



OPEN ACCESS

EDITED BY

Sheri Marina Markose,
University of Essex, United Kingdom

REVIEWED BY

Stefania Costantini,
University of L'Aquila, Italy
Neil Vaughan,
University of Exeter, United Kingdom

*CORRESPONDENCE

Sidney Pontes-Filho,
sidneyp@oslomet.no

SPECIALTY SECTION

This article was submitted to
Computational Intelligence in Robotics,
a section of the journal
Frontiers in Robotics and AI

RECEIVED 30 July 2022

ACCEPTED 03 October 2022

PUBLISHED 14 October 2022

CITATION

Pontes-Filho S, Olsen K, Yazidi A,
Riegler MA, Halvorsen P and Nichele S
(2022), Towards the Neuroevolution of
Low-level artificial general intelligence.
Front. Robot. AI 9:1007547.
doi: 10.3389/frobt.2022.1007547

COPYRIGHT

© 2022 Pontes-Filho, Olsen, Yazidi,
Riegler, Halvorsen and Nichele. This is
an open-access article distributed
under the terms of the [Creative
Commons Attribution License \(CC BY\)](#).
The use, distribution or reproduction in
other forums is permitted, provided the
original author(s) and the copyright
owner(s) are credited and that the
original publication in this journal is
cited, in accordance with accepted
academic practice. No use, distribution
or reproduction is permitted which does
not comply with these terms.

Towards the Neuroevolution of Low-level artificial general intelligence

Sidney Pontes-Filho^{1,2*}, Kristoffer Olsen³, Anis Yazidi^{1,4,5},
Michael A. Riegler^{6,7}, Pål Halvorsen^{1,6} and Stefano Nichele^{1,4,5,6,8}

¹Department of Computer Science, Oslo Metropolitan University, Oslo, Norway, ²Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway, ³Department of Informatics, University of Oslo, Oslo, Norway, ⁴AI Lab—OsloMet Artificial Intelligence Lab, Oslo, Norway, ⁵NordSTAR—Nordic Center for Sustainable and Trustworthy AI Research, Oslo, Norway, ⁶Department of Holistic Systems, Simula Metropolitan Centre for Digital Engineering, Oslo, Norway, ⁷Department of Computer Science, UiT the Arctic University of Norway, Tromsø, Norway, ⁸Department of Computer Science and Communication, Østfold University College, Halden, Norway

In this work, we argue that the search for Artificial General Intelligence should start from a much lower level than human-level intelligence. The circumstances of intelligent behavior in nature resulted from an organism interacting with its surrounding environment, which could change over time and exert pressure on the organism to allow for learning of new behaviors or environment models. Our hypothesis is that learning occurs through interpreting sensory feedback when an agent acts in an environment. For that to happen, a body and a reactive environment are needed. We evaluate a method to evolve a biologically-inspired artificial neural network that learns from environment reactions named Neuroevolution of Artificial General Intelligence, a framework for low-level artificial general intelligence. This method allows the evolutionary complexification of a randomly-initialized spiking neural network with adaptive synapses, which controls agents instantiated in mutable environments. Such a configuration allows us to benchmark the adaptivity and generality of the controllers. The chosen tasks in the mutable environments are food foraging, emulation of logic gates, and cart-pole balancing. The three tasks are successfully solved with rather small network topologies and therefore it opens up the possibility of experimenting with more complex tasks and scenarios where curriculum learning is beneficial.

KEYWORDS

neuroevolution, artificial general intelligence, spiking neural network, spike-timing-dependent plasticity, Hebbian learning, weight agnostic neural network, meta-learning

1 Introduction

Artificial General Intelligence (AGI) or strong Artificial Intelligence (AI) is commonly discussed among AI researchers. It is often defined as human-level AI. However, the generality of an AI does not need to be considered at such a level of complexity. Even an artificial neural network that performs lots of different tasks as a collection of specialized

or weak AI (Reed et al., 2022) may not provide the level of generality observed in simple biological systems. In fact, our current artificial intelligent systems cannot emulate the adaptability to unknown conditions and learning capabilities of an animal with a simple nervous system, such as a worm (Ardiel and Rankin, 2010; Randi and Leifer, 2020). An alternative approach is to start the quest for the generality of AI from the simplest tasks that animals can do, but machines cannot, like behaving intelligently even in new environments (Crosby et al., 2019), i.e., out-of-distribution generalization (Shen et al., 2021). Moreover, AGI systems should be tested in tasks that require self-learning on the fly from sensory feedback, as it is often done in meta-learning and continual learning (Najarro and Risi, 2020; Zohora et al., 2021).

We argue that a radical paradigm change is needed in order to reach general intelligence (Lake et al., 2017; Crosby et al., 2019). Our hypothesis is that such a new paradigm requires learning systems with self-organizing properties, as discussed by Risi, (2021). In this work, our goal is to achieve the learning capabilities of a primitive brain. Therefore, we aim at a low-level AGI, i.e., a system that can learn a map function through sensory experience. Interpreting and understanding sensory inputs are achieved through evolution, particularly supervised evolution (Zador, 2019) of agents interacting with their environment.

The brain is the organ that interprets the encoded signals from our sensory organs, thanks to the ability to distinguish between positive and negative sensory experiences depending on what is considered to be good or harmful, e.g., pleasure and pain. The experiences of pleasure and pain serve as reward and penalty mechanisms that may affect our behavior by conditioning associative positive and negative cues with specific memories.

In this work, we evaluate the Neuroevolution of Artificial General Intelligence (NAGI) framework (Pontes-Filho and Nichele, 2019). NAGI is a low-level biologically-inspired AGI framework. NAGI consists of an evolvable spiking neural network with adaptive synapses and randomly-initialized weights. The network is evolved by an extension of the method NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen, 2002). The source code of NAGI is available at <https://github.com/SocratesNFR/neat-nagi-python>.

The evolved spiking neural network controls an agent placed in a mutable environment. Its chances of reproduction are proportional to how long it can survive in an environment that is constantly changing, sometimes abruptly. Evolution optimizes how the neurons are connected in the network, their type of neurotransmitters (excitatory or inhibitory), their susceptibility to background electrical current noise (analogous to bias), and their neuroplasticity. With such degrees of freedom in the optimization process, we attempt to approximately recapitulate the evolutionary process of the simplest brains. The mutable environment and random weight initialization propitiate a benchmark for generality and adaptivity of the agent.

We test NAGI in three mutable environments. The first one is a simple food foraging task, in which the agent has one photoreceptor (or light intensity sensor) used to identify food. The food type (color) is either black or white. Food can be edible or poisonous and this feature changes over time. The agent can also taste the food as its sensory feedback for good and bad actions. The second environment is a logic gate task. The spiking neural network needs to emulate different logic gates in series where the only reward and penalty sensory signals are the supporting mechanisms to identify the correct output. The third environment is a cart-pole balancing task. In this environment, the goal of the agent is to control the forces applied to the cart in order to maintain the pole above itself upright. The mutable component of this environment is the pole length, which changes during the lifetime of the agent. Because this environment has sensory feedback for the agent's actions, there is no need to add reward and penalty sensory signals.

The article is organized as follows: Section 2 explains the theoretical basis for understanding NAGI. Section 3 discusses the related work to our approach. Section 4 describes the details of the method and experiments. Section 5 presents the experimental results. Section 6 concludes the article including a discussion of the results and plans for future work.

2 Background

The components of the NAGI framework are inspired by the overlapping research fields of artificial life (Langton, 2019), evolutionary robotics (Doncieux et al., 2015), and computational neuroscience (Trappenberg, 2009). In particular, the controller for the agents is a Spiking Neural Network (SNN) (Izhikevich, 2003), which is a more biologically-plausible artificial neural network. The neurons in an SNN communicate through spikes, i.e., binary values in time series. Therefore, an SNN adds a temporal dimension to binary data. A neuron propagates such data depending on whether its membrane potential crossed a threshold value or not. If the threshold is crossed, the neuron propagates a signal represented as neurotransmitters to its connected neurons; otherwise, the action potential is not propagated. When neurotransmitters are released by a neuron, they can be of two types: excitatory, which increases the membrane potential and the likelihood of producing an action potential; or inhibitory, which has the opposite effect by decreasing the membrane potential. Efficient optimization of an SNN cannot happen through gradient descent as spike trains are not differentiable (Tavanaei et al., 2019). Instead, spiking neurons have biologically inspired local learning rules, such as Hebbian learning and Spike-Timing-Dependent Plasticity (STDP) (Hebb, 1949; Li et al., 2014). Those neuroplasticity rules are unsupervised, and their functionality in the brain is still not fully understood. However, it is inferred that the supervision comes

from a certain network configuration acquired through evolution. Therefore, in this work, we use a modification of NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen, 2002). NEAT uses a Genetic Algorithm (GA) (Holland, 1992) to optimize the weights and the topology of a growing neural network that is initialized with a minimal and functional size. NEAT is typically used to search for a network configuration that improves a fitness score while maintaining population diversity (speciation) and avoiding loss of genes during crossover (historical marking). For an accessible and extensive explanation of NEAT, please refer to Ref. (Welleck, 2019).

