IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 10, No. 1, March 2021, pp. 175~183
ISSN: 2252-8938, DOI: 10.11591/ijai.v10.i1.pp175-183

Spike neuron optimization using deep reinforcement learning

Tan Szi Hui, Mohamad Khairi Ishak
School of Electrical and Electronic Engineering, Universiti Sains Malaysia, 14300, Nibong Tebal, Pulau Pinang, Malaysia

Article Info

Article history:

Received Sep 22, 2020
Revised Jan 17, 2021
Accepted Feb 9, 2021

Keywords:

Deep Q network
Deep reinforcement learning
Spike neuron

ABSTRACT

Deep reinforcement learning (DRL) which involved reinforcement learning and artificial neural network allows agents to take the best possible actions to achieve goals. Spiking neural network (SNN) faced difficulty in training due to the non-differentiable spike function of spike neuron. In order to overcome the difficulty, deep Q network (DQN) and deep Q learning with normalized advantage function (NAF) are proposed to interact with a custom environment. DQN is applied for discrete action space whereas NAF is implemented for continuous action space. The model is trained and tested to validate its performance in order to balance the firing rate of excitatory and inhibitory population of spike neuron by using both algorithms. Training results showed both agents able to explore in the custom environment with OpenAI Gym framework. The trained model for both algorithms capable to balance the firing rate of excitatory and inhibitory of the spike neuron. NAF achieved 0.80% of the average percentage error of rate of difference between target and actual neuron rate whereas DQN obtained 0.96%. NAF attained the goal faster than DQN with only 3 steps taken for actual output neuron rate to meet with or close to target neuron firing rate.

Corresponding Author:

Mohamad Khairi Ishak
School of Electrical and Electronic Engineering
Universiti Sains Malaysia
14300, Nibong Tebal Pulau Pinang, Malaysia
Email: khairiishak@usm.my

1. INTRODUCTION

Deep reinforcement learning (DRL) combines machine learning and artificial intelligence techniques [1]. Reinforcement learning algorithms with deep neural networks are implemented in DRL to select the best possible action to attain goals. A DRL agent interacts with a virtual environment as shown in Figure 1 and select actions to solve complex problem [2]. 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. The data consists of states, actions and rewards. The agent takes actions based on the current state and reward in a virtual environment [3]. The agent receives rewards or penalties based on the actions performed. The agent receives positive rewards when the outcome is closer to the target whereas when there is a faulty action taken, the agent obtains negative rewards. The agent learns from experience to decide the best suitable action to attain a goal [4].

Spike neuron is elementary unit in spiking neural network (SNN). Spike neuron has the characteristic of spiking behaviour. When spike neuron is fired,a spike is generated by using spike generation function. Spikes are sequences of action potential that is used in signal transmission in spike neurons [5].

Synapses of spike neuron which consists of excitatory and inhibitory population are required to be optimized before implement into a network to form a SNN. An optimization method is proposed to optimize spike neuron by using maximum likelihood [6]. The maximum likelihood optimization method is used to configure the single M-N neuron to predict the firing activity of the neuron. The average error between the actual and predicted of spike activity is 3%. A supervised multi-spike learning algorithm is proposed to train neurons in SNN [7]. 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 results showed that the algorithm able to achieve classification accuracy. Based on [6] and [7], the gaps can be seen from the need of training data to train the model. Maximum likelihood optimization method and supervised multi-spike learning algorithm required training dataset to train the model. Furthermore, an unsupervised training algorithm is implemented to train SNN [8]. Spike neuron model is trained using synaptic weight association training (SWAT). The training and testing results showed that the algorithm exhibits the capability for classification and convergence accuracy. A limited precision (SNN/LP) supervised learning algorithm of spiking neural networks is implemented in SNN training [9]. Synaptic weights and synaptic delays are applied with limited precision for supervised learning. The algorithm achieved low mean squared error in non-linear XOR classification problem and capable to achieve up to 97% of classification accuracy.

