Using Neuroevolved Binary Neural Networks to solve reinforcement learning environments

Raul Valencia
School of Computer Science
University of Auckland
New Zealand, 1010

Chiu-Wing Sham
School of Computer Science
University of Auckland
New Zealand, 1010

Oliver Sinnen
Department of Electrical,
Computer and Software Engineering
University of Auckland
New Zealand, 1023

***Abstract*—With the explosive interest in the utilization of Neural Networks, several approaches have taken place to make them faster, more accurate or power efficient; one technique used to simplify inference models is the utilization of binary representations for weights, activations, inputs and outputs. This paper presents a novel approach to train from scratch Binary Neural Networks using neuroevolution as its base technique (gradient descent free), to then apply such results to standard Reinforcement Learning environments tested in the OpenAI Gym. The results and code can be found in https://github.com/rval735/BiSUNA.**

***Index Terms*—Neuroevolution, binary neural networks, BiSUNA, discrete optimization.**

I. Introduction

The most frequent obstacle by the approach taken by binary neural network (BNN) publications to date is the utilization of gradient descent, given that the algorithm was originally designed to deal with continuous values, not with discrete spaces. NN can give promising results with Int4 [1] or binary in inference but not training. Even when it has been shown that precision reduction (Float32 -> Float16 -> Int16) can train NN at a comparable precision [2], the problem resides in the adaptation of a method originally designed for continuous contexts into a different set of values that create instabilities at time of training.

In order to tackle that problem, it is necessary to take a completely different approach to how a BNN[1]) are trained, which is the main proposition from this paper, expose a new methodology to obtain neural networks that use binary values in weights, activations, operations and is completely gradient free, which brings us to the brief summary of changes made to the published neuroevolutionary algorithm named SUNA [4]:

- Change floating point to unsigned short int values
- Use only logic operations instead of arithmetic functions
- Calculate gaps between individuals with hamming distance

With that context, this paper is organized as follows: a brief introduction to alternative algorithms based on evolution called ”TWEANN” is presented, with special focus on SUNA given its learning generalization characteristics. Followed by the main contribution of this paper: Binary SUNA (BiSUNA), which is then tested on three OpenAI Gym environments, with a slight description and experiments that compares how these binary extensions solve those problems. Finally, a conclusion is outlined with further research needed to explore the possibilities of this new discovery.

[1]For a richer explanation on how BNN work, check [3]

II. Neuroevolution

In general terms, Evolutionary Algorithms (EAs) are optimization procedures based on population metaheuristics, where proposed solutions to a problem are defined as individuals in that group, then subsequent iterated modifications of that population commonly leads to an optimal result. Qualifications of an individual to the environment is graded by design, to which a mechanism of commonly known biological evolution is applied to each character to form a new set in subsequent trials of that group.

When EA clash with traditional NN, they form a new area of research that combines the generalization strengths of artificial NN with the search capabilities of evolution, named as Neuroevolution. One of the many advantages of bio-inspired algorithms is their ability of global optima search in complex multidimensional spaces, and when they are used to train deep NN, their utilization shines [5].

One more algorithm that deserves further explanation is SUNA, which stands for Spectrum-diverse Unified Neuroevolution Architecture [4], a recent addition to the TWEANN family of algorithms that work towards the creation of neural network topologies using mutation operators, starting from a minimal structure and selecting those fittest individuals to the problem at hand.

A. SUNA's Unified Neural Model

Published in 2016, SUNA is a novel TWEANN algorithm with important generalization characteristics: neurons with different time scales, neurons that inhibit/excite others, synaptic plasticity and feedback loops; as an optimization algorithm, it works with a wide range of problems without any prior knowledge. In its original paper [4], it demonstrated how to solve reinforcement learning examples, like Mountain Car, (Non-Markov) Double Pole Balancing, Multiplexer and Function approximation, evolving populations from scratch and resolving those problems with better scores than NEAT [6], a widely popular TWEANN algorithm.

The main improvement of SUNA's paper was the creation of a TWEANN algorithm that joined multiple characteristics into one simple procedure with the capacity to generalize onto many environments, which is the reason for the name of Unified Neural Model; in other words, it was the conjunction of the properties described early into a single venue using 4 neuron activation types (Sigmoid, Threshold, Random and Identity), adaptation speeds, control assignations, neuromodulation and weights that can connect to previous neurons.

From the mathematical neuron model shown in Eq 1, w_i is the weight value at element i; x_i the input, both are multiplied and summed over all elements from i to n. The result is fed into function f to obtain the partial result of that neuron υ. In order to bring an adaptation speed, a factor is placed to modify the current result (υ).