A distinction from NEAT is that the weights in the NAGI framework are randomly initialized, and they change (adapt) after deployment. The adaptation is coordinated by a realistic Hebbian learning rule, i.e., STDP. This neuroplasticity adjusts the synaptic strength of a neuron's dendrites (i.e., input connections) when it fires an action potential (or spike) that goes through its axon (i.e., output connection). The weights are modified according to the difference in time between incoming spikes and the generated action potential. More detailed information about SNN and STDP is available in Ref. (Camuñas-Mesa et al., 2019).

The body and brain interaction (sensors and actuators vs. controller) is often described as “chicken and egg” problem (Funes and Pollack, 1998). The natural evolution of body and brain happens together with the evolution of the environment. They evolve in cooperation and response to each other (Mautner and Belew, 2000). The application of supervised evolution of agents interacting with the environment is defined as embodied evolution (Watson et al., 1999). As such, an agent needs a body to learn from the reaction of its environment. We hypothesize that low-level general intelligence in nature emerged through the evolution of a sensory feedback learning method.

3 Related work

Neuroevolution with adaptive synapses was introduced in 2003 by Stanley et al. (2003). Such a method is a version of NEAT where the synaptic strength of the connections changes with Hebbian local learning rules. In their work, they used a food foraging task where an agent moves around a field surrounded by edible and poisonous food. The type of food did not change over time, but it was initialized differently at every new run. The agents needed to try the food first before identifying it. Therefore, the agents possess reward and penalty sensory signals as in NAGI. This method is rather similar to ours. However, NAGI is more biologically plausible, weight agnostic, and is tested in a mutable environment. Risi and Stanley, (2010) proposed an extended version by replacing the direct encoding of the network in NEAT with an indirect encoding.

Additional related methods are described in Refs. (Gaier and Ha, 2019) and (Najarro and Risi, 2020) where randomly-initialized artificial neural networks are used. The work of Gaier and Ha, (2019) uses a version of NEAT where each neuron can have one activation function out of several types. While in the method of Najarro and Risi, (2020), the network topology is fixed and each connection evolves to optimize the parameters of its Hebbian learning rule.

In a recent review on neuroevolution (Stanley et al., 2019), NEAT and its extensions are comparable to deep neural networks trained with gradient-based methods for reinforcement learning tasks. Such methods allow evolving artificial neural networks with indirect encoding for scalability, novelty search for diversity, meta-learning for learning how to learn, and architecture search for deep learning models. Moreover, neuroevolution is described as a key factor for reaching AGI, particularly in relation to meta-learning and open-ended evolution. Meta-learning encompasses the training of a model with certain datasets and testing it with others. The goal of the model is therefore to learn any given dataset by itself from experience (Thrun and Pratt, 1998). Open-ended evolution is the ability to endlessly generate a variety of solutions of increasing complexity (Taylor, 2019). In NAGI, meta-learning is an implicit target in the mutable environments and is implemented as neuroplasticity in the spiking neural network.

In 2020, Nadji-Tehrani and Eslami, (2020) introduced the framework for evolutionary artificial general intelligence (FEAGI). This method uses an indirect encoding technique for a spiking neural network that resembles the growth of the biological brain, which is called “neuroembryogenesis.” As a proof of concept, FEAGI demonstrates successful handwritten digits classification by learning through association and being able to recall digits from different image samples in real-time.

4 Neuroevolution of Artificial General Intelligence

The NAGI framework aims at providing a simplified model of the initial stages of the evolution of biological general intelligence (Pontes-Filho and Nichele, 2019). The evolving agents in NAGI consist of randomly-initialized spiking neural networks. Thus, a genome in NAGI does not require the definition of synaptic weights of the connections between neurons, as it is done in NEAT. Therefore, the synaptic weights in the genome are replaced by an STDP rule and its parameters for each neuron. Since biological neurons may provide one of the two main neurotransmitters, NAGI's genome defines such a feature in the neurons' genes. As such, a neuron can be either excitatory or inhibitory. To imitate the function of bias in artificial neural networks, neurons may be also susceptible to a “background electrical current noise.”

The environment changes during the lifetime of the agent. This forces the agent to learn new environmental conditions. Therefore, the agent is encouraged to generalize and learn how to learn. The aforementioned random initialization and mutable environment aim at benchmarking the basic properties needed for low-level AGI.

4.1 Spiking neural network

The spiking neural network has a fixed number of input and output neurons depending on the task to be solved. The neuroevolution process defines the number of hidden neurons that will be available. Hidden neurons can be either excitatory or inhibitory, while input and output neurons are always excitatory. Self-loops and cycles are permitted while duplicate connections between two neurons in the same direction are prohibited. The SNN is stimulated from the input neurons, as such units are spike generators. The spikes are uniformly generated in an assigned frequency or firing rate.

As a spiking neuron model, we use a simplification of the leaky integrate-and-fire model (Liu and Wang, 2001). A neuron's membrane potential v is increased directly by its inputs and decays over time by a factor λ_{decay} . We can then express the change in membrane potential Δv with regards to a time step Δt by

$$\Delta v(\Delta t) = \sum_{i=1}^n w_i x_i - \Delta t \lambda_{decay} v, \quad (1)$$

where x_i is the input value 0 (no spike) or 1 (spike) from the presynaptic neuron i , the dendrite for this connection has the synaptic strength defined as w_i , and n is the total number of presynaptic neurons that the dendrites are connecting. If the membrane potential v is greater than the membrane threshold v_{th} , a spike is released and the membrane potential returns to the resting membrane potential v_{rest} , which is 0. The time step Δt we use in the experiments is 0.1 ms, and decay factor λ_{decay} is 0.01 Δt . An action performed by the SNN is calculated by the number of spikes in a time window. Such an actuator time window covers 250 ms or 2,500 time steps. In NAGI, the weights of the SNN are randomly initialized with a normal distribution. The mean is equal to 1 and the standard deviation is equal to 0.2. The weights are always positive. As mentioned, the excitation and inhibition of a neuron are defined by the neurotransmitter of the presynaptic neuron.

4.1.1 Homeostasis

Biological neurons have a plasticity mechanism that maintains a steady equilibrium of the firing rate, which is called homeostasis (Betts et al., 2013; Kulik et al., 2019). In our method, the spiking neurons can have non-homogeneous inputs, which could lead to very different firing rates. It is

desirable that all neurons have approximately equal firing rates (Diehl and Cook, 2015). In order to homogenize the firing rates of the neurons in a network, the membrane threshold v_{th} is given by

$$v_{th} = \min\left(v_{th}^* + \Theta, \sum_{i=1}^n w_i\right), \quad (2)$$

where v_{th}^* is the "resting" membrane threshold equals to 1; and Θ starts with value 0, increases 0.2 every time a neuron fires, and decays exponentially with a rate of 0.01 Δt . Each neuron has an individual Θ . Therefore, a neuron firing more often will get a larger membrane threshold and consequently a lower firing rate. To compensate for a neuron with weak incoming weights, which causes a low firing rate; we instead use the sum of the incoming weights as the threshold.

4.1.2 Spike-Timing-Dependent Plasticity

The adjustment of the weights of the connections entering into a neuron happens on every input and output spike to and from a neuron. This is performed by STDP. It is done by keeping track of the time elapsed since the last output spike and each input spike from incoming connections within a time frame. Such a time frame is called the STDP time window and is set to be ± 40 ms. The difference between presynaptic and postsynaptic spikes, or the relative timing between them, denoted by Δt_r is given by

$$\Delta t_r(t_{out}, t_{in}) = t_{out} - t_{in}, \quad (3)$$

where t_{out} is the timing of the output spike and t_{in} is the timing of the input spike.

