Biological Neurons vs Deep Reinforcement Learning: Sample efficiency in a simulated game-world

Forough Habibollahi [*]
Department of Biomedical Engineering
University of Melbourne
Melbourne, Australia

Moein Khajehnejad [*]
Department of Data Science and AI
Monash University
Melbourne, Australia

Amitesh Gaurav
Cortical Labs Pty Ltd
Melbourne, Australia

Brett J. Kagan
Cortical Labs Pty Ltd
Melbourne, Australia

Abstract

How do synthetic biological systems and artificial neural networks compete in their performance in a game environment? Reinforcement learning has undergone significant advances, however remains behind biological neural intelligence in terms of sample efficiency. Yet most biological systems are significantly more complicated than most algorithms. Here we compare the inherent intelligence of in vitro biological neuronal networks to state-of-the-art deep reinforcement learning algorithms in the arcade game 'pong'. We employed DishBrain, a system that embodies *in vitro* neural networks with in silico computation using a high-density multielectrode array. We compared the learning curve and the performance of these biological systems against time-matched learning from DQN, A2C, and PPO algorithms. Agents were implemented in a reward-based environment of the 'Pong' game. Key learning characteristics of the deep reinforcement learning agents were tested with those of the biological neuronal cultures in the same game environment. We find that even these very simple biological cultures typically outperform deep reinforcement learning systems in terms of various game performance characteristics, such as the average rally length implying a higher sample efficiency. Furthermore, the human cell cultures proved to have the overall highest relative improvement in the average number of hits in a rally when comparing the initial 5 minutes and the last 15 minutes of each designed gameplay session.

1 Introduction

The concept of reinforcement learning dates back to the early days of cybernetics and has been studied in statistics, psychology, neuroscience, and computer science. In the past decade, its use has become increasingly popular in the fields of machine learning and artificial intelligence. Its promise is highly convincing - a way of programming agents by rewarding and punishing them without having to specify how the task is to be accomplished. However, to deliver on this promise, formidable computational obstacles must be overcome. Reinforcement learning (RL) implies learning the best policy to maximize an expected cumulative long-term reward throughout many steps in order to achieve complex objectives (goals) [1]. A deep reinforcement learning (deep RL) approach integrates artificial neural networks with a reinforcement learning framework that helps the system to achieve its goals [2]. That is, it maps states and actions to the rewards they bring, combining

36th Conference on Neural Information Processing Systems (NeurIPS 2022), New Orleans, USA.

[*]Indicates equal contribution.

function approximation and target optimization. Reinforcement algorithms that incorporate deep neural networks have been developed to beat human experts playing numerous Atari video games [3], poker [4], multiplayer contests [5], and complex board games, including go and chess [6, 7, 8]. Nevertheless, reinforcement learning still faces real challenges including but not limited to complexities in the selection of reward structure, sample inefficiency [9, 10], reproduciblity issues [11], as well as requiring high levels of computing power [12]. All of these suggest that deep RL algorithms may differ fundamentally from the underlying mechanisms of human learning while also being too inefficient to be accepted as plausible models of human learning [10].
It was recently demonstrated that by using electrophysiological stimulation and recording in a real-time closed-loop system with a monolayer of living biological neurons, these cells could be trained to significantly improve performance in the simulated 'pong' gameworld [13]. The question arises as to whether this observed performance is notable in comparison to that of reinforcement learning at the same task. To compare the performance and efficiency of such a biological neuronal network (BNN) to that of deep RL, we use data gathered from the *DishBrain* system [13] against time-matched learning from DQN, A2C & PPO algorithms. *DishBrain* is a novel system shown to display biological intelligence by harnessing the inherent adaptive computation of neurons within a simulated gameplay environment in real time through closed-loop stimulation and recordings. In this system, *in vitro* neuronal networks are integrated with *in silico* computing via high-density multi-electrode arrays (HD-MEAs). We investigate whether these elementary learning systems achieve performance levels which can compete with state-of-the-art deep RL algorithms while varying the input information density required for training the RL algorithms to also determine the impact of information sparsity and ensure suitable comparisons to the biological system. This is the first comparison between a synthetic biological intelligence system and state-of-the-art RL algorithms.

2 Methods

2.1 DishBrain System

To investigate the learning efficiency of the BNNs in the task-present state, recordings from cultures integrated onto an MEA were used. The *DishBrain* environment is a low latency, real-time system which interacts with the MEA (Maxwell Biosystems, Switzerland) software to allow closed-loop stimulation and recording. *DishBrain* was utilised to embody neural cultures in a virtual game-world, to simulate the arcade game 'Pong'. Stimulation was applied into a predefined bounded two-dimensional sensory area consisting of 8 sensory electrodes to communicate ball's position on the x and y-axis using a combination of rate coding (4Hz - 40Hz) electrical pulses and place coding, respectively. The movement of the paddle was controlled by the level of electrophysiological activity measured in a predefined "motor area", which was recorded in real time (see Supplementary Materials for details). The cells also received information about the closed-loop response to their control of the paddle. Each recording session of the cultures was 20 minutes. This equaled an average number of 70 training episodes.

2.2 Deep Reinforcement Learning Algorithms

We use three state-of-the-art deep reinforcement learning algorithms: Deep Q Network (DQN) [3], Advantage Actor-Critic (A2C) [14] and Proximal Policy Optimization (PPO) [15], established to have good performance in Atari games. Benefiting from deep learning advantages in automated feature extraction, specifically exploiting Convolutional Neural Networks (CNN) in their structures, these methods are robust tools in reinforcement tasks, particularly in games where the system's input is an image. We only report the results of the deep RL algorithms in a design where the current state is a tensor of the difference of pixel values from the two most recent frames (i.e. another 40×40 grayscale pixel image). This current state is then input into the CNN to obtain the selected action.
However, to account for potential adversaries resulting from the high dimensionality of the image input, [16], we also designed two additional types of low-dimensional input information: 1) $[ball_x, ball_y, paddle_{top_y}, paddle_{bottom_y}]$, and 2) $[ball_x, ball_y]$, $ball_y \in \{1, 2, \cdots, 8\}$ as we divide the y-axis to 8 equal segments mimicking the 8 sensory electrodes which place code the $ball_y$ in the biological cultures. We compared all three different designs with the performance of biological cultures and observed no significant difference in the outcome of these comparisons (see the additional results in Supplementary Materials). In the training phase of all RL algorithms, we ran them for 40 random seeds and a total of 70 episodes for each seed (similar to BNNs). These seeds imply 40

different neural networks trained separately, resembling 40 different recorded cultures. We report the average value of each metric among all seeds.