In Eq 2, y_{t-1} represents the neuron's output at iteration $t-1$, also known as internal state. α is a constant value assigned to that neuron at creation with the purpose of dampening its excitation to reach a different scale neuron; y_t is the output of that element at execution t. As discussed previously, this enables the model to have different rates of change given certain circumstances. The second characteristic (inhibition/excitation) is achieved using control neurons which only activate when the stimulation exceeds the threshold as displayed in Eq 3.

$$\upsilon = f\left(\sum_{i=1}^{n} w_i x_i\right) \tag{1}$$

$$y_t = y_{t-1} + \frac{1}{\alpha}(\upsilon - y_{t-1}) \tag{2}$$

$$s = \sum_{i=0}^{n} (w_i cs_i) \quad \text{with} \quad c = \begin{cases} true, s >= tr \\ false, otherwise \end{cases} \tag{3}$$

Variable cs_i represents the stimulation a control neuron receives at position i; w_i are weights that connect to that neuron, to then be summed up and stored in s. The actual control value c is triggered in case s is more than or equal to a constant threshold tr, in any other case it will remain inactive.

Neuromodulation is present in the architecture as connections that affect other connections, which is changed as iterations continue and individuals are evolved. With that context, SUNA's connections have the ability to modify one another and change how results are propagated.

Lastly, feedback loops are added as part of mutations that allow new connections to be randomly selected from/to any neuron in the list, including calling the neuron itself. That covers a general overview of how SUNA manages to include its unified model.

III. Binary SUNA

Experiments demonstrated SUNA's ability in the solution of reinforcement learning problems with the utilization of continuous parameters as inputs/outputs; nonetheless the characteristics of the algorithm are easily transformed to binary operations that can extend its domain from floating to a discrete space. First, equations 1 and 2 are updated as follows:

$$\upsilon = popcount\left(\bigvee_{i=1}^{n} w_i \wedge x_i\right) \tag{4}$$

$$y_t = y_{t-1} \oplus \alpha \wedge (\upsilon \odot y_{t-1}) \tag{5}$$

In BiSUNA, variable υ becomes a bitset with a fixed range. The *popcount* function returns the number of bits "on" in a binary word, for example 100101 would return 3. All floating point multiplications are replaced with logic AND ($\wedge$); summation is reinstated with logic OR ($\vee$). y_{t-1} is also another discrete variable with the same properties as υ that are XNOR'ed ($\odot$). A substantial difference with respect to SUNA, is the utilization of α, a variable bitset that neurons with different firing rates use a *neuron inhibitor*, which hinders/excites the lower parts of the binary set. The last section applies logic XOR ($\oplus$) to the result of previous operations against y_{t-1}.

The utilization of these operations is not arbitrary, first with the intent to use functions that achieve a similar goal but in a discrete environment, with simple replacements like summations with OR, multiplications with AND. Also the presence of XOR serves the purpose of a logic summation for between previous values of the neuron state and new inputs presented, whereas XNOR function serves as a moderator of new values that must match the previous state given its properties as an equivalence gate. This type of applications have been shown as well in other BNN papers, for example [7].

It is possible to graphically visualize how a single binary neuron is connected to other elements in Fig 1. There are multiple elements represented, x_n are other neurons in the arrangement, each of which has their own state. w_n are the weights in every of the connections that link x_n to y_t; state values and weight values are bitset with values established at compile time. Also, y_t neuron has two important variables, y_{t-1} and α, the first represents previous state stored in the neuron and the second is the neuron firing rate, which has the purpose of creating neurons that work at different scales, performing activations given parameter α.

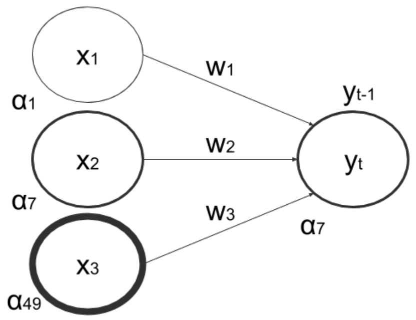


Fig. 1: Binary neuron representation with neuromodulation represented as the circle thickness.

Those changes allow BiSUNA's connections to completely switch how information is organized, from floating point to binary words, which in turn, replaces the need of Arithmetic Logic Units (ALU) with only binary operations. Memory reduction of connections is 16 bits, given that floats need 32 bits and unsigned short integer only 16; these subtle but important changes could mean the difference between running BiSUNA on a limited capabilities device (ex. ASIC/FPGA) or not. Even some researchers have shown that power consumption, while performing only inference, can be reduced 31x by simply switching ALU with logic functions [8].