In spiking neural network (SNN), information is emitted and processed by spike neuron through a sequence of action potentials which is also known as spikes [10]. Information is encoded in firing rate of spike neuron [11]. Spike neuron consists of a spike generation function for firing purpose. The spike function is non-differentiable which create a discontinuity at the instance of firing time. Non-differentiability of the function leads to difficulty to develop gradient descent to perform backpropagation in order to update the weight of spike neuron for minimizing loss [12]. This has caused training of SNN using backpropagation become difficult as compared to other artificial neural networks (ANN) [13]. SNN mimics biological nervous system more closely compared to conventional artificial neural networks [14]. Although SNN is biologically more realistic than artificial neural network (ANN) but receives less attention than ANN due to the difficulty to train SNN [15]. 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. 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 [16]. The firing rate of inhibitory population of the spike neuron is initialized as input and adjusted during 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, two algorithms of reinforcement learning are proposed to act as agents which are deep Q network (DQN) and deep Q-learning with normalized advantage functions (NAF) to interact with a custom environment with OpenAI Gym framework to optimize spike neuron into balance state. Other than previous research works that using deep learning or reinforcement learning, this research work applied deep reinforcement learning to solve the difficulty in SNN training by using backpropagation algorithm. The algorithm consists of reinforcement learning algorithm with deep neural network for approximation of Q function. The motivation of this paper is to train single spike neuron using deep reinforcement learning with the absence of training dataset in order to attain goals. The algorithms learn from experience to perform an action to maximize rewards.

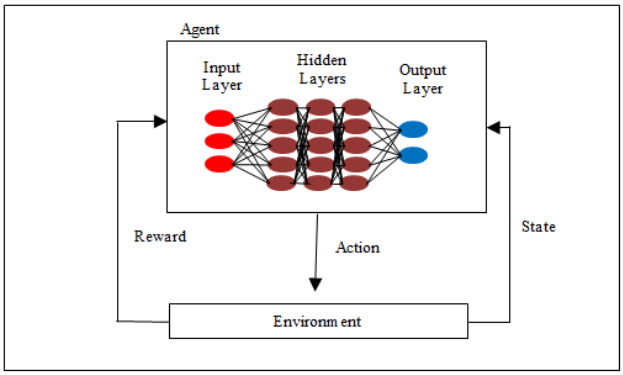


Figure 1. Architecture of Deep Reinforcement Learning

2. RESEARCH METHOD

A spike neuron is created by using neural simulation tool (NEST) simulator. Single spike neuron is used in training for optimization. A spike neuron is modeled by 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 1 [17].

Table 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

A custom environment is created using OpenAI gym toolkit. 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.

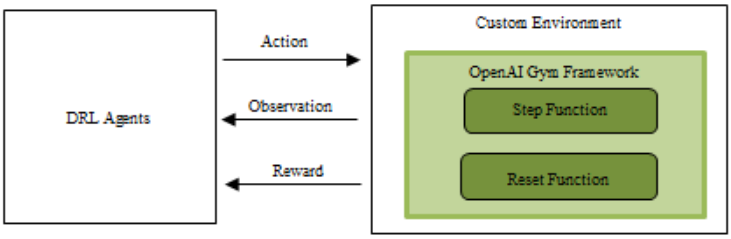


Figure 2. Interaction of DRL agents and a custom environment with OpenAI Gym framework

In DRL algorithm, no training dataset is required as input to provide raw data for training. The DRL agents select the action to be taken without training data. The agents generate their own data according to the given state, actions taken and reward by interacting with the custom environment with OpenAI Gym framework. The training data which also known as experience is stored in memory. The agents learn from experience to make decisions on the action to be taken to obtain the maximum rewards in order to achieve goal [18].

