A Genetic Deep Learning Model for Electrophysiological Soft Robotics

Hari Mohan Pandey[1]and David Windridge[2]

[1]Faculty of Arts and Science. Edge Hill University, St Helens Rd, Ormskirk L39 4QP, U.K.
[2]Middlesex University, The Burroughs, London NW4 4BT, U.K.
[1]{Hari.pandey/Pandeyh}@edgehill.ac.uk
[2]d.Windridge@mdx.ac.uk

Abstract. Deep learning methods are modeled by means of multiple layers of predefined set of operations. These days, deep learning techniques utilizing unsupervised learning for training neural networks layers have shown effective results in various fields. Genetic algorithms, by contrast, are search and optimization algorithm that mimic evolutionary process. Previous scientific literatures reveal that genetic algorithms have been successfully implemented for training three-layer neural networks. In this paper, we propose a novel genetic approach to evolving deep learning networks. The performance of the proposed method is evaluated in the context of an electrophysiological soft robot like system, the results of which demonstrate that our proposed hybrid system is capable of effectively training a deep learning network.

Keywords: Deep learning, Evolutionary algorithm, Genetic algorithm, Metaheuristics, Neural networks.

1 Introduction

Deep learning networks are composed of multiple processing layers of predefined set of operations [6]. They have significantly improved the state-of-the-art across domains, including text mining, logical and symbolic reasoning, speech processing, pattern recognition, robotics and big data. Training deep learning networks is known to be hard [5]. Many standard learning algorithms randomly initialize the weights of the neural network (NN) and apply gradient descent using backpropagation. However, this gives poor solutions for networks with 3 or more hidden layers. Hence, fine-tuning of deep network parameters is an important aspect of learning and can be treated as a problem in which the fitness (or objective) function is considered as a criterion for optimization alongside parameters required to construct an efficient deep learning network architecture.

In recent years, meta-heuristics algorithms were implemented to handle the problem of Restricted Boltzmann Machine (RBM) model selection. Kuremoto et al. [7] used a Particle Swarm Optimization (PSO) algorithm to optimize the size of neural networks (number of input (visible) and hidden neurons) and the learning rate for 3-

layer deep network of RBMs. Liu et al. [8] suggested a Genetic Algorithm (GA) based system for optimization of RBM. Later on, Levy et al. [9] proposed a hybrid approach (GA + RBM) for unsupervised feature learning, which was used for automatic painting classification. In [9], GA was applied to evolve weights of the RBM. Rodrigues et al. [10] employed Cuckoo Search (CS) algorithm for the fine-tuning of parameters of a Deep Belief Network (DBN). In order to validate the effectiveness results were compared against other meta-heuristic algorithms such as Harmony Search (HS), Improved Harmony Search (IHS) and PSO. Rosa et al. [11] utilized a Firefly algorithm for learning the parameters of DBN. They also took other optimization algorithms (HS, IHS and PSO) for performance comparison. Papa et al. [12] proposed a HS based method for fine tuning the parameter of a DBN, obtaining more accurate results than comparable methodologies. Horng [13] showed the implementation of Artificial Bee Colony (ABC) algorithms for calibration of the parameters of DBNs. Experimental results showed the superiority of the ABC and Firefly algorithms over HS, HIS and PSO algorithms. Authors [21] [22] have shown the utility of fuzzy controller system for parameters tuning. But in this paper, we have not yet included fuzzy based system for quantifying the DNN parameters.

The aforementioned results reveal that meta-heuristic algorithms can be employed successfully for fine-tuning of parameters of deep learning networks. A comprehensive work on parameter calibration was presented in [12], though the authors suggest that better results can be achieved through Evolutionary Algorithms (EAs). GAs have been found very effective in several areas including grammar inference [14] [15] [16] [17] [20], function optimization [18], time tabling [19]. Considering this view, we propose a hybrid deep learning mechanism which utilizes the merits of GAs to enhance Gradient Decent in backpropagation learning. Therefore, the main contributions of this paper are threefold: (a) introducing a GA-based approach to deep auto-encoder learning, (b) enhancing the working of gradient decent in backpropagation and (c) filling the gap in research regarding application of meta-heuristic algorithms to deep learning model selection.

