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 time span of an individual's life. 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 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.

We use the NEURON simulator with NetPyNE to create an SNN interfaced with the CartPole environment from OpenAI's Gym. In CartPole, the goal is to balance a vertical pole by moving left/right on a 1-D plane. Our SNN contains multiple populations of neurons organized in three layers: sensory layer, association/hidden layer, and motor layer, where neurons are connected by excitatory (AMPA/NMDA) and inhibitory (GABA) synapses. Association and motor layers contain one excitatory (E) population and two inhibitory (I) populations with different synaptic time constants. Each neuron is an event-based integrate-and-fire model with plastic connections between excitatory neurons. In our SNN, the environment activates sensory neurons tuned to specific features of the game state. We split the motor population into subsets representing each movement choice. The subset with more spiking over an interval determines the action.

During STDP-RL, we supply intermediary evaluations (reward/punishment) of each action by judging the effectiveness of a move (e.g., moving the CartPole to a balanced position). During EVOL, updates consist of adding together many random perturbations of the connection weights. Each set of random perturbations is weighted by the total episodic reward it achieves when applied independently. 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. Both EVOL and STDP-RL training produce SNNs capable of moving the cart left and right and keeping the pole vertical. Compared to the STDP-RL and EVOL algorithms operating on their own, our interleaved training paradigm produced enhanced robustness in performance, with different strategies revealed through analysis of the sensory/motor mappings. Analysis of synaptic weight

matrices also shows distributed vs clustered representations after the EVOL and STDP-RL algorithms, respectively. These weight differences also manifest as diffuse vs synchronized firing patterns. 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

Introduction

Evolutionary changes and changes to individuals during their lifetimes occur at vastly different timescales, and through different, but interacting mechanisms (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. In addition, genomic bottlenecks prevent encoding of all relevant environmental information within an animal's genome (Zador 2019; Koulakov, Shuvaev, and Zador 2021). For all these reasons, learning for survival must take place on an individual level as well.

Previously, computer models of evolutionary and individual learning have been used to study the dynamics and mechanisms of both processes individually (Rumbell et al. 2016; Farries and Fairhall 2007). However, the interaction between the two has not been investigated widely, particularly in the context of spiking neuronal networks (SNNs). Here, we used SNNs of sensory and motor circuits to investigate the effectiveness of these biologically inspired algorithms with different timescales to learn behaviors in a dynamic environment. We model individuals as SNNs that learn using spike-timing dependent reinforcement learning (STDP-RL), a method used previously to train SNNs to perform behaviors (Anwar et al., n.d.; Patel et al. 2019; Hazan et al. 2018; Chadderdon et al. 2012). STDP-RL modifies synaptic weights to push individuals to perform behaviors more optimally within their environment. At longer time-scales we use a version of the evolutionary strategies algorithm (EVOL)(Salimans et al. 2017) shown to be effective in training ANNs to solve complex control problems. In the biological context, EVOL treats the synaptic weight as an individual's genome. EVOL then perturbs the genome using the mutation genetic operator to produce a population of offspring. These offspring then interact with the environment to evaluate their fitness.
EVOL then performs a genetic combination of the offspring's genomes, with each genome being weighted by its corresponding relative environmental fitness. This process then iterates, leading to improved performance over time.

We first study the two learning algorithms individually, comparing performance in the commonly used control problem of CartPole, where the goal is to balance a pole vertically on a cart that can move left or right at each time-step. Both algorithms successfully train our SNN models and produce very different patterns in the weight matrices, with more distributed representations after EVOL and more clustering after STDP-RL. We evaluate the performance of each algorithm after training and create different sensory input to motor output maps, to demonstrate each network's retention of learning and decision-making capabilities. Afterward, we investigate an interleaved algorithm that combines STDP-RL and EVOL in sequence. This sequential algorithm, which parallels actual biological and individual learning, demonstrates superior performance, with enhanced robustness to initial conditions and improved stability. Our modeling offers a novel set of algorithms for training SNNs, that bridge the timescales of biological learning and could shed light on learning mechanisms in vivo.

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. 1A**) available in the OpenAI Gym platform (Brockman et al. 2016) (https://gym.openai.com). To keep the pole balanced (**Fig. 1A**), 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. 1A**). 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.

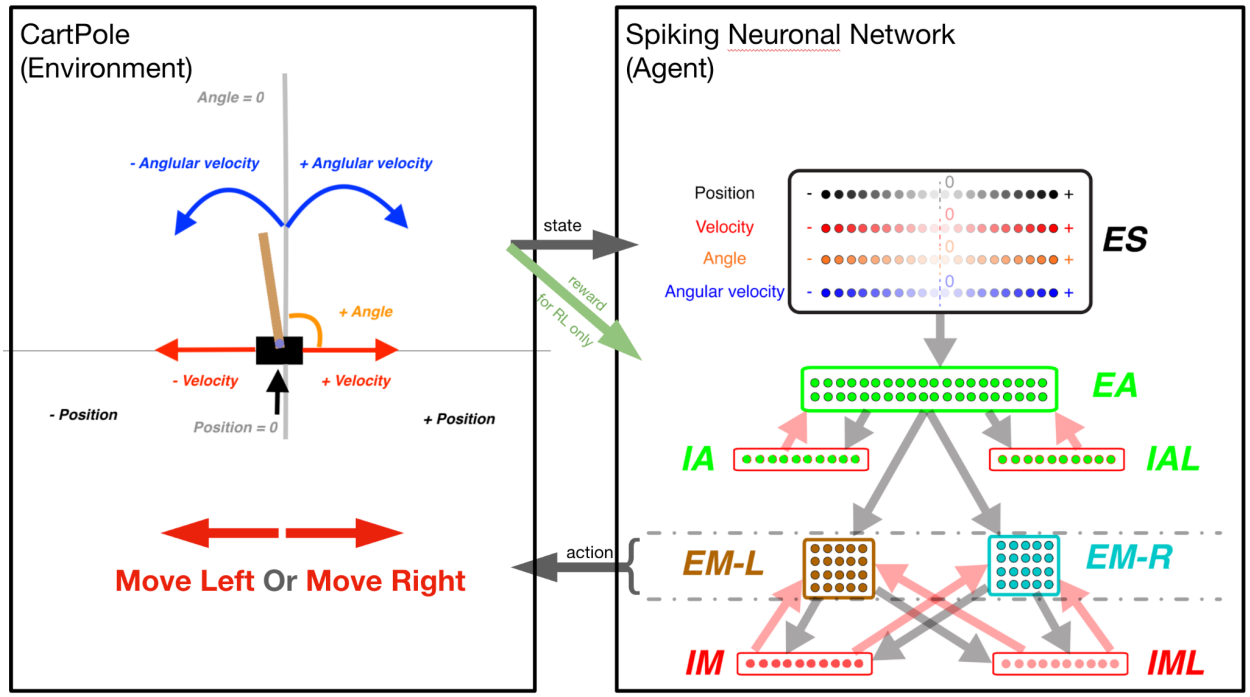


Figure 1: *The CartPole game environment interfacing with the SNN model. At each game-step, a new game state is produced which is described by the game state variables (as labeled): Position, Velocity, Angle, and Angular Velocity. The values of these variables are used to stimulate a unique quadruple of the ES neurons (1 for each state variable) in the spiking neuronal circuit, which produces action for the next game-step. The network is represented at scale, as each dot represents one neuron. The gray and red arrows represent excitatory and inhibitory synapses between neuronal populations.*

Simulations

We used the NEURON simulation environment (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 an SNN using the NetPyNE package that coordinates the network simulation environment and integrates it with CartPole. The CartPole environment and the network simulation (the agent) is coordinated 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, 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

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 encode position, velocity, angle, and angular velocity (**Fig. 1B**). Each neuron was assigned 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. When the balance was lost, i.e., the game-state variable deviated to larger values, the episode ended, requiring less precise encoding for those extreme values. 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 such that as we moved to lower neuron indices, it represented the negative values of the state-variables in decreasing order. When we moved to higher neuron indices, it represented the 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), which received inputs from the ES population. Each neuron in EA was connected to motor areas EM-L and EM-RL generating Right- and Left-actions 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). 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 times 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 (AMPA, NMDA, 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 (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 Plasticity "on" in **Table 2**).

The Neuronal Weights

Initially, the weights between populations were adjusted to allow reliable transmission of spiking activity across different layers/areas of the SNN model. Then, we used the same initialization for both training strategies. The initial connection weights were not subject to hyperparameter tuning.

Table 2: Initial connection weights

Pre-synaptic population	Post-synaptic population	Connection Convergence	Synapse Type	Synaptic Weight	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	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	IM	16	AM2	5.85	
			NM2	0.0585	
EM	IML	16	AM2	2.94	

			NM2	0.294	
IM	EM	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

Traditionally, when SNN models are trained to perform a behavior, biologically inspired learning mechanisms such as STDP and RL are used (Anwar et al., n.d.). These mechanisms establish 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 typically implemented by means of changes in the synaptic weights between sensory and motor neurons or other associated areas. Therefore, any approach that allows us to evaluate the performance of an SNN while changing synaptic weights between neurons can be used for behavioral training. Here, in addition to the reinforcement learning strategy: STDP-RL, we also used the evolutionary strategies method (Salimans et al. 2017), which we denote in this paper as EVOL, to optimize synaptic weights in the SNN model to play CartPole.

Spike-timing dependent Reinforcement Learning (STDP-RL)

We used the existing STDP-RL mechanism (**Fig. 2A**) in this work, which was 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). Our version of STDP-RL used a spike-time-dependent plasticity mechanism together with a reward or punishment signal for potentiation or depression of the targeted synapses. An exponentially decaying eligibility trace was included to assign temporally distal credits to the relevant synaptic connections. 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. 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.