A deep neural network is constructed in DQN as DQN is value-based method. Action space for DQN in the custom environment is discrete type with 4 possible actions. The agent takes actions based on the 4 possible actions defined as shown in Table 2. The agent receives rewards according to the action taken and state. 4000 training steps are taken to balance the firing rate of excitatory and inhibitory population of the spike neuron. The trained model is used for testing to validate the performance for 5 episodes. The flowchart of this algorithm is showed in Figure 3(a).

Table 2. 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

NAF agent is contructed with a state-value function and an advantage function [19]. In NAF, three networks are implemented in training to approximate Q function which are mu_model, V_model and L_model. The V_model is the network used to learn state value function and mu_model is the network applied to select action to be taken that can maximizes Q function. An advantage function is construct in L_model. Action space of NAF algorithm is continuous domain [20-21]. The action value is random selected by the NAF agent from the range of 0 to 50 which is the search interval range. Reward is feedback to the agent from the environment. The training steps is set to 26000 steps to balance the firing rate of exhibitory and inhibitory population of spike neuron. After trained the model, the model is implemented in testing in order to validate the performance for 5 episodes. The flowchart of this algorithm is showed in Figure 3(b).

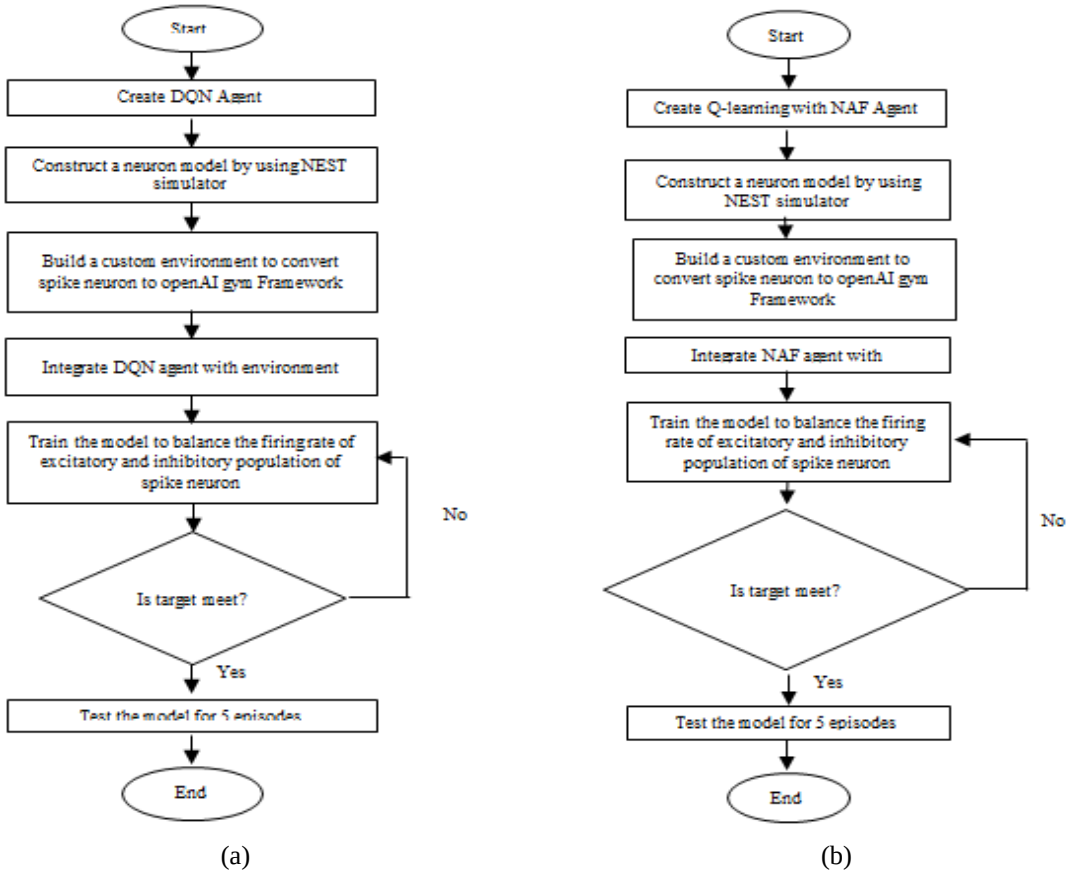


Figure 3. Flowchart of the algorithm, (a) DQN, (b) NAF

3. RESULTS AND DISCUSSION

The spike neuron model is trained in the custom environment built with OpenAI Gym framework. The model is trained until it able to meet the target neuron rate and achieve convergency. When the model is not given enough training, the model is not able to meet the convergence and the goal is unable to attain.

