Social Learning vs Self-teaching in a Multi-agent Neural Network System

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

Natural Computing Research & Applications Group
University College Dublin
Email: namlehai90@ucdconnect.ie

Abstract. Learning has been shown to be beneficial to in creating more adaptive algorithms, and also in evolving neural networks. Moreover, learning can be classified into two types, namely social learning, or learning from others (e.g., imitation), and individual trial-and-error learning. A *"social learning strategy"* – a rule governing whether and when to use social or individual learning, is often said to be more beneficial than relying on social or individual learning alone. In this paper we compare the effect on evolution of social learning in comparison with that of individual learning. A neural architecture called a "self-taught neural network" is proposed in order to allow an agent to learn on its own, without any supervision. We simulate a multi-agent system in which agents, each controlled by a neural network, have to develop adaptive behaviour and compete with each other for survival. Experimental results show that evolved self-teaching presents the most effective behaviour in our simulated world. We conclude this paper with some indications for future work.

Keywords: Meta-learning · Multi-agent · Baldwin Effect · Neural Networks · Hybrid Algorithms

1 Introduction

The idea that lifetime learning can influence evolution in a Darwinian framework was introduced over one hundred years ago in the famous paper *'A new factor in evolution'* [1]. The described phenomeon was later termed the **Baldwin Effect** [2]. Following the classic paper of ([3]) which demonstrated an instance of the Baldwin effect in a computer simulation a significant related literature has emerged including, [4], [5], and recently [6]. The so-called *effect* in computation can simply be understood as a *hybrid* algorithm combining an evolutionary algorithm (EA) with some form of local-search at the phenotypic level. 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 with [7] and [8] being two exemplar studies. Most recently, authors in [9], [10] proposed a self-teaching neural architecture which can learn without requiring any supervisory or external reinforcement signals.

Learning can generally be classified into two types, namely *social learning* (SL) and *individual (asocial) learning* (IL). IL can be broadly understood as learning when the learner directly interacts with its environment, e.g., via trial-and-error, without the presence of others or their information. SL, on the other hand, can be interpreted as learning from others, e.g., imitation, through observation or interaction. Several studies have shown that social learning in combination with asocial learning by some *strategy* can outperform individual or social learning alone [11], [12], [13]. This line of thought has also been employed in hybridising EA in which a phenotypic local search combining both social and individual learning presents an adaptive effect to keep an evolving population on track in dynamic environments [14], [13]. However, this still leaves an open challenge as these findings can be problem-specific, and the results may not generalise beyond the specific modelling of each learning mechanism.

This paper addresses the question of how effectively social and individual learning influence evolution in evolving self-taught neural networks. We simulate a situated multi-agent system in which each agent, controlled by a neural network, seeks to find and absorb resources from its environment, thereby competing for survival with other agents. In this simulation, agents have limited 'visibility' of their environment. As the environment is dynamically generated, the solution cannot be defined in advance. These factors make it harder for an agent to develop an *intelligent* behavioural policy.

In the remainder of the paper we present a brief overview of research on learning and evolution in neural networks, social learning and related concepts. In turn we describe simulations used to investigate our research question and then analyse and discuss the results obtained. Through the rest of paper, we use the terms *observer* / *learner* / *student*, and *teacher* / *demonstrator* / *teacher*, interchangeably, in order to keep our terminology in line with that used in the prior literature that we discuss.

2 Related Work

Evolution and learning are complementary forms of adaptation by which an organism can modify its behaviour in response to environmental challenges. Indeed, evolution and lifetime learning are closely intertwined, as a capability for lifetime learning can only arise as a result of an evolutionary process. Perhaps less evidently, the linkage also goes the other way and learning can influence the evolutionary process, enhancing the adaptivity of a species over time. This phenomenon is termed the **Baldwin Effect** and was demonstrated in a classic simulation paper [3]. The so-called Baldwin Effect in evolutionary computation (EC) means a combination of an evolutionary search over genotypic space and a local search process over phenotypic space, provided that what is found from phenotypic search is not directly encoded back into the genotype. This paper stimulated a number of important follow-on studies in which learning has been shown to help and guide evolution in different domains, including cases of search on NK-landscapes ([15]), and search in neuroevolution-controlling robots [7].

Learning and evolution in neural network learning has been studied in several papers following the original work of Hinton and Nowlan [3]. Notable studies include [16] in which the authors used a genetic algorithm to evolve the initial weights of a digit classifier neural network which then can be further adapted by backpropagation (life time learning). This study found that learning can take advantage of starting weights produced by evolution to further the classification performance.

