The Evolution of Self-taught Neural Networks in a Multi-agent Environment

Nam Le (✉), Anthony Brabazon, and Michael O'Neill

Natural Computing Research & Applications Group
University College Dublin
Corresponding email: namlehai90@gmail.com

Abstract. Evolution and learning are two different forms of adaptation by which the organism can change their behaviour to cope with problems posed by the environment. The second form of adaptation occurs when individuals exhibit plasticity in response to environmental conditions that may strengthen their survival. Learning has been shown to be beneficial to the evolutionary process through the **Baldwin Effect**. This line of thought has also been employed in evolving adaptive neural networks, in which learning algorithms, such as Backpropagation, can be used to enhance the adaptivity of the population. Most work focuses on evolving learning agents in separate environments, this means each agent experiences its own environment (mostly similar), and has no *interactive effect* on others (e.g., the more one gains, the more another loses). The competition for survival in such settings is not that strong, if being compared to that of a *multi-agent* (or shared) environment . This paper investigates an evolving population of *self-taught* neural networks – networks that can teach themselves – in a shared environment. Experimental results show that learning presents an effect in increasing the performance of the evolving multi-agent system. Indications for future work on evolving neural networks are also presented.

Keywords: The Baldwin Effect · Neural Networks · Neuroevolution · Meta-learning · Self-learning.

1 Introduction

For many biological organisms, adaptation is necessary for survival and reproduction in an uncertain environment. There are two important kinds of adaptation that should be distinguished. The first is a change at the genetic level of a population, in which organisms reproduce selectively subject to mechanisms, like mutation or sexual recombination, which maintain inter-individual variability. This is usually modeled in terms of biological evolution, which causes changes in the population from one generation to the next. The second adaptation mechanism, on the other hand, is the phenotypic change at the individual level. This can be called *lifetime-adaptation* which changes the phenotypic behaviour of the organism during its lifetime. Plausibly, lifetime adaptation happens at a quicker pace than the evolutionary process which takes place through the generational timescale, preparing the organism for rapid uncertain environments.

The idea that learning can influence evolution in a Darwinian framework was discussed by psychologists and evolutionary biologists over one hundred years ago ([1, 2]), through *'A new factor in evolution'* called **the Baldwin Effect**. However, it gradually gained more attention since the classic paper by Hinton and Nowlan ([3]) which demonstrated an instance of the Baldwin effect in computer simulation. This line of research motivated the idea of evolving neural networks, or *neuroevolution* (NE), in which one can observe learning and evolution interacting with each other in creating adaptive neural networks [4], [5], [6]. Regardless of how learning is implemented, most studies focus on how learning helps evolution to solve an individual problem – this means each agent solves its own problem (and their problems are copies of each other) – there is no mutual interactive effect between learning agents when they learn and compete for survival during their lifetime.

The main aim of this paper is to investigate, through computer simulation, how learning and evolution can provide adaptive advantage in a multi-agent system. We design a neural architecture allowing for lifetime learning, more specifically for the ability to teach oneself. To show the effect of self-taught agents we simulate a simple environment called MiniWorld in which agents have to compete with each others for food in order to survive. In the remainder of this contribution, we initially present some prior research on learning and evolution, including brief literature review on learning and evolving neural networks. We then describe the experiments undertaken. The results from these experiments are analysed and discussed, and finally, conclusions and several interesting future research opportunities are proposed.

2 Learning and Evolution

2.1 The Baldwin Effect

Most organisms need to learn to adapt to their changing environments. Learning is essential for both living organisms, including humans and animals, and artificial learning agents. Organisms, living or digital, need to learn new behaviors to solve tasks that cannot be solved effectively by innate (or genetically encoded) behaviors. Learning shows its powerful advantages when the environment changes [7]. During the lifetime of an individual organism, if its environment changes in a way that its previous knowledge cannot be enough to survive, it has to learn new behaviors or knowledge to adapt to the novel circumstance. Otherwise, it will be unfit, thus having less chance to *survive*.

It is very interesting that the Baldwin Effect was first demonstrated computationally [3] around 20 years before it was first empirically verified in nature [8]. In 1987, the Cognitive Scientist Geoffrey Hinton and his colleague Kevin Nowlan at CMU presented a classic paper [3] to demonstrate an instance of the Baldwin effect in the computer. Hinton and Nowlan (henceforth H&N) used a genetic algorithm [9] to evolve a population in an extreme landscape called *Needle-in-a-haystack*, showing that learning can help evolution to search for the solution when evolution alone fails to do so. An interesting idea that can be extracted from their work is that instead of genetically fixing the genotype, it is wiser to let just a portion of the genotype genetically fixed, and the other be *plastic* that allows for changes through learning. It is these plastic individuals that promote the evolutionary process to search for the optimal solution, although the H&N landscape is static.

