Physical Reinforcement Learning

Sam Dillavou[1], **Shruti Mishra**[2]
dillavou@upenn.edu, sm3056@cam.ac.uk

[1]**Department of Physics and Astronomy, University of Pennsylvania, USA**
[2]**Department of Applied Mathematics and Theoretical Physics, University of Cambridge, UK**

Abstract

Digital computers are power-hungry and largely intolerant of damaged components, making them potentially difficult tools for energy-limited autonomous agents in uncertain environments. Recently developed *Contrastive Local Learning Networks* (CLLNs), analog networks of self-adjusting nonlinear resistors, are inherently low-power and robust to physical damage, but have been exclusively used to perform supervised learning tasks (Dillavou et al., 2024). In this work, we demonstrate success on two simple Markov decision processes using tabular Q-learning adapted for simulated CLLNs. Doing so makes explicit the components (beyond the network being trained) required to enact various tools in the RL toolbox, some of which (policy function and value function) are more natural in this system than others (replay buffer). Limitations to external memory make the big world hypothesis, and learning via tracking and streaming, an attractive framework for reinforcement learning on CLLN-based agents (Javed & Sutton, 2024; Elsayed et al., 2024). We discuss assumptions such as the physical safety that digital hardware requires, CLLNs can forgo, and biological systems cannot rely on, and highlight secondary goals that are important in biology and trainable in CLLNs, but make little sense in digital computers. We share our code for Q-learning adapted for simulated CLLNs, for further consideration as a physical framework for learning in the big world [github].

1 Introduction

Digital computers are powerful and versatile machines, but have significant weaknesses, especially in areas relevant to reinforcement learning (RL). First, they are fault intolerant and thus susceptible to damage. Disable a handful or even a single transistor in a CPU or GPU and it may crash the entire system[1]. Having a single transistor erroneously flip even on 1% of write operations could catastrophically disable the entire machine. This is because the function of each component is intrinsically tied to its location within the system, and the entire operation relies on (nearly) error-free operation. Second, they are power-hungry, with laptop CPUs consuming approximately 50 W. This stems from the high energy cost of maintaining 'perfect' operation, as well as the shuttling of data between processing and memory.

Low-power operability and fault tolerance are areas where biological systems thrive, as they are autonomous agents in an often energy-scarce and dangerous world. Brains can take significant damage and continue to function, including destruction of single neurons (Wang et al., 2014), traumatic brain injuries (Chua et al., 2007), and even removal of large brain regions (Granovetter et al., 2022). This robustness stems from brains' distributed processing and emergent, rather than linear, computation. The human brain is also energy efficient, using 20 W in total (Balasubramanian, 2021) while

[1]https://www.ntchip.com/electronics-news/transistors-in-cpu

performing perception, cognition, motor control, homeostatic functions, learning, and more. This energy efficiency is in part due to the overlap of memory and computation, and to *natural primitives* (Mead, 1990), whereby low-level computations utilize low-power analog physical or chemical processes rather than digital logic.

In spite of the potential advantages, there are few instances of artificial and non-digital hardware performing RL tasks (Mak et al., 2007; 2010). Many instances of digitally-enhanced or simulated analog systems have been used in an RL setting (as in Mikaitis et al. (2018)), but few if any hardware demonstrations combine the distributed memory and computation with analog signals that give brains some significant survivability advantages.

Recently developed Contrastive Local Learning Networks (CLLNs) (Dillavou et al., 2024) may fill this gap. A network of self-adjusting nonlinear resistors, these systems perform supervised learning in an emergent manner by utilizing physical processes to perform computation. Learning occurs in each individual element using local rules, eschewing central and digital processing. As a result, they are energy efficient (Stern et al., 2024) and fault tolerant – operating even when some or even many elements are physically removed (Dillavou et al., 2022). However, CLLNs have not been demonstrated in an RL context, even in simulation.

Designing an RL implementation of CLLNs provides a setting to ask a range of questions. For instance, which RL tools are natural for a self-learning network, and which require additional pre-programmed hardware? There are numerous challenges associated with digital RL acting in the real (non-simulated) world (Dulac-Arnold et al., 2021); what additional challenges stem from placing the agent's brain outside of the pristine digital realm (Fig. 1)? A successful navigation of these challenges would open a path towards energy-efficient and fault-tolerant autonomous agents.

