Levels of Dynamics and Adaptive Behavior in Evolutionary Neural Controllers

Jesper Blynel **Dario Floreano**
Institute of Systems Engineering
Faculty of Engineering Science and Technology
Swiss Federal Institute of Technology (EPFL), CH-1015 Lausanne
Jesper.Blynel@epfl.ch, Dario.Floreano@epfl.ch

Abstract

Two classes of dynamical recurrent neural networks, Continuous Time Recurrent Neural Networks (CTRNNs) (Yamauchi and Beer, 1994) and Plastic Neural Networks (PNNs) (Floreano and Urzelai, 2000) are compared on two behavioral tasks aimed at exploring their capabilities to display reinforcement-learning like behaviors and adaptation to unpredictable environmental changes. The networks report similar performances on both tasks, but PNNs display significantly better performance when sensory-motor re-adaptation is required after the evolutionary process. These results are discussed in the context of behavioral, biological, and computational definitions of learning.

1 Introduction

In the great majority of experiments in Evolutionary Robotics (Nolfi and Floreano, 2000) evolved control systems consist of artificial neural networks most of which include recurrent connections. Recurrent connections potentially give a neural network rich temporal dynamics as well as the possibility to capture and exploit time-dependent events.

In recurrent neural networks, the activation states of the neurons can detect and maintain time-dependent activation patterns from sensors and/or other neurons that occur only briefly over time (Harvey et al., 1994). This information can modulate behavior (Beer and Gallagher, 1992), produce different actions for similar (or lacking) sensory information (Floreano and Mondada, 1996), and be used to detect and represent behavioral sequences (Tani, 1996).

Yamauchi and Beer (1994) showed that a particular class of Continuous-Time Recurrent Neural Networks (CTRNNs) (Hopfield, 1984) can be evolved to display reinforcement-learning-like behavior without modifications of the connection strengths. The authors evolved the connection strengths and neural time constants of networks that were asked to produce different output sequences in the presence of certain input patterns and reinforcement signals. The reinforcement signal consisted of toggling the value of an input neuron between zero and one. The authors analyzed the evolved CTRNNs using dynamical systems theory and showed that the reinforcement signals effectively modulated the state-space trajectories of the neuron activations in order to produce the desired output sequences.

Following a different approach, the dynamical and behavioral properties of Plastic Neural Networks (PNNs) was investigated in (Floreano and Urzelai, 2000, Urzelai and Floreano, 2001). The neuron sign, learning rate, and type of Hebbian learning (one of four possible learning rules) that was used to change on-line the strengths of all incoming connections to a neuron was evolved in networks with discrete-time recurrent connections.[1] In these experiments, the connection strengths were always initialized to small random values and modified after each neuron update using genetically-specified Hebb rules and learning rates. Analysis of experimental results showed that the evolved PNNs were capable of solving complex behavioral tasks that require precise sequences of actions by rapidly switching synaptic configuration whenever a new sequence of actions is required. Evolved PNNs also displayed remarkable on-line adaptability to new environmental conditions without requiring incremental evolution. However, no evidence was found that PNNs could actually learn and retain behavioral abilities over time or display reinforcement-learning-like properties.

These two investigation directions – CTRNNs and PNNs – provide apparently contrasting and non-intuitive results. On the one hand, neural networks with evolvable continuous-time dynamics can display learning-like behavior without synaptic plasticity. On the other hand,

[1] In networks with discrete-time recurrent connections the potential dynamics are limited compared to CTRNNs, but the system allows a simpler software implementation by maintaining a copy of the neural activities at the previous time step. These kinds of networks are sometimes referred to as having "memory units" and have been studied, among others, by (Elman, 1990).

networks with evolvable plastic connections can display rich behavioral dynamics without learning-like properties. This apparent contrast can be resolved if one considers that each model represents a specific implementation of a more general class of neural models with time-dependent states. In the case of CTRNNs it is the *neurons* which have a time-dependent state, whereas in the case of PNNs the *connection strengths* are time-dependent. Since the output of the network is a function of the product between neuron activations and connection strengths in both cases, one may arbitrarily decide where to apply the time-dependent property and, to a first approximation, the two models would be equivalent. Therefore, different abilities, such as reactive and learning-like behavior, could be explained purely in terms of different time-scales of dynamics.[2]

However, so far these two models have never been compared on a set of tasks which require time-dependent neural states. In this paper we begin to explore these issues by experimentally comparing evolutionary CTRNNs and PNNs on two sets of robotics experiments, one aimed at evolving reinforcement learning-like behaviors and the other aimed at testing the systems adaptability to various environmental changes.