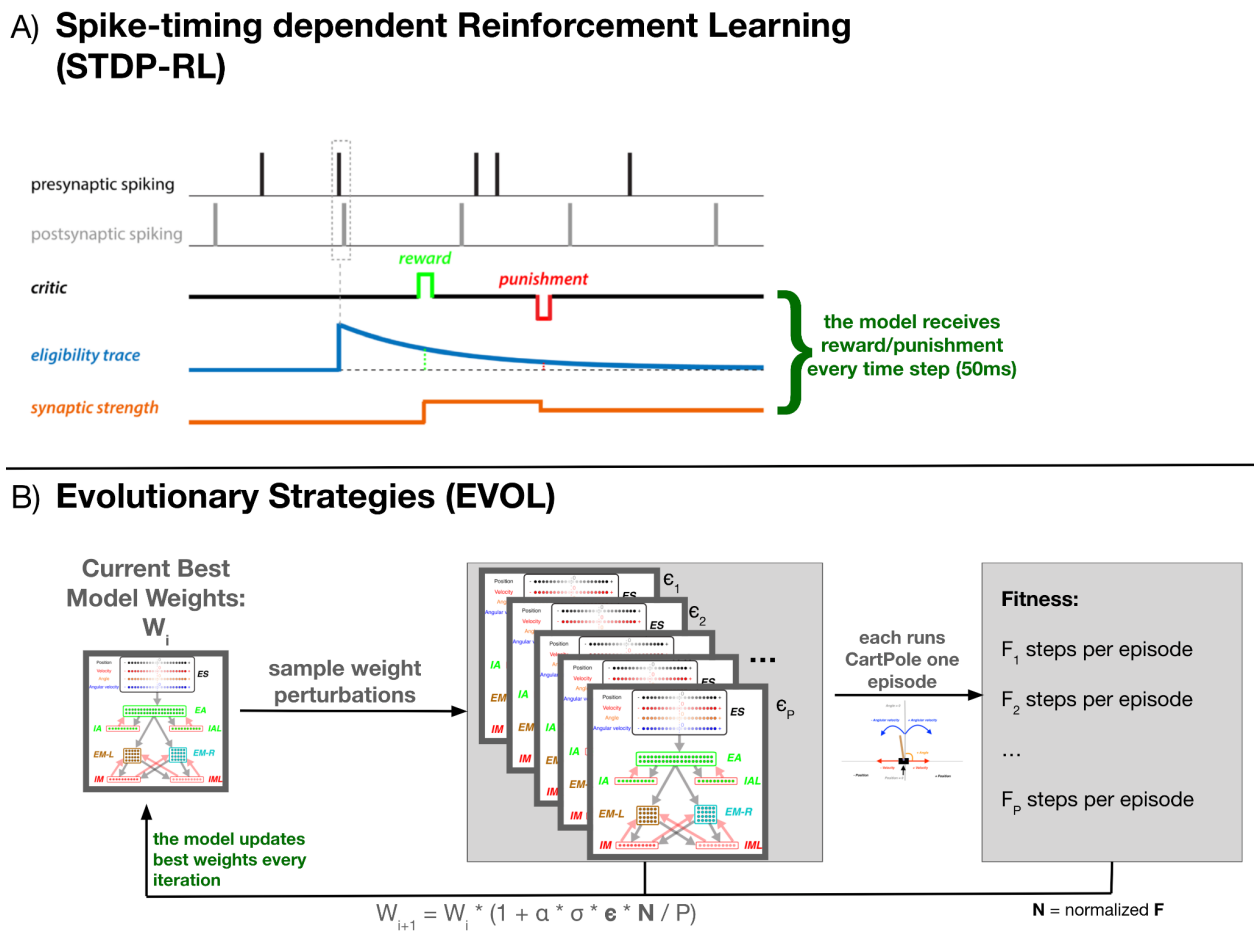


Figure 2: *Training SNN using STDP-RL and EVOL strategies. A) In STDP-RL, when a postsynaptic neuron produced a spike within a short-time interval of a presynaptic spike, the synapse between the pair of neurons was tagged with a decaying eligibility trace. The tagged synapse was strengthened or weakened proportional to the value of eligibility trace for a reward or punishment, respectively. B) Schematic showing the steps of EVOL. Note that when using the interleaved EVOL+STDP-RL algorithm, instead of using the performance of each model during an episode, performance is evaluated during STDP-RL training.*

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).

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: $[-\ max_reward,\ max_reward]$.

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

$$reward(t) = \begin{cases} 0 & if\ loss(t-1) < 10^{-2} \\ \frac{-max_reward}{\eta_{positivity}} & if\ no_move(t) \\ \frac{max_reward}{\eta_{positivity}} & if\ loss(t) < 10^{-2} \\ loss(t-1) - loss(t) & otherwise \end{cases} \qquad (2)$$

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

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

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.

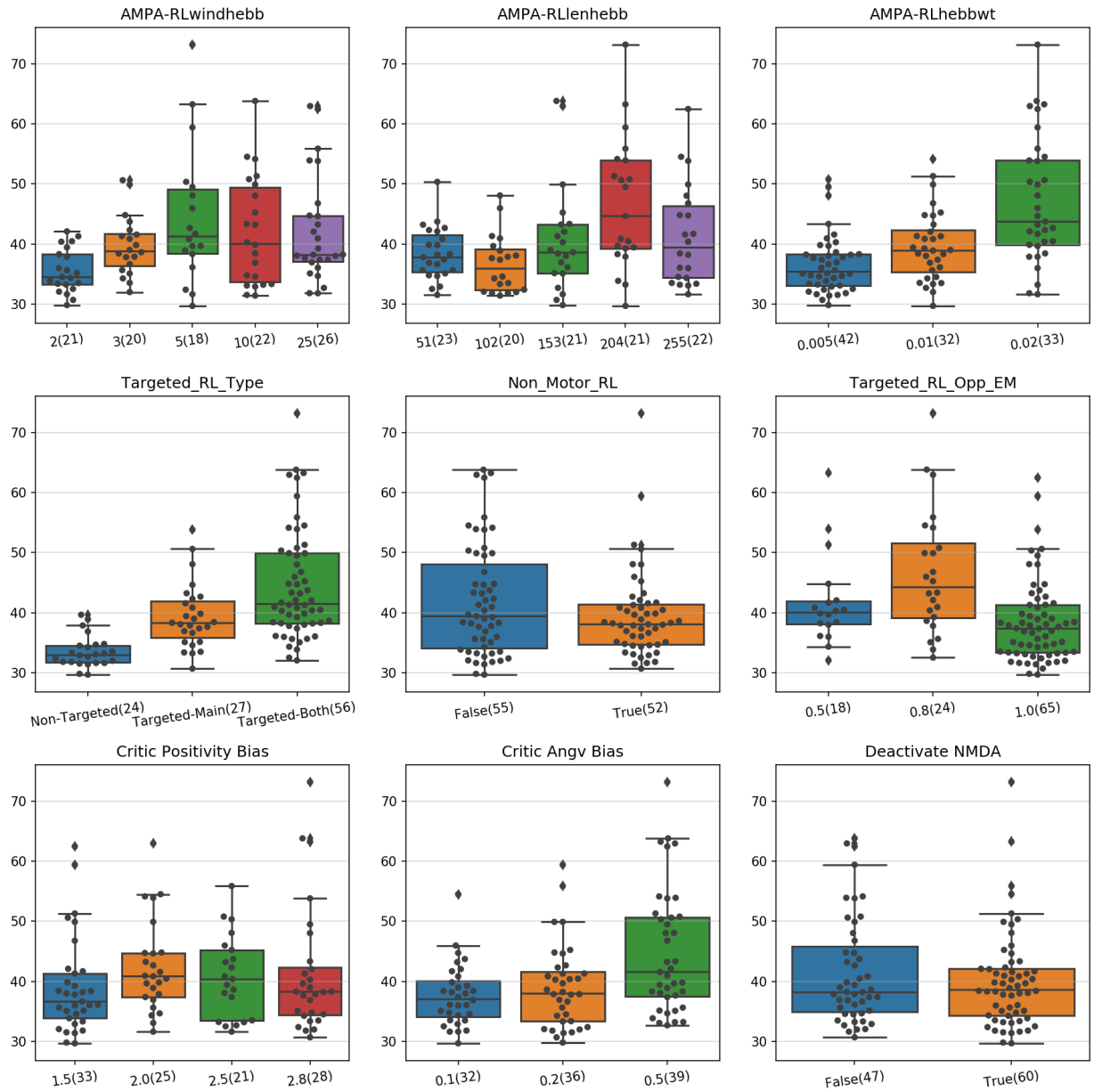


Figure 3: Performance distribution of the first hyperparameter search for training using STDP-RL (displaying averages over 100 episodes during training). *In each panel, the y-axis shows the performance, and the x-axis indicates the parameter values used in the evaluation. The number of models run with the specific parameter value is presented in parentheses, with 107 tested hyperparameter combinations.*

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 steps). The low performance indicated that these parameters might not be optimal for training. We next heuristically explored the effectiveness of 12 STDP-RL parameters in multiple steps. In

each step of hyperparameter search, we evaluated the performance of a set of STDP-RL parameters (see **Table 3**) by training the model using different combinations from the pool of STDP-RL parameters to balance the pole and looked at the distribution of performance against individual parameter values. We found that the learning performance was sensitive to some of the STDP-RL parameters values (**Fig. 3**). Comparing the averages and the scatter plot in **Fig. 3**, some parameter values clearly showed better performance (higher averages and smaller variance, e.g., *204ms* for AMPA-RLlenhebb, *Targeted-Both* for Targeted_RL_Type), whereas the others remained unclear.

In the next step of hyperparameter search, we fixed 4 parameter values and explored the effectiveness of the remaining 10 parameters. Additionally, in this second hyperparameter search, we tested different model initializations based on the six best models from the first run. We present the performance distribution of the second hyperparameter search step in **Supplementary Fig. 1.**

We resampled new parameter values around the best-found values of those parameters in the previous step and repeated simulations for 2000 sec. We present the performance distribution of the third hyperparameter search step in **Supplementary Fig. 2**. After three repeats, the model did not show any substantial improvement in performance. Therefore we chose all the STDP-RL values based on the first three hyperparameter search steps for training our SNN model.