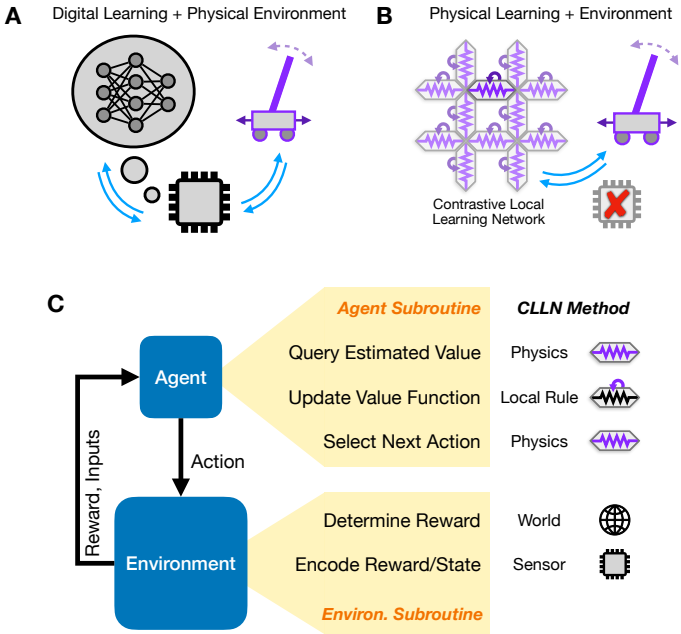

Figure 1: (A) Reinforcement learning envisioned for a physical task – a digital agent interacts with a physical environment, mediated by a computer. (B) Proposed: A Contrastive Local Learning Network (CLLN) interacts with a physical environment, and learns based on those interactions. There is no digital processing; learning is done in a distributed, analog fashion. One of the many identical self-adjusting components is highlighted. Physical and simulated components are shown in purple and grey, respectively. (C) A breakdown of the training algorithm for the CLLN performing physical Q learning. Actions taken by a digital network in deep Q learning are here performed by analog physics and local rules.

In the following text, we first present the dynamics of CLLNs. Following this, we highlight aspects of CLLNs that make the big world hypothesis an attractive framework for algorithm development in this setting (Javed & Sutton, 2024). We then illustrate the adaptation of Q-Learning to the dynamics of a CLLN in the context of two simple Markov decision processes (MDP), and discuss the broader implications of using CLLNs as a substrate for reinforcement learning methods.

2 Reinforcement learning with a simulated CLLN

2.1 Background: Contrastive Local Learning Networks

Contrastive Local Learning Networks (CLLNs) are networks of self-adjusting resistive elements whose individual dynamics approximate gradient descent on a global loss function (Dillavou et al., 2024). The standard operation of these systems is for supervised learning and is briefly summarized here.

We consider a network of resistors as shown schematically in Fig. 2. We will treat four of the nodes (yellow) as inputs, and encode data by enforcing the voltage values of these nodes ($U_1^{\text{in}}, \ldots, U_4^{\text{in}}$). We will also choose four nodes (blue), and treat their equilibrium voltage values, $U_1^{\text{out}}, \ldots, U_4^{\text{out}}$, as outputs. That is, we treat our network as a physical function, $(U_1^{\text{out}}, U_2^{\text{out}}, U_3^{\text{out}}, U_4^{\text{out}}) \equiv \mathcal{F}(U_1^{\text{in}}, U_2^{\text{in}}, U_3^{\text{in}}, U_4^{\text{in}})$. It is a consequence of our eventual choice of task that the number of inputs and outputs is equal; they are independently specified in general. When input voltages are applied, output values are 'calculated' by the physical processes, in this case the flow of current in the network. The result of this calculation is a consequence of the network structure and the conductance (inverse of resistance) values of each edge of the network, G_i. We note that at equilibrium, the voltages at unconstrained nodes have minimized power dissipation with respect to the boundary conditions[2].

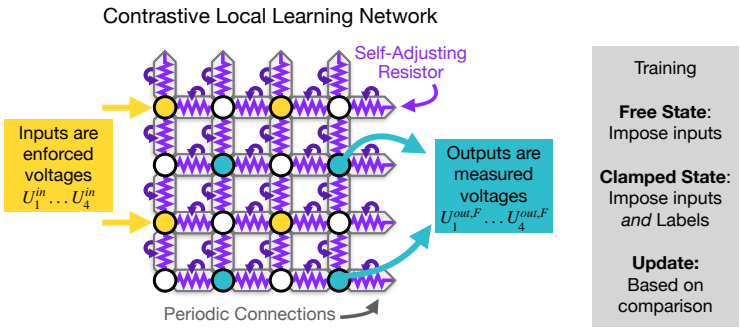

Figure 2: Schematic of a Contrastive Local Learning Network (CLLN). The configuration shown is used for the Markov decision process with four states and four actions. A modified network architecture and input-output position is used for the 9-state navigation task (see Fig. 4). A high-level description of the contrastive training protocol is outlined in the grey box. Note that each element in a CLLN requires only local measurements of the two states (free and clamped) to update itself, decentralizing the training process.

