PAPER • OPEN ACCESS

Balancing Excitation and Inhibition of Spike Neuron Using Deep Q Network (DQN)

To cite this article: Tan Szi Hui *et al* 2021 *J. Phys.: Conf. Ser.* **1755** 012004

Balancing Excitation and Inhibition of Spike Neuron Using Deep Q Network (DQN)

Tan Szi Hui[1], **Mohamad Khairi Ishak**[2,], **Mohamed Fauzi Packeer Mohamed**[3], **Lokman Mohd Fadzil**[4] **and Ahmad Afif Ahmarofi**[5]

[1,2,3]School of Electrical and Electronic Engineering, Universiti Sains Malaysia, 14300 Nibong Tebal, Pulau Pinang, Malaysia
[4]National Advanced IPv6 Center, Universiti Sains Malaysia, Penang, Malaysia
[5]Faculty of Industrial Management, Universiti Malaysia Pahang, Lebuhraya Tun Razak, 26300 Gambang, Kuantan, Pahang,Malaysia

Email: khairiishak@usm.my

Abstract. Deep reinforcement learning which involved reinforcement learning with artificial neural networks allows an agent to take the best possible actions in a virtual environment to achieve goals. Spike neuron has a non-differentiable spike generation function that caused SNN training faced difficulty. In order to overcome the difficulty, Deep Q Network (DQN) is proposed to act as an agent to interact with a custom environment. A spike neuron is modelled by using NEST simulator. Rewards are given to the agent for every action taken. The model is trained and tested to validate the performance of the trained model in order to attain balance the firing rate of excitatory and inhibitory population of spike neuron. Training result showed the agent able to handle the environment. The trained model capable to balance the excitation and inhibition of the spike neuron as the actual output neuron rate is close to or same with the target neuron firing rate. The average percentage error of rate of difference between output and target neuron rate for 5 episodes achieved 0.80%.

1. Introduction

Deep Reinforcement Learning (DRL) involves the concept of artificial intelligence and machine learning. DRL combines the principle of deep learning and reinforcement learning as shown in Figure 1.1 [1]. DRL applies reinforcement learning algorithms with deep neural networks to figure out the best possible action to achieve a goal.

Reinforcement learning is a subset of machine learning. It is a training of models that involved the concept of agents, actions, environment, states and rewards as shown in Figure 1.1. Agents play a role to take actions in virtual environment. Actions are all of the possible operations to be taken by agents to perform the tasks given. States are the observations received from the environment which contain some useful information to the agents. Rewards are feedback to evaluate the action taken by the agents based on the given state. Positive rewards are received by the agents if the action is a success whereas negative rewards are given when the action is a failure. Reinforcement learning provides various algorithms such as Deep Q Network (DQN), Deep Deterministic Policy Gradient (DDPG), State-action-reward-state-action (SARSA) and so on [2]. Every algorithm has its own action space, state space, model and operator [3]. These algorithms are used as an agent to learn a model in a virtual environment which is a custom environment that represents a physical world for operation of agent. The agent explores in the custom environment and decide the best possible action to be taken in order to achieve a goal which is to collect cumulative rewards as much as possible.

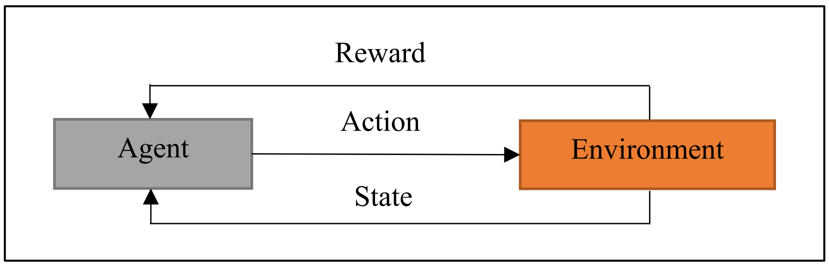

Figure 1.1. Block diagram of reinforcement learning.

DRL enables an agent to interact with a virtual environment and take actions to solve complex problem [4]. Deep neural network is used by agents to approximate a value or policy function in order to update and index the data instead of using a lookup table. With large action or state space, it is less effective to use reinforcement learning without deep neural network as the data obtained from functions is large and required longer time and larger memory to update lookup table. The data contain the information of state and action. The agent takes actions based on the current state and reward. Rewards or penalties are given to the agent based on the actions taken. The agent will gain rewards when the outcome gets close to the target whereas the action taken is faulty, the agent will receive negative rewards. The agent learns from experience to decide the best suitable action to attain a goal.

A spiking neural network (SNN) is one of the artificial neural networks that can be used in deep reinforcement learning. SNN is more closely to mimic biological nervous system compared to conventional artificial neural networks [5]. SNN is utilized to solve wider range of problems in many areas such as engineering, neuroscience and so forth. In SNN, neurons emit and process information through sequence of action potentials [6]. Action potentials represent as spikes and the neurons which generate spikes is known as spike neurons. The moment when spikes are generated by the spike neurons represent the spiking neurons are fired. Information is encoded in firing rate which is the average number of spikes generated by the spike neurons per unit time [7]. The output of the spike neuron is spike event which is a discrete over time. The spike generation function is non-differentiable as a discontinuity is created at the instance of firing time as shown in Figure 1.2. When the voltage of spike neuron crosses the threshold value, a spike is fired. After spike is fired, the spike neuron is reset and a discontinuity is created.

