Multi-timescale biological learning algorithms train spiking neuronal network motor control

Daniel Hasegan[1], Matt Deible[2], Christopher Earl[3], David D'Onofrio[1], Hananel Hazan[4], Haroon Anwar[1], Samuel A Neymotin[1,5]

[1]Center for Biomedical Imaging and Neuromodulation, Nathan S. Kline Institute for Psychiatric Research Orangeburg, NY

[2]Dept. Computer Science, University of Pittsburgh. Pittsburgh, PA

[3]Dept. Computer Science, University of Massachusetts, Amherst. Amherst, MA

[4]Allen Discovery Center, Tufts University. Boston, MA

[5]Dept. Psychiatry, NYU Grossman School of Medicine. NY, NY

Abstract

Biological learning operates at multiple interlocking timescales, from long evolutionary stretches down to the relatively short timespan of an individual's life. While each process has been simulated individually as a learning algorithm in spiking neuronal networks (SNNs), the integration of the two is limited. In this study, we first train SNNs separately using individual model learning using spike-timing dependent reinforcement learning (STDP-RL) and evolutionary (EVOL) learning algorithms to solve the CartPole reinforcement learning (RL) control problem. We then develop an interleaved algorithm inspired by biological evolution that combines the EVOL and STDP-RL learning in sequence (EVOL+STDP-RL). This sequential algorithm implements the Baldwin Effect, by allowing learning during a model's lifetime to indirectly influence future model generations produced by EVOL. We evaluate performance of each algorithm after training and through the creation of sensory/motor action maps that delineate the network's transformation of sensory inputs into higher-order representations and motor decisions. Compared to STDP-RL and EVOL algorithms operating on their own, our EVOL+STDP-RL interleaved training paradigm enhanced performance robustness and shortened the required training to reach optimal performance. The different model strategies were revealed through analysis of sensory/motor mappings. Our modeling opens up new capabilities for SNNs in RL and could serve as a testbed for neurobiologists aiming to understand multi-timescale learning mechanisms and dynamics in neuronal circuits.

Keywords

Reinforcement learning, evolution, spiking neuronal networks, computer simulation, motor control, Baldwin Effect

Introduction

Reinforcement Learning (RL) problems offer an ideal framework for comparing learning strategies of an interactive, goal-seeking agent (Sutton and Barto 2018). Most often, the best learning strategy is evaluated based on the time efficiency and algorithm complexity. While there are many deep reinforcement learning algorithms for solving dynamical control problems (Volodymyr Mnih et al. 2015), biologically realistic network architectures and training strategies are not yet as efficient. In this work, using the CartPole RL problem, we compare biologically inspired learning algorithms based on the training efficiency and the resulting network dynamics.

Our Spiking Neural Networks (SNNs) simulate individual neurons as event-based dynamical units that mimic functions of their biological counterparts, like adaptation, bursting, and depolarization blockade (Neymotin et al. 2011). As SNNs are shown to be Turing-complete (Maass 1996b), and computationally more powerful than Artificial Neural Networks (ANNs) (Maass 1997, [a] 1996), efficient learning strategies are still actively investigated (Tavanaei et al. 2019). SNNs have been effective for pattern recognition problems (Gupta and Long 2007; Escobar et al. 2009; Kasabov et al. 2014; Tavanaei and Maida 2017; Mozafari et al. 2018) but are rarely used for solving reinforcement learning control problems. As spiking neurons operate in the time domain, we show that RL problems are suitable for evaluating training strategies and for providing insight into circuit dynamics.

Traditionally, when SNN models are trained to perform a behavior using biologically inspired learning mechanisms, algorithms used are variations on either Spike Timing Dependent Plasticity(STDP) (Tavanaei et al. 2019) or Evolutionary Strategies (Espinal et al. 2014). For learning behaviors from the reinforcement learning domain, STDP can be extended to use reward modulated plasticity (Anwar et al., n.d.; Patel et al. 2019; Hazan et al. 2018; Chadderdon et al. 2012; Neymotin et al. 2013), an algorithm denoted Spike-timing dependent reinforcement learning (STDP-RL). In this work, we introduce a new Evolutionary Strategy variation, adapted from non-spiking neural networks (Salimans et al. 2017), for solving RLnex problems. Furthermore, we combine the two training methods and compare the approaches.

STDP-RL trains SNNs by establishing associations between the neurons encoding the sensory environment and neurons producing an action or sequence of actions, such that appropriate actions are produced for specific sensory cues. The sensory-motor associations are established from reward-modulated synaptic weight changes that occur at each timestep of the simulation. Hence, STDP-RL trains at the individual level, as we consider each separate initialization of an SNN network a separate “individual”.

For training spiking neuronal networks based on population-level fitness metrics, we adapt Evolutionary Strategies (EVOL) (Salimans et al. 2017) that has been shown to be efficient in training Artificial Neural Networks on RL problems (Salimans et al. 2017; Chrabaszcz, Loshchilov, and Hutter 2018). Our EVOL algorithm treats the synaptic weight as an individual’s genome, then perturbs the genome using the mutation genetic operator to produce an offspring population. Each offspring is an individual SNN that interacts with the environment by receiving sensory stimuli and producing motor actions based on its internal firing patterns. Based on the offspring fitness, we combine the population genomes to yield the next generation of genomes. In the EVOL algorithm, SNN weights are not adapting throughout the interaction with the environment.

Previously, computer models of evolutionary and individual learning have been used to study the dynamics and mechanisms of both processes separately (Rumbell et al. 2016; Farries and Fairhall 2007). The combination of the two strategies, i.e., using evolution to select based on an individual's ability to learn (Baldwin Effect), has been proposed as a faster alternative to evolutionary strategies (Hinton and Nowlan 1986). The Baldwin Effect has been empirically validated against Lamarckian evolution (Suzuki and Arita 2004; Whitley, Gordon, and Mathias 1994) and was used in training Artificial Neural Networks (Whiteson 2006; Boers, Borst, and Sprinkhuizen-Kuyper 1995). However, the Baldwin Effect has not been investigated using SNNs in dynamic environments. Of note, the evolution of hyperparameters (for network and STDP) has been explored in training SNNs (Kozdon and Bentley 2018).

As a third training strategy, we investigate a novel algorithm that uses evolutionary strategies for selecting the best SNNs that underwent synaptic plasticity to solve the reinforcement learning challenge. We name the algorithm EVOL+STDP-RL because it employs EVOL for population-level training and STDP-RL as an individual-level SNN learning strategy. As EVOL+STDP-RL implements the Baldwin Effect, the weights learned with STDP-RL during the individual-level training are discarded and the original genomes are combined for further generations. This sequential algorithm, which parallels biological and individual learning, demonstrates superior performance, with enhanced robustness to initial conditions and improved stability.

To summarize, in this work, we investigate three algorithms for training SNNs on the CartPole RL problem: one individual-level training using STDP-RL and two novel population-level Evolutionary Strategies, EVOL and EVOL+STDP-RL. Our contributions are as follows: (1) we analyze the efficiency of biologically inspired training strategies for solving the CartPole RL problem; (2) we analyze the effect between individual-level and population-level learning on SNNs sensorimotor mappings and neuronal dynamics.

Materials and Methods

CartPole Game

To test different learning strategies, we chose the classic CartPole problem of balancing a vertical pole on a cart (Barto, Sutton, and Anderson 1983; Geva and Sitte 1993). All the simulations were run using the CartPole-v1 environment (**Fig. 1 left**) available in the OpenAI Gym platform (Brockman et al. 2016) (https://gym.openai.com). To keep the pole balanced (**Fig. 1 left**), a force to the left (-1) or the right (+1) must be applied at each time step of the simulation. Once the force is applied, a new game-state is generated, resulting from the interaction between the previous game-state and the applied action. The environment is fully described by four observations: cart position, cart horizontal velocity, pole angle with respect to a vertical axis, and angular velocity of the pole. For simplicity, we will now reference those four observations as position, velocity, angle, and angular velocity (see **Fig. 1 left**). The position is a relative distance from the center of the screen with positive (negative) values to the right (left). Similarly, the pole's angle represents positive values for angles to the right (clockwise). The velocity and angular velocity represents the rate of change of the position and the angle, respectively. The game is played in episodes, where each episode consists of multiple steps (left or right) taken to keep the pole balanced. The pole loses its balance when either the angle is greater than 15 degrees from vertical in either direction or the position is more than 2.4 units from the center in either direction. Each episode is allowed a maximum duration of 500 steps. An episode can be instantiated to different initial positions, which deterministically affects the trajectory through observational space.

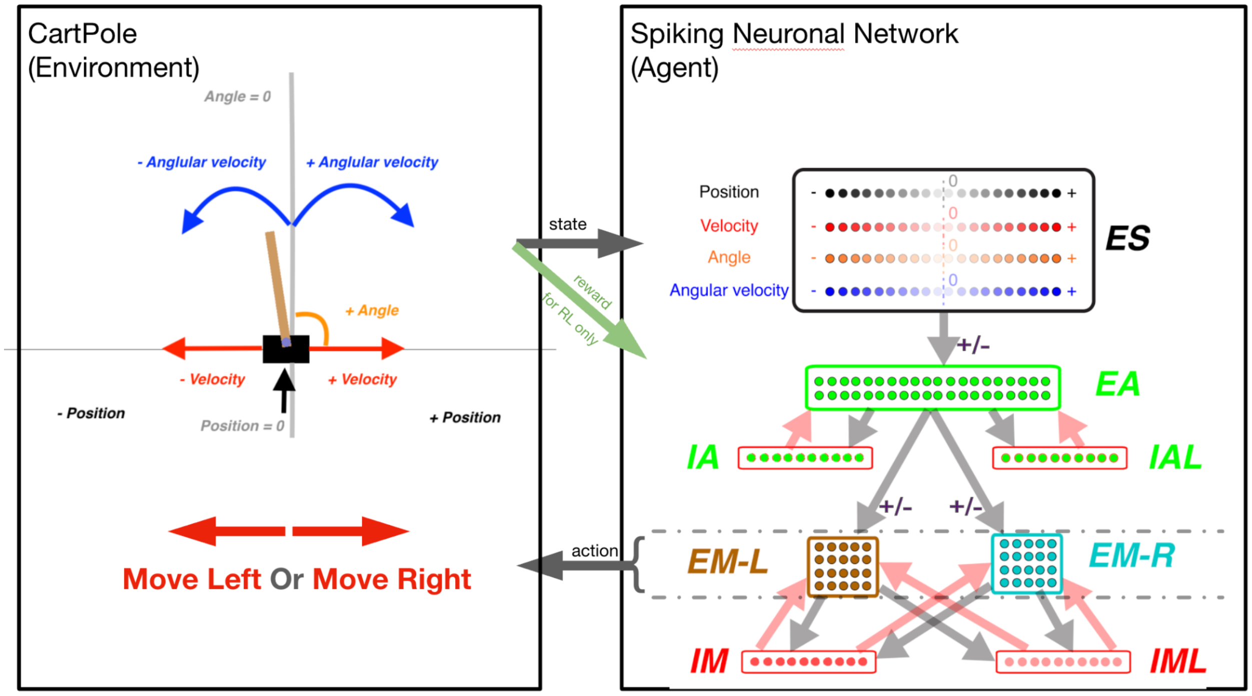


Figure 1: *The CartPole game environment (left) interfacing with the SNN model (right).* ***Left box****: At each game-step, a new game state is produced which is described by the game state variables (as labeled): Position, Velocity, Angle, and Angular Velocity.* ***Arrow "state"****: The values of these variables are used to activate a unique quadruple of neurons in the "ES" neuronal population (1 for each state variable).* ***Right box:*** *The neuronal network is represented as a diagram, as each dot represents one neuron, and arrows (light gray and light red) represent connectivity between populations of neurons. The light gray and light red arrows represent excitatory and inhibitory synapses between neuronal populations respectively. There are four excitatory neuronal populations: "ES" with 80 neurons (black-bordered box); "EA" with 40 neurons (green-bordered box); "EM-L" with 20 neurons (brown-bordered box); and "EM-R" with 20 neurons (cyan-bordered box). There are four inhibitory neuronal populations: "IA" with 10 neurons; "IAL" with 10 neurons; "IM" with 10 neurons; and "IML" with 10 neurons (each within a red-bordered box). Excitatory neuronal populations have outgoing excitatory connections while inhibitory neuronal populations have outgoing inhibitory connections. The connections marked with a "+/-" sign represent the connections that undergo synaptic plasticity: the connections between populations: "ES" to "EA", "EA" to "EM-L", and "EA" to "EM-R".* ***Arrow***

***"action"**: The activity within the "EM-L" and "EM-R" neuronal populations determines the action performed by the agent. Higher activity within either neuronal population will determine the agent to make a move to the left or to the right as exemplified by the large red arrows in the left box. **Green arrow "reward"**: for the Reinforcement Learning training strategy, the state of the environment is used to dictate the reward (or punishment) administered to the network for a correct (or incorrect) move.*

Simulations

We used the NEURON simulation environment (N. T. Carnevale and Hines 2006) with NetPyNE package (Salvador Dura-Bernal et al. 2019) for all modeling. NEURON simulates the firing of individual neurons based on the integration of input activation. Neurons are assembled into populations and into a connected network using the NetPyNE package that further coordinates the network simulation environment. The integration of the CartPole environment and the NetPyNE network was implemented in Python. The CartPole environment and the network simulation (the agent) is synchronized every time step T (50ms) in the following way:
- at the beginning of the time step, the environment is translated into neural activity in the input population (ES);
- for the duration of the time T, neurons are spiking based on induced activity;
- at the end of the time step (after 50ms), higher relative activation in the motor populations (EM-L, EM-R) determines the agent's action.

In the following sections, we will present the setup of the pre-plasticity simulation: the excitatory/inhibitory neurons, the network inputs, the movement generation, and the weights initialization.

Constructing a spiking neuronal network model to play CartPole

Our SNN model was adapted from one of our recent models (Anwar et al., n.d.). To allow the SNN model to capture the game-state space reliably, we included 80 neurons in the sensory area (ES) with four subpopulations (20 neurons each), each to independently encode position, velocity, angle, and angular velocity (**Fig. 1 right box, "ES"**). Each neuron was assigned to encode a different receptive field. Since the game's goal was to balance the pole, that would require more precision in encoding sensory information near balanced states, around the absolute value of 0. To capture higher sensory precision utilizing smaller receptive fields around the balanced state and less precision utilizing larger receptive fields at peripheries, we assigned receptive fields to each neuron based on percentiles of a Gaussian distribution with a peak value of 0 centered around the 11th neuron. As such, lower indices neurons (neurons 1-10) encode negative values of the state-variables in decreasing order. Similarly, higher indices neurons (neurons 11-20) encode positive values of the state-variables in increasing order. All four ES populations were assigned receptive fields using Gaussian distributions with each input state's expected mean and variance.

At each game-step, 4 ES neurons, one from each subpopulation, were activated, informing the SNN model about the game-state. To allow association between individual state-variable representing a game-state, we included 40 neurons in the association area "EA" (**Fig. 1 right box, "EA"**), which received inputs from the ES neuronal population. Each neuron in EA was connected to motor areas EM-L and EM-R generating Right- and Left-actions (**Fig. 1 "action" arrow**) by comparing the number of spikes in those populations (winner-takes-all). If both subpopulations have the same number of spikes, then a random move is performed.

To prevent hyperexcitability and depolarization-block (Anwar et al., n.d.; Chadderdon et al. 2012; Neymotin et al. 2013), we included inhibitory neuronal populations (10 IA, 10 IAL, 10 IM, and 10 IML).

Integrate-and-Fire neuron

Individual neurons were modeled as event-driven, rule-based dynamical units with many of the key features found in real neurons, including adaptation, bursting, depolarization blockade, and voltage-sensitive NMDA conductance (Lytton et al. 2008; Lytton and Stewart 2006; Neymotin et al. 2011; Lytton and Omurtag 2007). Event-driven processing provides a faster alternative to network integration: a presynaptic spike is an event that arrives after a delay at a postsynaptic neuron; this arrival is then a subsequent event that triggers further processing in the postsynaptic neurons. Neurons were parameterized (**Table 1**) as excitatory (E), fast-spiking inhibitory (I), and low threshold activated inhibitory (IL). Each neuron had a membrane voltage state variable (V_m), with a baseline value determined by a resting membrane potential parameter (V_{rest}). After synaptic input events, if V_m crossed the spiking threshold (V_{thresh}), the cell would fire an action potential and enter an absolute refractory period, lasting τ_{AR} ms. After an action potential, an after-hyperpolarization voltage state variable (V_{AHP}) was increased by a fixed amount ΔV_{AHP}, and then V_{AHP} was subtracted from V_m. Then V_{AHP} decayed exponentially (with the time constant τ_{AHP}) to 0. To simulate depolarization blockade, a neuron could not fire if V_m surpassed the blockade voltage (V_{block}). Relative refractory period was simulated after an action potential by increasing the firing threshold V_{thresh} by $W_{RR}(V_{block}-V_{thresh})$, where W_{RR} was a unitless weight parameter. V_{thresh} then decayed exponentially to its baseline value with a time constant τ_{RR}.

Table 1: Parameters of the neuron model for each type.

Cell type	V_{rest} (mV)	V_{thresh} (mV)	V_{block} (mV)	τ_{AR} (ms)	W_{RR}	τ_{RR} (ms)	ΔV_{AHP} (mV)	τ_{AHP} (ms)
Excitatory (E)	-65	-40	-25	5	0.75	8	1	400
Inhibitory (I)	-63	-40	-10	2.5	0.25	1.5	0.5	50
Low-threshold Inhibitory (IL)	-65	-47	-10	2.5	0.25	1.5	0.5	50

V_{rest}=resting membrane potential; V_{thresh}=spiking threshold, V_{block}=depolarization blockade voltage, τ_{AR}=absolute refractory time constant, W_{RR}=relative refractory weight, τ_{RR}=relative refractory time constant, ΔV_{AHP}=after-hyperpolarization increment in voltage, τ_{AHP}=after-hyperpolarization time constant.

Synaptic mechanisms

In addition to the intrinsic membrane voltage state variable, each cell had four additional voltage state variables V_{syn}, corresponding to the synaptic inputs. These represent AMPA (AM2), NMDA (NM2), and somatic and dendritic GABAA (GA and GA2) synapses. At the time of input events, synaptic weights were updated by step-wise changes in V_{syn}, which were then added to the cell's overall membrane voltage V_m. To allow for dependence on V_m, synaptic inputs changed V_{syn} by $dV=W_{syn}(1-V_m/E_{syn})$, where W_{syn} is the synaptic weight, and E_{syn} is the reversal potential relative to V_{rest}. The following values were used for the reversal potential E_{syn}: AMPA, 0 mV; NMDA, 0 mV; and GABAA, –80 mV. After synaptic input events, the synapse voltages V_{syn} decayed exponentially toward 0 with time constants τ_{syn}. The following values were used for τ_{syn}: AMPA, 20 ms; NMDA, 300 ms; somatic GABAA, 10 ms; and dendritic GABAA, 20 ms. The delays between inputs to dendritic synapses (dendritic GABAA) and their effects on somatic voltage were selected from a uniform distribution ranging between 3–12 ms, while the delays for somatic synapses (AMPA, NMDA, somatic GABAA) were selected from a uniform distribution ranging from 1.8–2.2 ms. Synaptic weights were fixed between a given set of populations except for those involved in learning (see **the "+/-" sign in Fig. 1 right box** and plasticity "on" in **Table 2**).