2 Network Models

2.1 Continuous-Time Recurrent Neural Networks (CTRNNs)

In the continuous-time recurrent neural networks used in this paper the state of each neuron is governed by the following equation:

$$\frac{d\gamma_i}{dt} = \frac{1}{\tau_i}\left(-\gamma_i + \sum_{j=1}^{N} w_{ij} A_j + \sum_{k=1}^{S} w_{ik} I_k \right) \qquad (1)$$

where N is the number of neurons, i $(= 1, 2, ..., N)$ is the index, γ_i describes the neuron state (cell potential), τ_i is the time constant, w_{ij} is the strength of the synapse from the presynaptic neuron j to the postsynaptic neuron i, $A_j = \sigma(\gamma_j - \theta_j)$ is the activation of the presynaptic neuron where $\sigma(x) = 1/(1 + e^{-x})$ is the standard logistic function and θ_j is a bias term. Finally, S is the number of sensory receptors, w_{ik} is the strength of the synapse from the presynaptic sensory receptor k to the postsynaptic neuron i and I_k is the activation of the sensory receptor ($I_k \in [0, 1]$). As in the work by (Yamauchi and Beer, 1994) the *Forward Euler* numerical integration method is used. The iterative update rule for the state of each neuron becomes:

$$\gamma_i(n+1) = \gamma_i(n) + \frac{\Delta t}{\tau_i}\left(-\gamma_i(n) + \sum_{j=1}^{N} w_{ij} A_j(n) + \sum_{k=1}^{S} w_{ik} I_k \right) \qquad (2)$$

where n is the iteration step number and Δt is the step size. This integration method is numerically stable when Δt is less than twice the smallest time-constant in the network(Hines and Carnevale, 1998). Initially the state of each neuron is $\gamma_i(0) = 0 \ \forall i$, the step size set to $\Delta t = 1$. The range of the other parameters were the following:

$$\tau \in [1, 70], \ \theta \in [-1, 1] \ \textit{and} \ w \in [-5, 5]$$

Notice from equation 2 that each neuron has an internal state and that the time constant τ controls its dynamics. Large time constants result in slowly changing neuron states while small time constants approximate reactive neurons.

2.2 Plastic Neural Networks (PNNs)

In the discrete-time, fully-recurrent, plastic neural networks used, each neuron activation A_i is updated using the following equation at every activation cycle:

$$A_i(n+1) = \sigma\left(\sum_{j=1}^{N} w_{ij} A_j(n) \right) + I_i, \qquad (3)$$

where the activation of the sensory receptor $I_i = 0$ for hidden and motor neurons. As a consequence, the range of A_i is $[0, 2]$ for input neurons and $[0, 1]$ for the hidden and output neurons. Each synaptic weight w_{ij} is randomly initialized in the range $[0, 0.1]$ and is updated after every sensory-motor cycle by the following rule:

$$w_{ij}(n+1) = w_{ij}(n) + \eta \Delta w_{ij}(n) \qquad (4)$$

where $0.0 < \eta < 1.0$ is the learning rate and Δw_{ij} is one of four Hebb rules specified in the genotype:[3]

1. *Plain Hebb rule*: strengthens the synapse proportionally to the correlated activity of the two neurons.

$$\Delta w = (1 - w)\, xy$$

2. *Postsynaptic rule*: behaves as the plain Hebb rule, but in addition it weakens the synapse when the postsynaptic node is active but the presynaptic is not.

$$\Delta w = w\,(-1 + x)\, y + (1 - w)\, xy$$

[2] This last point was raised during personal discussions with Inman Harvey and Ezequiel Di Paolo.

[3] Before applying a learning rule, the activation of input neurons are divided by 2 to scale them into the range $[0, 1]$.

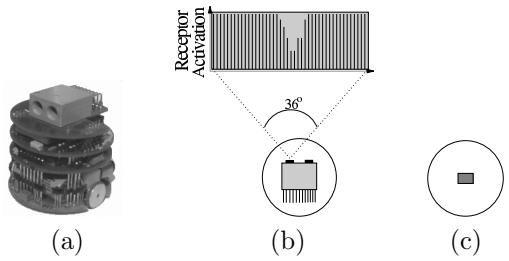

Figure 1: The Khepera robot used in the experiments. The robot has (a) 8 infrared sensors (small black rectangles) ditributed around the body that can measure object proximity and light intensity; (b) a linear vision module with 64 equally-spaced photoreceptors covering a visual field of 36°; and (c) a floor-color sensor under the body between the two wheels.

3. *Presynaptic rule*: weakening occurs when the presynaptic unit is active but the postsynaptic is not.

$$\Delta w = wx\,(-1+y) + (1-w)\,xy$$

4. *Covariance rule*: strengthens the synapse whenever the difference between the activations of the two neurons is less than half their maximum activity, otherwise the synapse is weakened.

$$\Delta w = \begin{cases} (1-w)\mathcal{F}(x,y) & \text{if } \mathcal{F}(x,y) > 0 \\ (w)\mathcal{F}(x,y) & \text{otherwise} \end{cases}$$

where $\mathcal{F}(x,y) = \tanh(4(1-|x-y|)-2)$ is a measure of the difference between the presynaptic and postsynaptic activity. $\mathcal{F}(x,y) > 0$ if the difference is bigger or equal to 0.5 (half the maximum node activation) and $\mathcal{F}(x,y) < 0$ if the difference is smaller than 0.5.

The the self-limiting component $(1-w)$ is maintaining synaptic strengths within the range $[0,1]$. As a consequence, synapses do not change sign.

3 The "Reinforcement Learning" Task

In this first experiment CTRNNs and PNNs are compared on a task that requires acquisition and storage of knowledge on the basis of a reinforcement signal provided to the input units of the network. Given the simple experimental settings, the experiment is carried out in a realistic simulation by sampling infrared sensor activations (Miglino et al., 1995), computing geometric projections for the linear camera inputs, and adding 5% uniform noise to every value.

