IMPLEMENTATION OF Q-LEARNING COMBINED WITH CONVOLUTIONAL NEURAL NETWORK ON AGENTS THAT PLAY FLAPPY BIRD GAME

Fajar Abdi Nugraha[1], Ednawati Rainarli[2]

[1,2] Informatics Engineering – Indonesian Computer University
Jl. Dipatiukur 112-114 Bandung
E-mail : nugrahafajar37@gmail.com[1], ednawati.rainarli@email.unikom.ac.id[2]

ABSTRACT

This study shows how to implement Convolutional Neural Network(CNN) as action value function approximation to predict Q-Value on Q-Learning for every actions available, on the case of agent learning to play Flappy Bird game. The action value function approximation is used to reduce number of experiments in the exploration process and increase the score obtained by agent when playing the game, because based on previouse research using Q-Learning only, agent needs a lot of experiments playing the game during the exploration process to get good abilities seen from the scores obtained. There are two main phase in the systems, namely exploration and exploitation. Exploration is a learning process to gain knowledge of playing the game, the knowledge gained at this phase, then used at the exploitation phase. Exploitation is the testing process of the knowledge gained in the previous stage. The results of the study show that combining Q-Learning and CNN shows the results, the agent can complete the exploration process with an average of 2336.6 experiments and the average score obtained is 575.2 where 1 in 5 trials successfully touched a score of 1000.

Kata kunci : *Convolutional Neural Network* (CNN), *action-value function approximation, Q-Learning, agent, Flappy bird*

1. PRELIMINARY

Flappy bird is a side-scrolling game that is played by controlling a bird up or down. The object of the game is to pass through the pipe gap, where the pipe gap has various heights. If the player makes it through the pipe then the score increases, if the player hits the pipe or the ground then the game is over.

The previous research on the completion of the flappy bird game by Moritz [2] using Q-learning showed that for 1000 experiments the highest score obtained was 169 [1] smaller than the ability of human experts, namely Ryu Dragon who achieved a score of 328 [3]. In addition, it was explained that the ability of agents to get scores depends on the number of experiments in the exploration process, the more number of experiments the better the ability [2].

This happens because flappy bird has a dynamic component, namely the position of birds, pipes and gaps between the various pipes, resulting in a large state space and the exploration process requires many experiments because the action value function in Q-Learning cannot predict the Q-value of an action in a new state [2]. These problems can be overcome by improvising to generalize state [2] so that the agent can predict the Q-value of an action in a state that has never been encountered before.

One improvisation is adding an action value function approximation to the Q-Lerning algorithm [4] to predict the Q-value of an action in a particular state. The use of this method can generalize the state space so that the agent can reduce the number of exploration process experiments while still getting good results [4]. Previous research related to the use of Convolutional Neural Network (CNN) as an action value function approximation in Q-Learning was conducted by Mnih and colleagues [4] as the pioneers of this combination for the game case of Atari 2600. Obtained results, 3 of 7 games managed to surpass the ability of human experts [5]. Another study by Ajay Rao and colleagues [6] used a combination of Q-Learning and CNN in 8 games from OpenAI-Gym. The research explains that the use of CNN as an action value function approximation can generalize dynamic state space and suitable for the case of autonomous systems [6].

Therefore, this research will implement Q-Learning combined with the Convolutional Neural Network in the case of agents playing flappy birds.

2. CONTENT OF STUDY

There are two main stages that will be carried out on the system being built, namely the exploration and exploitation stage. One iteration at this stage is same as one frame in the game. The description of the stages that will be carried out can be seen in **Figure 1.** System Overview.

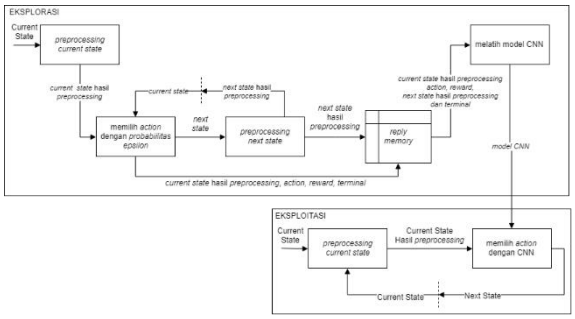

Figure 1. System Overview

The first stage is exploration, which is the agent learning phase of how to play the flappy bird game, this stage starts from the agent getting the current state of the game (in the form of a state of the game image), then preprocessing the image. Preprocessing consists of image cropping, conversion to grayscale, resize and thresholding. After preprocessing the agent will choose an action with epsilon probability, after the action is obtained and executed in the game the agent will get a reward, next state and terminal as feedback. Then, the agent preprocesses the next state. After that, the current state of the results of preprocessing, action, reward, next state and terminal (called experience) input into the reply memory and train the CNN model with training data (experience) taken randomly as many as 32 experiences (batch) from reply memory. Finally, update the next state to the current state and the process is repeated from the stage of choosing the action with epsilon probability. This stage has a stop condition when the agent touches a score of 20 or when the agent has played the game for 1000000 frames.

