien terhadap bobot pada filter B menggunakan rumus *chain rule*, berikut adalah persamaannya (2.15)

$$\frac{\partial L}{\partial Bi} = \frac{\partial L}{\partial Xi}\frac{\partial Xi}{\partial FBi}\frac{\partial FBi}{\partial Bi} \quad \textbf{(2.15)}$$

Sebelum melakukan perhitungan terhadap gradien filter B, perlu dihitung terlebih dahulu nilai dari $\frac{\partial L}{\partial X}$ menggunakan persamaan (2.14).

$$\frac{\partial L}{\partial X_k} = \sum_j^n \sum_i^m \frac{\partial L}{\partial Y_i} \frac{\partial Y_i}{\partial y_{in_i}} \frac{\partial y_{in_i}}{\partial Z_j} \frac{\partial Z_j}{\partial z\_in_j} \frac{\partial z\_in_j}{\partial X_k} \quad \textbf{(2.14)}$$

$$\frac{\partial L}{\partial X_k} = \sum_{j=0}^{9} \sum_{i=0}^{1} 2(\mathrm{Y_i} - \mathrm{Y'_i}) * 1 * W_{j,i} * \begin{cases} 1 & z\_in_j \geq 0 \\ 0 & z\_in_j < 0 \end{cases} * V_{k,j}$$

$$\frac{\partial L}{\partial X_0} = \sum_{j=0}^{9} \sum_{i=0}^{1} 2(\mathrm{Y_i} - \mathrm{Y'_i}) * 1 * W_{0,i} * \begin{cases} 1 & z\_in_j \geq 0 \\ 0 & z\_in_j < 0 \end{cases} * V_{0,j}$$

$$= \big(2\big((-1{,}3195995) - (-1)\big) * 1 * 0{,}21 * 1 * -14\big) + (2(0-0) * 1 * 0{,}27 * 1 * -14) + \cdots + \big(2\big((-1{,}3195995) - (-1)\big) * 1 * -0{,}43 * 1 * -0{,}26\big) + (2(0-0) * 1 * 0{,}04 * 1 * -0{,}26)$$

$$= -0{,}243406979$$

Hasil perhitungan $\frac{\partial L}{\partial X}$ keseluruhan dapat dilihat pada Tabel 3.37 Matriks $\frac{\partial L}{\partial X}$.

Tabel 3.37 Matriks $\frac{\partial L}{\partial X}$

i	aL / aX	*i*	aL/ aX	*i*	aL/ aX
0	-0,243406979	**16**	-0,324329573	**32**	-0,110325747
1	0,104700796	**17**	0,256766238	**33**	-0,124452045
2	-0,049793602	**18**	0,099011925	**34**	0,113841342
3	-0,093067374	**19**	-0,289429307	**35**	-0,103933757
4	-0,059765106	**20**	-0,129693477	**36**	-0,367347665
5	0,157946073	**21**	0,008629186	**37**	0,064431259
6	-0,071334608	**22**	-0,021541006	**38**	-0,01303966
7	0,066221016	**23**	-0,199749687	**39**	-0,284251795
8	0,390486669	**24**	-0,158585272	**40**	0,260857112
9	-0,226020766	**25**	-0,181148997	**41**	-0,216624541
10	-0,129246038	**26**	0,087186744	**42**	0,251333047
11	0,25574352	**27**	0,252419685	**43**	0,341076586
12	-0,036753942	**28**	-0,096391209	**44**	-0,054523675
13	-0,022371965	**29**	0,035795144	**45**	-0,263797427
14	0,059829026	**30**	0,046469767	**46**	-0,143755855
15	0,071142849	**31**	0,274216371	**47**	-0,412155515

Setelah nilai $\frac{\partial L}{\partial X}$ diketahui, susun nilai-nilai tersebut menjadi matriks 4x4 berjumlah 3 buah. Matriks $\frac{\partial L}{\partial X1}$ dapat dilihat pada Tabel 3.38 Matriks $\frac{\partial L}{\partial X1}$.

Tabel 3.38 Matriks $\frac{\partial L}{\partial X1}$

i, j	0	1	2	3
0	-0,243406979	0,104700796	-0,049793602	-0,093067374
1	-0,059765106	0,157946073	-0,071334608	0,066221016
2	0,390486669	-0,226020766	-0,129246038	0,25574352
3	-0,036753942	-0,022371965	0,059829026	0,071142849

Sedangkan untuk nilai pada matriks $\frac{\partial L}{\partial X2}$ dapat dilihat pada Tabel 3.39 Matriks $\frac{\partial L}{\partial X2}$.

Tabel 3.39 Matriks $\frac{\partial L}{\partial X2}$

i, j	0	1	2	3
0	-0,324329573	0,256766238	0,099011925	-0,289429307
1	-0,129693477	0,008629186	-0,02154101	-0,199749687
2	-0,158585272	-0,181148997	0,087186744	0,252419685
3	-0,096391209	0,035795144	0,046469767	0,274216371

Matriks terakhir, matriks $\frac{\partial L}{\partial X3}$ dapat dilihat pada Tabel 3.40 Matriks $\frac{\partial L}{\partial X3}$.

Tabel 3.40 Matriks $\frac{\partial L}{\partial X3}$

i, j	0	1	2	3
0	-0,110325747	-0,124452045	0,113841342	-0,10393376
1	-0,367347665	0,064431259	-0,01303966	-0,2842518
2	0,260857112	-0,216624541	0,251333047	0,341076586
3	-0,054523675	-0,263797427	-0,143755855	-0,41215552

Setelah menyusun matriks $\frac{\partial L}{\partial X}$ menjadi 4x4x3 yang terdiri dari $\frac{\partial L}{\partial X1}$, $\frac{\partial L}{\partial X2}$ dan $\frac{\partial L}{\partial X3}$. Selanjutnya melakukan perhitungan untuk mencari gradien $\frac{\partial L}{\partial FBi}$.

$$\frac{\partial L}{\partial FBi} = \frac{\partial L}{\partial Xi}\frac{\partial Xi}{\partial FBi}$$

$$\frac{\partial L}{\partial FB1_{0,0}} = \frac{\partial L}{\partial X1_{0,0}}\frac{\partial X1_{0,0}}{\partial FB1_{0,0}}$$

$$= \frac{\partial L}{\partial X1_{0,0}} * \begin{cases} 1 & FB1_{0,0} \geq 0 \\ 0 & FB1_{0,0} < 0 \end{cases}$$

$$= -0{,}243406979 * 1$$

$$= -0{,}243406979$$

Hasil keseluruhan perhitungan dari matriks $\frac{\partial L}{\partial FB1}$ dapat dilihat pada Tabel 3.41 Matriks $\frac{\partial L}{\partial FB1}$.

Tabel 3.41 Matriks $\frac{\partial L}{\partial FB1}$

i, j	0	1	2	3
0	-0,243406979	0,104700796	-0,049793602	-0,093067374
1	-0,059765106	0,157946073	-0,071334608	0,066221016
2	0,390486669	0	0	0,25574352
3	0	0	0	0

Lakukan proses yang sama untuk matriks $\frac{\partial L}{\partial X2}$, menghasilkan $\frac{\partial L}{\partial FB2}$ dapat dilihat pada Tabel 3.42 Matriks $\frac{\partial L}{\partial FB2}$ dan $\frac{\partial L}{\partial X3}$ menghasilkan $\frac{\partial L}{\partial FB3}$ yang dapat dilihat pada Tabel 3.43 Matriks $\frac{\partial L}{\partial FB3}$.

Tabel 3.42 Matriks $\frac{\partial L}{\partial FB2}$

i, j	0	1	2	3
0	-0,324329573	0,256766238	0,099011925	-0,289429307
1	-0,129693477	0	-0,02154101	-0,199749687
2	-0,158585272	0	0,087186744	0,252419685
3	0	0	0,046469767	0,274216371

Tabel 3.43 Matriks $\frac{\partial L}{\partial FB3}$

i, j	0	1	2	3
0	-0,110325747	0	0	0
1	-0,367347665	0	0	0
2	0,260857112	0	0	0
3	0	0	0	0

Setelah perhitungan $\frac{\partial L}{\partial FB}$, untuk mendapatkan gradien $\frac{\partial L}{\partial B}$ lakukan operasi konvolusi antara $\frac{\partial L}{\partial FB1}$ dengan *featuure map* RA1, RA2 dan RA3. Berikut adalah perhitungannya.

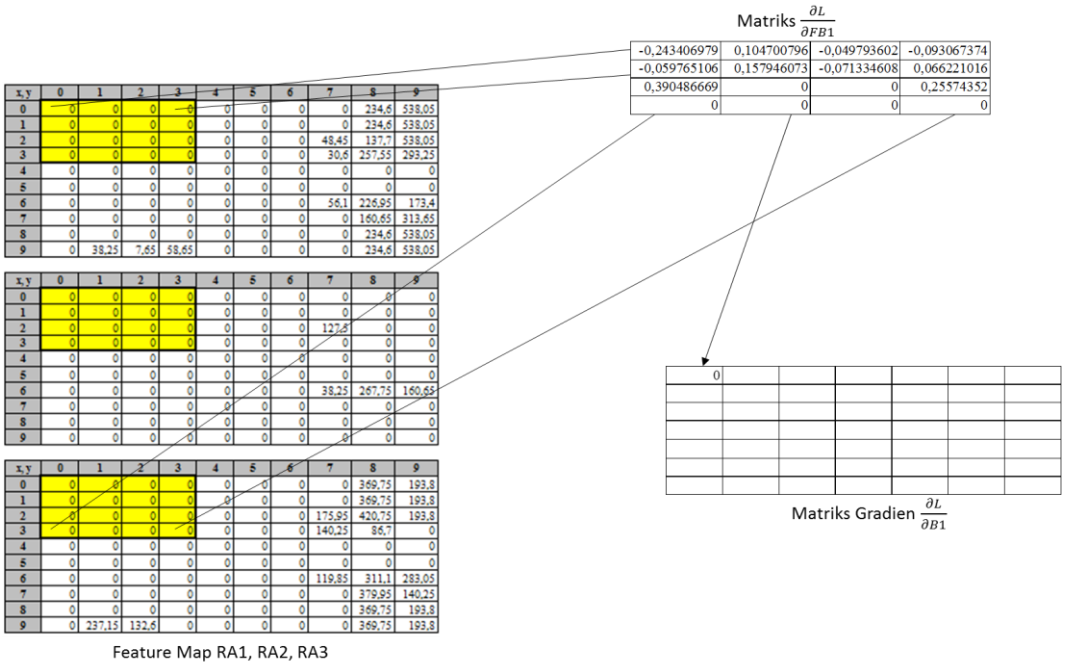
$$\frac{\partial L}{\partial B1_{i,j}} = \sum_{C=1}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} \frac{\partial L}{\partial FB1_{m,n}} \cdot RAc_{i*1+m,j*1+n} \quad \textbf{(2.5)}$$

$$\frac{\partial L}{\partial B1_{0,0}} = \frac{\partial L}{\partial FB1_{0,0}} . RA1_{0,0} + \frac{\partial L}{\partial FB1_{0,1}} . RA1_{0,1} + \frac{\partial L}{\partial FB1_{0,2}} . RA1_{0,2} + \cdots + \frac{\partial L}{\partial FB1_{3,3}} . RA3_{3,3}$$

= (0*-0,243406979 + 0*0,104700796 + 0*-0,049793602 + ... + 0*0)

= 0

Visualisasi proses tersebut dapat dilihat pada Gambar 3.15 Visualisasi 1 Operasi Konvolusi Matriks Gradien $\frac{\partial L}{\partial B1}$.



Gambar 3.15 Visualisasi 1 Operasi Konvolusi Matriks Gradien $\frac{\partial L}{\partial B1}$

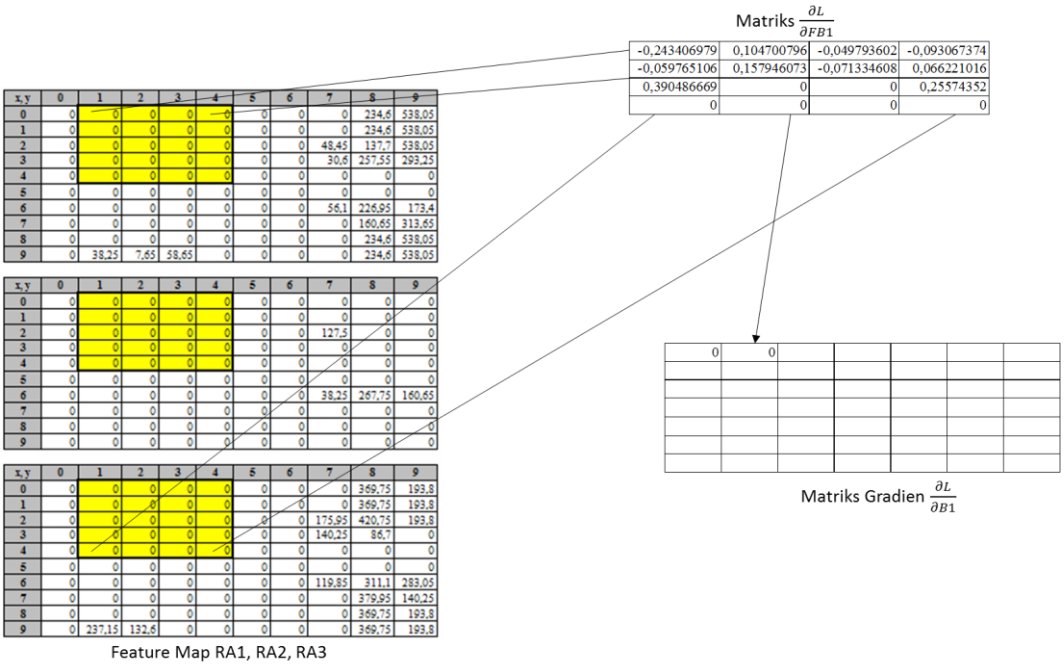
Setelah itu geser filter ke kanan dan lakukan kembali operasi konvolusi.

$$\frac{\partial L}{\partial B1_{0,1}} = \frac{\partial L}{\partial FB1_{0,0}} . RA1_{0,1} + \frac{\partial L}{\partial FB1_{0,1}} . RA1_{0,2} + \frac{\partial L}{\partial FB1_{0,2}} . RA1_{0,3} + \cdots + \frac{\partial L}{\partial FB1_{3,3}} . RA3_{3,4}$$

= (0*-0,243406979 + 0*0,104700796 + 0*-0,049793602 + ... + 0*0)

= 0

Visualisasi proses tersebut dapat dilihat pada Gambar 3.16 Visualisasi 2 Operasi Konvolusi Matriks Gradien $\frac{\partial L}{\partial B1}$.