Nolfi and his colleagues made a simulation of *animats*, or robots, controlled by neural networks situated in a grid-world, with discrete state and action spaces [7]. Each agent lives in its own copy of the world, hence there is no mutual interaction. 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, to update the weights of a neural network. It was shown that learning to predict the sensory inputs can enhance the evolutionary search, hence increasing the performance of the robot.

Researchers at Google DeepMind also employed the Baldwin Effect, using a genetic algorithm to evolve the initial weights for deep neural networks [17]. By combining the advantage of searching over a vast distribution of weights using evolutionary search, and the exploitative power of gradient descent learning, they reported a *meta-learner* that can solve a multiple tasks including regression and physical robot environments. This result is an indication to create meta-learning, another step towards Artificial General Intelligence (AGI).

More recently, authors in [9] proposed a technique called evolving self-taught neural networks (ESNN) which are used to control agents living within a food-patch for survival. Unlike those mentioned above, an ESNN is capable of teaching itself without external supervision or reinforcement. Evolution plus self-teaching was shown to provide a way to generate better self-reinforced signals over time, and to generate more adaptive behaviour than evolution or self-teaching in isolation. An ENSS can be understood as a form of meta-learning towards AGI.

Generally learning in neural networks can be thought of as part of neural plasticity. There have been some other ideas, like evolving local learning rules to update the weights [18], evolution of neuromodulation which facilitates the information transfer between neurons in hopes of creating meta-learning. Please refer to [8] for more recent studies on evolving plastic neural networks. In short, neural networks in most of these work still require some sort of supervisory or external reinforcement signals to guide the learning process.

Delving a little deeper into the term lifetime learning, it can be subdivided into *asocial* (or individual) learning (IL), and *social* learning (SL). Each is a plausible way for an individual agent to acquire information from the environment at the phenotypic level. By SL, we mean *learning that is influenced by the observation of, or the interaction with, another animal or its products* [19].

Several models have been proposed to investigate how to use SL effectively, both in biology [20] and in EC [12]. A key finding of these studies is that social

learning should be combined with individual learning in a strategic way, and such a strategy can potentially outperform both social and individual learning alone. Social learning strategies consist of rules specifying the way an individual relies on social learning by answering three questions: when SL should be used, from whom an agent should learn, and what an agent should learn or copy[1] from others. As an example of a social learning strategy, *critical social learning* proposed by [21] and also adopted in recent EC studies ([14], [13]), suggests a strategy as follows: first learn socially, then learn individually. Prima facie, this bears loose analogy with the way highly cognitive animals like humans learn in nature, since we copy significantly from others then innovate ourselves to make progress. Results obtained from these studies present an effect of a learning strategy on evolution better than that of social or individual learning alone, promoting evolutionary optimisation in dynamic environments [14].

Interestingly, an important strand of research in Artificial Life adopts a related approach in both simulation studies and robotics in which neural networks are mostly used to control the behaviour of a robot (for a brief recent survey please refer to [22]). For example, [23] showed that social learning when combined with individual learning can provide a better way for neural network-controlled robot to learn and adapt to its environment compared to robots which possessed either social or individual learning alone.

In this paper, we extend previous research on evolving self-taught neural networks [9] by investigating the effect of social learning, or social learning strategy, might have on evolving autonomous intelligence in a multi-agent system. Unlike previous work, we do not allow agents live within the food-patch, but force them do two tasks: first to find resources, and second to compete for food. The self-taught agent can be considered to be autonomous. Social learning is implemented in the simulations as the student (agent) seeks to copy the process that has produced "good" behaviour in its teacher. In the next section, we describe our simulation model in more detail.

3 Simulation setup

3.1 The Simulated World of Food and Agents

Suppose that 20 agents are situated in a continuous 640x640 2D-world, called **MiniWorld**. Agents seek to find resources to feed themselves in order to survive. In MiniWorld, 50 food particles are randomly dispersed and each particle is represented by a square image with size 10x10. Each agent in MiniWorld also has a similar size. We use two world maps (map A, and map B) in our simulations as described in Figure 2. In the simualtions we implement a *toroidal* environment – so that when an agent moves beyond an "edge", it appears on the opposite "side" of the environment.

