Exploration Of Encoding And Decoding Methods For Spiking Neural Networks On The Cart Pole And Lunar Lander Problems Using Evolutionary Training

Andrew W. Rafe
School of Computer Science
University of Technology Sydney
Sydney, Australia
andrew.w.rafe@student.uts.edu.au

Jaime A. Garcia
School of Computer Science
University of Technology Sydney
Sydney, Australia
jaime.garcia@uts.edu.au

William L. Raffe
School of Computer Science
University of Technology Sydney
Sydney, Australia
william.raffe@uts.edu.au

***Abstract*—Spiking Neural Networks are increasingly drawing interest due to their potential for large efficiency g ains when used with neuromorphic computers. However, when attempting to replicate the successes of Artificial Neural Networks, challenges are faced due to their vastly different architectures and therefore differing methods for training and optimisation. There has been minimal analysis of the differences between encoding and decoding methods and the effect of state space exposure periods on the performance of these networks. The core contribution of this paper is the detailed analysis of decoding methods, state exposure periods, and a learned input encoding method of an evolved Spiking Neural Network within the Reinforcement Learning context. This is demonstrated using the Cart Pole and Lunar Lander Reinforcement Learning problems. The paper discovers a negative correlation between the generation to reach the goal and the state space exposure period over all decoding methods tested. The state exposure period is also found to influence t he number of random actions taken due to the decoding methods being unable to select an action. This paper explores the differences in temporal and rate-based decoding as well as identifying benefits i n r esetting n etworks t o t heir d efault s tates between episode steps. Additionally, the novel input encoder, is effective at pre-processing state information using the same evolutionary algorithm as the rest of the network.**

***Index Terms*—genetic algorithm, spiking neurons, spiking neural network, spike train, reinforcement learning**

I. Introduction

Spiking Neural Networks (SNNs) are an alternative structure to traditional Artificial Neural Networks (ANNs). The networks themselves behave in a very different way, even though their structures appear similar. SNNs take their inspiration from the low-level cognitive structure in animal brains and act in a biologically similar way through binary spikes passed from one neuron to the next through weighted connections. Neurons will build in some potential until such a point that a threshold is surpassed where a spike will occur and be passed down its connections, which will work to build or diminish the potentials of those connected neurons [1]. Once a neuron has spiked, it will enter a period of refractory where it will require more input than usual to spike again. Mapping neurons spiking over time produces a spike train. This temporal information on the activity of a neuron can be used to extract information on action selection or classification when they occur in some output layer.

SNNs are not intrinsically differentiable and therefore in their basic form are unable to undergo the process of back-propagation. Wu et al. [2] was able to overcome the non-differentiable nature of the Leaky Integrate and Fire (LIF) neuron model by approximating a derivative based on spike characteristics such as neuron potentials and spike frequency. Lee et al. [3] using a similar method was able to, with a convolutional spiking neural network, outperform a convolutional ANN in the MNIST digit dataset [4]. By estimating derivatives, gradient descent methods can still be used in order to optimise the weighted connections.

Evolutionary and genetic algorithms on neural networks have been used to train game-playing agents. Their use on SNNs is demonstrated by Yee and Teo [5] through their application to a driving simulator called TORCS, whereby they were able to train an agent to drive three increasingly complex tracks. Markowska-Kaczmar and Koldowski [6], in their use of an SNN for a top-down racing simulator, use an evolutionary algorithm to optimise the Izhikevich (IZ) [7] neuron model parameters. They trained both an ANN and SNN on increasingly complex tracks and even though the SNN was able to achieve the goal on each of the problems, the average fitness of the SNNs was always lower than the ANN counterparts. In addition to game playing agents, evolutionary based reinforcement learning has been used by Slade and Zhang [8] in their attempt to utilize a topological and weight evolution on a SNN to implement the XOR logical gate. Although their results were promising in that they were able to create systems that could classify according to XOR, the size and complexity of the networks created were worse than ANN implementations.

Although these papers demonstrated the effectiveness of using evolutionary algorithms for SNNs, different decoding and state exposure periods[1] were not compared leading to some confusion over why particular decoding methods or exposure periods were selected.

The major aim of this paper is to compare rate based and temporal decoding methods over a range of different exposure periods when applied to the evolutionary training of a SNN in the Cart Pole and Lunar Lander problems as provided by the Open AI Gym [9]. In doing so, a novel input encoder is proposed for the encoding of the state space information into a spike train using learned weights. This is significant as both input preprocessing and network training use the same evolutionary process.