Gambar 3.16 Visualisasi 2 Operasi Konvolusi Matriks Gradien $\frac{\partial L}{\partial B1}$

Proses tersebut terus diulang hingga seluruh permukaan matriks. Hasil proses keseluruhan dapat dilihat pada Tabel 3.44 Matriks Gradien $\frac{\partial L}{\partial B1}$.

Tabel 3.44 Matriks Gradien $\frac{\partial L}{\partial B1}$

i, j	0	1	2	3	4	5	6
0	0	0	0	0	89,99614465	126,5953722	94,31455392
1	0	0	0	0	66,99695605	43,67291692	41,00092767
2	0	0	0	0	-21,4365484	-58,88677676	-37,22692419
3	0	0	0	0	-15,90056092	-40,54568056	-26,54532403
4	0	0	0	0	54,78026196	206,0781283	157,8193261
5	0	0	0	0	14,18454171	176,3359688	133,2975943
6	0	0	0	0	-19,9350316	104,6981979	103,5314743

Setelah melakukan perhitungan gradien $\frac{\partial L}{\partial B1}$, selanjutnya akan dilakukan perhitungan nilai bobot filter B1 baru. Perhitungan bobot baru akan menggunakan Adam *Oprimizer*. Hitung dahulu m_1 menggunakan persamaan (2.17).

$$\boldsymbol{m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \frac{\partial L}{\partial B1_{0,0}}} \quad \textbf{(2.17)}$$

$m_1 = 0{,}9 * 0 + (1 - 0{,}9) * 0$

$m_1 = 0$

Selanjutnya, hitung nilai v_1 menggunakan persamaan (2.18).

$$\boldsymbol{v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * \frac{\partial L}{\partial B1_{0,0}}^2} \quad \textbf{(2.18)}$$

$v_1 = 0{,}999 * 0 + (1 - 0{,}999) * 0^2$

$v_1 = 0$

Lalu hitung nilai $\hat{m}_1$ menggunakan persamaan (2.19).

$$\boldsymbol{\hat{m}_t = m_t / (1 - \beta_1^t)} \quad \textbf{(2.19)}$$

$\hat{m}_1 = 0 / (1 - 0{,}9^1)$

$\hat{m}_1 = 0$

Setelah itu lakukan perhitungan nilai $\hat{v}_1$ menggunakan persamaan (2.20).

$$\boldsymbol{\hat{v}_t = \mathbf{v_t} / \left(1 - \beta_2^t\right)} \quad \textbf{(2.20)}$$

$\hat{v}_1 = 0 / (1 - 0{,}999^1)$

$\hat{v}_1 = 0$

Terakhir, lakukan perhitungan bobot baru menggunakan persamaan (2.21).

$$\boldsymbol{B1'_{0,0} = B1_{0,0} - \alpha * \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)} \quad \textbf{(2.21)}$$

$B1'_{0,0} = B1_{0,0} - \alpha * \hat{m}_1 / (\sqrt{\hat{v}_1} + \varepsilon)$

$B1'_{0,0} = 0{,}31 - 0{,}001 * 0 / (\sqrt{0} + 10^{-8})$

$B1'_{0,0} = 0{,}31$

Nilai Hasilperhitungan bobot baru B1 keseluruhan dapat dilihat pada Tabel 3.45 Bobot Baru Filter B1.

Tabel 3.45 Bobot Baru Filter B1

x, y	**0**	**1**	**2**	**3**	**4**	**5**	**6**
0	0,31	-0,14	0,44	0,25	0,439	-0,361	0,049
1	-0,47	0,09	0,28	0,05	0,399	-0,061	0,089
2	-0,29	0,4	0,25	0,36	0,331	-0,199	0,361
3	-0,1	0,34	-0,1	-0,49	-0,369	-0,179	-0,189
4	-0,01	-0,01	0,21	-0,26	0,429	-0,451	-0,261
5	0,19	0,34	0,1	0,31	0,329	0,169	0,449
6	-0,43	0,06	-0,39	0,03	0,211	0,219	-0,021

Selanjutnya lakukan proses konvolusi antara $\frac{\partial L}{\partial FB2}$ dengan *feature map* RA1, RA2 dan RA3, hasil dari proses keseluruhan dapat dilihat pada Tabel 3.46 Matriks Gradien $\boldsymbol{\frac{\partial L}{\partial B2}}$.

Tabel 3.46 Matriks Gradien $\frac{\partial L}{\partial B2}$

i, j	0	1	2	3	4	5	6
0	0	0	0	0	98,22612581	93,94504257	-5,252700878
1	0	0	0	0	54,0089661	-34,79339772	-152,1630478
2	0	0	0	0	-39,08898758	-145,6932577	-185,0260869
3	0	0	0	0	-6,525697647	-21,72488787	16,94503765
4	0	0	0	0	54,59379483	254,2678203	270,5033375
5	0	0	0	0	-0,736414776	182,5645252	261,0261129
6	16,43828388	2,725451852	0	0	-30,95093603	89,30504269	213,4717783

Setelah melakukan perhitungan gradien $\frac{\partial L}{\partial B2}$, selanjutnya akan dilakukan perhitungan nilai bobot filter B2 baru menggunakan Adam *Oprimizer*, hasilnya dapat dilihat pada Tabel 3.47 Bobot Baru Filter B2.

Tabel 3.47 Bobot Baru Filter B2

x, y	0	1	2	3	4	5	6
0	-0,39	0,41	-0,16	-0,18	0,399	0,409	0,061
1	0,33	0,21	0,12	0,34	-0,001	0,071	0,131
2	0,46	-0,04	0,4	-0,2	-0,439	0,361	-0,119
3	-0,34	-0,08	-0,11	-0,16	0,101	-0,139	0,399
4	-0,39	0,26	-0,22	0,1	-0,051	-0,171	-0,391
5	0,14	-0,16	-0,22	0,26	0,401	0,419	-0,131
6	-0,261	-0,431	0,12	0,4	0,011	0,219	0,329

Terakhir lakukan proses konvolusi antara $\frac{\partial L}{\partial FB3}$ dengan *feature map* RA1, RA2 dan RA3, hasil keseluruhan dapat dilihat pada Tabel 3.48 Matriks Gradien $\frac{\partial L}{\partial B3}$.

Tabel 3.48 Matriks Gradien $\frac{\partial L}{\partial B3}$

i, j	0	1	2	3	4	5	6
0	0	0	0	0	77,60537111	97,67764513	-0,426885856
1	0	0	0	0	55,96279609	-6,022692778	-5,69930922
2	0	0	0	0	-18,95379721	-58,53128304	8,751567541
3	0	0	0	0	-13,05269926	-15,05249406	10,36718136
4	0	0	0	0	40,4330505	148,0371365	113,4733998
5	0	0	0	0	10,46954269	124,2237841	48,9295297
6	0	0	0	0	-14,71395189	57,9774235	-21,81681747

Setelah gradien $\frac{\partial L}{\partial B3}$ didapat, lakukan perhitungan nilai bobot filter B3 baru menggunakan Adam *Oprimizer*, hasilnya dapat dilihat pada Tabel 3.49 Bobot Baru Filter B3.

Tabel 3.49 Bobot Baru Filter B3

x, y	0	1	2	3	4	5	6
0	0,36	-0,32	0,39	-0,26	-0,311	-0,001	-0,189
1	0,03	0,25	0,08	0,26	0,169	0,101	-0,409
2	-0,37	-0,16	-0,37	0,07	0,061	0,411	-0,401
3	-0,29	0,15	0,04	0,32	-0,199	-0,149	-0,301
4	0,13	0,21	0,37	-0,17	-0,491	-0,441	0,339
5	0,49	-0,48	-0,11	-0,03	0,409	0,289	-0,431
6	0,46	-0,24	0,03	0,29	0,401	0,429	-0,479

5. Perhitungan Gradien Terhadap Bobot Filter A

Terakhir akan dilakukan perhitungan gradien terhadap bobot pada filter A menggunakan rumus *chain* rule yang dapat dilihat pada persamaan (2.16).

$$\frac{\partial L}{\partial Ai} = \sum_{j=1}^{n} \frac{\partial L}{\partial RAj} \frac{\partial RAi}{\partial FAi} \frac{\partial FAi}{\partial Ai} \quad \textbf{(2.16)}$$

Sebelum melakukan perhitungan terhadap gradien filter A, perlu dilakukan perhitungan dari $\frac{\partial L}{\partial RAi}$ dengan melakukan operasi *cross-correlation* antara $\frac{\partial L}{\partial FBi}$ dengan Filter Bi yang dapat dilihat pada Tabel 3.15 Bobot *Convolutional Filter* B1 [14] menggunakan persamaan (2.6).

$$\frac{\partial \mathrm{L}}{\partial \mathrm{RA1}_{i,\mathrm{j}}} = \sum_{m=0}^{-3} \sum_{n=0}^{-3} \boldsymbol{B1}_{i+m,j+n} \frac{\partial L}{\partial FB1_{-\mathrm{m},-\mathrm{n}}} \quad \textbf{(2.6)}$$

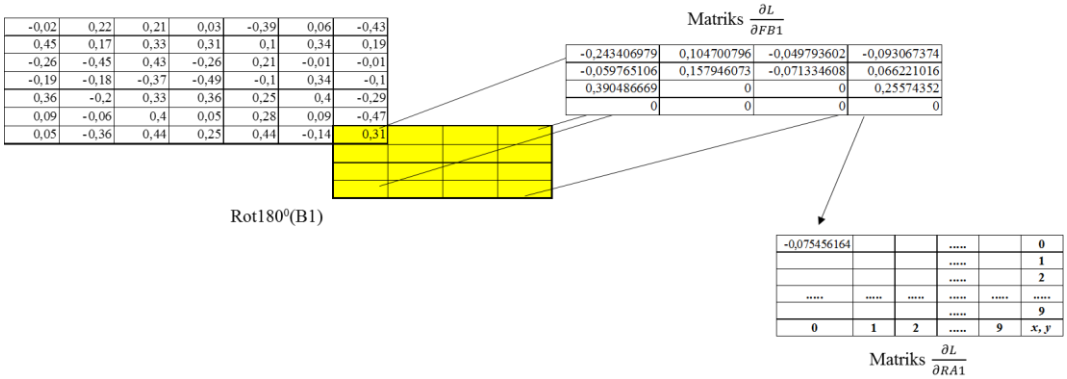
$$\frac{\partial \mathrm{L}}{\partial \mathrm{RA1}_{0,0}} = \sum_{m=0}^{-3} \sum_{n=0}^{-3} \boldsymbol{B1}_{0+m,0+n} \frac{\partial L}{\partial FB1_{-\mathrm{m},-\mathrm{n}}}$$

$$\frac{\partial \mathrm{L}}{\partial \mathrm{RA1}_{0,0}} = (\mathrm{B1}_{0,0} * \frac{\partial L}{\partial FB1_{0,0}} + 0 * \frac{\partial L}{\partial FB1_{0,1}} + 0 * \frac{\partial L}{\partial FB1_{0,2}} + \cdots + 0 * \frac{\partial L}{\partial FB1_{3,3}})$$

= (0,31 *-0,243406979 + 0*0,104700796 + 0*-0,049793602 + ... + 0*0)

= -0,075456164

Visualisasi proses tersebut dapat dilihat pada Gambar 3.17 Visualisasi 1 Operasi Cross-Correlation Matriks Gradien $\frac{\partial L}{\partial RA1}$.



Gambar 3.17 Visualisasi 1 Operasi Cross-Correlation Matriks Gradien $\frac{\partial L}{\partial RA1}$

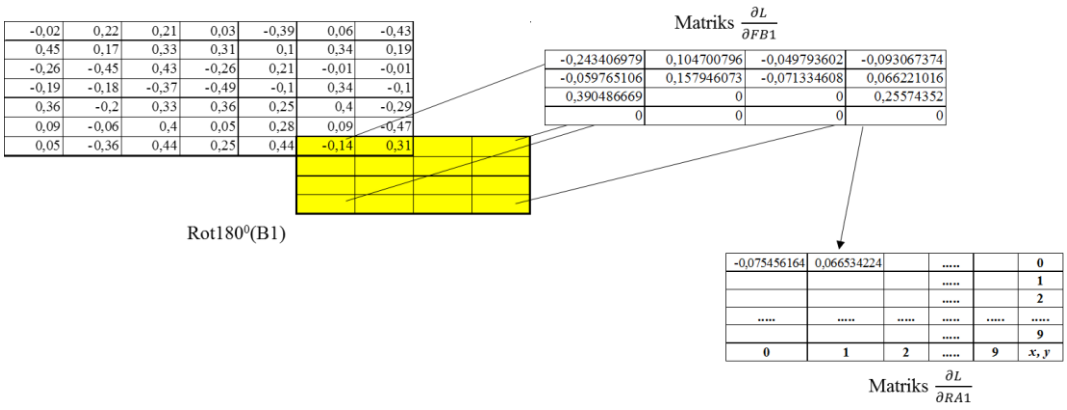
Setelah itu geser filter ke kanan dan lakukan kembali operasi konvolusi.

$$\frac{\partial \mathrm{L}}{\partial \mathrm{RA1}_{0,1}} = (\mathrm{B1}_{0,1} * \frac{\partial L}{\partial FB1_{0,0}} + \mathrm{B1}_{0,0} * \frac{\partial L}{\partial FB1_{0,1}} + 0 * \frac{\partial L}{\partial FB1_{0,2}} + \cdots + 0 * \frac{\partial L}{\partial FB1_{3,3}})$$

(-0,243406979 *0,31 + 0,104700796*-0,14 + 0*-0,049793602 + ... + 0*0)

= 0,066534224

Visualisasi dari proses tersebut dapat dilihat pada Gambar 3.18 Visualisasi 2 Operasi Cross-Correlation Matriks Gradien $\frac{\partial L}{\partial RA1}$.



Gambar 3.18 Visualisasi 2 Operasi Cross-Correlation Matriks Gradien $\frac{\partial L}{\partial RA1}$

Proses tersebut terus diulang hingga seluruh permukaan matriks. Hasil dari proses tersebut dapat dilihat pada Tabel 3.50 Matriks Gradien $\frac{\partial L}{\partial RA1}$.