3 Results

We studied both human cortical cells (HCCs; 174 sessions) and mice cortical cells (MCCs; 110 sessions) and compared them to the introduced RL baseline methods. To determine how the learning arises both in the cultures and the baseline methods, key gameplay characteristics were examined. The hit counts in the gameplay in each episode before the ball was missed for the first time, the number of times the paddle failed to intercept the ball on the initial serve (aces), and the number of long rallies (> 3 consecutive hits) were calculated for this data. For comparison purposes, we first mapped every 70-episode run of each RL algorithm to a real-time equivalent of 20 minutes by first normalizing to the actual total length of each run in minutes and then multiplying by 20 minutes.

The DQN algorithm is outperformed by all groups in the highest level of average hits per rally

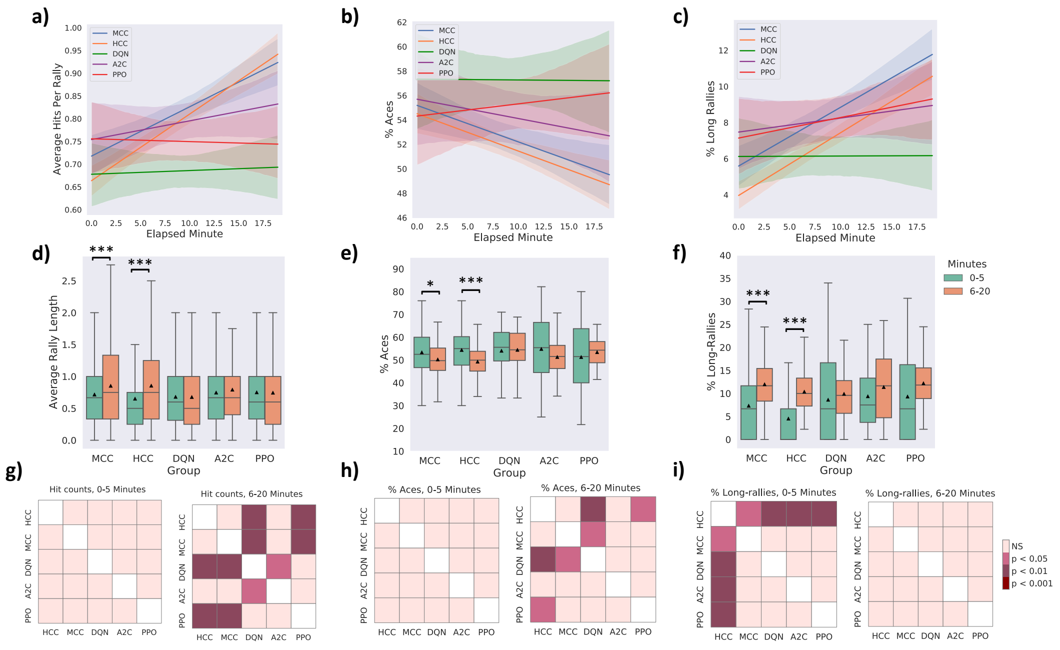


Figure 1: **a)** Average number of hits per rally, **b)** % aces, and **c)** % long rallies over 20 minutes real-time equivalent of training RL algorithms and biological cultures. A regressor line on the mean values with a 95% confidence interval highlights the learning trends. **d)** Average rally length in the first 5 minutes and last 15 minutes of the sessions. **e)** Average % of aces within groups and over time. **f)** Average % of long-rallies (>3) performed in each interval. **g, h, and i)** Pairwise Tukey's post hoc test among groups in each time interval and for **g)** average rally length, **h)** % aces, and **i)** % long rallies. Box plots show interquartile range, with bars demonstrating 1.5X interquartile range, the line marks the median and ▲ marks the mean. Error bands = 1 SE

achieved, while the biological cultures (i.e. HCC and MCC) outperform all the RL baselines (see Subfigure 1.a). This indicates that the cultures represent faster growing learning rates. Subfigure 1.b compares the % of missed balls on the initial serve, i.e. aces. HCC and MCC achieve the lowest % of aces in Subfigure 1.b. The % of long rallies has an increasing trend in all groups with the highest levels achieved by MCC and HCC as illustrated in Subfigure 1.c.

Next, for all the groups, we compared the key activity metrics in the first 5 minutes versus the last 15 minutes in each session. Our aim was to identify any significant improvement in the learning process within each group. Subfigure 1.d compares the average rally length between the two defined time intervals. The results imply that the intra-group improvement in the length of rallies is significant only in the biological groups (One-way ANOVA test). Subfigure 1.e represents the change in the average % of aces over time. A significant decrease in the number of aces implies an improved game performance. Only MCC and HCC groups had a significant decrease in the average % of aces (One-way ANOVA test). Subfigure 1.f shows that % of long rallies in the first 5 minutes versus the

last 15 minutes only significantly increased for the biological cultures (One-way ANOVA test). Pairwise inter-group comparison was carried out for both time intervals and all metrics using Tukey's post hoc test represented in Subfigures 1.g, h, and i for hit counts, % of aces, and % of long rallies. It should be noted that while certain metrics of the performance of the deep RL methods comes closets to the biological cultures, the density of input information is starkly different between RL methods and the biological cultures. While RL agents receive pixel data with a density of 40×40 pixels, biological cultures only receive input from 8 stimulation points with a given integer rate code of 4Hz–40Hz, highlighting important efficiency differences in informational input between these learning systems. The possibility of the higher input information dimensionality having adverse effects on the overall sample efficiency of these RL algorithms was further nullified by evaluating the two alternative input structures as discussed above.
To account for potential effects of paddle movement speed on the success rate of paddle control, we derived the average paddle movement (in pixels) for all groups. Subfigure S6.a represents these results with DQN having a significantly higher average paddle movement compared to biological cultures (Pairwise Tukey's post hoc). Interestingly, the higher paddle movement speed of the RL algorithms is not reflected as better game performance according to our results.
Subfigure S6.b compares the relative improvement in the performance of different groups over time. This measure identifies the relative increase in the average accurate hit counts in the second 15 minutes of the game compared to the first 5 minutes. The HCC group shows the highest improvement in time and performing Tukey's post hoc tests showed that the difference in this measure is significant between HCC and PPO, as well as HCC and DQN. The MCC group also outperforms DQN.

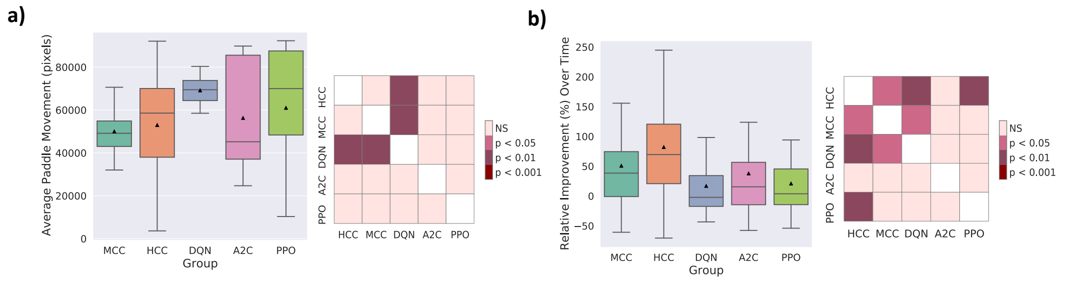


Figure 2: a) The average paddle movement in pixels and pairwise Tukey's post hoc test representing the significance of the differences. **b)** Relative improvement (%) in the average hit counts between the first 5 minutes and the last 15 minutes of all sessions in each separate group and pairwise Tukey's post hoc test.