The evolution of the gate voltages (the trainable parameters) in this network is *contrastive* in that it requires comparing two distinct states that we can impose. Given a single data point (inputs $U_1^{\text{in}}, \ldots, U_4^{\text{in}}$) and label (desired outputs $L_1, \ldots, L_4$), the two states are as follows. First, the *free* state, wherein only inputs $U_1^{\text{in}}, \ldots, U_4^{\text{in}}$ are imposed. In this state, each resistor experiences voltage drop ΔU_i^F and the outputs find values $U_n^{\text{out},F}$. We use subscript i to denote quantities on network edges, and n to denote quantities on nodes (inputs or outputs). Next, the *clamped* state, in which

[2]In a nonlinear network like the one we consider, the true minimized quantity is *co-content*, which reduces to power in the linear case. As this is a less-commonly known quantity we use power throughout. This does not change the intuition or derivations.

inputs as well as the desired outputs (labels) are enforced. Now each resistor experiences voltage drop ΔU_i^C, and the (enforced) outputs are equal to the labels $U_1^{\text{out},C} = L_1$, etc. In practice the clamped output is actually nudged *towards* the label: $U_n^{\text{out},C} = U_n^{\text{out},F}(1-\eta) + \eta L_n$, where η is a hyperparameter, and $\eta = 1$ simply clamps the label ($U_n^{\text{out},C} = L_n$). In this work we use $\eta = 0.1$.

The system will perform gradient descent on a *contrastive* function, with respect to its conductance values, following the Coupled Learning framework (Stern et al., 2021). This framework is closely related to Equilibrium Propagation (Scellier & Bengio, 2017) and Contrastive Hebbian Learning (Movellan, 1991). Specifically,

$$\delta G_i = -\alpha \frac{d}{dG_i}\Big[\underbrace{\mathcal{P}^C - \mathcal{P}^F}_{\text{contrastive fn}}\Big], \tag{1}$$

where $\mathcal{P}^C$ and $\mathcal{P}^F$ are the total dissipated power in the clamped and free states respectively, and α is a learning rate. We note that because the electronic network minimizes power with respect to its boundary conditions, and the clamped state is equivalent to the free state with added boundary conditions (the clamped outputs), the contrastive function must be non-negative. Further, it is only zero-valued when the clamping procedure does not change the outputs, that is, the labels are already satisfied. Thus, it has the same minima as a mean-squared error cost function, and in practice its minimization works well to minimize error (Stern et al., 2021; Dillavou et al., 2024; 2022).

Calculating the derivative on the RHS of eq. 1 for the clamped power with the chain rule, we find that

$$\frac{d\mathcal{P}^C}{dG_i} = \sum_j^{\text{edges}} \frac{\partial \mathcal{P}^C}{\partial G_j}\cancelto{\delta_{ij}}{\frac{dG_j}{dG_i}} + \cancelto{0}{\frac{\partial \mathcal{P}^C}{\partial \Delta U_j^C}}\frac{d\Delta U_j^C}{dG_i} = \frac{\partial \mathcal{P}^C}{\partial G_i}. \tag{2}$$

The first term cancels for $i \neq j$ as the conductances are independently set, and the second cancels as power is (as noted) at a minimum with respect to the voltages. As a result, the total derivative becomes a partial derivative, which we can further simplify by noting that power is a sum over local powers:

$$\mathcal{P}^C = \sum_j^{\text{edges}} \left(\Delta U_j^C\right)^2 G_j \quad \rightarrow \quad \frac{d\mathcal{P}^C}{dG_i} = \frac{\partial \mathcal{P}^C}{\partial G_i} = \left(\Delta U_i^C\right)^2 \tag{3}$$

Thus we complete the same calculation for the free power, and the result is a local learning rule,

$$\delta G_i = -\alpha \frac{d}{dG_i}\Big[\underbrace{\mathcal{P}^C - \mathcal{P}^F}_{\text{contrastive fn}}\Big] = \alpha\Big[\left(\Delta U_i^F\right)^2 - \left(\Delta U_i^C\right)^2\Big], \tag{4}$$

wherein each element needs only to measure its own voltage drop in both states to enact this gradient descent. In practice, this is done by creating twin, co-evolving networks, one holding the free state, the other the clamped state (Dillavou et al., 2022; 2024).