Tabel 3.50 Matriks Gradien $\frac{\partial L}{\partial RA1}$

i, j	0	1	2	3	.....	9
0	-0,075456164	0,066534224	-0,137193199	-0,036663176	.....	-0,004653369
1	0,095874097	-0,013785605	-0,105850715	0,141476471	.....	-0,005065013
2	0,219728491	-0,26200767	0,198291035	0,126215595	.....	-0,014757187
3	-0,141856156	-0,127795056	0,243178785	-0,029265726	.....	0,064539284
.....	.....	.....	.....	.....	.....	.....
9	0	0	0	0	.....	0

Selanjutnya, lakukan proses yang sama antara antara $\frac{\partial L}{\partial FB2}$ dengan Filter B2, hasil dari proses tersebut dapat dilihat pada Tabel 3.51 Matriks Gradien $\frac{\partial L}{\partial RA2}$ dan antara $\frac{\partial L}{\partial FB3}$ dengan Filter B3, hasil dari proses tersebut dapat dilihat pada Tabel 3.52 Matriks Gradien $\frac{\partial L}{\partial RA3}$.

Tabel 3.51 Matriks Gradien $\frac{\partial L}{\partial RA2}$

i, j	0	1	2	3	.....	9
0	0,126488533	-0,233113958	0,118552239	0,170769044	.....	-0,017365758
1	-0,056448303	-0,036550677	0,076827245	-0,061763882	.....	-0,049610791
2	-0,130142195	0,038830061	-0,125757929	-0,118214102	.....	0,023909239
3	-0,001720085	-0,089469323	-0,088697171	0,008845875	.....	-0,042534219
.....	.....	.....	.....	.....	.....	.....
9	0	0	-0,012082139	-0,091278256	.....	0,090491402

Tabel 3.52 Matriks Gradien $\frac{\partial L}{\partial RA3}$

i, j	0	1	2	3	.....	9
0	-0,039717269	0,035304239	-0,043027041	0,028684694	.....	0
1	-0,135554932	0,089969816	-0,152091649	0,066825699	.....	0
2	0,123708657	-0,157659073	0,113166987	-0,171056044	.....	0
3	0,175738816	0,107441042	0,152374175	0,006804273	.....	0
.....	.....	.....	.....	.....	.....	.....
9	0	0	0	0	.....	0

Setelah mendapat matriks $\frac{\partial L}{\partial RA1}$, $\frac{\partial L}{\partial RA2}$ dan $\frac{\partial L}{\partial RA3}$. Selanjutnya melakukan perhitungan untuk mencari nilai $\frac{\partial L}{\partial FAi}$.

$$\frac{\partial L}{\partial FAi} = \sum_{j=1}^{3} \frac{\partial L}{\partial RAj} \frac{\partial RAi}{\partial FAi}$$

$$\frac{\partial L}{\partial FA1_{0,0}} = \frac{\partial L}{\partial RA1_{0,0}} * \frac{\partial RA1_{0,0}}{\partial FA1_{0,0}} + \frac{\partial L}{\partial RA2_{0,0}} * \frac{\partial RA1_{0,0}}{\partial FA1_{0,0}} + \frac{\partial L}{\partial RA3_{0,0}} * \frac{\partial RA1_{0,0}}{\partial FA1_{0,0}}$$

$$= \frac{\partial L}{\partial RA1_{0,0}} * \begin{cases} 1 & FA1_{0,0} \geq 0 \\ 0 & FA1_{0,0} < 0 \end{cases} + \frac{\partial L}{\partial RA2_{0,0}} * \begin{cases} 1 & FA1_{0,0} \geq 0 \\ 0 & FA1_{0,0} < 0 \end{cases} + \frac{\partial L}{\partial RA3_{0,0}} *$$

$$\begin{cases} 1 & FA1_{0,0} \geq 0 \\ 0 & FA1_{0,0} < 0 \end{cases}$$

$$= (-0{,}075456164 * 1) + (0{,}126488533 * 1) + (-0{,}039717269 * 1)$$

$$= 0{,}011315101$$

Nilai hasil keseluruhan perhitungan keseluruhan dari matriks $\frac{\partial L}{\partial FA1}$ dapat dilihat pada Tabel 3.53 Matriks $\boldsymbol{\frac{\partial L}{\partial FA1}}$.

Tabel 3.53 Matriks $\boldsymbol{\frac{\partial L}{\partial FA1}}$

i, j	**0**	**1**	**2**	**3**	**.....**	**9**
0	0,011315101	-0,131275495	-0,061668002	0,162790562	.....	-0,022019127
1	-0,096129138	0,039633534	-0,181115119	0,146538288	.....	-0,054675804
2	0,213294954	-0,380836682	0,185700093	-0,163054551	.....	0,009152051
3	0,032162576	-0,109823337	0,30685579	-0,013615578	.....	0,022005065
.....	.....	.....	.....	.....	.....	.....
9	0	0	-0,012082139	-0,091278256	.....	0,090491402

Lakukan proses yang sama untuk matriks $\frac{\partial L}{\partial RA2}$ yang menghasilkan $\frac{\partial L}{\partial FA2}$, hasilnya dapat dilihat pada Tabel 3.54 Matriks $\boldsymbol{\frac{\partial L}{\partial FA2}}$ dan matriks $\frac{\partial L}{\partial RA3}$ yang menghasilkan $\frac{\partial L}{\partial FA3}$, hasilnya dapat dilihat pada Tabel 3.55 Matriks $\boldsymbol{\frac{\partial L}{\partial FA3}}$.

Tabel 3.54 Matriks $\boldsymbol{\frac{\partial L}{\partial FA2}}$

i, j	**0**	**1**	**2**	**3**	**.....**	**9**
0	0,011315101	-0,131275495	-0,061668002	0,162790562	.....	0
1	-0,096129138	0,039633534	-0,181115119	0,146538288	.....	0
2	0,213294954	-0,380836682	0,185700093	-0,163054551	.....	0
3	0,032162576	-0,109823337	0,30685579	-0,013615578	.....	0
.....	.....	.....	.....	.....	.....	.....
9	0	0	0	0	.....	0

Tabel 3.55 Matriks $\boldsymbol{\frac{\partial L}{\partial FA3}}$

i, j	**0**	**1**	**2**	**3**	**.....**	**9**
0	0,011315101	-0,131275495	-0,061668002	0,162790562	.....	-0,022019127
1	-0,096129138	0,039633534	-0,181115119	0,146538288	.....	-0,054675804
2	0,213294954	-0,380836682	0,185700093	-0,163054551	.....	0,009152051
3	0,032162576	-0,109823337	0,30685579	-0,013615578	.....	0
.....	.....	.....	.....	.....	.....	.....
9	0	0	-0,012082139	0	.....	0

Setelah nilai dari $\frac{\partial L}{\partial FA1}, \frac{\partial L}{\partial FA2}$ dan $\frac{\partial L}{\partial FA3}$ diketahui, lakukan proses konvolusi antara $\frac{\partial L}{\partial FA1}$ dengan citra input yang dapat dilihat pada Tabel 3.5 Nilai Pixel Citra Masukan C (*Current Sate*) untuk menghasilkan gradien $\frac{\partial L}{\partial A1}$. Karena filter pada A1 bernilai *stride* 3 maka pixel-pixel pada matriks $\frac{\partial L}{\partial FA1}$ akan dilakukan pelebaran (*dilatation*) sejumlah *stride*-1 sehingga besar lebaran adalah 2 pixel diantara dua pixel bersebelahan dan diisi 0. Berikut perhitungannya.

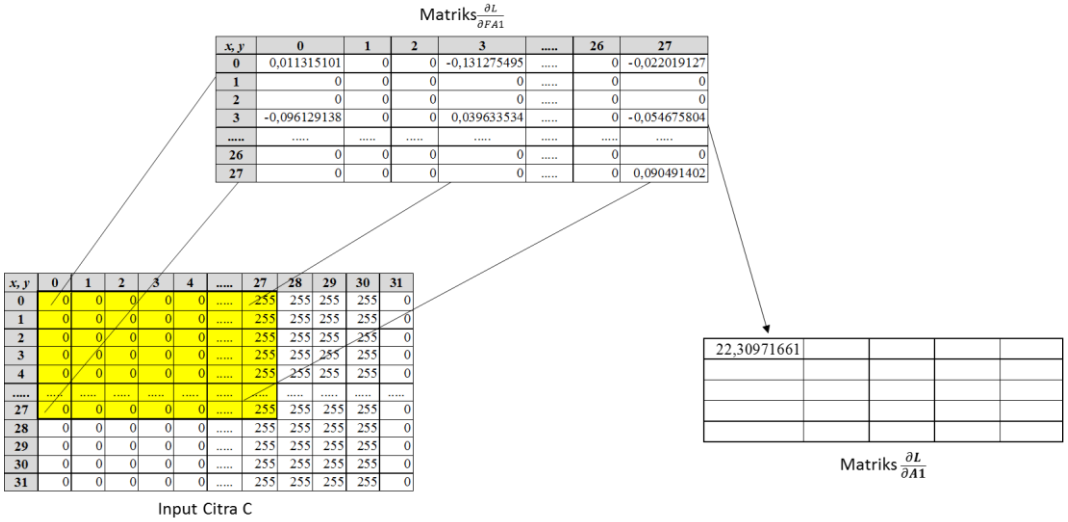
$$\frac{\partial L}{\partial A1_{i,j}} = \sum_{m=0}^{27} \sum_{n=0}^{27} \frac{\partial L}{\partial FA1_{m,n}} \cdot C_{i*1+m,j*1+n} \quad \textbf{(2.5)}$$

$$\frac{\partial L}{\partial A1_{0,0}} = \frac{\partial L}{\partial FA1_{0,0}} . C_{0,0} + \frac{\partial L}{\partial FA1_{0,1}} . C_{0,1} + \frac{\partial L}{\partial FA1_{0,2}} . C_{0,2} + \frac{\partial L}{\partial FA1_{0,3}} . C_{0,3} + \cdots + \frac{\partial L}{\partial FA1_{27,27}} . C_{27,27}$$

= (0,011315101*0 + 0*0+ 0*0 + -0,131275495*0 + ... + 0,090491402*255)

= 22,30971661

Visualisasi dari proses tersebut dapat dilihat pada Gambar 3.19 Visualisasi 1 Operasi Konvolusi Matriks Gradien $\frac{\partial L}{\partial A1}$



Gambar 3.19 Visualisasi 1 Operasi Konvolusi Matriks Gradien $\frac{\partial L}{\partial A1}$

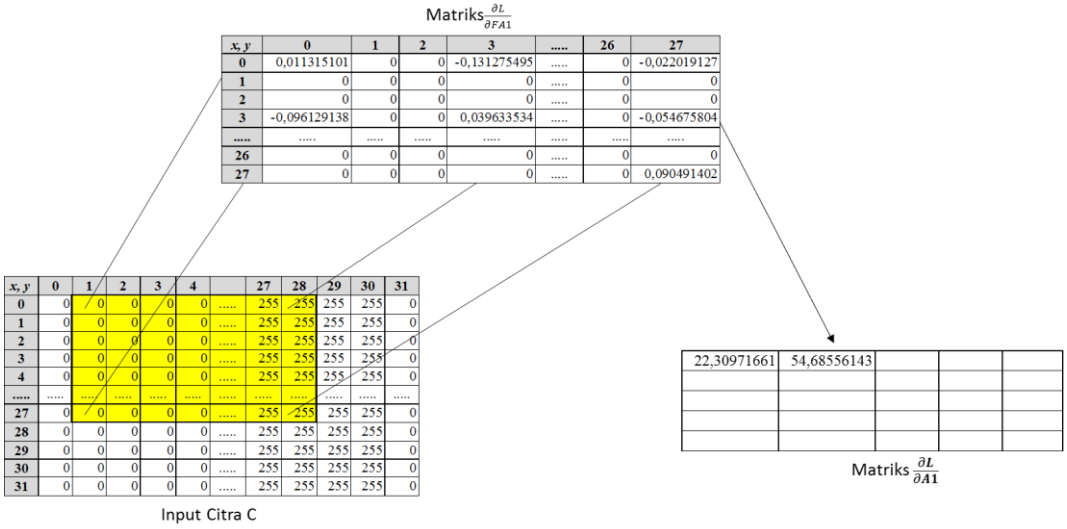
Setelah itu geser filter ke kanan dan lakukan kembali operasi konvolusi.

$$\frac{\partial L}{\partial A1_{0,1}} = \frac{\partial L}{\partial FA1_{0,0}} . C_{0,1} + \frac{\partial L}{\partial FA1_{0,1}} . C_{0,2} + \frac{\partial L}{\partial FA1_{0,2}} . C_{0,3} + \cdots + \frac{\partial L}{\partial FA1_{27,26}} . C_{27,27} + \frac{\partial L}{\partial FA1_{27,27}} . C_{27,28}$$

= (0,011315101*0 + 0*0+ 0*0 + ... + 0*255 + 0,090491402*255)

= 54,68556143

Visualisasi dari proses tersebut dapat dilihat pada Gambar 3.20 Visualisasi 2 Operasi Konvolusi Matriks Gradien $\frac{\partial L}{\partial A1}$.



Gambar 3.20 Visualisasi 2 Operasi Konvolusi Matriks Gradien $\frac{\partial L}{\partial A1}$

Proses tersebut terus diulang hingga seluruh permukaan matriks dan menghasilkan gradien $\frac{\partial L}{\partial A1}$ dapat dilihat pada Tabel 3.56 Matriks Gradien $\frac{\partial L}{\partial A1}$.