II. Network Overview

The extent to which realism of cognitive neuronal level structure is replicated in artificial systems brings about several methods of modeling these types of networks. Generally, the more realistic the neurons and synapses in the network are, the more computation is required during processing. The most general and computationally fastest method is using the Leaky Integrate and Fire (LIF) neuron model [1]. The most accurate model originates from the work by Hodgkin and Huxley (H-H) [10] through their study of membrane capacitance at the ionic level. Due to the complexity of this model, it is considered the most computationally inefficient. The Izhikevich model (IZ) [7] works to bridge the realism and computational efficiency gap between the LIF and H-H models and this is the model that will be used in this paper.

A. SNN Modeling Method

The IZ model revolves around two basic differential equations.

$$v' = \begin{cases} 0.04v^2 + 5v + 140 - u + I & \text{if } v < 30 \\ c & \text{if } v >= 30 \end{cases} \quad (1)$$

$$u' = \begin{cases} a(bv - u) & \text{if } v < 30 \\ d & \text{if } v >= 30 \end{cases} \quad (2)$$

I refers to the injection into the neuron, v is the membrane potential, and u is the recovery variable. When v exceeds the peak voltage of 30mV, both the membrane potential ,v, and the recovery variable, u, resets according to specified rules outlined in 1 and 2 respectively. This causes a spike to register, and it is passed down the weighted connections and causes an injection into that connected neuron. The other four variables, namely a, b, c and d, are the parameters that affect the behaviour of the neuron. Firstly a describes the time scale of the recovery variable with smaller values resulting in slower recovery. Secondly, b describes the sensitivity of the recovery variable to changes in the membrane potential of the neuron before an action potential takes place [7]. Thirdly, c determines the reset value after a spike occurs. Finally, d describes the behaviour of the reset of the recovery variable post spike. Typical values found in regular spiking neurons [7] are $a = 0.02$, $b = 0.2$, $c = -65$ and $d = 8$ and these are the values that will be used in this experiment.

These IZ neurons are connected to each other with weighted connections. Negative weighted connections lead to the depressing of possible injections into the next layer neurons whereas positive weighted connections encourage the next layered neuron to fire by injecting some charge into the membrane. When a neuron undergoes an action potential (spike) the value of the outgoing weight from that neuron is injected into the connected neuron. For all of the tests conducted, the IZ neuron model is used to construct the SNNs.

B. Input Encoder

One of the major differences with SNNs compared to ANNs is how information is parsed into and extracted from the network, otherwise known as encoding and decoding respectively. For encoding, the state information from the reinforcement learning problem must be converted into a spike train to be fed into the network. The original Cart Pole and Lunar Lander state spaces are made up of positive and negative input values often denoting direction where negative is positional or velocity in the left direction and positive in the right direction. The use of negative inputs with SNNs is not a well documented and researched issue as the majority of uses of SNNs have been in image classification and object detection whereby negative inputs based on pixel values never occur. Therefore, for each of the inputs where a negative value is possible, these have been split into two input values where one represents the left direction, and the other the right, otherwise referred to as double encoding [11] [6]. If the original input value was negative then the left direction input would be the positive magnitude of the original value and the right direction input would be zero. If the input was positive then the left direction input would be zero and the right direction input would be the same value as the original input value.

A similar issue with the inputs of both problems when compared to pixel input information is the non-normalized nature of them. Where pixel information is often represented with inputs between 0.0 and 1.0, these could be scaled equally among all inputs to encourage spiking behaviour in the input layer. However both of the problems have differently scaled inputs, some of which often become greater than 1 and some that never cross a threshold of 0.1. Therefore scaling them in a uniform way leads to unequal importance being placed on particular input values. Therefore an additional input encoder layer was added to the network which acts as a simple multiplier of the inputs by some learned weight and injected into the first SNN layer of the network. This mapping of input encoder to SNN first layer nodes is one to one, as in each initial layer SNN node has one input encoder node which is directly attached to it with some weight value as can be seen in Figure II-B. This is the scale that the input is changed by

[1]The length of the input spike train into the SNN, also can be described as the number of iterations of the SNN that a single state is exposed to it for.

before being injected into the SNN. This is not being counted as a layer in the network as it is simply a preprocessing step of the input before being fed into the network however the weights of the input encoder are learned through the same evolutionary process used by the SNNs.