The model developed by Hinton and Nowlan, though simple, is interesting, opening up the trend followed by a number of research papers investigating the interaction between learning and evolution. Following the framework of Hinton and Nowlan, there have been a number of other papers studying the Baldwin effect in the NK-fitness landscape, which was developed by Stuart Kauffman [10] to model 'tunably rugged' fitness landscapes. Problems within that kind of landscape are shown to fall in NP-completeness category [10]. Several notable studies of the Baldwin effect in the NK-model include work by Giles Mayley [11], and some others [12]. Their results, again, demonstrated that the Baldwin effect does occur and the allowance for lifetime learning, in the form of individual learning, helps evolutionary search overcome the difficulty of a rugged fitness landscape.

2.2 Learning and Evolution in Neural Networks

There have also been several studies investigating the interaction between learning and evolution in Neural Networks. Most use the so-called *Neuroevolution* approach to test if neural network learning facilitates an evolutionary process, often represented by an evolutionary algorithm. Some notable papers on this line of research include [13], and several works by Stefano Nolfi, Domenico Parisi on Evolutionary Robotics [4], [6], to name but a few. All of these papers attempted to confirm the existence of the Baldwin effect, by showing how learning interacts with evolution making the system perform better than with evolution alone.

Todd and Miller [14] investigated an imaginary underwater environment in which organisms have to eat food and avoid poison. Each agent was born in one of the two feeding patches, and has to decide whether to consume substances floating by. Those substances can be either food or poison, with colour either red or green. The association between the colour (red or green) and the substance (food or poison) varies between feeding patches. Therefore, an agent has to decide whether to eat or pass a substance based on its sensory experience. Moreover, there is no feedback given to an individual agent that could be used to discriminate between food and poison [14]. Todd and Miller showed that the combination of evolution and learning in the form of *Hebbian Rule* [15] in a simple neural network can do better than both evolution and learning alone in this scenario. One point to be noted here is that this is an imaginary environment, there is no movement in the environment by the agent. The story here can be understood as a population of *disembodied* neural network learning to classify the representation of food and poison.

Another study on this topic includes [13] in which neural networks have to classify food and poison. The neural synaptic connections are both evolved using an evolutionary algorithm and learned by the *Delta Rule* (a simple form of Backpropagation on single layer network). It was also shown that learning enhances the evolving population in terms of food-poison classification. Like the work by Todd and Miller, there is no movement in the environment or the interaction between the neural network and the environment containing food and poison. This is also a disembodied case.

Nolfi and his colleagues simulated a population of simulated embodied *animats* (or artificial organisms) controlled by neural networks situated in a grid-world, with four actions (turn 90° right or left, move one cell forward, or stay still in the current cell). The evolutionary task is to evolve action strategies to collect food effectively, while each agent learns to predict the sensory inputs to neural networks for each time step. Learning was implemented using Backpropagation based on the error between the actual and the predicted sensory inputs. It was shown that learning to predict is able to enhance the evolutionary process and increase the fitness of the population. The same observation was also validated as true when learning performs a XOR function ([6]. In these studies, each animat, or simulated embodied agent, lives in its own copy of the world. Therefore, there is no interaction between these agents while they are foraging during their lifetime.

There have also been several other studies using the idea of learning and evolution in neural networks, including [16], [17], [18], showing how the interplay between learning and evolution in evolving neural networks enhances the performance the evolving system. Yet most of the work use *disembodied* and *unsituated* neural networks or there is no interaction between learning agents in the environment – they live in their own copies of the environment, solving their own problems, having no effect on the performance of others. Please refer to [19] for some more recent studies on evolving plastic neural networks (neural networks that learn).

In this study, we investigate the interaction between learning and evolution in a multi-agent system – a system containing multiple situated agents living together and doing their tasks while competing with each other. Each agent is controlled by a neural network but situated (and has a *soft-embodiment*). This means, the way an agent acts and moves in the world affects the subsequent sensory inputs, hence the future behaviour of that agent. We also propose an architecture called self-taught neural network – neural network that can teach itself during the lifetime interaction with the environment. Simulations to investigate between evolution and learning in evolving self-taught neural networks are described in the following section.

