2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

A Bio-inspired Reinforcement Learning Rule to Optimise Dynamical Neural Networks for Robot Control*

Tianqi Wei[1] and Barbara Webb[2]

Abstract— **Most approaches for optimisation of neural networks are based on variants of back-propagation. This requires the network to be time invariant and differentiable; neural networks with dynamics are thus generally outside the scope of these methods. Biological neural circuits are highly dynamic yet clearly able to support learning. We propose a reinforcement learning approach inspired by the mechanisms and dynamics of biological synapses. The network weights undergo spontaneous fluctuations, and a reward signal modulates the centre and amplitude of fluctuations to converge to a desired network behaviour. We test the new learning rule on a 2D bipedal walking simulation, using a control system that combines a recurrent neural network, a bio-inspired central pattern generator layer and proportional-integral control, and demonstrate the first successful solution to this benchmark task.**

I. INTRODUCTION

Neural networks have been applied to robot control long before the development of deep learning, e.g., the learning of inverse kinematics [1] [2], and bio-inspired central pattern generators (CPGs) [3]. Recent developments demonstrate many application in robotics [4] such as supervised learning for robot vision and reinforcement learning in robot motion control [5][6]. Due to these successes, and the ability of deep learning to fit "arbitrary" functions, we might expect the approach should also be applicable for motor learning. However, the neural control of motor systems is essentially dynamic, which can pose problems. For example, the learning rules we have for deep learning are yet not suitable for learning of neural networks with CPGs, which are undifferentiable. Yet CPGs are found widely in animals [7] and have been productively adopted for robot motion control. Although some reinforcement learning models include both CPGs and neural networks [8], the CPG is placed between the neural network and the physical model of the robot and is treated as part of the physical model, so that the neural network is differentiable.

Obviously, animals are able to learn with undifferentiable neural circuits. If such learning rules could be introduced to robot learning, there will be more options for the reinforcement learning of robot motion control. In particular, a bridge can be built between learning methods and previous research into neural network dynamics, such as CPGs, for robot control. The learning rule proposed in this paper is aimed at learning of continuous motion in an single robot using neural network architectures that are beyond the scope of backpropagation methods.

In most neural network models, both the activation functions (mimicking the processing in the soma of a neuron) and weights (mimicking input synapses) are static. Although some of the parameters are updated during training, they are time invariant during calculations. However, biological neural networks are always dynamic [9], even without learning. Neural spiking codes information in a complex hybrid of discrete and analog electrical dynamics. When the information passes between neurons through synapses, neurotransmitter is released from the pre-synaptic regions and received by receptors in the post-synaptic regions. Both regions have inherent dynamics that alter the effective synaptic strengths [10].

The model presented here is inspired by the dynamics in post-synaptic regions, which includes the motion of neurotransmitter receptors. A receptor can be transported between different regions by thermodynamics or active transportation by the neuron [11]. As receptors contribute differently to synaptic strengths (or weights) according to their locations, the transportation causes spontaneous fluctuations of the weights [12]. Because the dynamics of transportation includes random and chaotic processes, the weights are not accurately predictable and not phase-locked. Hence, weights in a real neural circuit always explore adjacent values [13]. Here we propose that if the exploration is controllable, the neural network weights could be optimised. Specifically, exploration should become wider with punishment but narrower with reward; and the centre of the exploration should move towards values that coincide with rewards. Fig 1 shows the concept with an example.

In the following sections we first introduce the proposed learning mechanism, and then describe its application to a complex, hybrid neural circuit to control a bipedal walking simulation.

II. MODELS AND METHODS

A. Learning rule

The neural network learning rule consists of 2 parts: a mechanism to produce spontaneous synapse dynamics that result in exploration of the weight space; and an updating rule to control these dynamics according to the rewards obtained. In other work [14] we have considered a biologically plausible model of the synaptic receptor transport mechanisms

*This work was supported by FP7 FET-Open project 'Minimal'.

