LEARNING-BASED NEUROEVOLUTION

Ho Ko

h25ko@uwaterloo.ca

ABSTRACT

Neural network is capable of learning non-linear problems and genetic algorithm is capable of exploring undetermined solutions. Most of the solution exists in the world nowadays either use the neural network with back propagation or only the later one to solve the problem, however by emerging concepts from both methods, a more general model can be conducted to solve complex reinforcement learning problems.

Index Terms— Neuroevolution, Neural network, Genetic algorithm, Memetic Algorithm, Deep learning

1. INTRODUCTION

Genetic algorithm(GA) is widely known to be capable of solving wide range of reinforcement learning problems, and one of the main difficulties is to define an appropriate domain for GA to explore the best possible solutions. As neural network (NN) can represent and function as countless complex non-linear transformation, assigning the chromosome, the encoded solution in GA to represent NN architecture will endow the possibility of exploring and finding more complex non-linear solutions.

The algorithm that combines GA and NN is also known as neuroevolution (NE), by using the evolutionary method from GA to evolve the NN, it has proven to be successful in solving numerous reinforcement learning problems.[1][2][3][4][5] Although this concept is promising, most of the implementation did not utilize the full potential of neural network especially the learning process with back propagation. A recent article[6] in nature mentioned that the trend of hybridizing neuroevolution and deep-learning is emerging, as better performance and result can be produce by combining ideas from these two areas.

In this paper I introduce a model to implement the concept of GA in evolving the architecture of NN and make good use of back propagation that can be applied and fine tune the neural network.

2. BACKGROUND

In this section, I will address the related concepts used in neuroevolution.

2.1. Multilayer Perceptron

One of the basic forms of neural network (NN), also known as the multilayer perceptron (MLP) sends the input data through fully connected layers until it reaches the output layer.

Each neuron sum up the input data with corresponding weight and an non-linear activation function. If a MLP is well trained, as neuron consider a certain combination of different input that is critical to the deterministic of the output, the neurons can also be considered as having the function of feature extraction.

By feeding forward and process the input data layer by layer, more abstract and higher level of features are extracted, and complex non-linear transformation can be represent by this network.

Take the neural network structure in Fig. 1 as an example, the MLP have an input layer of size 4, a hidden layers with three neurons and two output at the last layer. It can be viewed as three features are extracted directly from the input, and the two outputs are decided according to those three features.

It has been proven that the weight of a neuron in a well trained network is equivalent to feature extraction, and this concept is also being utilized in TRANSFER LEARNING, ENCODER-DECODER, etc. In neuroevolution model, the parameters of the neuron in a successful individual can be

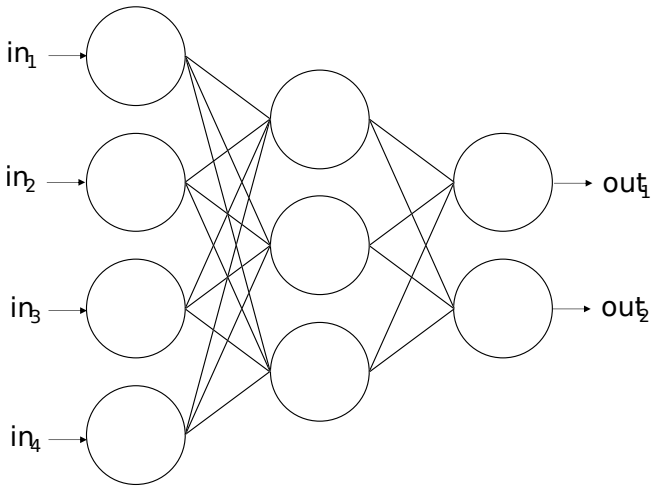

Fig. 1. Multilayer perceptron with the structure of 4:3:2

2.2. Genetic Algorithm

Genetic Algorithm (GA) was designed base on the concept of how natural selection make adaptation to the environment be possible. Most animals, even in the same species have different genetics, and that variety increases the possibility that some individuals in the species can withstand and have a higher adaptation to the environment and further increase the total species size.

