TurtleAI: Copying images with Neuro-Evolution

KHOA NGO THE

Compiled June 11, 2021

During our Evolutionary Computing course, we were able to build an L-system that makes the turtle fill up a 20x20 matrix by evolving the turtle's rule or expanding the structure for a sufficient number of times. The experiment was deemed a success as we were able to completely fill that matrix and obtained a fitness of 400. However, when we were trying to evolve a turtle to copy the 20x20 Mona-Lisa binary image, the experience failed badly. This paper seeks to address the question of whether a turtle can learn to imitate an image by looking at the difference of the image and its current landscape. While the training result showed several pitfalls, the testing phase proved the Turtle's effectiveness in copying any 20x20 image and the experience was a success.

1. INTRODUCTION AND MOTIVATION

Genetic-Algorithms have proven its incredible power in solving complex problems with simple fitness function. In class, we were able to experiments its power when we mutate and evolve a matrix to match perfectly the pictures of Mona-Lisa. We were also able to evolve an L-system to fully cover a 20x20, allowing our turtle to search through or corner of its world. However, the same experiment fails to replicate a simple image such as the 20x20 Mona-Lisa (figure 1). While we have seen examples of interesting virtual creatures and objects generated from an evolved L-system Turtle [1],[2], a task that requires a rigid-solution and less freedom in its fitness function like guiding the Turtle to draw a specific thing might be extremely hard for an L-system. We would like to test different Genetic Evolution Strategy in guiding a turtle in replicating any specific image, in this case, our choice of test image is the 20x20 Mona-Lisa shown in figure 1.

To do this, we take motivation in some research on reinforcement learning, treating the turtle as an agent to maximize the fitness function. Hence, we ultilized the complex structure generated by NEAT to evolve a complex network for the turtle.

In the following sections, we first provide a lists of related works and possible solutions to this problem, examine 5 different experimental results on the Mona Lisa Problem and perform ablation study to make inference to future study. Finally, we will end the paper with an extension of the result, possible future study and conclusion.

Fig. 1. The 20-bit Mona Lisa pictures

2. METHODS

When we first encountered the problem, we believed that this problem could be treated as a sequence-to-sequence problem [**?**], in which a 20x20 image can be flatten in to a 1x400 vector, and then fed in to a recurrent neural network, which is then optimized using Genetic Algorithm (the fitness is not differentiable), decoded in to sequences of action and fed into the turtle. This approach has also been employed to predict vehicle trajectory in self-driving car [3]. However, we later reject this approach, as the length of the correct sequence would be extremely long (500-1000 characters), and there are many sequences that can solve an image, making it impractical to test the models in such a short amount of time and a limited amount of resources.

Hence, we proceed to experiment a simpler solution: Treating the turtle as a reinforcement learning agent, and design a fitness function so that the agents can learn to perform the right action within the right circumstances. We rely on the complexity created by NEAT [4] to create a sufficient complex structure for the Turtle movement, also, we take inspiration from CPPN [5] and allow multiple activation function for more complex learning structure. In total, we perform 5 different experiment, the details of which will be elaborated in the next section.

3. EXPERIMENT

Before the experiments, we modify the code of LSTurtleWorld to allows the Turtle to simply move up, down, left, right, instead of having to turn 90^o before it moves. Next, we add the ability for the turtle to draw just a dot on its landscape to avoid any drawing inconsistency. We will also add a function to calculate the distance from the turtle to all of the point in the difference matrix (the absolute value of the substraction of the turtle landscape and an input image). Using the difference matrix, we can

either get the coordinate of a random point in the landscape that is different from the image, or the nearest point to the turtle in the landscape that is different from the benchmark image. The finding of those two points will be vital in our approach of experience 3 and 4.