[1]Tianqi Wei is a PhD candidate in the Insect Robotics Group, Institute of Perception, Action and Behaviour, School of Informatics, University of Edinburgh, Edinburgh, United Kingdom and the Stokes Research Group, the Institute for Integrated Micro and Nano Systems, School of Engineering, University of Edinburgh, Edinburgh, United Kingdom Tianqi-Wei@outlook.com

[2]Barbara Webb is Professor of Biorobotics and Director of the Institute of Perception, Action and Behaviour, School of Informatics, University of Edinburgh, Edinburgh, United Kingdom B.Webb@ed.ac.uk

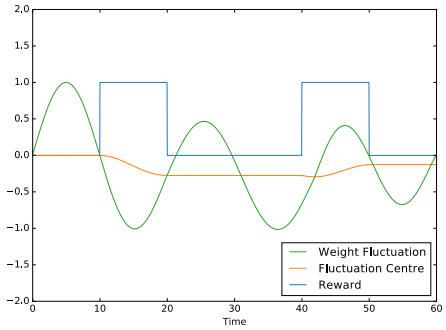

Fig. 1. Principle of the learning rule. A synaptic weight (green) fluctuates around a centre (red). When reward is received (blue) the centre is shifted in the direction of the ongoing fluctuation. The fluctuation amplitude decreases with positive reward for convergence of learning.

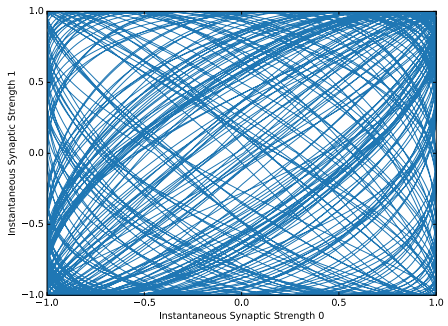

Fig. 2. Exploration Trajectory of 2 synapses

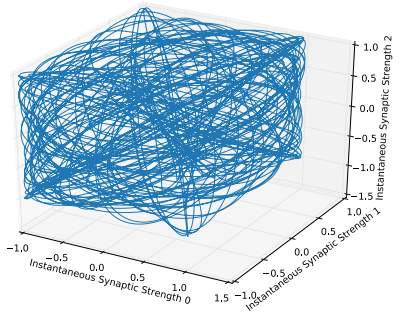

Fig. 3. Exploration Trajectory of 3 synapses

that could produce suitable dynamics, but here we present a simple version that is sufficient to provide the following properties:

- Spontaneous fluctuation should be free from the direct effect of information the synapse conveys, so that the input information does not limit the exploration in the synapse;
- Phases of the fluctuations in different synapses should not be locked, to avoid periodic exploration and instead help to sample the weight space densely;
- The fluctuation should be locally symmetric to be resistant to random biases, so that only rewards that correlate to the weight explorations contribute to the learning; and
- The periods of the fluctuations should be much longer than the periods of the learning objectives, such that when the new weights cause an effect and the reward arrives later, the weights should still be near the region that produced a reward.

1) Spontaneous Dynamics: Weight fluctuation is based on a sinusoid function with variable periods. For a single synapse:

$$W(t) = A\sin(2\pi\frac{t-\sum_0^i T_i}{T_i}) + C \quad \text{if } T_i < t < T_{i+1} \quad (1)$$

where t is time, A the amplitude of fluctuation, T_i the ith period, and C is the centre of fluctuation.

When a fluctuation crosses its centre from a specific direction, a new period is calculated by a Gaussian process:

$$T_i \sim \mathcal{N}(\mu, \sigma^2) \quad (2)$$

where μ is the centre of the distribution periods and the σ^2 the variance.

The fluctuations of weights in different synapses are independent. Hence, the weight space can be well explored. Fig 2 and Fig 3 show the exploration driven by the fluctuation in 2D and 3D weight spaces respectively.