4 Discussion

In this work, we compared the performance of BNNs with that of state-of-the-art deep RL algorithms in the game environment of *pong*. The results show that the game performance of the deep RL algorithms in terms of relative learning improvement in time and the ultimate number of average hits per rally is outperformed by biological cultures. Furthermore, their performance in the average rally length and percentage of aces only matches those of neuronal cultures at best. The RL algorithms showed the lowest sample efficiency having the lowest improvement in learning given the 70 episode training duration provided for all the groups.
This is the first comparison between a synthetic biological intelligence system and state-of-the-art RL algorithms. This early work establishes that even the most rudimentary SBI systems with limited informational input are a viable learning system that can compete and even defeat the established RL algorithms which receive significant more information input. Coupled with the promise of significant gains in power efficiencies, flexibility of tasks, and as data representation to the SBI system is improved, these biological intelligence systems present a compelling pathway for realizing real-time learning unachievable by current silicon-based approaches.

A Supplementary Materials

A.1 Cell Culture

Neural cells were cultured either from the cortices of E15.5 mouse embryos or differentiated from human induced pluripotent stem cells via a dual SMAD inhibition (DSI) protocol or through a lentivirus based NGN2 direct differentation protocols as previously described [13]. Cells were cultured until plating. For primary mouse neurons this occurred at day-in-vitro (DIV) 0, for DSI cultures this occurred at between DIV 30 - 33 depending culture development, for NGN2 cultures this occured at DIV 3.

A.2 MEA Setup and Plating

MaxOne Multielectrode Arrays (MEA; Maxwell Biosystems, AG, Switzerland) was used and is a high-resolution electrophysiology platform featuring 26,000 platinum electrodes arranged over an 8 mm2. The MaxOne system is based on complementary meta-oxide-semiconductor (CMOS) technology and allows recording from up to 1024 channels. MEAs were coated with either polyethylenimine (PEI) in borate buffer for primary culture cells or Poly-D-Lysine for cells from an iPSC background before being coated with either 10 µg/ml mouse laminin or 10 µg/ml human 521 Laminin (Stemcell Technologies Australia, Melbourne, Australia) respectively to facilitate cell adhesion.Approximately 10^6 cells were plated on MEA after preparation as per [13]. Cells were allowed approximately one hour to adhere to MEA surface before the well was flooded. The day after plating, cell culture media was changed for all culture types to BrainPhys™ Neuronal Medium (Stemcell Technologies Australia, Melbourne, Australia) supplemented with 1% penicillin-streptomycin. Cultures were maintained in a low O2 incubator kept at 5% CO2, 5% O2, 36°C and 80% relative humidity. Every two days, half the media from each well was removed and replaced with free media. Media changes always occurred after all recording sessions.

A.3 DishBrain platform and electrode configuration

DishBrain was utilised to embody neural cultures in a virtual game-world, to simulate the arcade game 'Pong'. Sensory stimulation was applied into a predefined bounded two-dimensional sensory area consisting of 8 sensory electrodes to communicate ball's position on the x and y-axis using a combination of rate coding (4Hz - 40Hz) electrical pulses and place coding, respectively. The movement of the paddle was controlled by the level of electrophysiological activity measured in a predefined "motor area", which was recorded in real time. The cells also received information about the closed-loop response to their control of the paddle.
It was possible to deliver five types of input. Either the sensory stimulation as explained above, or one of four feedback protocols: Unpredictable, Predictable, Silent, or No-feedback. The reported results in this work are obtained using the unpredictable feedback protocol. Cultures received unpredictable stimulation when they missed connecting the paddle with the 'ball', i.e. when a 'miss' occurred. Using a feedback stimulus at a voltage of 150 mV and a frequency of 5 Hz, unpredictable external stimulus could be added to the system. Random stimulation took place at random sites over the 8 predefined sensory electrodes at random timescales for a period of four seconds, followed by a configurable rest period of four seconds where stimulation paused, then the next rally began. Each recording session of the cultures was 20 minutes. This equaled an average number of 70 training episodes.

Figure S1 illustrates the the input information, feedback loop setup, and electrode configurations in the DishBrain system.

The current DishBrain platform is configured as a low-latency, real-time MEA control system with on-line spike detection and recording software. The DishBrain platform provides on-line spike detection and recording configured as a low-latency, real-time MEA control. The DishBrain software runs at 20 kHz and allows recording at an incredibly fine timescale. There is the option of recording spikes in binary files, and regardless of recording, they are counted over a period of 10 milliseconds (200 samples), at which point the game environment is provided with how many spikes are detected in each electrode in each predefined motor region as described below. Based on which motor region the spikes occurred in, they are interpreted as motor activity, moving the 'paddle' up or down in the virtual space. As the ball moves around the play area at a fixed speed and bounces

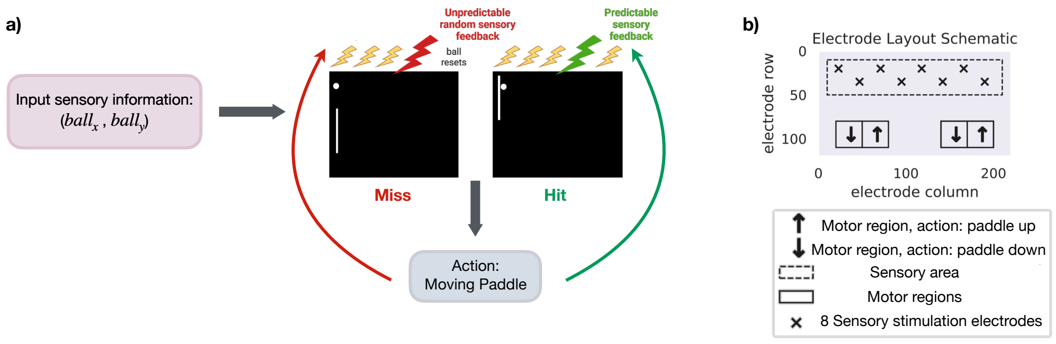

a)
Input sensory information: $(ball_x, ball_y)$
Unpredictable random sensory feedback
ball resets
Predictable sensory feedback
Miss
Hit
Action: Moving Paddle
b)
Electrode Layout Schematic
electrode row
electrode column
0
50
100
0
100
200
Motor region, action: paddle up
Motor region, action: paddle down
Sensory area
Motor regions
8 Sensory stimulation electrodes