The synaptic weight change Δw is calculated in accordance with one of the four Hebbian learning rules. The functions for each of the four learning rules are given by

$$\Delta w(\Delta t_r) = \begin{cases} A_+ e^{-\frac{\Delta t_r}{\tau_+}} & \Delta t_r > 0, \\ -A_- e^{\frac{\Delta t_r}{\tau_-}} & \Delta t_r < 0, \text{ Asymmetric Hebbian} \\ 0 & \Delta t_r = 0; \end{cases} \quad (4)$$

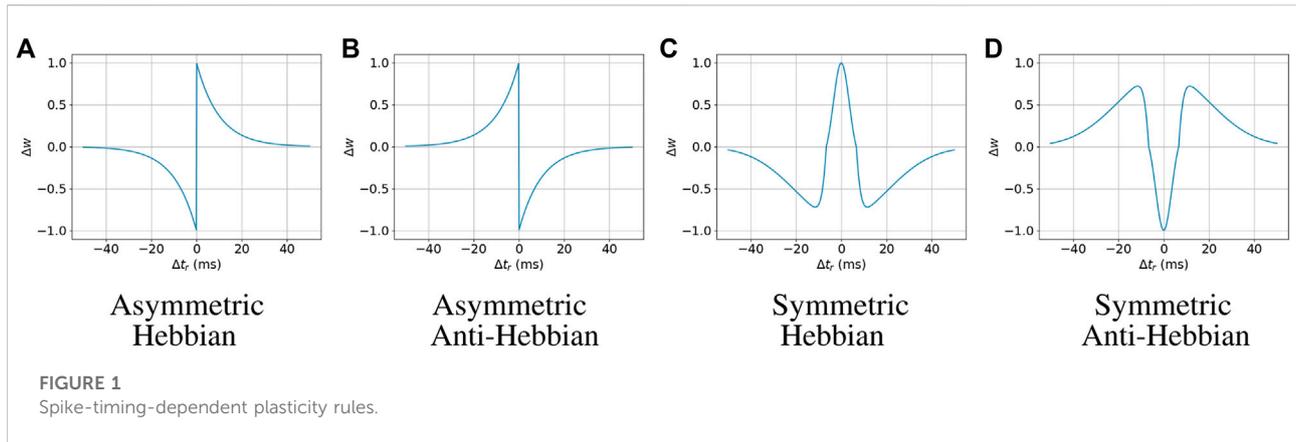
$$\Delta w(\Delta t_r) = \begin{cases} -A_+ e^{-\frac{\Delta t_r}{\tau_+}} & \Delta t_r > 0, \\ A_- e^{\frac{\Delta t_r}{\tau_-}} & \Delta t_r < 0, \text{ Asymmetric Anti-Hebbian} \\ 0 & \Delta t_r = 0; \end{cases} \quad (5)$$

$$\Delta w(\Delta t_r) = \begin{cases} A_+ g(\Delta t_r) & g(\Delta t_r) > 0, \\ A_- g(\Delta t_r) & g(\Delta t_r) < 0, \text{ Symmetric Hebbian} \\ 0 & g(\Delta t_r) = 0; \end{cases} \quad (6)$$

$$\Delta w(\Delta t_r) = \begin{cases} -A_+ g(\Delta t_r) & g(\Delta t_r) > 0, \\ -A_- g(\Delta t_r) & g(\Delta t_r) < 0, \text{ Symmetric Anti-Hebbian} \\ 0 & g(\Delta t_r) = 0; \end{cases} \quad (7)$$

where $g(\Delta t_r)$ is a Difference of Gaussian function given by

$$g(\Delta t_r) = \frac{1}{\sigma_+ \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{\Delta t_r}{\sigma_+}\right)^2} - \frac{1}{\sigma_- \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{\Delta t_r}{\sigma_-}\right)^2}, \quad (8)$$



A_+ and A_- are the parameters that affect the height of the curve, τ_+ and τ_- are the parameters that affect the width or steepness of the curve of the Asymmetric Hebbian functions, and σ_+ and σ_- are the standard deviations for the Gaussian functions used in the Symmetric Hebbian functions. It is also required that $\sigma_- > \sigma_+$. We experimentally found fitting ranges for each of these parameters, which are $A_+ = [0.1, 1.0]$, $A_- = [0.1, 1.0]$, $\tau_+ = [1.0, 10.0]$, and $\tau_- = [1.0, 10.0]$ for the asymmetric STDP functions; and $A_+ = [1.0, 10.6]$, $A_- = [1.0, 44.0]$, $\sigma_+ = [3.5, 10.0]$, and $\sigma_- = [13.5, 20.0]$ for the symmetric ones. The STDP curves with the maximum value of those parameters are illustrated in Figure 1.

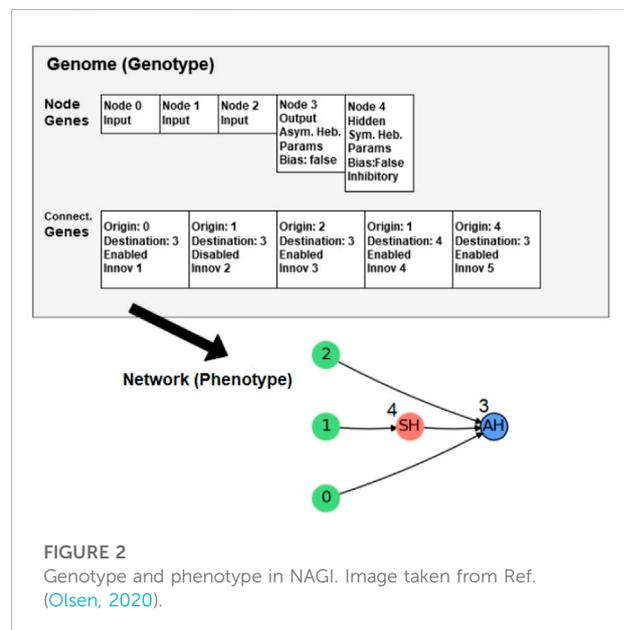
Weights can take values in a range $[w_{min}, w_{max}]$, and every neuron has a weight budget w_{budget} it must follow. What this means is that if the sum of a neuron's incoming weights exceed w_{budget} after initialization or STDP has been applied, they are normalized to w_{budget} , given by

$$\text{if } \sum_{i=1}^n w_i > w_{budget}, \text{ then } w_i = \frac{w_i w_{budget}}{\sum_{i=1}^n w_i}. \tag{9}$$

The parameters used during our experiments are $w_{min} = 0$, $w_{max} = 1$, and $w_{budget} = 5$. In case of a SNN without homeostasis, if a connection i has $w_i = w_{max}$, then $w_i = v_{th}$. Therefore, an action potential coming from i will always produce a spike. This is the reason why $w_{max} = v_{th}$.

4.2 Genome

The genome in NAGI is rather similar to the one in NEAT. Its node genes have three types: input, hidden, and output. Depending on the type of the node gene, there is a different collection of loci¹. The input node is a spike generator and



provides excitation to the neurons it is connected to. The gene of an input node is the same as in NEAT. The hidden and output nodes represent adaptable and mutable spiking neurons. They have three additional loci: the type of the learning rule, the set of the learning rule parameters, and a bias. The connection gene in NAGI has no weight locus as in NEAT. The reason for its removal is that the weights of the SNN are defined by a normal distribution.

The learning rule is one of the four STDPs. The set of learning rule parameters consists of four parameters that adjust the intensity of the weight change. They are different for symmetric and asymmetric learning rules. The symmetric parameters are $\{A_+, A_-, \sigma_+, \sigma_-\}$ and the asymmetric parameters are $\{A_+, A_-, \tau_+, \tau_-\}$. The bias is a Boolean value that determines if the neuron has a constant input of 0.001 being added to Δv , which is analogous to the background noise of the neuron.

¹ In the terminology of genetic algorithms, a value within a gene is also called a locus (plural loci).

The hidden node genes have a unique locus, which is a Boolean value that determines whether it represents an inhibitory or excitatory neuron. This locus is not included in the output node genes because they are always excitatory. As a result of combining all the descriptions of the genome in NAGI, the genotype and the phenotype are illustrated in Figure 2.

The initialization of the additional loci in the node genes can be conditional and non-uniform. The initialization of the neurotransmitter type of a neuron follows a similar proportion of excitatory and inhibitory neurons in the brain (Suknik et al., 2021). The probability of a neuron being added as excitatory is 70%. The probability of having a bias is 20%. Depending on the neurotransmitter, excitatory neurons have a 70% chance of initializing with Hebbian plasticity, and inhibitory neurons have the same chance but for anti-Hebbian plasticity. The learning rule parameters are initialized by sampling from a uniform distribution within the STDP parameter ranges.