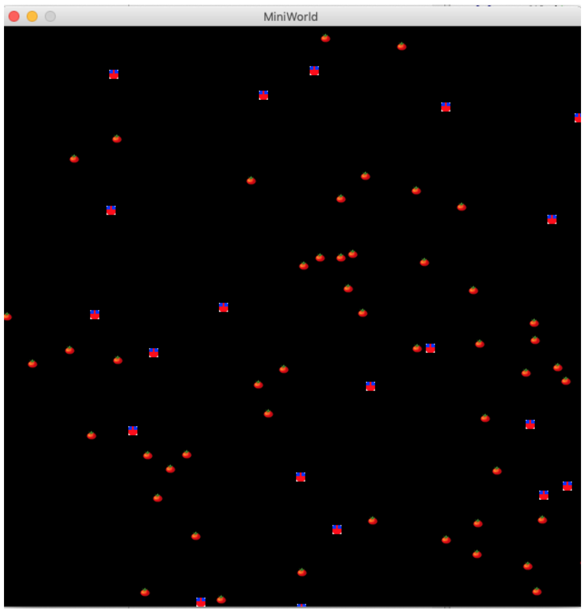

Fig. 1. MiniWorld – The environment of agents and food, 640x640.

3 Simulation setup

3.1 The simulated world

We simulate a continuous 2D-world, called **MiniWorld**, with dimension 640x640. It contains 50 food particles randomly uniformly distributed. Each piece of food is represented by a squared image with size 10x10, locating at a random position in range [0, 640] for each dimension. Each food has an energy value. For simplicity, in our simulation we set the energy value of 1 for every food particle. Because the state-space of the world is continuous, not discrete like that of a grid space, and the size of food is much smaller than the dimension of the world, the dispersal of food is sparse enough to complexify the foraging task investigated in this paper. One property of MiniWorld is it has no strict boundary, and we implement the so-called *toroidal* – this means when an agent moves beyond an edge, it appears in the opposite edge. The visualisation of the sample world is shown in Figure 1.

3.2 Agent

We also simulate an evolving population of 20 agents in MiniWorld. Each agent is represented by an squared image with size 10x10, the same size as the food particle. Each agent has a food counter representing the total energy the agent

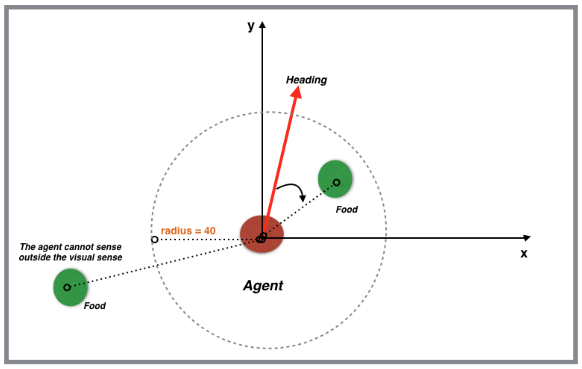

Fig. 2. Agent situated in an environment seeing food

got during its life. The food counter of an agent increases by the energy level of the food that the agent eats (hence by 1 in our simulation). When an agent happens to collide with a food patch, the food is eaten and another food piece randomly spawns in the environment but at a different random location. The collision detection criterion is specified by the distance between the two bodies. This shows how the soft-body affects the movement and the performance of an agent in its environment. By the re-appearance of food, the environment changes as an agent eats a food.

Every agent has a heading (in principle) of movement in the environment. In our simulation, we assume that every agent has a *priori* ability to sense the angle between its current heading and the food if appearing in its visual range. The visual range of each agent is a circle with radius 4. Each agent takes as inputs three sensory information, which can be the binary value 0 or 1, about what it sees from the left, front, and right in its visual range. If there is no food appearing in its visual range, the sensory inputs are all set to 0. If there is food appearing on the left (front, or right), the left (front, or right) sensor is set to 1; otherwise, the sensor is 0.

Let θ (in degree) be the angle between the agent and the food particle in its visual sense. An agent determines whether a food appears in its left, front, or right location in its visual range be the following rule:

$$\begin{cases} 15 < \theta < 45 & => right \\ \theta \leq 15 \text{ or } \theta \geq 345 & => front \\ 315 < \theta < 345 & => left \end{cases}$$

A examplar visualisation of an agent and its relationship with food in the environment is shown in Figure 2.

Unlike [4] [6] in which each agent has its own copy of the world, we let all agents live in the same MiniWorld. They feed for their own survival during their life. The more an agent eats, the less the chance for others to feed themselves.