3.1. DQN algorithm

Spike neuron is optimized using DQN with 4000 steps. The DQN agent interacted with the custom environment and selected the actions to be taken. A plot of episode reward versus episodes is generated as shown in Figure 4. The learning curve indicated that the DQN agent capable to explore in the custom environment with OpenAI Gym framework. The trained model able to react towards the custom environment to attain the goal to train spike neuron into balance state. During the initial state, the agents obtained negative reward as the agent is unable to explore in the custom environment with OpenAI Gym framework to optimize spike neuron. During training, the agent learns to explore in the custom environment and receives more rewards. The agent receives positive and negative rewards based on the given state and action taken throughout the training. Each action is selected randomly from 4 discrete actions that defined in action space

of the algorithm. With this capability to explore in the custom environment, the model became usable for testing. After training, the model is tested for 5 episodes to validate the performance of the model. The model received rewards for each episode during testing as shown in Figure 5.

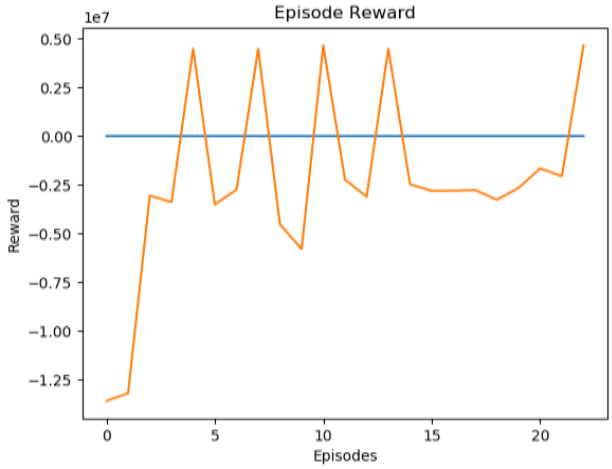


Figure 4. Learning curve of the spike neuron model using DQN algorithm

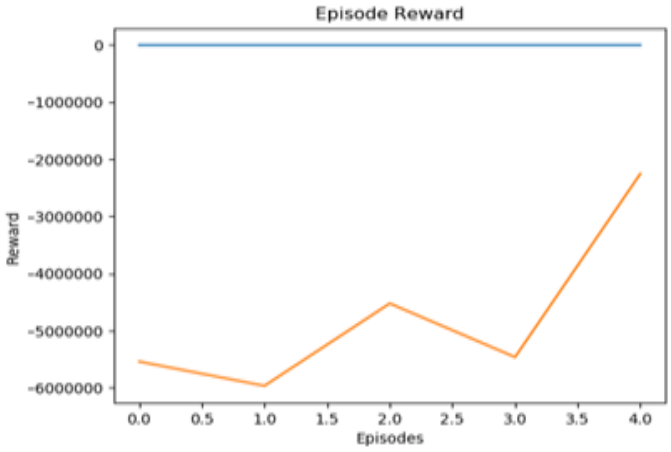


Figure 5. Testing curve of the trained model using DQN algorithm

The inhibitory population rate is fine-tuned by the agent in order to attain the goal. The testing result is tabulated in Table 3. The actual output neuron rate of two episodes are closer to the target neuron firing rate with the difference of 0.04Hz in third and fifth episodes whereas the actual output neuron rate is attained the goal in the fourth episode. In first and second episodes of testing obtained the highest value of difference between actual and target neuron firing rate which is 0.08Hz. The percentage of error between the rate of difference of actual and target output neuron rate and goal is calculated in Table 4. The average percentage of error achieved 0.96%. The lowest steps taken for actual output neuron rate to meet with target neuron firing rate is 84 steps. The result showed that the capability of the trained model to interact with custom environment with OpenAI Gym framework to optimize the firing rate of excitatory and inhibitory population of the spike neuron into balance state.