The variety of individuals exists because of the gene mutation mechanism and the crossover process during the production of the next generation. Similarly, by encoding any possible solution to a descriptive string, it can be process just like the chromosome within natural selection. By maintaining a constant population of individuals(solution in this case), remove the solutions with lower fitness score and generate new offspring with the combination of the better solutions in each iteration, the best solution within the group is guaranteed to be improved constantly.

A critical problem that have to be solved carefully in most of the GA implementation is the encoding method. The encoded string, or chromosome stored in each individual must not only represent the corresponding solution, it should also make sense when doing crossover operation with two chromosome.

2.3. Neuroevolution with GA for NN parameters

Pre-define the topology of NN structure and use GA to discover the best parameters is a typical traditional implementation of using neuroevolution to solve highly complicated problems. The NN topology defined at first have to be complex enough for solving the problem, but not too complex so that the exploration space for GA is not too large. The solution can usually be found simply by try and error for a appropriate NN topology, but it will not be the most optimized or efficient structure.

Fig. 2 shows a typical implementation of NE. Each neural network individual is being evaluated as a whole, and the crossover operation explore different combination of the neuron from successful individual. According to the example, if not all five neurons are extracting important features correctly, the fitness score of that whole NN will be low, and therefore possible of being eliminated from the pool.

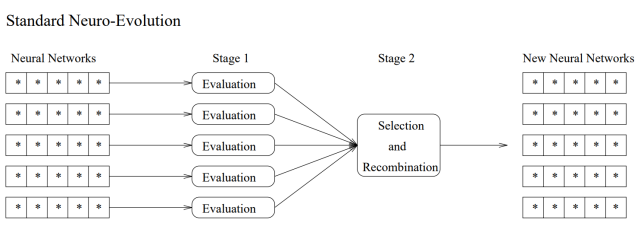

Fig. 2. Standard Neuroevolution[7]

It is ideal when all existing NN have well functioning neurons. But during the early stage of the exploration process, many neurons with proper feature extraction function will be eliminated due to worse behaving neurons within the same NN. This is highly inefficient and that is why SANE model[7] is purposed.

The symbiotic, adaptive neuroevolution (Sane)(Fig. 3) consider every neuron as an individual instead of the NN as a whole, and by evaluating the fitness score of each neurons, no neuron with beneficial function is being eliminated. The evaluation method(Fig. 4) is to randomly combine neurons to a NN structure and update fitness score of the neurons according to the performance of the temporary constructed NN in each iteration. After a number of iterations, the fitness score of each individual neuron is considered base on numerous combinations and scenarios, therefore largely reduce the possibility of having bias.

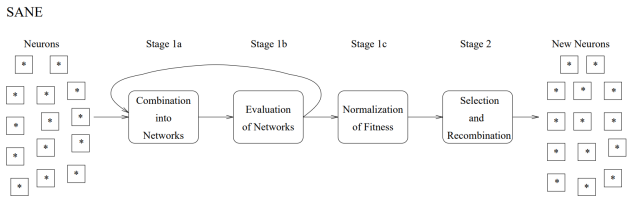

Fig. 3. SANE[7]

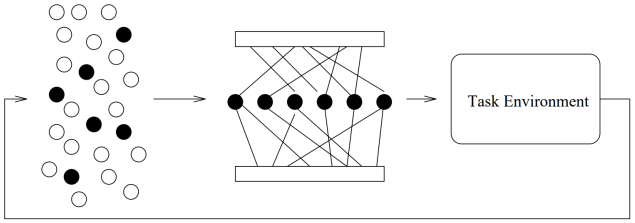

Fig. 4. Evaluation method in SANE[7]

One thing do have to be caution is that the reason each neuron can be evaluated and compare side-by-side is they are all extracting features directly from the input data. All variance and the performance of the individual are decided according to how the neuron process the data, but this will only be the case in NN with one hidden-layer. Neurons in the second or deeper layers will be extracting features extracted from the previous layers, and the performance of that neuron will be highly dependent on the specific combination of neuron in all the previous layers.