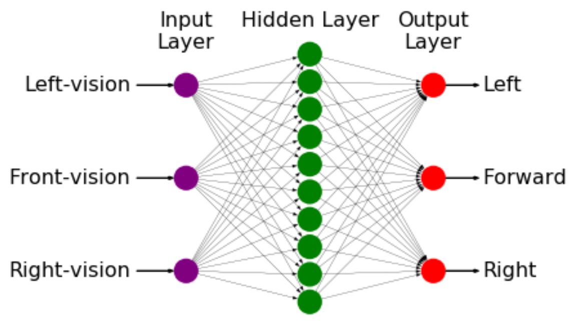

Fig. 3. Neural network controller for each situated agent. Connection weights can be created by evolutionary process, but can also be changed during the lifetime of an agent.

This creates a stronger competition in the population. When an agent moves for foraging, it changes the environment in which other agents live, changing how others sense the world as well. This forms a more complex dynamics, even in simple scenario we are investigating in this paper.

The default velocity (or speed) for each agent is 1. Every agent has three basic movements: Turn left by 9 degrees and move, move forward by double speed, turn right by 9 degrees and move. For simplicity, these rules are pre-defined by the system designer of MiniWorld. We can imagine the perfect scenario like if an agent sees a food in front, it doubles the speed and move forward to catch the food. If the agent sees the food on the left (right), it would like to turn to the left (right) and move forward to the food particle. The motor action of an agent is guided by its neural network as described below.

3.3 The neural network controller

Each agent is controlled by a fully-connected neural network to determine its movements in the environment. What an agent decides to do changes the world the agent lives in, changing the next sensory information it receives, hence the next behaviour. This forms a sensory-motor dynamics and a neural network acts as a situated cognitive module having the role to guide an agent to behave adaptively, or *Situated Cognition* even in such a simple case like what is presenting in this paper. Each neural network includes 3 layers with 3 input nodes in input layer, 10 nodes in hidden layer, and 3 nodes in output layers.

The first layer takes as input what an agent senses from the environment in its visual range (described above). The output layer produces three values as a motor-guidance for how an agent should behave in the world after processing sensory information. The maximum value amongst these three values is chosen as a motor action as whether an agent should turn left, right, or move forward (as described above). All neurons except the inputs use a sigmoidal activation function. All connections (or synaptic strengths) are initialised as Gaussian(0, 1). These weights are first initialised as *innate*, or merely specified by the genotype of an agent, but also have the potential to change during the lifetime of that agent.

The architectural design of neural network controller is visualised in Figure 3. In fact, the neural architecture as shown in Figure 3 has no ability to learn, or to teach itself. In the following section, we extend this architecture to allow for self-taught learning agents.

3.4 The Self-taught neural architecture

To allow for self-taught ability, the neural controller for each agent now has two modules: one is called **Action Module**, the other is called **Reinforcement Module**. The action module has the same network as previously shown in Figure 3. This module takes as inputs the sensory information and produces reinforcement outputs in order to guide the motor action of an agent. The reinforcement module has the same set of inputs as the action module, but possesses separate sets of hidden and output neurons. The goal of reinforcement network is to provide *reinforcement signals* to guide the behaviour of each agent. The topology of a neural network in this case is visualised in Figure 4.

The difference between the output of the reinforcement module and the action module is used as the error of the output behaviour of the action module. That error is used to update the weights in action modules through Backpropagation [20]. Through that learning process, the action module approximates its output activation towards the output of the reinforcement module. In fact, the reinforcement and the action modules are not necessary to have the same topology. For convenience, in our simulation we allow the reinforcement module possesses the same neuronal structure as the action module, but has 10 hidden neurons separate from the hidden neurons of the action module, hence the connections. The learning rate is 0.01.

In the following sections, we describe simulations we use to investigate the evolutionary consequence of lifetime learning.

3.5 Simulation 1: Evolution alone (EVO)

In this simulation, we evolving a population of agents without learning ability. The neural network controller for each agent is the one described in Figure 3. The *genotype* of each agent is the weight matrix of its neural network, and the evolutionary process takes place as we evolve a population of weights, a common approach in Neuroevolution (NE) [21].