A. Experiment 1 + 2: Feed Forward and Recurrence network with Location and Time input

First, we simply take the fitness function as the difference of the Turte's World and the benchmark image Mona Lisa. By default, there are 53 pixels different between the two images. Any genomes that result in a decrease in difference of the two images will be rewarded with 100 points in fitness. In contrast, all genomes that cause the difference to increase will result in a decrease in fitness and termination. In this experiment, we have 5 input for NEAT:

1. The number steps the turtle have taken
2. The x cooridnate of the turte
3. The y coordinate of the turtle
4. The distance to the right edge of the world
5. The distance to the bottom edge of the world

Neat will output 5 outputs ranging from 0 to 1, which we will take the one with the highest probability as the input for the turtle:

- If output 1 has the highest probability, move up
- If output 2 has the highest probability, move down
- If output 3 has the highest probability, move left
- If output 4 has the highest probability, move right
- If output 5 has the highest probability, draw

We believe that the number of steps the turtle have taken is needed so that the turtle can distinguish between a point it has drawn and a point it has not, standing at the same location. Hence, since there will be elements of recurrence in the inputs, so for experiment 2, we will use a time-recurrent-neural network in our NEAT configuration [6]. The detail of the parameters for NEAT will be provided in the code. Figure 2 is the result of the best fitness function after 5000 generation of NEAT in experiment 1.

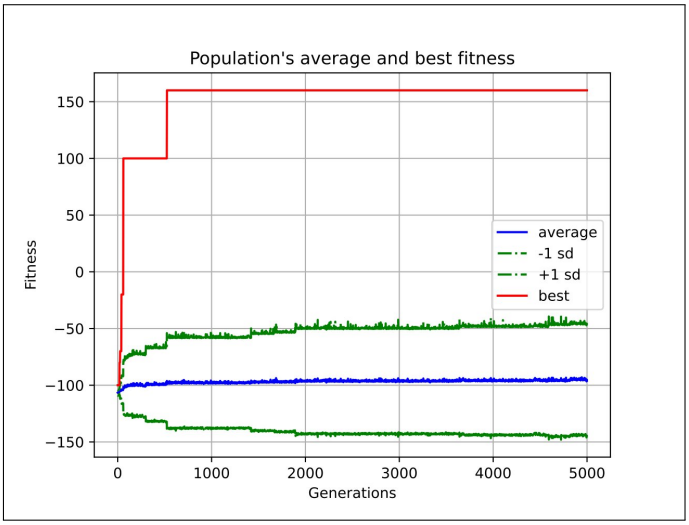

Fig. 2. Best/Average Fitness per generation of Experiment 1 after 5000 generations