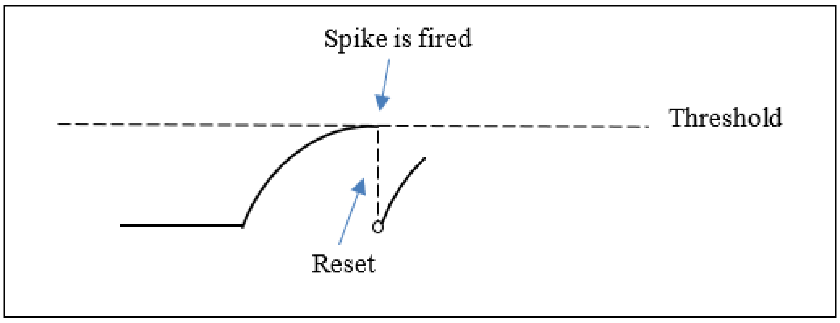

Figure 1.2. Spike event of a spike neuron.

The non-differentiable of spike event causing difficulty in training SNN by using backpropagation algorithm as the spike generation function is incompatible with backpropagation algorithm [8].

Backpropagation algorithm is a popular and useful algorithm to train neural network in machine learning as it is low in cost, fast and simple to program. Backpropagation algorithm is used to minimize the loss function which is a function to measure the difference between actual output result and the target output result.

In SNN, the firing rate of the excitatory and inhibitory population of a spike neuron is important to control the spike events. Spike neuron will have different firing rate of spikes with different configuration of the firing rate of excitatory and inhibitory population of spike neuron. In this research, the aim is to balance the firing rate of excitatory and inhibitory population of spike neuron to ensure the firing rate of inhibitory population trigger the spike neuron fire at the same firing rate as the target of excitatory population.

In spiking neural network (SNN), information is emitted and processed by spike neuron through a sequence of action potentials which is also known as spike. Information is encoded in firing rate of spike neuron. Spike neuron consists of a spike generation function for firing. The spike function is non-differentiable which create a discontinuity at the instance of firing time. Non-differentiability of the function leads to the limitation to train SNN using backpropagation algorithm as the function is not compatible to the algorithm. This has caused training of SNN using backpropagation become difficult as compared to other artificial neural network (ANN). Although SNN is biologically more realistic than artificial neural network (ANN) but receives less attention than ANN [9] due to the difficulty to train SNN. In order to overcome the non-differentiability of spike function that leads to difficulty in SNN training, deep reinforcement learning is applied to balance the firing rate of excitatory and inhibitory population of spike neuron. The excitatory and inhibitory population are very important in a spike neuron to control the spiking of the neuron. The excitatory and inhibitory population represents the weight in SNN. Spike neuron has different firing rate of spikes when different configuration on the firing rate of excitatory and inhibitory population of the neuron is applied. The firing rate of inhibitory population of the spike neuron is initialized as input and adjusted in the training to achieve the firing rate of excitatory population of the neuron has the same rate with the target neuron firing rate. In this research, an algorithm of reinforcement learning is used as agent which is Deep Q Network (DQN) to interact with a custom environment with OpenAI Gym framework to optimize spike neuron into balance state.

1.1. Background of Q-Learning

Q-learning is one algorithm used in reinforcement learning to find the best possible action based on the current state given. Q-learning is an off-policy as the function learns optimal policy which is independent of actions. In addition, Q-learning is a value-based algorithm. Q-learning updates the value function based on Bellman equation. In Q-learning, the action-value function Q(s,a) is learnt by using Bellman equation as shown in Equation (1) [10].

$$Q(s,a) = R(s,a) + \gamma \, max \, Q(s',a') \tag{1}$$

where Q(s,a) represents Q-value of current state, R(s,a) represents reward to be received for taking the action, γ represents the discount factor that controls further contribution of rewards in future and max Q(s’,a) represents optimal future Q-value for the next state-action pair. The equation indicates the sum of the current reward and the expected future discounted reward.

Q-value for the action taken from the previous state is updated by using Equation (2).

$$Q^*(s,a) = \sum [R_{t+1} + \gamma \, max \, Q^*(s',a')] \tag{2}$$

Q-value of the given state-action pair, $Q^*(s,a)$ is required to obtain as close as right hand side of the equation. When the value of $Q(s,a)$ is close to right hand side of the equation, this indicate that the $Q^*(s,a)$ converges to the optimal Q-value.

Learning rate is applied in Q-learning. Learning rate refers to a parameter for tuning in an optimization algorithm to control the adjustment of weight of neural network in order to minimize loss function. The range of learning rate is between 0 and 1. Learning rate is used to determine the step size of each episode. The new Q-value for given state-action pair, $Q(s,a)$ can be calculated by using Equation (3).

$$Q^*(s,a) = (1-\alpha)Q(s,a) + \alpha(R_{t+1} + \gamma\, max\, Q(s',a')) \quad (3)$$

where $Q^*(s,a)$ denotes new Q-value for given state-action pair, $Q(s,a)$ represents old Q-value, α is learning rate and $(R_{t+1} + \gamma\, max\, Q(s',a'))$ denotes learned value. Learned value is calculated from the sum of reward for the action taken and optimal future Q-value of the next state-action pair. The equation indicates that the new Q-value is obtained from the sum of old Q-value and learned value.

1.2. Background of Deep Q Network

Deep Q Network (DQN) combines deep neural network and Q learning to approximate Q function. DQN approximates Q-values by using neural network instead of using Q table as shown in Figure 1.3.