Selection chooses individuals based on the number of food eaten in the foraging task employed as the fitness value. The higher the number of food eaten, the higher the fitness value. For crossover, two individuals are selected as parents,

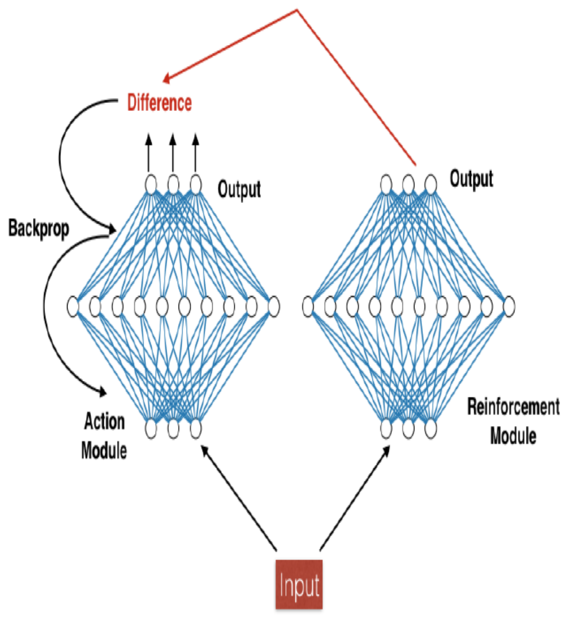

Fig. 4. Self-taught neural network.

namely $parent_1$ and $parent_2$. The two selected individuals produce one offspring, called *child*. We implement crossover as the more the successful of a parent, the more the chance its weights are copied to the child. The weight matrices of the child can be simply described as the algorithm below.

Algorithm 1 Crossover

```
1: function CROSSOVER($parent1, parent2$)
2:     $rate = parent1.fitness/parent2.fitness$ comment: fitness ratio
3:     $child.weights = copy(parent2.weights)$
4:     for $in \in len(child.weights)$ do
5:         if $random() < rate$ then
6:             $child.weights[i] = parent1.weights[i]$
7:         end if
8:     end for
9: end function
```

Once a child has been created, that child will be mutated based on a predefined *mutation rate*. In our work, *mutation rate* is set to 0.05. A random number is generated, if that number is less than *mutation rate*, mutation occurs, and vice versa. If mutation occurs for each weight in the child, that weight is added by a random number from the range [-0.05, 0.05], a slight mutation. After that, the newly born individual is placed in a new population. This process is repeated until the new population is filled 100 new individual agents. No elitism is employed in our evolutionary algorithm.

The population goes through a total of 100 generations, with 10000 time steps per generation. At each time step, an agent does the following activities: Receiving sensory inputs, computing its output, updating its new heading and location. In evolution alone simulation, the agent cannot perform any kind of learning during its lifetime. After that, the population undergoes selection and reproduction processes.

3.6 Simulation 2: Evolution of Self-taught agents (EVO+Self-taught)

In this simulation, we allow lifetime learning, in addition to the evolutionary algorithm, to update the weights of neural network controllers when agents interact with the environment. We evolve a population of **Self-taught** agents – agents that can teach themselves. The self-taught agent has a self-taught neural network architecture as described previously and shown in Figure 4. During the lifetime of an agent, the reinforcement modules produce outputs in order to guide the weight-updating process of the action module. Only the weights of action modules can be changed by learning, the weights of reinforcement module are genetically specified in the same evolutionary process as specified above in Evolution alone simulation.

We use the same parameter setting for evolution as in EVO simulation above. At each time step, an agent does the following activities: Receiving sensory inputs, computing its output, updating its new heading and location, and updating the weights in action module by **self-teaching**. After one step, the agent updates its fitness by the number of food eaten. After that, the population undergoes selection and reproduction processes as in Evolution alone.

Remember that we are fitting learning and evolution in a Darwinian framework, not Lamarckian. This means what will be learned during the lifetime of an agent (the weights in action module) is not passed down onto the offspring. Results and analysis are described in the section below.

3.7 Simulation 3: Random Self-taught agents (Random-Self-taught)

We conduct another simulation in which all agents are self-taught agents – having self-taught networks that can teach themselves during lifetime. What differs from simulation 2 is that at the beginning of every generation, all weights are randomly initialised, rather than updated by an evolutionary algorithm like in simulation 1. The learning agents here are initialised as *blank-slates*, or *tabula rasa*, having no predisposition to learn or some sort of *priori knowledge* about the world. The reason for this simulation is that we are curious whether evolution brings any benefit to learning in MiniWorld. In other words, we would like to see if there is a synergy between evolution and learning, not just how learning can affect evolution.

Experimental results are presented and discussed in the following section.