Initially agents have to forage to find the food sources. A visualisation of an agent and its relationship with food particles in MiniWorld is shown in Figure

[1] The term *copy* is often used to stand for any form of social learning, not just copying.

2. Through the visualisation, based on our assumption on initial heading, it can be seen that map B is likely to be more difficult than map A from an agent perspective.

All agents in the population live in the same MiniWorld and their behaviours interact. As an agent finds and consumes food particles, it changes the environment faced by the other agents. This creates *mutual-competition-within-patch* between the agents.

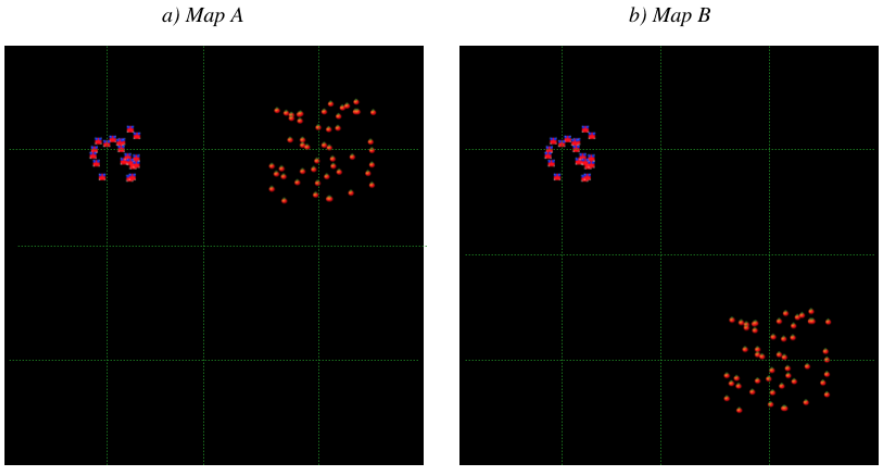

Fig. 1. MiniWorld – The environment of agents and food, 640x640. a) and b) Denote w and h as the width and the height of MiniWorld. In both map A and B, initially all agents are located in a radius of 40 (4 times the size of an agent) around a central point: ($w/4$, $h/4$). Food particles in map A have horizontal and vertical dimensions randomly chosen in the range ($5w/8$, $7w/8$) and ($h/8$, $3h/8$), respectively. The food region in map A is the square that has the same central point as the top right quarter, and each side of that square has the length of $w/4$. In map B, the food has its horizontal and vertical dimensions randomly chosen in the range ($5w/8$, $7w/8$) and ($5h/8$, $7h/8$), accordingly. The food region in map B is the square that has the same center as the bottom right quarter, and each side of that square has the length of $w/4$. When an agent's body happens to collide with a food particle, the food particle is "eaten", the energy level of the agent increases by 1, and another food piece is randomly spawned in the **same region** but at a different location. The collision detection criterion is specified by the distance between the two bodies (of the agent and of the food particle). The environment changes as an agent eats a particle.

The default velocity (or speed) of each agent is 1. Every agent has three basic movements: Turn left by 9 degrees and move; move forward by double speed; or turn right by 9 degrees and move. For simplicity, these rules are pre-defined by the system designer of MiniWorld. We can imagine a perfect scenario such as if an agent sees a food in front of its current location, it doubles its speed and moves forward to the food particle. If an agent sees the food on the left (right), it will 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.