The second stage is exploitation, which is the stage of using the CNN model that has been trained at the exploitation stage. This stage begins with the agent getting the current state of the game, then the current state is preprocessed. Furthermore, the agent will choose an action using the CNN model with the input current state that has been preprocessed. The output of the CNN model is the Q-value of each action an agent can take. After the Q-value of each action is obtained, the agent will choose the action with the largest Q-value. At the exploitation stage the maximum score used is 1000.

2.1. Input Data

There are several input data used in this method, including:

2.1.1. State

State is the image of the state of the game. There are two types of states, namely the current state which is the current state or the state before the action is executed and the next state is the state after the action is executed. Examples of states can be seen **Figure 2.** Game State.

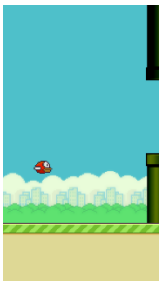

Figure 2. Game State

2.1.2. Action

In the flappy bird game there are 2 actions that can be done by the agent and each action is given an integer value. The following actions can be taken:

1. Value 0 to go down.
2. Value 1 to go up.

2.1.3. Reward

Reward is the value obtained by the agent for each action performed in a particular state. The rewards used in this study include:

1. +1 when the bird made it through the pipe.
2. 0,1 when then bird is still alive.
3. -1 when a bird crashes into a pipe or ground/

2.1.4. Terminal

Terminal is the game status after an action is performed, the data type of the terminal is boolean. The terminal is True, if after the action is executed the game ends. The terminal is false if the bird still survives after the action is executed.

2.2. Preprocessing

Preprocessing is the stage of processing the original input data (called state) before the data is used at a later stage. Original data input is an image of the state of the game. The image will go through a preprocessing process including cutting, converting RGB to grayscale, resizing and thresholding.

2.2.1. Cutting

Image cutting is the process by which parts of the image are removed with certain coordinates. In this process the original image has a size of 288 x 512 will be cut in the Y axis from coordinates 401 to 512, while the X axis will not be cut. So we get the coordinates of the upper left corner (0, 0) and lower right (288, 400). Then the images contained in coordinates (0, 0) through (288, 400) will be preserved and images with coordinates (0, 401) until coordinates (288, 512) will be discarded.

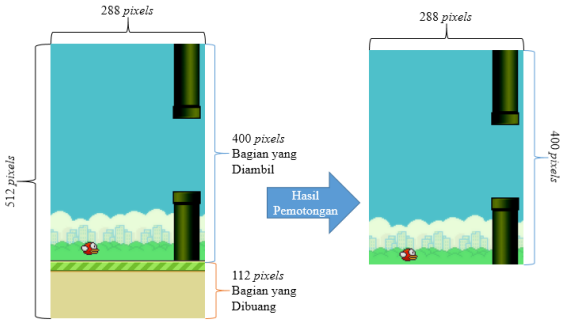

Figure 3. Cutting Illustration

2.2.2. Grayscale

The initial input image has 3 color channels namely red, blue and green. In the process of converting an RGB image to grayscale, the channel on the image will change to 1 and this can ease the process of computing the system [6]. Calculation of conversion of RGB to grayscale images can use the equation below.

$$I = 0.299*R + 0.587*G + 0.114*B \quad (1)$$

Information:

I = grayscale value.
R = red chanel value
G = green chanel value
B = blue chanel value

Here is the image of the conversion to grayscale.