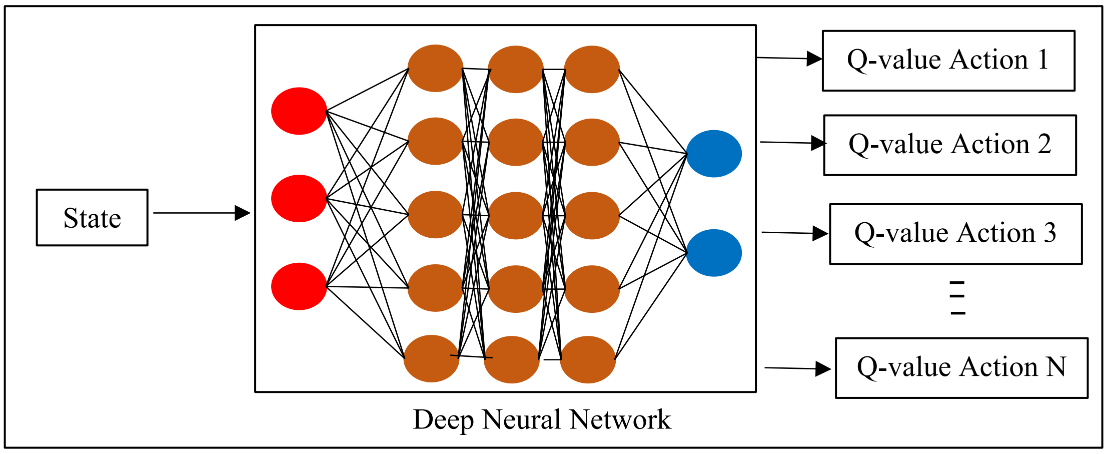

Figure 1.3. Illustration of DQN.

Experience replay is implemented in DQN. The DQN agent's experience is stored in memory to perform experience replay. The data stored consists is state, action, reward and next state. The tuple of the agent's experience is expressed In Equation (4).

$$e_t = (s_t, a_t, r_{t+1}, s_{t+1}) \quad (4)$$

where e_t denotes agent's experience, s_t denotes the state of the virtual environment, a_t represents the action taken in state s_t, r_{t+1}represents the reward given to the agent at time, t+1 based on the previous state-action pair and s_{t+1} represents the next state of the virtual environment.
Experience replay utilized the previous agent's experience to perform Q function in order to update deep neural networks.
Based on the Bellman equation of Q-learning as shown in Equation (2), $Q^*(s,a)$ has to be as close as right-hand side of the equation for convergence of Q-value. The goal of the DQN is to meet the

convergence of Q-value. Thus, the equation can be further expressed as the cost function of DQN by using Bellman Equation as shown in Equation (5) [11].

$$cost\ function = [Q(s,a;\theta) - (R(s,a) + \gamma \max Q(s',a';\theta)]^2 \tag{5}$$

where $Q(s,a;\theta)$ represents approximatiing function which is the prediction of the function, $R(s,a)$ represents previous rewards, γ represents discount factor that controls further contribution of rewards in future, $(R(s,a) + \gamma \max Q(s',a';\theta))$ denotes the immediate and future rewards and θ represents the weights of the network for training [12].
The lower the cost function, the lower the difference between predicted Q value and target Q-value. The cost function also referred as a Mean Square Error function as shown in Equation (6).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2 \tag{6}$$

where y represents the predicted value whereas y' represent targeted value.

1.3. Related Works

Spiking neural network (SNN) is artificial neural network where the communication of neuron models is through sequences of spikes [13]. The spikes are generated by spike neurons for information processing. Various types of machine learning such as supervised learning, unsupervised learning and reinforcement learning are used to train SNN for spike-timing exploration. The trained SNN models are widely used for various applications such as image recognition, data classification, path planing, decision making with various application and so forth.

SNN can be trained using different learning models. An unsupervised training algorithm is proposed to train SNN [14]. Spike neuron model is trained using Synaptic Weight Association Training (SWAT). The combination of Spike Timing Dependent Plasticity (STDP) and Bienenstock-Cooper-Munro (BCM) training rule is applied in the algorithm. An output spike train is generated from input neurons. The training and testing results showed the algorithm able to exhibit classification and convergence accuracy.

Furthermore, a supervised learning algorithm of spiking neural networks with limited precision (SNN/LP) is proposed to train SNN [15]. Synaptic weights and synaptic delays are applied with limited precision for supervised learning. 33 neurons in input layer, 8 neurons in each hidden layers and one neuron in output layer are applied in the proposed network as shown in Figure 2.6. The algorithm achieved low mean squared error in non-linear XOR classification problem. The algorithm is also used to solve Fisher iris classification problem. The algorithm capable to achieve up to 97% of classification accuracy.

A supervised multi-spike learning algorithm is proposed to train neurons in SNN [16]. A single neuron is trained to learn spike patterns in order to generate spike trains. The expression of membrane potential is simplified by the algorithm and enables the optimization of synaptic weights through the application of gradient descent. The algorithm is demonstrated in sound recognition and temporal encoding pattern classification. The results showed the algorithm able to achieve classification accuracy.

In addition, a high dimensional sensory input and perceptual ambiguity of model-free reinforcement learning is proposed to train SNN in order to solve partially observable reinforcement learning (PORL) problem [17]. The SNN that used in this algorithm consists of hidden neurons, action neurons, memory neurons and state neurons. The free-energy-based reinforcement learning (FERL) framework is implemented with the trained model to solve PORL issues. The algorithm is capable to be a solution for PORL tasks.

Deep reinforcement learning (DRL) utilized deep neural network and reinforcement learning. It enables the agent to explore in a virtual environment to learn the best possible actions in order to achieve goals. Various DRL agents are introduced such as Deep Q Network (DQN), Q-learning with normalized advantage function (NAF), Deep Deterministic Policy Gradient, Asynchronous Advantage Actor Critic (A3C) and many more. The implementation of DRL depends on the action space in the environment. Some DRL algorithms can be applied in discrete action space only whereas some work only on continuous action space. Certain DRL algorithms can be implemented for discrete and continuous action spaces. DRL solves tasks end to end and reduces the training episodes to learn simple tasks. Larger action and observation spaces able to implement in DRL.