Table 3. Testing result of the trained model using DQN algorithm

Episode	Simulation Parameter	Optimal Value given by DQN Agent (Hz)	Total Reward	Steps taken to attain goal
1	Mean Rate of Inhibitory Population	20.76	-5540000.00	89
	Initial Rate of Inhibitory Population	18.66		
	Output Neuron rate	5.08		
2	Mean Rate of Inhibitory Population	20.78	-5964000.00	92
	Initial Rate of Inhibitory Population	18.66		
	Output Neuron rate	4.92		
3	Mean Rate of Inhibitory Population	20.81	-4522000.00	91
	Initial Rate of Inhibitory Population	18.66		
	Output Neuron rate	5.04		
4	Mean Rate of Inhibitory Population	20.73	-5462000.00	84
	Initial Rate of Inhibitory Population	18.66		
	Output Neuron rate	5.00		
5	Mean Rate of Inhibitory Population	20.83	-1645000.00	86
	Initial Rate of Inhibitory Population	18.66		
	Output Neuron rate	4.96		

Table 4. Testing result for output and target neuron firing rate in DQN algorithm

Episode	Target neuron firing rate (Hz)	Actual output neuron rate (Hz)	Difference of output and target neuron firing rate (Hz)	Percentage of Error (%)	Average Percentage of Error (%)
1	5.00	5.08	0.08	1.60	0.96%
2		4.92	0.08	1.60	
3		5.04	0.04	0.80	
4		5.00	0.00	0.00	
5		4.96	0.04	0.80	

3.2. NAF algorithm

Spike neuron is optimized using NAF with 26000 training steps. The NAF agent is learnt to interact with the custom environment with OpenAI Gym framework and to select action to be taken to get maximum rewards. A plot of episode reward versus episodes is generated as shown in Figure 6. The learning curve showed the exploration of NAF agent in the custom environment. The result proved that the NAF agent has the capability to explore in the custom environment with OpenAI Gym framework. The trained model capable to react towards the environment to optimize the firing rate of excitatory and inhibitory population of the spike neuron into balance state. The agent obtained positive and negative rewards fluctuately due to the continuous action space. The range of the action value is between 0 to 50. Different action values are being selected randomly for actions taken in training. The model able to perform exploration in the custom environment using NAF algorithm and the model can be applied for testing. 5 episodes of testing is applied to test the performance of the trained model for validation purpose as shown in Figure 7. The testing result is recorded in Table 5.