Table 3: STDP-RL parameters obtained through hyperparameter search. *Values tested that are bolded are the hyperparameters picked for our model.*

Parameter	**Values tested**	**Description**
AMPA-RLwindhebb (ms)	(2, 3, 5, 10, **25**)[1],	Maximum time between presynaptic and postsynaptic spike times for considering plasticity.
AMPA-RLlenhebb (ms)	(51, 102, 153, 204, **255**)[1]	The decay time constant of the exponentially decreasing eligibility trace.
AMPA-RLhebbwt	(0.005, 0.01, **0.02**)[1], (0.005, 0.01, 0.015, **0.02**)[2], (0.015, 0.02, 0.025, **0.03**)[3]	Max synaptic weight adjustments based on reward or punishing signal.
Targeted_RL_Type	(non-targeted RL, targeted RL main, **targeted RL both**)[1]	The Targeted RL paradigm chosen
Non_Motor_RL	(**False**, True)[1], (False, **True**)[2]	Delivering RL to non EM population: EA
Targeted_RL_Main_EM	(0.5, **0.8**, 0.9, 1.0)[3]	Attenuation factor for the main subpopulation receiving reinforcement
Targeted_RL_Opp_EM	(0.5, 0.8, **1.0**)[1], (**0.8**,0.9)[2], (0.5, **0.8**, 0.9)[3]	Attenuation factor for the opposite subpopulation receiving reinforcement

Targeted_RL_Non_Motor	(0.2, 0.5, **0.8**, 1.0)[2] (**0.5**, 0.8, 0.9)[3]	Attenuation factor for the non motor subpopulation receiving reinforcement
Critic Positivity Bias	(**1.5**, 2.0, 2.5, 2.8)[1], (2.5, **2.8**, 3.0)[2], (2.5, **2.8**, 3.0)[3]	η_{angvel}: used in critic (**equation 1**)
Critic Angv Bias	(0.1, 0.2, **0.5**)[1], (0.5, 0.8, **1.0**, 2.0)[2], (**0.7**, 0.8, 0.9, 1.0)[3]	$\eta_{positivity}$: defined for critic evaluation (**equations 2-3**)
Critic Total Gain	(**2.5**, 3.0, 3.5)[2], (**2.5**, 3.0, 3.5)[3]	The final multiplier that increases the absolute value of the critic value. (**equation 4**)
Deactivate NMDA	(True, **False**)[1]	As an experiment, we tried to deactivate all non-plastic NMDA synapses.

Most of the hyperparameters had a minimal effect on the final training outcome for STDP-RL 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 with plasticity in the non-motor populations. These findings suggest that the plasticity of non-motor areas could enhance the learning ability of the model, consistent with earlier findings (Neymotin et al. 2013). Also, propagating the effects of rewards across multiple areas in the neuronal network in a nonuniform manner might be a more suitable choice for implementing reinforcement learning in spiking neuronal models. The best value for encoding the relative significance of the angular velocity compared with the deviation angle in reward prediction indicates the neural circuits underlying sensory-motor associations might preferentially encode some, but not all, sensory cues.

Evolutionary Strategies (EVOL)

The 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. 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 one episode; (4) measure the fitness (F_j) as the count of steps for that episode; (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 the EVOL model 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.

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. We note that for this algorithm, due to the heavy computational load, we used a population size of 5, which may account for some of the differences from the baseline EVOL algorithm. However, when using a population size of 10 for this interleaved algorithm, 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 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. 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.

Software

All modeling and analysis source code will be made available on https://github.com/NathanKlineInstitute upon publication.

Evaluations

t-distributed Stochastic Neighbor Embeddings

For generating **Fig. 12**, we evaluated the four models (before training and after training with STDP-RL, EVOL, EVOL+STDP-RL) on the same 105 initial game states. From the neural spike train, we divided each timestep into *B* time windows and measured if a neuron was activated at least once during the time window. From the resulting spike-train matrix, we extracted non-linear embeddings for each time window using t-distributed Stochastic Neighbor Embeddings (van der Maaten 2008). We used a time window of 12.5 ms (timestep / 4) and perplexity of 50.

Results

Training the SNN model to play CartPole

We trained our SNN model using initial synaptic weights and STDP-RL parameters obtained from hyperparameter search (see highlighted values in **Table 3**). At each game-step, we computed the amount of reward by comparing the deviation angle of the pole from the balanced position and angular velocity during the previous game-step with those values during the current game-step. If the pole moved towards (away from) the balanced position, a positive (negative) reward was delivered (see Methods for the details). The maximum reward was delivered when the pole was in a near-perfect balanced position. The number of steps before the pole irreversibly lost its balance marked the end of an episode and characterized the model's performance. Initially, the performance of the model for individual episodes fluctuated around 20 steps per episode on average. Throughout training, the performance slowly increased from 20 to around 400 episodes per episode (**Fig. 4A**), with an average reaching 190 steps per episode. We stopped training once we observed no further increase in the calculated median performance, around 180 steps per episode. It took 950 episodes of training for our SNN model with the STDP-RL mechanism to reach a median performance level of 180 steps per episode.

On the other hand, tuning synaptic weights using EVOL (**Fig. 2B**) took almost 12 times more training episodes to reach a performance level of 180 steps per episode (**Fig. 4B**) compared to STDP-RL. Part of this discrepancy in episodic efficiency is due to the reward structure. While STDP-RL training was provided with dense rewards from a hand-tuned critic,

the EVOL training only used episodic fitness, a much sparser form of reinforcement. It should also be noted that time used in the hyperparameter search has not been taken into account while comparing the computational efficiency of these two training approaches.

Learning using the EVOL+STDP-RL strategy took longer to reach the 180-200 performance level(**Fig. 4C**), given that there was additional STDP-RL learning and evaluation on each iteration of EVOL. After reaching the 180-200 performance levels, the EVOL+STDP-RL model progressed rapidly, reaching an initial performance peak around the 400 level. From then on, performance fluctuated and finally stabilized around the 400 level. However, there were some subsequent transient dips in performance, which then returned to ~400 levels. Note that in **Fig. 4C,** blue and orange dots represent the model performance before and after STDP-RL, respectively. The overall pattern for both of these tends to follow each other, though with larger variability for the before STDP-RL step. Notably, the EVOL+STDP-RL model showed the best overall performance compared to the other training algorithms.

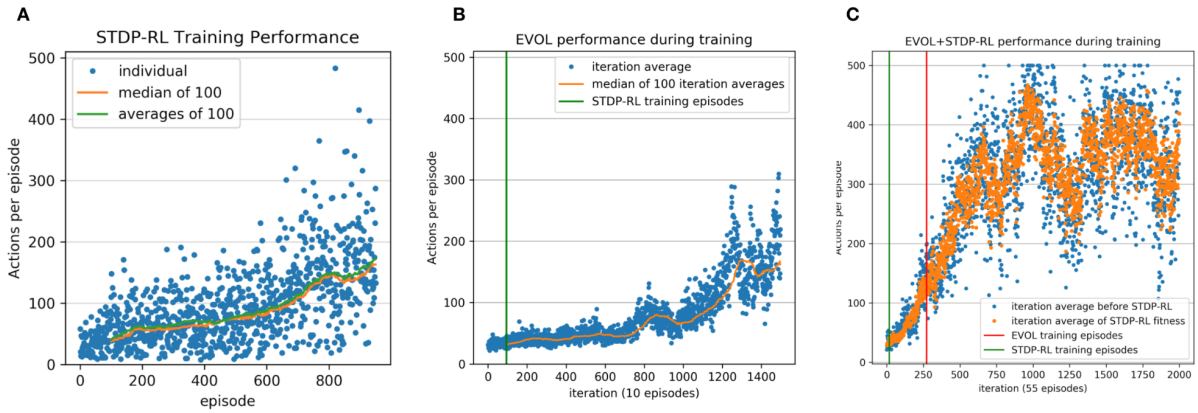


Figure 4: Performance of the SNN model increased with the training duration. *STDP-RL model **(A)** learned to perform faster than the EVOL model **(B)** and the EVOL+STDP-RL model **(C)**. In **(A),** each blue dot represents a training episode, while in **(B)** and **(C)**, the blue dot represents an average over the population. Each training episode began with a randomly reset environment/game-state. A moving window median (orange in **(A)** and **(B)**) over 100 episodes was used to analyze the stabilized performance values. The green vertical line in (**B), (C)** indicates the number of training episodes using STDP-RL. Similarly, the red vertical line in **(C)** indicates the number of training episodes using EVOL. Finally, in **(C),** blue, orange colors indicate model performance before and after STDP-RL.*