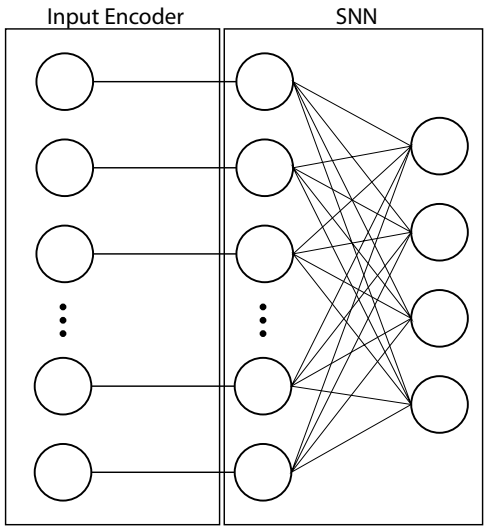

Fig. 1. An example of the input encoders connection to the SNN. In this example, the SNN has no hidden layer as the Input Encoder layer are not made up of IZ neurons and simply act as a preprocessing step for state space information.

C. Action Decoding

The resultant spike trains of the output neurons need to be processed and an action selected based on that. Decoding methods are generally broken into two categories; the first being rate based decoding whereby the rate of spiking of an output neuron is related to the selection of an action, and temporal coding in which the precise spike timings of the output neurons are relied upon to make a decision. For each of the described action decoding methods, there is an alternate reset method which after each of the episode steps and after an action has been selected, the network is reset to its default state.

1) Rate Based Decoding: Rate coding methods rely on the fact that the carriage of information is based on the frequency of spikes occurring in a neuron [12].

Temporal averaging is a method whereby the average firing rate is calculated for an individual neuron based on the time of exposure of the particular stimulus referred to in this paper as the state exposure period. This can be defined by Equation 3.

$$R = \frac{n^{sp}}{T} \tag{3}$$

Where R is the average firing rate of a single neuron over a single trial, n^{sp} is the number of spikes over the time of exposure of a particular stimulus and T is the total time of exposure of the stimulus. The output neuron that produces the largest of these temporal averages will be used for action selection. In order to produce more accurate temporal averaging, the time of exposure of the stimulus can be increased. It needs to be sufficiently long to be able to infer relevant information. This method can experience issues with the possibility of the same temporal average over multiple output neurons especially when the time of exposure of the stimulus is small. In this case a random action is selected from the set of actions that have the highest firing rates over that exposure period.

Although not tested in this paper, state exposure to the network over multiple trials can be conducted and the rate of firing of the single neuron can be calculated. The noisy nature of the input to the network may make spike rates differ from one trial to the next. Equation 4 demonstrates this idea.

$$R = \frac{\Sigma_{i=1}^{k} n_i^{sp}}{Tk} \tag{4}$$

The variable k in this instance refers to the number of trials of an exposure of a stimulus to a single neuron. In real neural systems it is not possible for this to be the coding method for deciding action selection as a decision is made with only one exposure to an event. However in artificial systems this can be used to ensure the correct action is taken when using rate coding. This is a slow process due to the repeated trials of multiple exposures of a stimulus. Population averaging is similar to that of the temporal averaging over trials method however it is based on the assumption that there is a pool of neurons responsible for action selection. If we find the average rate of firing over that population of neurons we can get a more accurate reading of which pool is being the most stimulated over a certain time period. The regular temporal averaging method referred to as rate decoding and described in Equation 3 and its reset counterpart will be analysed in this paper.

2) Temporal Decoding: Temporal decoding assumes that the carriage of information is dependant on the precise spike timings [12]. This means exposure of a state for a fixed time period is not needed in order to appropriately select an action to take. First-to-Fire (F2F) decoding selects the action based on the neuron that was first to fire since the exposure of the new state to the network [13]. This improves on the number of exposures needed in comparison to the rate decoding method due to the action being selected as soon as a single spike occurs in the output layer. In the case where no output neuron fires within the exposure period, a random action is selected. If two or more output neurons spike at the same time then a random action is selected from those output neurons that fired. The F2F and the F2F Reset methods are analysed in this paper.