2) Control of the dynamics: In this approach, learning consists in controlling the spontaneous dynamics according to the rewards generated. Given the use of a sinusoid function as the basis of the dynamics, we want to modulate two variables as a function of the reward (which could be positive or negative): the centre of fluctuation C which changes the average weight of the synapse; and the fluctuation amplitude A which balances exploration and convergence.

The fluctuation centre C is updated according to the reward and the current value of the fluctuating weight:

$$\dot{C} = \alpha(W(t) - C)R(t) \quad (3)$$

where α is learning rate, (here $1.2 \times 10^{-5}\text{ms}^{-1}$), R the reward. When the reward is positive, the fluctuation centre shifts in the same direction as the on-going fluctuation. If the reward is negative, the fluctuation centre shifts in the opposite direction.

The fluctuation amplitude A, which is positive, is updated according to the reward only:

$$\dot{A} = -\beta R(t) \quad (4)$$

where β is the convergence rate (here $1 \times 10^{-9}\text{ms}^{-1}$). When the reward is positive, the fluctuation converges; when the reward is negative, the fluctuation expands to explore a wider space.

B. Experiment

1) Simulation Environment: The experiment is based on the continuous robot motion control tasks BipedalWalker-v2 and BipedalWalkerHardcore-v2 available within the reinforcement learning environment OpenAI Gym [15]. The first task is a side-scrolling video game style environment with a

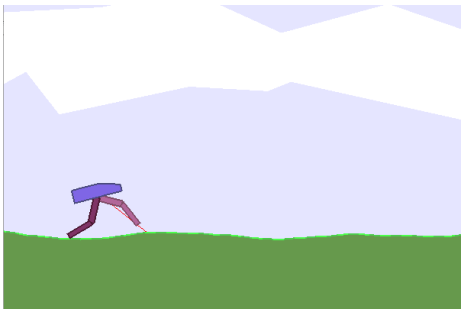

Fig. 4. BipedalWalker-v2 screenshot

2D bipedal robot moving on terrain with small slopes (Fig 4), the second task is the same type of robot but on terrain with stairs, boxes and trenches. OpenAI Gym provides the control API to the robot, which is torque that is applied to 4 joints of the robot (the values feed to the API are called Actions). The API also provides step by step states of the robot and a 10-point Lidar input, and reward values according to the robot's motions. In the experiments, we adopted the original reward provided by the API, which favours the robot's moving forward, keeping its head straight and decreasing the motors' torque. The reward is -100 when the robot falls down. The criterion of solving the task is getting average rewards of 300 over 100 consecutive trials.

For the details of the robot and tasks, please see `https://github.com/openai/gym/wiki/BipedalWalker-v2`, `https://gym.openai.com/envs/BipedalWalker-v2/` and `https://gym.openai.com/envs/BipedalWalkerHardcore-v2/`.

2) Control system: The architecture of the control system consists of a recurrent neural network (RNN) with multiple layers including FitzHugh-Nagumo Oscillators (Fig 5). There are two types of connections between layers as shown in Fig 5: buses of one-to-one connections with fixed weights; or full all-to-all connectivity where the learning rule is applied. Each neuron has one weighted sum input and one output, but different activation functions are used in different layers. Exceptionally, a Fitzhugh-Nagumo Oscillator has two states and we use both of them as outputs.

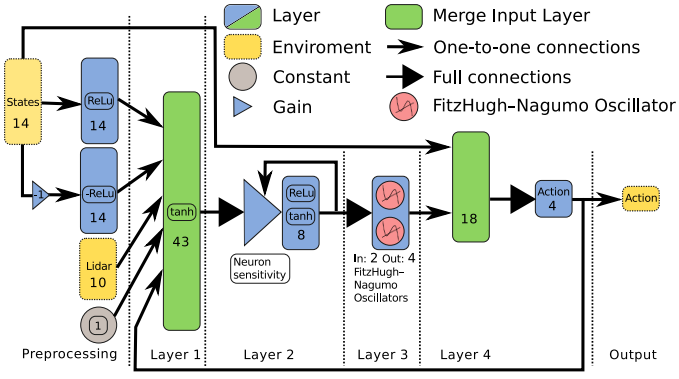

Fig. 5. The architecture of the control network (see text)

There are 5 major parts in the architecture: preprocessing and Layers 1 to 4.