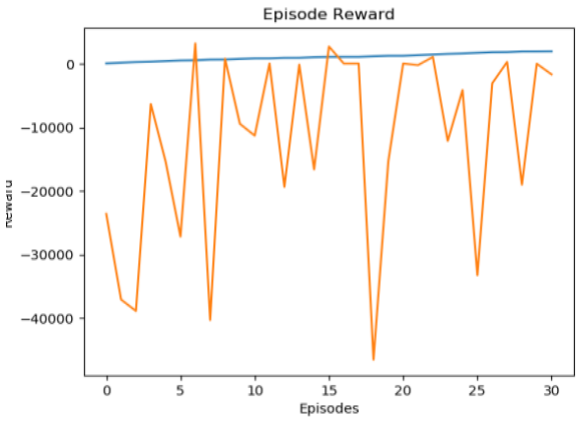


Figure 6. Learning curve of the spike neuron model using NAF algorithm

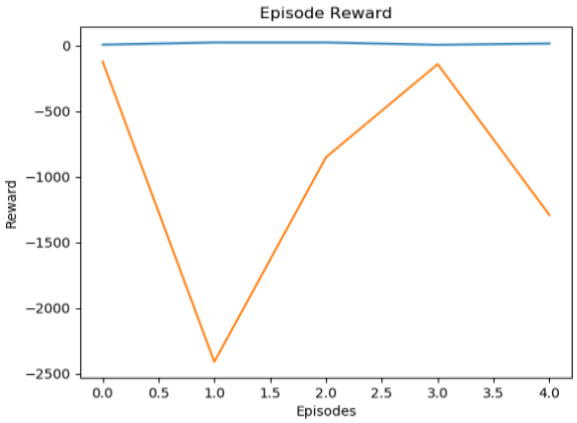


Figure 7. Testing curve of the trained model using NAF algorithm

The firing rate of inhibitory population is fine-tuned by the agent to attain the goal. The percentage of error between the rate of difference of actual output neuron rate and goal is recorded in Table 6. The average percentage of error between rate of difference of output and target neuron firing rate achieved 0.80%. The lowest steps taken for actual output neuron rate to meet with or close to the target excitatory population rate is 3 steps. The result showed the trained model able to interact with the custom environment with OpenAI Gym framework to achieve the balance state of spike neuron.

Table 5. Testing result of the trained model using NAF algorithm

Episode	Simulation Parameter	Optimal Value given by DQN Agent (Hz)	Total Reward	Steps taken to attain goal
1	Mean Rate of Inhibitory Population	20.80	-480.00	11
	Initial Rate of Inhibitory Population	19.23		
	Output Neuron rate	5.08		
2	Mean Rate of Inhibitory Population	20.80	40.00	3
	Initial Rate of Inhibitory Population	19.23		
	Output Neuron rate	5.04		
3	Mean Rate of Inhibitory Population	20.80	-710.00	13
	Initial Rate of Inhibitory Population	19.23		
	Output Neuron rate	4.96		
4	Mean Rate of Inhibitory Population	20.80	-2030.00	21
	Initial Rate of Inhibitory Population	19.23		
	Output Neuron rate	5.00		
5	Mean Rate of Inhibitory Population	20.80	-130.00	8
	Initial Rate of Inhibitory Population	19.23		
	Output Neuron rate	4.96		

Table 6. Testing result for output and target neuron firing rate in NAF algorithm

Episode	Target neuron firing rate (Hz)	Actual output neuron rate (Hz)	Difference of output and target neuron firing rate (Hz)	Percentage of Error (%)	Average Percentage of Error (%)
1	5.00	5.08	0.08	1.60	0.80%
2		5.04	0.04	0.80	
3		4.96	0.04	0.80	
4		5.00	0.00	0.00	
5		4.96	0.04	0.80	

3.3. Evaluation of DQN and NAF algorithm

Spike neuron is optimized to balance the firing rate of excitatory and inhibitory population by using DQN and NAF algorithms in the custom environment with OpenAI Gym framework. The evaluation of the performance of the DQN and NAF trained model is tabulated in Table 7.

DQN algorithm is applied to train the spike neuron in discrete action space whereas NAF algorithm is implemented to train the spike neuron in continuous domain. The types of action space to use depends on the applications. The training steps of DQN is lower than NAF as 4000 training steps are executed on the model and able to meet the goal. The training time is longer for NAF as the model is trained for 26000 steps to attain the goal. The average percentage error of rate of difference between target and actual output neuron firing rate in NAF is lower than DQN. NAF able to achieve 0.80% of percentage error in the testing trained model. Furthermore, steps taken for actual output neuron rate to meet with or close to the target neuron firing rate in NAF is lower compared to DQN which only 3 steps taken to attain the goal. This indicates that NAF algorithm able to optimize spike neuron into balance state faster than DQN.

The performance of DQN and NAF is compared with a previous research work which using maximum likelihood optimization method to optimize spike neuron as shown in Table 8.