In terms of the evolutionary path both algorithms take, there is no substantial change, BiSUNA's Spectrum Diversity, reproduction and mutation follow the same pattern as outlined in the original publication. Those procedures only require comparators, memory assignation and counters, which are basic instruments of any processing unit.

BiSUNA, as its parent algorithm does, takes advantage of a special structure named Novelty Map: a simple table that divides individuals given a *novelty* metric, which is calculated given the spectra of each individual, in other words, it counts the type neurons it has. One distinction with SUNA resides on the distance function used to discern between elements, where it would normally use a squared distance between individuals, $y = (x_1 - x_2)^2$; BiSUNA takes advantage of the hamming distance, $y = popcount(x_1 \oplus x_2)$, that keeps the utilization of only logic operations in place.

A. BiSUNA Impact

BiSUNA's main contribution stands around the adaptation of the current implementation of SUNA to work exclusively with binary values and operators. This paper serves as the first brick in a bridge being looked forward to construct, where BiSUNA is optimized to execute at the hardware level; the next natural step is its optimization onto FPGA devices, time at which it this algorithm will be expected to perform faster and consume less resources.

This paper is most impactful given that, to the authors' knowledge, there are no binary neural networks being trained using neuroevolutionary techniques as its main search algorithm, replacing the classical gradient descent used by most NN models. Where other researchers have focused on making marginal improvements over traditional discrete gradient descent implementations, this work takes on a completely newfangled approach.

One more important improvement shown is the utilization of binary neural networks to solve reinforcement learning (RL) problems, as shown in the next section. Most BNN publications reviewed apply their models to supervised learning, in areas like image recognition ([7]) or motion detention ([9]). The work presented in this paper can be generalized further in comparison to what has been analyzed by the authors, setting a precedence on the application of BNN to RL.

IV. Experiments

After a technical overview of the binary extensions in BiSUNA, this section will execute simulations provided by the OpenAI Gym [10] for selected environments: NChain, Duplicated Input & Copy. Each of them are going to be introduced with a description, objectives, range of values and results obtained with both, SUNA and BiSUNA. Also, this is the first time these algorithms are used with a standard reinforcement learning environment.

A. Settings

This publication uses the same configuration meta-parameters as the original SUNA implementation for both execution type (from [4], table III); all code is written in C++, interfacing with OpenAI Gym via the gRPC framework. Environments with discrete values, SUNA just clips numbers to the closest integer, whereas BiSUNA directly reads those as inputs; only cases that have a specific range of possible details (ex. states between $1 \ldots 5$) in the action range, the NN output is limited to what the environment considers acceptable numbers.

Another important consideration, every environment was repeated 10 epochs and the fastest execution time is the one being reported. Each test ran on a Nectar VM, similar to an AWS c5.large virtual machine (dual core, 4GB RAM), using Ubuntu 18.04 LTS. The code used along environment replication scripts are provided in repository https://github.com/rval735/BiSUNA, including as well results raw data, other RL tests and more related information.

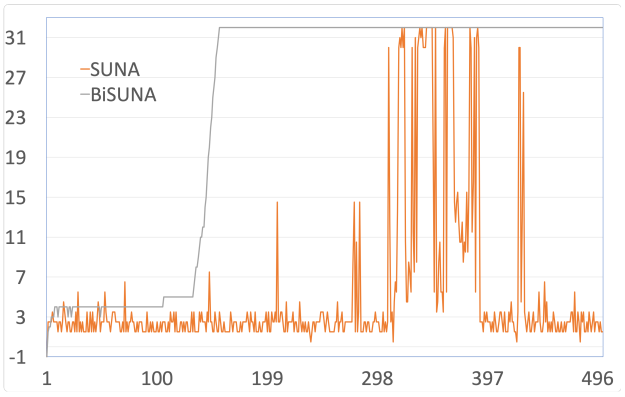
B. NChain

This environment [11] shows the possibility to move along a linear chain of states, with two feasible actions: stepping forward (no points), or backwards (small reward) but returns to the beginning. If an agent reaches the end of the chain, it will obtain a large bonification, which can be repeated by stepping "forward" to the chain's end. During tests performed, it was noticed how both executions exceeded over 4500 points, with an average reward of 4325.36 (BiSUNA) and 4348.60 (SUNA). In the OpenAI Gym solutions page for NChain [12], a reported result for this test was 1029.74 ± 9.19 over a 100 episode.