The Neuronal Weights

The neurons are organized into three overall layers: Sensory, Association, and Motor (**Fig. 1 right box**). The sensory layer consists of the excitatory sensory neuronal population (ES) which contains a total of 80 excitatory neurons. The association layer consists of 40 excitatory neurons in the excitatory association neuronal population (EA), 10 fast-spiking inhibitory neurons in the inhibitory association neuronal population (IA), and 10 low threshold activated inhibitory neurons in the "low" inhibitory association neuronal population (IAL). Similarly, the motor layer consists of 40 excitatory neurons in the excitatory motor neuronal population (EM), 10 fast-spiking inhibitory neurons in the inhibitory motor neuronal population (IM), and 10 low threshold activated inhibitory neurons in the "low" inhibitory association neuronal population (IML). Furthermore, the EM neuronal population is split into 20 neurons associated with left movements (EM-L) and 20 neurons associated with right movements (EM-R).

The weights between populations were adjusted to allow reliable transmission of spiking activity across different layers/areas of the SNN model. Each row in **Table 2** describes the synaptic connectivity between two different neuronal populations (no self connectivity) and follows the same diagram of neuronal connectivity described in **Figure 1**. Neurons belonging to the pre-synaptic population have axons that project to neurons in the post-synaptic population. Each neuron projects to a fixed number of post-synaptic neurons, a constant defined as the connection convergence (**Table 2**). The individual neuronal connections are picked randomly at the initialization process and different random seed values generate different connections, hence different networks. The excitatory neuron have both AMPA (AM2) and NMDA (NM2) synaptic connections, while the inhibitory neurons either have somatic GABAA synapses (GA) for fast-spiking inhibitory neurons or have the dendritic GABAA synapses (GA2) for low threshold activated inhibitory neurons. The synaptic weight W_{syn} for each neuronal connection was picked based on previous studies and fine-tuned on this network to start with a biologically reasonable spiking pattern (2-20 Hz).

Some of the synapse weights can be changed throughout the course of the training simulation as they undergo synaptic plasticity. For this work, we limited synaptic plasticity to AMPA synapses between excitatory populations (**Table 2: Plasticity column**). We found that this limitation is not hindering the network's ability to learn the dynamical behavior of the CartPole problem, and it rather simplifies the network analysis. To have a consistent examination between different training strategies, we used the same initialization methods and plastic synapses as defined in above and in **Table 2**.

Table 2: Initial connection weights

Pre-synaptic population	Post-synaptic population	Connection Convergence	Synapse Type	Synaptic Weight: W_{syn}	Plasticity (empty for Off)
ES	EA	25	AM2	10.0	On
			NM2	0.196	
EA	IA	15	AM2	5.85	
			NM2	0.0585	
EA	IAL	15	AM2	5.94	
			NM2	0.294	
EA	EM (EM-L + EM-R)	20	AM2	6.5	On
			NM2	0.1	
IA	EA	4	GA	18.0	
IA	IA	1	GA	4.5	
IA	IAL	2	GA	4.5	
IAL	EA	4	GA2	5.0	

IAL	IA	2	GA2	2.25	
IAL	IAL	1	GA2	5.5	
EM (EM-L + EM-R)	IM	16	AM2	5.85	
			NM2	0.0585	
EM (EM-L + EM-R)	IML	16	AM2	2.94	
			NM2	0.294	
IM	EM (EM-L + EM-R)	4	GA	18.0	
IM	IM	1	GA	4.5	
IM	IML	2	GA	4.5	
IML	EM	4	GA2	5.0	
IML	IM	2	GA2	2.25	
IML	IML	1	GA2	5.5	

Training strategies

Spike-timing dependent Reinforcement Learning (STDP-RL)

To train the neuronal networks, we used an existing STDP-RL mechanism, developed based on the distal reward learning paradigm proposed by Izhikevich (Izhikevich 2007), with variations used in spiking neuronal network models (Neymotin et al. 2013; Chadderdon et al. 2012; Salvador Dura-Bernal et al. 2016; Chadderdon and Sporns 2006; Anwar et al., n.d.). Our version of STDP-RL (**Fig. 2A**) uses a spike-time-dependent plasticity mechanism together with a reward or punishment signal for potentiation or depression of the targeted synapses. When a postsynaptic spike occurred within a few milliseconds of the presynaptic spike, the synaptic connection between this pair of neurons became eligible for STDP-RL and was tagged with an exponentially decaying eligibility trace. An exponentially decaying eligibility trace was included to assign temporally distal credits to the relevant synaptic connections. Later, when a reward or a punishment was delivered before the eligibility trace decayed to zero, the weight of the tagged synaptic connection was increased or decreased, depending on the 'critic' value and sign, i.e., increase for reward or decrease for punishment. Furthermore, the change in synaptic strength was proportional to the eligibility trace value at the time of the critic's delivery.

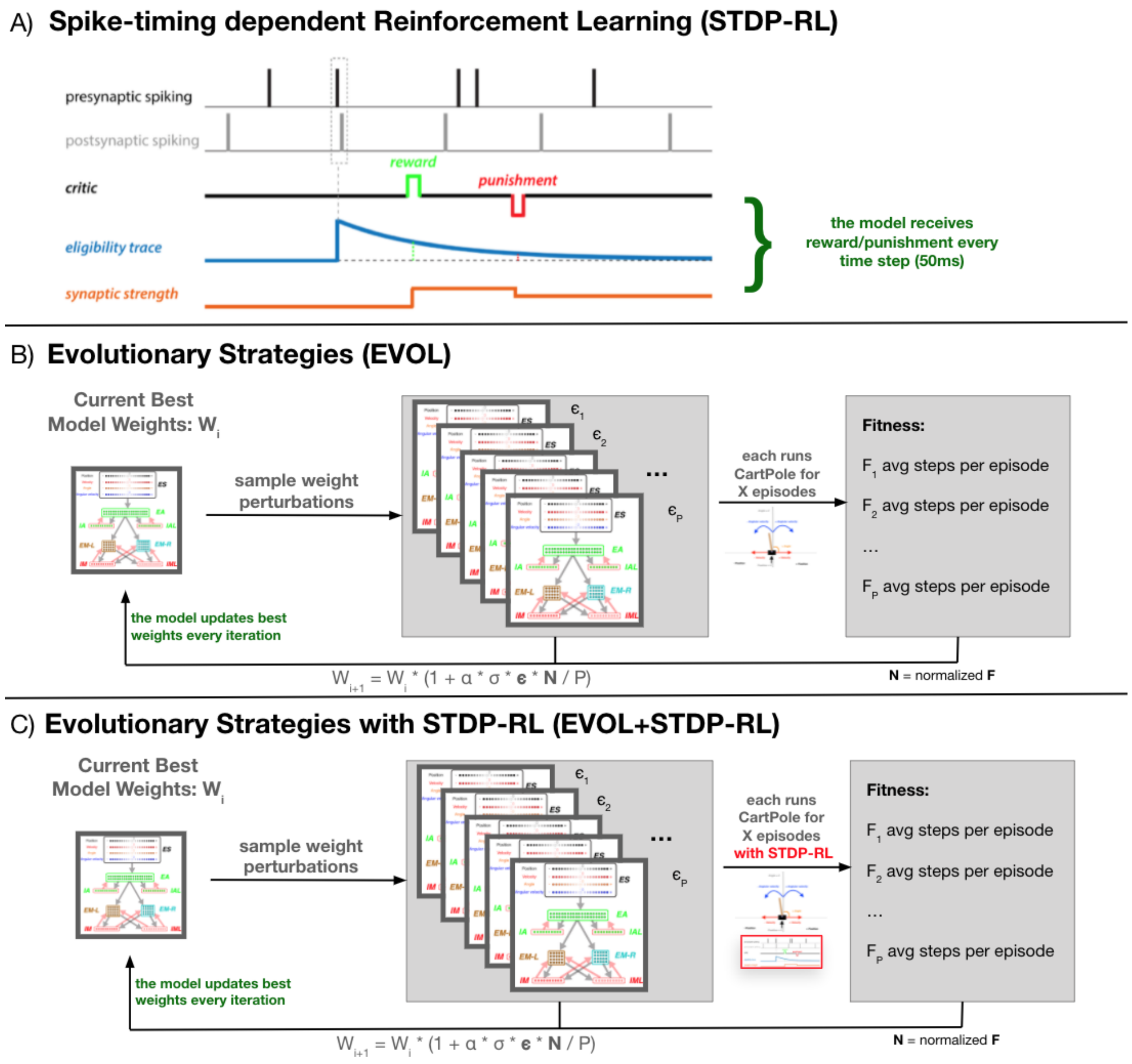

A) Spike-timing dependent Reinforcement Learning (STDP-RL)
presynaptic spiking
postsynaptic spiking
critic
reward
punishment
eligibility trace
synaptic strength
the model receives reward/punishment every time step (50ms)
B) Evolutionary Strategies (EVOL)
Current Best Model Weights: W_i
sample weight perturbations
ϵ_1
ϵ_2
ϵ_P
each runs CartPole for X episodes
Fitness:
F_1 avg steps per episode
F_2 avg steps per episode
...
F_P avg steps per episode
the model updates best weights every iteration
W_{i+1} = W_i * (1 + α * σ * ϵ * N / P)
N = normalized F
C) Evolutionary Strategies with STDP-RL (EVOL+STDP-RL)
Current Best Model Weights: W_i
sample weight perturbations
each runs CartPole for X episodes with STDP-RL
Fitness:
F_1 avg steps per episode
F_2 avg steps per episode
...
F_P avg steps per episode
the model updates best weights every iteration
W_{i+1} = W_i * (1 + α * σ * ϵ * N / P)
N = normalized F

Figure 2: *Training SNN using STDP-RL and EVOL strategies.* ***A)*** *In STDP-RL, when a postsynaptic neuron produced a spike within a short-time interval of a presynaptic spike, the synapse between the pair of neurons was tagged with a decaying eligibility trace. The tagged synapse was strengthened or weakened proportional to the value of eligibility trace for a reward or punishment, respectively.* ***B)*** *Schematic showing the steps of evolutionary strategies training algorithm (EVOL).* **C)** *Schematic showing the steps of evolutionary strategies combined with STDP-RL. Using the interleaved EVOL+STDP-RL algorithm, the fitness is evaluated after STDP-RL training instead of running the model with weights fixed.*

Traditionally, when using STDP-RL for learning behavior, all plastic synaptic connections in the neuronal network model are treated equally (*non-targeted STDP-RL*). This strategy considers that the underlying causality between pre and postsynaptic neurons and the associated reinforcement automatically changes only relevant synaptic connections. On top of the traditional STDP-RL approach, we used two recently developed versions of targeted reinforcement by selectively delivering reward and punishment to different subpopulations of the Motor population (EM) (Anwar et al., n.d.). In the first variation (*targeted RL main*), we delivered reward or punishment only to the neuronal subpopulation that generated the action (EM-LEFT or EM-RIGHT). In the second variation (*targeted RL both*), we additionally delivered opposite and attenuated reinforcement to the opposite-action neuronal subpopulation. Both targeted methods ensured that the learning was specific to the part of the circuit which generated the action. Moreover, we explored delivering (attenuated) '*critic*' values to the neuronal populations one synapse away from those directly generating motor actions (EA population). We used and evaluated all six STDP-RL mechanisms (three targeted RL versions X two nonMotor RL versions) for learning performance during hyperparameter search (see below for details). In all cases, although there is evidence of plasticity involving inhibitory interneurons in vivo(Anwar et al. 2017; Vogels et al. 2011), for the sake of simplicity STDP-RL was only applied between excitatory neurons.

Critic

For STDP-RL, the model relies on a critic to provide essential feedback to the model's actions (**Fig. 2A**). For CartPole, we picked a critic that responds positively to movements that bring the vertical pole closer to a balanced position. We computed a loss for each position, determined by the absolute values of the angle and the angular velocity input states. The critic's returned value will be the difference between the loss of the previous state and the loss of the current state. If the loss between the following states increases due to the agent's move, then the critic will return a negative value, corresponding to a punishment. Similarly, a decrease in loss will return a positive critic value, corresponding to a reward. If the agent could not decide on a motor move due to identical spiking activity in both subpopulations, then a constant negative punishment is returned. Additionally, as the critic is dominated by punishment at the beginning of training, to avoid the weights decreasing to zero, the model needs an associated boost in positive rewards ($\eta_{positivity}$). The critic is finally capped at a minimum and maximum value to keep rewards within the interval: $[-\ max\_reward,\ max\_reward]$.

$$loss(t) = \sqrt{ang(t)^2 + \eta_{angvel} * angvel(t)^2} \qquad (1)$$

$$reward(t) = \begin{cases} 0 & if\ loss(t-1) < 10^{-2} \\ \frac{-max\_reward}{\eta_{positivity}} & if\ no\_move(t) \\ \frac{max\_reward}{\eta_{positivity}} & if\ loss(t) < 10^{-2} \\ loss(t-1) - loss(t) & otherwise \end{cases} \qquad (2)$$

$$f(reinforcement) = \begin{cases} reinforcement * \eta_{positivity} & if\ reinforcement > 0 \\ reinforcement & otherwise \end{cases} \quad (3)$$

$$critic(t) = max(-\ max\_reward,\ min(\ f(reward(t))\ *\ gain,\ max\_reward)\ ) \quad (4)$$

Where:
- $ang(t)$and $angvel(t)$represent the input states at time step t;
- η_{angvel}represents the angular velocity bias, used to balance the dominance of each input state;
- $\eta_{positivity}$represents the positivity bias to reinforce rewarding behavior;
- $max_reward$is the maximum reward used, fixed at 1.0 throughout the whole experiment;
- $gain$represents a final multiplier that increases the absolute reward value.

The critic was implemented as a crude reinforcer of synaptic plasticity, working in conjunction with STDP events. As we found in the HyperParameter search (described below), most of the hyperparameters of the critic have little influence over the final performance of the model, and we believe that many different critic functions would have been suitable for our analysis. More importantly, for synaptic weight normalization, the critic values are further modulated by output gain and homeostatic gain control, as described below.

Hyperparameter Search

We first trained our SNN model using the STDP-RL parameters' values as used in earlier studies (S. Dura-Bernal et al. 2017) and found that the model did not perform very well since it could not learn to balance the CartPole for 50 steps per episode (averaged over 100 episodes). The low performance indicated that these parameters might not be optimal for training with STDP-RL. To find an optimal training strategy, we perturbed parameters of our STDP-RL training setup. We identified nine hyperparameters to optimize(**Supplementary Table 1**), related specifically to the STDP-RL mechanism that are somewhat independent of each other. Out of the nine: three parameters influence the timing and weight update of the STPD-modulated AMPA synapse(AMPA-RLwindhebb, AMPA-RLlenhebb, AMPA-RLhebbwt); four parameters determine the area of effect of STDP (Targeted_RL_Type, Non_Motor_RL, Targeted_RL_Opp_EM, Targeted_RL_Non_Motor); and two parameters influence the reinforcement derived from the critic(Critic Positivity Bias $\eta_{positivity}$, Critic Angv Bias η_{angvel}). To identify the best combination of the hyperparameters for training the network, we ran multiple random hyperparameter searches on those nine hyperparameters (**Supplementary Table 1**).

For the first hyperparameter search, we evaluated the nine parameters above by training networks with random choices of those hyperparameters. From the 7200 possible combinations, we sampled 50 combinations and trained randomly-initialized models using STDP-RL for 500 seconds in simulation time. We evaluated the performance of those models based on the average steps per episode over 100 episodes during training(**Supplementary Fig. 1**). Only the Targeted_RL_Type hyperparameter performance distributions showed a significant deviation from the mean performance (ANOVA p-value < 10^{-5}), but it failed the homogeneity of variance assumption test.

For the second step of the hyperparameter search, we continued training from the best four models from the previous step, and continued evaluating the nine parameters. For this step, we had 6912 combinations from which we sampled 54 combinations that we trained for 2000 seconds in simulation time. Similarly, we evaluated the performance of those models based on the average steps per episode over 100 episodes during training(**Supplementary Fig. 2**). The initialization model choice displayed a significant contribution to the final model performance (ANOVA p-value = 0.00012). Moreover, the hyperparameter defining the maximum time between pre- and postsynaptic spike (AMPA-RLwindhebb), also showed a minimal deviation from the mean performance (ANOVA p-value = 0.027).

For the third step of the hyperparameter search, we continued training from the best four models from the previous step, and continued evaluating the nine parameters. For this step, we had 1728 combinations from which we sampled 56 combinations that we trained for 2000 seconds in simulation time. Similarly, we evaluated the performance of those models based on the average steps per episode over 100 episodes during training(**Supplementary Fig. 3**). The initialization model choice displayed a significant contribution to the final model performance (ANOVA p-value = 0.00023). Moreover, the hyperparameter defining the choice of delivering reinforcement to non-motor populations (Non_Motor_RL), showed a minimal deviation from the mean performance (ANOVA p-value = 0.005).

Most of the hyperparameters had a minimal effect on the final training performance for the STDP-RL model as there is no significant difference between the performance of models with different hyperparameter values. Interestingly, the hyperparameter search revealed better preference when using the "*targeted RL both"* paradigm. These findings suggest that targeted plasticity of specific motor areas could enhance the learning ability of the model, consistent with earlier findings((Anwar et al., n.d.; Patel et al. 2019; Hazan et al. 2018; Chadderdon et al. 2012)).

Training Protocol for the STDP-RL model

Since we couldn't establish a best choice for each of the hyperparameters, we used the training protocol of the best model resulting from the third hyperparameter step. Hence, we trained our STDP-RL models with a sequence of the hyperparameter values for different durations (500s, 2000s, 2000s). Continuing training after the 4500s mark, we used the hyperparameters of the third configuration. The exact hyperparameter values used are highlighted in **bold** in **Supplementary Table 1**.

Evolutionary Strategies

The Evolutionary Strategies (EVOL) algorithm has been shown to be an effective gradient-free method for training ANNs in reinforcement learning control problems (Salimans et al. 2017). Here we adapt this learning technique to SNNs to solve the CartPole problem by procedurally adapting the plastic weights of the SNN. Our method progressively adapts only the weights and not the delays as it was used in previous Evolutionary Strategies for SNNs (Altamirano et al. 2015). It should be noted that in the ANN implementation of EVOL weights are allowed to be unrestricted in value, so additive weights were used. As SNNs don't have negative weights, we instead use a multiplicative noise, i.e. we increase or decrease the weights by a randomly selected percentage. In this way we are able to restrict the SNN weights to valid positive values while still effectively searching all possible parameterizations.

