A Survey of Evolving Spiking Neural Networks for Agent Control

Tristin Glunt

University of New Mexico

tglunt@unm.edu

Abstract

The following survey discusses the use and results of Spiking Neural Networks (SNN) in the field of autonomous agents. The introduction of the paper discusses why SNN are important as well as the purpose of specifically studying Evolving Spiking Neural Networks (eSNN). The approaches section lays out the different types of setups, SNN architectures and genetic algorithm (GA) implementations from each of the surveyed papers. Afterwards, we will present the results for each paper. Finally, we discuss our considered opinions on each paper, their different approaches, and possible improvements they could make. The paper concludes with a summary of the sections.

1.0 Introduction

Many advances have been made in the recent decades in the field of autonomous agents, specifically autonomous agents that have not been explicitly programmed with pre-defined rules. However, many of these advances have been made with models neglecting biological reasoning or inspiration. Because of this, it is important to study truly traceable and biologically feasible approaches to artificial intelligence. That is where SNN come in to play, especially because of the work done in [1], showing their computational efficiency. Using genetic algorithms as a learning mechanism will also keep the models following biological inspiration, as the canonical backpropagation methods today have no biological inspiration.

SNN are unique with their ability to encode temporal meaning behind an input. That is, SNN can be heavily impacted by the rate of stimulus being fed into their inputs. It is currently unknown how important temporal encoding may be in biological systems. However, it can be modeled that with more incoming connections into a neuron, the less each individual connection matters. Despite this, the overall spiking rate is still of interest as no one is certain about the importance of the stimulus rate.

This paper will be looking at approaches from multiple problem spaces using SNN for controlling their agents to accomplish a designated task. For each problem, a different type of neuron model was used, thus we will be discussing the differences between each neuron model. Afterwards, we will discuss each problem's use of genetic algorithms for learning. Finally, we will discuss approaches to the overall SNN architectures and their implementations of coding stimulus.

2.0 Approaches in the Literature

In this paper, we will be looking at several other papers using eSNN: *Evolving Spiking Neural Networks of Creatures Using Genetic Algorithms* [2], *Evolving Spiking Neural Networks for Robot Control* [3] and *Evolving Spiking Neural Network Controllers for Autonomous Robots* [4]. The following subsections will discuss each of these papers unique problems and how they used eSNN to solve those problems.

2.1 *Setup of simulated autonomous creatures with eSNN*

In [2], a simulated environment is created. The environment consists of one creature, where the creature is placed in a 2D, 50x50 grid, with 30 randomly placed foods across the grid. The creature starts with a base unit of energy at 675. As time dwindles, so does the creature's energy. The task for the creature is to collect food to increase the creature's energy level, as an energy level of 0 is the death of that creature.

The movement behind the creature in the 2D environment is decided by its SNN. The neurons in the SNN are modeled after Izhikevich neurons [5]. The SNN consists of 108 neurons; 36 neurons are used for input, 36 for output and another 36 are randomly connected in between the input and output based on Reservoir Computing [6]. Every 9 neurons of the input and output layers are used for a single direction: forward, backward, left or right. The input to the SNN is based on the distance between the creature and different food. The closer the creature is to food, the higher stimulus of input the SNN will receive. The experimenters decided to go with frequency coding, where a higher stimulus means a higher rate of spikes being output from the neuron. Thus, the closer the creature is to a certain food, the more likely it is to move in that direction.

Using the same general idea, another simulation was built with a colony of 100 creatures, where each creature had a much simpler SNN. Instead of 108 neurons, each creature only had 12 neurons, where there 4 input neurons, 1 for each direction, 4 reservoir neurons, and 4 output neurons, one for each direction. It is worth noting that when initialized, each creature within a colony has identical SNN with identical weights. The energy levels of each creature in a colony were initialized at 75 units vs. the 675 units of the single complex creature. The colonies of simple creatures used the same decision making as the single complex creature.

To compare the performance of a single creature vs. a colony of creatures, the authors developed one more approach. Instead of the single complex creature, a single simple creature was built using the same structure as that of the creatures in the colonies.