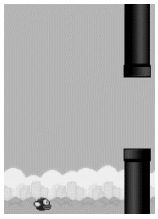

Figure 4. Grayscale Result Imagery

2.2.3. Resize

Resize is the process of changing the image size as desired. Image resizing can be downsampling or upsampling, in this study the image will be downsampled from the initial size of 288 x 512 pixels to 32 x 32 pixels. In the downsampling process, the middle value of a kernel is taken to represent the kernel [9]. The determination of the kernel size is taken from the initial image size quotient with the resulting image. Calculation of image downsampling using binary interpolation method. Bilinear interpolation is a continuation of linear interpolation for two-axis or variable interpolation [6]. The way it works is to use linear interpolation first on one axis then on the other axis. The interpolation equation is as follows.

$$f(x,y) = \frac{y2-y}{y2-y1}.f(x,y1) + = \frac{y-y1}{y2-y1}.f(x,y2) \quad (2)$$

Information:

f(x, y) = interpolated pixel value
x = x axis coordinates
y = y axis coordinates
y1 = y1 axis coordinates
y2 = y2 axis coordinates

Illustration of image resizing results can be seen below.

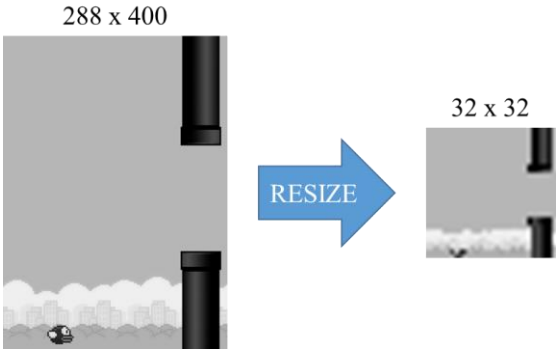

Figure 5. Resize Illustration

2.2.4. Thresholding

Thresholding is a method that makes the image value only consists of two values. Thresholding has a threshold, this study will use the threshold binary inverted method with a threshold value of 127, a maximum value of 255 and a minimum of 0, the following is the equation.

$$(x,y) = \begin{cases} 0, & img(x,y) > 127 \\ 255, & img(x,y) \leq 127 \end{cases} \quad (3)$$

Illustration of thresholding image conversion can be seen
below.

Figure 6. Thresholding Illustration

2.3. Selection of Action with Epsilon Probability

The process of selecting an action with epsilon probability consists of several steps, the following is an overview of the steps.

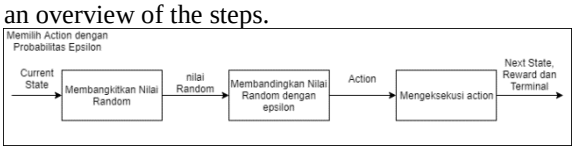

Figure 7. The Process of Choosing Actions with Epsilon Probability

The process begins with the initialization of the initial epsilon value of 1.0, the end of epsilon 0.0001 and the reduction of epsilon by 0.0000198, which will continue to reduce the epsilon value in each frame. Next, generate random values between 0 to 1 and compare with epsilon. If the random value is smaller than epsilon, then the agent will choose a random action between 0 or 1. If the random value is greater than epsilon, then the agent will choose the action with the largest Q-value obtained from the feedforward CNN output with current state input .

2.4. Q-Learning Method Combined with CNN

In the combination method, CNN will be used as an action value function approximation in Q-Learning to predict the Q-value of each action in the current state and the agent will determine the action to be taken by comparing the Q-value of each action, the action with the largest Q-value will be chosen. In the CNN training process, the target that will be used to calculate the loss and gradient calculation is the value iteration update equation contained in the Q-learning method, along with the equation. $Set\ Y' = \begin{cases} r, & terminal = True \\ r + \gamma\ max_a Q(S_{t+1}, a'; \theta), & terminal = False \end{cases}$ (4)

Keterangan:

Y'	= CNN training target
r	= reward the agent gets
γ	= *Discount rate*, $0 \leq \gamma \leq 1$,
$max_a Q(S_{t+1}, a'; \theta)$	= predictive value or CNN feedforward process in the next state of training data with maximum output values
Terminal	= The game status is in the next state, whether it ends or not

The architecture of CNN used in this study is as follows.