Figure S1: **a)** DishBrain feedback loop setup. **b)** Electrode configuration and predefined sensory and motor regions. Figures adapted and modified from [13]

off the edge of the play area and the paddle, the pong game is also updated at every 10ms interval. Once the ball hits the edge of the play area behind the paddle, one rally of pong has come to an end. The game environment will instead determine which type of feedback to apply at the end of the rally: random, silent, or none. Feedback is also provided when the ball contacts the paddle under the standard stimulus condition. A 'stimulation sequencer' module tracks the location of the ball relative to the paddle during each rally and encodes it as stimulation to one of eight stimulation sites. Each time a sample is received from the MEA, the stimulation sequencer is updated 20,000 times a second, and after the previous lot of MEA commands has completed, it constructs a new sequence of MEA commands based on the information it has been configured to transmit based on both place codes and rate codes. The stimulations take the form of a short square bi-phasic pulse that is a positive voltage, then a negative voltage. This pulse sequence is read and applied to the electrode by a Digital to Analog Converter (or DAC) on the MEA. A real-time interactive version of the game visualiseris available at https://spikestream.corticallabs.com/. Alternatively, cells could be recorded at 'rest' in a gameplay environment where activity was recorded to move the paddle but no stimulation was delivered, with corresponding outcomes still recorded. Using this spontaneous activity alone as a baseline, the gameplay characteristics of a culture were determined. Low level code for interacting with Maxwell API was written in C to minimize processing latencies-so packet processing latency was typically $<50\ \mu$s. High-level code was written in Python, including configuration setups and general instructions for game settings. A 5 ms spike-to-stim latency was achieved, which was substantially due to MaxOne's inflexible hardware buffering.

Figure S2 illustrates a schematic view of Software components and data flow in the DishBrain closed loop system.

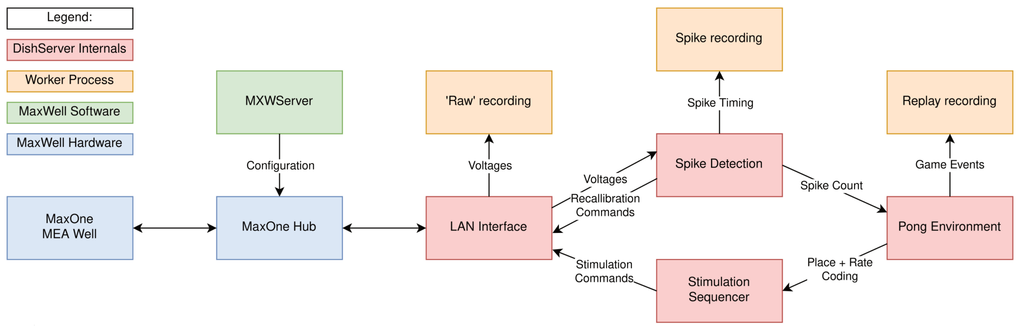
A.4 Deep Reinforcement Learning Algorithms

Deep Q Network (DQN): The utilized DQN algorithm begins by extracting spatiotemporal features from inputs, such as the movement of the ball in game of 'Pong'. Multiple fully connected layers are used to process the final feature map, which implicitly encodes the effects of actions. As opposed to traditional controllers that use fixed preprocessing steps, this method can adapt processing of the state based on changes in the learning signal.

Advantage Actor-Critic (A2C): In an A2C model, the total reward itself could be represented as a *value* of the state plus the advantage of the action. The *value* of each policy is learned while following it. The policy gradient can be calculated by knowing the *value* for any state. The policy network is then updated such that the probability of actions with a higher advantage values is increased. Here, the policy network (which returns a probability distribution of actions) is called the *actor*, as it tells the agents what to do. *Critic* is another network which enables the evaluation of the actions to decide whether they were good or not. In this case, policy and value are implemented as separate heads of the network, which transform the output from the common body into either probability distributions or single numbers representing the state's value. Thus, low-level features can be shared between the two networks.

Proximal Policy Optimization (PPO): PPO models are a family of policy gradient methods for reinforcement learning. The PPO method uses a slightly different training procedure: An extended

a)



b)

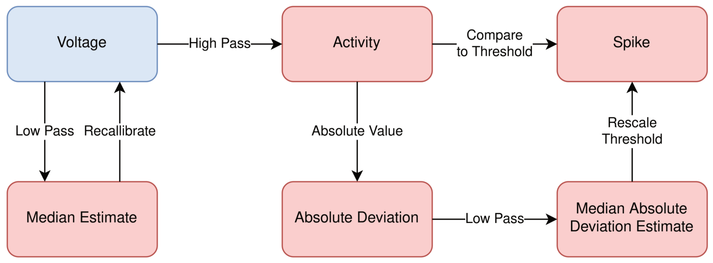


Figure S2: a, b) Schematics of software used for DishBrain. **a)** Software components and data flow in the DishBrain closed loop system. Voltage samples flow from the MEA to the 'Pong' environment, and sensory information flows from the 'Pong' environment back to the MEA, forming a closed loop. The blue rectangles mark proprietary pieces of hardware from MaxWell, including the MEA well which may contain a live culture of neurons. The green MXWServer is a piece of software provided by MaxWell which is used to configure the MEA and Hub, using a private API directly over the network. The red rectangles mark components of the 'DishServer' program, a high-performance program consisting of four components designed to run asynchronously, despite being run on a single CPU thread. The 'LAN Interface' component stores network state, for talking to the Hub, and produces arrays of voltage values for processing. Voltage values are passed to the 'Spike Detection' component, which stores feedback values and spike counts, and passes recalibration commands back to the LAN Interface. When the pong environment is ready to run, it updates the state of the paddle based on the spike counts, updates the state of the ball based on its velocity and collision conditions, and reconfigures the stimulation sequencer based on the relative position of the ball and current state of the game. The stimulation sequencer stores and updates indices and countdowns relating to the stimulations it must produce and converts these into commands each time the corresponding countdown reaches zero, which are finally passed back to the LAN Interface, to send to the MEA system, closing the loop. The procedures associated with each component are run one after the other in a simple loop control flow, but the 'Pong' environment only moves forward every 200th update, short-circuiting otherwise. Additionally, up to three worker processes are launched in parallel, depending on which parts of the system need to be recorded. They receive data from the main thread via shared memory and write it to file, allowing the main thread to continue processing data without having to hand control to the operating system and back again. **b)** Numeric operations in the real-time spike detection component of the DishBrain closed loop system, including multiple IIR filters. Running a virtual environment in a closed loop imposes strict performance requirements, and digital signal processing is the main bottleneck of this system, with close to 42 MB of data to process every second. Simple sequences of IIR digital filters is applied to incoming data, storing multiple arrays of 1024 feedback values in between each sample. First, spikes on the incoming data are detected by applying a high pass filter to determine the deviation of the activity, and comparing that to the MAD, which is itself calculated with a subsequent low pass filter. Then, a low pass filter is applied to the original data to determine whether the MEA hardware needs to be re-calibrated, affecting future samples. This system was able to keep up with the incoming data on a single thread of an Intel Core i7-8809G. Figures adapted from [13].

set of samples is taken from the environment, and then the advantage is estimated for the whole set or sequence of samples before several epochs of training are performed To estimate policy gradients, instead of using the gradient of action probabilities, the PPO method uses a different objective: the ratio between the new and the old policy scaled by the advantages.