Formally our EVOL algorithm (**Fig. 2B**) consists of the following steps to change the weight for each synapse, performed in parallel for the whole model: (1) at iteration i, keep track of current best synapse weight: w_i; (2) sample a population(P) of weight perturbations $\epsilon_{1..P}$ from the normal distribution; (3) for each weight perturbation (ϵ_j), evaluate the whole network weights ($w_i * (1 + \sigma * \epsilon_j)$) on the CartPole environment for a fixed number of episodes (X); (4) measure the fitness (F_j) as the average count of steps achieved during the X episodes; (5) Normalize population fitness values (N_j) by subtracting the population mean fitness and dividing by the population mean standard deviation (6) modulate the weight perturbations based on the normalized fitness and derive a new best synapse weight:

$$w_{i_{+1}} = w_i * (1 + \alpha * \sigma * \epsilon * N / P) \qquad (5)$$

Where α is the learning rate, σ is the noise variance, and **ϵ** and **N** are the vector representations of the weight perturbations for each synapse and the normalized fitness, respectively. We only update the weights that undergo synaptic plasticity (**Table 2**). The weights are initialized (w_0) as the same initial weights we used for the STDP-RL model. In this case, STDP was fully deactivated and the EVOL training procedure updated the synaptic weights every iteration.

We trained using the EVOL algorithm multiple models for 1500 iterations and a population of P=10 with synapse weight perturbations of σ=0.1 variance. We used a learning rate of α=1.0. We used 1 and 5 episodes (value X above) during fitness evaluation. The model weights were initialized based on the best and worst models trained with STDP-RL.

Interleaved EVOL with STDP-RL

Our interleaved EVOL+STDP-RL algorithm was the same as the EVOL algorithm applied above, with one modification: rather than use the fitness from step (4), we ran an additional STDP-RL training for each member of the population to simulate individual learning. Afterward, the fitnesses from these models were re-assessed and used to update the best weights for the next iteration. Note that we did not use the post-STDP-RL learned weights since synaptic learning during a lifetime is not typically transferred to offspring. Instead, the pre-STDP-RL weights were used with the post-STDP-RL fitness to produce the next best weights. Due to the heavy computational load, we used 1 and 5 episodes of STDP-RL(value of X) to evaluate the fitness value. When using 10 episodes for fitness evaluation, performance was qualitatively similar and improved even more quickly (Results not shown).

Synaptic weight normalization

Training the model with STDP-RL, the synaptic weights tend to increase without bound, leading to epileptic activity (M. S. Rowan, Neymotin, and Lytton 2014). To avoid this behavior, we incorporated biologically-realistic normalization methods (Sanda, Skorheim, and Bazhenov 2017; M. Rowan and Neymotin 2013). Apart from the inhibition mechanisms described earlier, we used the following techniques: balancing of synaptic input, output balancing, homeostatic gain control.

To balance the combined synaptic input to each neuron, the total reception weight (defined as the sum of synaptic weights onto each postsynaptic neuron from multiple presynaptic neurons) is normalized to initial values every 25 time steps. As this procedure keeps neuronal inputs constant, it either decreases the weights of specific unused synapses or promotes beneficial synapses, creating synaptic competition.

To prevent synapses from overwhelming the network with ever-increasing rewards, each synapse's reinforcement is modulated by the change in the neuron's total transmission weight (defined as the sum of synaptic weights from a presynaptic neuron onto multiple postsynaptic neurons). Hence, a neuron with a high transmission weight compared to initialization will not increase the synapse strength as much from a rewarding event but will decrease severely from a punishment event. The modulation is capped at a factor of 0.1 minimum and 2.0 maximum.

Homeostatic gain control is a method to bring a neuronal population to a target spiking rate by changing the target transmission rate during output balancing. Each neuron firing rate is measured every 75 steps, over 500 steps. If the firing rate is different from the target firing rate, then the target transmission rate increases or decreases by 0.0001. This procedure has the effect of reducing high neuronal activity and promoting baseline activity.

Validation, Testing, and All-Inputs datasets

For testing the trained models, we used two separate datasets, the validation dataset for selecting the best model timepoint and the testing dataset for reporting the final model performance. For those two datasets, we set a seed value for the OpenAI gym environment in order to fix the episodes to consistently get the same episode initialization. This environment seed should not be confused with the model seed that we use to randomly initialize the neuronal connections. The episodes are fixed for our datasets when they have the same initial starting conditions for all four environment parameters (position, velocity, angle, angular velocity). While training the models, the episodes are not fixed and are randomly selected for each new training instantiation. To select the best saved timepoint of a trained model, we evaluate the model with weights different timepoints throughout training using a validation dataset with 100 fixed episodes. Once the best timepoint of a model is picked, we evaluate and report the performance on the testing dataset which contains another 100 fixed episodes. For analysis, we selected some of the fixed episodes for further evaluation.

As each input parameter space is discretized into activation of twenty individual neurons, the total possible combinations of inputs is 20^4. We tested the models on all the possible combinations, the All-Inputs dataset, by activating each input combination at a time and then allowing enough time for all spiking activity to subside before giving the next input combination.

Software

All modeling and analysis source code is available on github at the following repository location https://github.com/NathanKlineInstitute/netpyne-STDP .

Results

To test the effect of different training strategies on the Spiking Neuronal Network, we fixed the initialized synapticl connection weights and inter area connections (see Methods, "The Neuronal Weights") and evaluated the efficiency of training using STDP-RL, evolutionary strategies(EVOL), and a combined method that runs evolutionary strategies on the fitness derived while learning (EVOL+STDP-RL).

Training the SNN model to play CartPole using STDP-RL

To train the SNN using STDP-RL, we used the same training protocol as used in the hyperparameter search to find the best performing model. To assess the performance (which is determined by the duration for which the pole is kept balanced), we trained 20 randomly initialized models, with the same synaptic weights and connections, until the performance converged or started decreasing. Training all models for 25000 seconds in simulation time showed enhanced yet variable performance (**Fig3A**). When we evaluated the performance during training based on steps per episode, averaged over 100 episodes (**Fig 3A**),he maximum performances of the models range from 75 to 157, with a mean of 118 steps per episode (**Fig 3B**).

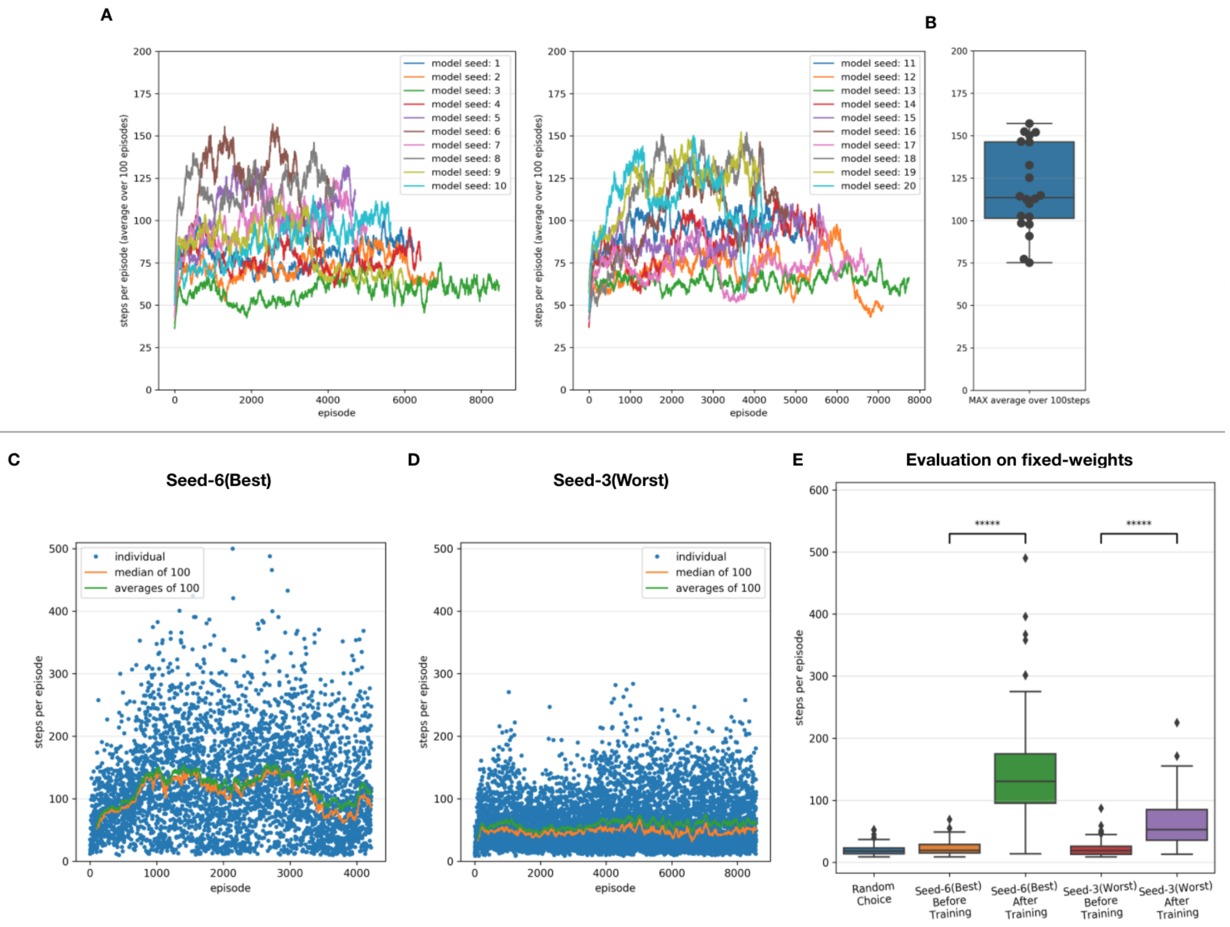


Figure 3: Performance of the SNN models when training with STDP-RL. (A) Training performance (defined as the length of time the model could keep the pole balanced) of the 20 randomly initialized models. All models trained for the same amount of simulated time. Out of

the tested models, the seed-3 model was the worst performer(green line of left plot), while the seed-6 model was the best performer(brown line of left plot). ***(B)*** *The box plot represents the distribution of maximum averages over 100 steps of the models. Each dot of the scatter plot represents the maximum average over 100 steps of one of the randomly initialized models.* ***(C,D)*** Detailed training performance of the seed-6 (best) and seed-3 (worst) models. Each blue dot represents the steps for that episode during training. The green and orange lines represent the averages and the median performances respectively. **(E)** The distribution of performance (steps per episode) calculated before and after training, on the testing dataset, using fixed weights of the seed-6 (best) and seed-3 (worst) models. The 5-star (*****) denotes $p < 10^{-10}$ when comparing the two distributions with a one-way ANOVA.

Out of the twenty models we trained using STDP-RL, we further evaluated two different models based on their performance during training; the best performing model (seed-6 in **Fig. 3C**) and the worst performing model (seed-3 model, **Fig. 3D**). The selected models were evaluated using the testing dataset consisting of 100 episodes, and further compared to the model before training and to a random choice null model (**Fig. 3E**). Both models improved their performance throughout STDP-RL training as we see an increase from 19 to 130 median steps per episode for the Seed-6(best) model, and from 19 to 53 median steps per episode for the Seed-3(worst) model. One-way ANOVA on the logarithm of the model performances before and after training resulted in $p < 10^{-10}$.

Training the SNN model to play CartPole using EVOL

While the STDP-RL training performance plateaued before reaching an average of 200 steps per episode, the EVOL training strategy was able to solve the game perfectly (an average of 500 steps per episode). To evaluate the EVOL strategy, we started training ten models with the same random initializations which we used for training STDP-RL (seed-1 to seed-10 in **Fig. 3B left**). All the models rapidly learned the task and achieved high performance (>400 steps per episode) in roughly 250-500 iterations (**Fig. 4A**). Though evolved variably, at some point all models reached peak average performance for 500 steps per episode. (**Fig. 4B**). During each iteration, EVOL generated *P* perturbations (each has a performance value) and running *E* episodes for each perturbation, for a total of *P* * *E* episodes. We display the performance minimum, average, and maximum for each iteration for the model with the Seed-3 random initialization (**Fig. 4C**).

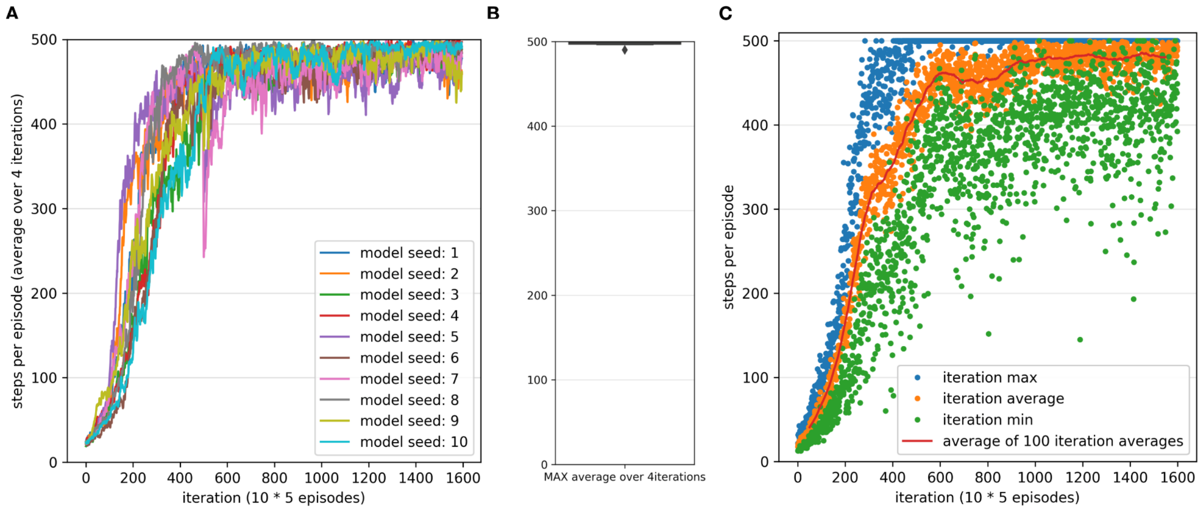


Figure 4: Performance of the SNN model when training with EVOL. (A) Training performance (defined as the length of time the model could keep the pole balanced) of 10 randomly initialized models. All models trained for the same number of iterations as opposed to the same amount of simulated time. ***(B)*** *The box plot represents the distribution of maximum averages over 100 steps of the models. Each dot of the scatter plot represents the maximum average over 100 steps of one of the randomly initialized models.* ***(C)*** *Detailed training performance for the seed-3 model. For each iteration step, we plot three points depending on the evaluated fitness while training: the maximum fitness (blue), the average fitness (orange), the minimum fitness (green) of each iteration. To better visualize the average fitness trend over training, we also plot an average over 100 iterations of the average fitness (red line).*

Compared with the STDP-RL training strategy that reached an average of 118 steps per episode (over 3931 episodes on average), the EVOL models used on average 145 iterations (for a total of 7250 training episodes) to reach the same performance level. While STDP-RL training was provided with frequent rewards from a hand-tuned critic, the EVOL training only used episodic fitness, a much sparser form of reinforcement. Although the STDP-RL models achieved a modest yet sustained performance, we found EVOL strategy to be better suited for solving the CartPole problem.

Comparing the performance of models trained using STDP-RL and EVOL

To demonstrate that the model learned the behavior and did not forget it, we earlier compared two different STDP-RL models (with different seeds) with the random action generating and untrained models (**Fig. 3E**) and found variable yet sustained learning. Less variable performance was observed for models trained using EVOL. This performance variability could be related to the initialization of synaptic weights or the presence/absence of synaptic connections between specific pairs of neurons. We first compared the performance of the EVOL model with the same seeds as used for STDP-RL model. On the testing dataset, both seed-6 and seed-3 EVOL models perfectly solved the CartPole problem with an average of around 499 steps per episode (**Table 3, Figure 5**). Another source of variability could be intrinsic noise in the model due to temporal and spatial crosstalk driven by imprecisely timed sensory inputs and heavily connected network areas or on the initial game-state (note that we reset the game-states at the end of each episode). To test the latter possibility, we handpicked nine unique initial game-states based on the performance of each model on the initial evaluation. For each initial game-state, we repeatedly simulated each trained model over 25 episodes (**Fig. 5**; note that at the end of each repeated episode, only the game-state was reset to the associated initial game-state, and the model was not reset). In **Fig 5**, Performance is shown only for STDP-RL models because EVOL models after training performed almost perfectly i.e 500 steps per episode. As indicated earlier in **Fig. 3E**, the STDP-RL model performance did not only depend on the model seed but also the game initial state. Altogether the EVOL model clearly showed perfect performance independent of model as well as game initialization, whereas STDP-RL models performance greatly depended on the seed as well as the game initial state.

Table 3: Performance comparison of training strategies on the testing dataset

Model	Average on Test Dataset	Median on Test Dataset	Trained Episodes	Iterations	Population
Seed-6 after initialization	23.38	19.5	0		
Seed-6 after STDP-RL training	144.67	130.5	4226		-
Seed-6 after EVOL training	499.42	500.0	80000	1600	10
Seed-3 after initialization	23.04	19.0	0		
Seed-3 after STDP-RL training	64.14	53.0	8577		-
Seed-3 after EVOL training	499.09	500.0	80000	1600	10

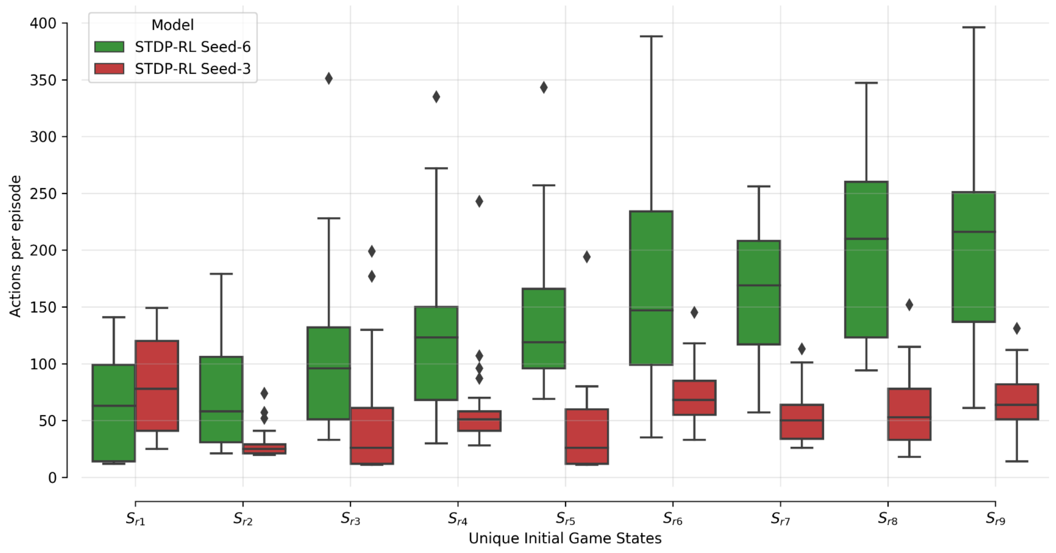


Figure 5: SNN STDP-RL trained models on different episodes. *Simulations for nine different episodes (Sr1-Sr9) were repeated 25 times each, using STDP-RL Seed-6 (best-performing) and*

Seed-3 (worst-performing) models. Note that each episode had a unique initial game-state which was identical across repeats of the same episode.