2.4. Neuroevolution with GA for NN topology

As the structure of a NN model will have a great influence on the performance, a number of NE models focus on this aspect

and uses GA to explore the best NN topology for solution.

The EPNet[8] is one of the model, it uses simulated annealing(SA) to explore possible improvement to the NN structure. Although the model uses the general concept of GA as it maintain a number of ANNs and keep exploring better candidates, it did not fully follow GA due to the decision of using direct encoding to represent structure of nodes and connections of the NN. After crossover, permutation problem might occur and invalid NN will be generated, so EPNet used SA for exploration instead. On the right side of Fig. 5, multiple adjustments and modifications to the topology including increase or reduction of the size of hidden layer and connection and were being explored. Evaluation to confirm the improvement were done in each iteration, and NN structure with the best performance will remain in the pool.

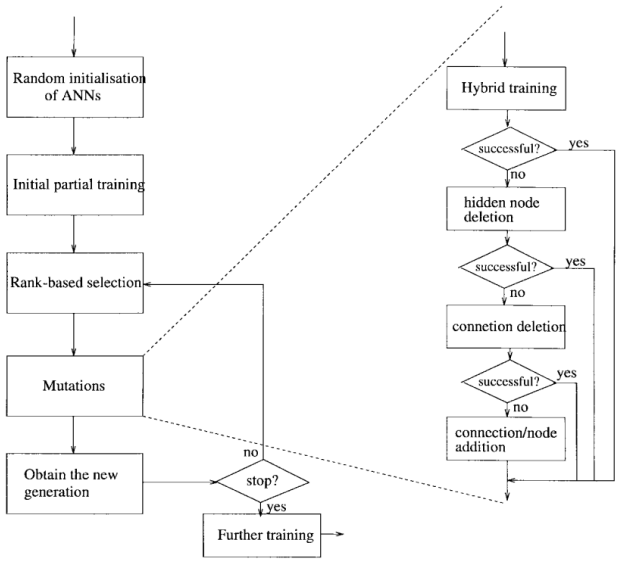

Fig. 5. EPNet[8]

Another method of exploring NN topology with GA is by applying an advanced version of GA called memetic algorithm (MA)[9] to NE model. By adding a local optimizing process for every individual in each iteration, improvements are no longer limited to only occur on offspring, the agents can also improve and fine tune itself over time.

For example in a memetic evolution of DNN[10], the author introduced Gaussian Mutation, a local optimizer to improve the topology of DNN structure. And by applying it to MA (Fig. 6), it was implemented as a NE model. Similar to how GA works, a population of NN individual are initialized at the beginning, offspring are generated by crossover operation and in addition to that a local optimizer are applied to explore the better topologies from existing individual.

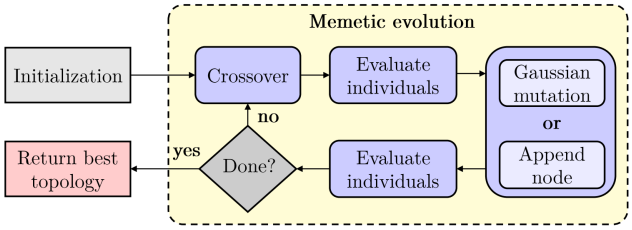

Fig. 6. Memetic evolution of DNN[10]

2.5. Neuroevolution with GA for NN topology and parameters

There are a few NE models that utilize GA to find the most optimized structure with the best parameters at the same time, and among them, NEAT is the one that is most widely being used nowadays.