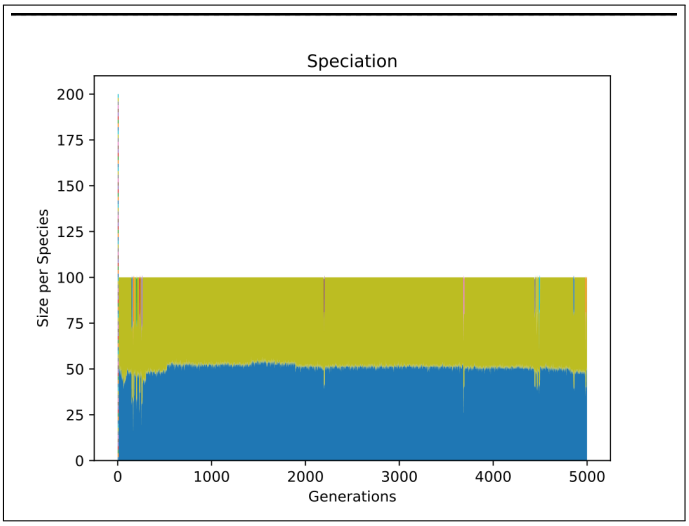

Fig. 3. Size and percentage of Species in each generation

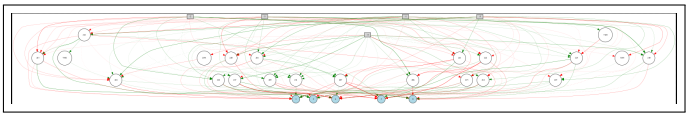

Fig. 4. The final graph of the NEAT neural network

As we can see, the variety of species in the population is very low. And the best fitness stops increasing after 1000 generations. From the initial 53 pixel difference, the network was able to bring down to 50 pixel different only. Since search space is too wide and the possibility of gaining information is too low (correcting putting a dot at the correct position) the network fails to have enough information to evolve into more complex or appropriate structure. The result of experiment 2, is the same. The network hits a best fitness of 100 and never increases ever since. We do not include a graph of experiment 2 since it is identical to experiment 1. The detail and significance of figure 4 will be discussed in later section.

B. Experiment 3: Recurrence network with per step fitness function

For this experiment, we take inspiration from the game of snake, which has already received extensive research [7, 8]. However, while the previous approach relies on either rule-based searching methods [7] or genetic evolution of predefined network [8], we will employ a NEAT approach, allowing the network to evolve itself instead of predefining it and optimize using GA.

Now, as we have seen how experiment 1 and 2 fails to solve the problem of a single Mona-Lisa image, we will look at a more powerful solution to this problem. Instead of updating fitness of a genome every time there's a decrease in difference between the two images. We will try to measure how well the turtle is doing every steps. To do that, first, we will find a random difference coordinate such that the two points at the corresponding coordinate of the landscape and the benchmark image are different. Then, we will define the fitness function as follow:

1. +1.5 everytime the turtle comes closer to that random point (compared to the previous steps)
2. -1.5 everytime the turtle move away from that point
3. +100 everytime the turtle draw a dot at that point. Then generate a new random difference point.
4. -5 every time the turtle draw a dot
5. -10 every time the turtle hit the edge.

Now, it is fairly intuitive to understand the first 3 condition of the fitness function. We wanted to encourage the turtle to move tackle each point at a time by moving closer to that point, draw a dot, and move on to the next one that is different. We add the 2 later condition in order to discourage certain behavior from the turtle. After several experiment, we realize that the turtle has a tendency to stay where it is after it has succeededly drawn a correct dot. Since we do not have punishment for staying at the same place, the turtle will continue drawing dots at that same location until it dies. Hence, we added in the forth condition in order to discourage such behavior. Likewise, the fifth condition come about whenever the turtle is standing on edge, it will constantly hit the edge in order to stay where they are, hence, we add -10 to discourage moving to edges, something also desirable if we want to have the optimal path for a turtle to draw the image. In this experiment we will be using 6 inputs for NEAT:

1. the x coordinate of the turtle
2. the y coordinate of the turtle
3. the distance to the right edge of the world
4. the distance to the bottom edge of the world
5. difference between x coordinate of the turtle minus and the x coordinate of difference point
6. difference between y coordinate of the turtle minus and the y coordinate of difference point

We no longer need time as an input, since we are now using the difference point as a guidance to the turtle movement. Still, in this experiment, we will still be using Recurrent Neat in order to test this new setting's performance. For each generation, random 20x20 image is generated, then a random difference point is picked out to calculate the fitness function of each genome. The result is show below

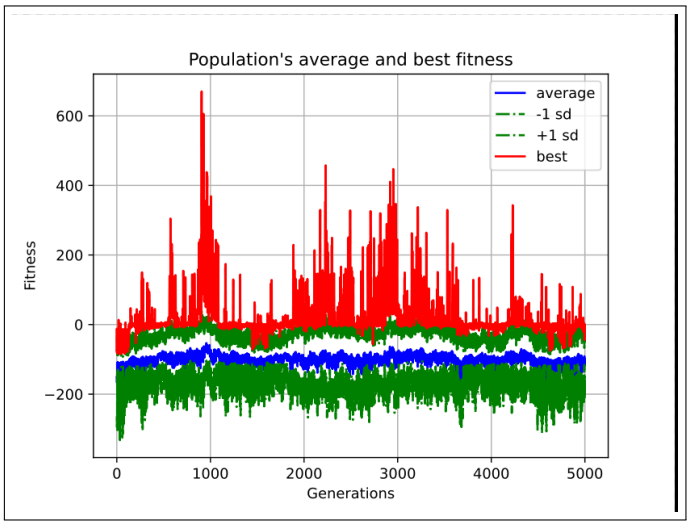

Fig. 5. Best/Average Fitness per generation of Experiment 3 after 5000 generations