The remainder the paper is organized as follows: Section 2 presents a background on Deep Auto-Encoders. Section 3 presents our methodology for the application of Genetic Algorithms to Deep Learning Networks. Computational simulation and results are shown in Section 4. Finally, Section 5 states conclusions and future plans.

2 Training of a Deep Autoencoder

In this section, we set the context for the deep learning network used for creating the current system. An auto-encoder is an unsupervised neural network where input and output neurons are kept equal to follow a certain optimization goal. For output neuron i set to $y_i = x_i$, where x_i and y_i respectively represents the value of input and output neurons. A hidden layer is introduced between input and output layers following the convention: "*number of neuron in the hidden layer is less than those in the input and output layers*" which helps the neural network in learning a higher level representation by introducing an information bottleneck. Backpropagation methods are usually employed for training of an auto-encoder. Once training is over, the decoder layer can be discarded and, the values of the encoder layer fixed, so that it cannot be modified further. At this stage, the output of hidden layer is considered as input to a new auto-encoder. This new auto-encoder can be trained in a similar fashion. The whole structure encompasses a stack of layers referred to as a deep auto-encoders or deep belief network. The deep belief networks can be utilized for supervised and unsupervised classification utilizing the implicit higher-level representation.

3 Methodology Adapted for Training Deep Autoencoder

In this paper, we introduce a GA-based method for training a deep neural network (deep autoencoder in our case). GA is a metaheuristic search and optimization algorithm proposed by Holland [2] that has been successfully implemented for training of neural networks [3]. More specifically, GAs have been employed as a substitute for the backpropagation methods. By contrast, we here propose to use GAs in conjunction with backpropagation to enhance the overall performance of deep neural networks.

We thus implement, as a proof-of-concept, a simple GA based deep learning network for the electrophysiological soft robot like system as described in [1]. During the training phase of the auto-encoder, we store multiple sets of weights (W) for each layer and these weights are used to create a population for the GA, where each chromosome represents one set of weights. We determine the fitness of each chromosome using equation (1).

$$F = \sum_{t=1}^{T} \left(f_{initial} + \left(f_{diff} - \left(1 - P_m \big/ P_M^{'} \right) \right) \right) \tag{1}$$

Where, $f_{initial}$: the initial fitness value (=0, in the beginning of the execution), f_{diff} : difference in the position of organism (green dot) after eating food (blue dot) from initialization and the end of T actuation cycles/time step (in our case T = 130), P_m : penalty matrix and $P_M^{'}$: maximum penalty matrix.

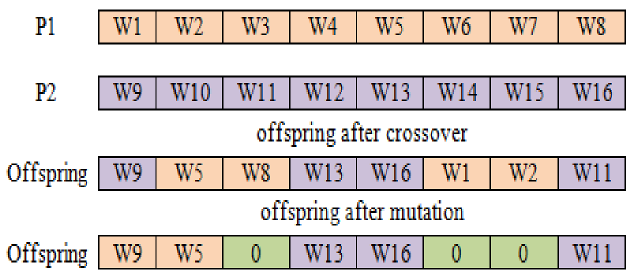


Fig. 1. A simple example of crossover and mutation operations used during simulation.

The fitness value of all the chromosomes is determined and then sorted in descending order of their value. Next, we utilize backpropagation to update the weights of the high ranking chromosomes and discard the lower ranked chromosomes from the pool by removing them from the population. We apply a uniform selection strategy to selection the chromosomes, so that all chromosomes have equal probability of selection for the next generation regardless of the fitness values of the chromosome. In our system, we use the fitness value to determine which chromosomes are to be removed from the population.