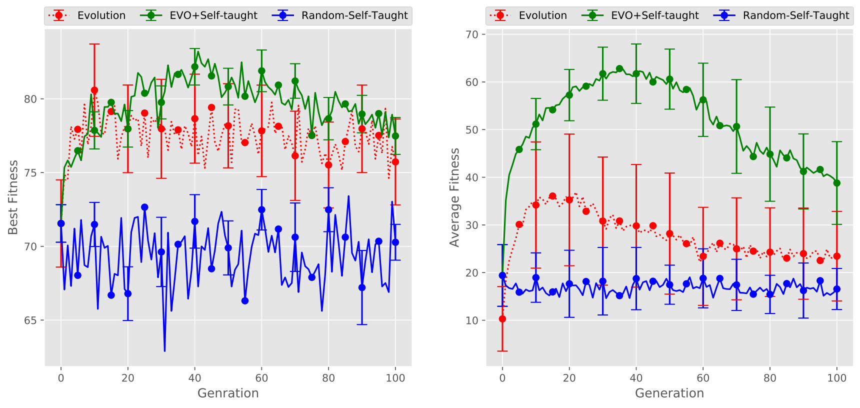

Fig. 5. Comparison of eating ability. a) Left: Best Fitness b) Right: Average Fitness

4 Results and Analysis

4.1 Learning Facilitates Evolution

At first, we compare the eating performance, in terms of the best and the average food eaten, of the population in EVO and EVO+Self-taught simulations. All results are averaged over 30 independent runs.

We look at the dynamics of the eating ability over generations. It can be observed in the left of Figure 5 that there is a difference in the best eating ability between the two populations, EVO and EVO+Self-taught. We can see that the best eating performance of self-taught population is more stable than that of the evolution alone. Even the distinction does not look obviously significant, but it still has some implication.

In biology, even a little difference in the dynamic over generations often shows something more interesting than it looks. In this scenario, we know that every agent has to compete with each others for energy during their lifetime, hence for the selection process at the end of each generation. The better movement an agent makes in an environment, the more the chance it approaches the food. The more an agent absorbs, the less the others can get in one time step during the lifetime. From this we can come up with an idea as follows: In an environmental circumstance in which there are quite a few agents without adaptive behaviour, or not having produced adaptive movements, the best agent tends to have more chance to eat because its competitors are *poorer* enough to go against it. In other words, in such a population the competition between individual agents is strong enough but there is little competitive pressure for the best agent because the remaining is inferior than it. On the other hand, if in a population there is not a small portion of agents producing good actions in the environment, the competition between individuals agents is absolutely stronger, even for the best agent.

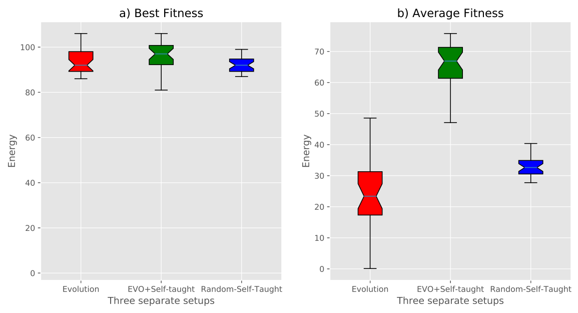

Fig. 6. Boxplot. a) Left: Best Fitness b) Right: Average Fitness.

With this in mind, we are curious whether the population of self-taught agents or that of *innate* agents alone has better performance. We also need to look at the average fitness of the two populations over time. It can be obviously seen in Figure 5b) that there is a clear difference between the average eating ability between the two populations of interest. The average agent in EVO+Self-taught eats 20-30 food particles more than the average agent in EVO population alone. This means, on the whole, the EVO+Self-taught multi-agent system eats approximately 400-600 food particles more than the EVO population alone.

Matching the analysis of both the best and the average eating ability, it can be claimed that the combination of evolution and learning results in a higher performance, promoting the whole population, more than evolution alone. The average fitness says an average self-taught agent eats more food and an average agent in EVO population. This also means the ability of self-teaching equips an average agent produces more adaptive actions in MiniWorld. Thus, the competition for food between self-taught agents is higher than that of agents in EVO alone population. Up to this point, the best agent in EVO+Self-taught can also be thought of having better ability than the best agent in EVO alone population, despite having not that higher performance as shown in Figure 5. The best self-taught agent has to compete more with better average agents, but still produce good enough actions and get more food than the best agent in EVO population with less competitive pressure.

In addition to the dynamic analysis, if we think more about statistical results we can also cut off the generational timescale, assuming that we are seeking for the best and the average food eaten in MiniWorld after 100 generations. This looks like an usual analysis when applying evolutionary algorithms to solve a particular problem. We average the result over 30 runs. The boxplots in Figure 6 show that the best food eaten in Self-taught population is likely to be similar to the best food eaten in EVO population after 100 generations.