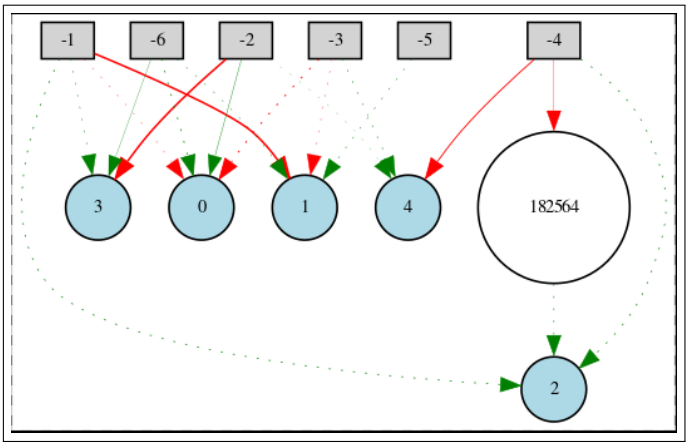

Fig. 6. The final graph of the NEAT neural network

As we can see, the best fitness is much higher than the previous experiments. During the course of training, we realize that most of the time, the best genomes increases by several hundred, but fail to significantly increases in later generations, hence reaching a plateau. This shows that the genomes have gotten the ability to come closer to the designated point, however they fail output the last instruction to draw the dots and gain a significant jump in fitness. Still, figure 5 shows that the network was able to best correctly draw 6 points, which is a great improvement comparing to the previous experiments. The network structure is also much more compact. However, experiment 4 and 5 are what made the project a success.

C. Experiment 4 + 5: Feed-Forward Network with per step fitness function

C.1. Experiment 4: Random difference point

In these two experiments, we use the same fitness function, as well as NEAT input/output as the previous experiments. However, we will be using Feed-Forward Network instead. For experiment 4, a random difference point approach is chosen. Below is the result of this experiment after 10000 generation.

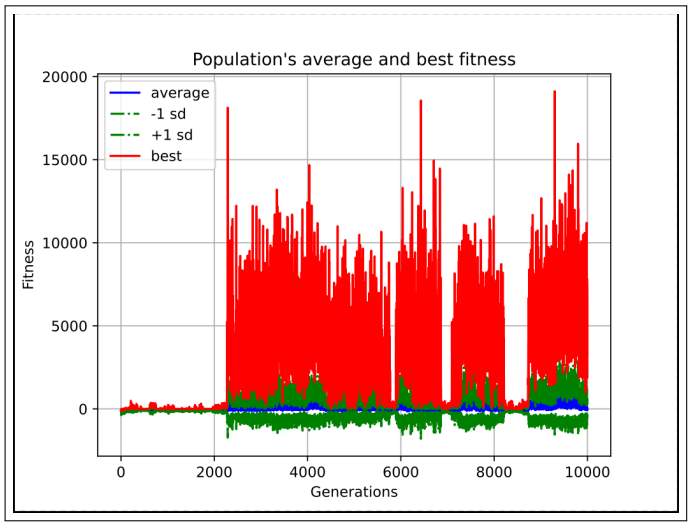

Fig. 7. Best/Average Fitness per generation of Experiment 3 after 5000 generations