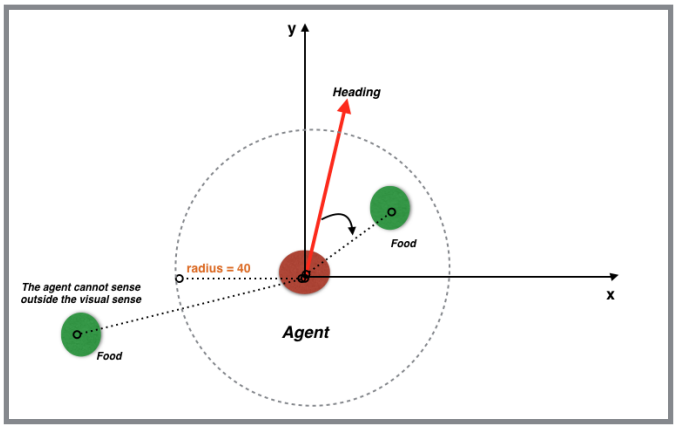

Fig. 2. Each agent has a heading (in principle) of movement in the environment. Rather than initialising all agents with random headings, all the agents are initialised with a **horizontal heading** (i.e., with a heading of 0 degrees). This somewhat explains the purpose of the design of map A and map B. In map A, all agents are initially created with a tendency to move forwards the food source. On the contrary, agents in map B are born with facing away from the food source resulting in a more difficult environment. Assume that every agent has an a priori ability to sense the angle between its current heading and the food if this appears in its visual range. The visual range of each agent is a circle with radius 40. Each agent takes as inputs, three pieces of sensory information. The three bits (left, front, right) are set to 0 or 1 depending on whether the substance appears (in the left, front, and right) or not. Let θ (in degrees) be the angle between the agent and the substance in its visual sense. An agent determines whether a food appears in the left, front, or right side in its visual range using the following rule: Right if $15 < \theta < 45$; Front if $\theta < 15$ or $\theta > 345$; and Left if $315 < \theta < 345$.

3.2 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 next sensory information it receives, and hence its future behaviour. This forms a sensory-motor dynamic, and the neural network acts as a situated cognitive module, guiding an agent to behave adaptively.

The architectural design of the neural network controller is visualised in Figure 3. 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. Note that 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.

3.3 The Self-taught neural architecture

To allow for self-teaching, two modules are implemented in the neural controller for each agent. One is called the **Action Module**, and the other is called the **Reinforcement Module**. The action module has the same network as previously shown in Figure 3. This module takes as inputs the relevant sensory

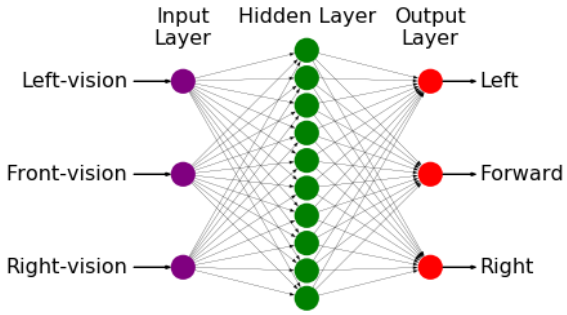

Fig. 3. Basic network without learning. Each neural network includes 3 layers with 3 input nodes, 10 hidden nodes, and 3 output nodes. The first layer takes as input what an agent senses from the environment in its visual range. The output layer produces 3 values in which the max value is chosen as a motor-guidance.

information and produces reinforcement outputs in order to guide the motor action of the agent. The reinforcement module has the same set of inputs as the action module, but possesses separate sets of hidden and output neurons.[2] 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.

In the following sections we describe the simulations we use to investigate the evolutionary consequence of lifetime learning, including self-taught learning and social learning.

3.4 Simulation 1: Evolution alone (EVO)

In this simulation, we evolve a population of agents which do not have a lifetime learning capability. The neural network controller for each agent is as described in Figure 3a, without any learning capability. 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.

Selection chooses individuals based on the number of food particles consumed. The higher the number of particles eaten, the higher the agent's fitness value. For crossover, two individuals are selected to produce one offspring. We implement crossover as follows. The more successful a parent, the greater the likelihood that its weights are copied to the child. Each weight element in the

[2] The reinforcement and the action modules need not have the same topology. In our simulation, 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.

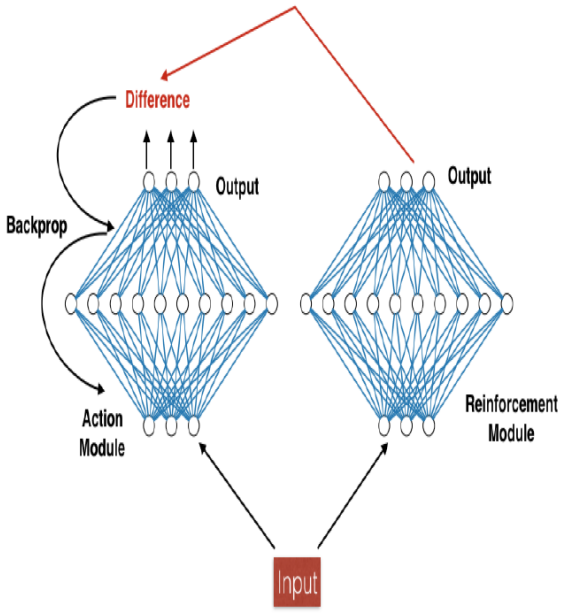

Fig. 4. Self-taught neural architecture. The difference between the output of the reinforcement module and the action module is used to update the weights in action modules through backpropagation. Through this self-learning process, the action module approximates its output activation towards the output of the reinforcement module. The learning rate is 0.01.

matrix of the child network is copied from the fitter parent if the random probability is greater than 0.5, and vice versa.