The learning mechanism behind the creatures is genetic algorithms. The genetic algorithm initializes a random population of 100, where each complex creature in the population has random network connections, axonal delays, and neuron types (inhibitory/excitatory). For the colony simulation, the GA initializes 100 colonies where each creature within the colonies have the same networks, but the networks differ across colonies. The fitness function for the GA is based on the number of foods the creature finds. The population of the next generation of creatures is based on the following split: 15% of the top performing, 55% from crossover, and 30% from random mutation of the initial population.

2.2 *Setup of autonomous robots with offline eSNN*

In [3], a SNN is used as the controller of a mobile robot. The task for this robot is to get through a maze of obstacles without collisions and reach the end of the maze which contains a bright light source. The robot has two types of sensors on board: an IR sensor to detect when nearing obstacles and a light sensor to detect the source of light of its destination. The wheels of the robot are directed by the SNN.

The robot was implemented with imitation learning. The heuristic rules the robot is supposed to learn are first demonstrated to the robot by an example of a successful run. The sensor values from this successful run are then fed into as input to the offline SNN simulator. The simulator will then try to replicate the outputs used in this demonstration. The sensors feed analog values as inputs into the SNN. A translator between the robot's inputs and the SNN simulator is used to encode the inputs into spikes for the SNN. The neurons of the SNN were modeled by the leaky integrate and fire model used in spike response models (SRM) [7]. Two coding methods were used, the first being a delay coding in which a stronger stimulus meant a neuron would fire sooner than when a weaker stimulus is applied. The second coding was a frequency coding as discussed in the previous sub-section. After the coding is applied, the spiking output from the SNN is then translated back into analog input for the wheel movement.

The network of the SNN simulator evolved over different trial runs in which the goal was to improve the robot's performance. The network parameters, such as the weights of the connections and the synaptic delay are tuned by the simulator based on the imitation learning procedure.

2.3 *Setup of autonomous robots with online eSNN*

In [4], the authors' goal is to have a mobile robot follow the right edge of a table at a defined distance from one end to the other. The mobile robot has two wheels with independent motors, 9 ultrasound sensors and 4 bumper sensors to detect collisions. The robot will be controlled by a SNN where the weights of the SNN are learning online by a GA.

The SNN in [4] uses SRM neurons. That is, it uses leaky integrate and fire modeled neurons with refractory periods after firing. The inputs into the SNN are the analog values from the ultrasound sensors. The outputs of the SNN are analog values for the wheels. The SNN uses delay coding as described in the previous sub-section. The weight values of the SNN are randomly initialized between -1 and 1.

The GA in [4] is an online learning model where the mutation and crossover percentage rate is adaptive [8]. The training of the robots SNN begins with a population of 4 chromosomes, where each chromosome represents a SNN with randomly initialized weights. The robot randomly chooses a chromosome from the population to begin attempting the task. After completing the task, the fitness of the robot is determined by calculating the scaled average deviation from the desired distance and subtracting it from 1. The fitness function will ensure the GA is attempting to minimalize the error from the desired distance. After the fitness is calculated for each chromosome, the adaptive mutation and crossover rate is used to generate the next generation, where the elite chromosome is guaranteed to be in the next population.

3.0 Results in the Literature

3.1 *Results of simulated autonomous creatures with eSNN*

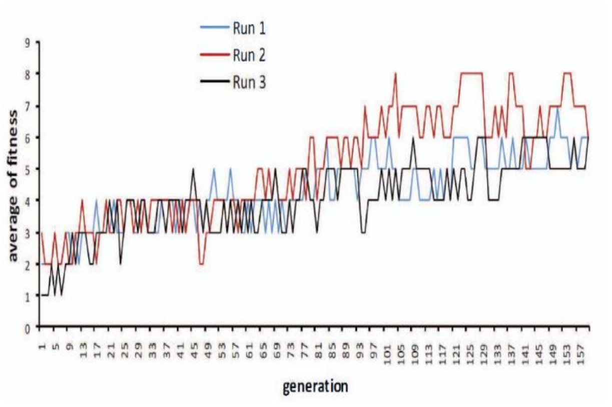

Figure 1: Complex single creature average fitness over multiple runs. (Thanks to [2])