DRL is utilized for many fields of applications. DRL combined deep neural network and reinforcement learning in video games. The algorithm is used to train machine learning model to achieve human-level performance for intelligence game playing. Deep Q-Learning and Asynchronous Advantage Actor-Critic algorithms (A3C) are proposed to learn to play Atari Breakout game [18]. Both algorithms able to learn well to play the game.

DRL is applied with deep neural network to learn exploration strategy from the input partial map [19]. DRL-based automatic exploration for navigation is used as the decision algorithm. The algorithm achieved better learning efficiency and adaptability to explore in unknown environment in simulation. The algorithm is also able to transfer from simulation to physical world and achieved better explored efficiency.

Furthermore, DRL is proposed as a marking decision via per-port in Data Center Networks (DCNs) to solve marking problems in multiple queue scenarios [20]. The algorithm is simulated for different flow size independently. The algorithm has the capability to avoid faulty Explicit Congestion Notification (ECN) mark and leads to near optimal achievement on flow completion time across different flow sizes. Deep Q Network is one of the most popular algorithms in DRL. DQN worked as an agent with the combination of supervised learning framework to learn to play game [21]. DQN has the capability to interact with gaming environment faster than humans. The proposed algorithm capable to obtain better performance with higher scores in performance test and Turing test than human.

An agent-aware dropout DQN is proposed for enhancement of online dialogue policy learning in terms of efficiency and safety to control consult rules of student policy and learn from experiences of teachers [22]. The dropout DQN is worked with a companion learning framework to integrate learning of online dialogue policy. Simulation results showed that the improvement on efficiency and safety of online policy optimization by using the proposed algorithm.

DQN can be used to solve the constraint of efficient exploration in deep reinforcement learning. A bootstrapped DQN is developed with the combination of deep exploration and deep neural networks. Experiment is conducted to test the exploration of the algorithm in Atari games. The algorithm achieved improvement on learning speed and better efficient exploration [23].

The application of DQN involved in various field of technology. An improved DQN-based learning policy is presented to achieve some enhancement in path planning technology [24]. The algorithm is proposed based on different experience's depth and different learning stages' breadth. The developed model had better performance in terms convergence speed, path accuracy and planning success rate.

2. Methodology

A spike neuron is modelled and a custom environment is built for training in order to balance the firing rate of excitatory and inhibitory population of the spike neuron by using DQN.

2.1. Related Works

A spike neuron is created by using Neural Simulation Tool (NEST) simulator. NEST simulator can be used for any size of spiking neural network model. Single spike neuron is used in training for optimization. A spike neuron is modeled using PyNEST command in Python programming language after a custom environment is built. Simulation parameters are required to be initialized for NEST simulator to model a spike neuron as shown in Table 2.1.

Table 2.1 Simulation Parameters of Spike Neuron

Simulation Parameters	Value
Simulation Time, t_sim	25000
Size of Excitatory Population, n_ex	16000
Size of Inhibitory Population, n_in	4000
Mean Rate of Excitatory Population	5.0Hz
Initial Rate of Inhibitory Population with a random number range, r_in	17.5 – 19.5Hz
Peak Amplitude of Excitatory Population, epsc	50
Peak Amplitude of Inhibitory Population, ipsc	-50
Synaptic Delay,d	0.01
Lower bound of search interval, self.lower	0
Upper bound of search interval, self.upper	50

In the spike neuron model, 4 nodes are created using NEST simulator which are leaky integrate-and-fire neuron, Poisson generator, voltmeter and spike detector. Leaky integrate-and-fire model is created with alpha-function shaped synaptic currents. The synaptic currents and post-synaptic potentials have finite rise time. The leaky integrate-and-fire neuron is simple and popular neuron model as it is easier to be simulated and analyzed. Two Poisson generator is used to simulates the spike neuron that firing with Poisson statistics for excitatory and inhibitory population. The function of voltmeter is to records the membrane potential of the connected nodes to memory and spike detector is created to detect spike event from the spike neuron and store into memory. After the 4 nodes are created, the configuration of leaky integrate-and-fire neuron and spike detector remain default as the default configuration is satisfied in the project. Poisson generator and voltmeter are configured by using SetStatus function. The leaky integrate-and-fire neuron is connected to Poisson generators, voltmeter and spike detector after configuration.

The spike neuron model is simulated according to the simulation parameters. The firing rate of inhibitory population is initialized as input of the simulation. The firing rate of inhibitory population is scaled and the Poisson generator of inhibitory population is configured. The simulation is started after the spike detector is reset 0. The firing rate of the actual output neuron rate is measured with given input firing rate of inhibitory population. The spiking activity of the spike neuron is recorded by the spike detector and the firing rate of output neuron is returned.

2.2. A Custom Environment with OpenAI Gym Framework

A custom environment using OpenAI gym toolkit is built. The spike neuron is converted into OpenAI Gym framework after the custom environment is built. The environment set the initial state for the problems to be solved. Action space and observation space are configured for both DRL algorithms. Action space represents how many possible actions for the DRL agents to interact with the environment and observation space represents all the data that generated by the environment and to be observed by the agents as shown in Figure 2.1.