In the preprocessing, each of the robot's 14 states provides two inputs: $[value, 0]$ for positive values and $[0, -value]$ for negative values. These are combined with ten Lidar values, a constant and a feedback signal of the 4 output actions. They are individually normalized according to their possible range and fed into Layer 1. Layer 1 thus has 43 neurons and uses Hyperbolic Tangent as the activation function.

The outputs of Layer 1 are fed into Layer 2 with full connectivity. The connections have uniformly distributed $U(-0.1, 0.1)$ random initial weights and are changed by the learning rule. Layer 2 has 8 neurons and serially uses Hyperbolic Tangent and Rectified Linear Unit as the activation function. To let the neurons in this layer remain sensitive to input changes, it has a slow adaptive gain to control the neuron sensitivity:

$$\dot{g_j} = (0.3 - |o_j|)\gamma \tag{5}$$

$$p_j(t) = g_j \sum o_i(t) W_{ij} \tag{6}$$

where i is the index of neurons in the Layer 1, j the index of neurons in the Layer 2, g the neuron sensitivity, γ the update rate, which is $1 \times 10^{-6} \mathrm{ms}^{-1}$, p the input to Layer 2, o the output of Layer 1, W_{ij} the weights from neurons i in Layer 1 to neuron j in Layer 2 (please note we use W instead of w for weights to avoid confusing with the state w in FitzHugh-Nagumo oscillators).

The outputs of Layer 2 are fed into Layer 3 with full connectivity. The connections have uniformly distributed $U(-0.1, 0.1)$ random initial weights and are changed by the learning rule. The 2 neurons in this Layer are FitzHugh-Nagumo oscillators [16], a simplified model of spiking neuron dynamics, widely applied in CPGs for robot control [3]. However, unlike a typical CPG, the 2 neurons are not coupled but get input from Layer 2 individually, hence their phases are adjusted individually by the upstream layers. In our model, the dynamics are scaled to an appropriate period by τ to fit the period of the walk.

$$\tau \dot{v} = v - v^3 - w + I \tag{7}$$

$$\tau \dot{w} = a(bv - cw) \tag{8}$$

where τ is the time constant for scaling, which is 0.02; w and v are 2 states; I is the input; a, b, and c are constants, among them $a = 0.08$, $b = 0.2$ and $c = 0.8$. v and w are used as outputs to the next layer. Fig 6 shows the dynamics of the oscillator with increasing input from -2 to 2, and Fig 7 shows the phase portrait of the oscillator. With low or high input, the oscillator is stable; with input from about -0.6 to 1.1, the oscillator is unstable.

The outputs of Layer 3 are combined with the 14 states of the robot and fed into Layer 4 with full connectivity and application of the learning rule. Layer 4 has 4 neurons without activation functions (i.e. output is just weighted sum of the input). Their outputs are used as torque of the 4 joints of the robot respectively. The initial weights for the