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

The population goes through a total of 100 generations, with 5000 time steps per generation. At each time step, an agent undertakes the following activities: Perceiving MiniWorld through its sensors, computing its motor outputs from its sensory outputs, moving in the environment which then updates its new heading and location. In the 'evolution alone' simulation, the agent cannot perform any kind of learning during its lifetime. After that, the population undergoes selection and reproduction processes.

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

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 as 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 the evolution alone simulation. We use the same parameter settings for evolution as in EVO simulation above.

At each time step, an agent does the following activities: Perceiving Mini-World through its sensors, computing its motor outputs from its sensory outputs, moving in the environment which then updates its new heading and location, and updating the weights in action module by **self-teaching**. After one step, the agent updates its fitness using the number of food particles consumed. After that, the population undergoes selection and reproduction processes as in the evolution alone simulation.

In these experiments, we implement learning and evolution in a Darwinian, not a Lamarckian framework. This means that the lifetime learning of an agent (the weights in its action module) is not passed down to its offspring.

3.6 Simulation 3: Evolution + Social Learning Alone (EVO+SL)

In simulation 3, we use social learning, instead of individual learning (self-taught learning), in combination with evolution. In order to implement social learning, we first propose the supervised learning-based imitation procedure by which an individual student learns from its teacher. The process of social learning by imitation between two agents is depicted in Figure 5. Please note that, imitation learning here happens between two action networks. The reinforcement module which is related to self-learning is untouched.

In this simulation, only social learning is implemented, there is no individual learning. The type of social transmission adopted is called *Oblique transmission*, which was also used in previous successful social learning applications in EC [14], [13]. The *Who strategy* specifies the most successful agent in terms of fitness from the previous generation as the teacher for all individuals in the current population. It is assumed that at each step, each agent keeps its input-output pair, in which the input includes three sensory values while the output comprises of three movement values as shown in the neural controller architecture (Figure 3).

For each social learning agent, the social learning strategy (i.e., the answers to the three questions of *When*, *Who*, and *What*) is defined as follows:

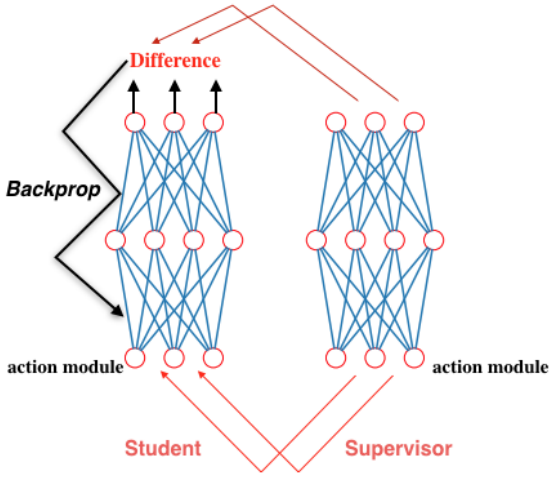

Fig. 5. Social learning process in the form of imitation learning between a student and its teacher. The teacher provide its input-output pairs during its lifetime, as a training set of experiences, to its student. Thus, the student is exposed to the same inputs as the teacher. The output difference between them is used to guide the student to approximate the teacher's output. The weight-updating in student's action network is carried out using a backpropagation learning algorithm, with a learning-rate of 0.01.

i. When: a student learns socially at birth before experiencing its own environment. We can interpret this scenario so that a newborn agent is naive, and learns from the most successful individual at birth.
ii. Who: learn from the most successful individual in the previous generation.
iii. What: learn the teacher's sensory-motor experience.

After undertaking social learning, every individual agent returns to and experiences its own environment using the same procedure as described in the previous simulations. Since we employ *Oblique transmission*, social learning does not appear at the initial generation, but starts from the second generation.

3.7 Simulation 4: Evolution in combination with social and individual learning (EVO+SL+IL)

In this simulation, we incorporate both social and asocial learning (self-teaching) in combination with evolution. Each type of learning is as already described above. In the initial generation, there is no social learning, only self-learning is performed. From the second generation, every agent performs social learning at birth, as described above, before experiencing its lifetime in its environment.