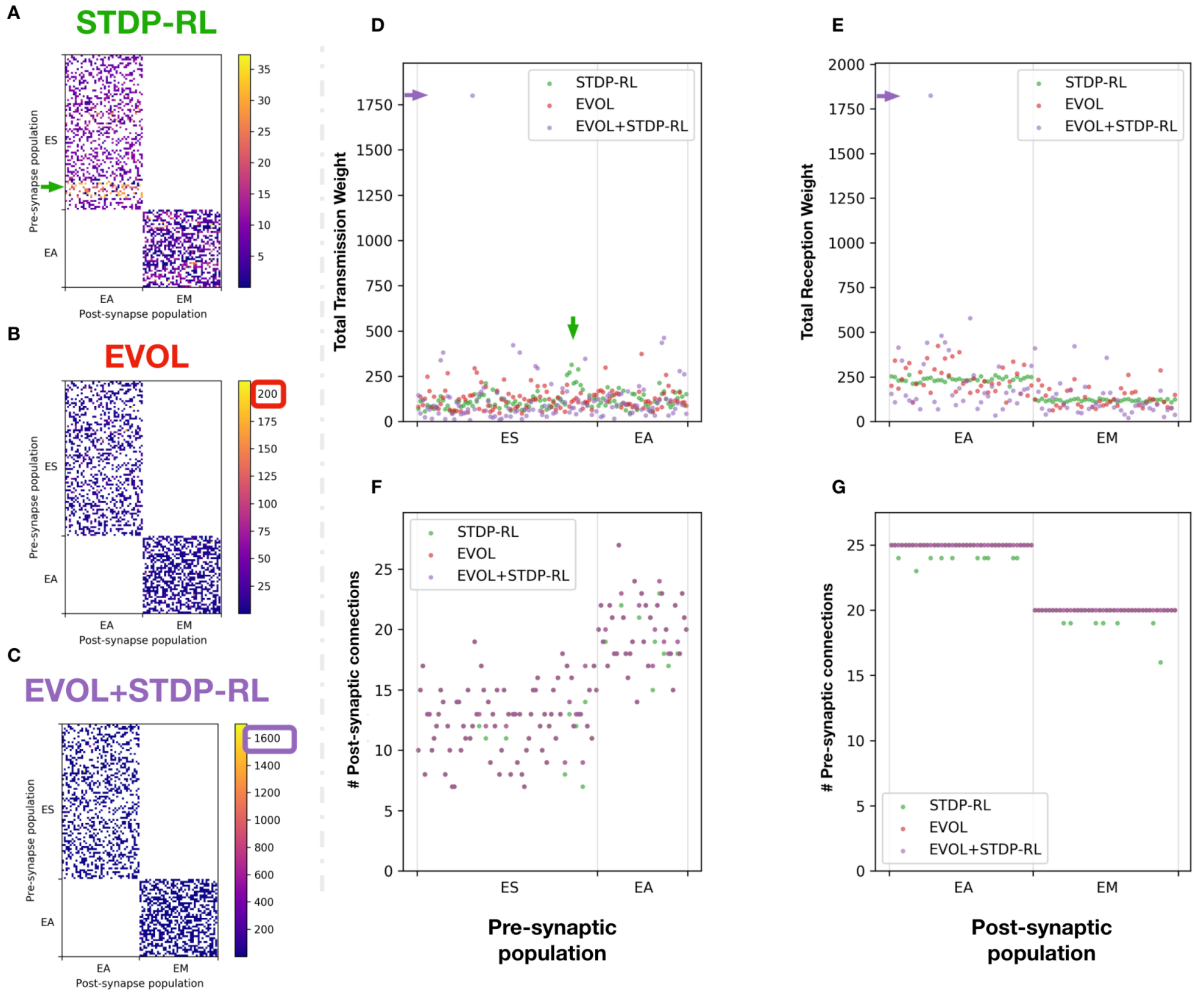


Figure 5: STDP-RL, EVOL, and EVOL+STDP-RL differentially modulate the synaptic weights of the SNN model during training. *Adjacency matrix showing weights of synaptic connections between ES-EA and EA-EM populations in STDP-RL model* ***(A)****, EVOL model* ***(B),*** *and EVOL+STDP-RL model* ***(C)****.* ***(D)*** *Total Transmission Weight: Sum of weights of synaptic connections formed by each presynaptic neuron onto multiple postsynaptic neurons.* ***(E)*** *Total Transmission Weight: Sum of weights of synaptic connections formed onto each postsynaptic neuron by multiple postsynaptic neurons.* ***(F)*** *The number of synaptic connections made by each presynaptic neuron* ***(G)*** *The number of synaptic connections received by each postsynaptic neuron. Note that purple dots from the EVOL+STDP-RL model obscure red dots from the EVOL model in* ***(F),(G)****.*

To compare the synaptic weights after training using STDP-RL, EVOL, and EVOL+STDP-RL strategies, we plotted adjacency matrices of synaptic weights from all three models (**Fig. 5**). After training using STDP-RL, the relative strength of synaptic connections between ES and EA neurons increased non-uniformly with high values for some EA neurons encoding angular velocity (marked by green arrows in **Fig. 5A**). The synaptic weights in the EVOL and EVOL+STDP-RL models were more sparsely distributed and with higher variability. Relatively smaller weights in STDP-RL were obtained because of the normalization mechanism

used in the model to keep the network output balanced. For the EVOL+STDP-RL model, we observe a synapse between ES and EA neuronal populations that increased to abnormal proportions (purple arrows in **Fig. 5D, E**).

We then compared the total transmission weights (**Fig. 5D**: sum of weights of synaptic connections made by each presynaptic neuron onto multiple postsynaptic neurons) and total reception weights (**Fig. 5E**: sum of weights of synaptic connections made onto each postsynaptic neuron by multiple presynaptic neurons). Although all models contained high transmitting neurons, those neurons were dispersed across ES and EA populations in the EVOL (**Fig. 5B**) and EVOL+STDP-RL (**Fig. 5C**) models in contrast to clustered ES neurons in the STDP-RL model (**Fig. 5D**). Because of the synaptic scaling used to maintain the network activity levels, despite the modifications in synaptic weights, the reception weights remained roughly constant for each postsynaptic neuronal population in the STDP-RL model (green dots in **Fig. 5E**) yet more variable for EVOL and EVOL+STDP-RL models (red and purple dots in **Fig. 5E**). These effects were not mediated through increased connections, as shown in **Fig. 5F, G** because there are no mechanisms that create new synapses. Instead, we see the STDP-RL mechanism removing connections that decrease below a threshold.

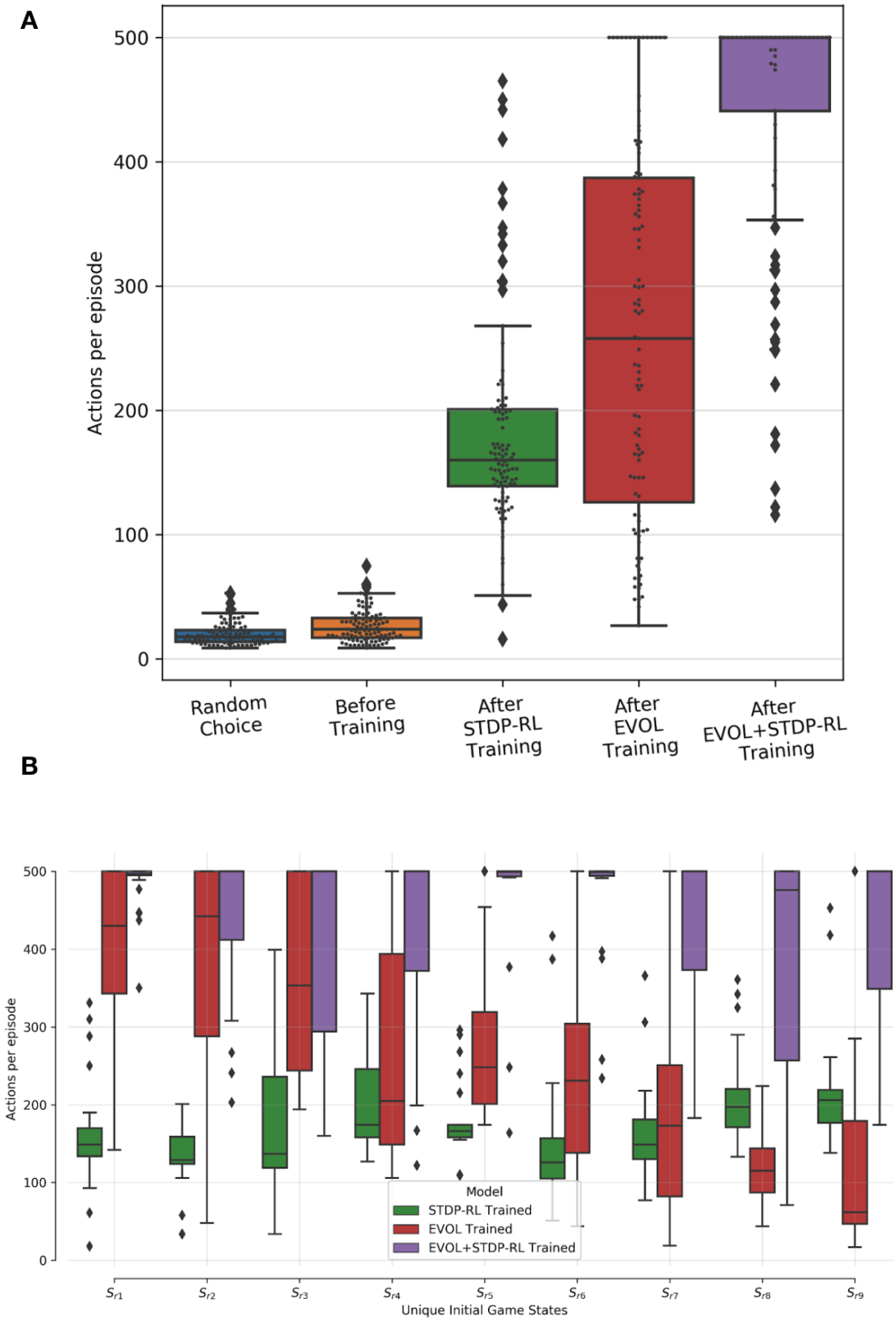

A
Actions per episode
500
400
300
200
100
0
Random Choice
Before Training
After STDP-RL Training
After EVOL Training
After EVOL+STDP-RL Training
B
Actions per episode
Model
STDP-RL Trained
EVOL Trained
EVOL+STDP-RL Trained
S_{r1}
S_{r2}
S_{r3}
S_{r4}
S_{r5}
S_{r6}
S_{r7}
S_{r8}
S_{r9}
Unique Initial Game States

Figure 6: SNN models learned to play CartPole, with similar maximum performance using STDP-RL, EVOL, and EVOL+STDP-RL strategies. (A) *Comparing performance distributions of the trained models with randomly generated actions and the untrained model. Each model's performance was evaluated over 105 episodes.* ***(B)*** *Simulations for nine different episodes (Sr1-Sr9) were repeated 25 times each, using STDP-RL, EVOL, and EVOL+STDP-RL models. Note that each episode had a unique initial game-state which was identical across repeats of the same episode.*

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

To demonstrate that the model learned the behavior, we compared the performance of STDP-RL, EVOL and EVOL+STDP-RL trained models with the random action generating and untrained models (**Fig. 6**). All three trained models' performance was evaluated using the fixed synaptic weights at the end of the respective training sessions. For this, we let each model play CartPole for 105 episodes where at the end of each episode, only the game-state was reset to a different initial condition. Before the training, the model performance was similar to the model generating random actions (**Fig. 6A**). However, after training, STDP-RL, EVOL, and EVOL+STDP-RL models learned to choose actions that allowed them to keep the pole balanced for longer durations (**Fig. 6A**). As expected from the rapidly increased performance during training, the EVOL+STDP-RL model performed better than both STDP-RL and EVOL models. Although comparing the medians showed marginally better performance of EVOL model than STDP-RL model, it was more variable (compare the InterQuartile Intervals of performance distributions of trained models in **Fig. 6A**).

Similarly, the EVOL+STDP-RL model also showed considerable variability in the performance (see performance data points in the purple bar graph in **Fig. 6A**). The variability in the performance might be related to 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. 6B**; 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). As indicated earlier in **Fig. 6A**, the STDP-RL model showed lower but comparable performances to EVOL for different initial game-states. In contrast, the EVOL model clearly showed much better performance for some initial game-states and lower performance for others with persistently larger variance for all game-states. On the other hand, the EVOL+STDP-RL model always showed better and robust performance with less variability across repeated game initiations.