We model our simulations after Dillavou et al. (2024), where each conductive element is in fact a MOSFET transistor in the triode (passive) regime, with conductance equation

$$G_i = S(U_{G,i} - U_T - \overline{U}_i) \tag{5}$$

where in our simulations $S = 1$ and $U_T = 0.7$ are constants, $U_{G,i}$ is the (adjustable) gate voltage that will act as weights in our system, and $\overline{U}$ is the average of the node voltages on either side of the edge. This final term allows the system to perform nonlinear input to output transformations. We note that because $\partial G_i / \partial U_{G,i} = S$, the gradient with respect to G_i and $U_{G,i}$ is co-linear, and we may enact eq 4 by changing $U_{G,i}$, which will be our learning procedure. We restrict the range of our parameters such that $1.0 < U_{G,i} < 5.5$ to mimic experimental conditions.

2.2 Connections to the big world hypothesis

Several properties of CLLNs make the big world hypothesis an attractive framework for learning directly with this physical substrate. Given that parameters and outputs in the CLLN are bounded,

the learning response to a signal will naturally be limited in the larger external environment. Perhaps most pertinently, common deep RL approaches incorporate the storage of history, through eligibility traces and replay buffers (Haarnoja et al., 2018; Schulman et al., 2017; Mnih et al., 2013). While function approximation is naturally part of a CLLN, it is not obvious how to naturally introduce history storage into a CLLN, and methods like tracking and streaming deep RL (Elsayed et al., 2024) are attractive as they better support the vision of 'a network in the wild' (Fig. 1). We share our code, for consideration as a simulated physical substrate for reinforcement learning in big world tasks [github].

2.3 Example 1: A toy Markov decision process with four states and four actions

In this section, we adapt the update rule for the CLLN to Q-Learning on a Markov decision process (MDP) with four states and four actions. We will treat our CLLN simulation as enacting the Q matrix. At each training step t, we encode the environmental state S_t as the input; states $S_1, \ldots, S_4 \equiv [U_1^{\text{in}}, U_2^{\text{in}}, U_3^{\text{in}}, U_4^{\text{in}}]$ = [1 0 1 0], [0 1 0 1], [1 1 0 0], and [0 0 1 1] V respectively. Note that all voltages are analog, despite the input appearing to be digital, and the outputs will fall between 0 and 1. We then use ϵ-greedy action selection where ϵ decays from 0.05 to 0 linearly over the training run. We select the maximum output of the four (with probability $1-\epsilon$) as our action A_t.

The environment then issues a state-dependent reward $R(S_t, A_t) + \mathcal{N}(0, 0.01)$ and the state changes to S_{t+1}. $R(S_t, A_t)$ was sampled from $\mathcal{N}(0.1, 0.1)$, and the selected values are shown in Fig. 3A. Our environment is extremely simple, with action 0 inducing state 0, action 1 inducing state 1 and so on. We then impose the inputs for state S_{t+1}, and take the maximum output minus the mean of all four outputs as our predicted future reward. We use a discount factor $\gamma = 0.5$ and all of this information to create a future-weighted score L_t:

$$L_t = R(S_t, A_t) + \gamma \left[\max(\mathcal{F}(S_{t+1})) - \text{mean}(\mathcal{F}(S_{t+1}))\right] \tag{6}$$

We find that subtracting the mean term improves performance, as it removes a constraint on the average set of outputs, permitting our small network additional flexibility. We allow the system to evolve (eq 4) while imposing encoded S_t as inputs, U_i^{out} on all clamped-state outputs corresponding to actions the system *did not* take, and L_t on the clamped-state output whose action was taken. Our approach is summarized in Algorithm 1.