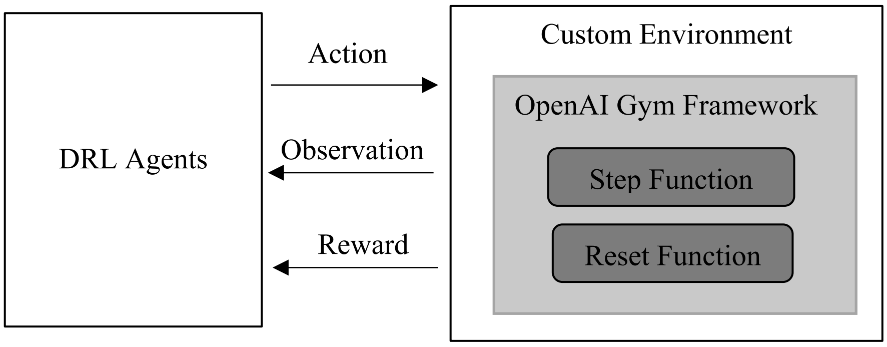

Figure 2.1. Interaction of DRL agents and a custom environment with OpenAI Gym framework.

2.2.1. Custom Environment in DQN. The simulation parameters to model a spike neuron is initialized and 4 nodes for spike neuron model are created in the custom environment. DQN agent is constructed with 4 hidden layers with 40 hidden neurons in each layer. There are 3 neurons in input layer and 4 neurons in output layer as the algorithm has 3 observation space and 4 action spaces in the network. A visualization of neural network in DQN is showed in Figure 2.2. Rectified Linear Unit (ReLU) is used as activation function in hidden layers of the network. The formula of ReLU is expressed in Equation (7) and its graph is showed in Figure 2.3.

$$ReLU = \begin{cases} z, z > 0 \\ 0, otherwise \end{cases} \tag{7}$$

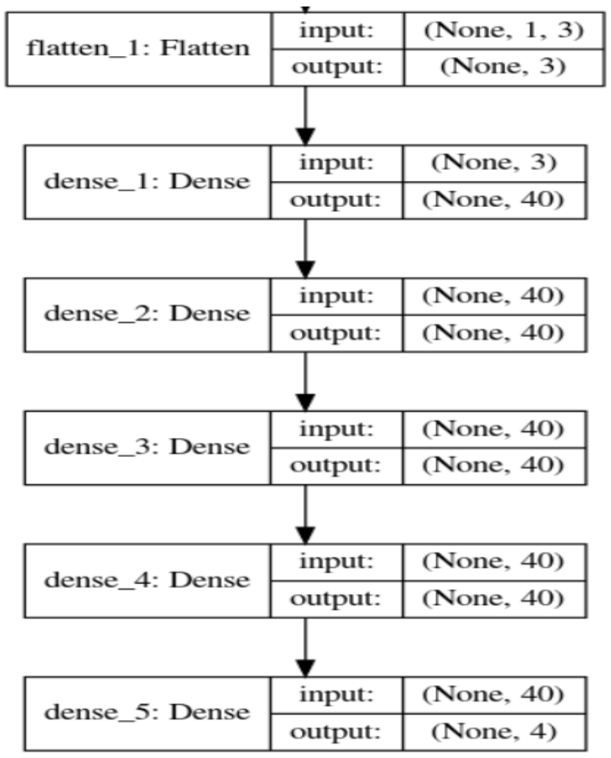

Figure 2.2. Visualization of neural network in DQN.

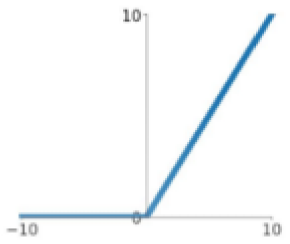

Figure 2.3. Graph of rectified linear unit (ReLU) activation function.

A linear activation function is applied in output layer of the network as the use of neural network in DQN agents is for approximation.

Model parameters are set for the DQN agent as shown in Table 2.2. Adam optimization algorithm is applied in the DQN agent as the algorithm is effective and capable to achieve good performance faster. The network weight is updated iteratively based on training data by using Adam optimization algorithm. The learning rate of the algorithm is set to 0.001 as this is the good default setting for the algorithm [25]. Epsilon-greedy policy is used in DQN for discrete action spaces for exploration.

Table 2.2 Setting of Model Parameter

Model Parameter	Setting
Memory	Sequential Memory
Policy	Linear Annealed Policy
Warm Up Step	50
Number of Actions	4
Model Update Target	1e-3
Optimization Algorithm	Adam with learning rate of 1e-3

Action space for DQN in the custom environment is discrete type with 4 possible actions. Discrete action space works better compared to continuous action space in DQN. In observation space, three information to be observed which are output neuron spike rate, different between the excitatory population spike rate and the actual output neuron rate and different between the excitatory population spike rate and the actual output neuron rate with normalization with respect to the excitatory population mean rate.

DQN agent makes a decision on the action to be taken in step function of the algorithm. During learning, the agent obtains the current state and reward and followed by update policy. The agent plays a role to take actions based on the learnt policy. The agent takes actions based on the 4 possible actions defined as shown in Table 2.3. The actions are divided into two groups which consists of addition and subtraction of the firing rate of inhibitory population rate. Two actions are fixed for the addition and subtraction value which is 0.01 on the firing rate of inhibitory population whereas another two actions are addition and subtraction of a random number from a range of 0.02 to 0.05. The firing rate of inhibitory population is applied into NEST simulator and the output spike neuron rate is collected and stored into a variable by the spike detector.

Table 2.3 Action list of DQN

Action	Details of action
0	Current inhibitory rate + 0.01
1	Current inhibitory rate - 0.01
2	Current inhibitory rate + random number from range of 0.02 to 0.05
3	Current inhibitory rate - random number from range of 0.02 to 0.05

The difference between actual output neuron rate and target neuron firing rate and the percentage of the error of the difference between actual output neuron rate and target neuron firing rate are computed. The current state of the environment is updated according to the action taken.
After the current state of the environment is updated, the rewards and done status of the environment are computed. Reward is feedback from environment and is given to the agent. Reward is categorized into reward after the action taken by the agent and reward for complete condition of environment. The agent receives rewards from both categories for every action taken as shown in Table 2.4.