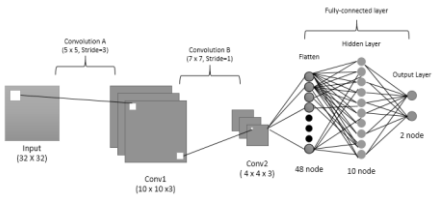

Figure 8. CNN Architecture

2.4.1. Feedforward

At the feedforward CNN stage the input data is the preprocessed state that will pass through the convolutional layer A, convolutional layer B and fully-connected layer to produce 2 outputs containing Q-Value for each action.

a. Convolution Layer A

Convolutional layer consists of arranged neurons that form a square (filter). The operation that occurs in the layer is the dot product between the filter and the input image, the filter will continue to be shifted by stride until the entire image surface. The results of the operation are called feature maps. In Convolution Layer A there are 3 5x5 filters with stride 3, the following is the equation of the operation that occurs in filter A.

$$\boldsymbol{FA1_{i,j} = \sum_{m=0}^{4} \sum_{n=0}^{4} A1_{m,n}\ C_{i*3+m,j*3+n} + bA_0} \quad (5)$$

Information:

FA1 = feature map of operating results
A1 = filter A1
C = image input
bA = bias
i = line in the feature map
j = column in the feature map
m = row in the filter
n = column in the filter.

After getting the FA1 feature map, then run the ReLU activation function, with the equation.

$$\boldsymbol{RA1(x, y) = max(0, RA1(x, y))} \quad (6)$$

Information:

RA1 = feature map of the activation function

The activation function will change the negative value to 0.

b. Convolution Layer

In convolutional layer B, the same process will be carried out as in convolutinal layer A, except that the input is triplicate, namely feature maps RA1, RA2 and RA3. In convolutional layer B there are 3 7x7 filters with stride 1. Here is the formula.

$$\boldsymbol{FB1_{i,j} = \sum_{C=1}^{3} \sum_{m=0}^{6} \sum_{n=0}^{6} B1_{m,n} \cdot RAc_{i*1+m,j*1+n} + bB_0} \quad (7)$$

Information:

FB1 = feature map of operating results
B1 = filter B1
RAc = RA filter input to c
bB = bias
i = line in the feature map
j = column in the feature map
m = row in the filter
n = column in the filter.

After getting the FA1 feature map, then perform the same function as the previous layer, namely ReLU and produce the FB feature map. This layer produces 3 4x4 feature maps, which will then enter the fully-connected layer.

c. Fully Connected Layer

Fully Connected Layer is a multilayer perceptron neural netrowk (MLP), which has hidden input and output layers. It is the Fully connected layer that will regress the input state into the Q-value of each action. The following are the layers.

- Input layer (flatten)

  In the input layer there will be an input leveling process in the form of 4x4 feature maps totaling 3 so that the number of nodes in the input layer is 48.

- Hidden layer

  In the hidden layer of architecture that is used has 10 nodes, 48 nodes in the input layer will be connected entirely with 10 hidden layer nodes. So that the total weight contained is 480, the following is the calculation equation.

  $$\boldsymbol{z\_in_i = \sum_{j=0}^{n=47} X_j * V_{j,i} + bV_i} \quad (8)$$

  Information:

  z_ini = calculation results that will be input to the hidden layer node
  Xj = value of node X to j
  Vj, i = the weight value of V to j and to i
  bV = bias value on hidden layer V

  then calculate the value of exit Z, which is the result of the process of the hidden layer node. After that, calculate the Z output value by

activating the ReLU activation function at the z_in input value.

- Output layer

The output layer has 2 output nodes (a number of actions agents can do), each node is connected to the output layer Z. An operation occurs using the equation below.

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

Information:

y_ini = input for Y layer output node
Zj = value of node Z to j
Wj, i = value of weight W to j, to i
bWi = bias value to i
n = number of nodes in hidden layer Z

After getting y_in then run the linear activation function to output the Y value. Here is the equation of the linear activation function.

$$\varphi(x) = x \quad (10)$$

Information

x = input
φ = activation function

The output layer has 2 outputs (number of actions, 0 and 1), the value generated from feedforward is the Q-Value of each action.

2.4.2. Backpropagation

At this stage the gradient calculation of each parameter is performed to correct the parameter value based on the error value. The error value used comes from the feedforward process. The backpropagation process consists of several calculations, along with the flow of calculations.

a. Calculation of Loss Value

Loss function is a function that will produce an error value from the prediction of a model to the target, for the general regression case using Mean Squared Error (MSE). Below is the equation.

$$L = \frac{1}{n}\sum_i^n (Y_i - Y'_i)^2 \quad (11)$$

Information:

L = Loss Value
n = amount of data (output)
Y = predictive value
Y '= target value

b. Gradient Calculations in The Output Layer

After calculating the loss value, then calculate the gradient parameter W to Loss, using the formula chain rule.