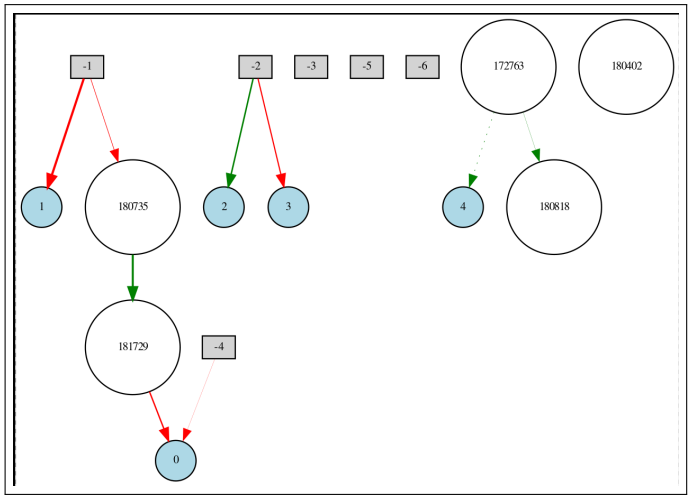

Fig. 8. The final graph of the NEAT neural network

The winner of this experiment solves the drawing problems. During testing, whenever we generate a random 20x20 image, the turtle is able to re draw the image perfectly. During training, we notice that it took a long time for the genomes to understand that they need to come closer to designated difference point. However, after 2000 generations, once the genomes learn to reach the difference points and draw a dot there, the fitness function increases exponentially. While we we training the network for 10000 generations, the genomes achieve optimal fitness after around 2200 generations, fail to find the designated point

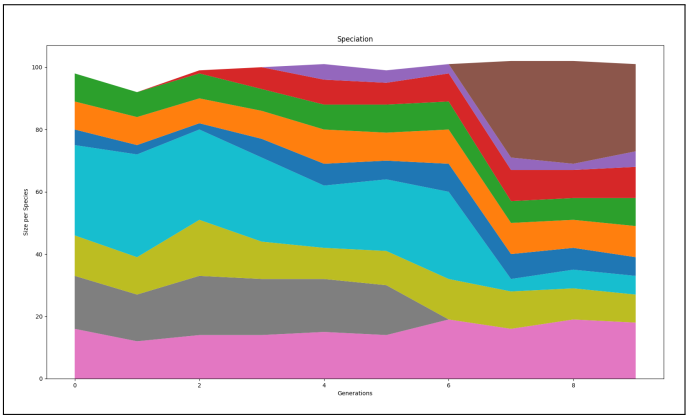

Fig. 9. The species/size per generation of 10 generation after a very good checkpoint

around generation 8000 before gaining reaching the optimal points again. It was quite interesting that while the difference in image was never able to reach 0 during training, we were able to achieve 0 pixels difference every time during testing. This might be because of the strict termination rule we employed in the genomes, terminating every genomes that hit the wall, while in testing, we relief the condition and the best genome has optimal accuracy in every cases.

C.2. Experiment 5: Closest difference point

While we have achieved satisfactory result, we still wanted to perform one last experiment. With the thought that reaching closer points would be easier to reach a random points, we employ the closest difference point approach: taking the closest points to the turtle as training inference. Surprisingly, the result was not as good as the random case, after 3000 generations, the best genome could only draw 20 points correctly before making an errors. The below figures are the result of this experiment.

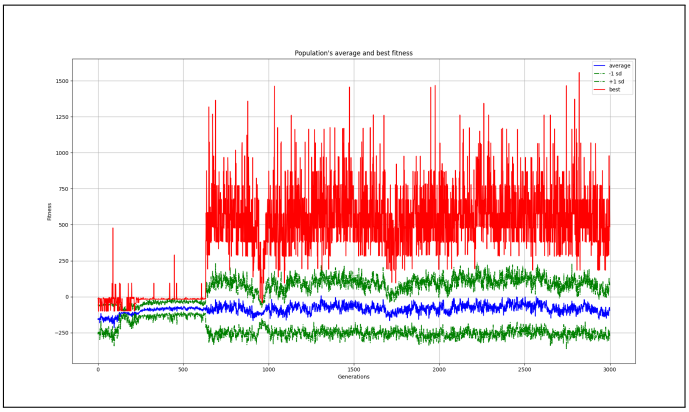

Fig. 10. The best/average fitness per generation of exp 4 after 3000 generations

Surprisingly, the best genome never got a score higher than 20000, even though the population had a very good start, learning how to draw at a very early stage.