The revolutionary model introduced in 2002 called NeuroEvolution of Augmenting Topologies (NEAT)[11] is often being applied for countless reinforcement learning task. It is a NE algorithm which utilizes GA to explore not only the most optimized NN topology, but also the parameters within it.

This model uses a direct genetic encoding method to stores connect genes as genome, which include the connection detail of input nodes, output nodes and the weights.(Fig. 7)

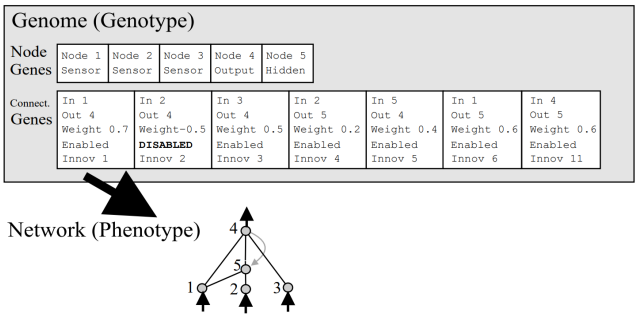

Fig. 7. A genetic encoding mapping example in NEAT[11]

Similar to what EPNet has encountered, if applying the traditional crossover operation from GA, permutation problem will occur and the NN topology will not be guaranteed to be valid. So the author designed an operation as an alternative to the crossover operation.(Fig. 8) It combines matching genes from both parents according to the reference number, which is called ”innovation number” in the paper. This removes the possibility of having redundant information and avoids losing critical connections to important nodes.

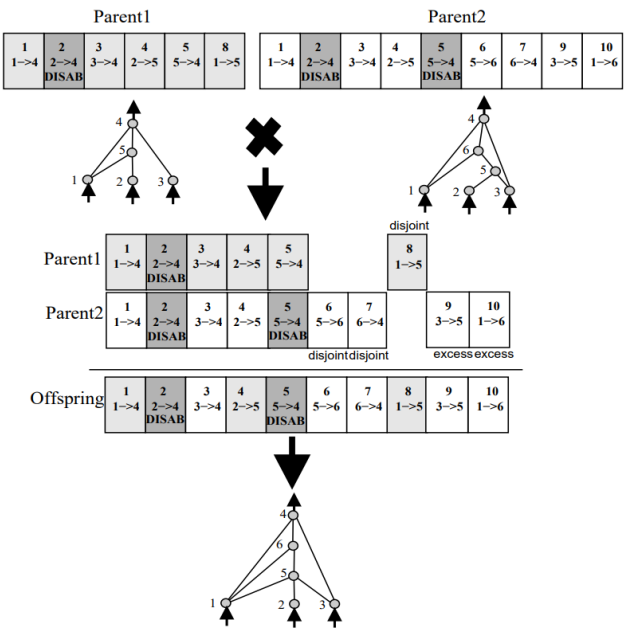

Fig. 8. Alternative operation to crossover in NEAT[11]