As an example: the system is in state $S_t = 1$, and U_2^{out} is the biggest (we assume we do not take the rare random action). The agent takes action $A_t = 2$ as a result, gets reward R_t, and the state moves to $S_{t+1} = A_t = 2$. L_t is calculated as above, and then we allow our system to evolve (eq 4) while S_t is imposed as input in both free and clamped states, and $\mathcal{F}(S_{t+1})$ is imposed as the clamped label with L_t inserted at position 2 (the action taken). That is, we train the system to keep the outputs of the untouched actions the same, while moving the chosen action's output value towards L_t. The

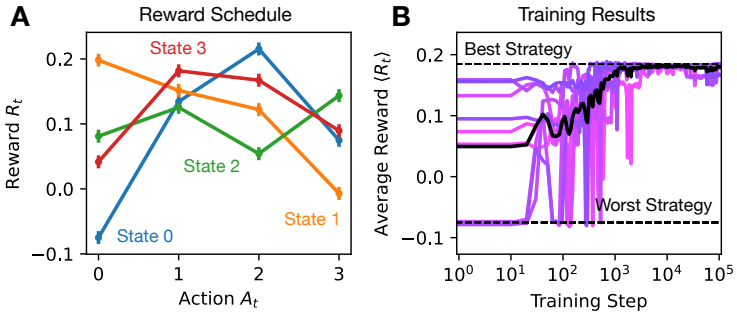

Figure 3: Q Learning with CLLN. (A) Reward schedule for each state (noise effect shown as small error bars). The optimal strategy involves a cycle through all four states. (B) Average reward over training for 10 trials (purples) overlaid with their average (black).

result is evolution similar to the Bellman equation. Updates are batched and imposed on the system every 50 steps.

2.3.1 Results

We perform 10 trials of 100,000 training steps, each time randomly initializing $U_{G,i}$ using $\mathcal{N}(1.5, 0.1)$. Rewards are tallied over time, and averaged in logarithmically spaced intervals (minimum of 10 steps) for each trial, shown as purple lines in Fig. 3B, with their average overlaid in black. We simulate rewards for our same protocol (with $\epsilon = 0$ but including reward noise) using 10,000 steps with each of the $4^4 = 256$ possible strategies, and denote the best and worst cases with black dashed lines in Fig. 3B. Our system always learns a near-optimal strategy, including ending at the best possible strategy in 8 of 10 trials.

2.4 Example 2: Navigation on a grid with nine states and four actions

We use a modified network architecture to attempt a slightly more difficult task, navigating towards a goal location in discretized 2D space. The reward schedule is shown in Fig. 4A, with a reward gradient (shaping) 5000 times smaller than the large reward (top left), invisible on the heatmap. The optimal strategy (overlaid) is to always move up or left.

We now simulate a larger, 44-edge network to perform this task. The architecture is equivalent to a densely connected neural network, with 6, 4, 4, and 1 node in each layer. Note however that our network is *not* feed forward, and thus we place our four output nodes in the third layer, and have each one represent a move (up, down, left, right). The row and state values are rescaled [0, 0.5, 1] and imposed as two inputs in the first layer. Their inverses [1, 0.5, 0] are also imposed in the first layer, and the final two nodes in this layer are always set to 0 and 1, respectively. The single node in the final layer is set to 0.5, making 7 total inputs, only 4 of which change (with the state). When training (and testing) the system is randomly placed in a new state every 5 steps.

2.4.1 Results

As before, we randomly initialize our network ($U_{G,i}$ from $\mathcal{N}(1.5, 0.1)$) and batch updates every 50 training steps. We train 10 trials for 300,000 steps, reduce ϵ from 0.1 to 0 linearly in each trial, and report the average rewards for each section of training in purple in Fig. 4B, with their average overlaid in black. In 8 of our 10 trials, our system finds one of the optimal strategies (several are equivalent). Note that the system cannot sit on the 'best strategy' line while $\epsilon > 0$, which is the case until the very end of training.

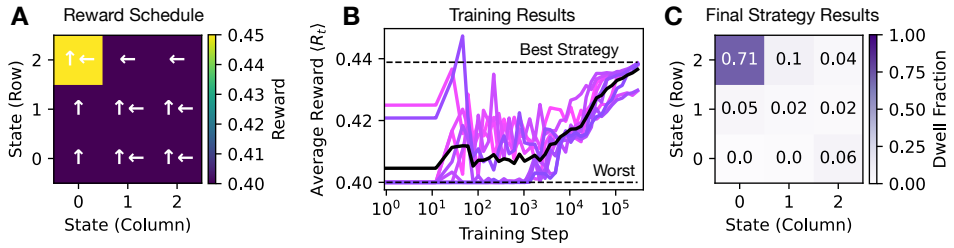

Figure 4: Navigation Task. (A) Reward schedule, with all states approximately equal except for the target (upper left) state. There is no reward noise. The optimal action in each grid is indicated via arrows. (B) Average reward over training for 10 trials (purples) is overlaid with their average (black). The single purple spike above the best average strategy is a result of a few lucky runs, and that binning is narrower early in training. (C) The 10 strategies at the end of training are each simulated for 10,000 steps (reset randomly every 5 steps), and the fraction of total time spent in each state is shown as a heatmap.