4 Results and Analysis

We compare agent performance across the various experimental setups, in terms of the best and the average amounts of food eaten, of the population in the four simulation settings as above. All results are averaged over 30 independent runs.

A similar trend can be observed in Figure 6 and Figure 7 in that all experiments produce higher performance in map A than in map B. This is as expected for the reasons already set out: Map B is designed to be more difficult than map A as the agents' initial trajectory is away from the food sources.

4.1 EVO Alone vs EVO+Self-taught

We initially compare the performance between the baseline experiment of evolution alone with the performance of evolution of self-taught agents. In all maps, EVO+Self-taught outperforms EVO alone in terms of both best and the average fitness. This could be explained by the effect of individual learning on evolution, or the *Baldwin Effect*. One more notable point here is that EVO alone cannot absorb any energy at all in map B, while evolved self-taught agents can.

Simply speaking, an agent that cannot learn can only use its *innate* ability, hardwired in its brain, to search the environment. However, as analysed above, in map B the agent in EVO alone is born without any tendency to sense relevant information (about food) in the environment, and also has no ability to change its motor program hardwired in its brain. Its sensory-motor experience cannot be changed since it cannot sense relevant information (i.e. information to find the food source).

Conversely, with an ability to teach oneself by leveraging the difference between the action and the reinforcement modules, the weights of the action module of some self-taught agent can alter to produce a wider range of movement. By undertaking some initial random movement, the sensory-motor experience of an agent can be expanded, and there may have been some agents that reached the food sources. The agents that reach food sources have a higher chance of being selected to leave offspring. Thus, its *good* genetic information, consisting of the *initial* weights of both the action and the reinforcement modules, is more likely to proliferate, hence its self-supervising or self-teaching ability. It is this ability that has made future evolved self-supervised agents better at teaching themselves in order to develop more effective movement in MiniWorld. This process repeats, as what has given advantage during lifetime of the self-taught agent is preserved and promoted by the evolutionary process. This can be considered the interaction between learning and evolution.

4.2 Social Learning vs Self-teaching

We can see that in all maps, EVO+Self-taught also outperforms both EVO+SL and EVO+SL+IL in terms of both average and best fitness. The difference is bigger in the harder map B than in map A.

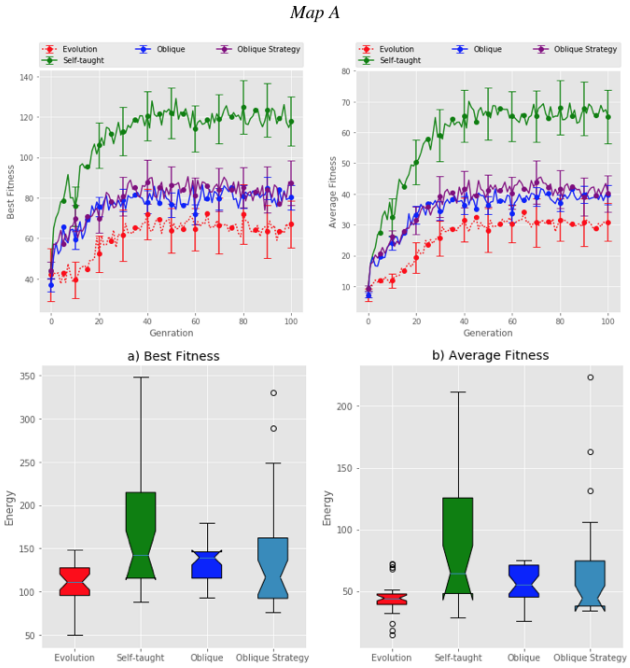

Fig. 6. Fitness Comparison in Map A.

One interesting behaviour that can be observed here that EVO+SL alone failed to eat anything in map B. Social learning is a form of *information-parasitism*, and cannot produce new information about the environment. Importantly, because there is no individual learning permitted, what social learners can learn is just what the evolutionary process has provided to them. More specifically, as of the second generation, social learners learn from the best agent of the preceding generation and with no agent performing well, there is nothing useful to learn.

Even when coupled with self-learning, social learning produces poorer results than self-learning alone in all maps. What is presented in Figure 7 shows that EVO+SL+IL can produce *good enough* behaviour (agents can find food), but not as good as EVO+Self-taught – without the presence of social learning. This means the employment of social learning here is not promoting, but rather reducing the power of self-learning.