Algorithm 1 Deep Q Network (DQN) with Experience Replay

```
Require:
 1: $\mathcal{D}$: Replay memory with capacity $N$ (Default: 10000)
 2: $\theta$: Initial network parameters
 3: $\tilde{\theta}$: Copy of $\theta$
 4: $N_b$: Training batch size (Default: 5)
 5: $\tilde{N}$: Target network update frequency (Default: 10)
 6: $x_t$: Input matrix at time $t$
 7: $S$: Number of seeds (Default: 40)
 8: $e_{max}$: Maximum number of episodes (Default: 70)
 9: for seed $\in \{1, \cdots, S\}$ do
10:     for episode $e \in \{1, \cdots, e_{max}\}$ do
11:         Set state $s_1 \leftarrow x_1$ and preprocess $\phi_1 = \phi(s_1)$
12:         $t = 1$
13:         while $\phi_t$ is non-terminal do
14:             With probability $\epsilon$ select a random action $a_t$
15:             otherwise select $a_t = max_a Q^*(\phi(s_t), a; \theta)$
16:             Execute action $a_t$ and observe reward $r_t$ and input $x_{t+1}$
17:             Set new state $s_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
18:             Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
19:             Sample random minibatch of $N_b$ transitions$(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
20:             Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
21:             Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$
22:             Replace target parameters $\tilde{\theta} \leftarrow \theta$ every $\tilde{N}$ steps
23:             $t = t + 1$
24:         end while
25:     end for
26: end for
```

Algorithm 2 Advantage Actor-Critic (A2C)

```
Require:
 1: $\theta_v$: Initial parameter vector for the value net (critic)
 2: $\theta_\pi$: Initial parameter vector for the policy net (actor)
 3: $N$: Number of consecutive steps to play current policy in the environment (Default: 5)
 4: $x_t$: Input matrix at time $t$
 5: $S$: Number of seeds (Default: 40)
 6: $e_{max}$: Maximum number of episodes (Default: 70)
 7: for seed $\in \{1, \cdots, S\}$ do
 8:     $t = 1$
 9:     $e = 1$
10:     repeat
11:         $\partial\theta_\pi \leftarrow 0$ and $\partial\theta_v \leftarrow 0$
12:         $t_{start} = t$
13:         Set state $s_t \leftarrow x_t$ and preprocess $\phi_t = \phi(s_t)$
14:         repeat
15:             Select $a_t$ according to $\pi(a_t|\phi_t; \theta)$
16:             Execute action $a_t$ and observe reward $r_t$ and input $x_{t+1}$
17:             Set new state $s_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
18:             $t \leftarrow t + 1$
19:         until $\phi_t$ is terminal or $t - t_{start} = N$
20:         $R = \begin{cases} 0 & \text{for terminal } \phi_t \\ V(\phi_t; \theta_v) & \text{for non-terminal } \phi_t \end{cases}$
21:         for $i \in \{t - 1, \cdots, t_{start}\}$ do
22:             $R \leftarrow r_i + \gamma R$
23:             Accumulate the policy gradients: $\partial\theta_\pi \leftarrow \partial\theta_\pi + \nabla_\theta \log \pi(a_i|\phi_i; \theta)\big(R - V(\phi_i, \theta_v)\big)$
24:             Accumulate the value gradients: $\partial\theta_v \leftarrow \partial\theta_v + \frac{\partial\left(R - V(\phi_i, \theta_v)\right)^2}{\partial\theta_v}$
25:         end for
26:         Update $\theta_\pi$ and $\theta_v$ using $\partial\theta_\pi$ and $\partial\theta_v$, respectively.
27:         if $\phi_t$ is terminal then
28:             $e \leftarrow e + 1$
29:         end if
30:     until $e > e_{max}$
31: end for
```

A.5 Alternative RL Input Designs: Additional Results

In this work, aiming to account for potential adversaries resulting from the increased dimensionality of the image input to the deep RL algorithms [16], we design two additional types of input information to the RL algorithms. We compare all three different designs with the performance of biological cultures. We attempt to study whether the curse of dimensionality and increased size of

Algorithm 3 Proximal Policy Optimization (PPO)

```
Require:
1: $\theta_0$: Initial policy parameter vector
2: $\epsilon$: Clipping threshold (Default: 0.2)
3: $N$: Number of consecutive steps to play current policy in the environment (Default: 5)
4: $x_t$: Input matrix at time $t$
5: $S$: Number of seeds (Default: 40)
6: $e_{max}$: Maximum number of episodes (Default: 70)
7: for seed $\in \{1, \cdots, S\}$ do
8:     $t = 1$
9:     $e = 1$
10:    repeat
11:        $t_{start} = t$
12:        Set state $s_t \leftarrow x_t$ and preprocess $\phi_t = \phi(s_t)$
13:        repeat
14:            Select $a_t$ according to $\pi(a_t|\phi_t;\theta)$
15:            Execute action $a_t$ and observe reward $r_t$ and input $x_{t+1}$
16:            Set new state $s_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
17:            $t \leftarrow t + 1$
18:        until $\phi_t$ is terminal or $t - t_{start} = N$
19:        Collect set of partial trajectories $\mathcal{D}$ on current policy $\pi$
20:        Estimate Advantages $\hat{A}_t^\pi = \sigma_t + (\gamma\lambda)\sigma_{t+1} + \cdots + (\gamma\lambda)^{N-t-1}\sigma_{N-1}$, where $\sigma_t = r_t + \gamma V(\phi_{t+1}) - V(\phi_t)$
21:        $\theta \leftarrow \text{argmax}_\theta \mathcal{L}_\theta^{CLIP}(\theta)$
22:        where $\mathcal{L}_\theta^{CLIP}(\theta) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\left[min(r_t(\theta)\hat{A}_t^\pi, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^\pi)\right]\right]$
23:        if $\phi_t$ is terminal then
24:            $e \leftarrow e + 1$
25:        end if
26:    until $e > e_{max}$
27: end for
```

the feature vectors when directly utilizing image inputs affects the comparison between biological cultures and RL algorithms in terms of their sample efficiency. The three different input categories and RL algorithm designs are introduced below:

- **IMAGE INPUT:** All the algorithms follow a common strategy although different in structure. In this design, the current state is a tensor of the difference of pixel values from the two most recent frames (i.e. another 40 × 40 grayscale pixel image). This current state is then input into the CNN to obtain the selected action. Next, based on the action taken, a reward is received, and a new state is formed. The ultimate goal is to find a policy that indicates the best action in each state to maximise the reward function.
- **PADDLE&BALL POSITION INPUT:** In this case, instead of the grayscale image, we obtain a 4-dimensional vector encoding the x and y coordinates of the ball (distance to the paddle/wall and distance to the floor in pixels) and the y coordinates of the paddle's top and bottom, all being integer values in $[1, 40]$. The current state which is the input to each algorithm is then a tensor of the difference of values from the two most recent 4-dimensional location vectors. No additional CNN layer is utilized in this case.
- **BALL POSITION INPUT:** Finally, we aim to examine a design as similar to the DishBrain system's input structure as possible. For this case, we divide the y-axis of the gameplay environment to 8 equal segments each mimicking one of the sensory electrodes in the biological cultures and place coding the information about the ball's y-axis position as an integer in the $[1, 8]$ interval. Then, the ball's x-axis position is used as the second element of this input vector being an integer value in $[4, 40]$ similar to the rate coded component of the stimulation applied to the biological cultures. No additional CNN layer is utilized in this design.

Figure S3 illustrates the comparison between the input information in the DishBrain system and the deep RL algorithms.

In addition to the results reported in Section Results which represented the findings from the IMAGE INPUT design, Figures S4, and S5 illustrate similar comparisons between the biological cultures and the two alternative RL input designs, namely: PADDLE&BALL POSITION INPUT and BALL POSITION INPUT.

Finally, accounting for potential effects of paddle movement speed, we derived the average paddle movement (in pixels) for all groups and all designs. Subfigures S6.a,c, and e represent these results

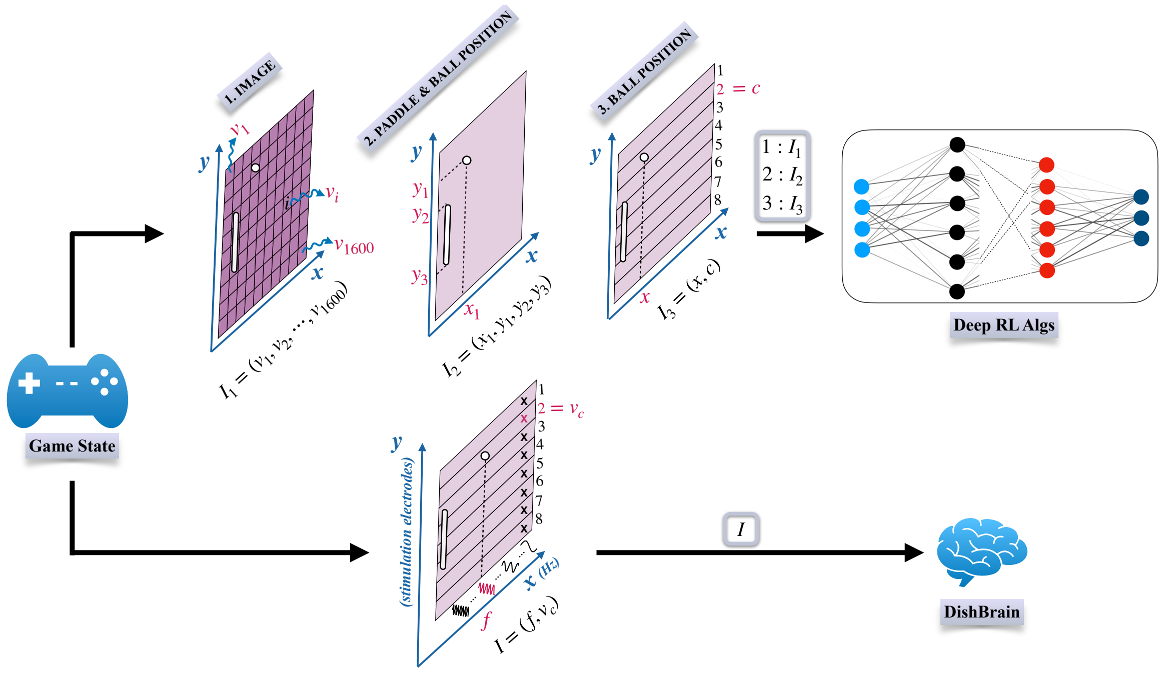


Figure S3: Schematic comparing the information feeding routes in the DishBrain system (bottom) and the three implementations of the deep RL algorithms (top). In each design, the input information to the computing module (deep RL algorithms or DishBrain) is denoted by a vector I.

for the Image Input, Paddle&Ball Position Input, and Ball Position Input designs respectively. Using pairwise Tukey's post hoc tests it was found that a consistent significant difference is present between pairs of DQN and HCC in terms of the average paddle movement with DQN having the higher average. This is when all the RL algorithms with the Paddle&Ball Position Input and Ball Position Input designs have significantly higher average paddle movement compared to both groups of biological cultures. Interestingly, the higher paddle movement speed of the RL algorithms is not reflected as better game performance according to previously discussed results.

Subfigures S6.b, d, and f compare the relative improvement in the performance of different groups over time when comparing the HCC and MCC groups to RL algorithms with Image Input, Paddle&Ball Position Input, and Ball Position Input respectively. This measure identifies the relative increase in the average accurate hit counts in the second 15 minutes of the game compared to the first 5 minutes. The HCC group shows the highest improvement in time and performing Tukey's post hoc tests showed that the difference in this measure is significant between HCC and PPO, as well as HCC and DQN in the Image Input case, between HCC and PPO and well as HCC and A2C in the Paddle&Ball Position Input case, and between HCC and A2C in the Ball Position Input case. The MCC group also outperform DQN, PPO, or A2C groups in the Image Input, Paddle&Ball Position Input, and Ball Position Input designs respectively.

Eventually, Subfigures S6.g, h, i, and j compare the frequency tables for the distributions of mean summed hits per minute amongst groups for the Image Input, Paddle&Ball Position Input, and Ball Position Input designs respectively. These tables were found to be not significantly different (Two-sample t-test).