4. ABLATION STUDY

In the first few experiments, when we look at the species graph, it is clear that the network does not have enough information

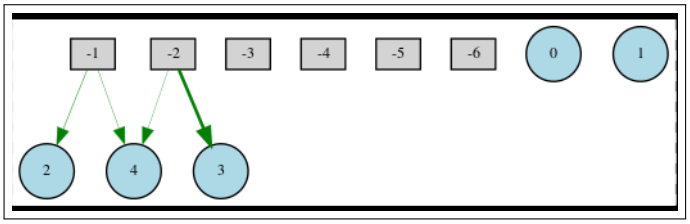

Fig. 11. The final graph of the network

as the search space was too large. Hence the variety of the species were not enough for the population to learn the search spaces. Also, the network of the first 2 experiment are much more complex than the best genome of experiment 5, this proves that the network does not have enough information to discard or add node. The network in figure 8 is very simple and does not even use 2 inputs to determine the output. It shows that this genome has correctly capture the neccessary movement of the turtle while remaining simple and light weight.

In the last three experiment, contrary to our initial belief, feed-forward neural network works better than recurrence neural network in this case. This might be because the turtle does not need as much recurrence information as we anticipated, but only need spatial information of its location and the designated difference points. Hence, it might need more complex representation of the input rather than a recurrence one.

Finally, we believe the choice of closest difference point limited the ability of the genomes to generalize. Since the turtle usually starts in the left corner, closest points to the turtle will most likely to be on the bottom right side. Hence, making it harder for the genomes to generalize and harder to learn all the scenario. In our own experiment, we found that the genomes will learn to draw very early, having had a few correct drawn in the first few hundred generations, and they will eventually have higher fitness and learn to generalize, however, it requires a much longer generation than in the case of training with a random difference approach.

5. SOME EXTENSION

During the training period, as we only pick out a point in the difference matrix, we make no assumption of the initial landspace of the turtle. Hence, if we modify the action draw of the turtle to erase whenever the current position of the turtle is occupied, we have the turtle modify the landscape to match the benchmark image.