A simulated Khepera robot (figure 1) is positioned in a white rectangular arena with two areas of potential reward (figure 2), a light bulb to the left and a black vertical stripe on the right. The robot has 6 trials to find out where the reward area is located, go there and stay

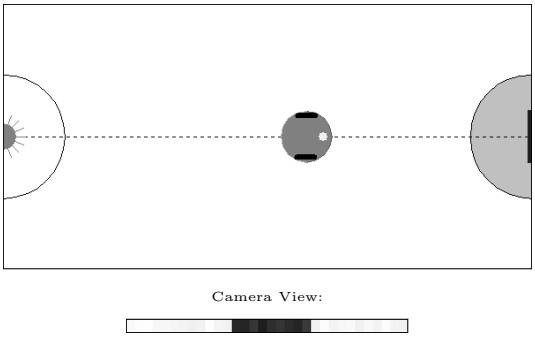

Figure 2: The environment used in the reinforcement learning task. To the left there is a light bulb and to the right there is black stripe on the wall. A gray reward-area can be randomly placed in either end (here to the right). The robot is constrained to move along the dashed line always facing the black stripe on the wall. Below the environment the view from the linear camera is shown.

over it. At the beginning, the position of the reward area (grey-filled sector in figure 2) is randomly chosen, either below the light bulb or below the stripe, and remains the same for 3 consecutive trials. After 3 trials, the reward area is switched to the other end of the environment. At the beginning of each trial, the robot is randomly positioned within the center third of the dashed line shown in figure 2, always facing the black stripe. In order to make the task simpler, the robot can only move along this line back and forth at variable speed, but cannot rotate.

The reinforcement signal comes from a floor-color sensor (figure 1, c) which is *on* when the robot is inside the gray reward-zone and *off* otherwise. Notice that this information is a sensory input just like others, in contrast to conventional reinforcement learning systems where the reward signal plays a special role in the architecture and in the learning algorithm (Sutton and Barto, 1998).

3.1 Network Architectures and Genetic Encoding

In order to maintain consistency with previous results reported in the literature, the architectures and genetic encodings of the CTRNNs and PNNs used are slightly different.

The architecture difference is that sensory receptors in CTRNNs receive information only from the sensors of the robot (figure 3), whereas in PNNs sensory neurons receive information from all other neurons in the network, including other sensory neurons and self-connections (figure 4).

In addition, the PNNs have a bias neuron, with activation fixed to 1, whose plastic connections act as a variable threshold on post-synaptic neurons. Instead, the bias values (or thresholds) of the CTRNNs are fixed

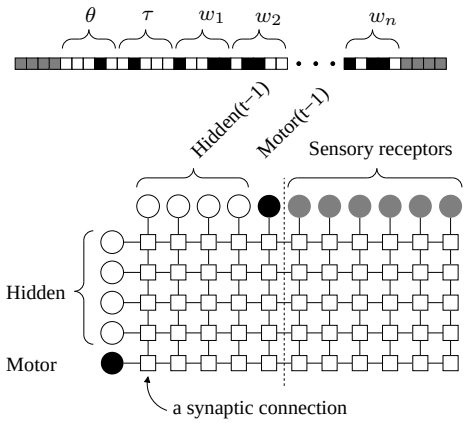

Figure 3: CTRNNs: Genetic encoding of the parameters for one neuron (top) and architecture of the CTRNN (bottom) used in the reinforcement learning task. *Genetic Encoding*: Each neuron parameter is encoded using 5 bits. θ is the bias, τ is the time constant, and $w_1 \dots w_n$ are the strengths of the incoming synapses to this neuron. *Neural Architecture*: The network consists of 5 neurons (4 hidden + 1 motor output). Every neuron has synaptic connections from all neurons and all sensory receptors. In total, the network has 55 synaptic connections.

and individually encoded in the genetic string (see description of genetic encoding below).

The networks have 6 sensory inputs, one from the each of the following receptors (figure 5):

- *2 Light receptors*: The robots infrared sensors in passive mode are used to measure the ambient light. Only the two sensors on the back of the robot are used.
- *3 Visual receptors*: The linear vision module has 64 equally spaced photoreceptors spanning a visual field of 36° (figure 1, b). The visual field is divided into 3 sectors and the average pixel-value (256 gray levels) in each sector is passed to the corresponding visual receptor.
- *1 Floor receptor*: An infrared sensor in the center of the robot pointing downwards (figure 1, c) measures the colour of the floor (in 256 gray levels). If the robot is inside the reward area the corresponding receptor is *on* and *off* otherwise.

The values from each receptor are scaled into the range $[0, 1]$. The activation of the motor neuron determines the speed of *both* wheels of the simulated robot. Activations above 0.5 correspond to forward motion, activations below 0.5 correspond to backward motion (the closer to 1 or 0, the faster the motion).

The parameters of both types of networks are encoded in genotype bitstrings. In the case of CTRNNs (figure 3, top), each neuron has 13 encoded parameters: a time constant (τ), a threshold (θ), and 11 synaptic strengths (w_{ij}). Each parameter is encoded using 5 bits giving a total genotype length of 325 bits.