After training, we simulate each trained agent operating in the environment for 10,000 steps, still randomly setting their location every 5 steps, but with no random actions beyond this ($\epsilon = 0$). We find the agents spend most of their time in the high-reward square, as shown in Fig. 4C.

3 Discussion

Our experiments illustrate that the reinforcement learning (RL) scheme developed for a physical network performs well on an MDP comprising four states and four actions, and on a navigational task in a two-dimensional grid, with nine states and four actions. Trained agents find optimal strategies with few exceptions. While the above algorithm is quite simple, its development required engaging with two atypical features when compared to digital RL:

- **Parameters and outputs in our system are bounded.** As a result, we must take care not to define rewards or discount factors such that the system cannot produce the required outputs.
- **CLLNs are not 'feed forward'.** A voltage anywhere in the network (or a change in resistance anywhere) potentially impacts every output. Thus, our protocol required us to train unselected outputs to remain stationary. The limitations this poses for the complexity of agent strategies is a subject for future study, as are the advantages of a network capable of sensing inputs from any location.

Further, envisioning a physical system performing our scheme necessitates a somewhat 'unnatural' step backwards in time:

1. System evaluates state S_t and selects action A_t
2. Environment returns reward R_t, and switches to S_{t+1}
3. System evaluates state S_{t+1} and calculates L_t
4. System updates based on **previous** state S_t and L_t

This requires storing and reapplying S_t after the environment has moved on (S_{t+1}). It is possible that some constructed hybrid state (involving S_{t+1} and L) could allow the system to avoid such 'backtracking' and still train successfully.

Even our current simple implementation required some external control, to implement the randomization and the max function in the epsilon-greedy algorithm, and to hold and swap back S_t as discussed above. While a future CLLN could certainly be trained to perform these functions, the system is then being asked to learn the learning algorithm before succeeding at the task. Other schemes in the RL literature are far more natural to include. For instance, implementing both a policy function and value function could be done with a single network, or by dividing the network in two. Our adaptation of tabular Q-learning to a CLLN substrate is illustrated in an open-source codebase.

By virtue of being a simulation, our CLLN has several features of the digital world. However when physically constructed, these aspects change. First, in hardware, imperfections in components and measurements can manifest as biased elements, hamstringing the learning process and preventing capture of subtle learning signals (Dillavou et al., 2025). Such effects are broadly avoided (for good reason) in RL and machine learning, but spotlight a distinct advantage that these algorithms have over physically instantiated learning systems like the brain.

Second, the notion of 'damage' is somewhat meaningless in simulation, but has very real consequences for biological systems that rely on a physical substrate. As CLLNs are collections of self-adjusting elements and can retrain after damage (Dillavou et al., 2022), developing architectures and algorithms to allow on-the-fly recovery in RL tasks is an exciting research direction, and one that makes little sense in the digital domain.

Third, the energy used by the processor in simulating our CLLN is independent of the resistances and currents *in silico*. However a physically constructed version burns power directly proportional

to these values, and small modifications to the learning have been demonstrated to bias physically instantiated CLLNs towards low power solutions (Stern et al., 2024). Perhaps networks could likewise be biased into states more robust to physical damage, or with faster input-output transmission. Modifications to learning algorithms that account for such secondary goals are an exciting avenue for future work, and as above, make little sense in digital systems.

4 Conclusion

This work highlights a new type of analog, distributed system, a Contrastive Local Learning Network (CLLN), whose features make it a natural fit for reinforcement learning (RL), particularly in the big world framework. We simulated two CLLNs, and developed a training scheme to leverage their self-learning dynamics to enact Q Learning on one Markov decision process each, the first with four states and four actions, the second with nine states (3×3 grid) and four actions (cardinal direction moves). The CLLN agents were largely successful, achieving near optimal rewards across trials. Our scheme highlighted several aspects of typical RL algorithms that are less natural to apply to physical self-learning networks, most including some form of additional nontrivial memory. The limitations associated with external memory make the big world an attractive framework for developing RL methods entirely outside the pristine digital realm, towards advantages like low-power operability and fault tolerance. To support further consideration of CLLNs as a physical substrate for RL, we shared the codebase demonstrating tabular RL on a simulated CLLN [github]. Finally, we discussed disadvantages like imperfect components and secondary goals such as low-power operation and robustness to damage, all of which may be improvable via modified training methods in CLLNs, a process that makes little sense in the digital domain.