Training SNNs through combined EVOL and STDP-RL strategy

We then compared training progress between EVOL and our combined EVOL+STDP-RL algorithm, both evaluated using the same number of episodes (n=20). As shown in **Fig. 6**, the initial untrained models display low performance levels slightly above zero. Afterwards, both display rapid improvements in performance through learning, with a rapid rise for the EVOL+STDP-RL model around generation 100. As a result, while both models reach similar high performance levels (400-500), the EVOL+STDP-RL model reaches close to its peak performance earlier after ~200 generations, compared to EVOL which reaches similar levels after ~300 generations. Overall, this shows the two individual algorithms (EVOL and STDP-RL) complement each other together within the EVOL+STDP-RL hybrid algorithm, enabling more rapid performance gains, as a function of generations.

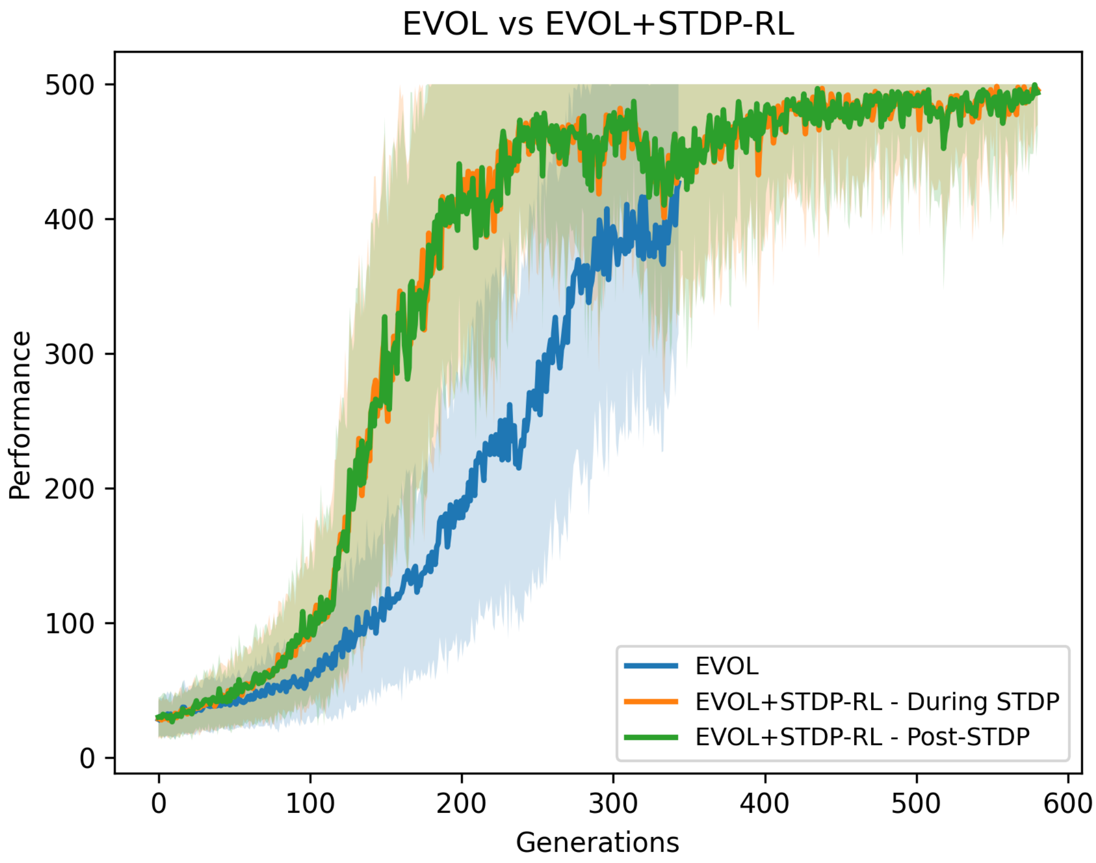


Figure 6: *SNN models learned to play CartPole, using EVOL, and EVOL+STDP-RL strategies starting from initial at-chance performance, and reaching optimal performance levels of 400-500 after training.* **(***blue trace shows the time course of performance using the standalone EVOL strategy, orange trace shows the time course of the SNN performance during the STDP-RL learning phase, green trace shows the performance of the SNN evaluated after further STDP-RL has been disabled; shaded lines show standard deviation of performance evaluations (n=20)).*

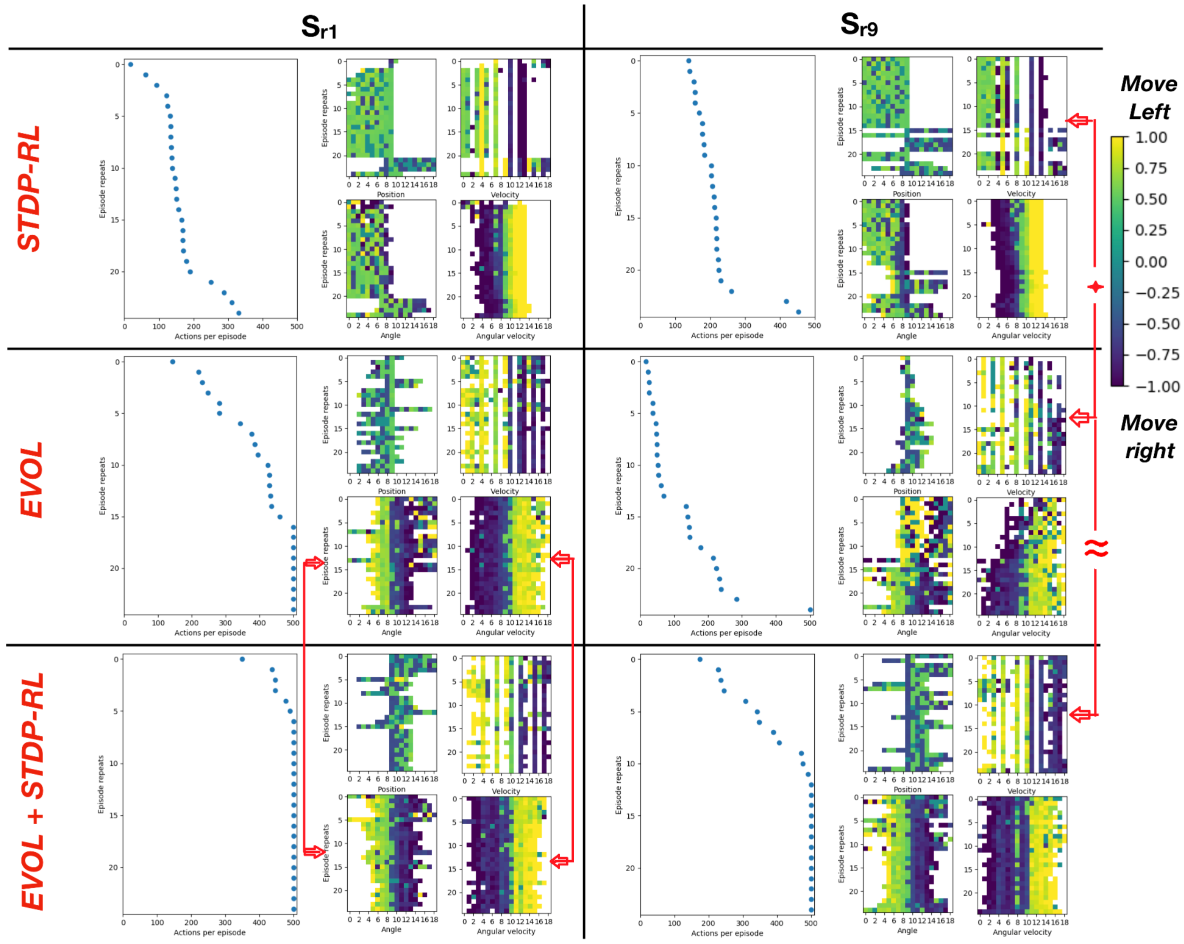


Figure 7: Learning produces specific preferences for right vs left actions, depending on sensory inputs. *These sensory-motor activation maps show the sensory-input parameters parsed during repeated episodes of Sr1 and Sr9 together with the probability of dominant action executed for each activated sensory-input neuron. Each episode of Sr1 and Sr9 was repeated 25 times, and the models' performances (shown on the left side of corresponding four panels) were evaluated using STDP-RL, EVOL, and EVOL+STDP-RL models. Red arrows/lines highlight similarities between specific sensory-motor activation maps across algorithms.*

Comparing the learned SNN circuit activations and dynamics

Next, we compared the sensory-motor activation maps of all three models with initial game-states Sr1 and Sr9 (**Fig. 7**). For each repetition of the episode, we counted the number of Left- and Right-actions generated for each activated sensory neuron and marked the respective sensory neuron with the probability of the produced action. For clarity, we marked the probability of left (right) actions with positive (negative) values. When a sensory neuron was not evoked during a repeat, no action was associated with that sensory neuron and therefore was left unmarked in the heatmap. We sorted the y-axis of the heatmaps representing the repeats of the episode in ascending order of performance to see if any specific sensory-motor mapping resulted in better performance or vice versa. The sensory-motor activation maps for STDP-RL and EVOL+STDP-RL were more similar for Sr1 and Sr9 as well as across repeats within each condition (seen as similar horizontal color patterns from top to bottom of each colormap). Individual features with similar action-space are marked in **Fig. 7** for comparison. Comparing sensory-motor heatmaps of the EVOL model for Sr1 and Sr9 in terms of Angle and Angular velocity, we found that the heatmaps for Sr9 were substantially more fragmented and sometimes overlapped in activations than those for Sr1, which indicated that either the model was undecided or was making inconsistent moves for the same inputs. We also noted that in the STDP-RL model, the cart moved through a full range of positions in both directions (left or right from the center position - 0), indicating that most of the time, the cart ran out of the game frame. With the EVOL and EVOL+STDP-RL based model, the cart remained around the center of the frame. Interestingly, the STDP-RL model showed a preference for Left-actions, whereas the EVOL and EVOL+STDP-RL models showed a more balanced preference for both actions. Angle and Angular Velocity sensory-motor heatmaps for Sr1 and Sr9 were largely conserved across the EVOL and EVOL+STDP-RL algorithms, though with less similarity for Sr9.

Despite the similar best performance of the models, the sensory-motor activation maps shaped by those strategies showed some notable differences (**Fig. 7**). Note that the range of activated sensory neurons was different across episodes when contrasted by different models, because of the closed-loop setup of the simulations. This analysis might not reveal the properties of the circuit to the full extent and might be biased by the activated receptive fields across the repeats. To further uncover the response properties of the network model, we simulated the already trained models by sequentially activating 20^4 unique quadruplets (20 neurons to encode each of the 4 sensory parameters) without interacting with the game and recorded the responses of all neurons in the SNN model. Each heatmap in **Fig. 8** shows the number of times neurons in the SNN model were active when a sensory-input neuron was stimulated (each sensory neuron was stimulated 8000 times for all combinations of the other 3 sensory inputs).

To highlight the evolved characteristics of the trained model using different learning strategies, we compared the input-response properties from before and after training. Most of the association neurons (EA neurons marked with ID 80-119) were sparsely active before training (**Fig. 8A**) without showing any distinct feature at the population level.

After STDP-RL based training, the overall activity of EA neurons increased with some EA neurons robustly responsive to specific inputs representing velocity (see Activated ES Neuron IDs 20-39; **Fig. 8B**) and angular velocity (see Activated ES Neuron IDs 60-79; **Fig. 8B**) suggesting robust learning for those parameters. For the velocity, the responsiveness was sparse and scattered but for the angular velocity, we found two sets of angular velocity inputs, one set for which many of the EA neurons responded strongly and the other set for which many of the EA neurons responded weaker than before learning (see high-value pixels clustered in the center of the band for ES Neuron IDs 60-79 and the surrounding areas in **Fig. 8B**). The responsiveness of EA neurons in the EVOL and EVOL+STDP-RL models (**Fig. 8C-D**) was sparse and scattered, which might indicate preference of neurons for multiple parameter sets instead of individual sensory features. Other excitatory neurons (EM-L: Neuron IDs 140-159 and EM-R: Neuron IDs 160-179) in each model showed similar characteristics as its EA neurons, except that well-defined vertical lines for some neurons in EVOL and EVOL+STDP-RL models (**Fig. 8C-D**) indicates responsiveness to broader parameter ranges. Alternatively, one might think this is attributed to the non-selective responsiveness of neurons to individual sensory-input features. However, that is not the case, as we observed more fragmented vertical lines when the input-response properties were characterized using pairwise input parameters (**Supplementary Fig. 3**). Interestingly, the EVOL+STDP-RL model showed sparser activity than the EVOL model (**Fig. 8C-D**), with significantly improved performance (**Fig. 4A**).

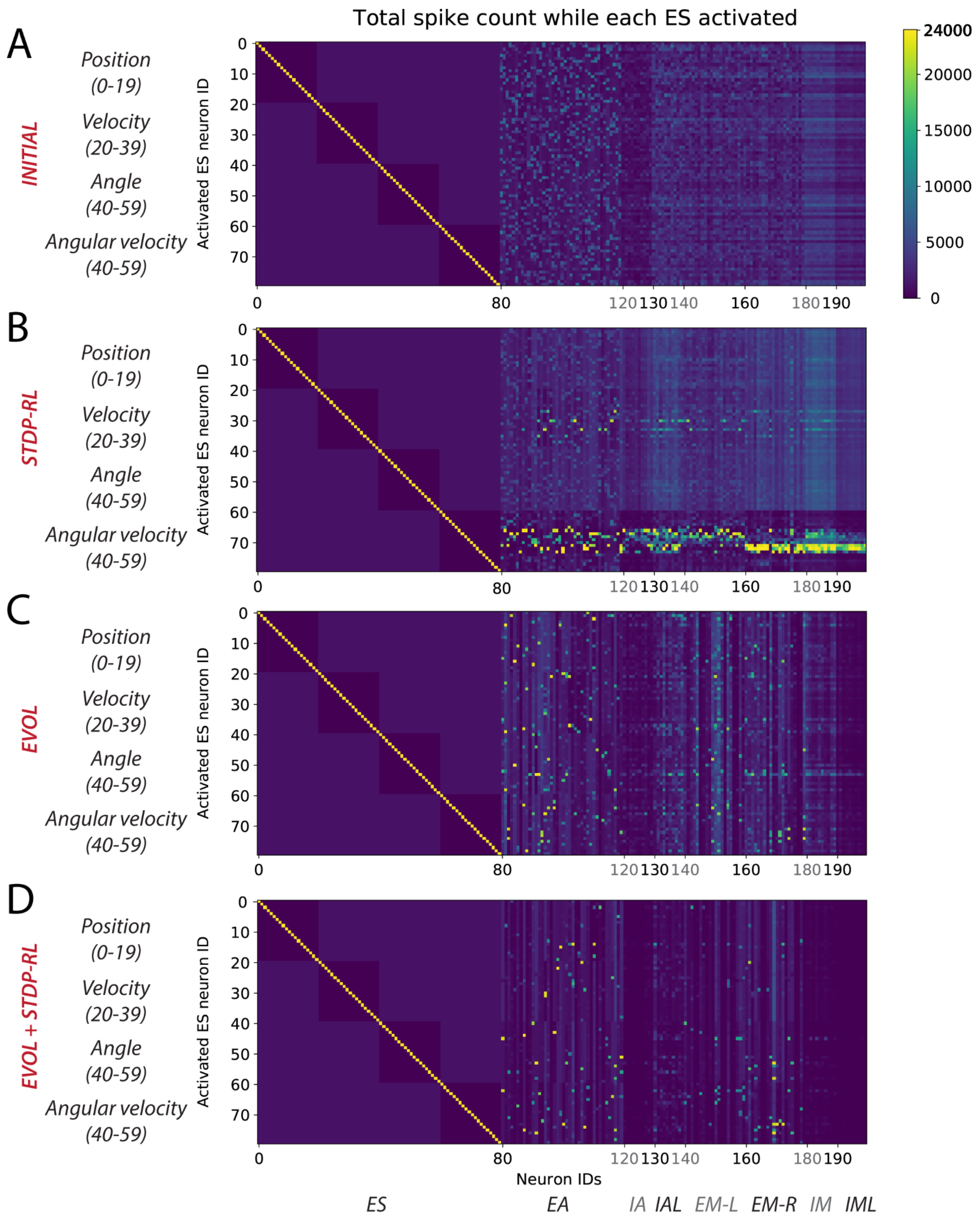


Figure 8. Learning induces changes in network responses to specific sensory inputs, with distinct representations produced by the different training algorithms. *The plot represents the number of spikes generated by each neuron in the model in response to stimulating sensory input neurons in ES. Each sensory input neuron was stimulated 8000 times*

with combinations of 3 other sensory input neurons, and each stimulation generated 3 spikes in ES neurons. The neuronal responses of the model before training (A), after training using STDP-RL (B), EVOL (C), and EVOL+STDP-RL(D).