III. Evolution of Networks

For each of the experiments conducted, a population of 50 initially random agents were produced with weight values ranging from -20.0 to 80.0 for Izhikevich neurons and input encoder layer weights ranging from 0.0 to 150.0. Each trial had a maximum of 100 generations or would be concluded when the problem solution requirements were met. Each

agent conducted 5 separate episodes and the average of all episode rewards was the fitness of that agent for that particular generation. If any of the networks achieved the goal over the 5 episode average, that network was run through 100 consecutive episodes to determine if the overall goal was achieved as set out by the problems. If it was, then the experiment was concluded otherwise the evolutionary process would continue. After all agents conducted their individual trials, the top 10 agents were prevented from undergoing any evolutionary changes. For the remaining 40 agents, their weights were replaced by the random crossover of two of the top 10 agents. Each weight was replaced by either the first parent ($P = 0.5$) or second parent ($P = 0.5$) weight value. They then underwent a random mutation phase where each weight in the network was transformed according to Equations 5, 6 and 7.

$$X \sim U(0.0, 1.0) \tag{5}$$

$$w' = \begin{cases} w + 10\alpha & \text{if } 0.00 \leq X < 0.01 \\ w - 10\alpha & \text{if } 0.01 \leq X < 0.02 \\ w + \alpha & \text{if } 0.02 \leq X < 0.12 \\ w - \alpha & \text{if } 0.12 \leq X < 0.22 \end{cases} \tag{6}$$

$$\alpha' = \alpha\lambda \tag{7}$$

X refers to some mutation chance value selected from a uniform distribution, w' is the new transformed weight and w is the weight after crossover, α is the learning rate which is 10.0 at the beginning of training, α' is the new learning rate after decay λ which is constant at 0.99. X is calculated separately for each weight. These values were selected empirically as a result of conducting a range of hyperparameter experiments. These experiments also demonstrated a benefit in using a decaying learning rate. For each network, each weight is updated according to Equation 6 once per generation. The learning rate update from Equation 7 occurs once per generation.

IV. Experiment Setup

In order to analyse the different decoding methods and the input encoder, the Cart Pole (CartPole-v1) and Lunar Lander discrete (LunarLander-v2) problems were used from the Open AI Gym architecture [9]. These two problems were chosen based on their different complexities to analyse any trends emerging with the various tested methods. Cart Pole, the simplest of the two problems, was used due to the ability of using single layer perceptrons to solve, meaning that single layer network trends could be analysed. The Lunar Lander problem, traditionally solved using Deep Q learning methods, generally requires a multi-layer perceptron to solve and therefore can be used to verify any trends emerging with SNNs using hidden layers. For both Cart Pole and Lunar Lander, a random benchmark trial was run as the control. This random benchmark involved running agents in batches identical to those used in the evolutionary trials whereby over 50 generations, 50 batches of 5 episodes were run with an agent that selects a random action at each episode step. Over the single generation, the best average fitness of a 5 episode run was recorded as the fitness of that generation.

TABLE I
THE TRANSFORMED STATE SPACE FOR CART POLE TO REMOVE NEGATIVE INPUT VALUES.

Input	Description	Min	Max
0 (1)	$X + (X-)$	0	4.8
2 (3)	$v_x + (v_x-)$	0	∞
4 (5)	$\theta + (\theta-)$	0	0.13π
6 (7)	Pole Velocity at Tip + (-)	0	∞

For both the Cart Pole and Lunar Lander problems, exposure periods of 20, 40, 60, 80 and 100 were used. Networks with and without hidden layers were tested. For those multi-layered networks, a single hidden layer of 16 and 32 neurons was used. All networks used the Input Encoder method and F2F, F2F Reset, Rate and Rate Reset methods were used for decoding. Three separate trials were conducted for each of the decoding methods and each of the exposure periods and the average fitness of each generation were used for analysis.

A. Cart Pole Problem

The goal of the cart pole problem, also known as the inverted pendulum problem, is to balance an upright pole on a cart that can either move left or right at each step in the episode. The episode is terminated if the angle of the pole is greater than 12 degrees from the Y axis, the cart reaches further than 2.4 units from the origin point or the episode length is greater than 200 steps. A reward of 1 is received by the agent for each step of the episode including the terminating step. The problem is considered solved if an agent receives a total episode reward of greater than or equal to 195 over 100 consecutive episodes. The Cart Pole environment produces a state space of four float values representing the cart position from center point, the cart velocity, the pole angle and the current velocity of the tip of the pole. Negative state space values represent the current state in terms of movement to the left and positive to the right. With an Izhikevich neuron model, negative inputs mean that the input neuron will rarely, if ever spike and will therefore not pass this information onto future layers. It is therefore required to eliminate negative values from the state space. As a result, the four value state space was transformed into eight values where the positive and negative representations of each input are split into two neurons and transforming them both into positive values as seen in Table I which is commonly referred to as double encoding [11] [6].