Tabel 3.56 Matriks Gradien $\frac{\partial L}{\partial A1}$

i, j	0	1	2	3	4
0	22,30971661	54,68556143	54,90446472	54,90446472	158,4784809
1	-1,12890853	41,02879994	41,24770323	41,24770323	168,2603446
2	-6,740200047	3,041663597	3,260566883	3,260566883	10,00076693
3	-33,097101	-38,8469389	0,179621311	0,179621311	122,6692937
4	-30,48933208	-158,552644	-116,4451382	-113,3641926	72,20678904

Setelah melakukan perhitungan gradien $\frac{\partial L}{\partial A1}$, lalu lakukan perhitungan bobot baru filter A1 menggunakan Adam *Oprimizer*, diawali menghitung m_1.

$$\boldsymbol{m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \frac{\partial L}{\partial A1_{0,0}}} \quad \textbf{(2.17)}$$

m_1 = 0,9 * 0 + (1 – 0,9) * 22,30971661

m_1 = 2,230971661

Selanjutnya, hitung nilai v_1 menggunakan persamaan (2.18).

$$\boldsymbol{v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * \frac{\partial L}{\partial A1_{0,0}}^2} \quad \textbf{(2.18)}$$

v_1 = 0,999 * 0 + (1 – 0,999) * 22,30971661^2

v_1 = 0,497723455

Lalu hitung nilai $\hat{m}_1$ menggunakan persamaan (2.19).

$$\boldsymbol{\hat{m}_t = m_t / (1 - \beta_1^t)} \quad \textbf{(2.19)}$$

$\hat{m}_1$ = 2,230971661 / (1 – 0,9^1)

$\hat{m}_1$ = 22,30971661

Setelah itu lakukan perhitungan nilai $\hat{v}_1$ menggunakan persamaan (2.20).

$$\boldsymbol{\hat{v}_t = v_t / \left(1 - \beta_2^t\right)} \quad \textbf{(2.20)}$$

$\hat{v}_1$ = 0,497723455 / (1 – 0,999^1)

$\hat{v}_1$ = 497,723455

Terakhir, lakukan perhitungan bobot baru menggunakan persamaan (2.21).

$$\boldsymbol{A1'_{0,0} = A1_{0,0} - \alpha * \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)} \quad \textbf{(2.21)}$$

$A1'_{0,0} = A1_{0,0} - \alpha * \hat{m}_1 / (\sqrt{\hat{v}_1} + \varepsilon)$

$A1'_{0,0}$ = 0,05 - 0,001 * 22,30971661 / ($\sqrt{497,723455}$ + 10^{-8})

$A1'_{0,0}$ = 0,049

Nilai hasil perhitungan bobot baru A1 keseluruhan dapat dilihat pada Tabel 3.57 Bobot Baru Filter A1.

Tabel 3.57 Bobot Baru Filter A1

x, y	**0**	**1**	**2**	**3**	**4**
0	0,049	0,339	0,379	0,109	0,289
1	0,211	0,269	-0,451	0,239	-0,171
2	0,411	0,219	0,359	-0,181	-0,141
3	0,231	-0,239	-0,451	-0,071	-0,031
4	0,011	-0,139	0,461	0,351	0,219

Selanjutnya, lakukan proses yang sama antara $\frac{\partial L}{\partial FA2}$ dengan citra input yang dapat dilihat pada Tabel 3.5 Nilai Pixel Citra Masukan C (*Current Sate*). Hasil dari proses tersebut adalah gradien $\frac{\partial L}{\partial A2}$ dapat dilihat pada Tabel 3.58 Matriks Gradien $\boldsymbol{\frac{\partial L}{\partial A2}}$.

Tabel 3.58 Matriks Gradien $\frac{\partial L}{\partial A2}$

i, j	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	112,6685268
4	-23,74913203	-116,6247595	-116,6247595	-116,6247595	62,20602211

Lalu lakukan perhitungan nilai bobot baru A2 menggunakan Adam *Oprimizer*, hasil perhitungan bobot baru A2 dapat dilihat pada Tabel 3.59 Bobot Baru Filter A2.

Tabel 3.59 Bobot Baru Filter A2

x, y	0	1	2	3	4
0	0,07	0,21	0,34	-0,31	-0,43
1	-0,35	0,14	-0,48	-0,27	-0,25
2	0,05	0,15	-0,11	0,05	0,16
3	-0,18	0,13	-0,45	-0,23	0,349
4	-0,269	0,291	0,391	0,221	0,149

Terakhir, lakukan proses yang sama antara $\frac{\partial L}{\partial FA3}$ dengan citra inputan. Hasil proses tersebut adalah gradien $\frac{\partial L}{\partial A3}$ yang dapat dilihat pada Tabel 3.60 Matriks Gradien $\frac{\partial L}{\partial A3}$.

Tabel 3.60 Matriks Gradien $\frac{\partial L}{\partial A3}$

i, j	0	1	2	3	4
0	16,69842509	49,07426992	49,2931732	49,2931732	158,4784809
1	-6,740200047	35,41750843	35,63641171	35,63641171	168,2603446
2	-6,740200047	3,041663597	3,260566883	3,260566883	10,00076693
3	-9,821145619	-38,8469389	0,179621311	0,179621311	122,6692937
4	-30,48933208	-158,552644	-116,4451382	-113,3641926	72,20678904

Lalu lakukan perhitungan nilai bobot baru filter A3 menggunakan Adam *Oprimizer*. Hasil perhitungan keseluruhan dapat dilihat pada Tabel 3.61 Bobot Baru Filter A3.

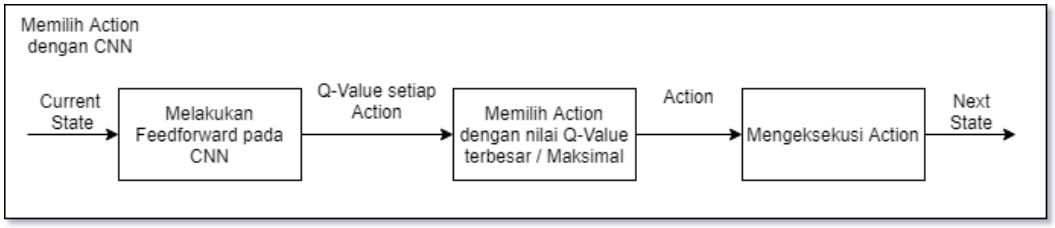
Tabel 3.61 Bobot Baru Filter A3

x, y	0	1	2	3	4
0	0,219	-0,221	0,239	-0,031	0,189
1	-0,259	-0,041	-0,411	0,249	0,359
2	-0,169	0,479	-0,311	0,299	-0,441
3	-0,459	-0,189	-0,211	0,459	0,219
4	0,361	0,391	0,341	0,021	0,469

Setelah bobot-bobot baru didapatkan, selanjutnya bobot tersebut akan digunakan untuk perhitungan *Q-value* untuk setiap *action* di suatu *state*. Detail proses pemilihan *action* akan dijelaskan pada bagian Analisis Pemilihan Action.

3.7.2. Analisis Pemilihan Action

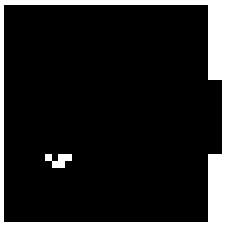
Setelah proses pelatihan CNN dan menghasilkan bobot baru, selanjutnya bobot baru tersebut dipergunakan pada tahap eksploitasi untuk menghitung *Q-value* dari setiap action yang bisa dilakukan *agent* dengan *inputan current state* dari permainan *flappy bird*. Gambaran proses pemilihan action dengan CNN dapat dilihat pada Gambar 3.21 Tahapan Proses Memilih Action dengan CNN.



Gambar 3.21 Tahapan Proses Memilih Action dengan CNN

Proses dimulai dengan *feedforward* pada CNN dengan *input current state*, *output* dari feedforward adalah nilai *Q-value* untuk setiap *action* (0 dan 1). Selanjutnya *agent* akan memutuskan *action* yang akan diambil dengan membandingkan nilai *Q-value* dari seluruh *action*, *action* dengan nilai *Q-value* terbesar akan menjadi *action* yang dipilih dan selanjutnya dieksekusi di permainan *flappy bird*.

Sebagai contoh, dilakukan pemilihan action terhadap state C yang sudah di*preprocessing*, untuk state C dapat dilihat pada Gambar 3.22 *State* C. Gambar 3.22 *State* tersebut memiliki ukuran 32 x 32 dengan nilai disetiap *pixel*-nya dapat dilihat pada Tabel 3.62 Nilai Pixel State C.



Gambar 3.22 *State* C

Tabel 3.62 Nilai Pixel State C

x, y	0	1	2	3	4	5	6	.....	30	31
0	0	0	0	0	0	0	0	.....	255	255
1	0	0	0	0	0	0	0	.....	255	255
2	0	0	0	0	0	0	0	.....	255	255
3	0	0	0	0	0	0	0	.....	255	255
4	0	0	0	0	0	0	0	.....	255	255
5	0	0	0	0	0	0	0	.....	255	255
6	0	0	0	0	0	0	0	.....	255	255
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
30	0	0	0	0	0	0	0	.....	255	255
31	0	0	0	0	0	0	0	.....	255	255

Pertama-tama Tabel 3.62 Nilai Pixel State C akan memasuki layer pertama yaitu *convolution layer* A.

1. *Convolution Layer* A

Pada *convolution layer* A akan dilakukan konvolusi antara matriks citra masukan Tabel 3.62 Nilai Pixel State C dengan matriks-matrik pada filter A. Pada layer ini terdapat 3 filter dengan ukuran 3x3 dengan *stride* 3. Proses pertama akan dilakukan konvolusi antara filter A1 yang dapat dilihat pada Tabel 3.63 *Convolution Filter* A1 dan biasnya 0.

Tabel 3.63 *Convolution Filter* A1

x, y	0	1	2	3	4
0	0,049	0,339	0,379	0,109	0,289
1	0,211	0,269	-0,451	0,239	-0,171
2	0,411	0,219	0,359	-0,181	-0,141
3	0,231	-0,239	-0,451	-0,071	-0,031
4	0,011	-0,139	0,461	0,351	0,219

Berikut adalah operasi konvolusi menggunakan persamaan (2.5) antara citra *state* C dengan filter A1 yang akan menghasilkan *feature map* FA1.

$$\boldsymbol{FA1_{i,j} = \sum_{m=0}^{4}\sum_{n=0}^{4} A1_{m,n} \cdot C_{i*3+m,j*3+n} + b} \quad \textbf{(2.5)}$$

$$FA1_{0,0} = (A1_{0,0}.C_{0,0} + A1_{0,1}.C_{0,1} + A1_{0,2}.C_{0,2} + \cdots + A1_{4,4}.C_{4,4}) + bA_0$$

=(0,049*0 + 0,339*0 + 0,379*0 + ⋯ + 0,219*0)+0

= 0

$$FA1_{0,1} = (A1_{0,0}.C_{0,3} + A1_{0,1}.C_{0,4} + A1_{0,2}.C_{0,5} + \cdots + A1_{4,4}.C_{4,7}) + bA_0$$

=(0,049*0 + 0,339*0 + 0,379*0 + ⋯ + 0,219*0)+0

= 0

Operasi tersebut akan terus diulang hingga seluruh permukaan citra *state* C. Hasil keseluruhan proses pada filter A1 dapat dilihat pada Tabel 3.64 *Feature Map* FA1.

Tabel 3.64 *Feature Map* FA1

x, y	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	0	0	0	0	0	156,06
1	0	0	0	0	0	0	0	0	0	156,06
2	0	0	0	0	0	0	0	0	0	156,06
3	0	0	0	0	0	0	0	0	0	118,83
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	89,505	209,865	2,805	0	0	0	0	0	145,35
7	0	24,99	147,135	53,805	0	0	0	0	0	54,57
8	0	0	0	0	0	0	0	0	0	156,06
9	0	0	0	0	0	0	0	0	0	156,06

Langkah selanjutnya adalah menjalankan fungsi aktivasi pada di *feature map* FA menggunakan fungsi aktivasi ReLU dan menghasilkan *feature map* RA1. Berikut adalah persamaannya (2.7).

$$f(x) = max(0, x) \quad (2.7)$$

$RA1_{0,0} = max(0, FA1_{0,0})$

$= max(0, 0)$

$= 0$

$RA1_{0,1} = max(0, FA1_{0,1})$

$= max(0, 0)$

$= 0$

Hasil dari fungsi aktivasi dapat dilihat pada Tabel 3.65 *Feature Map* RA1.

Tabel 3.65 *Feature Map* RA1

x, y	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	0	0	0	0	0	156,06
1	0	0	0	0	0	0	0	0	0	156,06
2	0	0	0	0	0	0	0	0	0	156,06
3	0	0	0	0	0	0	0	0	0	118,83
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	89,505	209,865	2,805	0	0	0	0	0	145,35
7	0	24,99	147,135	53,805	0	0	0	0	0	54,57
8	0	0	0	0	0	0	0	0	0	156,06
9	0	0	0	0	0	0	0	0	0	156,06

Setelah proses pada filter A1 selesai, lakukan proses konvolusi terhadap *state* C dengan filter A2. Bobot pada filter A2 dapat dilihat pada Tabel 3.66 *Convolution Filter* A2.

Tabel 3.66 *Convolution Filter* A2

x, y	0	1	2	3	4
0	0,07	0,21	0,34	-0,31	-0,43
1	-0,35	0,14	-0,48	-0,27	-0,25
2	0,05	0,15	-0,11	0,05	0,16
3	-0,18	0,13	-0,45	-0,23	0,349
4	-0,269	0,291	0,391	0,221	0,149

Proses ini akan menghasilkan *feature map* FA2. Hasil keseluruhan proses pada filter A2 dapat dilihat pada Tabel 3.67 *Feature Map* FA2.

Tabel 3.67 *Feature Map* FA2

x, y	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	-143,055
1	0	0	0	0	0	0	0	0	0	-143,055
2	0	0	0	0	0	0	0	0	0	-143,055
3	0	0	0	0	0	0	0	0	0	-321,3
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	56,355	87,465	-68,595	0	0	0	0	0	94,35
7	0	-28,05	-270,3	-89,25	0	0	0	0	0	45,645
8	0	0	0	0	0	0	0	0	0	-143,055
9	0	0	0	0	0	0	0	0	0	-143,055

Selanjutnya jalankan fungsi aktivasi ReLU pada *feature map* FA2 dan menghasilkan *feature map* RA2 dapat dilihat pada Tabel 3.68 *Feature Map* RA2.

Tabel 3.68 *Feature Map* RA2

x, y	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	56,355	87,465	0	0	0	0	0	0	94,35
7	0	0	0	0	0	0	0	0	0	45,645
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Terakhir, lakukan proses konvolusi terhadap *state* C dengan filter A3. Bobot pada filter A3 dapat dilihat pada Tabel 3.69 *Convolution Filter* A3.

Tabel 3.69 *Convolution Filter* A3

x, y	0	1	2	3	4
0	0,219	-0,221	0,239	-0,031	0,189
1	-0,259	-0,041	-0,411	0,249	0,359
2	-0,169	0,479	-0,311	0,299	-0,441
3	-0,459	-0,189	-0,211	0,459	0,219
4	0,361	0,391	0,341	0,021	0,469

Proses ini akan menghasilkan *feature map* FA3. Hasil keseluruhan proses pada filter A2 dapat dilihat pada Tabel 3.70 *Feature Map* FA3.