The mutations of the additional loci happen in 10% of chance to switch the neurotransmitter type, bias, learning rule, and learning rule parameters. Those parameters have 2% chance of a fully re-initialization. When the parameters are assigned to be mutated, a random value sampled from a normal distribution with $\mu = 0$ and $\sigma^2 = m(p)$ is added to the parameter p . The equation of $m(p)$ is

$$m(p) = 0.2(p_{max} - p_{min}), \quad (10)$$

where p_{max} and p_{min} are the maximum and minimum values the parameter can have, given by the STDP parameter ranges. During the neuroevolution, 10% of the genotypes with the best fitness scores will be passed to the next generation unchanged, i.e., elitism.

4.3 Mutable environments

The benchmark tasks for NAGI are meant to evaluate the agent's ability to generalize and self-adapt. Therefore, they consist of environments that change during the lifetime of the agent. Two types of tasks are provided, binary classification (two tasks of this kind are provided) and control (one task of this kind is provided). The first type (binary classification) is the simplest one, however, it provides the most abrupt changes in the environment. The binary classification tasks are food foraging with one input, and logic gates with two inputs. The control task in a simulated physical environment is the cart-pole balancing from OpenAI Gym (Brockman et al., 2016). The changes are less abrupt in this last task as they consist in modifying the pole size. The fitness scores are calculated using the number of time steps t that the agent survived in these environments, normalized to the range $[0, 1]$ using the maximum possible lifetime L_{max} and minimum possible lifetime L_{min} . Therefore, the fitness function f is given by

$$f(t) = \frac{t - L_{min}}{L_{max} - L_{min}}. \quad (11)$$

In the binary classification tasks, the agents have an initial amount of health points that is reduced every time step as continuous damage. If a correct action is chosen, the health point amount is reduced by d_c health point. Otherwise, it is reduced by d_i . The input sample is given to the agent for 1 s or 10,000 time steps, then it is changed to a new one. The mutation of the environment condition happens when the agent has seen four samples. The order of the input samples and the environment conditions is fixed and cyclic.

We noticed that the number of spikes within the actuator time window can be the same for the output neurons and therefore allowing for a tie in many cases. Our solution to avoid spiking neural networks with this behavior is to include a "confidence" factor in the fitness score calculation. Therefore, the higher the difference between the spike count, the more confident the action is. If the action is correct and highly confident, the damage is d_c or closer. If the action is incorrect but highly confident, the damage is d_i or closer. The lack of confidence would make the damage lie between the values d_c and d_i . The spike count for the correct action s_c and incorrect one s_i are used to calculate the participation of the spikes for deciding the correct action p_c and the participation for the incorrect action p_i . In the iterations without spikes of the output neurons, normally the initial ones; the agent takes d_i as damage. Otherwise, the damage is calculated by

$$p_c(s_c, s_i) = \begin{cases} \frac{\max(0, \min(s_c, s_i)) - \max(0, \min(s_c, s_i)) + s_i}{2s_i} & s_c + s_i \leq 2s_i \\ \frac{s_c}{s_c + s_i} & s_c + s_i > 2s_i \end{cases} \quad (12)$$

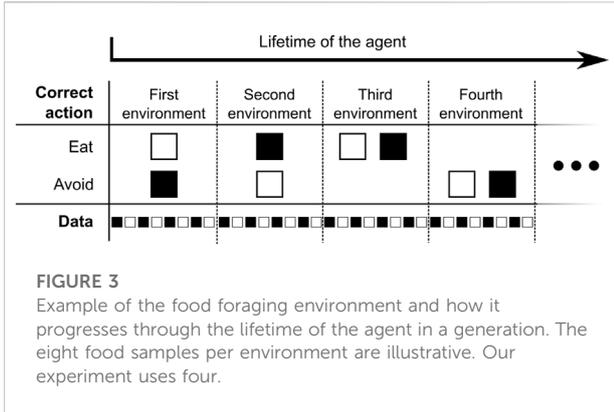
$$p_i(s_c, s_i) = 1 - p_c(s_c, s_i) \quad (13)$$

where s_t is the minimum "target" number of spikes. The purpose of s_t is to avoid assigning a too high or low fitness to agents that fire few spikes through their outputs. The agent takes damage at every time step and is given by

$$d(s_c, s_i) = d_c p_c(s_c, s_i) + d_i p_i(s_c, s_i) \quad (14)$$

Damaging is performed until the agent runs out of health points and 'dies'. Subsequently, the fitness score of the agent is calculated from the fitness function expressed in Eq. 11. The damage to the health points in a correct action d_c is 1, in an incorrect one d_i is 2. Therefore, correct actions result in a longer lifetime. The value for the minimum 'target' number of spikes s_t is 3 spikes.

In the control task of cart-pole balancing, the behavior of the mutable environment is different. A new environment is presented to the agent either after its failure or after the maximum number of environment iterations is reached. Moreover, the agents do not have health points. The fitness score is the normalization of the number of iterations that the agent survived after all environment conditions were executed.



4.3.1 Food foraging

The agent in the food foraging environment possesses just one light sensor for identifying the food “in front of it.” There are two types of food: edible and poisonous. As such, food is represented in two colors: black and white. The environment changes by randomly defining which food color is edible or poisonous. In this environment, the agent can act in two ways: eating or avoiding the food. The sample has a predefined time of exposure to the agent. An action is performed after the first spike and it continues for every time step in the environment simulation. After this exposure time, the food is replaced by a new one. The agent can only discover whether it is exposed to an edible or poisonous food by interacting with it. An incorrect action is defined as eating poisonous food, or avoiding edible food, while a correct action is defined as eating edible food or avoiding poisonous one. If the agent makes an incorrect action, it receives a penalty signal, from which the agent should learn over the generations that it represents pain, revulsion, or hunger. If the agent makes a correct action, it receives a reward signal, from which it should learn that it represents the pleasure of eating delicious food or recognizing that the food is poisonous. In [Figure 3](#), the food foraging environment is illustrated, how the environment changes and provides new food samples. In our experiment, the change of the environment occurs after presenting four food samples to the agent. The first food sample type is chosen randomly and alternates in every sample change. In [Table 1](#), the four combinations of edible and poisonous food for the white and black ones are shown. To evolve the spiking neural network for the food foraging task, the parameters of the genetic algorithm are the following: the population size is set to 100 individuals, and the number of generations is set to 1,000. This task was chosen because of its simplicity. In particular, it allows a virtual wheeled robot to forage for food using proximity sensors, such as in the related work of [Stanley et al. \(2003\)](#).

4.3.2 Logic gates

In this environment, the mutable environmental state is a two-input logic gate. The environment provides the agent with two binary inputs, i.e., 0’s and 1’s. The agent’s task is to predict

TABLE 1 Correct actions for all combinations of input food color and edible food in the food foraging task.

Food foraging environment conditions

Edible	Black	White	None	Both
Black	Eat	Avoid	Avoid	Eat
White	Avoid	Eat	Avoid	Eat

the correct output for the current logic gate given the current input. Similar to the food foraging environment, it receives a reward signal if it is currently predicting the correct output, and a penalty signal if it is currently predicting the wrong output.

In order to measure the generalizing properties of agents, we use two different sets of environments: a training environment, which is used in calculating the fitness score while running the evolutionary algorithm, and a test environment which has a fully disjoint set of possible environmental states. A full overview of the logic gates found in both the training and the test environments, as well as the truth values for all input and output combinations, are found in [Table 2](#) and [Table 3](#). The evolution of the spiking neural network is performed by a population of 100 individuals through 1,000 generations.

4.3.3 Cart-pole balancing

The cart-pole balancing is a well-known control task used as a benchmark problem in reinforcement learning. In this environment, there is a cart that moves when a force is applied to the left or to the right every time step. In the middle of the cart, there is a vertical pole connected to a non-actuated joint. The goal of this environment is to maintain the pole balanced upright by controlling the forces that move the cart. Moreover, the cart cannot move beyond the limits of the track. The observations available to the controller are the cart position, the cart velocity, the pole angle, and the pole angular velocity.

For training, we use poles of different sizes, which are 0.5 (default), 0.3, and 0.7. For testing, the sizes are 0.4, and 0.6. Those pole sizes are depicted in the [Supplementary Material](#). Each size can run up to 200 environment iterations and it is repeated three times during training for promoting stable controllers. If there are no more environment iterations or the pole falls, the cart-pole environment restarts with the next pole size while using the same SNN or finishes when all pole sizes were executed. The fitness score is calculated using the number of iterations the pole kept balanced. Subsequently, it is normalized to values between 0 and 1. The evolution for this task occurs with a population size of 256 during 500 generations.

TABLE 2 Truth table showing the correct output for each training logic gate.