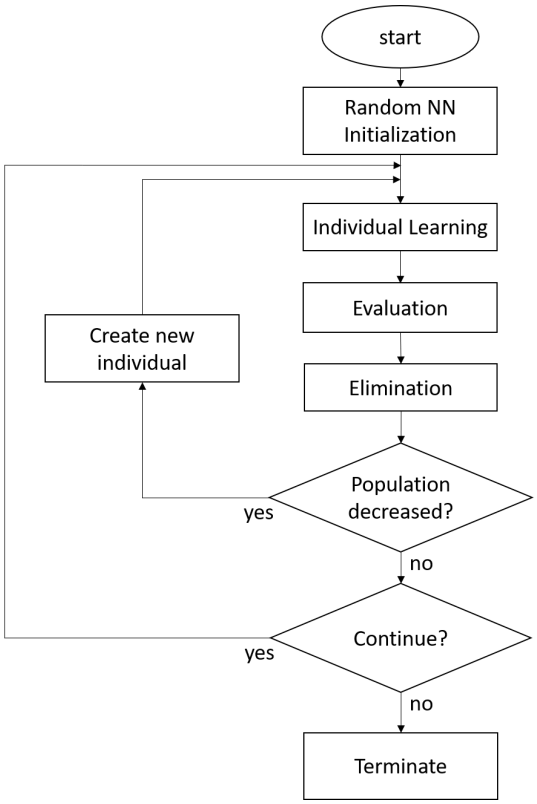

Fig. 9. Major steps of LBNE

3. LEARNING-BASED NEUROEVOLUTION

Genetic Algorithm was designed based on Darwinian evolution, memetic algorithm was invented from the concept of Dawkins' notion of meme. Inspired by the multiple natural phenomenons including those mentioned above, I proposed this new model as a general solution to solve complicated reinforcement learning problems.

In the natural world, most of the intelligent behavior such as hunting techniques and flying skills observed from animals are not inherited. They learn these knowledge from parents, from peers or even by the experience they encountered after birth. Since most of the advanced knowledge was acquired by the process of learning, this concept should be implemented and be used in NE.

Both EPNet[8] and MA[9] uses evolutionary algorithm (EA) to explore different NN topologies and perform learning task on each individual. The problem is that it is based on if the learning task is well defined and being considered as the ground truth. It is not always the scenario in the real world, as numerous complicated problems especially for reinforcement task which no absolute answers exists.

I introduce the learning-based NE(Fig. 9) as a more general solution to reinforcement learning problems. This model uses GA to explore both NN topologies and parameters, but also include the two learning mechanisms inspired by MA for each NN individual.

3.1. Framework of the model

The major steps of the model are as following

1. Initialize a population of N NN individuals with random topology and weights.
2. Improve each individual by applying back propagation with random input and the corresponding sample output from individual with a higher fitness score. In the first iteration, no individual is better than others, so this step will be skipped.
3. Evaluate the fitness score of each NN
4. Eliminate the worse behaving individual or those that did not reach the minimum fitness score
5. Maintain N individual in the population pool by creating new ones.
6. Repeat from step 2 until the best individual is satisfied.

3.2. Genetic Encoding

In order to find the most optimized NN topology and the most appropriate corresponding parameters, both the structural details and weight data should be stored inside the chromosome,

and usually a hybrid encoding method such as the one used in NEAT is applied. But the learning concept being considered in this model does not apply to NEAT directly. As a result, I decided to use two chromosome to represent one individual NN, one for the topology data and one for the parameters.

A MLP can represent and function as most of the feed forward NN if the weights are fine tuned. So for simplicity, this model uses MLP as the framework for every individual NN. The chromosome that stores the topology data is an array of integers, each indicating the the size of the corresponding hidden layer. The crossover operation will simply be the reconstruction of an array where the first half comes from parent 1 and the second half is inherited from parent 2.

Neurons in different layers represent different abstraction level of feature extraction. So for the chromosome to store the NN parameters, I decided to store the weights according to the number of layers where they are located. This allows the feature extraction function of certain level being inherited to the same level in the new NN. And because of this hierarchy storing concept, crossover will be operate on each level to maintain this property. Caution that the crossover point should only be located in between neurons, else the feature extraction function of the neuron where the crossover point is located will be destroyed and no longer function as before. A crossover example is showed in Fig. 10, where crossover point x_1 and x_2 are located in between neurons, and the neuron can function normally as all corresponding weights are preserved.