$$\frac{\partial L}{\partial W_{j,i}} = \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y_{in_i}}\frac{\partial y_{in_i}}{\partial W_{j,i}} \quad (12)$$

Information:

$\frac{\partial L}{\partial W_{j,i}}$ = The parameter gradient W respect to loss L
$\frac{\partial L}{\partial Y_i}$ = MSE partial derivative, i.e. 2 (Yi - Y'i)
$\frac{\partial Y_i}{\partial y_{in_i}}$= The derivative of the Linear activation function, i.e. 1
$\frac{\partial y_{in_i}}{\partial W_{j,i}}$= The parameter gradient W respect to the y_in matrix, i.e. Z

c. Gradient Calculations in The Hidden Layer

Next, calculate the gradient of the parameter V to Loss, with the chain rule formula as follows.

$$\frac{\partial L}{\partial V_{k,j}} = \sum_i^m \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y_{in_i}}\frac{\partial y_{in_i}}{\partial Z_j}\frac{\partial Z_j}{\partial z\_in_j}\frac{\partial z\_in_j}{\partial V_{k,j}} \quad (13)$$

Information:

$\frac{\partial L}{\partial V_{k,j}}$ = The parameter gradient V with respect to loss L
$\frac{\partial y_{in_i}}{\partial Z_{j,i}}$ = Z parameter gradient to the matrix yini, i.e. weight value of $W_{j,i}$
$\frac{\partial Z_j}{\partial z\_in_j}$ = derivative of the ReLU with respect to z_in, i.e. $\begin{cases} 1 & z\_in_j \geq 0 \\ 0 & z\_in_j < 0 \end{cases}$
$\frac{\partial z\_in_j}{\partial V_{k,j}}$ = The parameter gradient V with respect to z_in, i.e. nilai X_k

d. Gradient Calculations in The Convolution Layer B

Next calculate the gradient of the filter parameter B for loss, using the following chain rule.

$$\frac{\partial L}{\partial Bi} = \frac{\partial L}{\partial Xi}\frac{\partial Xi}{\partial FBi}\frac{\partial FBi}{\partial Bi} \quad (14)$$

Before calculating the gradient of filter B, first calculate $\frac{\partial L}{\partial X_k}$, using the chain rule formula, as follows.

$$\frac{\partial L}{\partial X_k} = \sum_j^n \sum_i^m \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y_{in_i}}\frac{\partial y_{in_i}}{\partial Z_j}\frac{\partial Z_j}{\partial z\_in_j}\frac{\partial z\_in_j}{\partial X_k} \quad (15)$$

Information:

$\frac{\partial L}{\partial X_k}$ = The Gradient X matrixs with respect to loss L
$\frac{\partial z\_in_j}{\partial X_k}$ = The Gradient X matrixs respect to z_in_j, i.e. weights $V_{k,j}$

After $\frac{\partial L}{\partial X_k}$, calculate the value of $\frac{\partial L}{\partial FBi}$ with the following formula.

$$\frac{\partial L}{\partial FBi} = \frac{\partial L}{\partial Xi}\frac{\partial Xi}{\partial FBi} \quad (16)$$

Information:

$\frac{\partial L}{\partial FBi}$ = Gradient feature map FB respect to Loss L.
$\frac{\partial L}{\partial X_k}$ =Gradient X matrixs respect ti Loss L
$\frac{\partial Xi}{\partial FBi}$ = derivative of the ReLU with respect to FB, i.e. $\begin{cases} 1 & FB1_{0,0} \geq 0 \\ 0 & FB1_{0,0} < 0 \end{cases}$

After $\frac{\partial L}{\partial FB}$ is known, conduct a convolution between $\frac{\partial L}{\partial FB1}$ and the RA1, RA2 and RA3 featuure to obtain the gradient $\frac{\partial L}{\partial B}$, following the formula

$$\frac{\partial L}{\partial B1_{i,j}} = \sum_{c=1}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3}\frac{\partial L}{\partial FB1_{m,n}} \cdot RAc_{i+1+m,j+1+n} \quad (16)$$

Information:

$\frac{\partial L}{\partial B1_{i,j}}$ = The parameter Gradient Filter B respect ti Loss L.