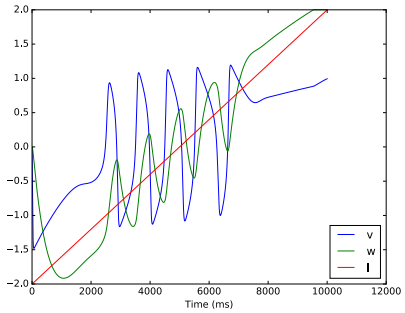

Fig. 6. A FitzHugh-Nagumo oscillator with increasing input

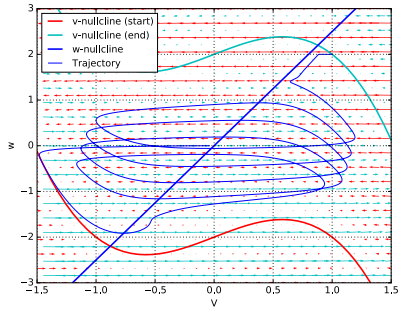

Fig. 7. Phase Portait of the FitzHugh-Nagumo oscillator

layer use the state inputs v and $-w$ to control knee and hip respectively, and take the joint speed and joint angle as the feedback for PI control. The weight values do not need to be carefully selected but depend on the learning rule for tuning. The 4 outputs of Layer 4 also feedback as inputs to Layer 1.

III. Results

The control system is trained with the learning rule on a computer with Intel®Core™i5-5200U CPU. It took about 11 hours wall clock time, corresponding to 47956 Episodes to solve the BipedalWalker-v2 (see the associated video `https://youtu.be/B7mLVY1NKgI`). We note that according to the official Leaderboard of the task (`https://github.com/openai/gym/wiki/Leaderboard`) on Feb. 27, 2018, no previous solution has been obtained. The source code of the model and experiment are available online: `https://github.com/InsectRobotics/DynamicSynapseSimplifiedPublic.git`.

As shown in Fig 8, the average episode reward had a quick increase from -100 to 200 during episodes 2000 to 14000, then gradually reached up to more than 300 after Episode 47956. As getting average rewards of 300 over 100 consecutive trials is the threshold of solving the task, our approach solved the task. Fig 9 shows a Poincare Map of the exploration, which gradually converges to smaller space and became denser. With negative reward more than positive reward, the fluctuation amplitude increases for the first 30 hours of simulated time, which leads to broader exploration

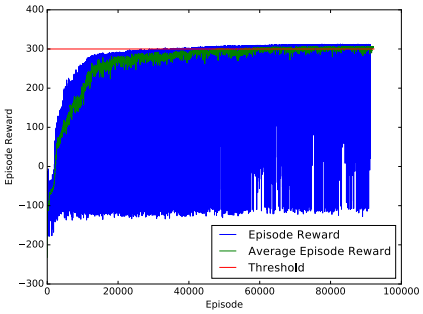

Fig. 8. Episode reward and its average in every 100 episodes. Success on the task is defined as an average above 300.

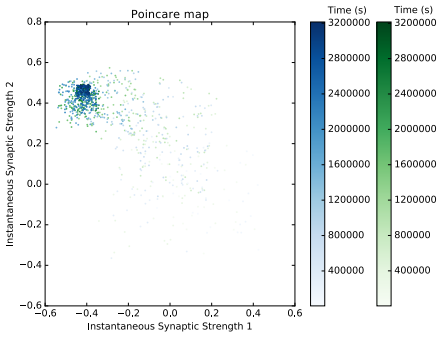

Fig. 9. A Poincare Map of the exploration. The points are intersections of the exploration trajectory and a section of the weight space. Two colours mark the side the trajectory came from, and the gradients of the colours indicate the time of intersection.

of weights. Then it gradually decreases approaching zero, which leads to convergence of the learning. As shown in Fig 10, the weight fluctuations generally explore wider around their centres at the beginning than later. The fluctuation centres moved to optimized values after the training.

The control system and learning rule were also tested on the BipedalWalkerHardcore-v2. The robot learnt to walk up and down stairs, stride over small boxes but was not successful in striding over large boxes and trenches. Striding over large boxes requires the robot to jump, so the control system might not able to support the two distinguished gaits. Striding over trenches requires the robot to control the feet to

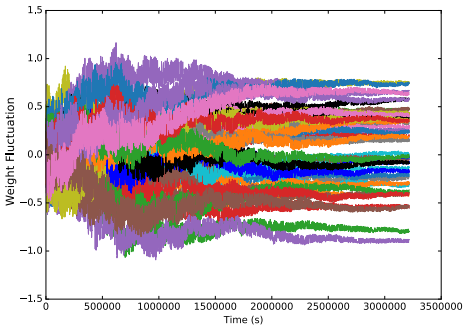

Fig. 10. Fluctuation of the weights: the amplitude and centre of fluctuation is altered by reward and converges on a solution.

correct footholds accurately, so the control system is either unable to accurately control the feet to a specific point or unable to find the correct footholds. Overcoming these limitations may require a change to the architecture of the control system.