Tabel 3.70 *Feature Map* FA3

x, y	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	456,96
1	0	0	0	0	0	0	0	0	0	456,96
2	0	0	0	0	0	0	0	0	0	456,96
3	0	0	0	0	0	0	0	0	0	195,33
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	5,355	184,365	92,055	0	0	0	0	0	124,95
7	0	-48,96	-64,515	-66,045	0	0	0	0	0	416,67
8	0	0	0	0	0	0	0	0	0	456,96
9	0	0	0	0	0	0	0	0	0	456,96

Lalu jalankan fungsi aktivasi ReLU pada *feature map* FA3 dan menghasilkan *feature map* RA3 yang dapat dilihat pada Tabel 3.71 *Feature Map* RA3.

Tabel 3.71 *Feature Map* RA3

x, y	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	456,96
1	0	0	0	0	0	0	0	0	0	456,96
2	0	0	0	0	0	0	0	0	0	456,96
3	0	0	0	0	0	0	0	0	0	195,33
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	5,355	184,365	92,055	0	0	0	0	0	124,95
7	0	0	0	0	0	0	0	0	0	416,67
8	0	0	0	0	0	0	0	0	0	456,96
9	0	0	0	0	0	0	0	0	0	456,96

Hasil dari *convolutional layer* A, yaitu Feature Map RA1, RA2 dan RA3 akan digunakan pada layer selanjutnya, yaitu *convolutional layer* B.

2. *Convolution Layer* B

Pada convolutional layer B akan dilakukan proses yang sama seperti pada convolutinal layer A hanya saja inputannya rangkap tiga, yaitu feature map RA1,

RA2 dan RA3. Pada convolutional layer B terdapat 3 filter berukuran 7x7 dengan stride 1. Pertama-tama lakukan proses konvolusi antara *feature map* RA1, RA2 dan RA3 dengan bobot pada filter B1. Bobot pada filter B1 dapat dilihat pada Tabel 3.72 *Convoliton Filter* B1 dan biasnya 0.

Tabel 3.72 *Convoliton Filter* B1

x, y	0	1	2	3	4	5	6
0	0,31	-0,14	0,44	0,25	0,439	-0,361	0,049
1	-0,47	0,09	0,28	0,05	0,399	-0,061	0,089
2	-0,29	0,4	0,25	0,36	0,331	-0,199	0,361
3	-0,1	0,34	-0,1	-0,49	-0,369	-0,179	-0,189
4	-0,01	-0,01	0,21	-0,26	0,429	-0,451	-0,261
5	0,19	0,34	0,1	0,31	0,329	0,169	0,449
6	-0,43	0,06	-0,39	0,03	0,211	0,219	-0,021

Berikut adalah operasi konvolusi menggunakan persamaan (2.5) antara *feature map* RA1, RA2 dan RA3 dengan filter B1 akan menghasilkan *feature map* FB1.

$$\boldsymbol{FB1_{i,j}} = \sum_{C=1}^{3} \sum_{m=0}^{6} \sum_{n=0}^{6} \boldsymbol{B1_{m,n}} \cdot \boldsymbol{RAc_{i*1+m,j*1+n}} + \boldsymbol{b} \qquad \textbf{(2.5)}$$

$$FB1_{0,0} = (B1_{0,0}.RA1_{0,0} + B1_{0,1}.RA1_{0,1} + B3_{0,2}.RA1_{0,2} + \cdots + B1_{6,6}.RA3_{6,6}) + bB_0$$

= (0,31*0 + -0,14*0 + 0,44*0 + ... + -0,021*0) + 0

= -175,37625

$$FB1_{0,1} = (B1_{0,0}.RA1_{0,1} + B1_{0,1}.RA1_{0,2} + B3_{0,2}.RA1_{0,3} + \cdots + B1_{6,6}.RA3_{6,7}) + bB_0$$

= (0,31*0 + -0,14*0 + 0,44*0 + ... + -0,021*0) + 0

= -72,81015

Operasi tersebut akan terus diulang hingga seluruh permukaan *feature map* RA1, RA2 dan RA3. Hasil keseluruhan dapat dilihat pada Tabel 3.73 *Feature Map* FB1

Tabel 3.73 *Feature Map* FB1

x, y	0	1	2	3
0	-175,94235	-73,11615	-201,43725	198,07329
1	74,7201	179,0916	63,7347	345,769035
2	114,86985	73,746	40,4838	191,306865
3	-26,57865	148,7466	-17,9265	63,916515

Langkah selanjutnya adalah menjalankan fungsi aktivasi ReLU pada di *feature map* FB dan menghasilkan *feature map* RB1. Berikut adalah persamaannya (2.7).

$$f(x) = max(0, x) \quad (2.7)$$

$RB1_{0,0} = max(0, FB1_{0,0})$

$= max(0, -175{,}37625)$

$= 0$

$RB1_{0,0} = max(0, FB1_{0,0})$

$= max(0, -72{,}81015)$

$= 0$

Hasil dari fungsi aktivasi dapat dilihat pada Tabel 3.74 *Feature Map* RB1.

Tabel 3.74 *Feature Map* RB1

x, y	0	1	2	3
0	0	0	0	198,07329
1	74,7201	179,0916	63,7347	345,769035
2	114,86985	73,746	40,4838	191,306865
3	0	148,7466	0	63,916515

Setelah proses pada filter B1 selesai, lalu lakukan proses konvolusi antara *feature map* RA1, RA2 dan RA3 dengan bobot pada filter B2. Bobot pada filter B2 dapat dilihat pada Tabel 3.75 *Convoliton Filter* B2.

Tabel 3.75 *Convoliton Filter* B2

x, y	0	1	2	3	4	5	6
0	-0,39	0,41	-0,16	-0,18	0,399	0,409	0,061
1	0,33	0,21	0,12	0,34	-0,001	0,071	0,131
2	0,46	-0,04	0,4	-0,2	-0,439	0,361	-0,119
3	-0,34	-0,08	-0,11	-0,16	0,101	-0,139	0,399
4	-0,39	0,26	-0,22	0,1	-0,051	-0,171	-0,391
5	0,14	-0,16	-0,22	0,26	0,401	0,419	-0,131
6	-0,261	-0,431	0,12	0,4	0,011	0,219	0,329

Hasil dari proses tersebut akan menghasilkan *feature map* FB2 yang dapat dilihat pada Tabel 3.76 *Feature Map* FB2.

Tabel 3.76 *Feature Map* FB2

x, y	0	1	2	3
0	30,573735	-235,69446	-166,607055	265,31169
1	-77,09619	-140,251275	-9,33249	201,83811
2	-79,5498	13,51755	-151,20735	40,479975
3	-100,75305	-83,7114	-214,75845	30,698685

Lalu jalankan fungsi aktivasi ReLU pada *feature map* FB2 dan menghasilkan *feature map* RB2 yang dapat dilihat pada Tabel 3.77 *Feature Map* RB2.

Tabel 3.77 *Feature Map* RB2

x, y	0	1	2	3
0	30,573735	0	0	265,31169
1	0	0	0	201,83811
2	0	13,51755	0	40,479975
3	0	0	0	30,698685

Terakhirlakukan proses konvolusi antara *feature map* RA1, RA2 dan RA3 dengan bobot pada filter B3. Bobot pada filter B3 dapat dilihat pada Tabel 3.78 *Convolution Filter* B3.

Tabel 3.78 *Convolution Filter* B3

x, y	0	1	2	3	4	5	6
0	0,36	-0,32	0,39	-0,26	-0,311	-0,001	-0,189
1	0,03	0,25	0,08	0,26	0,169	0,101	-0,409
2	-0,37	-0,16	-0,37	0,07	0,061	0,411	-0,401
3	-0,29	0,15	0,04	0,32	-0,199	-0,149	-0,301
4	0,13	0,21	0,37	-0,17	-0,491	-0,441	0,339
5	0,49	-0,48	-0,11	-0,03	0,409	0,289	-0,431
6	0,46	-0,24	0,03	0,29	0,401	0,429	-0,479

Hasil dari proses tersebut akan menghasilkan *feature map* FB2 yang dapat dilihat pada Tabel 3.76 *Feature Map* FB2 Tabel 3.79 *Feature Map* FB3.

Tabel 3.79 *Feature Map* FB3

x, y	0	1	2	3
0	20,04045	-34,09095	102,9486	-577,99677
1	-62,28375	-110,36655	89,0103	-622,449645
2	97,88685	99,16695	54,66945	-498,087165
3	49,5873	26,5608	-65,02245	-388,731435

Lalu jalankan fungsi aktivasi ReLU pada *feature map* FB3 dan menghasilkan *feature map* RB3 yang dapat dilihat pada Tabel 3.80 *Feature Map* RB3.

Tabel 3.80 *Feature Map* RB3

x, y	**0**	**1**	**2**	**3**
0	20,04045	0	102,9486	0
1	0	0	89,0103	0
2	97,88685	99,16695	54,66945	0
3	49,5873	26,5608	0	0

Hasil proses dari *convolutional layer* B, yaitu *Feature Map* RB1, RB2 dan RB3 akan digunakan pada layer selanjutnya, yaitu *fully-connected layer*. *Fully-connected layer* adalah sebuah *neural network* yang memiliki *input layer, hidden layer dan output layer*. Berikut adalah operasi-operasi yang terjadi di masing-masing layer.

3. *Input Layer* (*Flatten*)

Pada input layer akan terjadi penggabungan dari matriks *feature map* RB1, RB2 dan RB3. Matriks akan diratakan menjadi sebuah vektor dengan panjang sejumlah keseluruhan pixel dari RB1, RB2 dan RB3. Total pixel dari ketiga *feature map* adalah 48, sehingga *input layer* X memiliki 48 node. Vector *input layer* X dapat dilihat pada Tabel 3.81 *Input Layer X*.

Tabel 3.81 *Input Layer X*

i	***X***	***i***	***X***	***i***	***X***
0	0	**16**	30,5737	**32**	20,0404
1	0	**17**	0	**33**	0
2	0	**18**	0	**34**	102,949
3	198,073	**19**	265,312	**35**	0
4	74,7201	**20**	0	**36**	0
5	179,092	**21**	0	**37**	0
6	63,7347	**22**	0	**38**	89,0103
7	345,769	**23**	201,838	**39**	0
8	114,87	**24**	0	**40**	97,8868
9	73,746	**25**	13,5176	**41**	99,1669
10	40,4838	**26**	0	**42**	54,6694
11	191,307	**27**	40,48	**43**	0
12	0	**28**	0	**44**	49,5873
13	148,747	**29**	0	**45**	26,5608
14	0	**30**	0	**46**	0
15	63,9165	**31**	30,6987	**47**	0

Selanjutnya nilai dari *input layer* X akan digunakan pada *hidden layer* Z.

4. *Hidden Layer*

Pada hidden layer, setiap node Xj akan dilakukan operasi menggunakan persamaan (2.9). Matriks nilai bobot V dapat dilihat pada Tabel 3.82 Bobot *Hidden Layer V* dan untuk nilai biasnya semua bernilai 0

Tabel 3.82 Bobot *Hidden Layer V*

x, y	0	1	2	3	4	5	6	7	8	9
0	-0,14	0,13	0,15	-0,33	0,35	-0,34	-0,41	0,35	0,19	-0,26
1	-0,479	0,249	-0,28	0,349	0,381	0,39	-0,299	-0,409	-0,35	-0,061
2	-0,299	-0,421	0,47	0,069	0,211	0,25	-0,279	0,161	0,09	0,349
3	0,041	0,359	-0,02	0,469	0,151	0,16	0,261	0,451	-0,42	-0,121
4	0,24	0,2	0,34	-0,12	0,37	0,44	0,04	-0,16	-0,21	0,18
5	-0,059	0,479	-0,19	0,489	-0,079	0,36	0,231	-0,189	0,19	-0,231
.....	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**	**.....**
44	0,24	0,21	-0,48	-0,17	-0,06	-0,41	0,3	0,3	-0,44	0,05
45	0,1	-0,19	-0,49	-0,42	0,27	-0,38	-0,08	0,14	0,01	-0,13
46	0,13	-0,31	-0,3	0,42	-0,08	-0,35	-0,1	0,29	0,19	-0,25
47	0,29	-0,08	0,12	-0,35	0,24	-0,15	0	0,49	-0,19	-0,42

Berikut adalah contoh perhitungannya menggunakan persamaan (2.9).

$$\boldsymbol{z\_in_i} = \sum_{j=0}^{n=47} \boldsymbol{X_j} * \boldsymbol{V_{j,i}} + \boldsymbol{bV_i} \quad \textbf{(2.9)}$$

$$z\_in_0 = \sum_{j=0}^{n=47} X_j * V_{j,0} + bV_0$$

$$= (0*\text{-}0{,}14 + 0*\text{-}0{,}479 + 0*\text{-}0{,}299 + \ldots + 0*0{,}29) + 0$$

$$= 141{,}6030275$$

Hasil keseluruhan perhitungan *Z_in* dapat dilihat pada Tabel 3.83 Nilai *Z_in*.

Tabel 3.83 Nilai *Z_in*

i	*z_in*
0	141,6030275
1	505,9909181
2	-41,4783612
3	91,0151406
4	-12,71595495
5	115,2726684
6	269,8081356
7	407,033193
8	-199,6155759
9	18,3301038

Langkah selanjutnya adalah menjalankan fungsi aktivasi pada setiap nilai-nilai di *z_in* menggunakan fungsi aktivasi ReLU dan menghasilkan nilai Z. Berikut adalah contoh perhitungannya.

$Z = max(0, z\_in_0)$

$= max(0, 141{,}6030275)$

$= 141{,}6030275$

Proses tersebut menghasilkan nilai Z yang dapat dilihat pada Tabel 3.84 Nilai *Z*.

Tabel 3.84 Nilai *Z*

i	*Z*
0	141,6030275
1	505,9909181
2	0
3	91,0151406
4	0
5	115,2726684
6	269,8081356
7	407,033193
8	0
9	18,3301038

5. Output Layer

Selanjutnya, hasil perhitungan pada *hidden layer Z* akan digunakan untuk menghitung nilai *output Y*. Matriks bobot *V* dapat dilihat pada Tabel 3.85 Bobot *Output Layer W*.