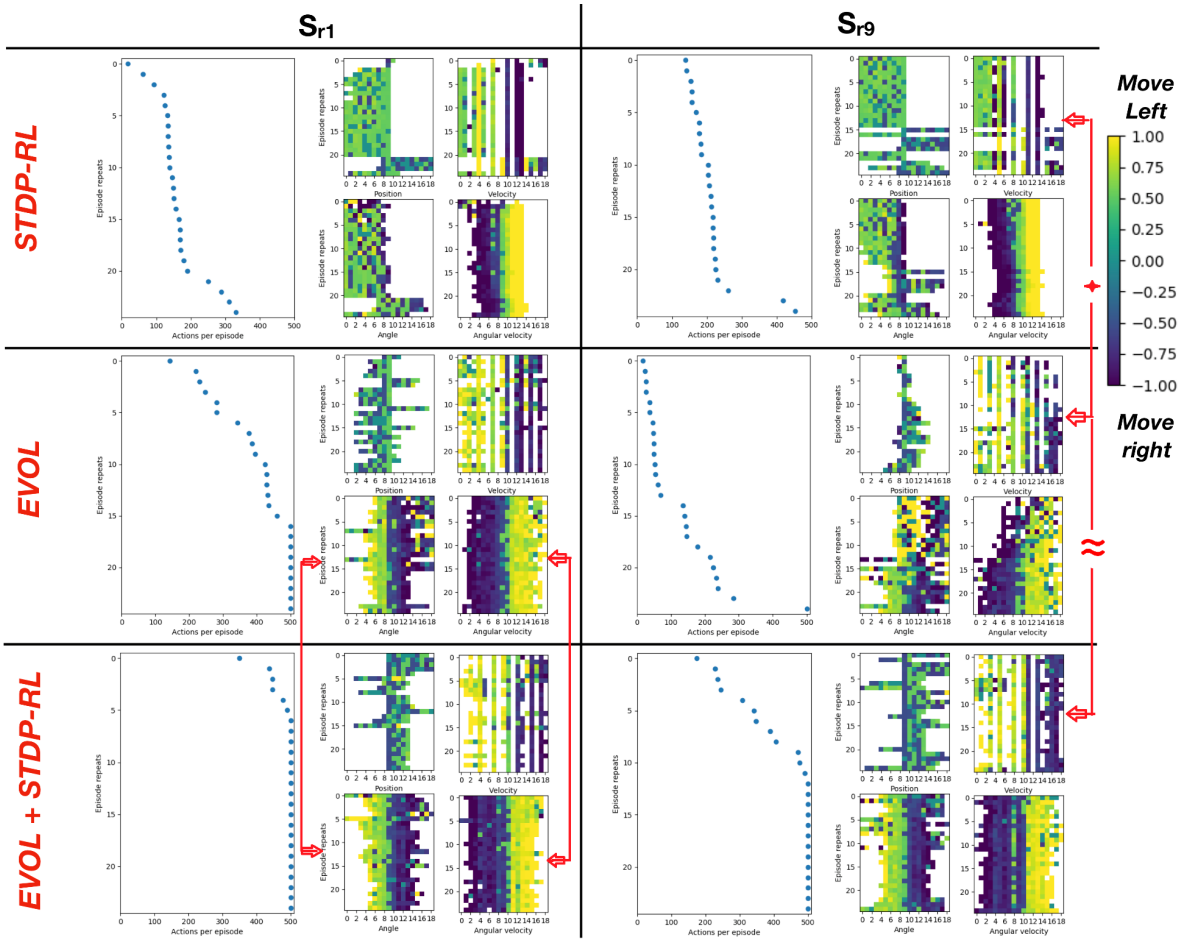


Figure 7: Learning produces specific preferences for right vs left actions, depending on sensory inputs. *These sensory-motor activation maps show the sensory-input parameters parsed during repeated episodes of Sr1 and Sr9 together with the probability of dominant action executed for each activated sensory-input 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.*

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), explaining the smaller observed variance in performance using both models as shown in **Fig. 6A**. 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.

Comparing the learned SNN circuit dynamics.

Despite the similar best performance of the models (**Fig. 6**), 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**).

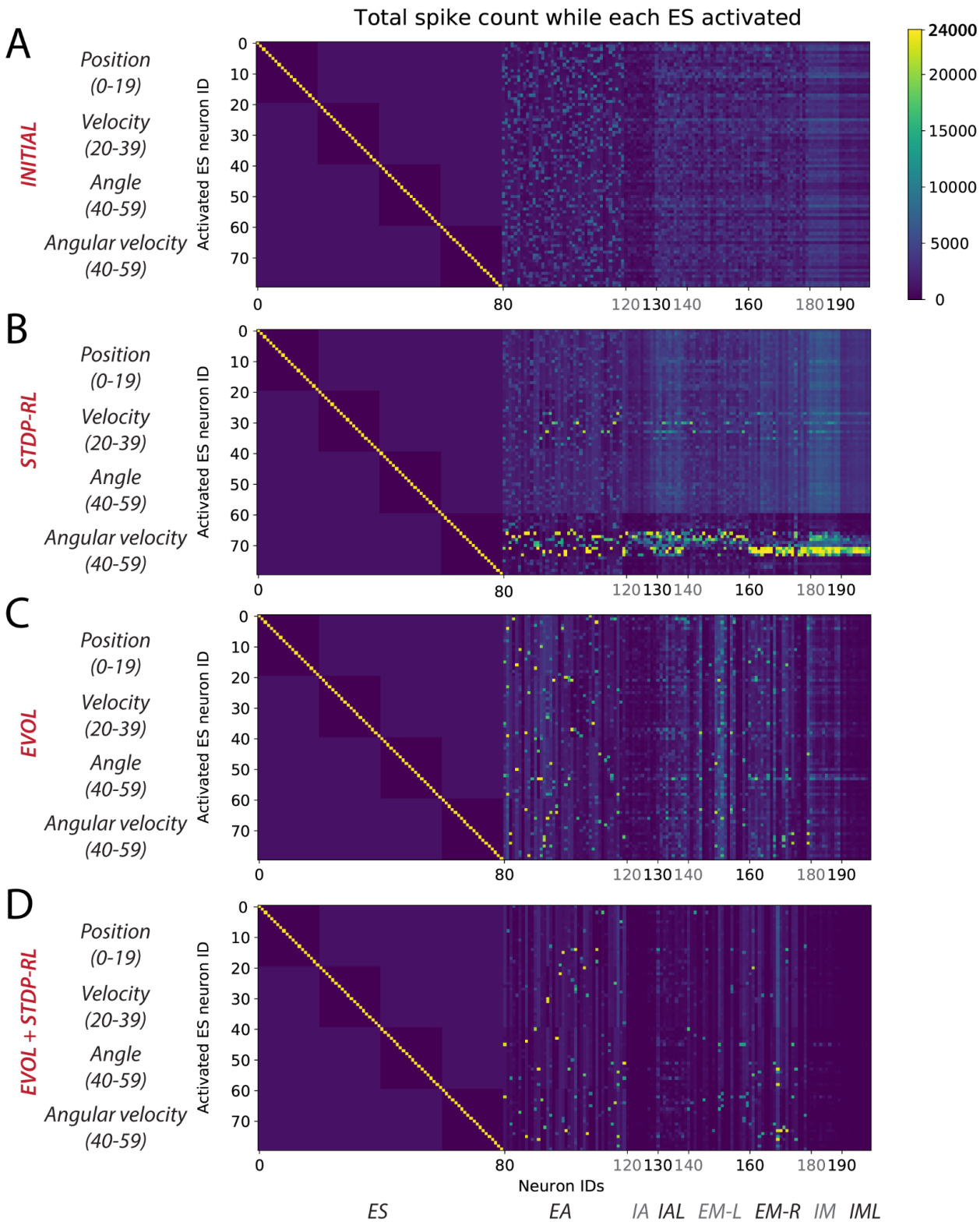


Figure 8. Learning induces changes in network responses to specific sensory inputs, with distinct representations produced by the different training algorithms. *The plot represents the number of spikes generated by each neuron in the model in response to stimulating sensory input neurons in ES. Each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

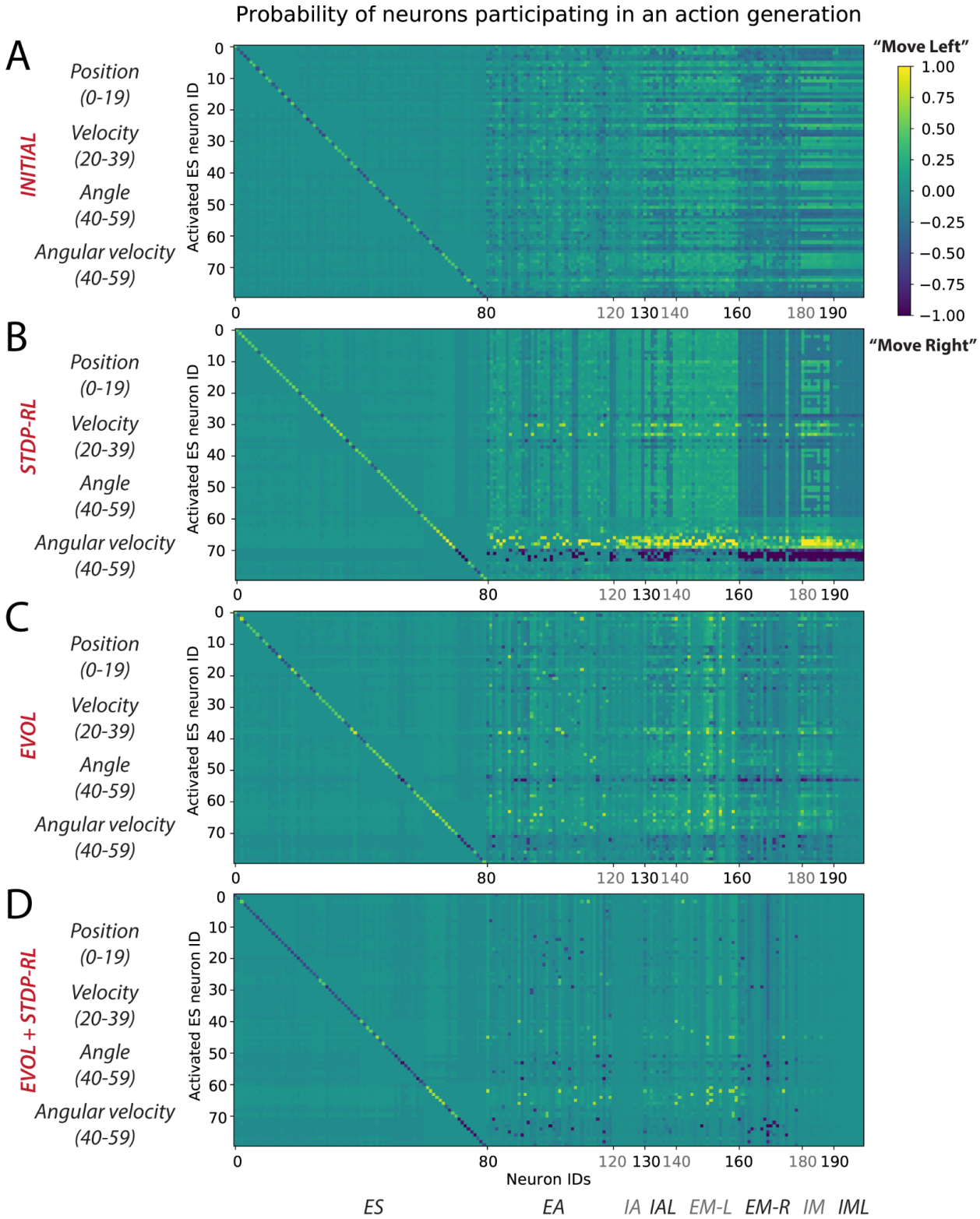


Figure 9: Sensory-Motor mappings show differential participation of neurons in the network in action generation. *In response to a particular sensory input, each neuron was accounted for Move-left or Move-right action and then labeled with the probability of frequent action.* ***A)*** *INITIAL,* ***B)*** *STDP-RL,* ***C)*** *EVOL, and* ***D)*** *EVOL+STDP-RL*

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 probabilities of more frequent actions associated with each pair of sensory-input and model neurons 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**.