Training logic gates

Input		A	B	NOT A	NOT B	Only 0	Only 1	XOR	XNOR
A	B								
0	0	0	0	1	1	0	1	0	1
0	1	0	1	1	0	0	1	1	0
1	0	1	0	0	1	0	1	1	0
1	1	1	1	0	0	0	1	0	1

TABLE 3 Truth table showing the correct output for each testing logic gate.

Test logic gates

Input		AND	NAND	OR	NOR
A	B				
0	0	0	1	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	0	1	0

environment, such data must be converted from the sensors and to the actuators. The flow of spikes over time can be quantified as firing rate, which corresponds to a frequency, or the number of spikes per second. The firing rate is the data representation that is converted as inputs and outputs for the SNN. However, the input firing rate must be within a minimum and a maximum value. In our experiments, we use the value range [5Hz, 50Hz]. The minimum and maximum value of the firing rate are simplified to a real number range [0, 1]. It is preferable that the data from the sensors has also a minimum and a maximum value. Otherwise, it will be necessary to clip sensory values or map the values to a desirable range.

In the binary classification tasks, all inputs and outputs are binary. Therefore, the minimum and maximum values for the input firing rate stand for, respectively, 0 and 1, or *False* and *True*. To avoid having a predefined threshold firing rate for the output neurons, we opt to have two output neurons for one binary value. The neuron with the highest firing rate within the actuator time window is the one defining the binary output value. If these two output neurons have the same firing rate, then the last one with

4.4 Data representation

The data type in a spiking neural network is a binary time series or a spike train. Because the agent senses and acts in the

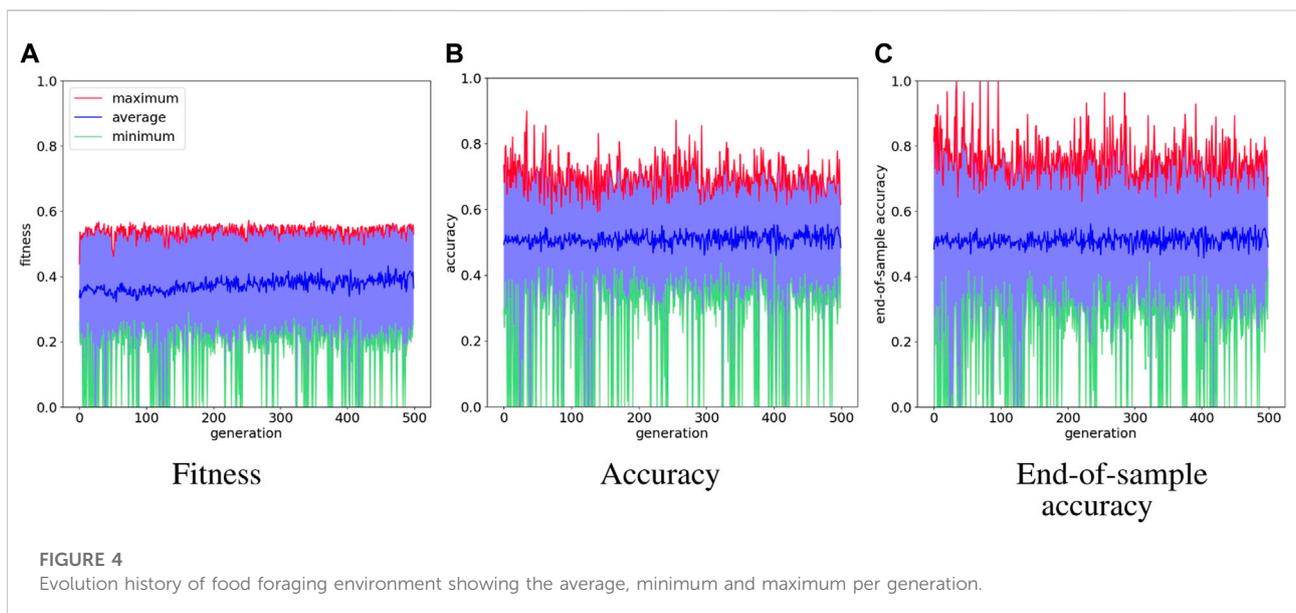
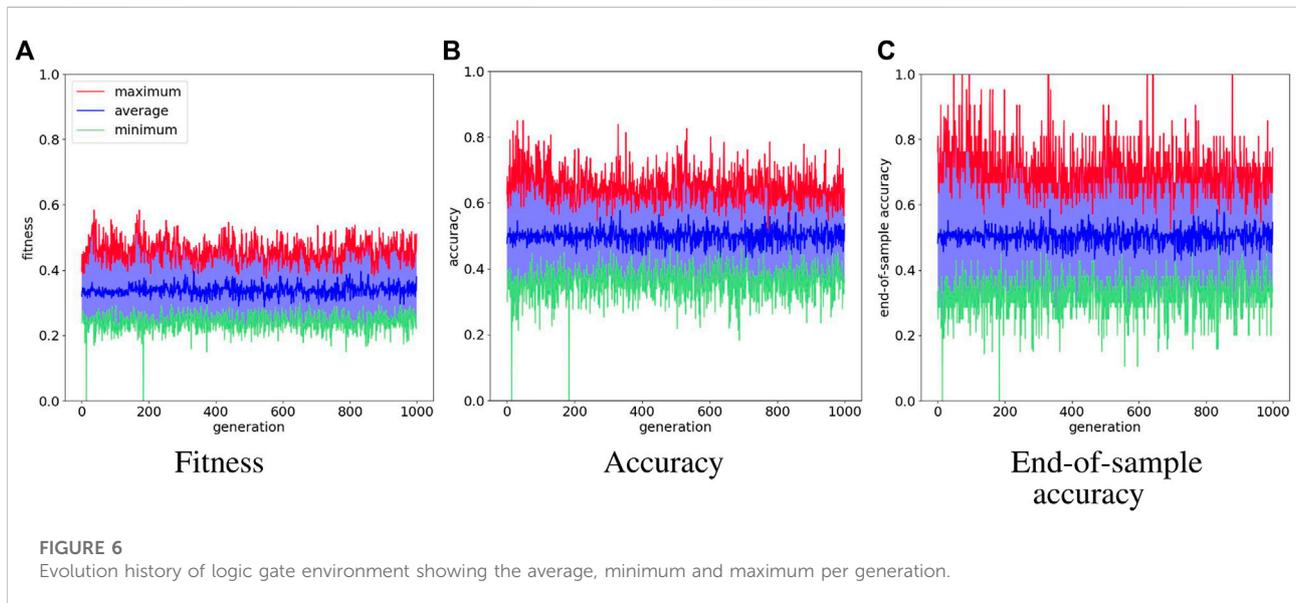


TABLE 4 Test simulations of the highest accuracy agent in the food foraging experiment. “Acc.” stands for accuracy and “EOS Acc.” for end-of-sample accuracy.

Food foraging test simulations

#	Acc. (%)	EOS Acc. (%)	Input order	Environment order
1	88.0	92.6	black, white	white, both, black, none
2	90.6	100	white, black	white, none, both, black
3	91.3	100	black, white	white, both, none, black
4	85.4	92.3	white, black	white, black, both, none
5	89.5	96.3	white, black	both, none, white, black
6	89.2	100.0	black, white	both, white, black, none
7	87.7	92.6	black, white	white, black, none, both
8	84.9	92.6	black, white	black, both, white, none
9	89.8	100	black, white	white, black, both, none
10	88.4	92.6	white, black	black, none, white, both
Avg	88.4	95.9	n/a	



1, z is the shift coefficient to adjust the function on the horizontal axis, h is the highest firing rate possible applied to an input neuron, and l is the lowest firing rate possible. The Gaussian function for converting observation value to firing rate is expressed by

$$\mathcal{F}_{Gaussian}(x | \mu, \sigma, h, l) = he^{-\frac{(x-\mu)^2}{2\sigma^2}} + l, \quad (16)$$

where μ is the mean and σ is the standard deviation. We replace $\frac{1}{\sigma\sqrt{2\pi}}$ in the original Gaussian function to h because, in this way, we can define the highest firing rate when the observation value is the mean. Neurons #1 and #3 use $\mathcal{F}_{sigmoid}$, while neuron #2 uses

$\mathcal{F}_{Gaussian}$. The parameters and the figures with the illustration of those equations are included in the [Supplementary Material](#).