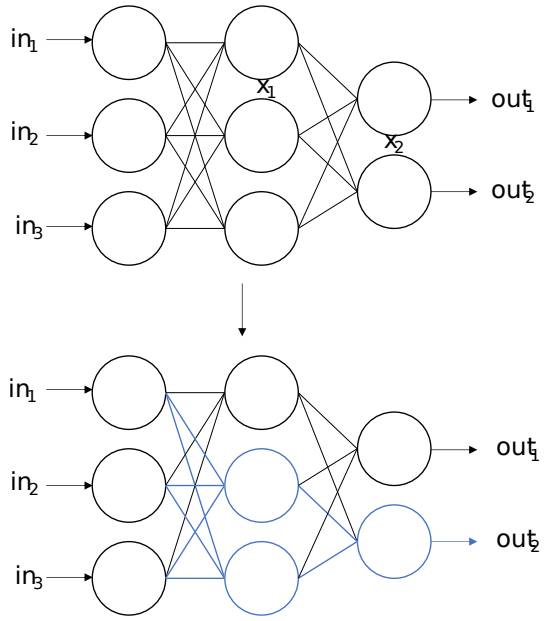

Fig. 10. crossover for inheriting NN parameters

3.3. Learning process

There are two learning processes, the first one happens when a new individual is created, and the second occurs during competition within the population pool. Every individual is born with either randomized initial parameters, or the combination that were inherited from parents. The probability of the individual can have a good performance right after birth is very small, and the first learning process steps in.

Before entering the society, human will learn as much as possible from books, where the top discovered knowledge were stored, and try to shape oneself to the best possible state. Similarly, when a new NN offspring is being created(Fig. 11), a back propagation training processing will be applied with sample input and sample response of the role model, the best individual ever exists. Theoretically, the individual will obtain enough knowledge and be able to largely improve itself and rising the possibility of becoming a better individual.

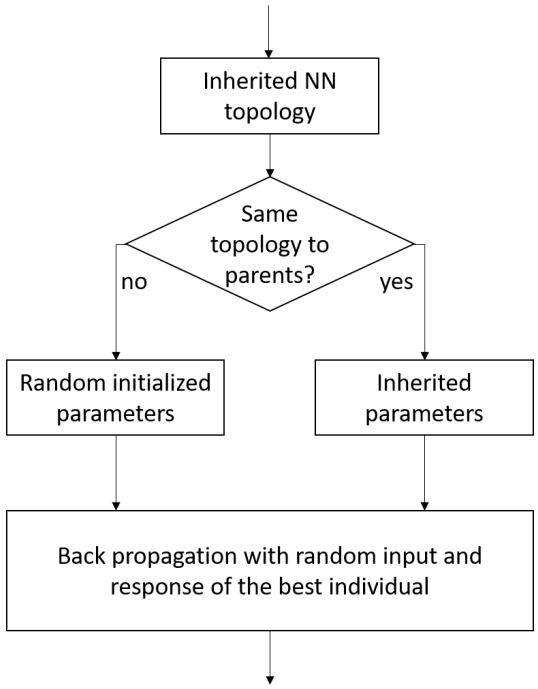

Fig. 11. Process of generating new individual

The number of epoch and learning rate should be carefully setup, as both over fitting and under fitting will result in a significant drop of fitness score. Also, the back propagation iterations for training a randomly initialized parameters offspring should be way more than the one with inherited parameters. As the later one have functioning feature extraction neurons before the training process, it is equivalent to transfer learning to some extends.

We human will never stop learning, even after graduating from school. The influential and knowledge learn from peers is always underestimated, and that actually helps im-

prove ourselves to be more adaptive to the society. The same concept were applied in this model, and each individual will keep on learning from the better "peers" to adjust its own NN parameters.(Fig. 9)

An interesting concept can also be implemented in this learning process, as agents, especially the long living one should be smart enough to keep on surviving and no longer need to keep learning. Only after a certain amount of time if the agents are performing well enough, computational power can be saved for other usage.