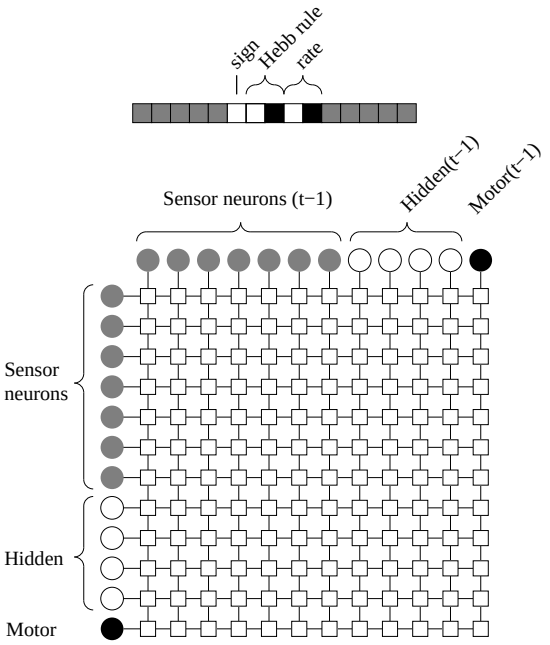

Figure 4: PNNs: Genetic encoding of the parameters for one neuron (top) and architecture of the PNN (bottom) used in the reinforcement learning task. *Genetic Encoding*: Each neuron is encoded using 5 bits. The first bit determines the sign of all outgoing synapses and the remaining four bits determine the Hebb rule (one out of four types) and learning rate (one out of four values) for all incoming synapses to that neuron. *Neural Architecture*: The network consists of 7 sensory input neurons, 4 hidden neurons and one motor output neuron. The network is fully recurrent giving a total of 144 synaptic connections.

In the case of PNNs (figure 4, top), each neuron has 3 encoded parameters: the sign bit determines the sign of all the outgoing synapses and the remaining four bits determines the properties of all incoming synapses to this neuron, (2 bits for one of four Hebb rules and 2 bits for one of four learning rates, namely $0.0, 0.3, 0.6, 1.0$). Consequently, all incoming synapses to a given node have the *same* properties. The total genotype length in this case is 60 bits.

3.2 Experiments

The experiments are carried out in simulation using a rank-based selection. A population of 100 neural controllers is evolved for 100 generations. At every generation the best 20 individuals make 5 copies each. One copy of the best individual remains unchanged (elitism). Single-point crossover with a 0.04 probability and bit-

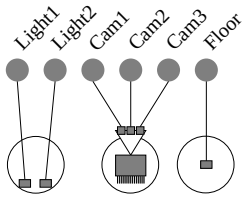

Figure 5: Configuration of the 6 sensory neurons used in the reinforcement learning task.

switch mutation with a 0.02 probability per bit are used. Every neural controller is tested for 3 times (epochs) of 6 trials each. Each trial lasts 150 sensory-motor cycles (one sensory-motor cycle corresponds to 100 ms on the real robot). At the beginning of each *epoch*, the neural controller is re-initialized. In the case of CTRNNs, the states of the neurons are set to 0 and in the case of PNNs the synaptic strengths are initialized to small random values in the interval $[0, 0.1]$ (recall that the sign of the synapse is given by the neuron from where the signal starts).

The fitness of the individual is proportional to the amount of time spent on the reward areas subtracted a penalty for spending time in only one of the two areas:

$$fitness = \frac{\sum_{i=1}^{epochs} f_1(i) + f_2(i) - |f_1(i) - f_2(i)|}{epochs \times trials\_per\_epoch \times steps\_per\_trial}$$

where f_1 is the number of steps spent in reward area in trials with the reward area to the left and f_2 is the number of steps spent in reward area in the trials with the reward area to the right. Notice that if f_1 or f_2 is zero in an epoch the total fitness is zero in that epoch. Without taking the absolute difference between f_1 and f_2, evolved controllers find the sub-optimal solution of only accumulating fitness points in one end of the environment.

For each type of neural controller (CTRNN and PNN) the experiment is repeated 10 times with different initializations of the pseudo-random number generator. The results of the experiments using CTRNNs are plotted in figure 6. Only 40 generations are required for the fitness values to reach a stable level.

The typical behavior of an evolved controller is visualized in figure 7. The x-position of the robot over time is plotted for an entire epoch of 6 trials [4].

Before the first step of each trial the robot is randomly placed within the center third of the x-axis. This evolved robot begins to move to the left where the reward area is positioned for the first 3 trials for this individual. After 3 trials the reward area is moved to the right side. The robot still moves to the left by default, but when it

[4] Each trial is shortened from 150 to 100 steps in figure 7 because the position of the robot always stabilizes within this timeframe for this individual.

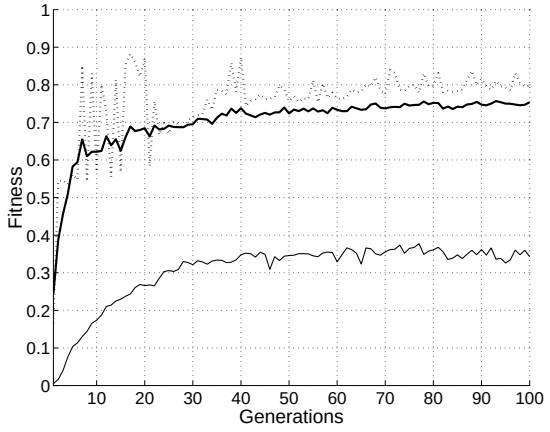

Figure 6: *Reinforcement Learning Task, CTRNNs.* Thick line = best individual; thin line = population average; dotted line = best individual of best replication.