5 Results

The evolution of the spiking neural networks in NAGI is evaluated with fitness score, accuracy, and end-of-sample accuracy for the binary classification tasks, which are food foraging and logic gate. The accuracy is measured at every time step of the simulation. The end-of-sample accuracy stands for the accuracy measured in the last time step of a

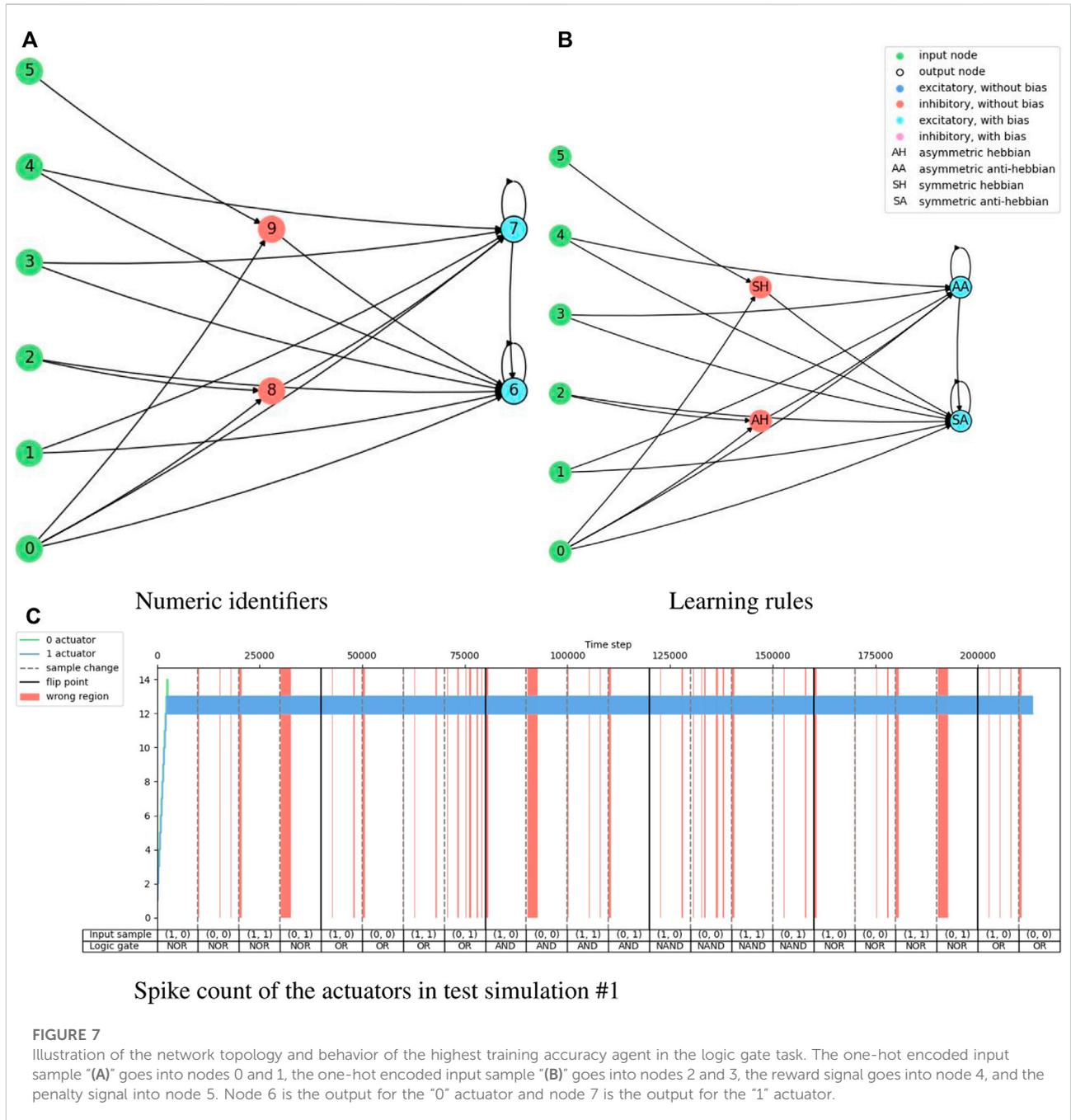


FIGURE 7 Illustration of the network topology and behavior of the highest training accuracy agent in the logic gate task. The one-hot encoded input sample “(A)” goes into nodes 0 and 1, the one-hot encoded input sample “(B)” goes into nodes 2 and 3, the reward signal goes into node 4, and the penalty signal into node 5. Node 6 is the output for the “0” actuator and node 7 is the output for the “1” actuator.

sample. The assessment performed for the control task with cart-pole balancing is done with the fitness score. We test the best performing agent in a task with ten simulations where their details are also provided.

Figure 4 shows the evolution history of the food foraging task. The average fitness score has a slight increase, but the maximum fitness score does not follow this trend. The accuracy and end-of-sample accuracy have high variation with their maximum values, but they consist of high accuracies. Moreover, some early

generations register 100% end-of-sample accuracy. The three measurements do not improve through the generations. However, good solutions are already found in the first generation. Therefore, this is an easy task that requires a small SNN. For test simulations, we select the individual with the highest accuracy, which is found in generation number 34 and has an accuracy of 89.8%. Its fitness score is 0.541395 and its end-of-sample accuracy is 100%. Its topology is shown in Figure 5. Paying attention to this topology, the hidden nodes are not

TABLE 5 Test simulations of the highest training accuracy agent in the logic gate experiment. “Acc.” stands for accuracy and “EOS Acc.” for end-of-sample accuracy.

Logic gate test simulations

#	Acc. (%)	EOS Acc. (%)	Input order (A, B)	Environment order
1	89.8	100	(1, 0), (0, 0), (1, 1), (0, 1)	NOR, OR, AND, NAND
2	85.2	95.2	(1, 1), (1, 0), (0, 0), (0, 1)	OR, NOR, NAND, AND
3	86.0	100	(1, 0), (1, 1), (0, 1), (0, 0)	NOR, OR, AND, NAND
4	85.9	95.2	(0, 0), (1, 1), (0, 1), (1, 0)	NAND, AND, OR, NOR
5	79.9	85.7	(0, 0), (0, 1), (1, 0), (1, 1)	NAND, AND, NOR, OR
6	88.8	100	(1, 0), (0, 0), (1, 1), (0, 1)	AND, NAND, OR, NOR
7	85.1	90.5	(0, 0), (1, 1), (1, 0), (0, 1)	OR, NOR, NAND, AND
8	84.8	90.5	(1, 1), (0, 1), (0, 0), (1, 0)	NOR, NAND, OR, AND
9	83.7	85.7	(0, 0), (1, 0), (0, 1), (1, 1)	NAND, NOR, OR, AND
10	88.5	100	(1, 1), (1, 0), (0, 0), (0, 1)	NOR, AND, OR, NAND
Avg	85.7	94.2	n/a	

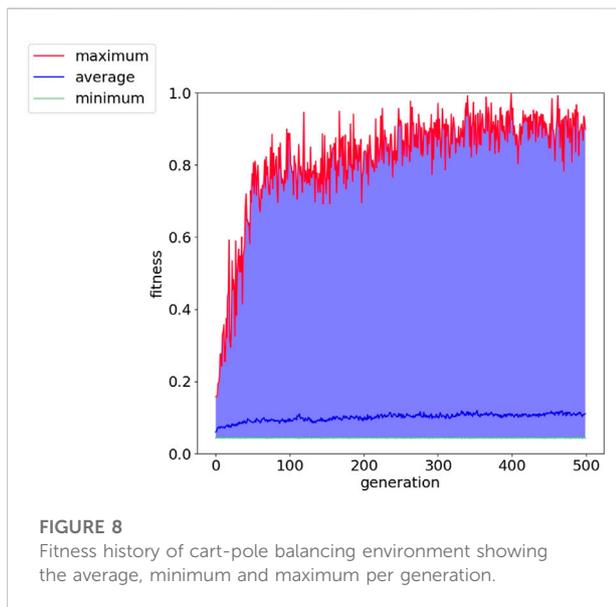


TABLE 6 Test simulations of the highest fitness agent in the cart-pole balancing experiment.

Cart-pole balancing test simulations

#	Fitness	# Steps 0.4	# Steps 0.6	Environment order
1	1.000	200	200	0.4, 0.6
2	1.000	200	200	0.4, 0.6
3	1.000	200	200	0.6, 0.4
4	0.943	200	177	0.4, 0.6
5	0.800	154	166	0.6, 0.4
6	0.792	179	138	0.4, 0.6
7	0.835	200	134	0.6, 0.4
8	0.845	200	138	0.4, 0.6
9	0.873	200	149	0.6, 0.4
10	0.720	88	200	0.6, 0.4
Avg	0.874	178.3	171.5	n/a

needed. They form a loop that does not connect with the output nodes. The topology summarizes in one of the one-hot encoded input nodes (node 1) connecting to the two output nodes. Then, the node with the penalty signal (node 3) connects only with the node for the “eat” actuator (node 4). The behavior of the network is illustrated in Figure 5C. The topology of the network indicates that the two output neurons have the same data input from node 1, but the neuron for “avoid” action has a bias, which gives it a small excitatory current. If “avoid” is the wrong action, the penalty input signal from node 3 excites the output neuron for the “eat” action. This is how the spiking neural network decides the actions from “understanding” the feedback of the

environment given by the penalty input signal. The result of the ten test simulations is presented in Table 4.