We additionally tested whether successful learning can occur if the learning rule is only 'attractive', that is, the fluctuation centre is shifted towards the current value of the fluctuation when positive reward is received, but there is no 'repulsion', i.e., shifting away from the current value with negative reward. We also examined the contribution of the CPG input weight adaptation, using either no adaptation, linear adaption or non-linear adaption. The combined results of these two manipulations are summarised in (Figure 11).

Note that (as for other reinforcement learning algorithms), random seeds of the simulator can affect the process of learning (Figure 11 (A) without control of random seeds; (B) and (C) with control of random seeds, which show the difference between different configurations more clearly). We also notice random seeds also affect various learning configurations differently (see figure 11 (B) and (C)). Hence, we did multiple groups of experiments and show some typical results here.

Among all the experiments, those with both repulsive learning and nonlinear CPG adaptation have the highest possibility to solve the tasks, providing the only solution in Figure 11 (B, green line), and the first solution in Figure 11 (C, green line). The experiments lacking either repulsive learning or adaptation have the smallest possibility to solve the tasks. In Figure 11 (B), the configuration lacking both never reached 200 (red line). In Figure 11 (C), the configuration without repulsive learning and with linear adaptation (purple line) failed to solve the task.

Normally, with same neural network configuration and random seeds, the learning rule without repulsive learning works less well than the learning rule with it (see Figure 11 (B) and (C), respectively). A possible reason is that repulsive learning provides disturbances to push the exploration centre away from the region of weight space that leads to negative rewards. When the weight fluctuation reaches the region of weight space that leads to positive rewards, the attractive learning provides an attractor to higher reward region. Without repulsive learning, if the system under training initialised in a punishment region and cannot reach the reward region with fluctuation, the system is not able to learn.

IV. Discussion

Inspired by the dynamics of biological synapses, a new learning rule is developed. With this learning rule, we trained a hybrid control system for a bipedal 2D walker traversing terrain with small-slopes, stairs and small boxes. The architecture of the control system is different from typical reinforcement learning using neural networks that it includes not only feed-forward connections, but also feedback connections in different levels, a CPG, and a layer that work as a complex PI control. In summary, it is a mix of neural networks and classical robot control that cannot be optimised with backpropagation. Our learning mechanism instead uses spontaneous weight fluctuations which are modulated by reward to converge to the region of state space that maximises reward. We show that this can produce successful robot control.

TABLE I
HIGHEST AVERAGE SCORES REPORTED

Algorithm	Highest average score	Window size	Source
NEAT	54	50	[17]
NES	-23	50	[17]
CMA	-75	50	[17]
P3O	161	50	[17]
CA3C	129	50	[17]
D3PG	90	50	[17]
PPO	251	100	[18]
PPO(8 actors)	221	100	[18]
PPO-ER	270	100	[18]
PPO-ER(8 actors)	285	100	[18]
TRPO	238	100	[18]

Our approach exceeds the performance of previous attempts at solving the bipedalWalker-v2 (see Table I). Methods that have been explored include deep reinforcement learning methods such as Continuous Asynchronous Advantage Actor-Critic, Parallelized Proximal Policy Optimization, and Distributed Deep Deterministic Policy Gradient; and evolutionary methods such as Covariance Matrix Adaptation Evolution Strategy, NeuroEvolution of Augmenting Topologies, and Natural Evolution Strategy[17]. Our approach also exceeds the performance of the proximal policy optimization algorithm with multi-batch experience replay scheme [18]. We believe there are three main reasons for the improvement in performance:

- Typical neural networks and existing robot control approaches cannot be optimised using the same rule continuously, but this is possible with our learning rule, thus our architecture can take advantage of existing robot control approaches (such as CPG and PI control) in a hybrid network.
- The sampling happens in the parameter space instead of the action space, which decreases the complexity of the optimized generated random numbers, which become a fixed-length vectors instead of unfixed-length sequences.
- For the same reason as above, the output actions are continuous and smooth during exploration, while previous approaches typically use random number generators to explore the action space, which introduces noise and impacts the quality of actions.

The main existing alternative approach to this kind of problem is based on genetic algorithms. However, they require transient changes of variables, which is less suitable than our learning rule when applied to cases in which a robot is hard to reset, such as an actual physical robot.

The gait produced by our learning method for the BipedalWalker is successful for locomotion, but does not resemble a typical bipedal walking gait, i.e., the legs do not swing past

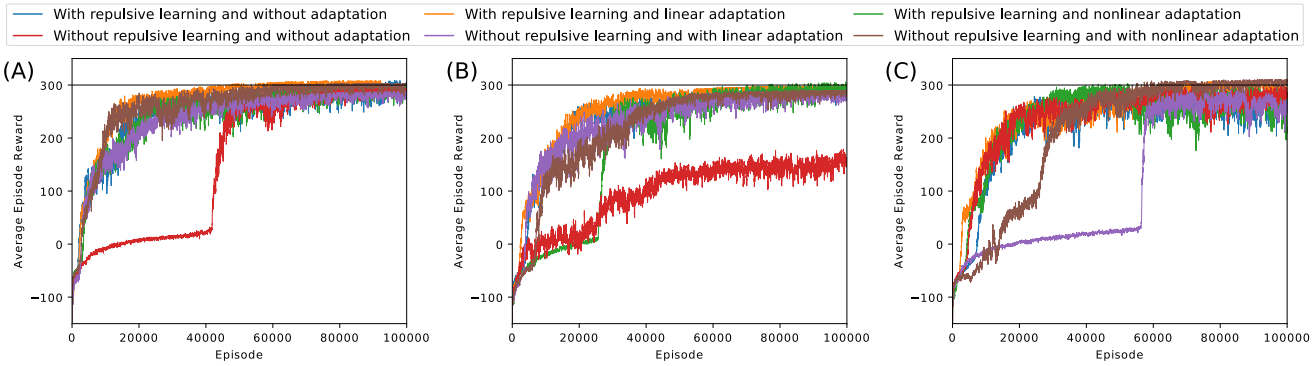

Fig. 11. Comparison between trainings with/without repulsive learning or CPG input weight adaptation. (A) Without control of random seeds. (B) and (C) With control of random seeds, each group of experiments have the same random seed.

each other in alternation. We believe this is due to the fact that the robot is planar, and as such does not need to maintain lateral balance. Keeping one leg always in front and the other at the rear is the most stable configuration for forward-backward balance, particularly in early stages, and the robot then optimises its gait for this configuration. The same gait is often found as a solution in this task. Alternatively, the observed movement can be compared to the bounding gait naturally exhibited by some quadrapeds, where the two front and two hind legs are moving in unison.

Our learning rule can be applied to more complex and dynamical neural networks than backpropagation, because it can be applied to a wider range of architectures and dynamics. This could be particularly valuable in scaling to more complex tasks, as the curse of dimensionality might be addressed by introducing richer internal structures, e.g., internal rewards that structure learning towards distant final rewards. Other applications system identification, dynamic model based robot control and so on. Moreover, it can be used in hybrid systems that combine classical robot control approaches and neural networks, thus facilitating the introduction of neural networks into robot control tasks.