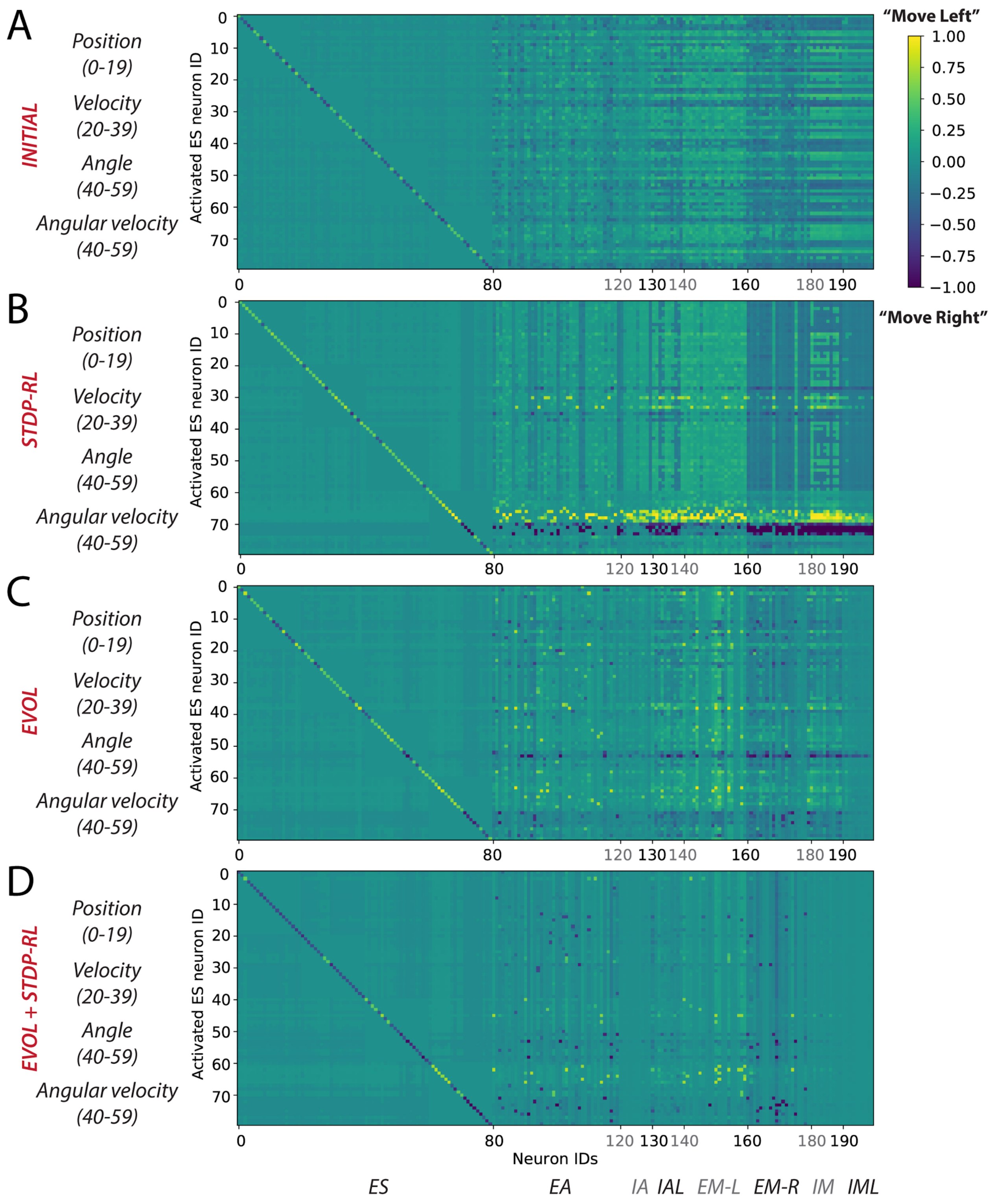


Figure 9: Sensory-Motor mappings show differential participation of neurons in the network in action generation. *In response to a particular sensory input, each neuron was considered to contribute to Move-left or Move-right actions and then labeled with the frequency*

of the number of times it was activated relative to the frequency of those actions (with positive sign indicating activations relative to number of moves left, and negative sign indicating activations relative to moves right). ***A)*** *INITIAL,* ***B)*** *STDP-RL,* ***C)*** *EVOL, and* ***D)*** *EVOL+STDP-RL. Note that propensity towards an action is defined as the number of times a neuron is active during a particular action choice. For example, if out of 8000 sensory inputs, 6000 times the action was Move left and 2000 times the action was Move Right, for each of those sets of conditions we compute how many times each neuron was active. If the neuron was active 3000 out of the 6000 moves left, then the participation value is 0.5. If the same neuron gets activated 2000 times to move right, then its participation value is -1 (negative sign is for move right, positive for move left).*

Comparing the emergent Input-Output mappings after training with STDP-RL, EVOL, and EVOL+STDP-RL strategies.

Activation of each sensory neuron led to the activation of many neurons at multiple postsynaptic levels, pushing the model towards making a certain decision (Move Left or Right). Since the decisions were always based on the population level activity of motor areas, it was nontrivial to establish causality. Instead, we compared the representations of different sensory-motor mappings throughout the circuit by computing the probability of each motor action (Left, Right, and Stay) over multiple instances when a neuron is active in response to each sensory input. In the colormaps (**Fig. 9**), we only included the most frequent action associated with the input and that particular neuron in the circuit. Before training (**Fig. 9A**), all the neurons in the circuit were minimally biased towards each action as all probabilities shown in the colormap are 0 (stay), or marginally greater than 0.5 (for left action) or smaller than -0.5 (for right action). After training (**Fig. 9B-D**), the decisiveness of some neurons improved towards a single action as the range of colors in the map expanded towards +1 (for left action) and -1(for right action). As we saw differences in the activations of neurons earlier in **Fig. 8** for STDP-RL and EVOL models, we observed similar trends in these sensory-motor mappings (**Fig. 9B-D**). In the STDP-RL model (**Fig. 9B**), most of the excitatory motor neurons in the circuit developed a non-overlapping association between positive angular velocity (ES neuron IDs 70-79) and move-right action, and negative angular velocity (ES neuron IDs 60-69) and move-left action. i.e., a subset of neurons in EM-L was robustly activated only when the game-state had a negative angular velocity, and a subset of neurons in EM-R was robustly activated only when the game-state had a positive angular velocity. We observed similar neurons in EVOL and EVOL+STDP-RL models (**Fig. 9C-D**), but those were only a few with less robust activation. In contrast to the EVOL model, where some EM-L and EM-R neurons dominantly participated in the generation of opposite action (EM-L for positive angular velocity and EM-R for negative angular velocity), the EVOL+STDP-RL model did not show any subpopulation with such characteristics (**Fig. 9C-D**). In addition to angular velocity, we found that some scattered motor neurons in the STDP-RL model also formed associations with some velocity values of the game-state (**Fig. 9B**). For most of the other game-state parameters, neurons were marginally biased towards one or the other action. In the EVOL and EVOL+STDP-RL models, we found a few more scattered sensory-input associated neurons (**Fig. 9C-D**).

Plotting the activity levels of neurons during specific actions in response to each pair of sensory-inputs could diffuse the learned sensory-motor associations, especially since the model learned not about individual sensory parameters but a full game-state consisting of four sensory parameters at each game-step. Therefore, we next computed sensory-motor maps for pairs of sensory input parameters (**Fig. 10**). Before training, for only a few pairs of sensory input parameters, the model had weak preference either to move left, right, or stay (**Fig. 10A**). After training using STDP-RL, the model developed modest action preference for a single value of position (see the horizontal yellow line at *Position: "10"* in **Fig. 10B**) and some values of velocity (see horizontal yellow lines at *Velocity*: *"10"* and *"13"* for ***"Move Left"*** and *"7"*, *"15"* and *"17"* for ***"Move Right"*** heatmaps in **Fig. 10B**) but very strong action preference for broader bands of positive and negative values of *Angular velocity* around *"10"* (indicating 0 angular velocity; **Fig. 10B**). Surprisingly, we did not find any learned action preference for any value of the angle in the STDP-RL model. In contrast, we found quite a few input-output associations (*Position*: *"2"*, *"14"* and *"18"* for ***"Move Left"*** and *"11"* for ***"Move Right"*** heatmaps; *Velocity*: *"18"* for ***"Move Left"*** and *"4"* for ***"Move Right"*** heatmaps; *Angle*: *"4"* and *"8"* for ***"Move Left"*** and *"13"* for ***"Move Right"*** heatmaps; *Angular velocity*: *"3"*, *"4"* for ***"Move Left"*** and *"11-14"* and *"18"* for **"Move Right"** heatmaps in **Fig. 10C**) in the EVOL model, indicating distributed learned associations between actions and some values of all sensory input parameters. Surprisingly, much stronger associations were found for the input parameter values of angular velocity in the EVOL+STDP-RL model (**Fig. 10D**), somewhat similar to the STDP-RL model (**Fig. 10B**), despite very different network dynamics revealed by activation maps (**Fig. 8**). In addition to a few associations for *Position* and *Velocity*, broader associations emerged for the parameter *Angle* as indicated by wide yellow bands in **Fig. 10D**.

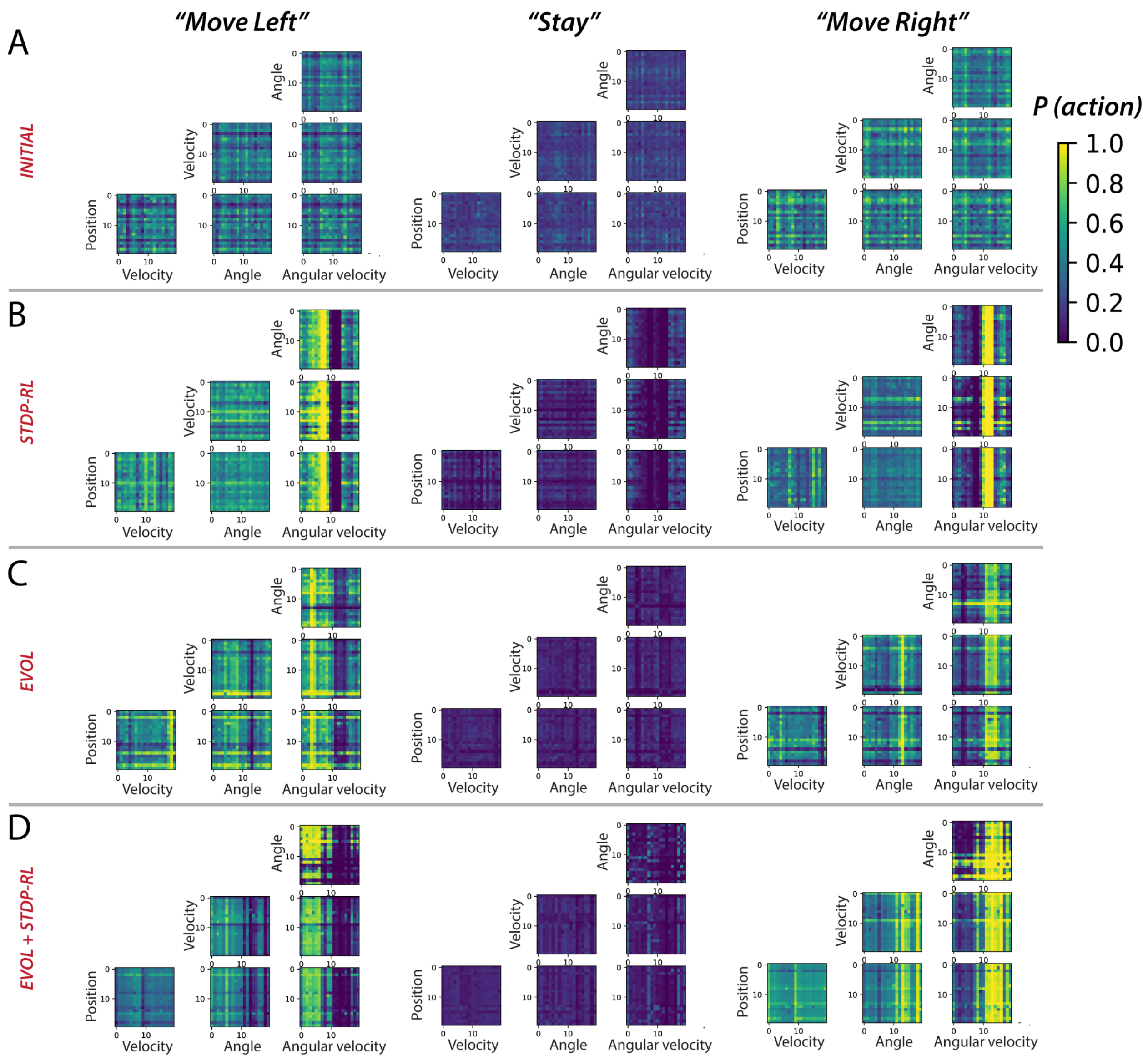


Figure 10: Sensory-Motor mappings show different behaviorally-relevant input-action associations learned using STDP-RL, EVOL, and EVOL+STDP-RL strategies. *Probability of "Move Left" (left panel), "Stay" and "Move Right" for all pairs of sensory input parameters for the model before training* ***(A)****, after training using STDP-RL* ***(B)****, after training using EVOL* ***(C)****, and after training using EVOL+STDP-RL* ***(D)****.*

To summarize sensory-motor mappings, we computed the total number of left-, right- and stay-actions for each sensory receptive field (**Fig. 11**), showing that the changes in synaptic weights in the STDP-RL model primarily encoded the associations between the actions and the angular velocity, followed by some weaker associations of actions with velocity and position. In contrast, relatively weaker associations were found between the actions and sparsely distributed receptive fields related to the game-state parameters in the EVOL and EVOL+STDP-RL models. However, even in the EVOL, EVOL+STDP-RL models, there was still a clear association between actions and Angular Velocity, and between actions and Angle.

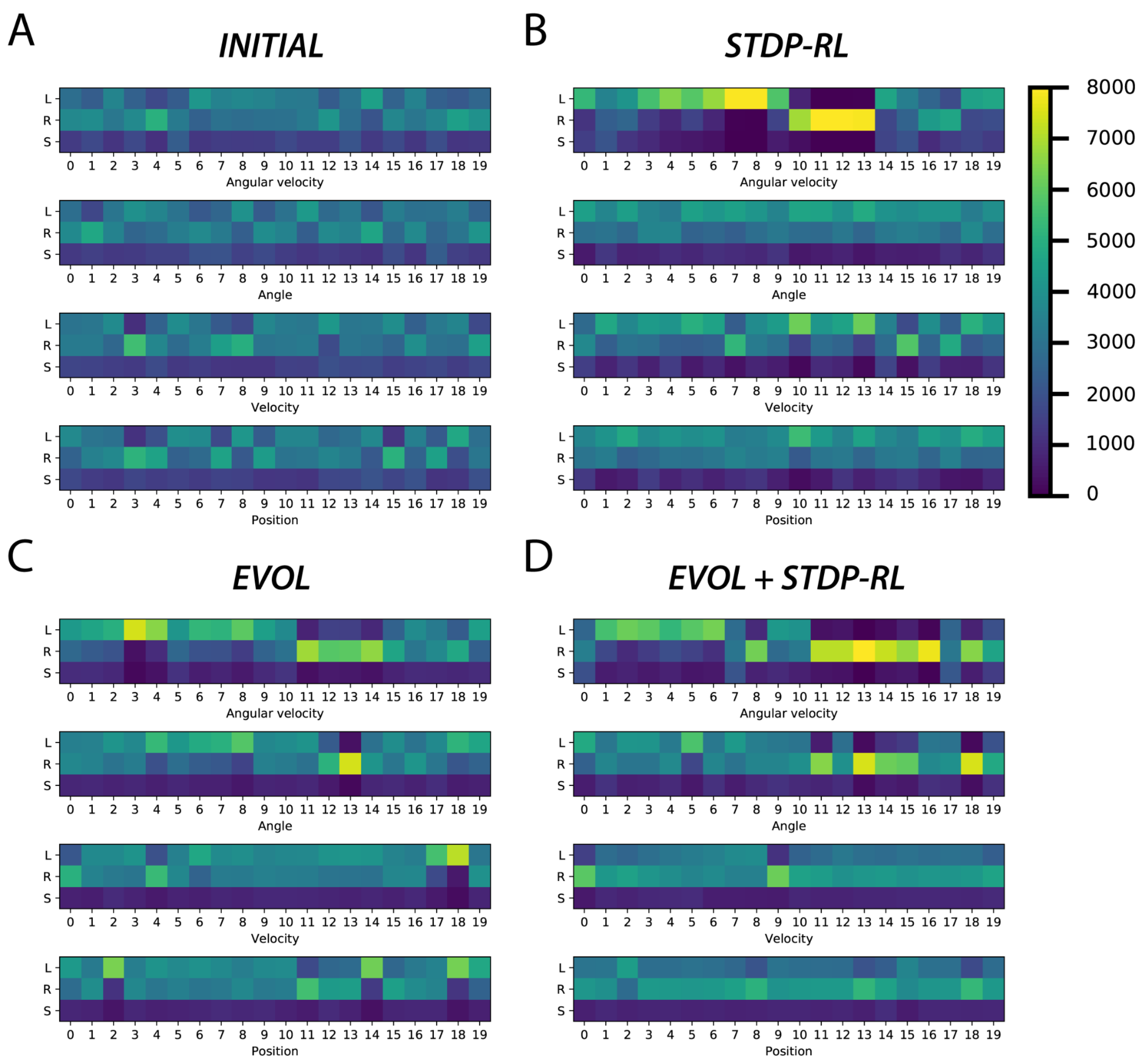


Figure 11: *Occurrence of Left-, Right- and Stay-actions for all receptive fields associated with the position, velocity, angle, and angular velocity. Heatmaps are shown to highlight the differences between the INITIAL model* ***(A)*** *before training and models trained using STDP-RL* ***(B)****, EVOL* ***(C),*** *and EVOL+STDP-RL* ***(D)*** *strategies.*

Discussion

In this work, we developed and trained an SNN using biologically inspired STDP-RL and evolutionary search (EVOL) based algorithms. One of our goals was to investigate biologically-plausible learning algorithms that operate at different timescales, and determine their strengths and weaknesses, to enable offering insights into biological processes. Comparing the performance of our SNN model trained using STDP-RL, EVOL, and EVOL+STDP-RL algorithms (**Fig. 3E, 5, 6 and Table 3**), we demonstrated that all of our strategies could be used for training our SNNs to play CartPole. Our hybrid EVOL+STDP-RL algorithm also converged to optimal performance levels at earlier generations compared to the standalone EVOL algorithm.

There were also noticeable differences in resulting neuronal dynamics and behavior (**Fig. 8-11**). EVOL and EVOL+STDP-RL strategies showed excellent performance training our SNNs to play CartPole (**Fig. 4 and 6**), however, EVOL+STDP-RL showed faster training compared to EVOL alone. When using the EVOL strategy, the algorithm tries to improve the network performance by adjusting the weights that lead to maximum performance. However, when using the EVOL+STDP-RL strategy, it seems that the algorithm maximizes the performance of the STDP change and not only the weights of the network. In a way, EVOL bootstraps the STDP changes and maximizes the overall performance by improving the STDP training.

It is interesting to see how training our models using different strategies produced similar sensory-motor associations without explicitly training the models to execute those specific motor actions for a given game-state. Regardless of the training algorithm, all models learned to associate actions dominantly to specific values of *Angular velocity* (**Fig. 8-11**). Analyzing the single input-action mappings (**Fig. 9**), we did not find clearly strong or broad associations between the Position, Angle, and Velocity with actions, which appeared in analyzing the pairwise input-action mappings (**Fig. 10**). It is possible that the associations between sensory inputs and motor actions for other parameters were not fully revealed because the SNN was trained using all four parameter values, and some weak associations between parameters other than Angular velocity may be broadly present, but did not appear in analysis which was limited to pairwise inputs. Another reason for limited tuning for some parameters could be that during training the receptive fields for those parameters were not parsed, and therefore the model could not learn any specific associations for those inputs. This is indicated in the receptive fields parsed during the episodes of the game played after training (**FIg. 7**).