Table 7. Evaluation result of DQN and NAF Algorithm

	DQN	NAF
Action Space	Discrete	Continuous
Training steps	4000	26000
Average percentage of error of rate of difference between output and target neuron firing rate	0.96%	0.80%
The lowest steps taken for actual output neuron rate to meet with or close to the target excitatory population rate	84	3
The highest steps taken for actual output neuron rate to meet with or close to the target excitatory population rate	92	21

Table 8. Comparison of Spike Neuron Optimization Method

Method	Average error between actual and target output
Maximum likehood optimization method	3.04%
Deep Q Network (DQN)	0.96%
Deep Q-Learning with Normalized Advantage Function (NAF)	0.80%

Both proposed algorithms achieved lower average error between actual and target output compared to maximum likehood optimization method. The environment used in DQN and NAF are different with the maximum likehood optimization method as the custom environment of DQN and NAF is constructed with OpenAI Gym framework. The environment for DQN and NAF is customized in order to ensure both agents is capable to explore in the environment in order to optimize spike neuron.

4. CONCLUSION

Deep reinforment learning is proposed as a method to overcome the difficulty in SNN training due to non-differentiable of spike function of spike neuron. Deep Q network and Deep Q-learning with normalized advantage functions algorithms are proposed to balance the firing rate of excitatory and inhibitory population of a spike neuron. A spike neuron is trained in the custom environment with OpenAI Gym framework. Both algorithms able to interact with the custom environment with OpenAI Gym Framework to attain the goal. The average percentage error of rate of difference between target and actual output neuron firing rate for NAF and DQN algorithms obtained 0.80% and 0.96% respectively. In terms of steps taken for actual output neuron rate to meet with the target neuron firing rate, NAF achieved faster than DQN to meet the target neuron firing rate. The results proved that the algorithms able to explore in the custom environment to optimize the spike neuron. In future work, DQN and NAF algorithm can be used for further development to train a spiking neural network (SNN) since both algorithms are capable to explore in the custom environment with OpenAI Gym framework by using DRL to optimize a spike neuron. The developed SNN can be demonstrated in various types of applications such as playing game, classification, image recognition and so on.

ACKNOWLEDGEMENTS

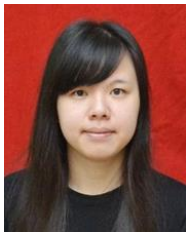
The authors would like to thank the Universiti Sains Malaysia for financial support through the Research University Grant Scheme PELECT /1001/8014049.

REFERENCES

[1] V. Francois-Lavet, *et al.*, "An Introduction to Deep Reinforcement Learning," *Foundation and Trends® in Machine Learning*, pp. 219-354, 2018.
[2] K. Arulkumaran, *et al.*, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, pp. 26-38, 2017. DOI: 10.1109/MSP.2017.2743240.
[3] R. Sutton, and A. Barto, "Reinforcement Learning: An Introduction," *IEEE Transactions on Neural Networks*, p. 1054, 1998.
[4] M. Pieters and M. A. Wiering, "Q-learning with experience replay in a dynamic environment," *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Athens, pp. 1-8, 2016. DOI: 10.1109/SSCI.2016.7849368.
[5] H. Jang, *et al*, "An Introduction to Probabilistic Spiking Neural Networks: Probabilistic Models, Learning Rules, and Applications," in *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 64-77, Nov. 2019. DOI: 10.1109/MSP.2019.2935234.
[6] A. Russell*et al.*, "Optimization methods for spiking neurons and networks," *IEEE transactions on neural networks*, vol. 21, no. 12, pp. 1950-1962, 2010. DOI: 10.1109/TNN.2010.2083685.
[7] Y. Miao, *et al.*, "A supervised Multi-Spike Learning Algorithm for Spiking Neural Networks*," 2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-7, 2018. DOI: 10.1109/IJCNN.2018.8489175.
[8] J. Wade, *et al.*, "SWAT: An Unsupervised SNN Training Algorithm for Classification Problems," *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 2648-2655, 2008. DOI: 10.1109/IJCNN.2008.4634169.
[9] E. Stromatias and J. Marsland, "Supervised learning in Spiking Neural Networks with Limited Precision: SNN/LP," *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-7, 2015. DOI: 10.1109/IJCNN.2015.7280732.
[10] S. Park, *et al.*, "Fast and Efficient Information Transmission with Burst Spikes in Deep Spiking Neural Networks," *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1-6, 2019.