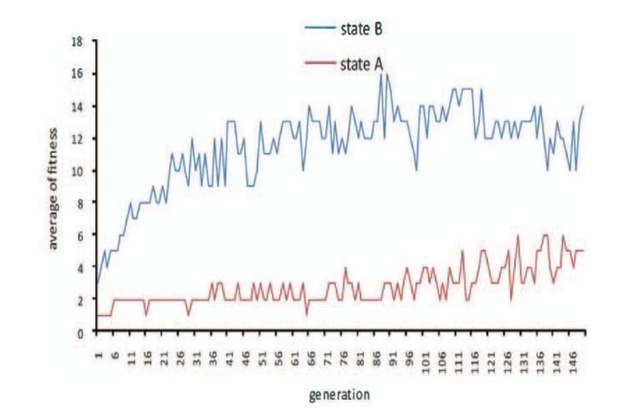

Figure 2: Average fitness of single simple creature (state A) vs. Colony of simple creatures (state B). (Thanks to [2])

In [2], the results of the single complex creature were best demonstrated in Fig. 1, which shows the average fitness of the single complex creature improving with the amount of generations. Remember, the fitness of a creature was calculated simply by the number of foods it found before its energy reached 0. The runs in Fig. 2 were all initialized with a population of 100 creatures.

The colony of simple creatures heavily outperformed the single complex creature, sometimes being able to find all 30 foods. To properly compare the idea of a colony and a single creature, results were analyzed between the colony and the simple single creature. The results were best demonstrated in Fig. 2, which shows the average fitness of the colonies (state B) outperforming the simple creatures (state A).

The conclusion from [2] states the superiority of the colony approach and specifically refers to the approach being highly desirable due its clear reflection in nature. The study is done in hope that the understanding of the morphology of biologically inspired artificial creatures can be better understood by using simulations. One of the key aspects of the experiments were the effects of the use of axonal delays within the SNN.

3.2 *Results of autonomous robots with offline eSNN*

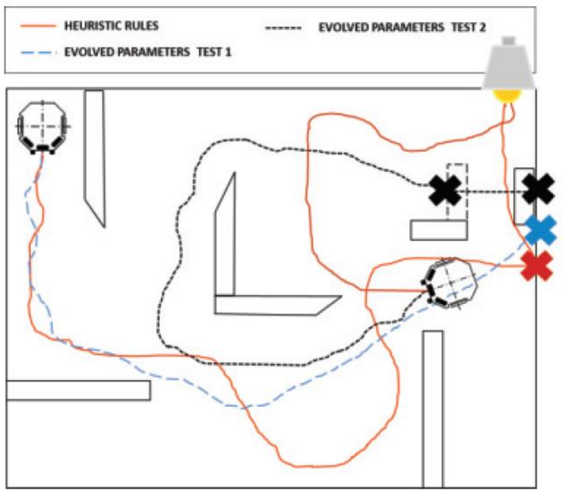

Figure 3: Final results of autonomous robot with final SNN architecture. (Thanks to [3])

In [3], they present their step-by-step empirical process, concluding with the demonstration represented in Fig. 3. They initially started off with what was deemed “a guess” of a SNN architecture. After a couple of iterations of failures with only excitatory synapses, they ended up adding inhibitory synapses from each sensor to the input neurons, as well as directly to the output neurons, which control the wheel motions. This SNN architecture produced the “best” performing robot, where “best” is deemed by looking at the above Fig. 3 and seeing the robots evolved test performance. We see that the robot was first demonstrated a desired run (heuristic rules), in which a set of 48 rules are displayed. The robot then used imitation learning to try and replicate that performance, ending up with the evolved parameters that performed the tests in Fig. 3. The results weren’t completely ideal, but the authors concluded that the general purpose of the project had been accomplished.