Tabel 3.85 Bobot *Output Layer W*

x, y	**0**	**1**
0	0,211	0,27
1	-0,349	-0,09
2	-0,09	0,27
3	-0,149	-0,21
4	0,611	0,32
5	0,32	-0,19
6	0,111	0,37
7	0,361	-0,16
8	0,35	0,34
9	-0,429	0,04

Berikut adalah contoh perhitungannya

$$y\_in_i = \sum_{j=0}^{n=9} Z_j * W_{j,i} + bV_i \quad \textbf{(2.9)}$$

y_in_0 = $Z_0*W_{0,0} + Z_1*W_{0,1} + Z_2*W_{0,2} + Z_3*W_{0,3} + \ldots. + Z_9*W_{0,9}$

= (141,6030275*0,211 + 505,9909181*-0,339 + 0*-0,09 + ... + 18,3301038*-0,429) + 0

= 45,63747753

Hasil perhitungan keseluruhan *y_in* dapat dilihat pada Tabel 3.86 Nilai *y_in*.

Tabel 3.86 Nilai *y_in*

i	*y_in*
0	45,63747753
1	-12,88444829

Selanjutnya, aktifkan fungsi aktifasi untuk setiap nilai dengan fungsi aktifasi linear dengan persamaan (2.7). berikut adalah contoh perhitungannya.

$Y_0 = y\_in_0$

Y_0 = 45,63747753

Hasil perhitungan Y dapat dilihat pada Tabel 3.87 Nilai *Y*.

Tabel 3.87 Nilai *Y*

i	*Y*
0	45,63747753
1	-12,88444829

Nilai *Y* merupakan nilai output pada CNN yang akan menjadi *Q-value* untuk setiap *action.* Nilai Y_0 akan menjadi *Q-value* dari *action* 0 dan Nilai Y_1 akan menjadi *Q-value* dari *action* 1. Pengambilan *action* akan dilihat dari nilai *Q-value*, *action* dengan nilai *Q-value* terbesar akan menjadi *action* yang dipilih, untuk state Gambar 3.22 *State* C nilai *Q-value* terbesar adalah *action* 0 maka *action* 0 menjadi *action* yang dipilih oleh *agent.*

3.8 Analisis Kebutuhan Non-fungsional

Kebutuhan non-fungsional meliputi kebutuhan *hardware* dan kebutuhan *software* minimal agar implementasi algoritma pada penelitian ini dapat berjalan dengan baik.

3.8.1. Analisis Kebutuhan *Hardware*

Spesifikasi *Hardware* yang dibutuhkan pada penelitian ini dapat dilihat pada Tabel 3.88 Spesifikasi Kebutuhan *Hardware*.

Tabel 3.88 Spesifikasi Kebutuhan *Hardware*

Hardware	**Spesifikasi**
Processor	*Dual Core* atau lebih tinggi
RAM	4 GB atau lebih tinggi
Display	Resolusi 1024x768 atau lebih tinggi
Keyboard dan *Mouse*	*Standard*

3.8.2. Analisis Kebutuhan *Software*

Spesifikasi dari *software* yang dibutuhkan pada penelitian ini dapat dilihat pada Tabel 3.89 Spesifikasi Kebutuhan *Software*.

Tabel 3.89 Spesifikasi Kebutuhan *Software*

Software	**Spesifikasi**
Operating System	*Windows* 8
Development Kit	Python 3 keatas, TensorFlow 2 keatas, Keras 2.0 keatas, PyQt5, OpenCV 4 dan PyGame

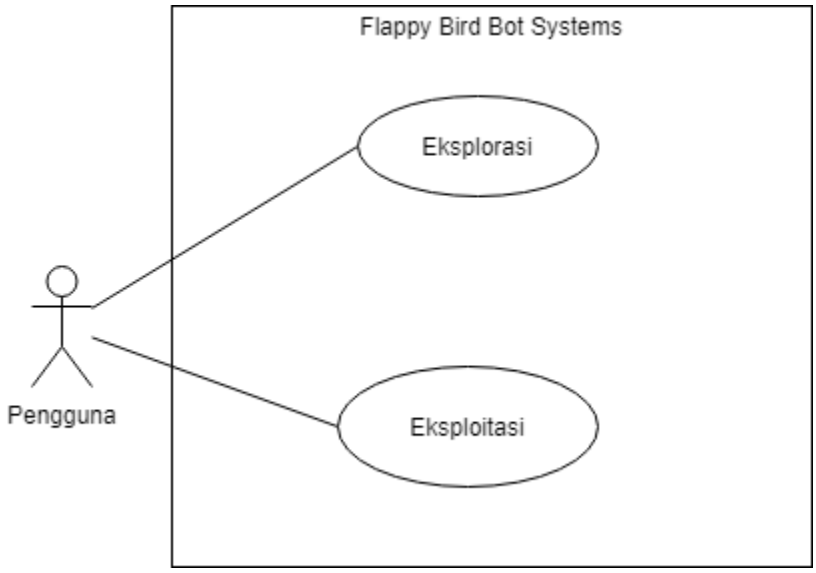
3.9 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional berisi analisis proses yang akan diterapkan di dalam sistem dan memberikan gambaran mengenai rencana desain dari sistem agar menggunakan UML (*Unified Modeling Language*) agar lebih mudah dipahami. Kebutuhan fungsional meliputi beberapa diagram UML, antara lain *use case diagram, activity diagram, class diagram* dan *sequence diagram*

.

3.9.1. *Use Case Diagram*

Use case diagram adalah diagram yang digunakan untuk menggambarkan unit fungsionalitas yang tersedia pada sistem. Adapun *use case diagram* pada sistem di penelitian ini dapat dilihat pada Gambar 3.23 *Use Case Diagram*.



Gambar 3.23 *Use Case Diagram*

3.9.1.1. Definisi Aktor

Definisi aktor berfungsi untuk memberi penjelasan mengenai aktor yang terdapat pada *use case diagram*. Adapun aktor yang terdapat pada *use case diagram* dapat dilihat pada Tabel 3.90 Definisi Aktor.

Tabel 3.90 Definisi Aktor

No	Aktor	Definisi
1	Pengguna	Pengguna yang dapat melakukan proses eksplorasi dan proses eksploitasi.

3.9.1.2. Definisi *Use Case*

Definisi *use case* berfungsi untuk memberikan penjelasan mengenai fungsi dari *use case* yang terdapat pada *use case diagram.* Adapun definisi *use case* yang terdapat pada *use case* diagram yang digunakan dapat dilihat pada Tabel 3.91 Definisi Use Case.

Tabel 3.91 Definisi Use Case

No	Use Case	Definisi
1	Eksplorasi	Fungsionalitas yang digunakan oleh pengguna untuk melakukan proses eksplorasi.
2	Eksploitasi	Fungsionalitas yang digunakan oleh pengguna untuk melakukan proses eksploitasi.

3.9.1.3. Skenario *Use Case*

Skenario *use case* adalah skenario dari setiap bagian pada *use case* yang menunjukkan proses apa saja yang terjadi pada setiap bagian di dalam *use case* tersebut. Adapun tabel-tabel yang menunjukkan skenario dari *use case,* antara lain: Skenario dari *use case* Eksplorasi dapat dilihat pada Tabel 3.92 Skenario Eksplorasi.

Tabel 3.92 Skenario Eksplorasi

Use Case Name	Eksplorasi	
Related Reqirement	*None*	
Goal	Melakukan Proses Eksplorasi	
Precondition	*None*	
Success End Condition	Berhasil Melakukan Proses Eksplorasi	
Failed End Condition	Gagal Melakukan Proses Eksplorasi	
Primary Actor	*Pengguna*	
Main Flow	***Step***	***Action***
	1	Pengguna melakukan pengaturan parameter algoritma dan menekan tombol Eksplorasi.
	2	Sistem menampilkan permainan *flappy bird*.

	3	Sistem mendapat *current state* dari permainan.
	4	Sistem melakukan Pemotongan *current state*
	5	Sistem melakukan *Grayscaling current state*
	6	Sistem melakukan *Resize current state*
	7	Sistem melakukan *Thresholding current state*
	8	Sistem memilih *action*
	9	Sistem mengeksekusi *action* di permainan
	10	Sistem mendapat *reward, next state* dan *terminal* dari permainan
	11	Sistem melakukan Pemotongan *next state*
	12	Sistem melakukan *Grayscaling next state*
	13	Sistem melakukan *Resize next state*
	14	Sistem melakukan *Thresholding next state*
	15	Sistem menyimpan *current state, action, reward, next state* dan *terminal* ke *reply memory*
	16	Sistem mengambil data latih secara acak dari reply memory
	17	Sistem Melatih model CNN dengan data latih yang telah diambil
Extention	***Step***	***Branching Action***
	2.1	Sistem menampilkan pesan kesalahan proses eksplorasi gagal dilakukan

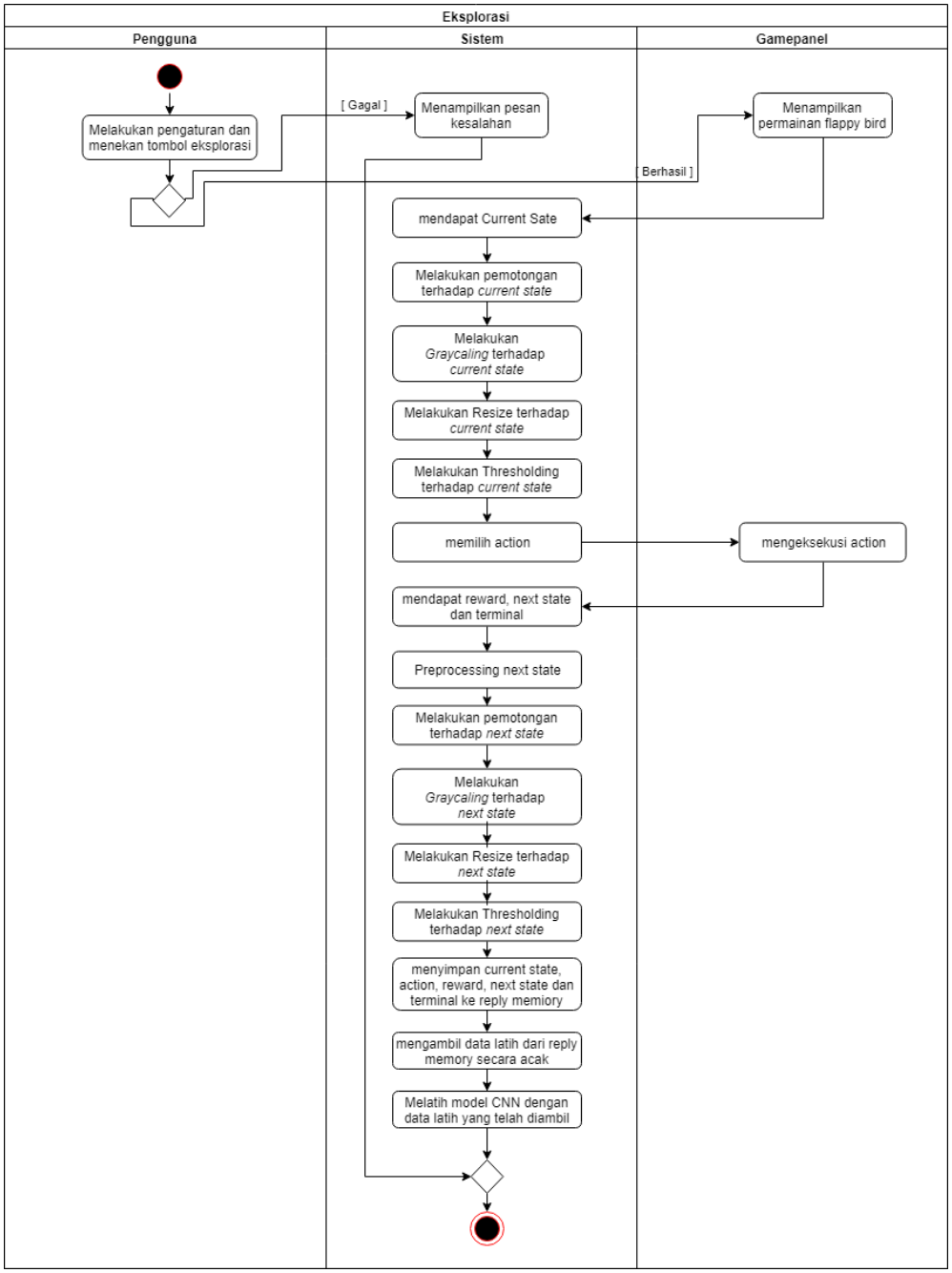
Skenario dari *use case* eksploitasi dapat dilihat pada Tabel 3.93 Skenario Eksplotasi.