$\frac{\partial L}{\partial FBi}$ = Gradient feature map FB respect to Loss L

RAc = Featuremap RA

e. Gradient Calculations in The Convolution Layer A

Finally, the gradient calculation of the filter parameter A uses the chain rule formula.

$$\frac{\partial L}{\partial Ai} = \sum_{j=1}^{n}\frac{\partial L}{\partial RAj}\frac{\partial RAi}{\partial FAi}\frac{\partial FAi}{\partial Ai} \quad (17)$$

Before calculating the gradient of filter A, first calculate $\frac{\partial L}{\partial RAj}$, using the chain rule formula by cross-correlating between $\frac{\partial L}{\partial FBi}$and Filter Bi. Here's the equation.

$$\frac{\partial L}{\partial RA1_{ij}} = \sum_{m=0}^{-3}\sum_{n=0}^{-3} B1_{i+m,j+n}\frac{\partial L}{\partial FB1_{-m-n}} \quad (18)$$

Information:

$\frac{\partial L}{\partial RA1_{ij}}$= Gradient feature map RA respect to Loss L

$B1_{i,j}$ = Filter B1

$\frac{\partial L}{\partial FBi}$ = Gradient feature FB respect to Loss L

After getting the $\frac{\partial L}{\partial RAi}$. matrix. Next do the calculation of the value of $\frac{\partial L}{\partial FAi}$.

$$\frac{\partial L}{\partial FAi} = \sum_{j=1}^{3}\frac{\partial L}{\partial RAj}\frac{\partial RAi}{\partial FAi} \quad (19)$$

Information:

$\frac{\partial L}{\partial FAi}$ = Gradient feature map F respect to Loss L

$\frac{\partial L}{\partial RAj}$ = Gradient feature map RA respect to Loss L

$\frac{\partial RAi}{\partial FAi}$ = derivative of the ReLU with respect to FA, i.e. $\begin{cases} 1 & FA1_{0,0} \geq 0 \\ 0 & FA1_{0,0} < 0 \end{cases}$

After the value is known $\frac{\partial L}{\partial FAi}$, do the convolution process between $\frac{\partial L}{\partial FAi}$ and the input image C to produce the gradient $\frac{\partial L}{\partial Ai}$. Here's the equation.

$$\frac{\partial L}{\partial A1_{i,j}} = \sum_{m=0}^{27}\sum_{n=0}^{27}\frac{\partial L}{\partial FA1_{m,n}} \cdot C_{i+1+m,j+1+n} \quad (20)$$

Infotmation:

$\frac{\partial L}{\partial A1_{i,j}}$ = Gradient Filter A respect to loss L

$\frac{\partial L}{\partial FA1}$ = Gradien feature map F respectto loss L

C = input image

f. Parameter Update

The weight update process will use Adam based on the obtained gradient value. Adam is a first order gradient based weight optimization method of stochastic objective functions, based on the adaptive estimation of low order moments, with equations.

$$m_t = \beta_1 * m_{t-1} + (1-\beta_1) * g_t \quad (21)$$

$$v_t = \beta_2 * v_{t-1} + (1-\beta_2) * g_t^2 \quad (22)$$

$$\hat{m}_t = m_t / (1-\beta_1^t) \quad (23)$$

$$\hat{v}_t = v_t / (1-\beta_2^t) \quad (24)$$

$$\theta_t = \theta_{t-1} - \alpha * m'_t / (\sqrt{v'_t} + \epsilon) \quad (25)$$

Information:

β_1 = decay of momentum 1, beta1

β_2 = decay of momentum 2, beta2

g_t = parameter gradien value

m_t = first vector moment, where $m_0 = 0$

v_t = second vector moment, where $v_0 = 0$

$\hat{m}_t$ = estimated bias-corrected vector from the first moment.

$\hat{v}_t$ = estimated bias-corrected vector from the second moment.

α = learningrate, alpha

θ_{t-1}= old weights value

θ_t = new weights value

ϵ = epsilon preventive of divider 0, 10^{-8}

2.5. Selecting Actions with CNN

The process of selecting actions with CNN has several steps, the following is an overview of the steps.

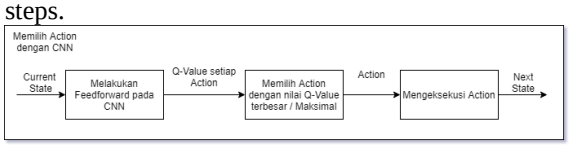

Figure 9. steps for selecting actions with CNN

The process starts with feedforward on CNN with current state input, the output of feedforward is the Q-value for each action (0 and 1). Then the agent will decide the action to be taken by comparing the Q-value of all actions, the action with the largest Q-value will be the selected action and then executed in the flappy bird game. The following is an illustration of the process of selecting action with CNN.