Figure 6 shows the training results of the logic gate task and it includes the test of the maximum individual of the measurement in every generation. The fitness score, accuracy, and end-of-sample accuracy maintain average values with high variation. However, the evolution of the agents in the logic gate task is similar to the one in the food foraging. The early generations already contain good spiking neural networks for the task. The best-performing agent is selected from the accuracy measurement. This individual is in generation 48 and has an accuracy of 85.0%. Its fitness score is 0.4421625 and its end-of-sample accuracy is 100%. The topology of this spiking neural

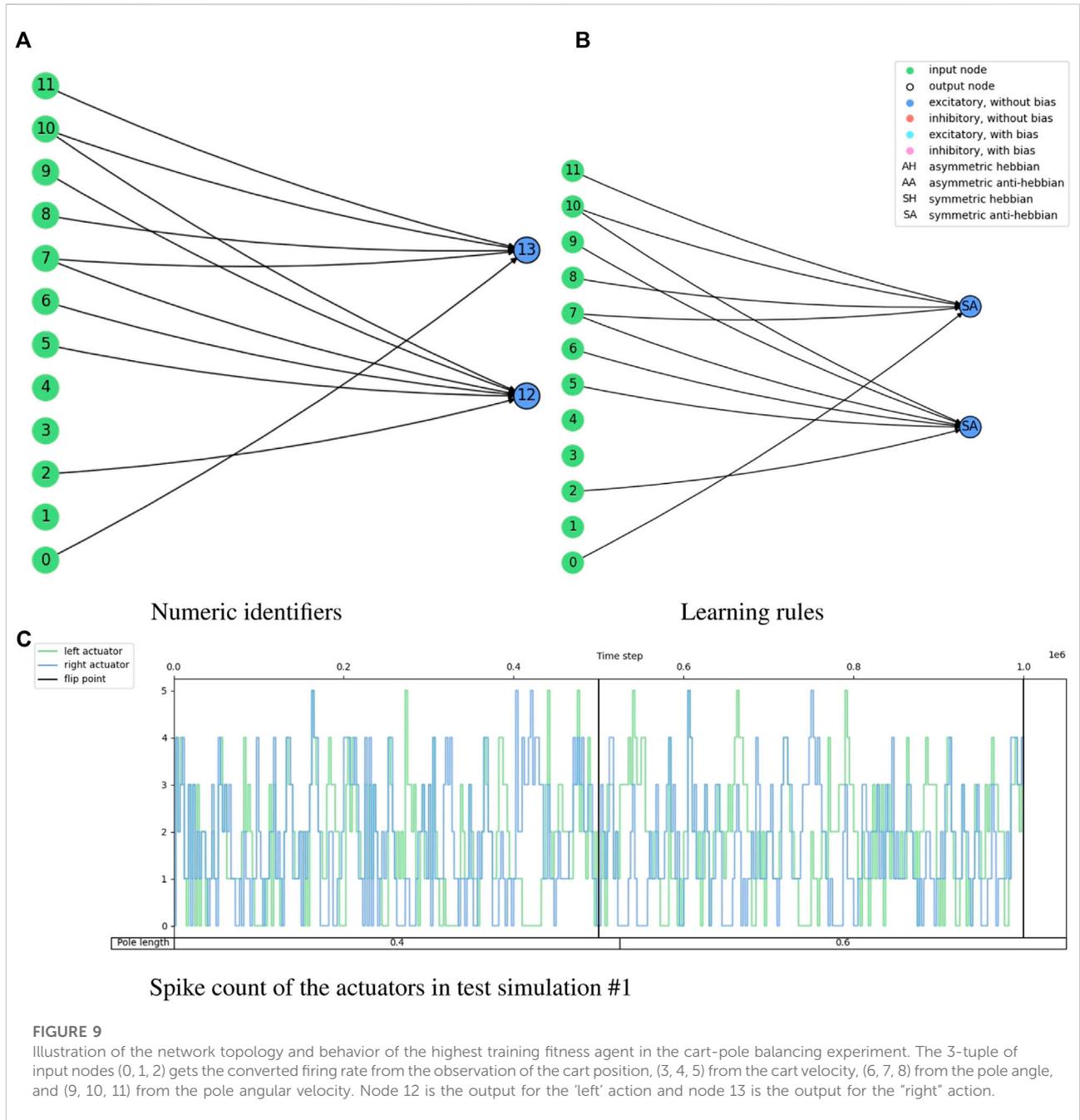


FIGURE 9 Illustration of the network topology and behavior of the highest training fitness agent in the cart-pole balancing experiment. The 3-tuple of input nodes (0, 1, 2) gets the converted firing rate from the observation of the cart position, (3, 4, 5) from the cart velocity, (6, 7, 8) from the pole angle, and (9, 10, 11) from the pole angular velocity. Node 12 is the output for the ‘left’ action and node 13 is the output for the ‘right’ action.

network is shown in [Figure 7](#). Its behavior is shown in [Figure 7C](#). Even though we have trained with a “confidence” factor in the fitness function, the spike counts are still with almost the same values. [Table 5](#) contains the accuracy and end-of-sample accuracy of ten test simulations, which indicates that the SNN can be general to reproduce the behavior of logic gates without being trained to them.

[Figure 8](#) shows the fitness score history through the evolution for the cart-pole balancing task. This task is the

one with the highest difficulty to find a good genome for the adaptive spiking neural network. It can be noted that the fitness score improves through the generations. The maximum fitness score in a generation goes from around 0.16 in the first generation to 0.99944 in generation number 399. Such an individual is the one selected for the test simulations. Its topology is illustrated in [Figure 9](#) and the spike counts of the actuators for “left” and “right” actions are shown in [Figure 9C](#). The spiking neural network has no

hidden neurons. Therefore, the SNN works as an input selection for the output neurons. The result of the ten test simulations is presented in Table 6. When the pole is balanced for more than 100 iterations, the controller is considered successful.

6 Discussion and conclusion

We successfully solved all three presented tasks with the NAGI framework. The spiking neural networks found showed generality to the binary classification tasks, even to unseen conditions in the case of the emulation of logic gates. The neuroevolution produced rather simple topologies for the SNNs. We infer that binary classification is easy due to the binary performance feedback. For further research, multi-class classification is considered.

The cart-pole balancing task was successfully solved without any hidden neurons. The conversion of one observation into three input neurons is used to avoid the requirement of weight fine-tuning due to small differences in firing rate and also to the assumption that Hebbian plasticity works better with binary data (active and inactive) (Pontes-Filho and Liwicki, 2019). With such a conversion, the SNN became an input selection.

The topologies for the three tasks caught our attention because almost all output excitatory neurons were anti-Hebbian, and the two inhibitory hidden neurons in the logic gate solution have Hebbian neuroplasticity. Our initial hypotheses were that excitatory neurons mainly have Hebbian learning rules, and inhibitory neurons are anti-Hebbian. That was the reason for having different probabilities for anti-Hebbian and Hebbian learning rules depending on the type of the neurotransmitter when adding a new neuron through mutation.

Even though there is elitism, the performance measurements are unstable through generations. This is a demonstration of the randomness in the initialization of the weights, and input and environment order. This can be perceived in the results of the ten test simulations of the three tasks.

For future work, we plan to attempt more challenging tasks. If there is a failure in executing the task, the constraints imposed on NAGI can be eased. A major constraint is that one neuron has one plasticity rule for all dendrites. Maybe its removal can simplify issues in difficult tasks. This constraint was intended to reduce the dimensionality of the search space in the neuroevolution and an assumption that the dendrites in the same neuron adapt under one learning rule. This modification is also aligned with the work of Najarro and Risi, (2020), which has meta-learning properties for more difficult control tasks than the cart-pole balancing, such as top-down car racing and quadruped walk. Another opportunity is the addition of curriculum learning (Bengio et al., 2009; Narvekar et al., 2020) for increasing the complexity of the task while the agent becomes better over the generations.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

Author contributions

SP-F had the main idea, supervised most part of the work, implemented additional experiments, and wrote the initial draft of the manuscript. KO contributed to the writing of the initial draft and performed most of the experimental work as part of his master's thesis while being mainly supervised by SP-F. AY, MR, PH, and SN co-supervised the work. All authors reviewed the manuscript.