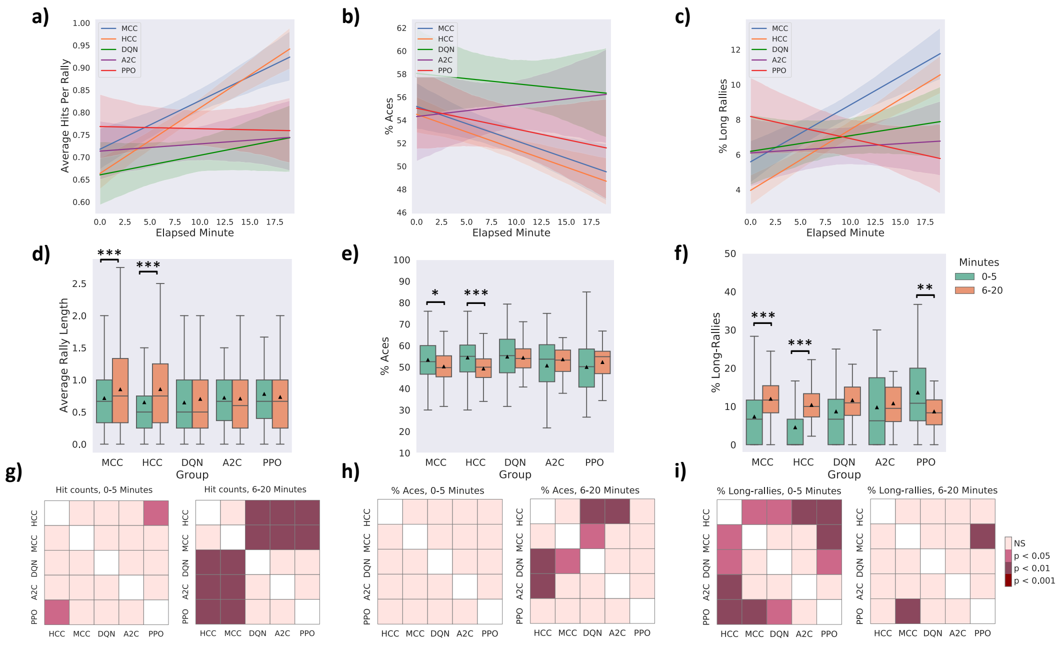


Figure S4: PADDLE&BALL POSITION INPUT to the deep RL algorithms: Average number of **a)** hits per rally, **b)** % of aces, and **c)** % of long rallies over 20 minutes real-time equivalent of training DQN, A2C, PPO, and MCC, HCC cultures. A regressor line on the mean values with a 95% confidence interval highlights the learning trends. The highest level of average hits per rally is achieved by the neuronal MCC and HCC cultures. Average % of aces is lowest for the neuronal cultures compared to all deep RL baseline methods. Average % of long rallies reaches its highest levels for MCC and HCC. Comparing to the same findings for the HCC and MCC groups, **d)** average rally length over time only showed a significant increase in the biological cultures between the two time intervals (One-way ANOVA test, p = 0.241, p = 0.756, and p = 0.315 for DQN, A2C, and PPO respectively). **e)** Average % of aces within groups and over time only showed a significant difference in the MCC and HCC groups. No significant change was detected within the DQN, A2C, or PPO groups (One-way ANOVA test, p = 0.858, p = 0.279, and p = 0.398 respectively). **f)** Average % of long-rallies (>3) performed in a session increased in the second time interval in all groups except the PPO group. This intra-group difference was significant for MCC, HCC, and PPO (One-way ANOVA test, p = 1.172e-7, p = 1.525e-24, p = 0.008 respectively). **g)** Pairwise Tukey's post hoc test shows that the HCC group significantly outperforms PPO in the first 5 minutes in terms of the hit counts or rally length. The biological cultures do significantly better compared to all deep RL opponents in the 15 minutes interval. **h)** Using pairwise Tukey's post hoc test, HCC group significantly outperforms A2C and DQN in the last 15 minutes interval with a lower average of % Aces. DQN is also outperformed by the MCC group in this time interval. **i)** Pairwise comparison using Tukey's test shows a significant difference in the percentage of long rallies between HCC and the rest of the groups in the first 5 minutes all outperforming the HCC. PPO also shows a significantly higher % of long rallies in the first time interval compared to MCC and DQN. However, this is later altered in the last 15 minutes with only MCC putperforming PPO significantly having an increased % of long rallies. Box plots show interquartile range, with bars demonstrating 1.5X interquartile range, the line marks the median and ▲ marks the mean. Error bands = 1 SE