B. Lunar Lander Problem

The Lunar Lander problem requires the landing of a lunar module in a 2D environment within a landing zone. The landing zone is always in the same position however the surrounding landscape is procedurally generated in each episode. It includes four actions being the left, right an main thrusters

TABLE II
THE TRANSFORMED STATE SPACE FOR LUNAR LANDER TO REMOVE NEGATIVE INPUTS.

Input	Description	Min	Max
0 (1)	$X + (X-)$	0	∞
2 (3)	$Y + (Y-)$	0	∞
4 (5)	$v_x(v_x-)$	0	∞
6 (7)	$v_y(v_y-)$	0	∞
8 (9)	$\theta + (\theta-)$	0	π
10 (11)	$\omega + (\omega-)$	0	∞
12 (13)	Left (Right Leg Contact)	0	1

as well as an action for do nothing. The episode terminates if the lander crashes or comes to rest. A reward of 200 or greater over 100 consecutive episodes is required for the trial to be a success. Around 100 to 140 reward comes from moving from the top of the screen to the bottom and also for having zero speed. An additional 10 reward points come from each leg making contact with the ground. If the lander body touches the ground it is considered crashed and gets -100 reward and if both legs are in contact and the body is at rest, an additional 100 points are awarded. Similarly with the Cart Pole problem, input values that could be negative or positive were transformed into two separate inputs based on the negative and positive component as described in Table II.

V. RESULTS

Over all four decoding methods, all exposure period intervals and over both the single and multi-layered network combinations, the SNN was able to produce goal achieving agents in the Cart Pole problem as shown in Table III. For single layer networks, there is no significant difference in the generations needed to reach the goal across all four decoding methods. The only slight difference appears to be in a better capacity for F2F and F2F Reset methods to solve the problem in fewer generations in lower exposure periods. When looking at an exposure period of 20, F2F and F2F reset were able to solve the problem in 14 and 16 generations respectively where as Rate and Rate Reset decoding methods took 31 and 18 generations respectively. This is largely due to Rate based decoding methods at lower exposure periods tending not to produce multiple spikes per output neuron and therefore a calculation of the rate of firing of the neurons often results in the taking of a random action due to the rates across both output neurons being the same. As greater exposure periods are used, the output neurons are more likely to produce more variance in the rate of spiking of the output neurons and that disadvantage goes away.

When looking at the number of random moves taken by a goal achieving agent, as demonstrated in Figure 2, it is clear that the longer the exposure period, the less random actions that are taken. Random actions occurred if the decoding method over the exposure period was not able to determine an action to take. The occurrence of random actions is far fewer when using temporal decoding methods like

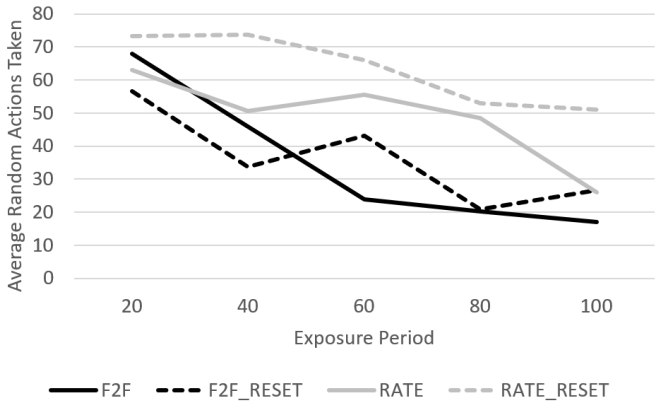

Fig. 2. The number of random actions taken on average for each of the exposure periods tested and decoding methods used for single layer, goal achieving networks in the Cart Pole problem.

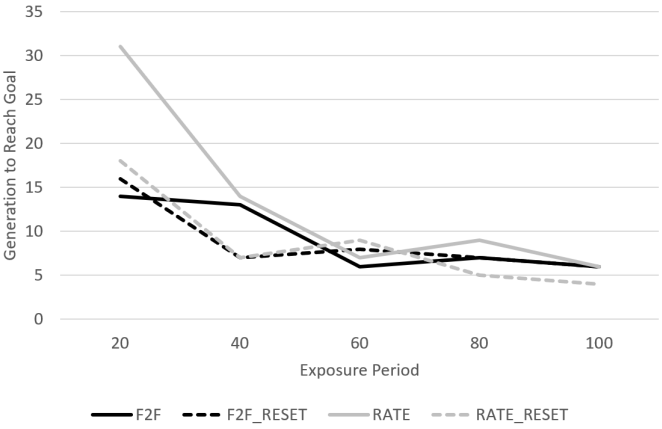

Fig. 3. The generation to reach the goal for single layered networks for each of the decoding methods and exposure periods in the Cart Pole problem.