Funding

This work was partially funded by the Norwegian Research Council (NFR) through their IKTPLUSS research and innovation action under the project Socrates (grant agreement 270961).

Acknowledgments

This manuscript includes materials from the Master's thesis of Olsen, (2020). Its preprint is available in Ref. (Pontes-Filho et al., 2022).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: <https://www.frontiersin.org/articles/10.3389/frobt.2022.1007547/full#supplementary-material>

References

- Ardiel, E. L., and Rankin, C. H. (2010). An elegant mind: Learning and memory in *Caenorhabditis elegans*. *Learn. Mem.* 17, 191–201. doi:10.1101/lm.960510
- Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). “Curriculum learning,” in Proceedings of the 26th annual international conference on machine learning, 41–48.
- Betts, J. G., Young, K. A., Wise, J. A., Johnson, E., Poe, B., Kruse, D. H., et al. (2013). *Anatomy and physiology*. Houston, TX: OpenStax. Available at: <https://openstax.org/details/books/anatomy-and-physiology>.
- Bowmaker, J. K., and Dartnall, H. J. (1980). Visual pigments of rods and cones in a human retina. *J. Physiology* 298, 501–511. doi:10.1113/jphysiol.1980.sp013097
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*
- Camuñas-Mesa, L. A., Linares-Barranco, B., and Serrano-Gotarredona, T. (2019). Neuromorphic spiking neural networks and their memristor-CMOS hardware implementations. *Materials* 12, 2745. doi:10.3390/ma12172745
- Crosby, M., Beyret, B., and Halina, M. (2019). The animal-ai olympics. *Nat. Mach. Intell.* 1, 257. doi:10.1038/s42256-019-0050-3
- Diehl, P., and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9, 99. doi:10.3389/fncom.2015.00099
- Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. G. (2015). Evolutionary robotics: What, why, and where to. *Front. Robot. AI* 2, 4. doi:10.3389/frobt.2015.00004
- Funes, P., and Pollack, J. (1998). Evolutionary body building: Adaptive physical designs for robots. *Artif. Life* 4, 337–357. doi:10.1162/106454698568639
- Gaier, A., and Ha, D. (2019). “Weight agnostic neural networks,” in *Advances in Neural Information Processing Systems*. Editors H. Wallach, H. Larochelle, A. Beygelzimer, F. D. Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc.) 32. Available at: <https://proceedings.neurips.cc/paper/2019/file/e98741479a7b998f88b8f8c9f0b66f1-Paper.pdf>.
- Han, J., and Moraga, C. (1995). “The influence of the sigmoid function parameters on the speed of backpropagation learning,” in Proceedings of the International Workshop on Artificial Neural Networks: From Natural to Artificial Neural Computation (London, UK, UK: Springer-Verlag), 195–201.
- Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. New York: Wiley.
- Holland, J. H. (1992). Genetic algorithms. *Sci. Am.* 267, 66–72. doi:10.1038/scientificamerican0792-66
- Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi:10.1109/tnn.2003.820440
- Kulik, Y., Jones, R., Moughamian, A. J., Whippen, J., and Davis, G. W. (2019). Dual separable feedback systems govern firing rate homeostasis. *Elife* 8, e45717. doi:10.7554/elife.45717
- Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behav. Brain Sci.* 40, e253. doi:10.1017/s0140525x16001837
- Langton, C. (2019). *Artificial life: Proceedings of an interdisciplinary workshop on the synthesis and simulation of living systems*. London: Routledge.
- Li, Y., Zhong, Y., Zhang, J., Xu, L., Wang, Q., Sun, H., et al. (2014). Activity-dependent synaptic plasticity of a chalcogenide electronic synapse for neuromorphic systems. *Sci. Rep.* 4, 4906. doi:10.1038/srep04906
- Liu, Y.-H., and Wang, X.-J. (2001). Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *J. Comput. Neurosci.* 10, 25–45. doi:10.1023/a:1008916026143
- Mautner, C., and Belew, R. K. (2000). Evolving robot morphology and control. *Artif. Life Robot.* 4, 130–136. doi:10.1007/bf02481333
- Nadji-Tehrani, M., and Eslami, A. (2020). A brain-inspired framework for evolutionary artificial general intelligence. *IEEE Trans. Neural Netw. Learn. Syst.* 31, 5257–5271. doi:10.1109/tnnls.2020.2965567
- Najarro, E., and Risi, S. (2020). “Meta-Learning through Hebbian Plasticity in Random Networks,” in *Advances in Neural Information Processing Systems*. Editors H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Curran Associates, Inc.) 33, 20719–20731. Available at: <https://proceedings.neurips.cc/paper/2020/file/ee23e7ad9b473ad072d57aaa9b2a5222-Paper.pdf>.
- Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *J. Mach. Learn. Res.* 21 (181), 1–50.
- Olsen, K. (2020). *Neuroevolution of artificial general intelligence*. Oslo: University of Oslo.
- Patel, J. K., and Read, C. B. (1996). *Handbook of the normal distribution*, 150. CRC Press.
- Pontes-Filho, S., and Liwicki, M. (2019). “Bidirectional learning for robust neural networks,” in 2019 International Joint Conference on Neural Networks (IEEE), IJCNN '19, 1–8.
- Pontes-Filho, S., and Nichele, S. (2019). A conceptual bio-inspired framework for the evolution of artificial general intelligence. *arXiv preprint arXiv:1903.10410*
- Pontes-Filho, S., Olsen, K., Yazidi, A., Riegler, M., Halvorsen, P., and Nichele, S. (2022). Towards the neuroevolution of low-level artificial general intelligence. *arXiv preprint arXiv:2207.13583*
- Randi, F., and Leifer, A. M. (2020). Measuring and modeling whole-brain neural dynamics in *Caenorhabditis elegans*. *Curr. Opin. Neurobiol.* 65, 167–175. doi:10.1016/j.conb.2020.11.001
- Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., et al. (2022). A generalist agent. *arXiv preprint arXiv:2205.06175*
- Risi, S., and Stanley, K. O. (2010). “Indirectly encoding neural plasticity as a pattern of local rules,” in International Conference on Simulation of Adaptive Behavior. Springer, 533–543.
- Risi, S. (2021). The future of artificial intelligence is self-organizing and self-assembling. Available at: sebastianrisi.com
- Shen, Z., Liu, J., He, Y., Zhang, X., Xu, R., Yu, H., et al. (2021). Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*
- Stanley, K. O., Bryant, B. D., and Miikkulainen, R. (2003). Evolving adaptive neural networks with and without adaptive synapses. In The 2003 Congress on Evolutionary Computation (IEEE), 2557–2564.
- Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nat. Mach. Intell.* 1, 24–35. doi:10.1038/s42256-018-0006-z
- Stanley, K. O., and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evol. Comput.* 10, 99–127. doi:10.1162/106365602320169811
- Sukenik, N., Vinogradov, O., Weinreb, E., Segal, M., Levina, A., and Moses, E. (2021). Neuronal circuits overcome imbalance in excitation and inhibition by adjusting connection numbers. *Proc. Natl. Acad. Sci. U. S. A.* 118, e2018459118. doi:10.1073/pnas.2018459118
- Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. *Neural Netw.* 111, 47–63. doi:10.1016/j.neunet.2018.12.002
- Taylor, T. (2019). Evolutionary innovations and where to find them: Routes to open-ended evolution in natural and artificial systems. *Artif. Life* 25, 207–224. doi:10.1162/artl_a_00290
- Thrun, S., and Pratt, L. (1998). “Learning to learn: Introduction and overview,” in *Learning to learn* (Springer), 3–17.
- Trappenberg, T. (2009). *Fundamentals of computational neuroscience*. Oxford: OUP Oxford.
- Watson, R. A., Ficci, S. G., and Pollack, J. B. (1999). “Embodied evolution: Embodying an evolutionary algorithm in a population of robots,” in Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406).
- Welleck, S. (2019). Evolving networks. Available at: wellecks.wordpress.com
- Zador, A. M. (2019). A critique of pure learning and what artificial neural networks can learn from animal brains. *Nat. Commun.* 10, 3770–3777. doi:10.1038/s41467-019-11786-6
- Zohora, F. T., Karia, V., Daram, A. R., Ziyarah, A. M., and Kudithipudi, D. (2021). “Metaplasticnet: Architecture with probabilistic metaplastic synapses for continual learning,” in 2021 IEEE International Symposium on Circuits and Systems (ISCAS) (IEEE).