There are several factors contributing to what we can call the *discouraging-effect* of social learning. First, unlike previous studies as mentioned in Section 2, here there is no *pre-defined optimal solution* in MiniWorld. Thus, social learning here cannot simply copy a *known* optimal solution.

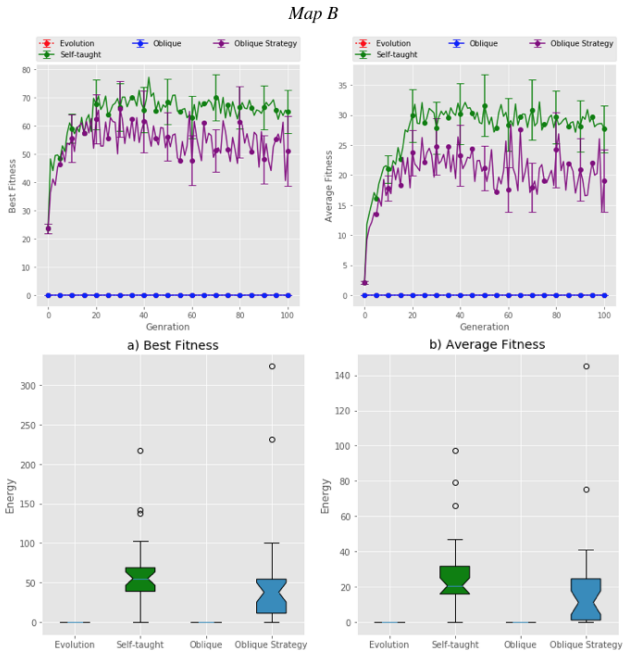

Fig. 7. Fitness Comparison in Map B.

Secondly, through the supervisory-based social learning process, the student has to develop its own policy based on the previous sensory-motor experience of its teacher as *off-line* training-samples. The experience of the teacher can make more sense if the student would be likely to face the same experience in *its own world* like what it has learned from the teacher. However, the nature of the multi-agent in MiniWorld is that the sensory-motor experience of an agent depends on not only its own actions, but also on those of others which might change the world the agent is experiencing. What the student is going to experience depends on what other agents in the same world are doing which can be different from what the teacher and the other agents were doing in the teacher's world. This creates a variety of dynamics dependent on **space and time**. This is what can make the world the student agent is experiencing dynamically different from the world the supervisor has experienced. Social learning, or cultural learning, from any previous generation is more likely to produce outdated information, thus *discouraging effect* as we have seen. Approximating an action network based on outdated information, even from the *best teacher*, is not promising.

One more factor which can be added is the fact that an agent in MiniWorld has little knowledge about its environment.

5 Conclusion

We have investigated the effect of different forms of learning (social and individual learning) on evolving self-taught neural networks in a situated multi-agent system in which knowledge of the environmental state is dynamic and cannot be completely accessed by the agent. Experimental results have shown that the combination of self-teaching and evolution is most effective in evolving intelligent agents as the agent develops its own policy based on its evolved self-teaching capability. When self-teaching is powerful enough, social learning even when being used selectively reduces the power of self-teaching.

This work continues to consolidate the power of hybrid algorithms by combining the metaphor of evolution and learning. Learning has been again shown to facilitate evolution in developing intelligent behaviour, even when the *good* behaviour is unknown and dynamic over time and space. Previous studies have shown the power of the combination of both social and individual learning in evolutionary dynamic optimisation [13], [14]. While this finding does not concord with that of some previous studies, we note that the experiments undertaken in this work implement a more complex environment and hence, are likely to be more generalisable to "harder" real world environments.

Indeed, what can be extracted here is that if a learner is good at *self-teaching*, self-learning can sometimes result in a better outcome than learning from external supervisory signals.

Another philosophical point here is that a constructive approach towards social learning via the synthesis of artificial agents can yield important insights into mechanisms that can inform biologists, psychologists and Artificial Intelligence researchers by fleshing out theory. From the computational side, another contribution is the use of an evolved self-taught neural network. This provides a framework for building intelligent autonomous adaptive systems in the environment without engineered rewards and where the state of this environment is only partially observable [24]. The evolved ability to teach oneself produces a form of autonomous intelligence, without any kind of *external supervision*. Building adaptive autonomous multi-agent system is potentially a promising way to reach general intelligence. In future work, both weights and topology could be evolved rather than assuming a fixed architecture. MiniWorld can also be extended into more complex environmental settings by incorporating for example, food & poison, and via the inclusion of obstacles which would add further complexity to the learning task, potentially forcing a stronger learning / evolutionary response from agents.