Since the NN parameters of a neuron, or known as the feature extraction function, the knowledge that one individual owns are most likely be learnt and adjusted by the process of learning, it seems to be contradiction to pass this knowledge directly to the offspring as according to my core idea of applying concepts from nature, human are not born with knowledge from their parents. But actually, according to a recent research[12], DNA methylation and non-coding RNA, which are affected by our daily life might make a difference in the next generation. For example if an athlete is constantly exercising regularly, the state of the body is well prepared for growing proteins and translating energy, the off spring will have a higher possibility to be more capable of doing sports. Same idea applies in the model, although the parameters are being inherited by GA, it is combined with noises due to mutation and the crossover operation that combined neurons from different sources will disrupt the original information processing flow. With that being said, it still have a more advantage in learning what its parents are capable of as similar feature extracting neurons exists, just like the mechanism from the nature.

4. EXPERIMENT

The whole experiment is written in pure JavaScript programming language with two libraries, three.js and tensorflow.js. Both libraries are free and open sourced, and supports GPU acceleration. Three.js uses WebGL to utilize GPU acceleration when rendering 3D graphics in web browser. Tensorflow.js is a released in 2018, designed to run machine learning related task in web browser or other JavaScript supported area such as Node.js.

A driving task is simulated in a 3D virtual environment(Fig. 12). There are five ultrasonic sensors with different angles in the front of the car which will detect the obstacles within a certain amount of distance. Agents have full control over turning the wheel, accelerating, braking or do nothing.

The input size of the MLP model for each individual is 6, 5 corresponding to the sensor's input and one indicating the agent's current speed. There will be two outputs each with a range of -1 to 1, one is for controlling the car's direction corresponding to the range from $-\pi/8$ to $\pi/8$, the other controls the acceleration of the car.

An agent will be eliminated if it crashed on any walls or

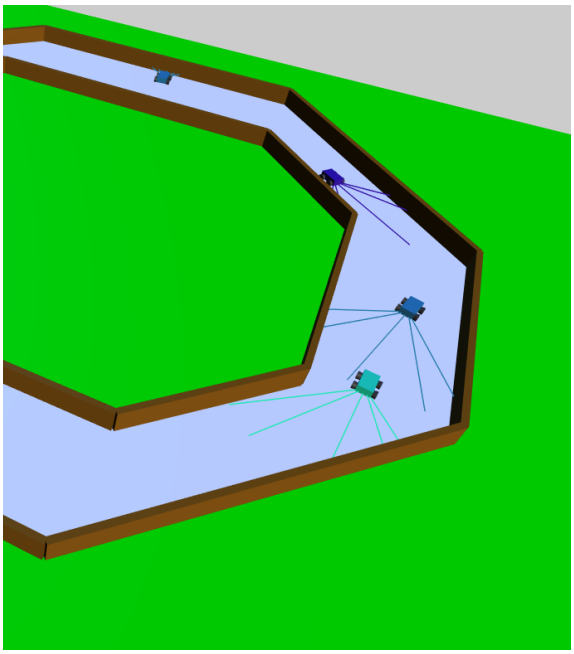

Fig. 12. A circular race track as the branch mark for the experiment

if the evaluation system determined that it is not making any progress, for example parking at the middle of the road or backward driving. The evaluation function is designed as :

$$Score_t = Score_{t-1} + Speed * 2 - |Direction| * 3 - 1$$

The fitness score will be accumulated along iterations, encouraging agents to stay alive as long as possible. The variable $|Direction|$ is equivalent to the angle offset, as smooth driving route is what we commonly want ideally, zigzag driving should be punished. Also, in different simulated environments, the cars might have enough space to rotate and move in a small circle, that is why the coefficient of the angle offset factor is the largest. Lastly, the constant negative 1 is to eliminate those that are not making progress significant enough.

The learning-based NE(LBNE) will be implemented to challenge this task, and GA model will also be implemented as a baseline for comparison. The performance index will be the ratio of the fitness score of the best agent to the number of agents that failed. Each test will stop when the highest fitness score reached 1000, meaning a solution is found, and it will also stop if 1000 agents failed before that, meaning it failed to find a solution. This indicates both the robustness and efficiency of the model, as given a limited number of tries to success or find the solution is related to robustness, and average tries needed for convergence is related to efficiency.