Table 2.4 Reward System of DQN

Reward	Category of reward	Type of Reward	Condition to Meet
Temporary reward + 10	Reward after action taken by the agent	Positive reward	Current rate of difference between actual output neuron rate and the excitatory population rate less than previous rate of difference between actual output neuron rate and the excitatory population rate
Temporary reward - 10	Reward after action taken by the agent	Negative reward	Current rate of difference between actual output neuron rate and the excitatory population rate more than previous rate of difference between actual output neuron rate and the excitatory population rate
Temporary reward - 100	Reward after action taken by the agent	Negative reward	Actual output neuron rate is 0Hz or higher than 100Hz.
Temporary reward + 2000	Reward for complete condition of environment	Positive reward	Rate of difference between actual output neuron rate and the excitatory population rate less than or equal to 0.02
Temporary reward - 500	Reward for complete condition of environment	Negative reward	Step taken for actual output neuron rate to meet the excitatory population rate more than 100

A reset function is used in the algorithm. The state of the environment is reset to initial state which is also can be referred as problem to be solved. The observation space is reset to initial state and the variables that used to store information from observation in the custom environment is reset. Once an episode is terminated or completed, the reset function is triggered to initialize all the state and observation in order to start a new episode of learning.

The training steps is set to 4000 steps to balance the firing rate of excitatory and inhibitory population of the spike neuron. After trained the model, the model is used to test the performance for 5 episodes. Five episodes are tested to get comparable results and reducing testing time as longer training time required for more episodes in testing. Steps taken for actual output neuron rate to meet with excitatory population rate are recorded when testing the model. A sequence of actions, states and rewards are performed in an episode and the episode is ended with terminal state if meet the goal. The done status which indicate to end one episode of training is trigger if the algorithm meets the goal or meet the maximum steps taken for actual output neuron rate to meet with target excitatory population rate. The maximum steps taken for actual output neuron rate to meet with target excitatory population rate is set as 100 steps. If the actual output neuron rate does not meet the target neuron firing rate within 100 steps, the episode will be terminated and penalty is given to the agent. The simulation is continued with another new episode of training with initialization parameters after one episode is terminated or completed. The flowchart of the DQN algorithm is shown in Figure 2.4.

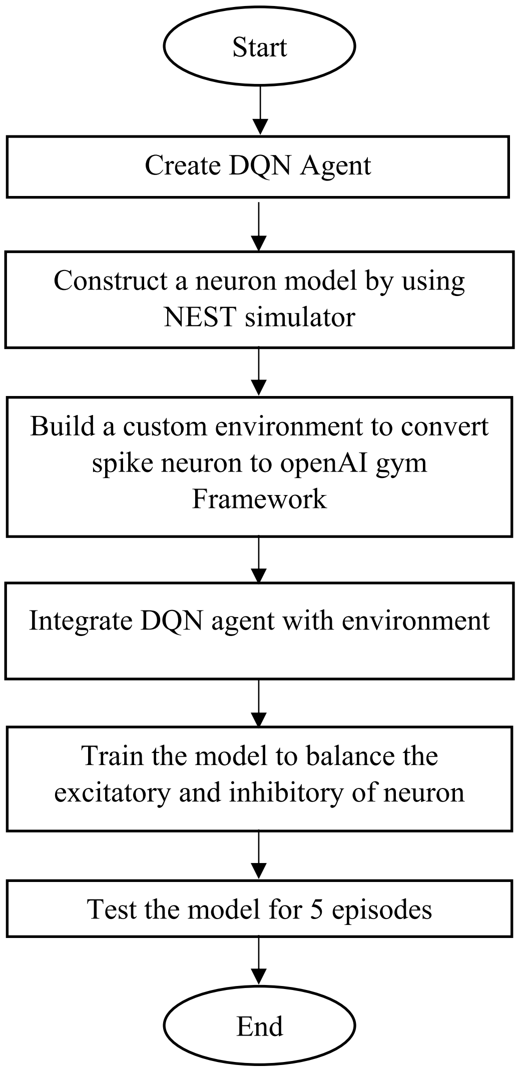
Start
Create DQN Agent
Construct a neuron model by using NEST simulator
Build a custom environment to convert spike neuron to openAI gym Framework
Integrate DQN agent with environment
Train the model to balance the excitatory and inhibitory of neuron
Test the model for 5 episodes
End

Figure 2.4. Flowchart of DQN algorithm.

3. Results and Discussion

The spike neuron model which is created is trained in the environment with OpenAI Gym framework. The model is trained for 4000 steps. Due to the constraint of memory overflow, we trained and save the model for first 2000 steps and followed by two iterations of 1000 steps training by using the saved model. The DQN agent is trained to interact with the environment and decide the actions to be taken. A learning curve of episode reward versus episodes is plotted as shown in Figure 3.1. The learning curve showed the DQN agent able explore in the environment and the model is trained to react towards the environment to achieve balance state of spike neuron. The goal of the agent is to obtain the actual output neuron rate meet with the excitatory population rate which is 4.0Hz. With this capability, the model became usable for testing.