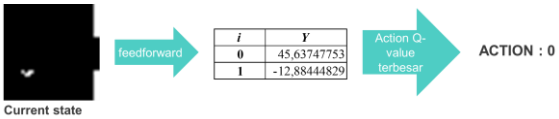

Figure 10. Illustration of Action Selection with CNN

2.6. Peformance Testing

Performance testing is done to measure the performance of the algorithm applied and see the effect of each change in the parameters of the algorithm. the purpose of testing is to find the optimal combination of parameters so that the average number of smallest experiments needed in

the exploration process and the largest score that can be obtained by the agent can be known. As a note in the process of testing the position of a random pipe and not set the same between one experiment with another experiment. The following is the performance testing scenario.

Table 1. Testing Scenario

Parameter	Nilai
Learning Rate	0.1, 0.01, 0.001
Discount Rate	0.1, 0.5, 0.99
Ukuran Batch	8, 16, 32
Kondisi Henti	10, 15, 20

2.6.1. Learning Rate Testing

In this test using three values, namely 0.001, 0.01 and 0.1, the other parameters are the discount rate of 0.99, batch size of 32 and stop conditions at a score of 20. Experiments that fail to meet the stop conditions are marked with a star (*) on the contents of the table cell with the column number an experimental exploration process.

Table 2. Learning Rate Testing Results

No	0,1		0,01		0,001	
	Number of experiments	Score	Number of experiments	Score	Number of experiments	Score
1	323 experiments	892	*7420 experiments	11	*6279 experiments	4
2	222 experiments	1000	*7558 experiments	22	*7005 experiments	2
3	*4797 experiments	5	*4336 experiments	14	348 experiments	351
4	*5417 experiments	5	576 experiments	724	*4843 experiments	13
5	924 experiments	974	*4851 experiments	15	*6418 experiments	8
Average	2336,6 experiments	575,2	4948,2 experiments	157,2	4978,6 experiments	75,6

The test results show the use of the value of 0.001 has the best results where the exploration process is completed with an average of 2336.6 experiments and an average score of 575.2. In addition, experiments that fail to meet stop conditions always score poorly. This shows the use of a very small learning rate increases the guarantee of the learning process to find optimal results while a learning rate that is too large will increase the likelihood of learning outcomes not converging and giving poor results [7].

2.6.2. Discount Rate Testing

In this test using three values, namely 0.1, 0.5 and 0.9, for other parameters using a learning rate of 0.1, batch size 32 and stop conditions at a score of 20. Trials that fail to meet the stop conditions are marked with a star (*) on the contents of the table cell with the column number an experimental exploration process.

Table 3. Discount Rate Testing Results

No	0,1		0,5		0,99	
	Number of experiments	Score	Number of experiments	Score	Number of experiments	Score
1	*11203 experiments	1	*4342 experiments	4	323 experiments	892
2	*8422 experiments	0	*5717 experiments	12	222 experiments	1000
3	*9257 experiments	2	*4372 experiments	5	*4797 experiments	5
4	*9510 experiments	1	*4290 experiments	5	*5417 experiments	5
5	*11048 experiments	1	*4088 experiments	7	924 experiments	974
Average	9888 experiments	1	4561,8 experiments	6,6	2336,6 experiments	575,2

The test results show the use of 0.99 has the best results where 3 out of 5 trials successfully met the stop conditions with an average of 2336.6 trials and an average score of 575.2. In addition, the assumptions in the previous test occur, namely experiments that fail to meet the stop conditions tend to give a bad score. If seen as a whole, the use of a smaller discount rate tends to give worse results, this shows the use of a discount rate that is closer to 1 will give optimal results seen from the score obtained by the agent [8].

2.6.3. Batch Size Testing

In this test using three values, namely 8, 16 and 32, for other parameters using a learning rate of 0.1, a discount rate of 0.99 and a stop condition at a score of 20. Experiments that fail to meet the stop conditions are marked with a star (*) on the contents of the table cell with the column number an experimental exploration process.