Our modeling underlines the benefits to training from a set of multiple initial network configurations, achieved through varying synaptic connection weights or network architectures (Stanley and Miikkulainen 2002; Neymotin et al. 2013), and testing the populations' performance. The best performing model can then be used for longer-term training. An alternative is to use multiple models from the populatio, and ideally promote model diversity. We highlight these issues by means of the training strategies used in this work. In our STDP-RL model, we first selected a set of hyperparameters suitable for optimizing learning and then tuned for initial synaptic weights using the fixed optimal hyperparameter values. Thus, the initial weights and the learning parameter values differed from those at the beginning of the hyperparameter search. In our EVOL strategy, without using any explicit synaptic learning mechanism, we evaluated the model's performance with different synaptic weights and found synaptic weight distributions showing good performance. Comparing the distributions of synaptic weights in the model trained using STDP-RL and EVOL, we showed that multiple network configurations and synaptic weight patterns can lead to similar performance in CartPole.

The standard EVOL algorithm we used creates populations of models, evaluates them, and then applies a mutation based on each model's fitness to create a new generation for evaluation (see **Materials and Methods**). Our results show that using EVOL and STDP-RL strategies interleaved enhances the training capabilities and produces network models with diverse synaptic weights. This interleaved EVOL+STDP-RL algorithm was inspired by the fact that in addition to genetic modifications occurring during evolution, learning takes place during an animal's lifetime. Successful learning from evolution leads to more fit offspring, providing favorable initial conditions for the STDP-RL algorithm to learn better from experience during the model's *lifetime*. In this setup, the EVOL algorithm creates a set of conditions that optimize an individual model's STDP-RL learning. In EVOL+STDP-RL, the fitness score of the trained model alone influences the creation of the new generation, and the synaptic weights changed using STDP-RL are not directly transferred to succeeding generations. However, there is evidence that learning during an animal's lifetime could enable changes to its phenotype to transfer to offspring through epigenetic modifications (Harper 2005). These epigenetic modifications are impermanent and do not involve direct modification of the genetic sequence (Berger et al. 2009). Future extensions to our algorithm could draw inspiration from this process to allow transfer of STDP-RL weights across generations. We also plan to explore optimization of network architecture including the numbers of neurons, numbers of layers, and the inclusion of structural plasticity rules (Stanley et al. 2019; Martin and Pilly 2019; Kolouri et al. 2019; Parisi et al. 2019; Whitelam et al. 2021). These types of enhancements could further improve performance and offer additional insights into biological processes.

In the past, the use of evolutionary algorithms in neurobiological models has mainly been limited to optimizing individual neurons (Werner Van Geit, Achard, and De Schutter 2007; W. Van Geit, De Schutter, and Achard 2008; Rumbell et al. 2016; Neymotin et al. 2017), or neuronal networks through hyperparameter tuning (S. Dura-Bernal et al. 2017). Although more recent work makes changes to network architectures (Stanley et al. 2019), modifications of synaptic weight matrices in spiking or biophysical neuronal networks have rarely been performed, partly due to the large computational costs associated with searching through the high dimensional space. Here we have demonstrated that evolutionary algorithms operating on synaptic weight matrices are an effective strategy to train SNNs to perform behaviors in a dynamic environment.

We previously used the STDP-RL learning rules to train a visual/motor cortex model to play Pong (Anwar et al., n.d.). That model required additional complexity for encoding the visual scene (object location, motion direction). This complexity made it more challenging to decipher the role of different components/parameters of the learning algorithms and how to optimize them. In the present SNN model, the lack of visual cortex was a simplification that allowed us to perform a more extensive hyperparameter search to increase the chances for STDP-RL to succeed. After hyperparameter optimization, the STDP-RL algorithm was effective in producing excellent performance in CartPole. Furthermore, the use of CartPole and the simpler sensory/motor cortex model also allowed us to test long optimizations using EVOL in parallel on supercomputers. Our individual STDP-RL and EVOL training results also led to the interleaved EVOL+STDP-RL algorithm, which displayed superior performance. In the future, with the knowledge gained here, we will test our new algorithms using more complex models, tasks, and environments.

Comparison between artificial and biological neural networks

Despite the progress made by the artificial intelligence (AI) community in creating ANNs able to solve complex sensory-motor tasks with super-human performance (Volodymyr Mnih et al. 2015), biological intelligence remains only an inspiration for incorporating continual and efficient learning in neural network models. A well-known AI success is the deep reinforcement learning algorithms (e.g., deep Q network), which can train ANNs to play Atari games (V. Mnih et al. 2013; Volodymyr Mnih et al. 2015). This line of research was recently extended to solve more complex tasks including the game of Go (Silver et al. 2016), although the strategy relied on non-biological data structures (*trees*) and search procedures, which limit the applicability of using the algorithms for understanding neurobiology. As with all algorithms, DL's performance also depends on the computational resources and training data available. DL is often considered data and power hungry (Thompson et al. 2020), typically requiring training over many iterations and high computer memory to store vast sets of training weights that scale with task and network architecture complexity. Since the goal in DL models is to optimize input/output relationships, training for multiple types of datasets or complex behaviors can increase computational and memory requirements beyond a typical laptop's processing power. Although there is no proof yet that the more biologically-faithful SNNs can perform these more difficult tasks with less computational resources, our inspiration to build SNN models tests the hypothesis that brain-like architectures and biological learning mechanisms can be computationally and energy-efficient despite *imperfect* performance. The human brain demonstrates particularly energy-efficient computation (Beaulieu-Laroche et al. 2021), although it developed after a long process of evolution, which incurred a high cost. Using inspiration from the human brain's dynamics and architecture should inform future SNN designs.

The recent successes in deep RL to train ANNs to solve control problems robustly has inspired the scientific community to devote additional efforts towards developing algorithms to address issues like adaptation to rapidly changing environments, continual learning, and generalization to solve novel problems (Parisi et al. 2019). These efforts sometimes derive inspiration from biology, yet little effort is devoted towards developing biologically plausible SNNs trained to solve complex tasks. Limited effort in this direction is partly due to inefficient learning and suboptimal performance of existing SNN models. Lack of interest in building biologically detailed models is also attributed to the scarcity of biological data needed to constrain model parameters. With increasing availability of neurobiological data and access to efficient computational resources and modeling tools (Salvador Dura-Bernal et al. 2019; Hazan et al. 2018; Neymotin et al. 2020; N. T. Carnevale and Hines 2006; T. Carnevale et al. 2014), it is becoming feasible to develop biologically detailed neural network models with biologically realistic learning algorithms (Anwar et al., n.d.; Calderon, Verguts, and Frank 2022; Sanda, Skorheim, and Bazhenov 2017).

A question is often asked by the machine learning community: why do we need biologically detailed neuronal networks when we already have effective ANNs? One rationale for building biologically detailed models is to understand the mechanisms of learning, which are difficult to dissect in behaving subjects due to the complexity of the brain's circuits, and the lack of noninvasive recording technologies offering high spatiotemporal resolution. Also it is desirable to explore the untapped potential of biological learning and intelligence that emerges from nonlinear interactions between microcircuit, neuronal, and synaptic elements. Including biological learning mechanisms and circuit details have already improved the efficiency and robustness of ANNs (Serre, Oliva, and Poggio 2007). We expect similar advances in biologically detailed neuronal networks by incorporating biologically inspired learning mechanisms. Even simplistic SNNs are useful, and have been shown to be Turing-complete (Maass 1996b), and more powerful than ANNs (Maass 1997, [a] 1996). Developing more advanced learning SNNs will also be useful in neuromorphic hardware because of its low energy use (van Albada et al. 2018).

Timescales of learning algorithms

Despite the substantial differences between ANNs and SNNs, learning in both is primarily realized by adjusting the weights of connections or synaptic strengths among interconnected neurons. In ANNs, this usually occurs through the following sequence, repeated many times: 1) inputs are sampled, 2) corresponding outputs are evaluated, and 3) weights are adjusted to minimize the output error via back-propagation through hidden network layers (Schmidhuber 2015). In SNNs, the weights are often adjusted using hebbian or spike-timing dependent plasticity (STDP) rules (Dan and Poo 2004, 2006; Izhikevich 2007; Farries and Fairhall 2007; Caporale and Dan 2008). These strategies are useful when there is a temporally proximate relationship between inputs and outputs and there is no feedback involved.

Sensory-motor RL is a more difficult problem to solve because behaviors require evaluation of many sub-actions, and are associated with different environmental cues integrated over time. Moreover, the reward/punishment feedback is delivered later, which makes it difficult to attribute reward/punishment only to relevant actions and neuron groups producing those actions (Izhikevich 2007). Recent ANNs have taken advantage of a *replay and update* strategy to re-sample previous experiences and shape the action policy for given sensory cues maximizing the cumulative reward (Hayes et al. 2021). In SNNs using STDP-RL, eligibility traces can be used to associate reward/punishment to corresponding actions and neuronal assemblies backwards in time, as we have implemented in our work here: when a postsynaptic neuron fires within a few milliseconds of presynaptic neurons firing, a synaptic eligibility trace is activated, allowing the synapse to undergo potentiation or depression during the following 1-5 seconds (Anwar et al., n.d.; Patel et al. 2019; Hazan et al. 2018; Chadderdon et al. 2012). STDP-RL trains SNNs by establishing associations between the neurons encoding the sensory environment and neurons producing actions or sequence of actions, such that appropriate actions are produced for specific sensory cues. The sensory-motor associations are established from reward-modulated synaptic weight changes that occur at each timestep of the simulation. STDP-RL trains at the individual model level, as we consider each separate initialization of an SNN network a separate “individual”.

In contrast to individual-based algorithms such as STDP-RL, evolutionary algorithms operate at vastly different timescales, and typically use populations of models (Feldman, Aoki, and Kumm 1996; Parisi et al. 2019). Evolution is successful when individuals who are fit enough to produce offspring pass their genes to the next generation (Garrett 2012). While individual learning is restricted to an animal’s lifespan, it still confers powerful competitive advantages. This, in turn, feeds into the evolutionary process: animals that learn the idiosyncrasies of their environment, including its threats and rewards, are more likely to survive and propagate. The obvious genomic storage limitations prevent the encoding of all important environmental information within an animal’s genome (Zador 2019; Koulakov, Shuvaev, and Zador 2021), underlining how individual learning must complement evolution. As noted by James Mark Baldwin, learning on an individual level drives the evolutionary process (“A NEW FACTOR IN EVOLUTION” 1896). To simulate this process, our EVO+STDP-RL strategy implicitly chose the next generation’s weights and connectivity patterns that provided advantages in learning, thereby accelerating population-level fitness improvements. This is consistent with the accelerated learning of models when deploying the Baldwin effect, as has been recently demonstrated in embodied intelligence tasks (Gupta et al. 2021). The extent to which these strategies translate to more complex tasks and circuit architectures could offer further insights into multi-scale neurobiological learning.

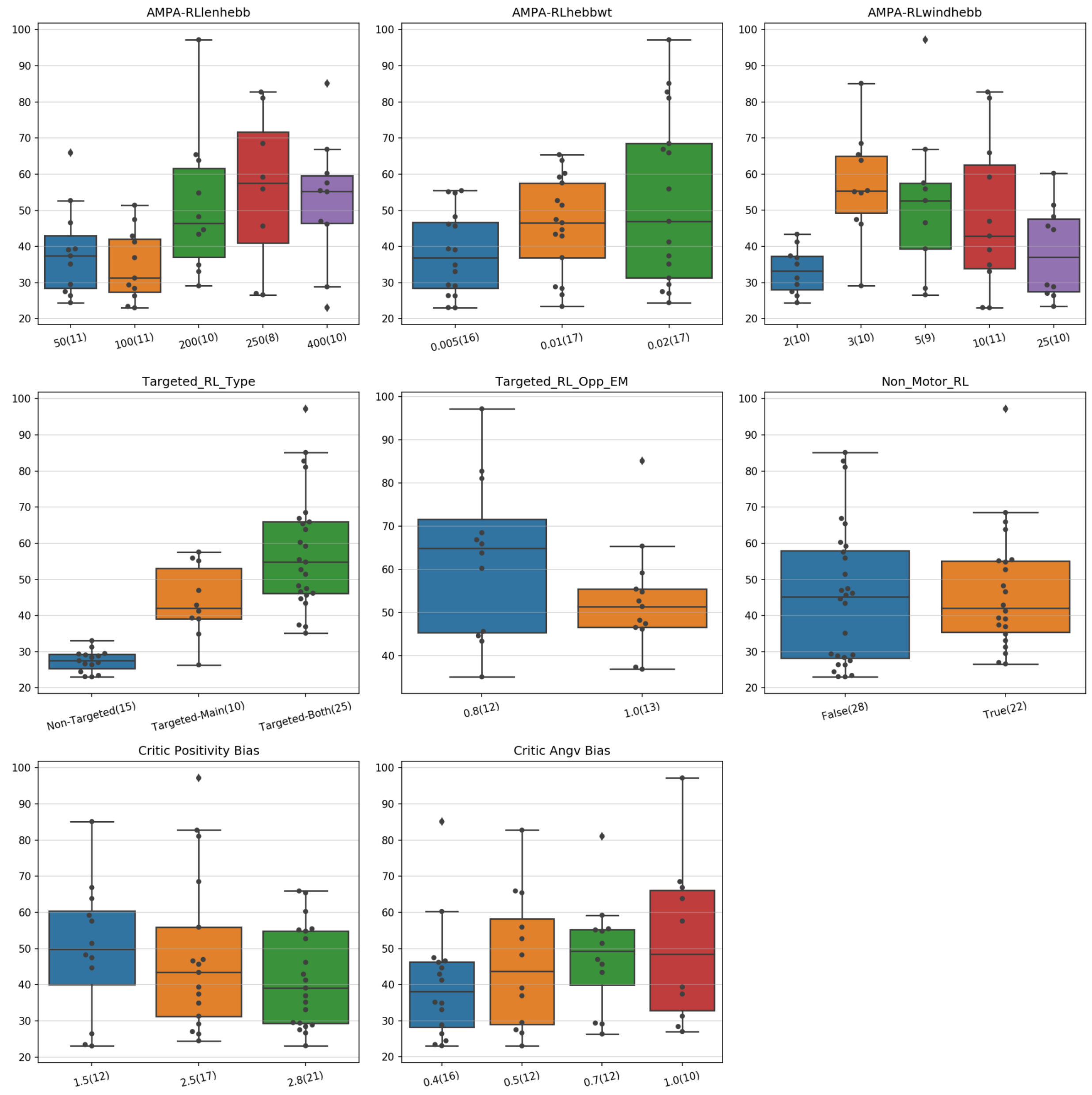
Acknowledgments

This research was funded by ARO-W911NF-19-1-0402, ARO-URAP supplement, and NIDCD-R01DC012947-06A1. The authors acknowledge the Tufts University High Performance Compute Cluster (https://it.tufts.edu/high-performance-computing) which was utilized for the research reported in this paper. In addition, some of the calculations done in this work used the Extreme Science and Engineering Discovery Environment (XSEDE) Bridges GPU Artificial Intelligence at Pittsburgh Supercomputing Center through allocation CCR200032. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S.Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

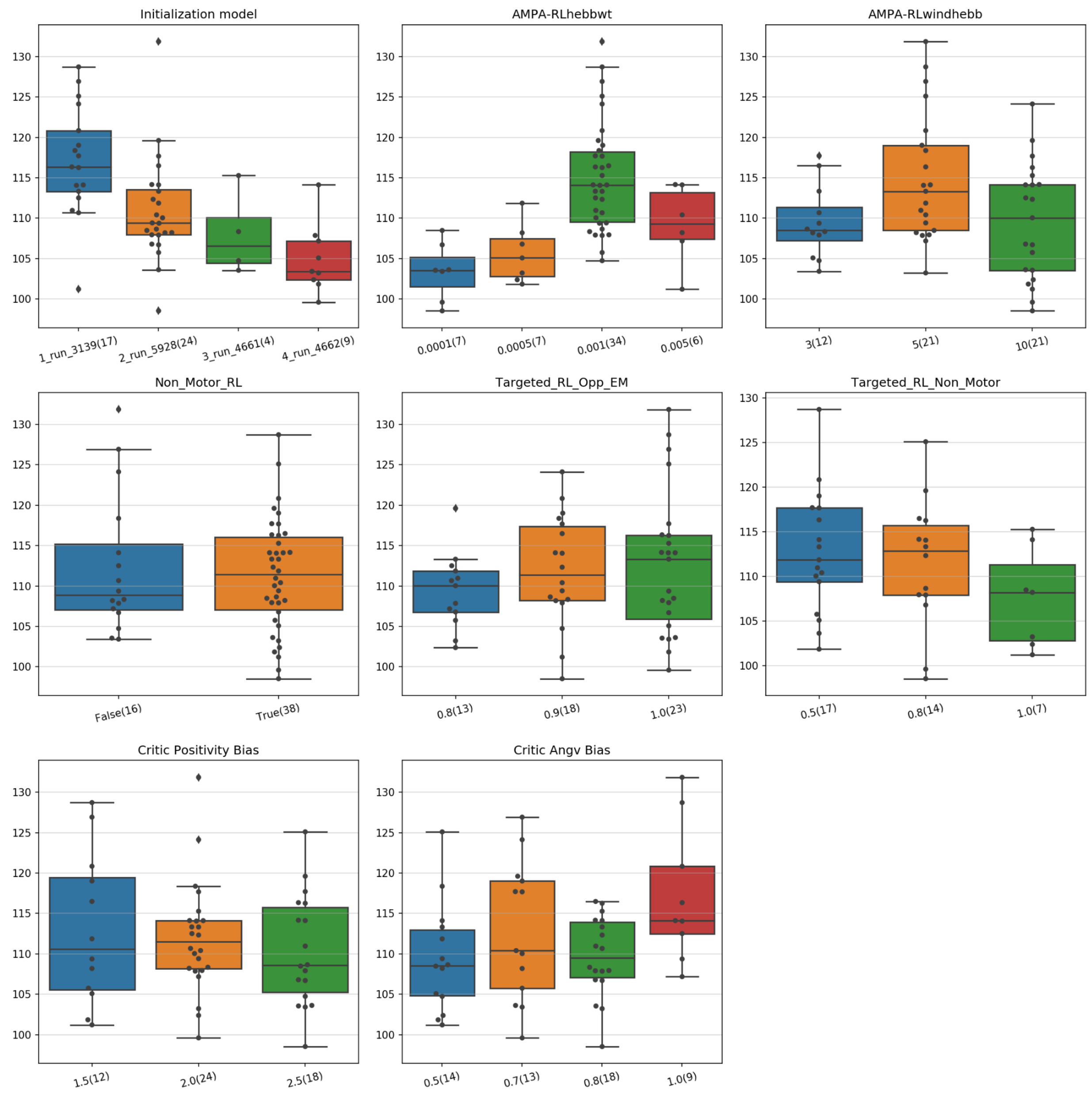
Contribution to the field statement

Biological learning operates at multiple interlocking timescales, over long evolutionary stretches, down to an individual’s life’s relatively short time span. While each process has been simulated individually as a basic learning algorithm in the context of spiking neuronal networks (SNNs), the integration of the two has remained limited. In this study, we first train SNNs to play the CartPole game using: 1) learning during a model’s lifetime and 2) simulated evolutionary learning processes. We then develop an integrated algorithm that combines these types of learning in sequence, more closely mimicking actual evolutionary processes. We evaluate the performance of each algorithm after training and through the creation of sensory/motor action maps that delineate the network’s transformation of sensory inputs into higher-order representations and eventual motor decisions. Our algorithms produced SNNs capable of moving the cart left and right and keeping the pole vertical. Our interleaved training paradigm produced the most robust learning performance. Analysis of synaptic weight matrices also showed different weight patterns that influenced neuronal dynamics. Our modeling opens up new capabilities for SNNs in reinforcement learning and could serve as a testbed for neurobiologists aiming to understand multi-timescale learning mechanisms and dynamics in the brain.