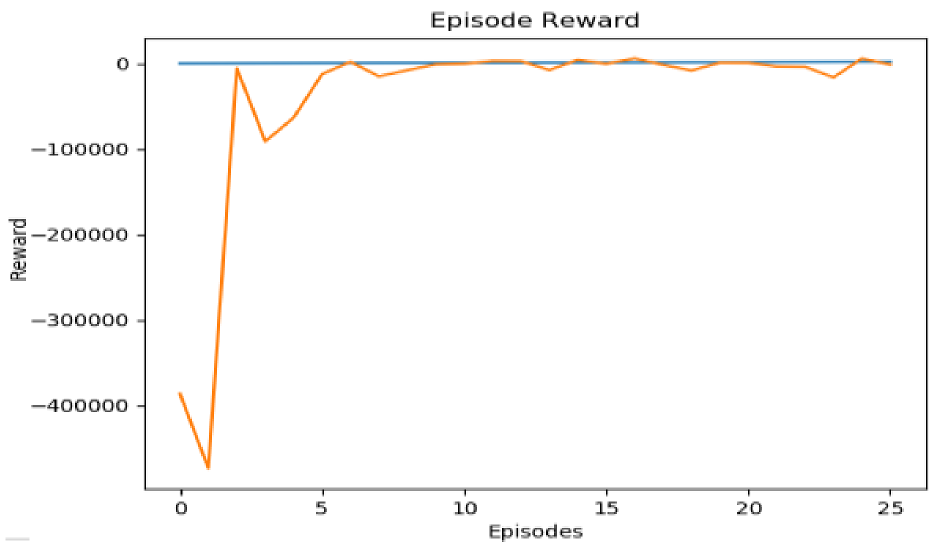

Figure 3.1. Learning curve of the spike neuron model using DQN.

After training, the model is tested for 5 episodes to validate the performance of the model. The model able to obtain positive reward for every episode during testing as shown in Figure 3.2.

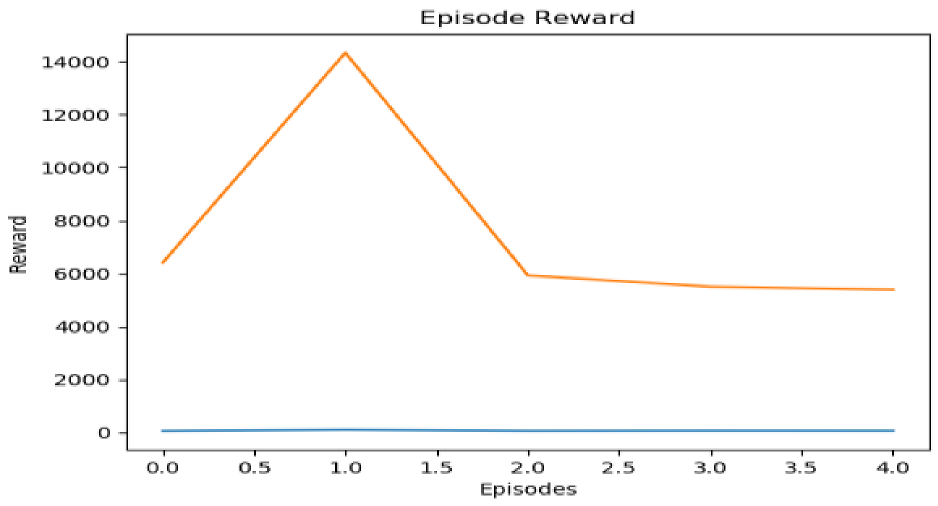

Figure 3.2 Testing curve of the trained model using DQN.

The testing result is tabulated in Table 3.1. For the first four episodes, the actual output neuron rate is very close to goal whereas the actual output neuron rate is attained the goal in the fifth episode. The inhibitory population rate is fine-tuned by the agent in order to attain the goal. The percentage of error between the rate of different of actual output neuron rate and goal is calculated in Table 3.2. The average percentage of error between rate of difference of output and target neuron firing rate achieved 0.8%. The lowest steps taken for actual output neuron rate to meet with target excitatory population rate is 54 steps. The result proved that the trained model able to interact with custom environment with OpenAI gym framework to balance the firing rate of excitatory and inhibitory population of the spike neuron.

Table 3.1 Testing result of the trained model using DQN

Episode	Simulation Parameter	Optimal Value given by DQN Agent (Hz)	Total Reward	Steps taken to attain goal
1	Mean Rate of Inhibitory Population	16.69	6410	54
	Initial Rate of Inhibitory Population	17.80		
	Output Neuron rate	4.04		
2	Mean Rate of Inhibitory Population	16.87	5250	62
	Initial Rate of Inhibitory Population	17.80		
	Output Neuron rate	4.04		
3	Mean Rate of Inhibitory Population	16.68	5930	57
	Initial Rate of Inhibitory Population	17.80		
	Output Neuron rate	4.04		
4	Mean Rate of Inhibitory Population	16.77	5500	64
	Initial Rate of Inhibitory Population	17.80		
	Output Neuron rate	3.96		
5	Mean Rate of Inhibitory Population	16.71	5390	61
	Initial Rate of Inhibitory Population	17.80		
	Output Neuron rate	4.00		

Table 3.2 Testing result for output and target neuron firing rate in DQN

Episode	Actual output neuron rate (Hz)	Target neuron firing rate(Hz)	difference of output and target neuron firing rate(Hz)	Percentage of Error (%)	Average Percentage of Error (%)
1	4.04	4.00	0.04	1	0.80%
2	4.04	4.00	0.04	1	
3	4.04	4.00	0.04	1	
4	3.96	4.00	0.04	1	
5	4.00	4.00	0.00	0	

4. Conclusion

Due to non-differentiability of spike generation activity of spike neuron, the training of spiking neural network (SNN) using backpropagation learning faced difficulty. In order to overcome this issue, Deep Reinforcement Learning (DRL) is proposed as a solution to train a spike neuron. Deep Q Network is proposed to balance the firing rate of excitatory and inhibitory population of a spike neuron. The training involved a spike neuron only as the research focus to optimize the spike neuron into balance stated before further develop into a network. The excitatory and inhibitory population is referred as weight of a spike neuron to control the spiking of the neuron. A spike neuron is trained in the custom environment with OpenAI Gym framework. The algorithm able to interact with the custom environment to attain the goal. The average percentage of error of rate of difference between output and target neuron firing rate for the algorithms obtained 0.80%. The results proved that the algorithm able to explore in the custom environment to train the spike neuron. The results also showed that the spike neuron is optimized into balance state as the actual output neuron rate is close to or same to target neuron firing rate. In future work, the spike neuron model can be further developed to create a network in order to train a spiking neural network (SNN).