3.3 Results of autonomous robots with online eSNN

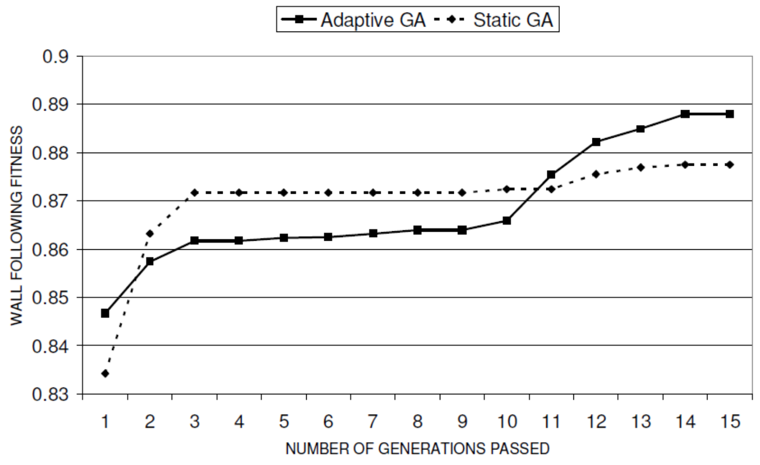

Figure 4: Adaptive GA vs. Static GA fitness over multiple generations (Thanks to [4])

To determine the results of the autonomous robot in [4], the experimenters attached a pen to the back of the robot to get the robot's path. Using the path drawn by the robot, they calculated the robot's fitness as discussed in section 2.3.3. The results are best shown in Fig. 4, where it is clear the adaptive GA outperformed the static GA. However, both optimizers dealt with fighting a local minimum between generation 3 and 9, where finally the adaptive GA was able to get to a lower local minimum.

The experimenters concluded that the SNN was not only enough, but ideal for their nano-robot applications because of the computational efficiency and non-expensive hardware representation of the SNN. These results were promising for future SNN applications where computation efficiency and storage of concern and need to be closely monitored.

4.0 Discussion

Looking through each of the literature's approaches to their given problem, we saw very similar setups. Each paper had a specific task their agent was trying to accomplish whether that be a simulated task or a physical one. Each paper had a different implementation of their SNN neurons and their stimulus coding. Finally, each paper also had a different implementation of their evolving learning algorithm to tune the SNN weights. It was particularly interesting to see each paper their representation for stimulus coding, whether they used a delay coding or frequency coding. To better defend their choice, it would have been nice if they presented on the results of each coding. Furthermore, it seemed as though each paper went only as far as to get the minimal success rate, instead proving their solution was the best it could be for their problem space.

Each paper presents a solution to their initial problem, that is, broadly speaking, building some controller to control an agent in making decisions to complete some given task. However, the solutions in each paper were merely found from empirical tests, and not defended against other possible solutions. For each paper, they choose only certain parameters to tune (usually the synaptic weights in the SNN) and continued. Instead, the authors could have tried surveying a wider range of possible parameters and describing their results for each. I understand this would be a costly operation in time specifically, but it would much further defend their final decisions on an architecture.

In paper [2], one could imagine it being extremely beneficial to better understand the difficulties of the problem if results were gathered from a human agent performing the task. For any test, a human performing it would immediately give the reader a better understanding of the scope of the problem and the interesting behaviors the agent would have to learn to succeed at it.

In paper [3], the results and the evolutionary process were not presented clearly and not presented at all, respectively. There is no explanation for the next iteration of parameters after an unsuccessful test, other than that they were using imitation learning. The overall lack of quantitative results presented makes it difficult to fully understand their successes and failures.

Finally, paper [4] presented their approaches and results in a clean manner, making it very easy to understand. It would be extremely interesting to see if they could successfully build a simulation reflecting the same results as the physical robot had with the same SNN. As they discuss, this is an extremely difficult task and they will face only difficulties that occur due to the physics in the simulation.

I firmly believe it’s worth investing significant amounts of time to have a realistic simulation for any problem, as time can be fast-forwarded in simulations. To get truly remarkable biologically inspired results, one may have to simulate all the time we as humans have been evolving. Despite there being some weaknesses among the papers in the breadth of their solution search, each paper succeeded in developing a biologically plausible controller using biologically plausible optimization techniques with GAs.