Acknowledgments

This work is supported by funding from the UK Advanced Research + Invention Agency (ARIA) under project NACB-SE02-P06 and ARIA and Pillar VC under the Encode: AI for Science Fellowship.

References

Vijay Balasubramanian. Brain power. *Proceedings of the National Academy of Sciences*, 118(32): e2107022118, August 2021. DOI: 10.1073/pnas.2107022118.

Karen Sg Chua, Yee-Sien Ng, Samantha Gm Yap, and Chek-Wai Bok. A Brief Review of Traumatic Brain Injury Rehabilitation. *Annals of the Academy of Medicine, Singapore*, 36(1):31–42, January 2007. ISSN 0304-4602. DOI: 10.47102/annals-acadmedsg.V36N1p31.

Sam Dillavou, Menachem Stern, Andrea J. Liu, and Douglas J. Durian. Demonstration of Decentralized Physics-Driven Learning. *Physical Review Applied*, 18(1):014040, July 2022. DOI: 10.1103/PhysRevApplied.18.014040.

Sam Dillavou, Benjamin D. Beyer, Menachem Stern, Andrea J. Liu, Marc Z. Miskin, and Douglas J. Durian. Machine learning without a processor: Emergent learning in a nonlinear analog network. *Proceedings of the National Academy of Sciences*, 121(28):e2319718121, July 2024. DOI: 10.1073/pnas.2319718121.

Sam Dillavou, Marcelo Guzman, Andrea J. Liu, and Douglas J. Durian. Understanding and Embracing Imperfection in Physical Learning Networks, May 2025.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, September 2021. ISSN 1573-0565. DOI: 10.1007/s10994-021-05961-4.

Mohamed Elsayed, Gautham Vasan, and A Rupam Mahmood. Streaming deep reinforcement learning finally works. *arXiv preprint arXiv:2410.14606*, 2024.

Michael C. Granovetter, Sophia Robert, Leah Ettensohn, and Marlene Behrmann. With childhood hemispherectomy, one hemisphere can support—but is suboptimal for—word and face recognition. *Proceedings of the National Academy of Sciences*, 119(44):e2212936119, November 2022. DOI: 10.1073/pnas.2212936119.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Khurram Javed and Richard S Sutton. The big world hypothesis and its ramifications for artificial intelligence. In *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*, 2024.

Terrence Mak, Kai-Pui Lam, H. S. Ng, Guy Rachmuth, and Chi-Sang Poon. A CMOS Current-Mode Dynamic Programming Circuit. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(12):3112–3123, December 2010. ISSN 1558-0806. DOI: 10.1109/TCSI.2010.2052661.

Terrence S.T. Mak, K.P. Lam, H.S. Ng, G. Rachmuth, and C.-S. Poon. A Current-Mode Analog Circuit for Reinforcement Learning Problems. In *2007 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1301–1304, May 2007. DOI: 10.1109/ISCAS.2007.378410.

C. Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, October 1990. ISSN 1558-2256. DOI: 10.1109/5.58356.

Mantas Mikaitis, Garibaldi Pineda García, James C. Knight, and Steve B. Furber. Neuromodulated Synaptic Plasticity on the SpiNNaker Neuromorphic System. *Frontiers in Neuroscience*, 12, February 2018. ISSN 1662-453X. DOI: 10.3389/fnins.2018.00105.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Javier R. Movellan. Contrastive Hebbian Learning in the Continuous Hopfield Model. In David S. Touretzky, Jeffrey L. Elman, Terrence J. Sejnowski, and Geoffrey E. Hinton (eds.), *Connectionist Models*, pp. 10–17. Morgan Kaufmann, January 1991. ISBN 978-1-4832-1448-1. DOI: 10.1016/B978-1-4832-1448-1.50007-X.

Benjamin Scellier and Yoshua Bengio. Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers in Computational Neuroscience*, 11, 2017. ISSN 1662-5188. DOI: 10.3389/fncom.2017.00024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Menachem Stern, Daniel Hexner, Jason W. Rocks, and Andrea J. Liu. Supervised Learning in Physical Networks: From Machine Learning to Learning Machines. *Physical Review X*, 11(2): 021045, May 2021. DOI: 10.1103/PhysRevX.11.021045.