5. References

[1] Francois-Lavet V, Henderson P, Islam R, Bellemare M and Pineau J 2018 An Introduction to Deep Reinforcement Learning. *Foundation and Trends® in Machine Learning* p 219-354.

[2] Samsuden M A, Diah N M and Rahman N A 2019 A Review Paper on Implementing Reinforcement Learning Technique in Optimising Games Performance *2019 IEEE 9th International Conference on System Engineering and Technology (ICSET)* p 258-263.

[3] Wang Q and Zhan Z 2011 Reinforcement learning model, algorithms and its application *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)* p 1143-1146.

[4] Arulkumaran K, Deisenroth M P, Brundage M and Bharath A A 2017 Deep Reinforcement Learning : A Brief Survey *IEEE Signal Processing Magazine* p 26-38.

[5] Abiyev R H, Kaynak O and Oniz Y 2012 Spiking Neural Networks for Identification and Control of Dynamic Plants *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)* p 1030-1035.

[6] Park S, Kim S, Choe H and Yoon S 2019 Fast and Efficient Information Transmission with Burst Spikes in Deep Spiking Neural Networks *2019 56th ACM/IEEE Design Automation Conference (DAC)* p 1-6.

[7] Oniz Y, Kaynak O and Abiyev R 2013 Spiking Neural Networks for the Control of a Servo System *2013 IEEE International Conference on Mechatronics (ICM)* p 94-98.

[8] Wu Y, Deng L, Li G, Zhu J and Shi L 2018 Spatio-Temporal Backpropagation for Training High Performance Spiking Neural Networks *Frontiers in Neuroscience.*

[9] Abusnaina A A, Abdullah R and Kattan A 2019 Supervised Training of Spiking Neural Network by Adapting the E-MWO Algorithm for Pattern Classification *Neural Process Letters* pp 661–682.

[10] Pandey D and Pandey P 2010 Approximate Q-Learning: An Introduction *2010 Second International Conference on Machine Learning and Computing* p 317-320.

[11] Sasaki H, Horiuchi T and Kato S 2017 A study on vision-based mobile robot learning by deep Q-network *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)* p 799-804.

[12] Mismar F B and Evans B L 2018 Deep Q-Learning for Self-Organizing Networks Fault Management and Radio Performance Improvement *2018 52nd Asilomar Conference on Signals, Systems, and Computers* p 1457-1461.

[13] Ponulak F and Kasinski A 2011 Introduction to spiking neural networks : Information processing, learning and applications *Acta Neurobiologiae Experimentalis*.

[14] Wade J, McDaid L, Santos J and Sayers H 2018 SWAT: An Unsupervised SNN Training Algorithm for Classification *Problems 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* p 2648-2655.

[15] Stromatias E and Marsland J 2015 Supervised learning in Spiking Neural Networks with Limited Precision : SNN/LP *2015 International Joint Conference on Neural Networks (IJCNN)* p 1-7.

[16] Miao Y, Tang H and Pan G 2018 A supervised Multi-Spike Learning Algorithm for Spiking Neural Networks *2018 International Joint Conference on Neural Networks (IJCNN)* p 1-7.

[17] Nakano T, Otsuka M, Yoshimoto J and Doya K 2015 A spiking neural network model of model-free reinforcement learning with high-dimensional sensory input and perceptual ambiguity *PloS one.*
[18] Jeerige A, Bein D and Verma A 2019 Comparison of Deep Reinforcement Learning Approaches for Intelligent Game Playing *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)* p 366-371.
[19] Li H, Zhang Q and Zhao D 2019 Deep Reinforcement Learning-Based Automatic Exploration in Navigation in Unknown Environment *IEEE Transactions on Neural Networks and Learning Systems* p 2064-2076.
[20] Majidi A, Gao X, Jahanbakhsh N, Jamali S, Zheng J and Chen G 2019 Deep-RL : Deep Reinforcement Learning for Marking-Aware via per-port in Data Centers *2019 IEEE 25th International Conference on Parallel and Distributed Systems* p 392-395.
[21] Miyashita S, Lian X, Zeng X, Matsubara T and Uehara K 2017 Developing Game AI Agent Behaving Like Human by Mixing Reinforcement Learning and Supervised Learning *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)* p 489-494.
[22] Chen L, Zhou X, Chang C, Yang R and Yu K 2017 Agent-Aware Dropout DQN for Safe and Efficient On-line Dialogue Policy Learning *The 2017 Conference on Empirical Methods on Natural Language Processing* p 2454-2464.
[23] Osband I, Blundell C, Pritzel A and Roy B 2016 Deep Exploration via Bootstrapped DQN *Advances in Neural Information Processing Systems 29 (NIPS).*
[24] Lv L, Zhang S, Ding D and Wang Y 2019 Path Planning via an Improved DQN-Based Learning Policy *IEEE Access* p 67319-67330.
[25] Kingma D and Ba J 2015 ADAM : A Method For Stochastic Optimization *International Conference on Learning Representations*.