If we look into the average fitness (food eaten) after 100 generations, it can be clearly seen that the self-taught population shows a clear higher performance than the EVO population. This all shows that the combination of learning and evolution increases the adaptivity of the population measured by the number of food eaten in any case.

4.2 Evolution Facilitates Learning

We have seen how learning during lifetime facilitates the evolving population of self-taught agents, having higher performance in a multi-agent environment compared to EVO alone. One curious question here is whether the Baldwin-like Effect has occurred?

This is why we conduct the third simulation in which the neural networks of self-taught agents are all randomly initialised, without the participation of evolution. It can be observed in both the left and the right of 5 that the population of randomly self-taught agents has lower performance than that of EVO+Self-taught, especially when it comes to the performance of the whole population (average fitness in our scenario). It is also interesting that in our simulation, the **blank-slate** population by learning even cannot outperform the evolving population without learning.

It is plausible here to conclude that learning, as a faster adaptation, can provide more adaptive advantage than the slower evolutionary process when the environment is dynamic like in MiniWorld. However, it is evolution that provides a good base for self-taught agents to learn better adaptive behaviours in future generations rather than learning as *blank-slates* in Random-Self-taught population.

5 Conclusion and Future Work

We have shown how the learning can enhance the performance of the multi-agent system in MiniWorld. The architecture of self-taught neural networks was proposed to illustrate the idea of self-taught agents – learning by oneself without external supervision in the environment. Based on a specific world and parameter settings, the Baldwin-like Effect has been been demonstrated. The ability to teach oneself has been shown to provide some sort of adaptive advantage over agents without any learning ability. Simultaneously, evolution has also been shown to facilitate future learning, better than learning as blank-slates without priori knowledge about the world. Computer simulations are simple enough to illustrate the idea of research, yet still have indications for future work.

First, with respect to computational modeling technique the Neuroevolution method used in this paper is a bit highly engineering design – this means the architecture of the neural network (both action module and reinforcement module) is handcrafted by the system designer. Evolution and learning only affect the weights of the fixed topology to find good enough combination of weights. One trivial thought can be varying the structure of the neural networks used in our simulation and see how the performance would be varying. Different methods and algorithms in evolving neural networks (both weights and topology) can be taken into account [19].

Parameter setting of MiniWorld environment can show some more interesting effects than trivially for tuning the performance, and this can be worth future investigations. For example, the number of agents in MiniWorld is 20 and the number of food is kept constant as 50. What if we vary the number of food so that it is much larger than the number of agent? In this scenario, there distribution of food in MiniWorld is denser and, on average, each agent has more chance to get more energy, reducing competition pressure between agents during their lifetime. But what if we set the number of agents more than the number of food in MiniWorld? This would lead to a severe competition between individual agents in the population. Whether the better self-taught learners have more competitive advantage over the others in this scenario is an interesting question to be investigated in the future.

Another way to complexify MiniWorld is to include other substances, for example food and poison in the same environment. The agent then has to solve two tasks: the task of producing adaptive movement to approach substance in the environment and simultaneously the task of discriminating between food and poison to consume. It is plausible to think that having the ability to teach oneself will show more adaptive since there is no external supervision in the environment and relying on evolution alone is not adaptive when the environment is increasingly dynamic.

When it comes to theoretical and biological understanding, learning can also be classified into two types: asocial (or individual) learning and social learning. The former is learning by directly interacting with the environment, e.g. trial-and-error. The latter is learning from others, e.g. imitation learning. The ability to teach oneself can be considered a form of asocial learning. Social learning has been shown to complimentary to asocial learning in some work, including [22], [23] [24]. It is often said that the combination of social and asocial learning can result in higher performance than both social and asocial learning alone [22]. Yet in these studies each agent solves its own problem – there is no *real* interactive effect between individual agents. There is no competition for survival between agents during their lifetime. Individual agents compete with each other only through the selection process. The competition in these circumstances is a bit low. One curious question arising here is whether the same observation can be made when asocial and social agents are living in the same environment, sharing the resource of energy and competing for survival during their lifetime. If the competition is too strong, it is plausible to think that learning by oneself seems to be the best strategy when no one is motivated to share *survival information* to others. On the other hand, when the resource is surplus, learning from others can have more advantages. This will be an interesting exploration in the future, which can contribute to the understanding of the evolutionary consequences of learning (asocial and social) in different environments.

Last but not less important, the idea of self-taught neural networks can be powerful when there is no external supervision (or *label* in Machine Learning terminology). The algorithm and technique used in this paper can be a potential technique to solve unsupervised learning or reinforcement learning problems. We are curious whether evolution can provide a better base to learn than learning as blank-slates, to achieve human-level intelligence like what was claimed by DeepMind in games [25].