Menachem Stern, Sam Dillavou, Dinesh Jayaraman, Douglas J. Durian, and Andrea J. Liu. Training self-learning circuits for power-efficient solutions. *APL Machine Learning*, 2(1):016114, February 2024. ISSN 2770-9019. DOI: 10.1063/5.0181382.

Xueying Wang, John A Hayes, Ann L Revill, Hanbing Song, Andrew Kottick, Nikolas C Vann, M Drew LaMar, Maria Cristina D Picardo, Victoria T Akins, Gregory D Funk, and Christopher A Del Negro. Laser ablation of Dbx1 neurons in the pre-Bötzinger complex stops inspiratory rhythm and impairs output in neonatal mice. *eLife*, 3:e03427, July 2014. ISSN 2050-084X. DOI: 10.7554/eLife.03427.

Appendix

Algorithm 1 Q-Learning with a Contrastive Local Learning Network (CLLN)

given CLLN with nodes $n \in \{1, \ldots, M\}$, connected by edges $i \in \{1, \ldots, E\}$ each with a gate voltage $U_{G,i}$; environment (R, step); hyperparameters $\eta, \gamma, \epsilon, R_{\text{reset}}$, total steps T, batch period B
initialize $U_{G,i} \sim \mathcal{N}(1.5, 0.1)$, clipped to $[U_G^{\min}, U_G^{\max}]$
initialize accumulated update $\delta U_{G,i} \leftarrow 0$ for all i
sample initial state $S_0 \sim \text{Uniform}\{\text{states}\}$

for $t = 0, 1, \ldots, T-1$ **do**
  // Free state for S_t
  impose input voltages encoding S_t; equilibrate (free) $\Rightarrow$ node voltages $U^F(S_t)$
  record edge drops $\Delta U_i^F = U_{\text{from}}^F - U_{\text{to}}^F$ and free outputs $\mathcal{F}(S_t) = \{U_n^{\text{out},F}\}_{n=1}^{N_A}$

  // ϵ-greedy action selection
  $\epsilon_t \leftarrow \epsilon_0 \,(1 - t/T) + \epsilon_\infty \,(t/T)$
  if $\text{rand}() < \epsilon_t$ **then**
    $A_t \leftarrow \text{Uniform}\{1, \ldots, N_A\}$
  else
    $A_t \leftarrow \arg\max_n \mathcal{F}(S_t)$
  end if

  $R_t \leftarrow R(S_t, A_t)$, $S_{t+1} \leftarrow \text{step}(S_t, A_t)$ *// Record reward and step environment*

  // Bootstrap target using evaluation of next state S_{t+1}
  impose S_{t+1}; equilibrate (free) $\Rightarrow \mathcal{F}(S_{t+1})$
  $L_t \leftarrow R_t + \gamma[\max(\mathcal{F}(S_{t+1})) - \text{mean}(\mathcal{F}(S_{t+1}))]$

  // Clamped state for S_t
  labels $L_n \leftarrow U_n^{\text{out},F}(S_t)$ for $n \neq A_t,\ L_{A_t} \leftarrow L_t$ *// measured outputs for actions not taken*
  nudge outputs: $U_n^{\text{out},C} \leftarrow (1-\eta)\, U_n^{\text{out},F} + \eta\, L_n$ for $n = 1, \ldots, N_A$
  impose inputs of S_t *and* $U_n^{\text{out},C}$ on outputs; equilibrate $\Rightarrow U^C(S_t)$
  record clamped edge drops $\Delta U_i^C = U_{\text{from}(i)}^C - U_{\text{to}(i)}^C$
  // Coupled-Learning contrastive update (local rule)
  for $i = 1, \ldots, E$ **do**
    $\delta U_{G,i} \leftarrow \delta U_{G,i} + {}^{\alpha}/_{\eta}\Big[\big(\Delta U_i^F\big)^2 - \big(\Delta U_i^C\big)^2\Big]$
  end for
  // Apply batched update every B steps
  if $(t+1) \bmod B = 0$ **then**
    $\delta U_{G,i} \leftarrow \text{clip}\,(\delta U_{G,i}, -c, +c)$
    $U_{G,i} \leftarrow \text{clip}\big(U_{G,i} + \delta U_{G,i},\, U_G^{\min}, U_G^{\max}\big)$
    $\delta U_{G,i} \leftarrow 0$
  end if
  // State carry-over and periodic random reset
  $S_t \leftarrow S_{t+1}$
  if $(t+1) \bmod R_{\text{reset}} = 0$ **then**
    $S_t \leftarrow \text{Uniform}\{\text{states}\}$
  end if
end for