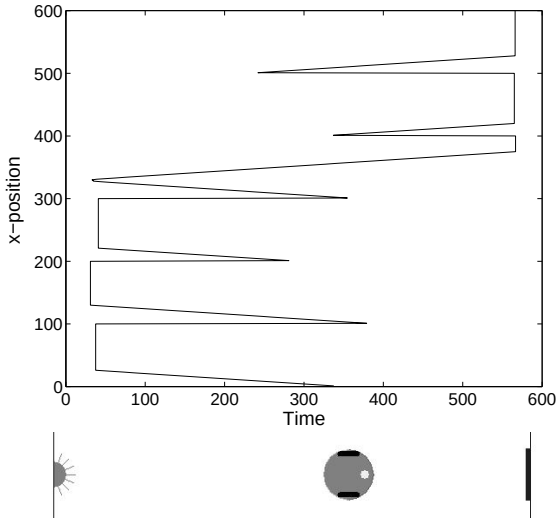

Figure 7: Typical behavior of an evolved robot in the reinforcement learning task. *Top*: Robot x-position is plotted against time over 6 trials. *Bottom*: Environment layout. See text for description.

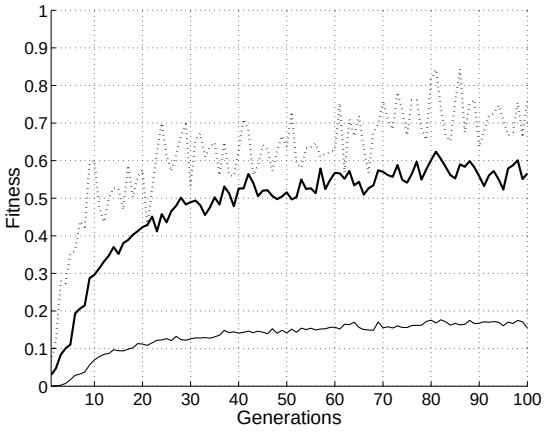

Figure 8: *Reinforcement Learning Task, PNNs.* Thick line = best individual; thin line = population average; dotted line = best individual of best replication.

discovers that the reward signal remains off, it reverses direction and moves towards the right side. For the remaining trials, it "remembers" that the reward area is on the right side and always moves in that direction.

The fitness data for the experiments with PNNs are plotted in figure 8 and the behavior of the best evolved individuals are similar to the one shown in figure 7. The fitness values in these experiments are lower for two reasons. The first reason is that PNNs must develop synaptic weights at the beginning of each epoch (which takes time and has an indirect fitness cost) whereas CTRNNs are functional from the very beginning of the epoch. The second reason is that only one replication of the experiment was successful at evolving individuals capable of solving the task reliably. In the other replications the succes of the best individuals depended on the initial position of the robot and the random initializtion of the synaptic weights resulting in a drop in performance under some *unlucky* initial conditions.

4 Adaptation to Changes

In order to further explore the adaptation capabilities of the two types of networks, an experimental setup investigated in (Floreano and Urzelai, 2000, Urzelai and Floreano, 2001) is now used. The environment (figure 9) is identical to the one used in the reinforcement learning task, but the position of the gray fitness area is now fixed under the light bulb and a black lightswitch area has been added under the black stripe. Initially the light is off, but it is switched on if the robot passes over the black area. The task of the robot is to spend as much time as possible under the light bulb when the light is on. Therefore, the robot must first discover how to switch on the light. Each individual of the population is tested for 3 trials of 500 sensory-motor cycles (100 ms) each. The experiments are carried out in simulation and some evolved individuals are then tested in the real environment (evolutionary experiments on the real robot with PNNs are reported in (Urzelai and Floreano, 2001) and the results are similar to those obtained in simulation).

Figure 9: *The Lightswitch Task*: The Khepera robot can gain fitness by staying on the gray area when the light is on. Initially the light is off, but the robot can switch it on by passing first over the black area. The neural controller has no information from the floor-sensor which is relayed to an external computer in order to automatically switch the light on and compute fitness values.

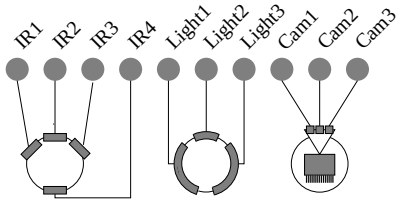

Figure 10: Configuration of the 10 sensory neurons used in the lightswitch task.

The fitness function is given by the number of sensory motor cycles spent on the gray area below the light bulb *when the light is on* divided by the total number of cycles available (500). In order to maximize this fitness function, the robot should find the lightswitch area, go there in order to switch the light on, and then move towards the light as soon as possible, and remain on the gray area.[5] Since this sequence of actions takes time (several sensory motor cycles), the fitness of a robot will never be 1.0. A robot that cannot manage to complete the entire sequence will get zero fitness.

The architectures of the two types of neural networks are identical to those described earlier, but the configuration of the sensory receptors and motor output neurons is adapted to this new task. The number of sensory receptors is extended to 10: 4 infrared proximity receptors to detect distance form walls (within a range of 5 cm),

[5]Notice that the fitness function does not explicitly reward this sequence of actions, but only the final outcome of the overall behavior.