F2F and F2F Reset in higher exposure periods. For lower exposure periods, occurrences of random action taking is fairly consistent between all decoding methods with rate based decoding selecting random actions only slightly more often than temporal decoding.

As is demonstrated in Table IV, a clear negative correlation has emerged between the exposure period and the number of generations needed to reach the goal in the Cart Pole problem. This negative correlation is consistent across all decoding methods and network structures used in this experiment. This trend appears to level off at the higher exposure periods indicating that there may be an optimal exposure period for this problem and introduces a trade off between the execution time of the network and the number of generations that will be needed to reach the goal. The larger the exposure period, the more execution time is needed as more network steps are required per episode step. This levelling off appears to occur at around the 60 exposure period mark as evidenced in Figures 3,4 and 5.

TABLE III

TABLE OF RESULTS FOR CART POLE EXPERIMENT SHOWING THE HIGHEST AVERAGE FITNESS ACHIEVED AND THE GENERATION IT ACHIEVED THE GOAL BY EACH DECODING METHOD, EXPOSURE PERIOD AND NETWORK STRUCTURE WITH A GOAL FITNESS OF GREATER THAN 195.0.

		Highest Achieved Fitness / Solved Generation			
Decoding Method	**Exposure Period**	**No Hidden**	**16 Hidden**	**32 Hidden**	**Random**
F2F	20	196.80 / 14	196.82 / 37	196.55 / 45	
	40	197.10 / 13	195.27 / 26	196.41 / 61	
	60	**197.00 / 6**	**196.00 / 21**	197.31 / 36	
	80	196.28 / 7	195.36 / 23	**197.07 / 30**	
	100	**197.87 / 6**	195.47 / 24	197.25 / 31	
F2F RESET	20	195.75 / 16	199.49 / 21	195.47 / 18	
	40	198.07 / 7	**197.52 / 11**	196.52 / 20	
	60	197.1 / 8	198.13 / 14	195.42 / 17	
	80	195.56 / 7	196.22 / 17	**197.27 / 12**	
	100	**198.54 / 6**	**199.52 / 11**	195.48 / 17	47.0667 / -
RATE	20	195.31 / 31	195.54 / 56	195.58 / 31	
	40	195.45 / 14	195.26 / 23	198.27 / 22	
	60	195.27 / 7	196.38 /23	196.85 / 26	
	80	196.16 / 9	195.08 / 19	196.43 / 19	
	100	**197.55 / 6**	**195.00 / 12**	**195.22 / 18**	
RATE RESET	20	196.92 / 18	197.20 / 20	196.84 / 22	
	40	197.25 / 7	197.60 / 24	196.81 / 21	
	60	198.00 / 9	197.91 / 17	195.31 / 17	
	80	196.72 / 5	**198.13 / 12**	195.40 / 14	
	100	**195.09 / 4**	197.87 / 16	**197.27 / 12**	

TABLE IV

CORRELATION COEFFICIENTS FOR CART POLE PROBLEM BETWEEN THE GENERATION THAT EACH TRIAL REACHED COMPLETION AND THE EXPOSURE PERIOD USED FOR THAT TRIAL.

	NO HIDDEN	**16 HIDDEN**	**32 HIDDEN**
F2F	-0.88	-0.73	-0.73
F2F RESET	-0.77	-0.52	–0.54
RATE	-0.84	-0.85	-0.86
RATE RESET	-0.85	-0.70	-0.99

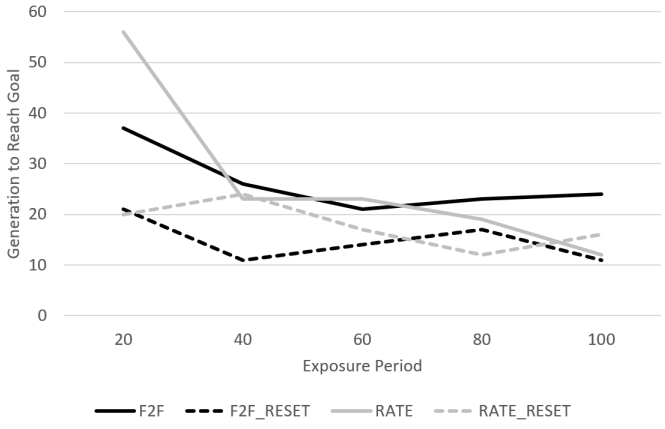

Fig. 4. The generation to reach the goal for multi-layered networks with 16 hidden neurons for each of the decoding methods and exposure periods in the Cart Pole problem.