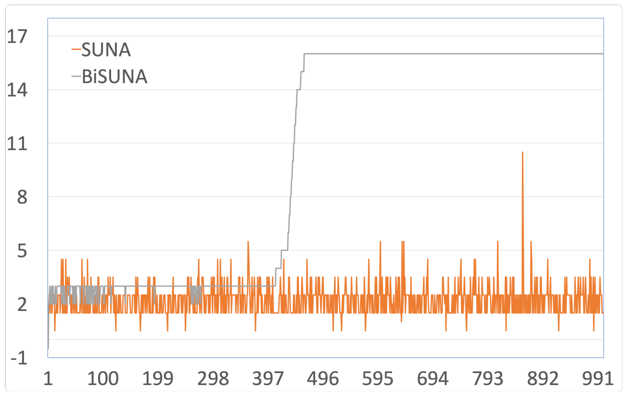
C. Copy

This simple task involves the replication of symbols from the input to the output tape. Although elementary, a trained model has to learn the correspondence between symbols in the observation and actions, which is translated into executing the *move right* action on the input tape.

With 500 iterations, BiSUNA implementation was able to reach a 32 score, SUNA only a maximum of 15; what is more interesting, only BiSUNA solved it in each trial (10 epochs), whereas SUNA did it only in 6. The longer it took to finish the test is explained as the evolutionary algorithm tried to increase the complexity of the population even though the result was satisfactory for this case, which made more complex organisms and took longer to evaluate them.



(a) Copy



(b) Duplicated Input

Fig. 2: Implementation performance in Copy and DuplicatedInput. X-axis represents the number of iterations and the Y-axis plots the best agent score obtained.

TABLE I: Results of executing Duplicated Input and Copy with SUNA and BiSUNA. (Better result is in **bold**).

	Duplicated Input		Copy	
Parameter	SUNA	BiSUNA	SUNA	BiSUNA
Iterations	1000	1000	500	500
Maximum Result	15	**16**	32	32
Solved out of 10	0	**9**	6	**10**
Best time (min:sec)	4:32	**3:13**	**5:01**	6:11
Worst time	**10:11**	12:55	**5:56**	7:44

D. *Duplicated Input*

As its name indicates, the input tape has been structured so that it repeats values, for example (x_1, x_1, x_1, x_2, x_2, x_2, ..., x_k, x_k, x_k) with the expected output as (x_1, x_2, ..., x_k). In this case, each input symbol is replicated 3 times, therefore, the best reward would be obtained by writing to the tape every third input symbol.

BiSUNA once more outperformed SUNA, reaching a solution of 16 points in 9 out of 10 trials; on the other hand, SUNA had a maximum result of 15 in only 2 executions. As shown in Fig 2b, this particular case, after 461 iterations, BiSUNA solved the problem and effectively learned how to deal with the environment, as it did with the Copy test.

V. Conclusion

This work provides insights on a different method the artificial intelligence community can take when approaching Binary Neural Networks using evolutionary principles, which was shown as an alternative to train them while avoiding completely the use of gradient descent. BiSUNA's spectra utilization for individual analysis in a population, the application of the novelty map and pillars of a unified learning model have shown their generalization capacity, now that the neuroevolution principles have been applied to fully discrete environments with binary operations. This is the first publication aimed to explore an extension to the original algorithm that transforms floating point operations to logic functions, confirms that agents' behavior were not affected and even improve for two cases, Copy and Duplicated Input tests, provided by the OpenAI Gym; corroborating how BiSUNA is capable of finding solutions to standardized problems.

References

[1] C.-Y. Lo, F. C. M. Lau, and C.-W. Sham, "Fixed-point implementation of convolutional neural networks for image classification," in *International Conference on Advanced Technologies for Communications (ATC)*, Oct 2018, pp. 105–109.

[2] M. Blott, T. B. Preusser, N. J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, and K. Vissers, "Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, pp. 16:1–16:23, Dec. 2018.

[3] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, 2019.

[4] J. M. Danilo Vargas, "Spectrum-diverse neuroevolution with unified neural models," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 8, pp. 1759–1773, Aug 2017.

[5] G. I. Sher, *Handbook of Neuroevolution Through Erlang*. New York, NY: Springer New York, 2013.

[6] K. O. Stanley, "Efficient evolution of neural networks through complexification," Ph.D. dissertation, Department of Computer Sciences, The University of Texas at Austin, 2004.

[7] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 525–542.

[8] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 243–254.

[9] J. Kung, D. Zhang, G. van der Wal, S. Chai, and S. Mukhopadhyay, "Efficient object detection using embedded binarized neural networks," *Journal of Signal Processing Systems*, vol. 90, no. 6, pp. 877–890, Jun 2018.

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv e-prints*, Jun. 2016.

[11] M. J. A. Strens, "A bayesian framework for reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 943–950.

[12] blole. (2016) Algorithm on nchain-v0. [Online]. Available: https://gym.openai.com/evaluations/eval_8KB72MuwRqqdbRG4UMs1w/