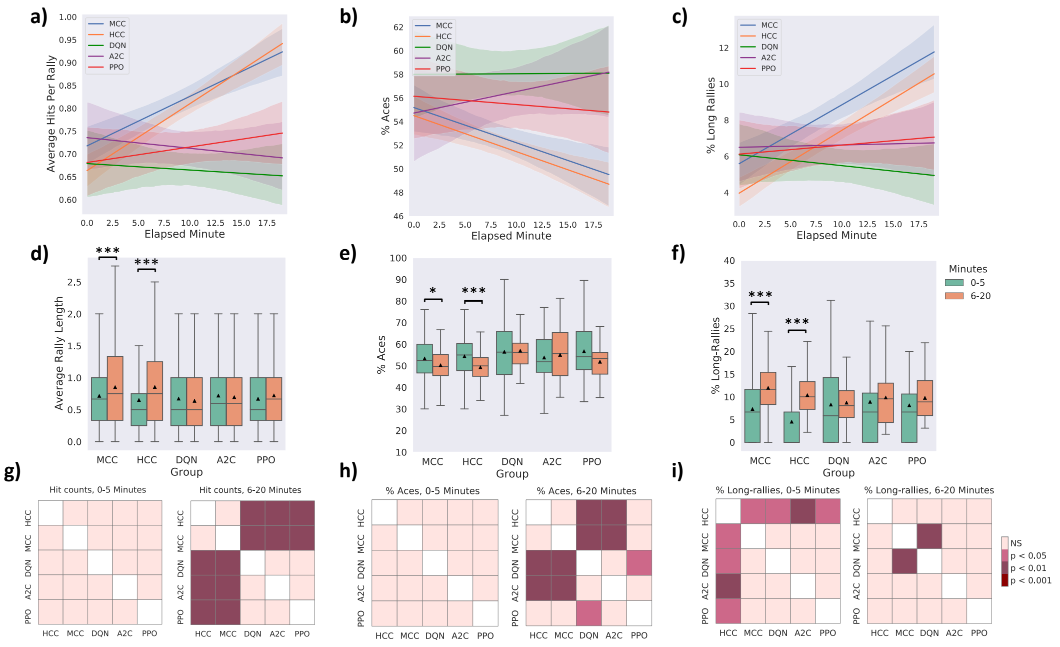


Figure S5: Ball Position Input to the deep RL algorithms: Average number of **a)** hits per rally, **b)** % of aces, and **c)** % of long rallies over 20 minutes real-time equivalent of training DQN, A2C, PPO, and MCC, HCC cultures. A regressor line on the mean values with a 95% confidence interval highlights the learning trends. The highest level of average hits per rally is achieved by the neuronal MCC and HCC cultures. Average % of aces is lowest for the neuronal cultures compared to all deep RL baseline methods. Average % of long rallies reaches its highest levels for MCC and HCC. Comparing to the same findings for the HCC and MCC groups, **d)** average rally length over time only showed a significant increase in the biological cultures between the two time intervals (One-way ANOVA test, p = 0.436, p = 0.612, and p = 0.240 for DQN, A2C, and PPO respectively). **e)** Average % of aces within groups and over time only showed a significant difference in the MCC and HCC groups. No significant change was detected within the DQN, A2C, or PPO groups (One-way ANOVA test, p = 0.858, p = 0.279, and p = 0.398 respectively). **f)** Average % of long-rallies (>3) performed in a session increased in the second time interval in all groups. This intra-group difference was only significant for MCC and HCC groups. **g)** Pairwise Tukey's post hoc test shows that biological cultures significantly outperform all deep RL groups in the last 15 minutes in terms of the hit counts or rally length. **h)** Using pairwise Tukey's post hoc test, HCC and MCC groups significantly outperform A2C and DQN in the last 15 minutes interval with a lower average of % Aces. DQN is also outperformed by the PPO group in this time interval. **i)** Pairwise comparison using Tukey's test shows a significant out-performance of all groups over HCC in the percentage of long rallies in the first 5 minutes. MCC also shows a significantly higher % of long rallies in the second time interval compared DQN. Box plots show interquartile range, with bars demonstrating 1.5X interquartile range, the line marks the median and ▲ marks the mean. Error bands = 1 SE

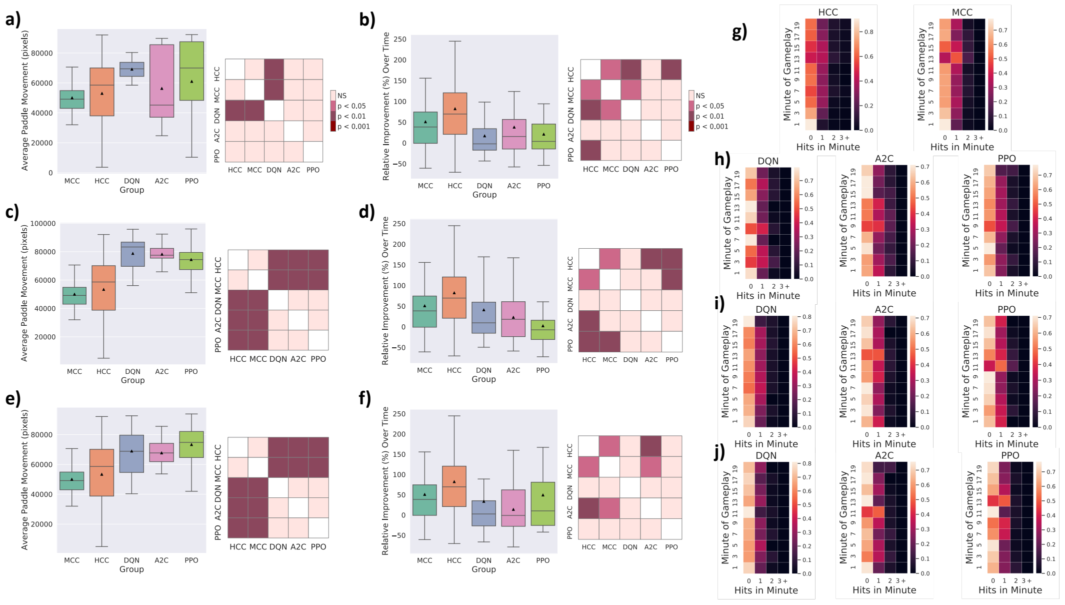


Figure S6: The average paddle movement in pixels in all the difference groups for the **a)** IMAGE INPUT, **c)** PADDLE&BALL POSITION INPUT, and **e)** BALL POSITION INPUT to the deep RL algorithms. Pairwise Tukey's post hoc test was conducted showing that DQN had a significantly higher average paddle movement compared to HCC and MCC in all scenarios. A2C and PPO also had a significantly higher average movement of the paddle in the case of Location Vector Input, and 2-dimensional Input. Relative improvement (%) in the average hit counts between the first 5 minutes and the last 15 minutes of all sessions in each separate group for the **b)** IMAGE INPUT, **d)** PADDLE&BALL POSITION INPUT, and **f)** BALL POSITION INPUT to the deep RL algorithms. The biological groups show higher improvements with HCC outperforming all. **b)** Using pairwise Tukey's post hoc test, the inter-group differences were significant for HCC and DQN, HCC and PPO, MCC and DQN, as well as MCC and HCC. **d)** HCC showed a significantly higher relative improvement compared to PPO and A2C while MCC also outperformed PPO in terms of relative improvement over time. **f)** Finally, MCC and HCC groups could significantly perform better than the A2C group with the 2-dimensional Input vector inputted to the deep RL algorithms. Distribution of frequency of mean summed hits per minute amongst groups for **g)** biological cultures and deep RL algorithms with **h)** IMAGE INPUT, **i)** PADDLE&BALL POSITION INPUT, and **j)** BALL POSITION INPUT.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Matteo Hessel et al. "Rainbow: Combining improvements in deep reinforcement learning". In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[3] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[4] Matej Moravčík et al. "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker". In: *Science* 356.6337 (2017), pp. 508–513.

[5] M Jaderberg et al. "Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. arXiv". In: *arXiv preprint arXiv:1807.01281* (2018).

[6] David Silver et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm". In: *arXiv preprint arXiv:1712.01815* (2017).

[7] David Silver et al. "Mastering the game of go without human knowledge". In: *nature* 550.7676 (2017), pp. 354–359.

[8] David Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (2018), pp. 1140–1144.

[9] Pedro A Tsividis et al. "Human learning in Atari". In: *2017 AAAI spring symposium series*. 2017.

[10] Gary Marcus. "Deep learning: A critical appraisal". In: *arXiv preprint arXiv:1801.00631* (2018).

[11] Elizabeth Gibney et al. "This AI researcher is trying to ward off a reproducibility crisis". In: *Nature* 577.7788 (2020), pp. 14–14.

[12] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. "Deep reinforcement learning: an overview". In: *Proceedings of SAI Intelligent Systems Conference*. Springer. 2016, pp. 426–440.

[13] Brett J Kagan et al. "In vitro neurons learn and exhibit sentience when embodied in a simulated game-world". In: *Neuron* (2022).

[14] Kai Arulkumaran et al. "Deep reinforcement learning: A brief survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[15] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[16] Richard Bellman and Robert Kalaba. "Dynamic programming and statistical communication theory". In: *Proceedings of the National Academy of Sciences* 43.8 (1957), pp. 749–751.