Acknowledgments.

This research is funded by Science Foundation Ireland under Grant No. 13/IA/1850.

References

1. J. M. Baldwin, “A new factor in evolution,” *The American Naturalist*, vol. 30, no. 354, pp. 441–451, 1896.
2. C. L. Morgan, “On modification and variation,” *Science*, vol. 4, no. 99, pp. 733–740, nov 1896.
3. G. E. Hinton and S. J. Nowlan, “How learning can guide evolution,” *Complex Systems*, vol. 1, pp. 495–502, 1987.
4. S. Nolfi, D. Parisi, and J. L. Elman, “Learning and evolution in neural networks,” *Adaptive Behavior*, vol. 3, no. 1, pp. 5–28, 1994.
5. K. O. Stanley, B. D. Bryant, and R. Miikkulainen, “Evolving adaptive neural networks with and without adaptive synapses,” in *Proceedings of the 2003 Congress on Evolutionary Computation.* Piscataway, NJ: IEEE, 2003.
6. D. Parisi, S. Nolfi, and F. Cecconi, “Learning, behavior, and evolution,” 1991.
7. N. Le, “How the baldwin effect can guide evolution in dynamic environments,” in *7th International Conference on the Theory and Practice of Natural Computing, 2018.* IEEE Press, 12-14 Dec forthcoming.
8. F. Mery and T. J. Kawecki, “The effect of learning on experimental evolution of resource preference in drosophila melanogaster,” *Evolution*, vol. 58, no. 4, p. 757, 2004.
9. J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press, 1975, second edition, 1992.
10. S. A. Kauffman and E. D. Weinberger, “The NK model of rugged fitness landscapes and its application to maturation of the immune response,” *Journal of Theoretical Biology*, vol. 141, no. 2, pp. 211–245, nov 1989.
11. G. Mayley, “Guiding or hiding: Explorations into the effects of learning on the rate of evolution.” in *In Proceedings of the Fourth European Conference on Artificial Life.* MIT Press, 1997, pp. 135–144.
12. L. Bull, “On the baldwin effect,” *Artif. Life*, vol. 5, no. 3, pp. 241–246, Jun. 1999.
13. J. Watson and J. Wiles, “The rise and fall of learning: a neural network model of the genetic assimilation of acquired traits,” in *Proceedings of the 2002 Congress on Evolutionary Computation.* IEEE.
14. P. M. Todd and G. F. Miller, “Exploring adaptive agency ii: Simulating the evolution of associative learning,” in *Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats.* Cambridge, MA, USA: MIT Press, 1990, pp. 306–315.
15. D. Hebb, *The Organization of Behavior: A Neuropsychological Theory*, ser. A Wiley book in clinical psychology. Wiley, 1949.
16. I. Harvey, “Is there another new factor in evolution?” *Evol. Comput.*, vol. 4, no. 3, pp. 313–329, Sep. 1996.

17. D. Ackley and M. Littman, "Interactions between learning and evolution," in *Artificial Life II, SFI Studies in the Sciences of Complexity*, C. G. Langton, C. Taylor, C. D. Farmer, and R. S., Eds. Reading, MA, USA: Addison-Wesley, 1992, vol. X, pp. 487–509.
18. K. L. Downing, "The baldwin effect in developing neural networks," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '10. New York, NY, USA: ACM, 2010, pp. 555–562.
19. A. Soltoggio, K. O. Stanley, and S. Risi, "Born to learn: The inspiration, progress, and future of evolved plastic artificial neural networks," *Neural Networks*, vol. 108, pp. 48–67, dec 2018.
20. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Learning Representations by Back-propagating Errors, pp. 696–699.
21. X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.
22. N. Le, M. O'Neill, and A. Brabazon, "The baldwin effect reconsidered through the prism of social learning," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, July 2018, pp. 1–8.
23. L. Rendell, L. Fogarty, W. J. Hoppitt, T. J. Morgan, M. M. Webster, and K. N. Laland, "Cognitive culture: theoretical and empirical insights into social learning strategies," *Trends in Cognitive Sciences*, vol. 15, no. 2, pp. 68–76, feb 2011.
24. N. Le, M. O'Neill, and A. Brabazon, "Adaptive advantage of learning strategies: A study through dynamic landscape," in *Parallel Problem Solving from Nature – PPSN XV*, A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, Eds. Cham: Springer International Publishing, 2018, pp. 387–398.
25. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, feb 2015.