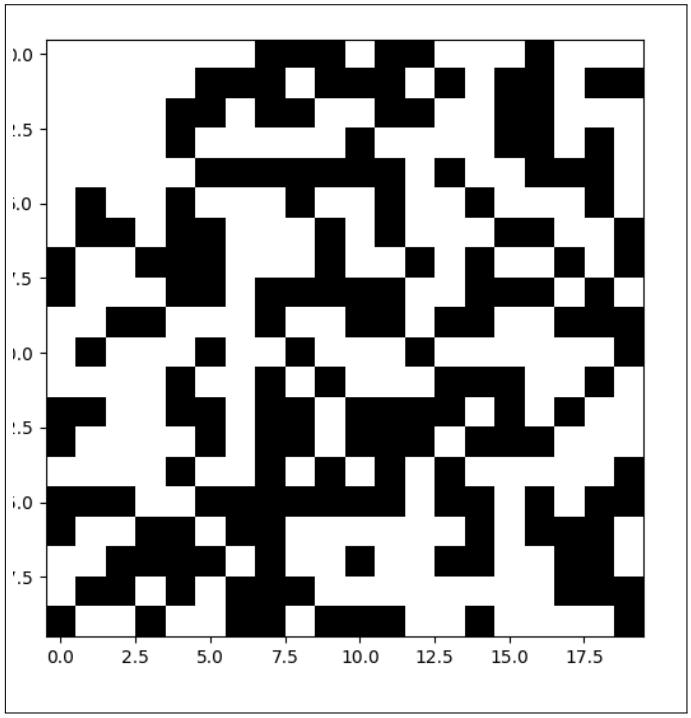

Fig. 12. The random generated picture

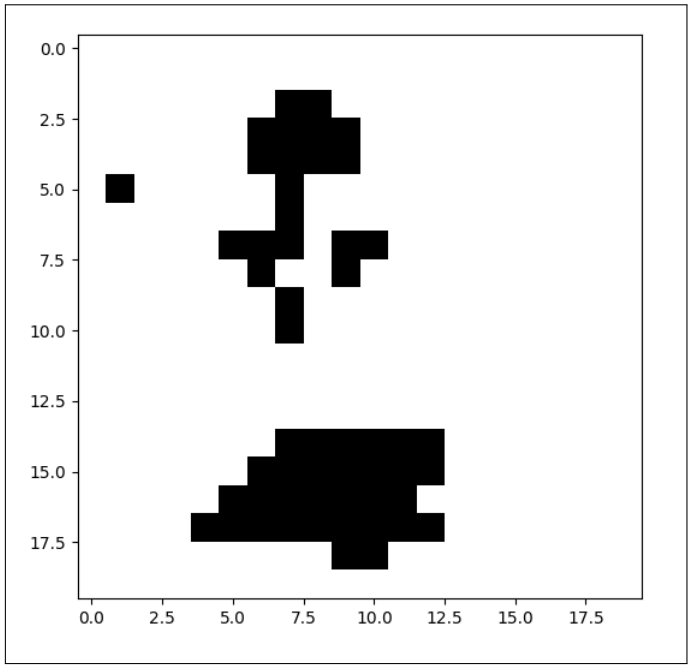

Fig. 13. The inverted Mona-Lisa after the Turtle has erased and add in the correct points

6. CONCLUSION AND FUTURE STUDY

Using NEAT, we were able to evolve a turtle that can translate its landscape to mimic a desired picture. Not only did we solve the mona-lisa problem, but with a random picture input in, the turtle can now translate however the landscape look like to match the desired input pictures. While this small projects only

work for 20x20 images. The input of the final network does not indicate any reliance on image size. Hence, this project can be extended to images of any size, as the input only requires the difference between the input images and the existing landscape of the turtle. Without the time and resource constrain, we will try to perform an experiment with just 2 inputs on arbitrary image sizes to determine if the turtle can still learn to replicate the image well. The success of such experiments will emphasizes the power of Evolutionary Algorithms as a tool to solve agent-based applications.

REFERENCES

1. G. S. Hornby and J. B. Pollack, Comput. Graph. **25**, 1041 (2001). Artificial Life.
2. G. Hornby and J. Pollack, "The advantages of generative grammatical encodings for physical design," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546),* , vol. 1 (2001), pp. 600–607 vol. 1.
3. S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," (2018).
4. K. O. Stanley and R. Miikkulainen, Evol. Comput. **10**, 99–127 (2002).
5. K. O. Stanley, Genet. Program. Evolvable Mach. **8**, 131 (2007).
6. C. Miguel, C. Feher da Silva, and M. Netto, "Structural and parametric evolution of continuous-time recurrent neural networks," (2008), pp. 177–182.
7. S. Sharma, S. Mishra, N. Deodhar, A. Katageri, and P. Sagar, "Solving the classic snake game using ai," in *2019 IEEE Pune Section International Conference (PuneCon),* (2019), pp. 1–4.
8. J.-F. Yeh, P.-H. Su, S.-H. Huang, and T.-C. Chiang, "Snake game ai: Movement rating functions and evolutionary algorithm-based optimization," in *2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI),* (2016), pp. 256–261.