References

1. J. M. Baldwin, "A new factor in evolution," *The American Naturalist*, vol. 30, no. 354, pp. 441–451, 1896.
2. N. Le, "Organic selection and social heredity: The original baldwin effect revisited," in *The 2019 Conference on Artificial Life*. MIT Press, 2019.

3. G. E. Hinton and S. J. Nowlan, "How learning can guide evolution," *Complex Systems*, vol. 1, pp. 495–502, 1987.
4. 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.
5. I. Harvey, "Is there another new factor in evolution?" *Evol. Comput.*, vol. 4, no. 3, pp. 313–329, Sep. 1996.
6. N. Le, "Evolving self-taught neural networks: The baldwin effect and the emergence of intelligence," in *2019 AISB Annual Convention – 10th Symposium on AI & Games*, Falmouth, UK, 16-18 Apri 2019.
7. S. Nolfi, D. Parisi, and J. L. Elman, "Learning and evolution in neural networks," *Adaptive Behavior*, vol. 3, no. 1, pp. 5–28, 1994.
8. 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.
9. N. Le, A. Brabazon, and M. O'Neill, "The evolution of self-taught neural networks in a multi-agent environment," in *Applications of Evolutionary Computation*, P. Kaufmann and P. A. Castillo, Eds. Cham: Springer International Publishing, 2019, pp. 457–472.
10. N. Le, "Evolution and self-teaching in neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO 19*. ACM Press, 2019.
11. K. N. Laland, "Social learning strategies." *Learning and Behavior*, vol. 32, pp. 4–14, 2004.
12. 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.
13. N. Le, M. O'Neill, and A. Brabazon, "How learning strategies can promote an evolving population in dynamic environments," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, June 2019, pp. 2284–2291.
14. N. Le, M. O'Neill, and A. Brabazon, "Evolutionary consequences of learning strategies in a dynamic rugged landscape," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19. New York, NY, USA: ACM, 2019, pp. 812–819. [Online]. Available: http://doi.acm.org/10.1145/3321707.3321741
15. R. Suzuki and T. Arita, "Repeated occurrences of the baldwin effect can guide evolution on rugged fitness landscapes," in *2007 IEEE Symposium on Artificial Life*. IEEE, apr 2007.
16. R. Keesing and D. G. Stork, "Evolution and learning in neural networks: The number and distribution of learning trials affect the rate of evolution," in *NIPS 1990*, 1990.
17. C. T. Fernando, J. Sygnowski, S. Osindero, J. Wang, T. Schaul, D. Teplyashin, P. Sprechmann, A. Pritzel, and A. A. Rusu, "Meta-learning by the baldwin effect," *CoRR*, vol. abs/1806.07917, 2018. [Online]. Available: http://arxiv.org/abs/1806.07917
18. S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei, "On the optimization of a synaptic learning rule," in *Optimality in Biological and Artificial Networks*, D. S. Levine and W. R. Elsberry, Eds. Lawrence Erlbaum, 1995.
19. C. M. Heyes, "Social learning in animals: Categories and mechanisms," *Biological Reviews*, vol. 69, no. 2, pp. 207–231, may 1994.

20. M. W. Feldman, K. Aoki, and J. Kumm, "Individual versus social learning: Evolutionary analysis in a fluctuating environment," Santa Fe Institute, Working Papers, 1996.
21. M. Enquist, K. Eriksson, and S. Ghirlanda, "Critical social learning: A solution to rogers' paradox of nonadaptive culture," *American Anthropologist*, vol. 109, no. 4, pp. 727–734, dec 2007. [Online]. Available: https://doi.org/10.1525/aa.2007.109.4.727
22. C. Marriott, J. M. Borg, P. Andras, and P. E. Smaldino, "Social learning and cultural evolution in artificial life," *Artificial Life*, vol. 24, no. 1, pp. 5–9, feb 2018.
23. A. Acerbi and S. Nolfi, "Social learning and cultural evolution in embodied and situated agents," in *2007 IEEE Symposium on Artificial Life*. IEEE, apr 2007.
24. N. Le, "Evolving self-supervised neural networks: Autonomous intelligence from evolved self-teaching," *CoRR*, vol. abs/1906.08865, 2019. [Online]. Available: http://arxiv.org/abs/1906.08865