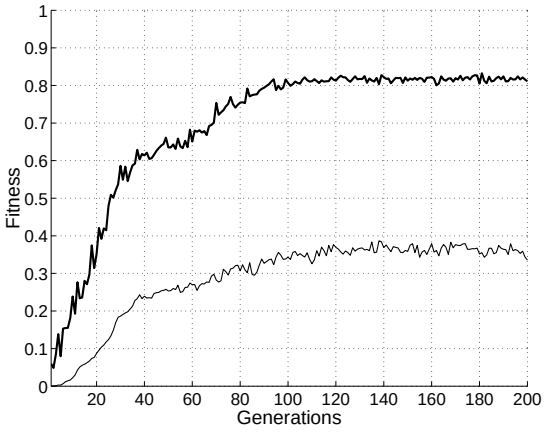

Figure 11: *Ligthswitch Task, CTRNNs.* Thick line = best individual; thin line = population average.

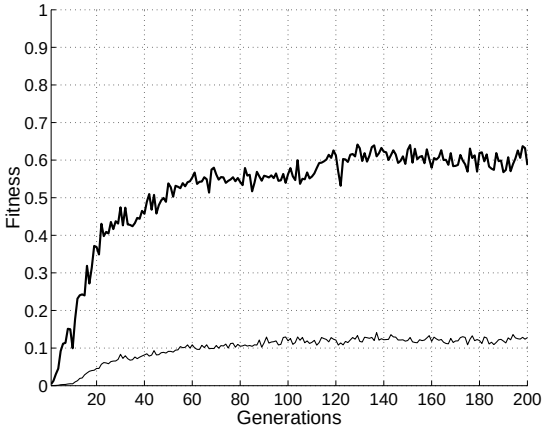

Figure 12: *Ligthswitch Task, PNNs.* Thick line = best individual; thin line = population average.

3 light receptors, and 3 visual receptors (figure 10).

4.1 Experiments

Two sets of experiments are run, one for each type of network. The results are shown in figures 11 and 12. In both cases, evolved controllers are capable of performing the entire sequence of actions. The reason why the PNNs report lower fitness values is that synaptic adaptation takes time and therefore has an implicit fitness cost. In addition, evolved PNNs tend to select lower motor speeds for the robot. However, all experimental runs end up with successful strategies.

A typical trace of a successful robot is shown in figure 13. The robot turns on the spot, moves directly towards the black area to turn on the light and finally moves towards the light bulb and remains over the gray fitness area. Small differences in the behaviors generated by the two types of networks are noticed. Robots controlled by CTRNNs generally turn and move directly towards the

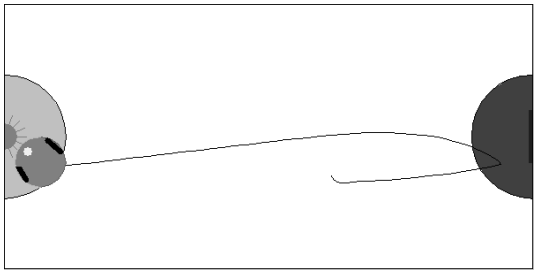

Figure 13: Typical trace of an evolved robot in the lightswitch task.

black area at first. Instead, robots controlled by PNNs often start out by first moving forward hitting the first wall and afterwards doing a backward turn while orienting towards the black stripe and approaching the black area. Recall the PNNs start out with randomly initialized synapses and must develop all abilities, including obstacle avoidance from scratch.

4.2 Environmental Changes

In contrast to the reinforcement learning task, this task requires more interactions between the robot and the environment and includes a larger number of sensory-motor correlations that must be taken into account by evolutionary neural controllers. Therefore, it is better suited for applying a number of modifications *after evolution* to test on-line adaptive abilities of evolved CTRNNs and PNNs.

In order to measure the performances of evolved controllers in environments with new characteristics, the best individual of the last generation for each of the 10 replications is tested in environments with white (as used during evolution), gray, and black walls. The responses of infrared proximity sensors are lower for darker walls, which requires the robot to avoid walls at lower sensory activation levels than with brighter walls.

Figure 14 shows the average fitness values of the best individuals in environments with white, gray and black walls for CTRNNs (left) and PNNs (right). In the CTRNN case, the performance drops significantly in the gray environment case and dramatically in the black environment. In the gray environment, evolved individuals are only able to complete the correct sequence of actions in one third of the trials and in the black environment this number drops to one out of 20 trials. In the PNN case a performance drop is observed in gray and black environments, but this is not so drastic when compared to the performance in white environments. Most importantly, PNN individuals can still solve the task but require longer time to do so because they interact with the walls for longer time. Evolved PNNs are capable of adapting to new sensory situations even in the black environment while CTRNNs cannot cope with it.

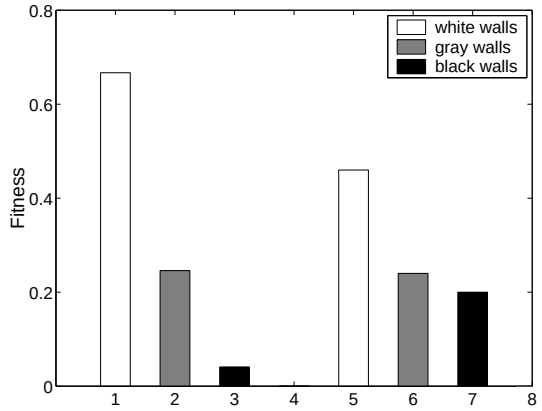

Figure 14: Performance of the evolved CTRNNs (left) and PNNs (right) tested in environments with white, gray and black walls. Each fitness value is the average over 100 tests (the best individual of each of the 10 replications is tested 10 times each).