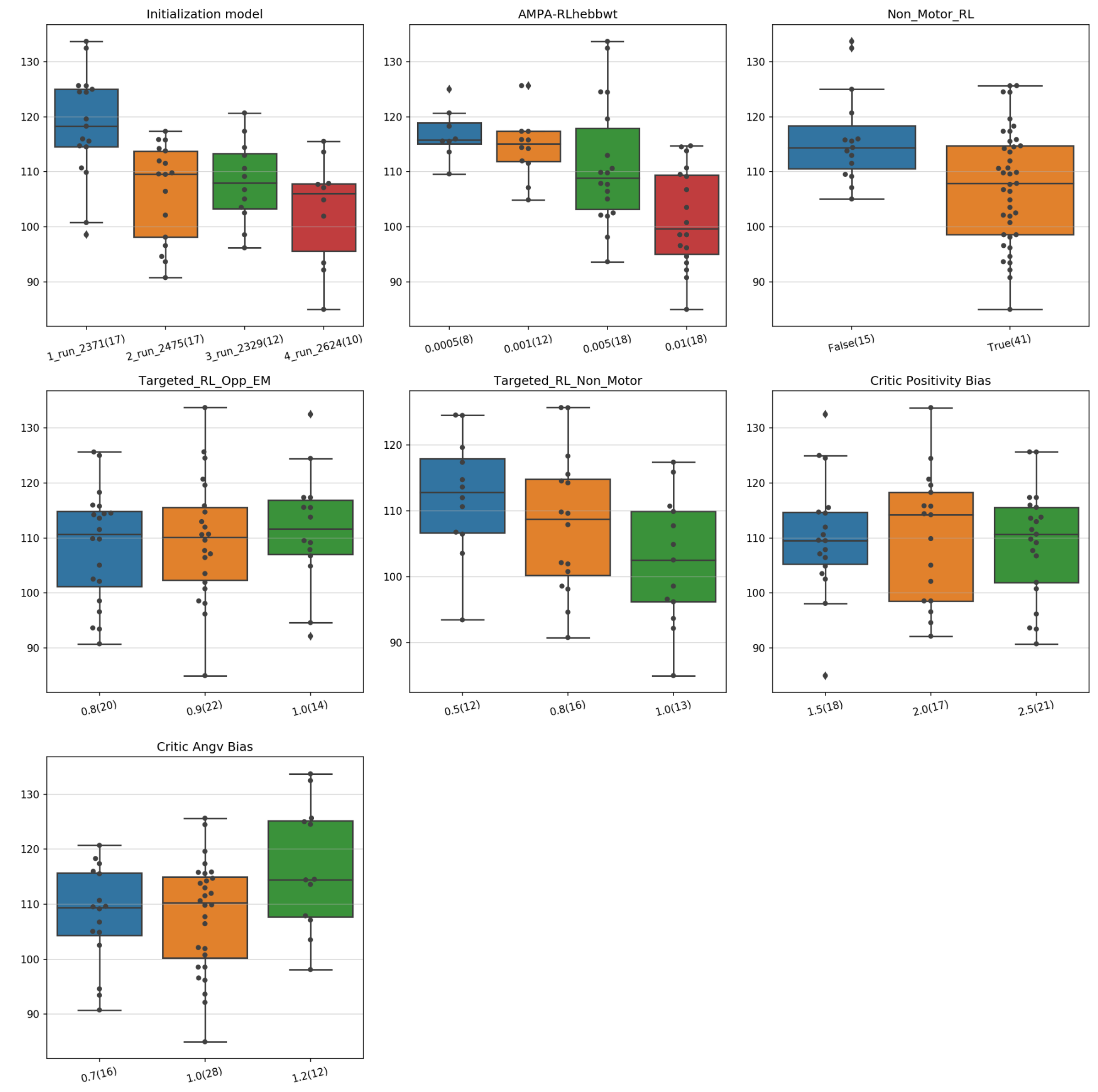
Supplementary Figures



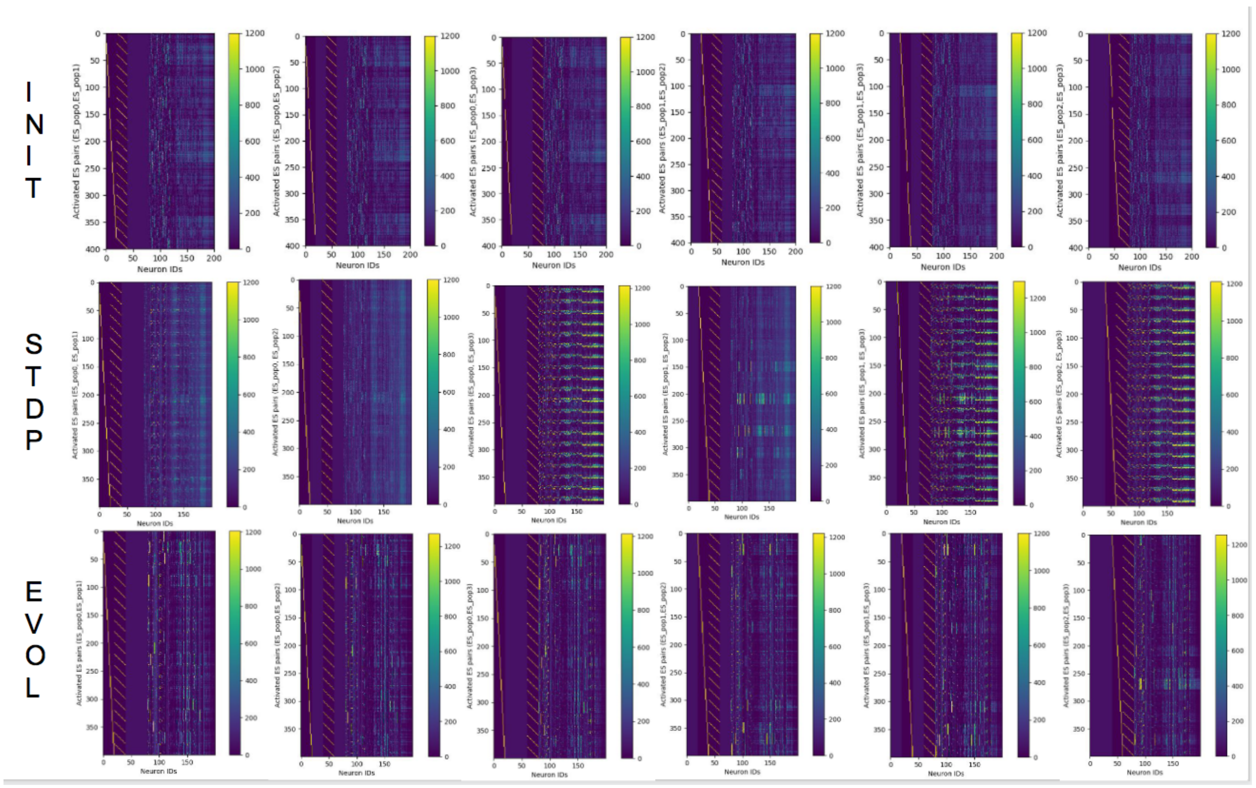
Supplementary Figure 1: Performance distribution of the first hyperparameter search for training using STDP-RL (displaying averages over 100 episodes during training). *In each panel, the y-axis shows the performance, and the x-axis indicates the parameter values used in the evaluation. The number of models run with the specific parameter value is presented in parentheses, with 107 tested hyperparameter combinations.*



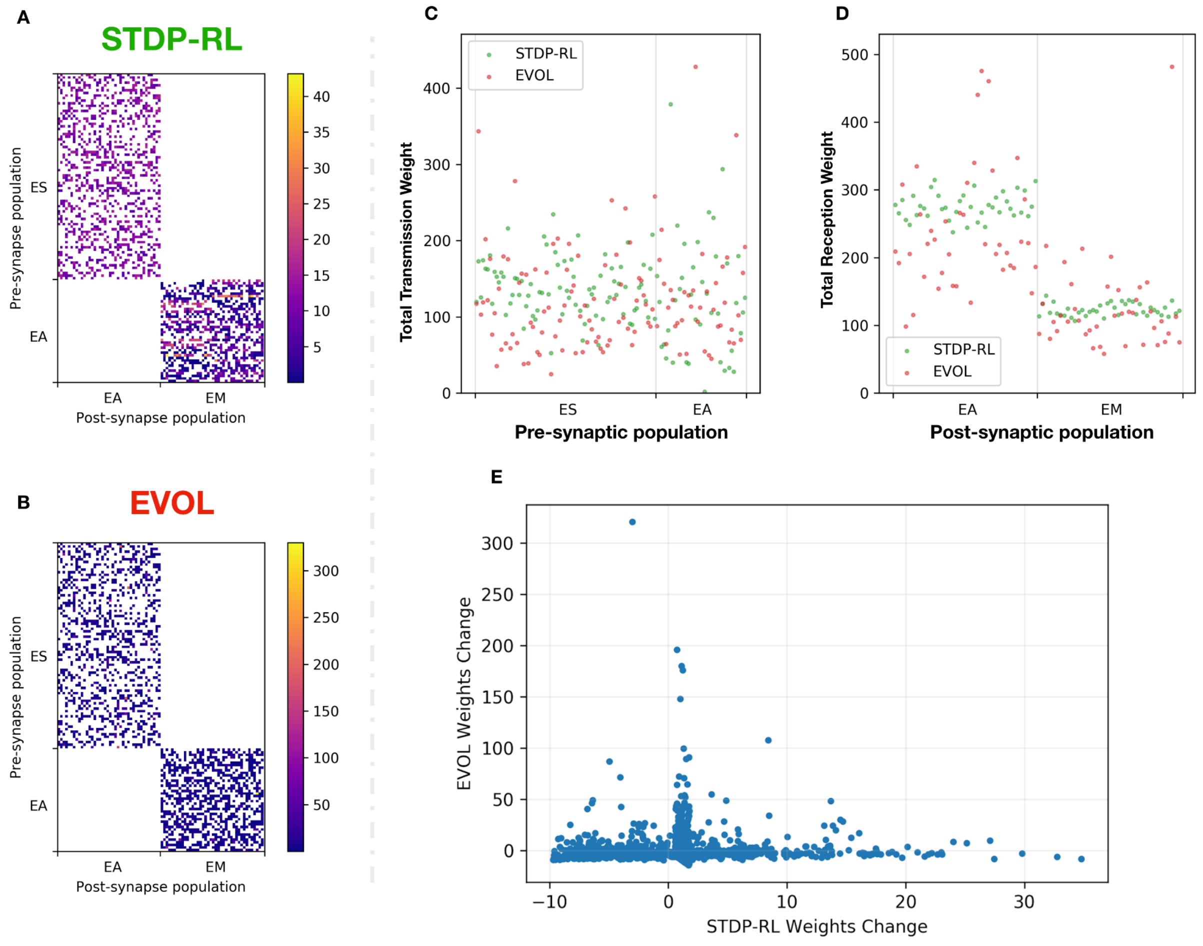
***Supplementary Figure 2:** Performance distribution of hyperparameters for training using STDP-RL evaluated in the second step (using averages over 100 episodes).*



Supplementary Figure 3: *Performance distribution of hyperparameters for training using STDP-RL evaluated in the third step (using averages over 100 episodes).*



Supplementary Figure 4: *Number of spikes generated by each neuron in the model in response to stimulating pairs of sensory input neurons in ES. Each pair of sensory input neurons was stimulated 400 times with combinations of 2 other sensory input neurons. The neuronal responses of the model before training (top row), for after training using STDP-RL (middle row) and EVOL(bottom row).*



***Supplementary Figure 5: STDP-RL and EVOL differentially modulate the synaptic weights of the SNN model during training. (A-B)** Adjacency matrices showing the weights of synaptic connections between ES-EA and EA-EM populations in STDP-RL model **(A)**, EVOL model **(B).** Note that the adjacency matrices have different scales as EVOL model has weights reaching 300. **(C)** Total Transmission Weight: Sum of synaptic weights from a presynaptic neuron onto multiple postsynaptic neurons. **(D)** Total Reception Weight: Sum of synaptic weights onto each postsynaptic neuron from multiple presynaptic neurons. **(E)** As each model changed the weights from the original initialization, each dot represents a synaptic connection weight after training with STDP-RL (x-axis) and EVOL (y-axis) with a Spearman correlation of -0.09 (p-value = 0.0001).*

***Supplementary Table 1: STDP-RL parameters obtained through hyperparameter search.** Values tested that are bolded are the hyperparameters picked for our model.*

Parameter	**Hyperparameter Search Step 1**	**Hyperparameter Search Step 2**	**Hyperparameter Search Step 3**	**Description**
Duration (s) (not varying)	**500**	**2000**	**2000+**	Duration in seconds that was trained using each. A

				step was performed in 50ms
AMPA-RLwindhebb (ms)	(2, **3**, 5, 10, 25)	(3, **5**, 10)	**5**	Maximum time between presynaptic and postsynaptic spike times for considering plasticity.
AMPA-RLlenhebb (ms)	(50, 100, 200, 250, **400**)	**250**	**250**	The decay time constant of the exponentially decreasing eligibility trace.
AMPA-RLhebbwt	(0.005, 0.01, **0.02**)	(0.0001, 0.0005, **0.001**, 0.005)	(0.0005, 0.001, **0.005**, 0.01)	Max synaptic weight adjustments based on reward or punishing signal.
Targeted_RL_Type	(non-targeted RL, targeted RL main, **targeted RL both**)	**targeted RL both**	**targeted RL both**	The Targeted RL paradigm chosen
Non_Motor_RL	(**False**, True)	(**False**, True)	(**False**, True)	Delivering RL to non EM population: EA
Targeted_RL_Opp_EM	(0.8, **1.0**)	(0.8, 0.9, **1.0**)	(0.8, **0.9**, 1.0)	Attenuation factor for the opposite subpopulation receiving reinforcement
Targeted_RL_Non_Motor	**1.0**	(0.5, 0.8, **1.0**)	(0.5, 0.8, **1.0**)	Attenuation factor for the non motor subpopulation receiving reinforcement
Critic Positivity Bias	(**1.5**, 2.5, 2.8)	(1.5, **2.0**, 2.5)	(1.5, **2.0**, 2.5)	η_{angvel}: used in critic (**equation 1**)
Critic Angv Bias	(**0.4**, 0.5, 0.7, 1.0)	(0.5, 0.7, 0.8, **1.0**)	(0.7, 1.0, **1.2**)	$\eta_{positivity}$: defined for critic evaluation (**equations 2-3**)

References

Altamirano, J.S., Ornelas, M., Espinal, A., Santiago-Montero, R., Puga, H., Carpio, J.M. and Tostado, S., 2015. Comparing Evolutionary Strategy Algorithms for Training Spiking Neural Networks. Res. Comput. Sci., 96, pp.9-17.

Albada, Sacha J. van, Andrew G. Rowley, Johanna Senk, Michael Hopkins, Maximilian Schmidt, Alan B. Stokes, David R. Lester, Markus Diesmann, and Steve B. Furber. 2018. "Performance Comparison of the Digital Neuromorphic Hardware SpiNNaker and the Neural Network Simulation Software NEST for a Full-Scale Cortical Microcircuit Model." *Frontiers in Neuroscience* 12 (May). https://doi.org/10.3389/fnins.2018.00291.

"A NEW FACTOR IN EVOLUTION." 1896. *Science* 4 (83): 139.

Anwar, Haroon, Simon Caby, Salvador Dura-Bernal, David D'Onofrio, Daniel Hasegan, Matt Deible, Sara Grunblatt, et al. n.d. "Training a Spiking Neuronal Network Model of Visual-Motor Cortex to Play a Virtual Racket-Ball Game Using Reinforcement Learning." https://doi.org/10.1101/2021.07.29.454361.

Anwar, Haroon, Xinping Li, Dirk Bucher, and Farzan Nadim. 2017. "Functional Roles of Short-Term Synaptic Plasticity with an Emphasis on Inhibition." *Current Opinion in Neurobiology* 43 (April): 71–78.

Barto, Andrew G., Richard S. Sutton, and Charles W. Anderson. 1983. "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems." *IEEE Transactions on Systems, Man, and Cybernetics*. https://doi.org/10.1109/tsmc.1983.6313077.

Beaulieu-Laroche, Lou, Norma J. Brown, Marissa Hansen, Enrique H. S. Toloza, Jitendra Sharma, Ziv M. Williams, Matthew P. Frosch, Garth Rees Cosgrove, Sydney S. Cash, and Mark T. Harnett. 2021. "Allometric Rules for Mammalian Cortical Layer 5 Neuron Biophysics." *Nature*, November. https://doi.org/10.1038/s41586-021-04072-3.

Berger, Shelley L., Tony Kouzarides, Ramin Shiekhattar, and Ali Shilatifard. 2009. "An Operational Definition of Epigenetics." *Genes & Development* 23 (7): 781–83.

Boers, Egbert J. W., Marko V. Borst, and Ida G. Sprinkhuizen-Kuyper. 1995. "Evolving Neural Networks Using the 'Baldwin Effect.'" In *Artificial Neural Nets and Genetic Algorithms*, edited by David W. Pearson, Nigel C. Steele, and Rudolf F. Albrecht, 333–36. Springer Vienna.

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. "OpenAI Gym." *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1606.01540.

Calderon, Cristian Buc, Tom Verguts, and Michael J. Frank. 2022. "Thunderstruck: The ACDC Model of Flexible Sequences and Rhythms in Recurrent Neural Circuits." *PLoS Computational Biology* 18 (2): e1009854.