5.0 Summary and Conclusion

SNN are complex controllers able to achieve interesting tasks given a virtual or physical agent, much like today's canonical artificial neural network models. However, as it was shown in the surveyed papers, SNN have an advantage in their simple architectures and computational efficiency. SNN can be made easily due to their simple spiking nature and their simple structure allows them to take advantage of temporal input. As we saw, many different types of structures, including different types of neuron models and different types of coding of stimulus, saw success. eSNN were focused in this survey to find biologically inspired learning algorithms unlike today's backpropagation methods. The GAs used in the eSNN were particularly interesting as some conducted online learning, offline learning, and imitation learning.

In [2], we saw an artificial creature face the difficulty of finding food before its energy ran out and learned that a colony of simple creatures outperformed a single complex creature. The simple ensemble approach reflects the ideas taken by random forests [9]. In [3], we saw a physical agent face the difficulty of getting through a maze of obstacles looking for the brightest source of light. We learned that inhibitory synaptic connections were as important as excitatory to improve the performance of the robot using imitation learning. Finally, in [4], we saw a physical agent face the difficulty of online learning with GAs to follow the right-side edge of a table closely. We learned that an adaptive GA can outperform the static GA.

We saw that each paper had certain strengths and weaknesses and overall it would have been interesting if each paper could have presented the best solution to their problem space with defendable results. This is most likely a common theme in neural network solutions due to their massive search space of possible solutions.

Future work to be done in this area is to mainly scale the problems with more complexity and compare results to many of todays canonical models. It would be interesting to start from something simple like the simulation in [2], and compare the best results to a simple ANN. Then, continuously introduce more complexity into the simulations and compare results from eSNN to ANN. This process would be continued until finally one starts to outperform the other. Then, one would analyze these differences and try to determine why this may be happening and what kind of advantage the winner must have to give it the ability to perform better. All of this would be ideal to start out in simulations and then compare results to physical experimentations, as agents interacting with the physical world is the ultimate goal.

6.0 Acknowledgements

I would like to thank Prof. Thomas Caudell for teaching me the necessary prerequisite information needed for understanding these papers. I would also like to thank the Dept. of Computer Science at UNM for giving me other general prerequisite knowledge for understanding these papers.

7.0 References

[1] W.Maass, "Networks of spiking neurons: the third generation of neural network models," *Australian Conference on Neural Networks*, 1996.

[2] E. Eskandari, A. Ahmadi, S. Gomar, M. Ahmadi and M. Saif, "Evolving Spiking Neural Networks of artificial creatures using Genetic Algorithm," 2016 *International Joint Conference on Neural Networks* (IJCNN), Vancouver, BC, 2016, pp. 411-418.

[3] R. Batllori, C.B. Laramee, W. Land, J.D. Schaffer, "Evolving spiking neural networks for robot control," *Procedia Computer Science*, Volume 6, 2011, pp. 329-334

[4] H. Hagras, A. Pounds-Cornish, M. Colley, V. Callaghan and G. Clarke, "Evolving spiking neural network controllers for autonomous robots," *IEEE International Conference on Robotics and Automation*, 2004. Proceedings. ICRA '04. 2004, New Orleans, LA, USA, 2004, pp. 4620-4626 Vol.5.

[5] E. M. Izhikevich, "Simple model of spiking neurons," in *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569-1572, Nov. 2003.

[6] Schrauwen, B., Verstraeten, D., & Van Campenhout, J. (2007). An overview of reservoir computing: theory, applications and implementations. *Proceedings of the 15th European Symposium on Artificial Neural Networks*. p. 471-482 2007 (pp. 471–482).

[7] W. Gerstner and W.M. Kistler, Spiking neuron models - single neurons, populations, plasticity, Cambridge University Press, August 2002.

[8] D. A. Linkens and H. O. Nyongesa, "Genetic algorithms for fuzzy control.2. Online system development and application," in *IEE Proceedings - Control Theory and Application*s, vol. 142, no. 3, pp. 177-185, May 1995.

[9] L. Breiman, "Random Forests," in *Machine Learning* vol. 45, no. 1, pp. 5-32, Oct. 2001