REFERENCES

[1] Guo and Cherkassky, "A solution to the inverse kinematic problem in robotics using neural network processing," in *International Joint Conference on Neural Networks*. IEEE, 1989, pp. 299–304 vol.2. [Online]. Available: http://ieeexplore.ieee.org/document/118714/

[2] S. Tejomurtula and S. Kak, "Inverse kinematics in robotics using neural networks," *Information Sciences*, vol. 116, no. 2-4, pp. 147–164, jan 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025598100981

[3] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[4] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: a review of recent research," *Advanced Robotics*, vol. 31, no. 16, pp. 821–835, 2017. [Online]. Available: http://doi.org/10.1080/01691864.2017.1365009

[5] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates," oct 2016. [Online]. Available: http://arxiv.org/abs/1610.00633

[6] D. Ribeiro, A. Mateus, P. Miraldo, and J. C. Nascimento, "A real-time Deep Learning pedestrian detector for robot navigation," in *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, apr 2017, pp. 165–171. [Online]. Available: http://ieeexplore.ieee.org/document/7964070/

[7] I. A. Rybak, K. J. Dougherty, and N. A. Shevtsova, "Organization of the Mammalian Locomotor CPG: Review of Computational Model and Circuit Architectures Based on Genetically Identified Spinal Interneurons," *eNeuro*, vol. 2, no. 5, 2015. [Online]. Available: http://eneuro.sfn.org/cgi/doi/10.1523/ENEURO.0069-15.2015

[8] T. Mori, Y. Nakamura, M.-a. Sato, and S. Ishii, "Reinforcement Learning for a CPG-driven Biped Robot," *Aaai 2004*, pp. 623–630, 2004.

[9] E. M. Izhikevich, *Dynamical Systems in Neuroscience*. The MIT Press, 2007, vol. 25, no. 1. [Online]. Available: http://www.izhikevich.org/publications/dsn.pdf

[10] D. Choquet and A. Triller, "The dynamic synapse," *Neuron*, vol. 80, no. 3, pp. 691–703, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.neuron.2013.10.013

[11] K. Czondor, M. Mondin, M. Garcia, M. Heine, R. Frischknecht, D. Choquet, J.-B. Sibarita, and O. R. Thoumine, "Unified quantitative model of AMPA receptor trafficking at synapses," *Proceedings of the National Academy of Sciences*, vol. 109, no. 9, pp. 3522–3527, 2012. [Online]. Available: http://www.pnas.org/cgi/doi/10.1073/pnas.1109818109

[12] L. a. Cingolani and Y. Goda, "Actin in action: the interplay between the actin cytoskeleton and synaptic efficacy," *Nature Reviews Neuroscience*, vol. 9, no. 5, pp. 344–356, 2008. [Online]. Available: http://www.nature.com/doifinder/10.1038/nrn2373

[13] A. Minerbi, R. Kahana, L. Goldfeld, M. Kaufman, S. Marom, and E. Noam, "Long-Term Relationships between Synaptic Tenacity , Synaptic Remodeling , and Network Activity," *PLOS Biology*, vol. 7, no. 6, 2009.

[14] T. Wei and B. Webb, "A model of operant learning based on chaotically varying synaptic strength," in preparation.

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[16] R. FitzHugh, "Impulses and Physiological States in Theoretical Models of Nerve Membrane," *Biophysical Journal*, vol. 1, no. 6, pp. 445–466, 1961. [Online]. Available: http://dx.doi.org/10.1016/S0006-3495(61)86902-6

[17] S. Zhang and O. R. Zaiane, "Comparing Deep Reinforcement Learning and Evolutionary Methods in Continuous Control," no. Williams 1992, 2017. [Online]. Available: http://arxiv.org/abs/1712.00006

[18] S. Han and Y. Sung, "Multi-Batch Experience Replay for Fast Convergence of Continuous Action Control," vol. 34141, pp. 1–10, 2017. [Online]. Available: http://arxiv.org/abs/1710.04423