Tabel 3.93 Skenario Eksplotasi

Use Case Name	Eksploitasi	
Related Reqirement	*None*	
Goal	Melakukan Proses Eksploitasi	
Precondition	*None*	
Success End Condition	Berhasil Melakukan Proses Eksploitasi	
Failed End Condition	Gagal Melakukan Proses Eksplorasi	
Primary Actor	Pengguna	
Main Flow	***Step***	***Action***
	1	Pengguna melakukan pengaturan parameter algoritma dan menekan tombol Eksplorasi.
	2	Sistem menampilkan permainan *flappy bird*.
	3	Sistem mendapat *current state* dari permainan.
	4	Sistem melakukan Pemotongan *current state*
	5	Sistem melakukan *Grayscaling current state*
	6	Sistem melakukan *Resize current state*
	7	Sistem melakukan *Thresholding current state*
	8	Sistem memilih action dengan CNN
	9	Sistem mengeksekusi *action* di permainan
Extention	***Step***	***Branching Action***
	2.1	Pengguna gagal melakukan proses eksplorasi

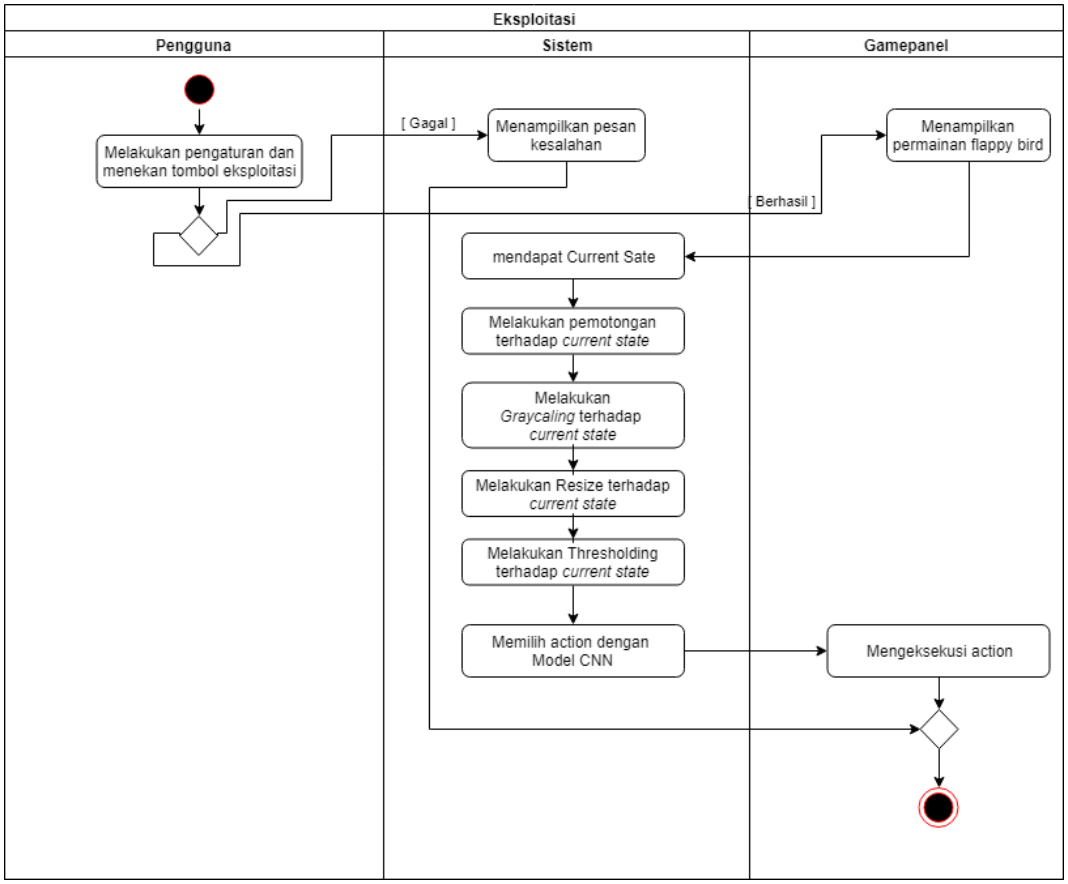
3.9.2. *Activity Diagram*

Activity diagram adalah diagram yang menunjukkan *flow control* dari satu atau lebih objek dalam memproses suatu kegiatan. Berikut adalah *activity diagram* untuk setiap *use case* pada *use case* diagram. Adapun *activity diagram* dari *use case* eksplorasi dapat dilihat Gambar 3.24 *Activity Diagram* Eksplorasi.



Gambar 3.24 *Activity Diagram* Eksplorasi

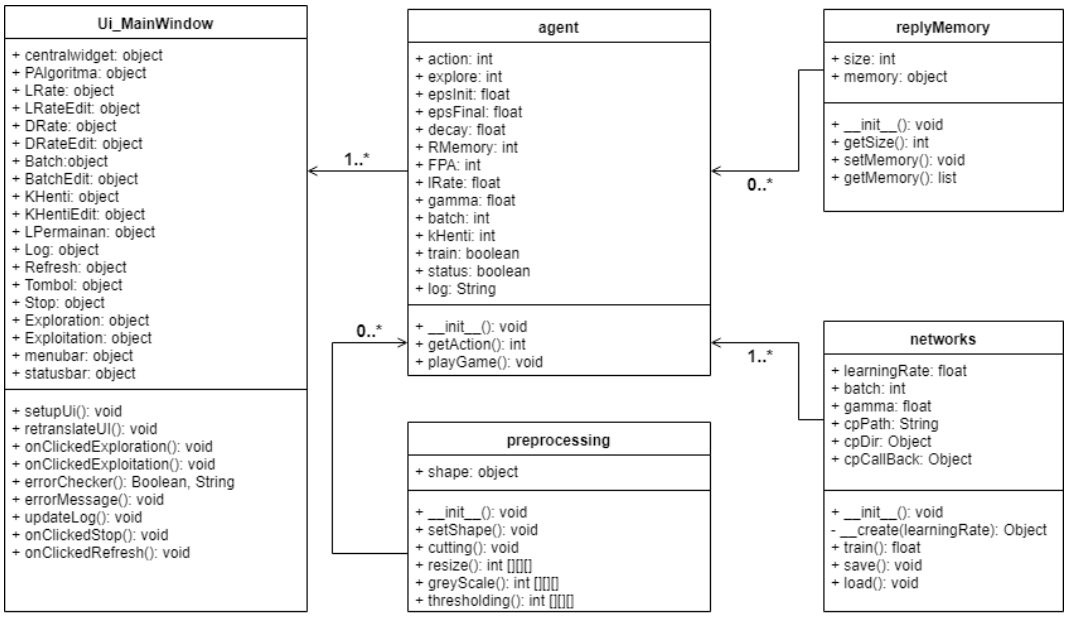
Sedangkan *activity diagram* yang menunjukkan *use case* eksploitasi dapat dilihat pada Gambar 3.25 *Activity Diagram* Eksploitasi.



Gambar 3.25 *Activity Diagram* Eksploitasi

3.9.3. *Class Diagram*

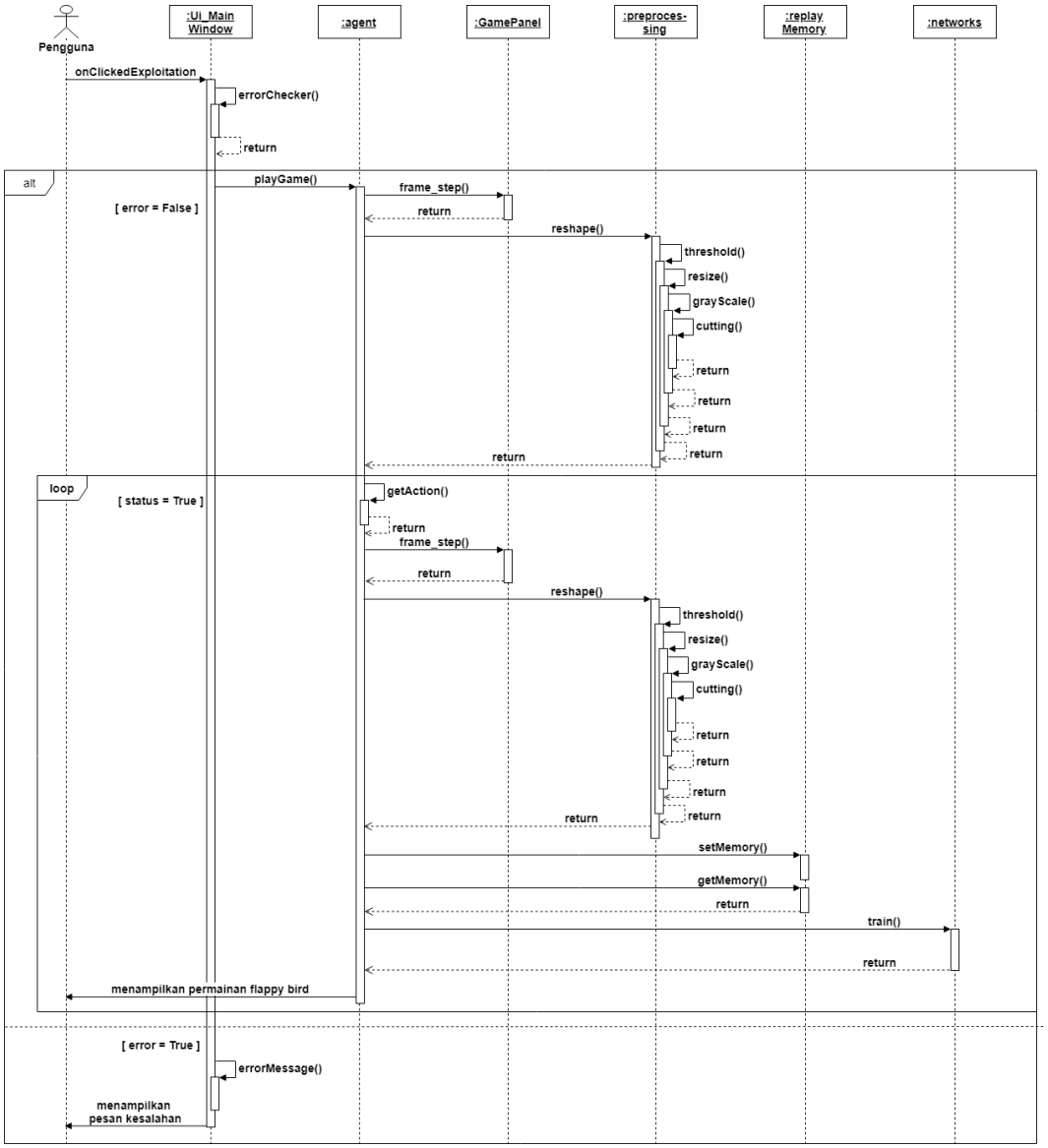
Class diagram menunjukkan suatu hubungan antara satu entitas dengan entitas lainnya. Berikut adalah *class digaram* pada sistem di penelitian ini dapat dilihat pada Gambar 3.26 *Class Diagr*.



Gambar 3.26 *Class Diagram*

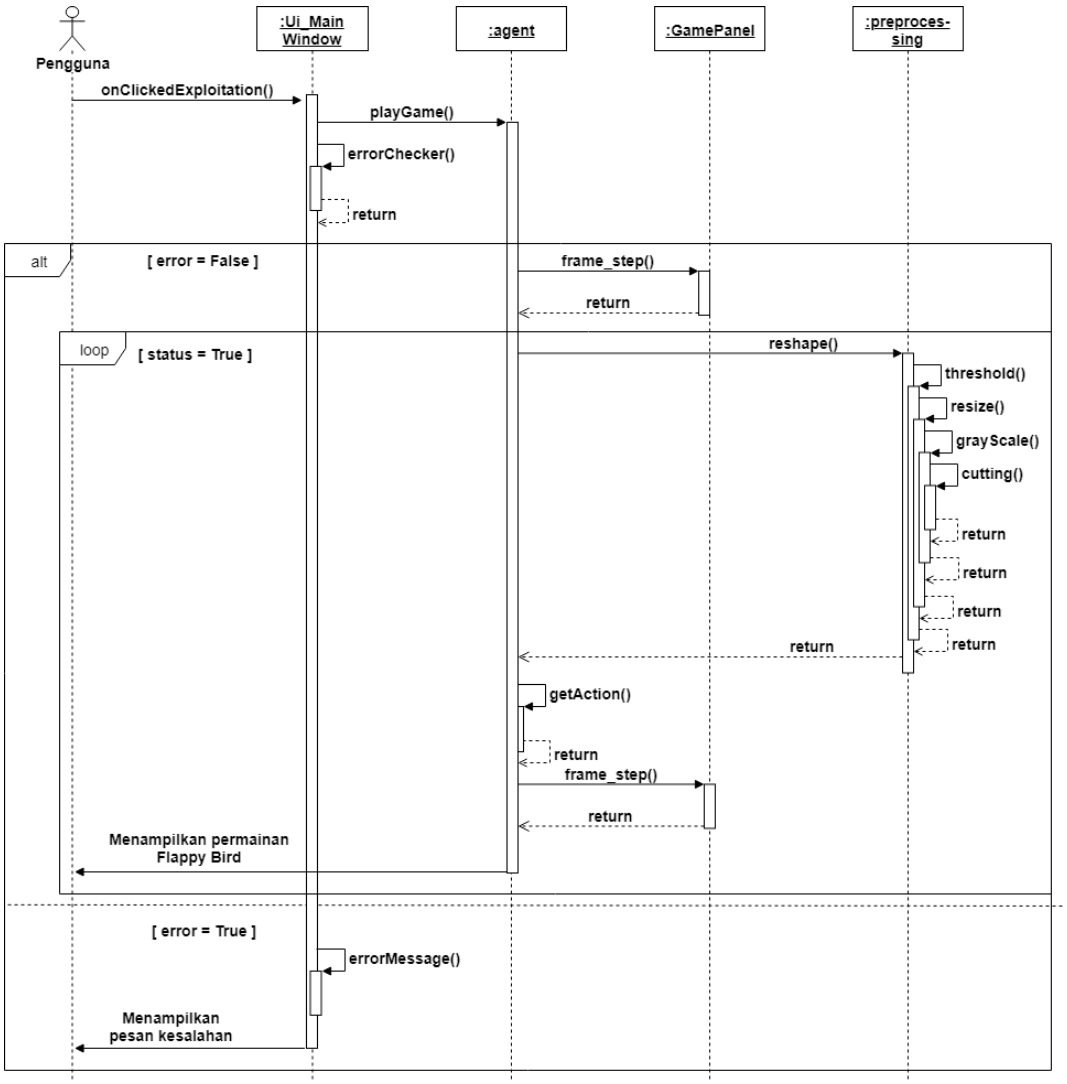
3.9.4. *Sequence Diagram*

Sequence diagram memberi aliran rinci antar objek untuk *use case* tertentu atau hanya bagian dari *use case* tertentu. *Sequence diagram* untuk *use case* eksplorassi dapat dilihat pada Gambar 3.27 *Sequence Diagram* Eksplorasi.



Gambar 3.27 *Sequence Diagram* Eksplorasi

Sedangkan untuk *Sequence diagram* yang menunjukkan *use case* eksploitasi dapat dilihat pada Gambar 3.28 *Sequence Diagram* Eksploitasi.