Caporale, Natalia, and Yang Dan. 2008. "Spike Timing–Dependent Plasticity: A Hebbian Learning Rule," June. https://doi.org/10.1146/annurev.neuro.31.060407.125639.

Carnevale, N. T., and M. L. Hines. 2006. *The NEURON Book*. Cambridge University Press.

Carnevale, Ted, Amit Majumdar, Subha Sivagnanam, Kenneth Yoshimoto, Vadim Astakhov, Anita Bandrowski, and Maryann Martone. 2014. "The Neuroscience Gateway Portal: High Performance Computing Made Easy." *BMC Neuroscience* 15 (S1): P101.

Chadderdon, George L., Samuel A. Neymotin, Cliff C. Kerr, and William W. Lytton. 2012. "Reinforcement Learning of Targeted Movement in a Spiking Neuronal Model of Motor Cortex." *PloS One* 7 (10): e47251.

Chadderdon, George L., and Olaf Sporns. 2006. "A Large-Scale Neurocomputational Model of Task-Oriented Behavior Selection and Working Memory in Prefrontal Cortex." *Journal of*

Cognitive Neuroscience 18 (2): 242–57.

Chrabaszcz, Patryk, Ilya Loshchilov, and Frank Hutter. 2018. “Back to Basics: Benchmarking Canonical Evolution Strategies for Playing Atari.” *arXiv [cs.NE]*. arXiv. http://arxiv.org/abs/1802.08842.

Dan, Yang, and Mu-Ming Poo. 2004. “Spike Timing-Dependent Plasticity of Neural Circuits.” *Neuron* 44 (1): 23–30.

———. 2006. “Spike Timing-Dependent Plasticity: From Synapse to Perception.” *Physiological Reviews* 86 (3): 1033–48.

Dura-Bernal, Salvador, Kan Li, Samuel A. Neymotin, Joseph T. Francis, Jose C. Principe, and William W. Lytton. 2016. “Restoring Behavior via Inverse Neurocontroller in a Lesioned Cortical Spiking Model Driving a Virtual Arm.” *Frontiers in Neuroscience* 10 (February): 28.

Dura-Bernal, Salvador, Benjamin A. Suter, Padraig Gleeson, Matteo Cantarelli, Adrian Quintana, Facundo Rodriguez, David J. Kedziora, et al. 2019. “NetPyNE, a Tool for Data-Driven Multiscale Modeling of Brain Circuits.” *eLife* 8 (April). https://doi.org/10.7554/eLife.44494.

Dura-Bernal, S., S. A. Neymotin, C. C. Kerr, S. Sivagnanam, A. Majumdar, J. T. Francis, and W. W. Lytton. 2017. “Evolutionary Algorithm Optimization of Biological Learning Parameters in a Biomimetic Neuroprosthesis.” *IBM Journal of Research and Development* 61 (2-3): 6.1–6.14.

Escobar, Maria-Jose, Guillaume S. Masson, Thierry Vieville, and Pierre Kornprobst. 2009. “Action Recognition Using a Bio-Inspired Feedforward Spiking Network.” *International Journal of Computer Vision* 82 (3): 284.

Espinal, Andrés, Martín Carpio, Manuel Ornelas, Héctor Puga, Patricia Melin, and Marco Sotelo-Figueroa. 2014. “Comparing Metaheuristic Algorithms on the Training Process of Spiking Neural Networks.” In *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, edited by Oscar Castillo, Patricia Melin, Witold Pedrycz, and Janusz Kacprzyk, 391–403. Cham: Springer International Publishing.

Farries, Michael A., and Adrienne L. Fairhall. 2007. “Reinforcement Learning With Modulated Spike Timing–Dependent Synaptic Plasticity.” *Journal of Neurophysiology* 98 (6): 3648–65.

Feldman, Marcus W., Kenichi Aoki, and Jochen Kumm. 1996. “Individual Versus Social Learning: Evolutionary Analysis in a Fluctuating Environment.” *Anthropological Science: Journal of the Anthropological Society of Nippon = Jinruigaku Zasshi* 104 (3): 209–31.

Garrett, A. 2012. “Inspyred: Bio-Inspired Algorithms in Python.” *URL Https://pypi. Python. Org/pypi/inspyred*.

Geva, S., and J. Sitte. 1993. “A Cartpole Experiment Benchmark for Trainable Controllers.” *IEEE Control Systems Magazine* 13 (5): 40–51.

Gupta, Ankur, and Lyle N. Long. 2007. “Character Recognition Using Spiking Neural Networks.” In *2007 International Joint Conference on Neural Networks*, 53–58.

Harper, Lawrence V. 2005. “Epigenetic Inheritance and the Intergenerational Transfer of Experience.” *Psychological Bulletin* 131 (3): 340–60.

Hayes, Tyler L., Giri P. Krishnan, Maxim Bazhenov, Hava T. Siegelmann, Terrence J. Sejnowski, and Christopher Kanan. 2021. “Replay in Deep Learning: Current Approaches and Missing Biological Elements.” *arXiv [q-bio.NC]*. arXiv. http://arxiv.org/abs/2104.04132.

Hazan, Hananel, Daniel J. Saunders, Hassaan Khan, Devdhar Patel, Darpan T. Sanghavi, Hava T. Siegelmann, and Robert Kozma. 2018. “BindsNET: A Machine Learning-Oriented Spiking Neural Networks Library in Python.” *Frontiers in Neuroinformatics* 12 (December): 89.

Hinton, Geoffrey E., and Steven J. Nowlan. 1986. *How Learning Can Guide Evolution*. Computer Science Department, Carnegie-Mellon Univ.

Izhikevich, Eugene M. 2007. “Solving the Distal Reward Problem through Linkage of STDP and Dopamine Signaling.” *Cerebral Cortex* 17 (10): 2443–52.

Kasabov, Nikola, Valery Feigin, Zeng-Guang Hou, Yixiong Chen, Linda Liang, Rita

Krishnamurthi, Muhaini Othman, and Priya Parmar. 2014. “Evolving Spiking Neural Networks for Personalised Modelling, Classification and Prediction of Spatio-Temporal Patterns with a Case Study on Stroke.” *Neurocomputing* 134 (June): 269–79.

Kolouri, S., N. Ketz, X. Zou, J. Krichmar, and P. Pilly. 2019. “Attention-Based Structural-Plasticity.” *arXiv Preprint arXiv*. https://arxiv.org/abs/1903.06070.

Koulakov, A., S. Shuvaev, and A. Zador. 2021. “Encoding Innate Ability through a Genomic Bottleneck.” *bioRxiv*. https://www.biorxiv.org/content/10.1101/2021.03.16.435261v1.abstract.

Kozdon, Katarzyna, and Peter Bentley. 2018. “The Evolution of Training Parameters for Spiking Neural Networks with Hebbian Learning.” In *ALIFE 2018: The 2018 Conference on Artificial Life*, 276–83. MIT Press.

Lytton, William W., and Ahmet Omurtag. 2007. “Tonic-Clonic Transitions in Computer Simulation.” *Journal of Clinical Neurophysiology: Official Publication of the American Electroencephalographic Society* 24 (2): 175–81.

Lytton, William W., Ahmet Omurtag, Samuel A. Neymotin, and Michael L. Hines. 2008. “Just-in-Time Connectivity for Large Spiking Networks.” *Neural Computation* 20 (11): 2745–56.

Lytton, William W., and Mark Stewart. 2006. “Rule-Based Firing for Network Simulations.” *Neurocomputing* 69 (10): 1160–64.

Maass, Wolfgang. 1996a. “Noisy Spiking Neurons with Temporal Coding Have More Computational Power than Sigmoidal Neurons.” *Advances in Neural Information Processing Systems* 9. https://proceedings.neurips.cc/paper/1996/hash/f93882cbd8fc7fb794c1011d63be6fb6-Abstract.html.

———. 1996b. “Lower Bounds for the Computational Power of Networks of Spiking Neurons.” *Neural Computation* 8 (1): 1–40.

———. 1997. “Networks of Spiking Neurons: The Third Generation of Neural Network Models.” *Neural Networks: The Official Journal of the International Neural Network Society* 10 (9): 1659–71.

Martin, Charles E., and Praveen K. Pilly. 2019. “Probabilistic Program Neurogenesis.” In *ALIFE 2019: The 2019 Conference on Artificial Life*, 440–47. MIT Press.

Mnih, V., K. Kavukcuoglu, D. Silver, and A. Graves. 2013. “Playing Atari with Deep Reinforcement Learning.” *arXiv Preprint arXiv*. https://arxiv.org/abs/1312.5602.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. 2015. “Human-Level Control through Deep Reinforcement Learning.” *Nature* 518 (7540): 529–33.

Mozafari, Milad, Saeed Reza Kheradpisheh, Timothee Masquelier, Abbas Nowzari-Dalini, and Mohammad Ganjtabesh. 2018. “First-Spike-Based Visual Categorization Using Reward-Modulated STDP.” *IEEE Transactions on Neural Networks and Learning Systems* 29 (12): 6178–90.

Neymotin, Samuel A., George L. Chadderdon, Cliff C. Kerr, Joseph T. Francis, and William W. Lytton. 2013. “Reinforcement Learning of Two-Joint Virtual Arm Reaching in a Computer Model of Sensorimotor Cortex.” *Neural Computation* 25 (12): 3263–93.

Neymotin, Samuel A., Dylan S. Daniels, Blake Caldwell, Robert A. McDougal, Nicholas T. Carnevale, Mainak Jas, Christopher I. Moore, Michael L. Hines, Matti Hämäläinen, and Stephanie R. Jones. 2020. “Human Neocortical Neurosolver (HNN), a New Software Tool for Interpreting the Cellular and Network Origin of Human MEG/EEG Data.” *eLife* 9 (January): 740597.

Neymotin, Samuel A., Heekyung Lee, Eunhye Park, André A. Fenton, and William W. Lytton. 2011. “Emergence of Physiological Oscillation Frequencies in a Computer Model of Neocortex.” *Frontiers in Computational Neuroscience* 5: 19.

Neymotin, Samuel A., Benjamin A. Suter, Salvador Dura-Bernal, Gordon M. G. Shepherd, Michele Migliore, and William W. Lytton. 2017. "Optimizing Computer Models of Corticospinal Neurons to Replicate in Vitro Dynamics." *Journal of Neurophysiology* 117 (1): 148–62.

Parisi, German I., Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. 2019. "Continual Lifelong Learning with Neural Networks: A Review." *Neural Networks: The Official Journal of the International Neural Network Society* 113 (May): 54–71.

Patel, Devdhar, Hananel Hazan, Daniel J. Saunders, Hava Siegelmann, and Robert Kozma. 2019. "Improved Robustness of Reinforcement Learning Policies upon Conversion to Spiking Neuronal Network Platforms Applied to ATARI Games." *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1903.11012.

Rowan, Mark S., Samuel A. Neymotin, and William W. Lytton. 2014. "Electrostimulation to Reduce Synaptic Scaling Driven Progression of Alzheimer's Disease." *Frontiers in Computational Neuroscience* 8 (April): 39.

Rowan, M., and S. Neymotin. 2013. "Synaptic Scaling Balances Learning in a Spiking Model of Neocortex." *International Conference on Adaptive and Natural*. https://link.springer.com/chapter/10.1007/978-3-642-37213-1_3.

Rumbell, Timothy H., Danel Draguljić, Aniruddha Yadav, Patrick R. Hof, Jennifer I. Luebke, and Christina M. Weaver. 2016. "Automated Evolutionary Optimization of Ion Channel Conductances and Kinetics in Models of Young and Aged Rhesus Monkey Pyramidal Neurons." *Journal of Computational Neuroscience* 41 (1): 65–90.

Salimans, Tim, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. "Evolution Strategies as a Scalable Alternative to Reinforcement Learning." *arXiv [stat.ML]*. arXiv. http://arxiv.org/abs/1703.03864.

Sanda, Pavel, Steven Skorheim, and Maxim Bazhenov. 2017. "Multi-Layer Network Utilizing Rewarded Spike Time Dependent Plasticity to Learn a Foraging Task." *PLoS Computational Biology* 13 (9): e1005705.

Schmidhuber, Jürgen. 2015. "Deep Learning in Neural Networks: An Overview." *Neural Networks: The Official Journal of the International Neural Network Society* 61 (January): 85–117.

Serre, Thomas, Aude Oliva, and Tomaso Poggio. 2007. "A Feedforward Architecture Accounts for Rapid Categorization." *Proceedings of the National Academy of Sciences of the United States of America* 104 (15): 6424–29.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, et al. 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search." *Nature* 529 (7587): 484–89.

Stanley, Kenneth O., Jeff Clune, Joel Lehman, and Risto Miikkulainen. 2019. "Designing Neural Networks through Neuroevolution." *Nature Machine Intelligence* 1 (1): 24–35.

Stanley, Kenneth O., and Risto Miikkulainen. 2002. "Evolving Neural Networks through Augmenting Topologies." *Evolutionary Computation* 10 (2): 99–127.

Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning, Second Edition: An Introduction*. MIT Press.

Suzuki, Reiji, and Takaya Arita. 2004. "Interactions between Learning and Evolution: The Outstanding Strategy Generated by the Baldwin Effect." *Bio Systems* 77 (1-3): 57–71.

Tavanaei, Amirhossein, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. 2019. "Deep Learning in Spiking Neural Networks." *Neural Networks: The Official Journal of the International Neural Network Society* 111 (March): 47–63.

Tavanaei, Amirhossein, and Anthony Maida. 2017. "Bio-Inspired Multi-Layer Spiking Neural Network Extracts Discriminative Features from Speech Signals." In *Neural Information Processing*, 899–908. Springer International Publishing.

Thompson, Neil C., Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. 2020. "The

Computational Limits of Deep Learning." *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/2007.05558.

Van Geit, W., E. De Schutter, and P. Achard. 2008. "Automated Neuron Model Optimization Techniques: A Review." *Biological Cybernetics* 99 (4-5): 241–51.

Van Geit, Werner, Pablo Achard, and Erik De Schutter. 2007. "Neurofitter: A Parameter Tuning Package for a Wide Range of Electrophysiological Neuron Models." *Frontiers in Neuroinformatics* 1 (November): 1.

Vogels, T. P., H. Sprekeler, F. Zenke, C. Clopath, and W. Gerstner. 2011. "Inhibitory Plasticity Balances Excitation and Inhibition in Sensory Pathways and Memory Networks." *Science* 334 (6062): 1569–73.

Whitelam, Stephen, Viktor Selin, Sang-Won Park, and Isaac Tamblyn. 2021. "Correspondence between Neuroevolution and Gradient Descent." *Nature Communications* 12 (1): 6317.

Whiteson, Shimon. 2006. "Evolutionary Function Approximation for Reinforcement Learning." *Journal of Machine Learning Research: JMLR* 7. https://www.jmlr.org/papers/volume7/whiteson06a/whiteson06a.pdf.

Whitley, Darrell, V. Scott Gordon, and Keith Mathias. 1994. "Lamarckian Evolution, the Baldwin Effect and Function Optimization." In *Parallel Problem Solving from Nature — PPSN III*, 5–15. Springer Berlin Heidelberg.

Zador, Anthony M. 2019. "A Critique of Pure Learning and What Artificial Neural Networks Can Learn from Animal Brains." *Nature Communications* 10 (1): 3770.

Baldwin, J.M., 1896. A new factor in evolution. Adaptive individuals in evolving populations: Models and algorithms, pp.59-80.

Boers, E.J., Borst, M.V. and Sprinkhuizen-Kuyper, I.G., 1995. Evolving Neural Networks Using the "Baldwin Effect". In Artificial neural nets and genetic algorithms (pp. 333-336). Springer, Vienna.

Chrabaszcz, P., Loshchilov, I. and Hutter, F., 2018. Back to basics: Benchmarking canonical evolution strategies for playing atari. arXiv preprint arXiv:1802.08842.

Escobar, M.J., Masson, G.S., Vieville, T. and Kornprobst, P., 2009. Action recognition using a bio-inspired feedforward spiking network. International journal of computer vision, 82(3), pp.284-301.

Espinal, A., Carpio, M., Ornelas, M., Puga, H., Melin, P. and Sotelo-Figueroa, M., 2014. Comparing metaheuristic algorithms on the training process of spiking neural networks. In Recent Advances on Hybrid Approaches for Designing Intelligent Systems (pp. 391-403). Springer, Cham.

Gupta, A. and Long, L.N., 2007, August. Character recognition using spiking neural networks. In 2007 International Joint Conference on Neural Networks (pp. 53-58). IEEE.

Hinton, G.E. and Nowlan, S.J., 1987. "How learning can guide evolution". Complex Systems, 1, 495–502

Kasabov, N., Feigin, V., Hou, Z.G., Chen, Y., Liang, L., Krishnamurthi, R., Othman, M. and Parmar, P., 2014. Evolving spiking neural networks for personalised modelling, classification and prediction of spatio-temporal patterns with a case study on stroke. Neurocomputing, 134, pp.269-279.

Kozdon, K. and Bentley, P., 2018, July. The evolution of training parameters for spiking neural networks with hebbian learning. In Artificial Life Conference Proceedings (pp. 276-283). One Rogers Street, Cambridge, MA 02142-1209 USA journals-info@ mit. edu: MIT Press.

Maass, W., 1996. Lower bounds for the computational power of networks of spiking neurons. Neural computation, 8(1), pp.1-40.

Maass, W., 1997a. Networks of spiking neurons: the third generation of neural network models. Neural networks, 10(9), pp.1659-1671.

Maass, W., 1997b. Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. In Advances in neural information processing systems (pp. 211-217).

Mozafari, M., Kheradpisheh, S.R., Masquelier, T., Nowzari-Dalini, A. and Ganjtabesh, M., 2018. First-spike-based visual categorization using reward-modulated STDP. IEEE transactions on neural networks and learning systems, 29(12), pp.6178-6190.

Sutton, R. S., & Barto, A. G. 2018. “Reinforcement learning: An introduction.” MIT press.

Suzuki, R. and Arita, T., 2004. Interactions between learning and evolution:: The outstanding strategy generated by the baldwin effect. Biosystems, 77(1-3), pp.57-71.

Tavanaei, A. and Maida, A., 2017, November. Bio-inspired multi-layer spiking neural network extracts discriminative features from speech signals. In International conference on neural information processing (pp. 899-908). Springer, Cham.

Tavanaei, A., Ghodrati, M., Kheradpisheh, S.R., Masquelier, T. and Maida, A., 2019. Deep learning in spiking neural networks. Neural Networks, 111, pp.47-63.

Whiteson, S., 2006. Evolutionary function approximation for reinforcement learning. Journal of Machine Learning Research, 7.

Whitley, D., Gordon, V.S. and Mathias, K., 1994, October. Lamarckian evolution, the Baldwin effect and function optimization. In International Conference on Parallel Problem Solving from Nature (pp. 5-15). Springer, Berlin, Heidelberg.