[11] S. Shrestha and G. Orchard, "SLAYER: Spike Layer Error Reassignment in Time," *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.
[12] Y. Oniz, *et al.*, "Spiking Neural Networks for the Control of a Servo System," *2013 IEEE International Conference on Mechatronics (ICM)*, pp. 94-98, 2013. DOI: 10.1109/ICMECH.2013.6518517.
[13] Y. Wu, *et al.*, "Spatio-Temporal Backpropagation for Training High Performance Spiking Neural Networks," *Frontiers in Neuroscience*, 2018. https://doi.org/10.3389/fnins.2018.00331.
[14] A. Abusnaina, *et al.*, "Supervised Training of Spiking Neural Net-work by Adapting the E-MWO Algorithm for Pattern Classification," *Neural Process Letters*, pp. 661-682, 2019. https://doi.org/10.1007/s11063-018-9846-0.
[15] R. Abiyev, *et al.*, "Spiking Neural Networks for Identification and Control of Dynamic Plants," *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1030-1035, 2012. DOI: 10.1109/AIM.2012.6265983.
[16] B. Chandra and K. V. N. Babu, "A new spiking neuron model," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-5, 2013. DOI: 10.1109/IJCNN.2013.6706930.
[17] J. Eppler, *et al.*, "PyNEST: a convenient interface to the NEST simulator," *Frontiers in neuroinformatics*, 2008. DOI: 10.3389/neuro.11.012.2008.
[18] Y. Wang and Z. Zhang, "Experience Selection in Multi-agent Deep Reinforcement Learning," *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 864-870, 2019. DOI: 10.1109/ICTAI.2019.00123.
[19] C. Qi, *et al*, "Deep Reinforcement Learning With Discrete Normalized Advantage Functions for Resource Management in Network Slicing," in *IEEE Communications Letters*, vol. 23, no. 8, pp. 1337-1341, Aug. 2019.
[20] P. Tuyen, *et al.*, "Deep Reinforcement Learning Algorithms for Steering a Underactuated Ship," *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2017)*, 2017. https://doi.org/10.1109/MFI.2017.8170388.
[21] O Ali, and MK Ishak. "Bringing intelligence to IoT Edge: Machine Learning based Smart City Image Classification using Microsoft Azure IoT and Custom Vision," *Journal of Physics: Conference Series*, 2020. DOI: 10.1088/1742-6596/1529/4/042076.

BIOGRAPHIES OF AUTHORS



Tan Szi Hui received B. Sc. degree in electronic and computer engineering from Universiti Teknikal Malaysia Melaka in 2019, and the M. Sc. degree in embedded system engineering from Universiti Sains Malaysia in 2020.

Mohamad Khairi Ishak received the B.Eng degree in Electrical and Electronics Engineering from the International Islamic University Malaysia (IIUM), Malaysia, the MSc. in Embedded System, from the University of Essex, United Kingdom and PhD from the University of Bristol, United Kingdom. He is a member of IEEE and a registered graduate engineer with the Board of Engineers Malaysia (BEM). Currently, he is a Senior Lecturer in Mechatronics Engineering at School of Electrical and Electronic Engineering, Universiti Sains Malaysia (USM). His research interests are Embedded System, Real-Time Control Communications and Internet of Things (IoT). Emphasis is given towards the development of theoretical and practical methods which can be practically validated. Recently, significant research effort has been directed towards important industrial issues of embedded networked control systems and IoT.