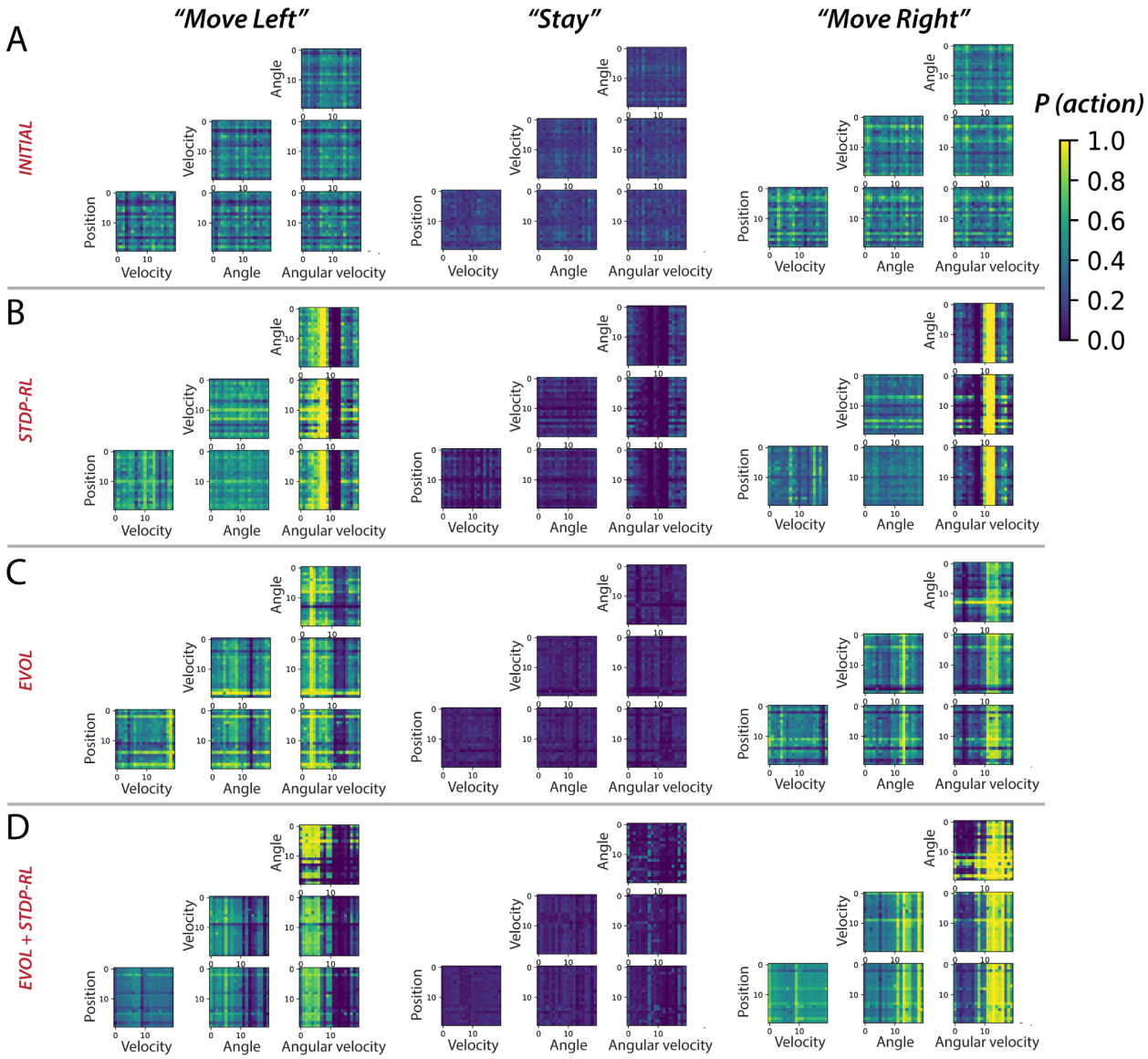


Figure 10: Sensory-Motor mappings show different behaviorally-relevant input-action associations learned using STDP-RL, EVOL, and EVOL+STDP-RL strategies. *Probability of "Move Left" (left panel), "Stay" and "Move Right" for all pairs of sensory input parameters for the model before training* ***(A)****,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

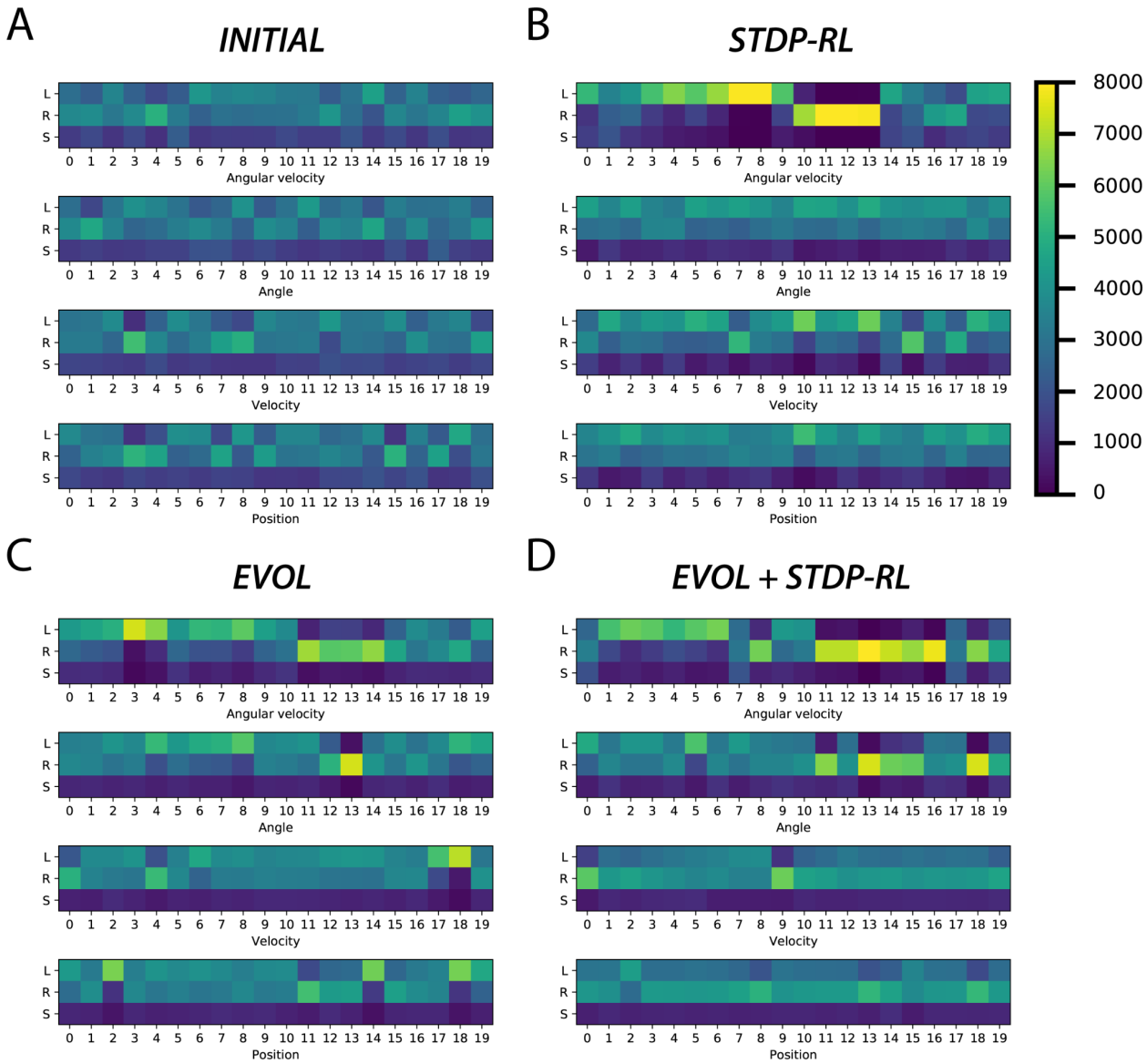


Figure 11: *Occurrence of Left-, Right- and Stay-actions for all receptive fields associated with the position, velocity, angle, and angular velocity. Heatmaps are shown to highlight the differences between the INITIAL model* ***(A)*** *before training and models trained using STDP-RL* ***(B)****, EVOL*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

Finally, we distilled the whole-network activations during evaluation into a 2-dimensional embedding using t-distributed Stochastic Neighbor Embeddings (**Fig. 12**). Each dot, representing a whole-network activation pattern during a time window, is colored by the action performed at the end of the timestep (see **Materials and Methods**). As expected, the structure of the embeddings before training reveals no discernable structure (**Fig. 12A**). The activations of the trained model with STDP-RL (**Fig. 12B**) group into discernable clusters of left or right moves signifying that all neurons are involved in action generation. In contrast, the EVOL and

EVOL+STDP-RL training strategies (**Fig. 12C-D**) produce activations that are less distinguishable between left and right moves, consistent with more distributed representations and decision-making produced by EVOL and EVOL+STDP-RL.