Notice that this experiment is conducted in a local machine, a laptop with GTX-950M. Although usually if not comparing the convergence time, the processing speed do not matter. But due to the asynchronous programming in

JavaScript for acceleration of the frame update mechanic, precision will be traded-off.

5. PERFORMANCE AND RESULT

The agent's MLP topology will remain the same for comparing the LBNE model and the traditional GA baseline to highlight the advantage of the learning-base mechanism in LBNE. Each test will maintain the same population size of 10, and the result is the average of 10 consecutive tests.

The result(Fig. 13-15) shows that the LBNE model out performs GA in average score and success rate in each cases. The sigma, also known as the error rate is an indication of how consistent the result is. But do notice that the due to the large difference in average score, the sigma value no longer is meaningful and comparable.

	Average Score	σ	Success rate
GA	1.26	0.878	60%
LBNE	2.94	1.305	90%

Fig. 13. Case 1, agent NN topology is MLP with 6:3:2 structure.

	Average Score	σ	Success rate
GA	0.71	0.656	30%
LBNE	2.47	0.694	100%

Fig. 14. Case 2, agent NN topology is MLP with 6:5:2 structure.

	Average Score	σ	Success rate
GA	2.79	3.109	80%
LBNE	3.65	2.589	100%

Fig. 15. Case 3, agent NN topology is MLP with 6:5:3:2 structure.

6. CONCLUSION

By applying concepts and ideas from how animal acquire knowledge in nature, The learning-based neuroevolution(LBNE) model is created and shows some promising result in the experiments. It is robust and efficient if considering each agents as a unit of resource. Although computational wise it might be power-consuming, there are not that much shot and opportunity for each individual to try and error, for example an exploration or complicated reinforcement task need to be solved and the number of robots is limited, this model will certainly be a better choice comparing to GA.

The LBNE model also have a very high potential in adapting to the fast developing machine learning techniques, for example the concept of Drop-out layer can be easily implemented into the topology of the NN individual with a custom defined crossover operation. Because this model is the combination of numerous concepts, any new discoveries in these area can benefit to improving this model.

7. REFERENCES

[1] Dario Floreano, Peter Dürr, and Claudio Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[2] Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen, "Real-time neuroevolution in the nero video game," *IEEE transactions on evolutionary computation*, vol. 9, no. 6, pp. 653–668, 2005.

[3] Faustino J. Gomez and Risto Miikkulainen, "Solving non-markovian control tasks with neuroevolution," in *Proceedings of the International Joint Conference on Artificial Intelligence*, San Francisco, CA, 1999, pp. 1356–1361, Kaufmann.

[4] Kenneth O. Stanley and Risto Miikkulainen, "Efficient reinforcement learning through evolving neural network topologies," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, San Francisco, 2002, p. 9, Morgan Kaufmann.

[5] V Scott Gordon and Darrell Whitley, "Serial and parallel genetic algorithms as function optimizers," in *ICGA*, 1993, pp. 177–183.

[6] Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.

[7] David E. Moriarty, *Symbiotic Evolution Of Neural Networks In Sequential Decision Tasks*, Ph.D. thesis, Department of Computer Sciences, The University of Texas at Austin, 1997, Technical Report UT-AI97-257.

[8] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, 1997.

[9] Pablo Moscato, Carlos Cotta, and Alexandre Mendes, *Memetic Algorithms*, pp. 53–86, 01 2004.

[10] Pablo Ribalta Lorenzo and Jakub Nalepa, "Memetic evolution of deep neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, New York, NY, USA, 2018, GECCO '18, p. 505–512, Association for Computing Machinery.

[11] Kenneth O. Stanley and Risto Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[12] You-Yuan Pang, Jui-Hsien Lu, and Pao-Yang Chen, "Behavioral epigenetics: Perspectives based on experience-dependent epigenetic inheritance," *Epigenomes*, vol. 3, pp. 18, 08 2019.