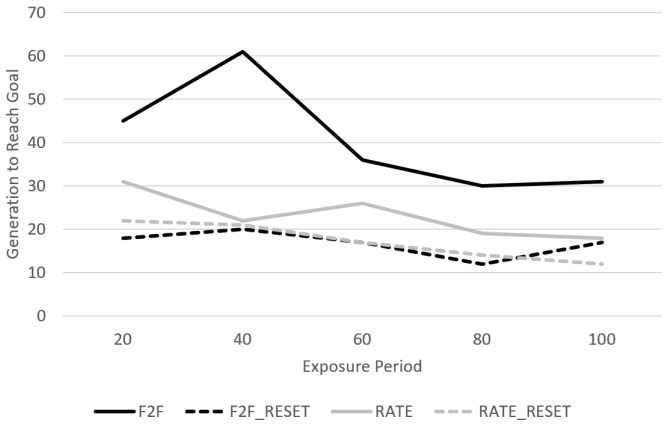

Fig. 5. The generation to reach the goal for multi-layered networks with 32 hidden neurons for each of the decoding methods and exposure periods in the Cart Pole problem.

For the Lunar Lander problem, no multi-layer network was able to achieve the goal with fitness rarely reaching above 100. The F2F Reset and Rate Reset decoding methods were able to achieve the highest average fitnesses across all exposure periods when compared to the other decoding methods as can be seen in Table V. The single layer networks came close to achieving the goal state in almost all of the decoding methods and exposure periods utilised and was able to averagely achieve the goal in one occurrence with the use of the Rate Reset decoding method at 60 exposure periods. The fact that this occurred at generation 92 out of the tested 100 generations indicates that given more time, other network structures may

TABLE V
TABLE OF RESULTS FOR LUNAR LANDER EXPERIMENT SHOWING THE HIGHEST AVERAGE FITNESS ACHIEVED AND THE GENERATION IT ACHIEVED THE GOAL (IF GOAL WAS ACHIEVED) BY EACH DECODING METHOD, EXPOSURE PERIOD AND NETWORK STRUCTURE WITH A GOAL FITNESS OF GREATER THAN 200.0.

		Highest Achieved Fitness / Solved Generation			
Decoding Method	**Exposure Period**	**No Hidden**	**16 Hidden**	**32 Hidden**	**Random**
F2F	20	190.93 / -	33.59 / -	-6.33 / -	-71.3948 / -
	40	184.24 / -	27.09 / -	-32.34 / -	
	60	**192.98 / -**	6.04 / -	**44.16 / -**	
	80	89.18 / -	**35.51 / -**	25.56 / -	
	100	136.17 / -	20.68 / -	27.15 / -	
F2F RESET	20	148.79 / -	26.90 / -	8.49 / -	
	40	**199.31 / -**	**157.42 / -**	**45.50 / -**	
	60	194.85 / -	63.52 / -	-2.01 / -	
	80	176.84 / -	129.51 / -	20.32 / -	
	100	188.74 / -	156.28 / -	29.39 / -	
RATE	20	172.08 / -	13.42 / -	16.52 / -	
	40	**192.04 / -**	-9.42 / -	1.87 / -	
	60	184.68 / -	10.44 / -	9.21 / -	
	80	59.78 / -	**70.07 / -**	**67.91 / -**	
	100	128.73 / -	44.90 / -	2.15 / -	
RATE RESET	20	185.46 / -	77.04 / -	51.16 / -	
	40	57.47 / -	**131.91 / -**	**68.64 / -**	
	60	**202.26 / 92**	54.72 / -	64.59 / -	
	80	167.50 / -	52.84 / -	53.97 / -	
	100	133.73 / -	41.02 / -	61.37 / -	

have been able to solve the problem. Introducing more neurons in the multi-layer networks meant that the highest achieved fitness over those 100 generations were consistently lower in networks with more neurons. This is expected as more neurons means more synapse weights to learn and more generations that are needed to improve fitness.