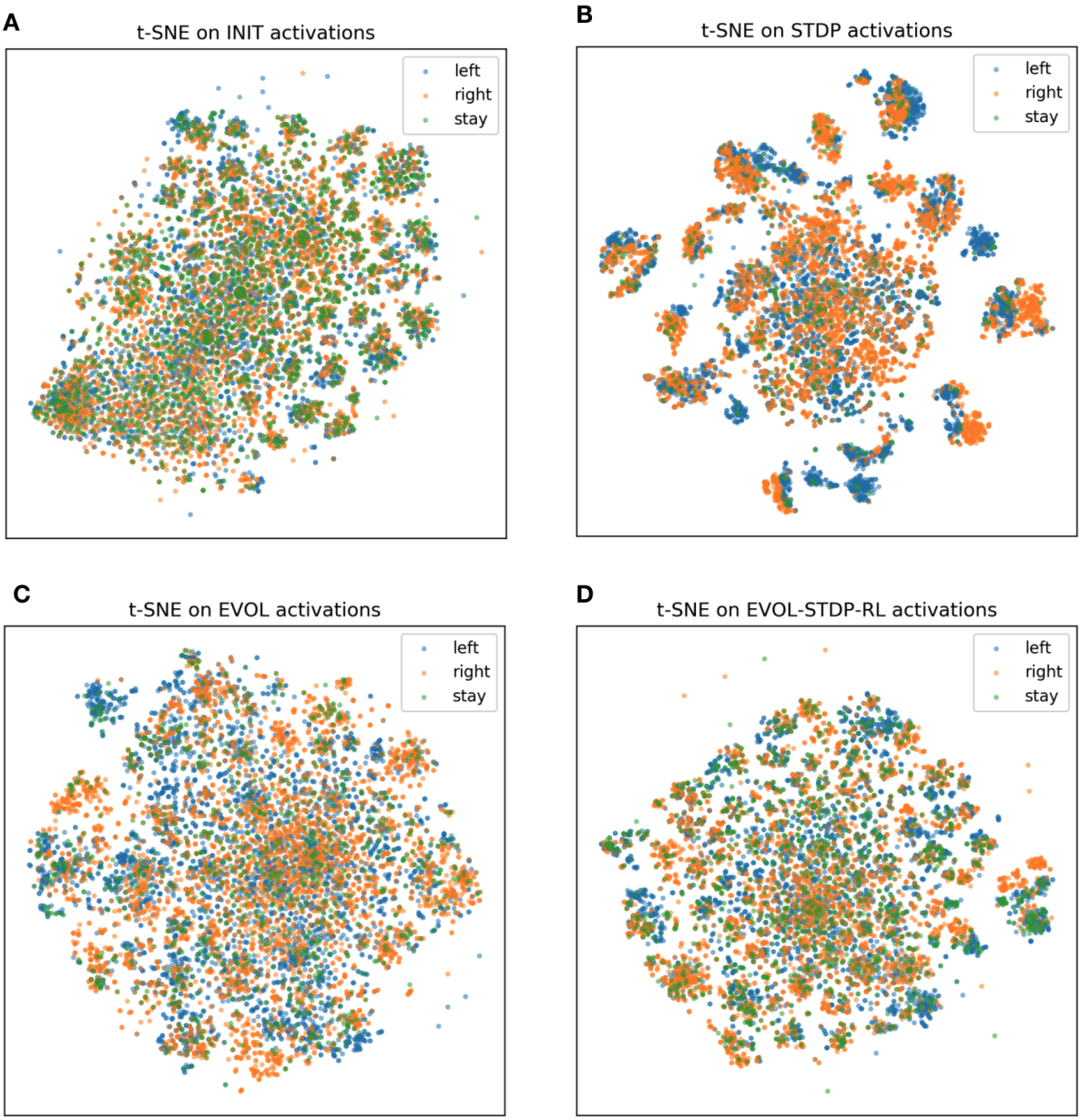


***Figure 12**: **Whole-Network neuronal activations during evaluation shows clustering behavior for left and right moves. A:** t-SNE embeddings of activations during the evaluation of model before training. **B:** t-SNE embeddings of evaluation activations after the model was trained with the STDP-RL algorithm. **C:** t-SNE embeddings of evaluation activations after the model was trained with Evolutionary Strategies Algorithm. **D:** t-SNE embeddings of evaluation activations after the model was trained with EVOL+STDP-RL algorithm (see methods).*

Discussion

Despite many efforts in building biophysically detailed (Markram et al. 2015; Markram 2006) and spiking neuronal network (SNN) models, very little progress has been made in training these types of models to perform behaviors in dynamic environments and understanding the underlying circuit mechanisms. In this work, we first developed an SNN and then trained it 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 biologically-relevant 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. 6A**), we demonstrated that all of our strategies could be used for training our SNNs to play CartPole, and therefore should be effective in training for other more complicated behaviors/tasks.

Although all three algorithms were effective on their own, there were noticeable differences in variability of performance (**Fig. 6,7**), network connectivity patterns (**Fig. 5**), and resulting neuronal dynamics and behavior (**Fig. 8-12**). While the peak performance of the STDP-RL model converged faster than the EVOL model (**Fig. 4**), if we take into account the time consumed for the three hyperparameter search steps of 500, 2000, and 2000 seconds, both strategies took similar training time to converge. Therefore, depending on the number of hyperparameters and total synaptic connections, either strategy could dominate the other in terms of required computational resources. Although EVOL showed on average better performance (**Fig. 6A**), it remained unreliable for some initial game-states (**Fig. 6B**). On the other hand, STDP-RL was more reliable with less variance across initial game-states (**Fig. 6B**). These latter differences might be due to a more sparsely distributed representation seen after EVOL training, caused by less temporally local feedback provided. Interestingly, the hybrid EVOL+STDP-RL algorithm overcame some of the limitations of either approach alone, displaying both the excellent performance in CartPole seen through EVOL, with the additional stability and robustness to different initializations seen using STDP-RL (**Fig. 6A**).

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 our approach, 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 (**Flg. 7**).

Many of the earlier SNN models trained to perform simple behaviors such as extending an arm to a target location (Neymotin et al. 2013) and foraging tasks (Sanda, Skorheim, and Bazhenov 2017), start learning from a single network configuration. However, there are benefits to training from a set of multiple initial network configurations, achieved through varying synaptic connection weights or network architectures (Stanley and Miikkulainen 2002), and testing the populations' performance. The best performance can then be used to derive a final configuration for longer-term training. However, this practice cannot guarantee that the configuration will produce the best longer-term training since longer-term dynamics and learning are impossible to predict. This could then lead to developing an underperforming model. Such an approach also leads us to a single starting configuration, or even when multiple configurations are used, those configurations can be very similar. We want to 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 (see **Table 3**). 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 models with different network configurations with diverse synaptic weights. This interleaved EVOL+STDP-RL algorithm was inspired by the obvious fact that in addition to the 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 even better from experience during the model's *lifetime*. Therefore, we chose to evaluate models that learned using STDP-RL since it is a biologically inspired learning rule hypothesized to operate during an animal's lifetime, directly impacting the synaptic weights. 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 were not directly transferred to succeeding generations. However, there is evidence that epigenetic modifications during an animal's lifetime could enable changes to the genotype to transfer to offspring (Harper 2005). Future extensions to our algorithm could allow this type of direct transfer of STDP-RL weights across generations. We also plan to explore other adjustments, including optimizing network architecture: numbers of neurons, layers, and structural plasticity (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.

Although we made use of two previously used algorithms, including evolutionary optimization and STDP-RL, we both extended the context in which they were applied and also enhanced the algorithm's effectiveness using hyperparameter search and designing the new EVOL+STDP-RL algorithm. Overall, our algorithms demonstrated effectiveness in training SNNs to perform behaviors in a dynamic environment. 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 high computational costs associated with traversing the high dimensional search 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 also previously used the STDP-RL learning rules to train a visual/motor cortex model to play a variant of the Pong game (Anwar et al., n.d.). That model required additional complexity in the visual scene representations (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. Indeed, 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. This development would not have been possible using the larger visual/motor cortex model used previously. 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 to artificial neural networks

Over the past decade, significant progress has been made towards developing artificial neural network (ANNs) architectures and algorithms for training and performing behaviors. A well-known example is using deep reinforcement learning algorithms, such as the deep Q network algorithm, to train ANNs to play Atari games (V. Mnih et al. 2013; Volodymyr Mnih et al. 2015). These algorithms were able to achieve human and even superhuman performance levels on some of the Atari games. More recently, this line of research was extended to solving more complex tasks like the game of Go (Silver et al. 2016), although the strategy relied on non-biological data structures (*trees*) and search procedures for high-performing solutions. While external data structures are not a drawback from an engineering perspective, they could 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 these SNN models is an effort to test the hypothesis that brain-like architectures and biological learning mechanisms can be computationally and energy-efficient despite *imperfect* performance. Clearly, the primate brain is proof of energy-efficient computation, although it, of course, depends on its long history of evolution, which incurred a high cost. Recent studies show that the human neocortex, in particular, may be substantially more energy-efficient than in other mammals (Beaulieu-Laroche et al. 2021). With all this, we note that our SNN models trained to play CartPole already display excellent performance. In the future, we plan to generalize our work by training our SNN models on multiple and more complex tasks, using the insights we have gained from the algorithms developed in this work.