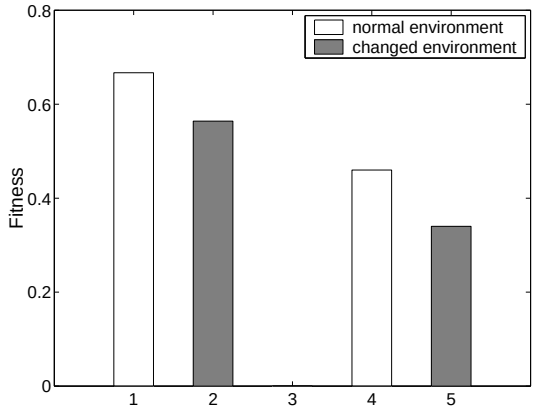

Figure 15: Performance of evolved CTRNNs and PNNs tested in the environment used during evolution (white bar) and in an environment where the lightswitch area and the fitness area are positioned randomly along the walls (gray bar). Each fitness value is the average over 100 tests (the best individual of each of the 10 replications is tested 10 times each).

In a second set of tests, the spatial relationships of the objects in the environment are modified. For each trial, the lightswitch area and the light bulb area is randomly located along the walls (discarding combinations with area overlap). The best individual of the last generation for each of the 10 replications is tested in 10 randomly generated environments. The results are reported in figure 15. To the left are the fitness values for individuals controlled by CTRNNs and to the right the fitness values for the individuals controlled by PNNs. In both cases there is a small drop in performance in the new environments, but both CTRNNs and PNNs succeed in solving the task. In other words, this modification does not show any difference in the adaptation capabilities of the two types of networks.

As a final test, the neural controllers (which were evolved in simulation) are tested in the real environment on a real Khepera robot (see figure 9). Despite the use of a realistic simulation, there is still a significant difference in sensory and motor properties of simulated and physical robots. The best individual of the last generation in each of the 10 replications is tested 3 times on the physical robot. Figure 16 shows that CTRNNs fail on the real robot, while PNNs are less affected by the transfer and still function in the real environment. The small performance drop in the PNN case is due to the fact that the robot sometimes performs looping trajectories around the fitness area without coming to rest. The main reason for the failure of the CTRNNs is that the robots not are able to reliably approach the black stripe on the wall. These networks are not capable of adapting to changed sensory-motor conditions and the robot often gets attracted to shadows or keeps spinning and bumping into walls.

5 Discussion

The experimental results presented in this paper question the definition of learning. From a behavioral perspective, learning is usually associated with acquisition of new knowledge, skills, and memory retention (Gallistel, 1990). From a biological and computational perspective, learning is associated to synaptic change (Churchland and Sejnowski, 1994). However, the experiments described in this paper show that both assumptions are not necessary and unique. On the one hand, dynamical neural networks without synaptic plasticity display reinforcement learning behavior (confirming earlier experiments described in (Yamauchi and Beer, 1994)). On the other hand, networks with plastic synapses do not necessarily retain previously acquired knowledge (data regarding this latter point are described in (Urzelai and Floreano, 2001)).

The experiments show that each type of network model – CTRNN and PNN – is capable of displaying learning-like abilities and can solve complex tasks which require sequential behavior. However, the PNNs display a clear advantage when confronted with environmental conditions not seen during the evolutionary process. In the two tests on unpredictable changes where PNNs scored significantly better than CTRNNs – walls of different brightness and transfer from simulated to physical robots – both sensory and motor adaptation is required. On the contrary sensory-motor adaptation is not necessary in the tests with spatial reconfiguration of the environment where the same stripe-directed navigation and light following behaviors as developed during evolution is efficient in several environments. In this latter test, both network models indeed display the same

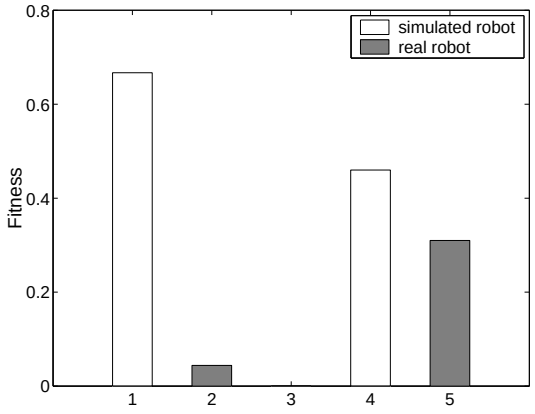

Figure 16: Performance of evolved CTRNNs and PNNs tested in the simulated robot used during evolution (white bar) and in the real Khepera robot (gray bar). Each fitness value is an average over 30 tests (the best individual of each of the 10 replications is tested 3 times each).

performance.

Correlation-based synaptic plasticity therefore seems useful to maintain coherent sensory-motor mappings in a variety of different environments. Considering the importance of sensory-motor coordination in even the most primitive organisms equipped with neural cells, one may (wildly) speculate that correlation-based (Hebbian) synaptic plasticity was discovered by evolution as a simple homeostatic mechanism (Ashby, 1960) to cope with partially changing and unpredictable environments. However, one may argue that the PNN architecture used in these experiments provides an advantage with respect to this because the sensory neurons are interconnected and therefore may better capture and maintain sensory-motor correlation than in the CTRNN case where correlations must be captured and maintained by hidden units.