Table 4. Batch Size Testing Results

No	8		16		32	
	Number of experiments	Score	Number of experiments	Score	Number of experiments	Score
1	*4187 experiments	0	873 experiments	83	323 experiments	892
2	*5968 experiments	6	1053 experiments	937	222 experiments	1000
3	*5244 experiments	7	*6580 experiments	6	*4797 experiments	5
4	*7513 experiments	3	*8478 experiments	0	*5417 experiments	5
5	*6378 experiments	21	*7069 experiments	13	924 experiments	974
Average	5858 experiments	7,4	4810,6 experiments	207,8	2336,6 experiments	575,2

The test results show the use of a value of 32 has the best results, both seen from the average number of experiments in the exploration process and the scores obtained. The smaller the batch size used the results obtained tend to get worse. In addition, in experiment 1 with a batch size of 16 where the agent

managed to meet the stop conditions at 873 but the scores obtained were quite far compared to other experiments that managed to meet the stop conditions, the results showed the batch size had an effect on the stability of exploration process, resulting in a decrease in the ability of the agent. This is consistent with J Lin's research regarding the relationship of batch size with the stability of the learning process [9].

2.6.4. Stop Condition Testing

In this test using three values, namely 10, 15 and 20, for other parameters using a learning rate of 0.1, a discount rate of 0.99 batch size 32. Experiments that fail to meet the stop conditions are marked with a star (*) on the contents of the table cell.

Table 5. Stop Condition Testing Results

No	10		15		20	
	Number of experiments	Score	Number of experiments	Score	Number of experiments	Score
1	76 experiments	11	202 experiments	1000	323 experiments	892
2	1014 experiments	2	701 experiments	61	222 experiments	1000
3	129 experiments	1	165 experiments	1000	*4797 experiments	5
4	1212 experiments	20	*4914 experiments	5	*5417 experiments	5
5	373 experiments	21	*7408 experiments	8	924 experiments	974
Average	560,8 experiments	11	2678 experiments	414,8	2336,6 experiments	575,2

The test results show the use of stop conditions 10 showed the best results with an average of 560.8 times the experiment and always managed to meet the stop conditions. However, the stop condition 10 actually shows the worst results when viewed from the score obtained, which is an average of 11. It shows that the stop conditions that are too early can accelerate the exploration process, but can result in the score that is not too good. So that the selection of stop conditions in the exploration process will greatly affect the ability of agents to get scores when playing games [11].

3. CLOSING

In this section contains the results of the study, namely conclusions and suggestions.

3.1. Conclusion

Based on the results of research conducted on the implementation of the Q-Learning algorithm combined with CNN for the case of an agent playing a flappy bird, the average number of experiments needed to complete the exploration process is 2336.6 experiments and the average score obtained is 575 , 2. However, the exploration process is inconsistent where the agent does not always succeed in meeting the stop conditions and the large number of experiments during the exploration process does not always be directly proportional to the score obtained by the agent during the exploitation process.

3.2. Suggestion

In this study there are many deficiencies that occur. As for suggestions that can be given for development in further research to be better, including:

1. Adding a weight intializayion method.
2. Reserve weights used as checkpoints when the exploration process is not going well.
3. Try to compare another methods with CNN to find out the best method in the case of flappy bird game.

BIBLIOGRAPHY

[1] M. Ebeling-Rump, M. Kao dan Z. Hervieux-Moore, "Applying Q-Learning to Flappy Bird," *Department Of Mathematics And Statistics, Queen's University.*

[2] T. M. Buffalo, "Flappy Bird World Record," Tech Marketing Buffalo, 27 February 2014. [Online]. Available: https://techmarketingbuffalo.com/flappy-bird-world-record/.

[3] R. S. Sutton dan A. G. Barto, Reinforcement Learning An Introduction 2nd Edition, Cambridge: The MIT press, 2018.

[4] V. Mnih, K. Kavukcuoglu dan et al, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[5] P. A. Rao, B. N. Kumar dan et al, "Distributed Deep Reinforcement Learning using TensorFlow," dalam *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, IEEE, 2017, pp. 171-174.

[6] G. Bradski dan A. Kaehler, Learning OpenCV, Ebastopol: O'Reilly, 2008.

[7] S. Haykin, Neural Networks and Learning Machines, New York: Prentice Hall, 2009.

[8] C. J. Watkins dan P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279-292, 1992.

[9] R. Liu dan J. Zou, "The Effects of Memory Replay in Reinforcement Learning," *arXiv preprint arXiv:1710.06574v1*, 2017.

[10] Adriansyah dan E. Rainarli, "Implementasi Q-Learning dan Backpropagation pada Agen yang Memainkan permainan Flappy Bird," *JNTETI*, vol. 6, no. 1, pp. 1-7, 2017.