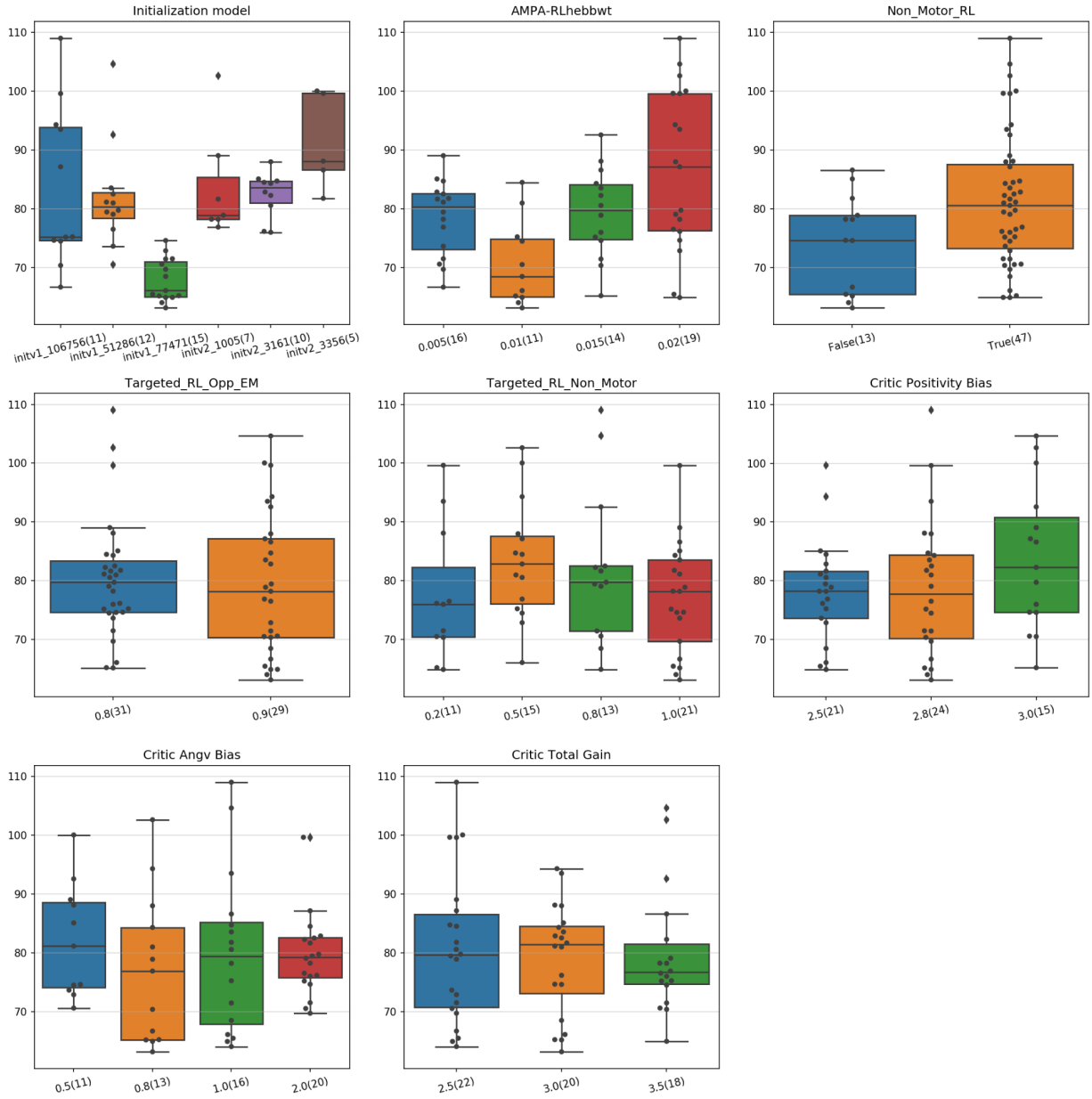
Acknowledgments

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.

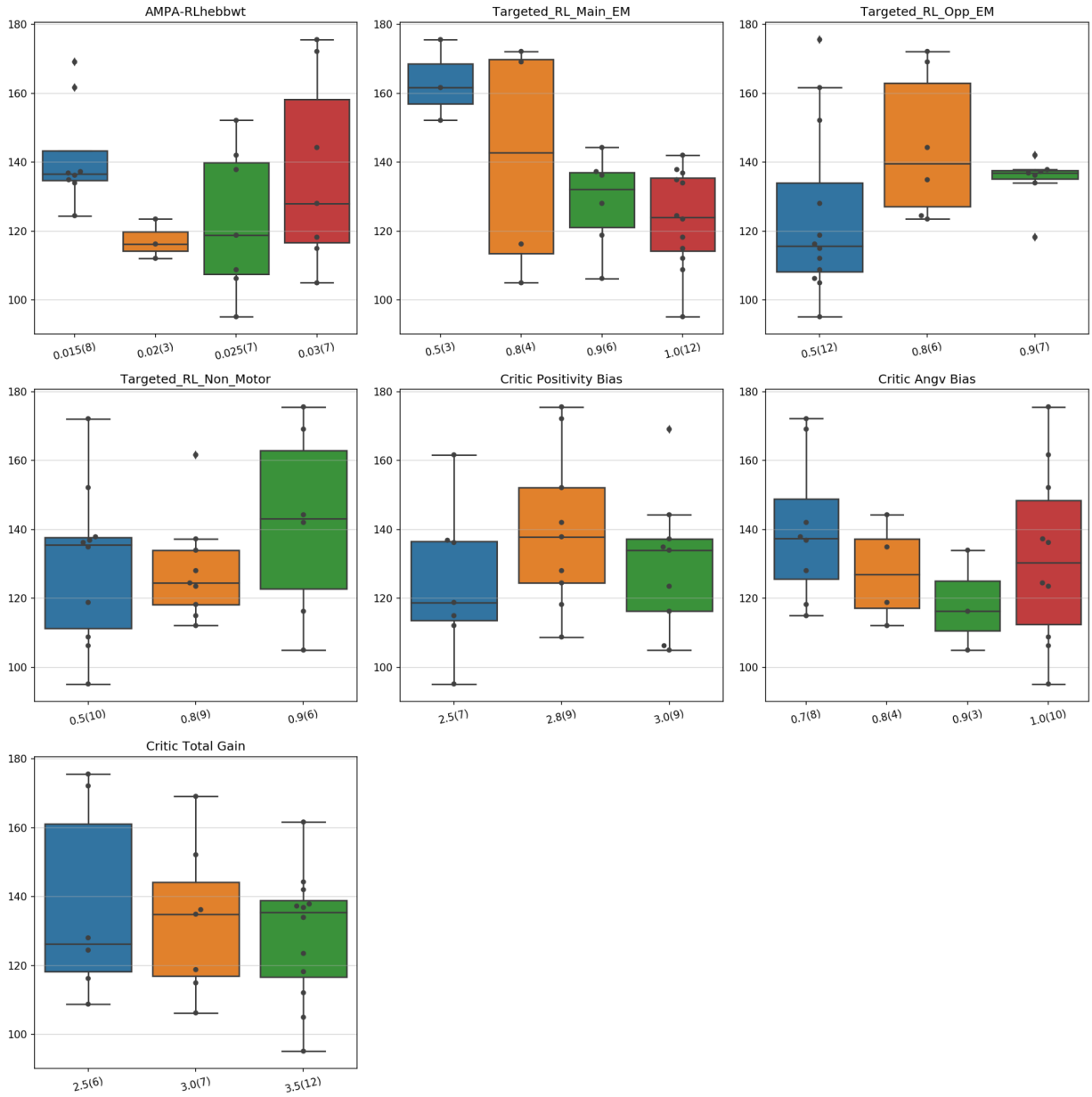
Contribution to the field statement

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.

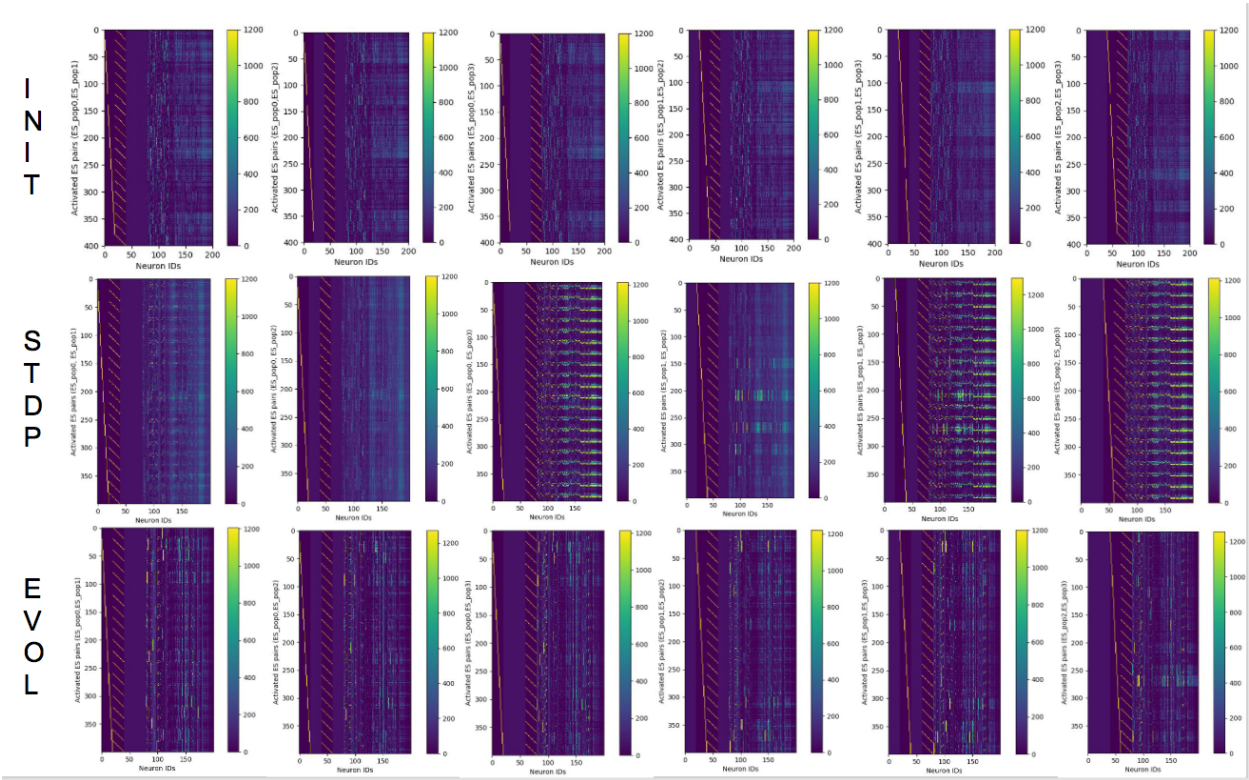
Supplementary Figures



Supplementary Figure 1: *Performance distribution of hyperparameters for training using STDP-RL evaluated in the second step (using averages over 100 episodes).*



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



Supplementary Figure 3: *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).*

References

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.

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.

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.

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

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.

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.

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.

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

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.

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

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.
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.
Maaten, Laurens van der. 2008. “Visualizing Data Using T-SNE.” 2008. https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf?fbclid=IwA.
Markram, Henry. 2006. “The Blue Brain Project.” *Nature Reviews. Neuroscience* 7 (2): 153–60.
Markram, Henry, Eilif Muller, Srikanth Ramaswamy, Michael W. Reimann, Marwan Abdellah, Carlos Aguado Sanchez, Anastasia Ailamaki, et al. 2015. “Reconstruction and Simulation of Neocortical Microcircuitry.” *Cell* 163 (2): 456–92.
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.
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., 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.
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.
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.
Whitelam, Stephen, Viktor Selin, Sang-Won Park, and Isaac Tamblyn. 2021. "Correspondence between Neuroevolution and Gradient Descent." *Nature Communications* 12 (1): 6317.
Zador, Anthony M. 2019. "A Critique of Pure Learning and What Artificial Neural Networks Can Learn from Animal Brains." *Nature Communications* 10 (1): 3770.