In a previous work (Urzelai and Floreano, 2001), it was shown that PNNs rapidly change most synaptic strengths when the sensory information changes significantly. This fast re-wiring of the entire network is quite efficient in the simple sensory-motor tasks described in this paper, but may not be suitable for more complex situations where there is a survival advantage in acquiring and retaining complex skills or spatial configurations. Indeed, not all evolutionary runs with PNNs were capable of solving completely the reinforcement-learning task described here. A promising direction consists of adding mechanisms for enabling and disabling synaptic plasticity of the Hebb-rules described here. This idea has been recently explored to evolve reinforcement learning-like foraging (Niv et al., 2001), adapt to distorted optical information (DiPaolo, 2000), and adapting walking patterns to terrains of variable inclination (Fujii et al., 2001). However, the tasks explored in those works are not significantly more complex than those described here and it can be argued whether the enabling/disabling mechanism is necessary. Ironically, a serious problem seems to be the definition of sufficiently complex learning problems that require more than "minimally cognitive behaviors" and yet remain sufficiently simple for analysis.

As a final note, the size of the populations of CTRNNs in the experiments described here is one order of magnitude smaller than the one used in the experiments by (Yamauchi and Beer, 1994). Although the two types of networks are slightly different in the sensory-motor interface, the major difference is, that in the work presented in this paper, binary, instead of real-valued, genetic encoding is used. Consequently, the search space is much smaller, but this does not seem to limit the rich dynamics of the networks.

6 Conclusion

Two classes of dynamical recurrent neural networks, CTRNNs and PNNs, have been compared on two behavioral tasks aimed at exploring their capabilities to display reinforcement-learning like behaviors and adaptation to unpredictable environmental changes. Although both networks displayed similar performances (slightly offset in PNNs by the time required for synaptic development), PNNs displayed significantly better performance when sensory-motor re-adaptation is required after the evolutionary process.

These results have been discussed within the perspective of behavioral, biological, and computational definitions of learning. It has been argued that correlation-based synaptic plasticity may play a major role in developing and maintaining sensory-motor coordination in partially changing and unpredictable environments.

Although the combination of the two models may have some interest, we believe that more complex behavioral and cognitive abilities may be achieved by employing mechanisms for enabling and disabling synaptic plasticity in environmental conditions that require the development and retention of sensory-motor skills and spatial memories.

Acknowledgements

The authors thank Inman Harvey (Sussex University), Ezequiel Di Paolo (Sussex University), and Claudio Mattiussi (EPFL) for fruitful discussions and suggestions. This work was supported by the Swiss National Science Foundation, grant nr. 620-58049.

References

Ashby, W. R. (1960). *Design for a Brain: The Origin of Adaptive Behavior*. Chapman and Hall, London.

Beer, R. D. and Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122.

Churchland, P. M. and Sejnowski, T. J. (1994). *The Computational Brain*. MIT Press, Cambridge, MA.

DiPaolo, E. A. (2000). Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions. In Meyer, J., Berthoz, A., Floreano, D., Roitblat, H., and Wilson, S., (Eds.), *From Animals to Animats VI: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, pages 440–449. MIT Press-Bradford Books, Cambridge, MA.

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, 14(2):179–211.

Floreano, D. and Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:396–407.

Floreano, D. and Urzelai, J. (2000). Evolutionary robots with online self-organization and behavioral fitness. *Neural Networks*, 13(4–5):431–443.

Fujii, A., Ishiguro, A., Aoki, T., and Eggenberger, P. (2001). Evolving bipedal locomotion with a dynamically-rearranging neural network. In Kelemen, J. and Sosík, P., (Eds.), *Advances in Artificial Life (ECAL 2001)*, pages 509–518, Berlin. Springer Verlag.

Gallistel, C. R., (Ed.) (1990). *The Organization of Learning*. MIT Press, Cambridge, MA.

Harvey, I., Husbands, P., and Cliff, D. (1994). Seeing the light: Artificial evolution, real vision. In Cliff, D., Husbands, P., Meyer, J., and Wilson, S. W., (Eds.), *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 392–401. MIT Press-Bradford Books, Cambridge, MA.

Hines, M. and Carnevale, N. T. (1998). Computer modeling methods for neurons. In Arbib, M. A., (Ed.), *The Handbook of Brain Theory and Neural Networks*, pages 226 – 230. MIT Press – Bradford Books, Cambridge, MA.

Hopfield, J. J. (1984). Neurons with graded response properties have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences USA*, 81:3088–3092.

Miglino, O., Lund, H. H., and Nolfi, S. (1995). Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life*, 2(4):417–434.

Niv, Y., Joel, D., Meilijson, I., and Ruppin, E. (2001). Evolution of reinforcement learning in uncertain environments: Emergence of risk-aversion and matching. In Kelemen, J. and Sosík, P., (Eds.), *Advances in Artificial Life (ECAL 2001)*, pages 252–261. Springer Verlag, Berlin.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA.

Sutton, R. and Barto, A. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA.

Tani, J. (1996). Model-Based Learning for Mobile Robot Navigation from the Dynamical Systems Perspective. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:421–436.

Urzelai, J. and Floreano, D. (2001). Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evolutionary Computation*, 9:495–524.

Yamauchi, B. and Beer, R. D. (1994). Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior*, 2(3):219–246.