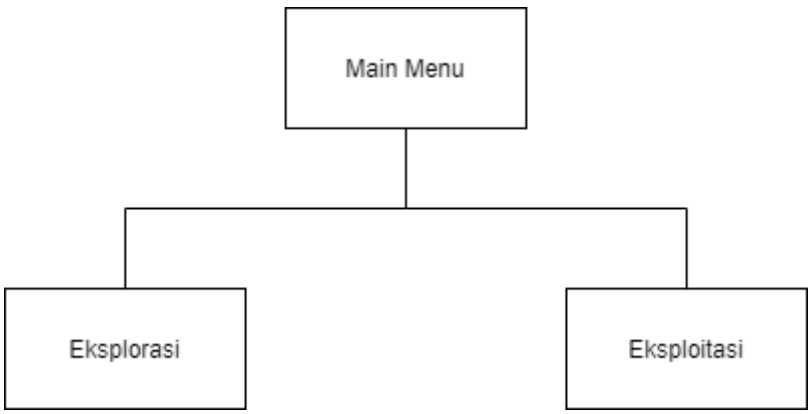
Gambar 3.28 *Sequence Diagram* Eksploitasi

3.10 Perancangan Sistem

Perancangan sistem memberikan gambaran mengenai sistem yang akan dibangun. Dalam perancanngan sistem ini, teradiri dari struktur menu, perancangan antar muka dan jaringan

3.10.1. Struktur Menu

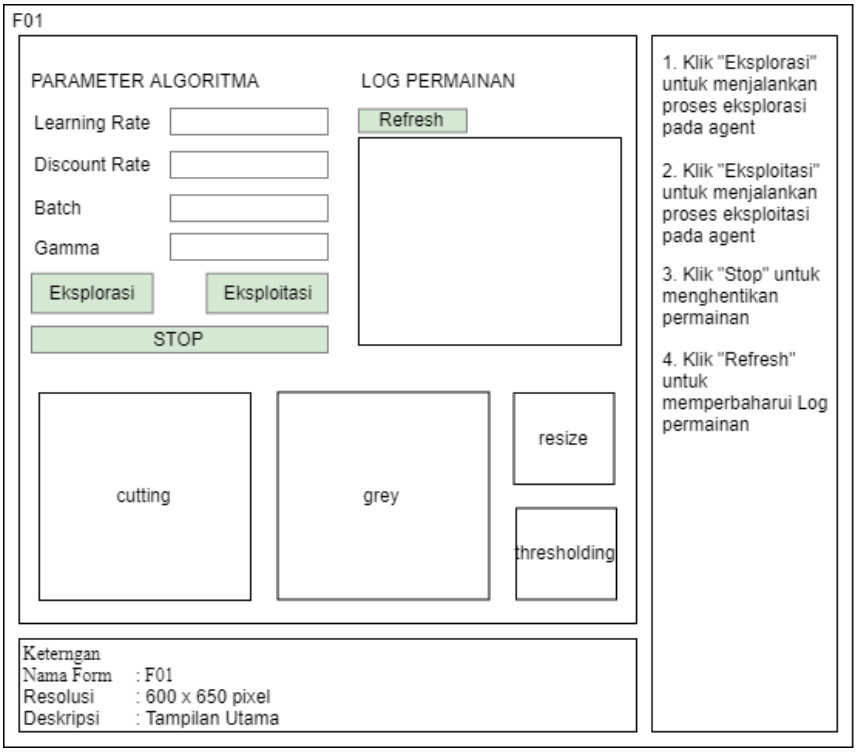
Perancangan Struktur menu adalah gambaran alur sistem yang akan dibangun, juga dapat berfungsi sebagai navigasi halaman yang ada dalam sistem. Perancangan struktur menu pada sistem yang akan dibangun dapat dilihat pada Gambar 3.29 Struktur Menu.



Gambar 3.29 Struktur Menu

3.10.2. Perancangan Antarmuka

Perancangan antarmuka menggambarkan rencana tampilan dari sistem yang akan dibangun. Rancangan antarmuka dari sistem yang akan dibangun untuk tampilan utama dapat dilihat pada Gambar 3.30 Tampilan Utama.



Gambar 3.30 Tampilan Utama

Sedangkan untuk tampilan antramuka permainan *flappy bird* dapat dilihat pada Gambar 3.31 Tampilan Permainan Flappy Bird.

Gambar 3.31 Tampilan Permainan Flappy Bird

3.10.3. Perancangan Pesan

Perancangan pesan merupakan pesan yang muncul guna memberi peringatan atau pemberitahuan kepada user. Pada sistem yang akan dibangun, perancangan

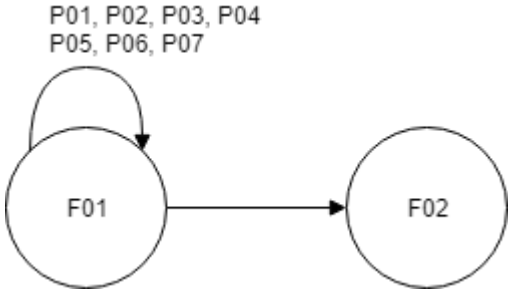
pesan digunakan untuk memvalidasi inputan nilai *learning rate* dan *batch*. Adapun perancangan pesan tersebut dapat dilihat pada Tabel 3.94 Perancangan Pesan.

Tabel 3.94 Perancangan Pesan

NO	Kode	Keterangan
1	P01	Nilai Learning Rate Tidak Boleh Kurang Dari 0
2	P02	Nilai Discount Rate Tidak Boleh Kurang Dari 0
3	P03	Nilai Discount Rate Tidak Boleh Lebih Dari 1
4	P04	Nilai Batch Tidak Boleh Kurang Dari 0
5	P05	Nilai Kondisi Henti Tidak Boleh Kurang Dari 0
6	P06	Nilai Kondisi Henti Tidak Boleh Lebih Dari 1000
7	P07	Kolom Inputan Tidak Boleh Kosong dan Harus Berupa Angka

3.10.4. Jaringan Semantik

Jaringan Semantik memeberikan gambaran mengenai keterhubungan dari satu antar muka ke antar muka lainnya. Jaringan semantik yang terbentuk dari sistem ini dapat dilihat pada Gambar 3.32 Jaringan Semantik.



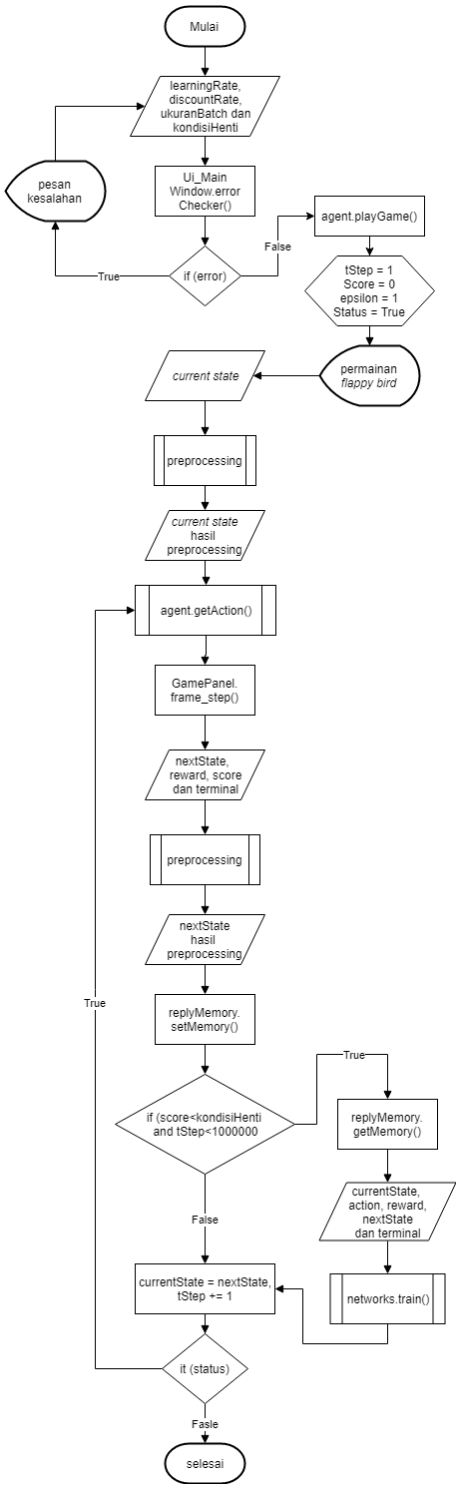
Gambar 3.32 Jaringan Semantik

3.10.5. Perancangan Method

Perancangan method adalah kumpulan metode yang terdapat pada system yang akan dirancangan. Berikut adalah perancangan method yang terdapat pada sistem:

1. Proses Eksplorasi

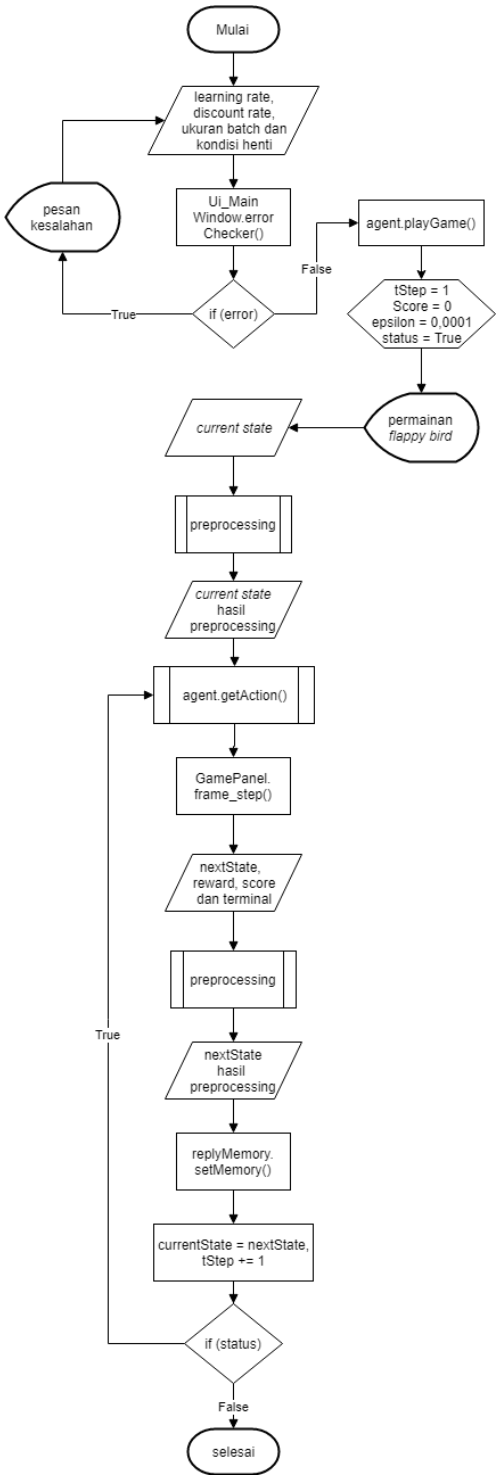
Adapun perancangan method eksplorasi, yaitu method onClickedExploration() dapat dilihat pada Gambar 3.33 Ui_MainWindow.onClickedExploration().



Gambar 3.33 Ui_MainWindow.onClickedExploration()

2. Proses Eksploitasi

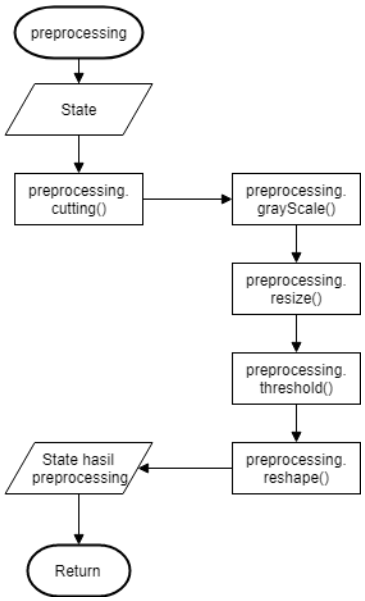
Adapun perancangan method eksploitasi, yaitu onClickedExploitation() dapat dilihat pada Gambar 3.34 Ui_MainWindow.onClickedExploitation().



Gambar 3.34 Ui_MainWindow.onClickedExploitation()

3. Preprocessing

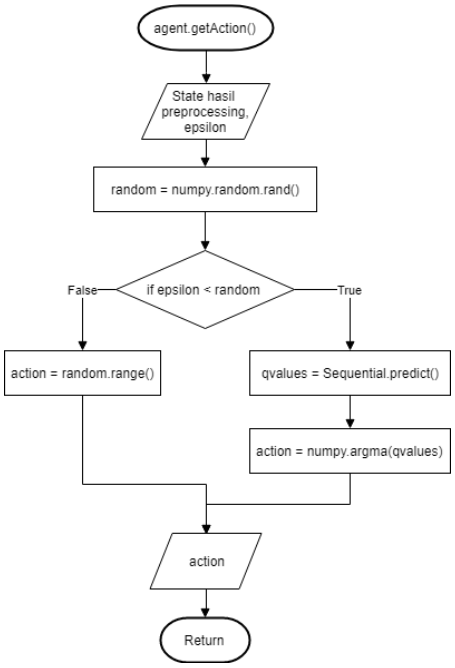
Adapun perancangan method *predefined* preprocessing dapat dilihat pada Gambar 3.35 *Predefined* Preprocessing.



Gambar 3.35 *Predefined* Preprocessing

4. Memilih action

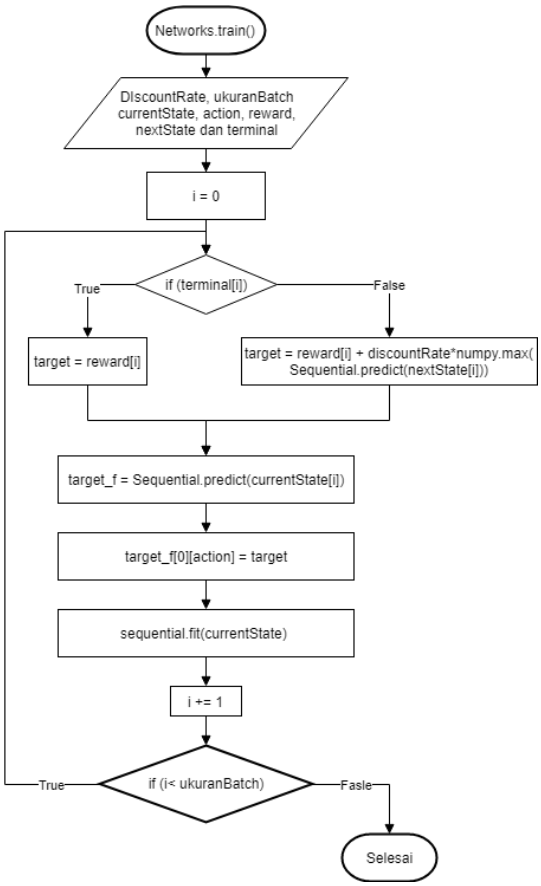
Adapun perancangan method *predefined* memilih action, yaitu getAction() dapat dilihat pada Gambar 3.36 *Predefined* agent.getAction().



Gambar 3.36 *Predefined* agent.getAction()

5. Melatih CNN

Adapun perancangan method *predefined* melatih model CNN, yaitu train() dapat dilihat pada Gambar 3.37 *Predefined* networks.train().



Gambar 3.37 *Predefined* networks.train()