VI. DISCUSSION AND FUTURE WORK

Possible trends emerged in the negative correlation between the exposure period and the generations taken to reach a certain fitness when looking at the results of the Cart Pole experiment. This suggests that minimising the chance of random actions being chosen due to none of the output neurons firing in the output layer or, in the case of rate decoding, the same number of spikes over more than one output neuron over the exposure period, hinders the rate of learning in the network. Due to the failure of most Lunar Lander trials to reach the goal, it makes it difficult to confirm this trend and therefore requires more investigation. Similarly, the number of trials and exposure periods tested prevents strong conclusions being made about this correlation. It is worth expanding the number of trials and exposure periods tested to formulate better evidence of this. Additionally, investigating whether this trend continues in non-evolutionary training methods such as with Hebbian learning processes [14] including Reward based Spike Timing Dependant Plasticity [15] and with alternate reinforcement learning problems. This would determine whether this trade off between network efficiency, by lowering the number of exposure periods, and network accuracy by increasing the number of exposure periods is a universal trait of SNNs.

The input encoder proposed in this paper was successful at learning state space data preprocessing before being fed into the IZ SNN for the Cart Pole problem and for some individual Lunar Lander trials when using all tested decoding methods. This is significant for evolutionary learning as it can use the same algorithm for both learned pre-processing and network learning. As long as negative inputs are removed from the state space, this input encoder method can be used without any other pre-processing of state space information. Although this method may work well for evolutionary training of SNNs, future investigations into training those pre-processing weights in the input encoder using a non-evolutionary framework is needed for comparison.

It is clear that the reset variants of both F2F and Rate decoding methods achieves better fitness over less generations when compared to their non-reset variants, which is evident by both the Cart Pole and Lunar Lander networks. This is most likely a result of not needing information of previous states to select the optimal action in the current state. Not resetting the network to its default between steps introduces noise into the next state interpretation. This may not be the case for all problems where having some residual knowledge of previous states may be useful. It does however suggest that in these two problems, that previous state knowledge is not important. The failure of the Lunar Lander networks to reach the goal fitness is most likely a result of a lack of generations in training or incorrect network shape specifically evident by the poor fitness of the multi-layer networks. Additionally, the noisiness of the results of the Lunar Lander experiments indicate that

more individual trials for each set of parameters needs to be included.

VII. Conclusion

This research has identified some interesting trends that need to be investigated further. Firstly, the reset variants of the F2F and Rate based decoding methods seem to have a steeper initial learning than other methods especially evident in multi-layer networks. Secondly, a higher exposure period tends to produce goal reaching agents in less generations than lower exposure periods, specifically evident in the Cart Pole experiment. This therefore introduces a trade off between network accuracy and network efficiency when choosing an exposure period for a SNN. Thirdly, the input encoder method proposed allows for evolutionary training of networks and evolutionary learning of state preprocessing using the same algorithm, simplifying the need for complex preprocessing algorithms and making it a more general method for input encoding. Finally, although there is no clear benefit from using either temporal decoding or rate based decoding, temporal decoding methods like F2F seem to produce better performing agents at lower exposure periods when compared to Rate based decoding.

References

[1] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. homogeneous synaptic input," *Biological cybernetics*, vol. 95, no. 1, pp. 1–19, 2006.

[2] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in neuroscience*, vol. 12, 2018.

[3] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in Neuroscience*, vol. 10, p. 508, 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2016.00508

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[5] E. Yee and J. Teo, "Evolutionary spiking neural networks as racing car controllers," in *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, Dec 2011, pp. 411–416.

[6] U. Markowska-Kaczmar and M. Koldowski, "Spiking neural network vs multilayer perceptron: who is the winner in the racing car computer game," *Soft Computing*, vol. 19, no. 12, pp. 3465–3478, Dec 2015. [Online]. Available: https://doi.org/10.1007/s00500-014-1515-2

[7] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[8] S. Slade and L. Zhang, "Topological evolution of spiking neural networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–9.

[9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540

[10] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[11] L. Wiklendt, S. Chalup, and R. Middleton, "A small spiking neural network with lqr control applied to the acrobot," *Neural Computing and Applications*, vol. 18, no. 4, pp. 369–375, 2009.

[12] M. Li and J. Z. Tsien, "Neural code—neural self-information theory on how cell-assembly code rises from spike time and neuronal variability," *Frontiers in cellular neuroscience*, vol. 11, p. 236, 2017.

[13] R. VanRullen, R. Guyonneau, and S. J. Thorpe, "Spike times make sense," *Trends in neurosciences*, vol. 28, no. 1, pp. 1–4, 2005.

[14] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. Lawrence Erlbaum Associates, 1949.

[15] Z. Bing, C. Meschede, K. Huang, G. Chen, F. Rohrbein, M. Akl, and A. Knoll, "End to end learning of spiking neural network based on r-stdp for a lane keeping vehicle," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1–8.