In order to perform the crossover operation, a couple of parent populations are selected. Then, by selecting weights randomly from each parent the new offspring is created. On the other hand, the mutation operation is performed by replacing a selection of weights with zero values in the offspring. We demonstrate the crossover and mutation operations via the simple example depicted in Figure 1.

Crossover and mutation operations are powerful mechanisms for introducing diversity in the population; - David and Greental [4] indicate that gradient descent methods such as backpropagation are susceptible to trapping in local minima. By adding the merits (in particular recombination operations) of a GA, we can alleviate propensity for the system to get stuck at local optima.

In the preceding we set a maximum number of generations as the termination criteria. At the end of this process, the best value of the chromosomes are selected and shared among all the chromosomes of the new layer of the auto-encoder. Hence, the new layer currently being trained only contains the best value of the chromosomes, helping to improve the performance of the overall system.

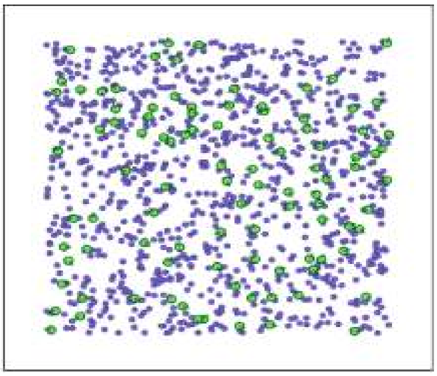
4 Computational Simulation and Results

All the experiments are conducted on Anaconda Spider (Tensorflow) with python 3.5. For our experiments we used a simple electrophysiological robot like system as presented in [1]. The problem setup in our case consists of a DNN with a stack of 4-layers (50 neurons at 1st layer, whereas other three layers consist of 40, 30 and 20 neurons. We train each layer separately: we started training with 40 - 30 layers, then utilize the 30 output neurons as inputs to the 30 - 20 layers.

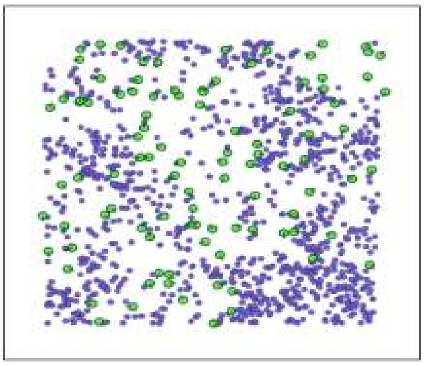
We used a simple GA (SGA) with the following configuration: population size = 100, chromosome size = 15, crossover rate = 0.6, mutation rate = 0.4 and termination condition = maximum number of generations = 100.

We executed the GA based deep learning network 30 times (independent runs with identical initial conditions) and collated the results. The objective function is the cost function in our experimental setup; the cost function is called once every generation (after a cycle of 130 time steps a generation is said to be complete). The fitness function value depends upon both: the collisions between the organism (green dot) and food particles (blue dot) as shown in Figure 2.

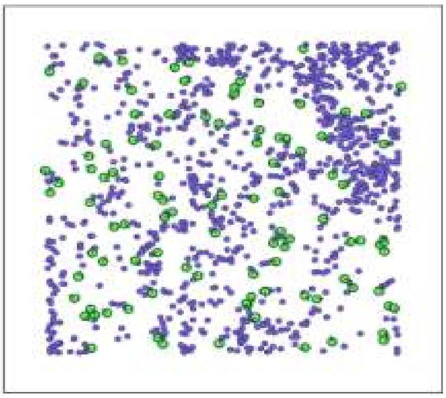
When an organism coincides with a food particle, the fitness function value of that organism is updated and the food particle reappears at a new random location. In the second case, when an organism collides with any other organism, then we penalize the system. In each iteration, the GA provides training to the network in layered manner, identifies the closet food particle, determines the direction of the food particle and based on the response updates the position and velocity of the organism. We record the best, average and worst fitness value for each generation (graphically shown Figure 3).



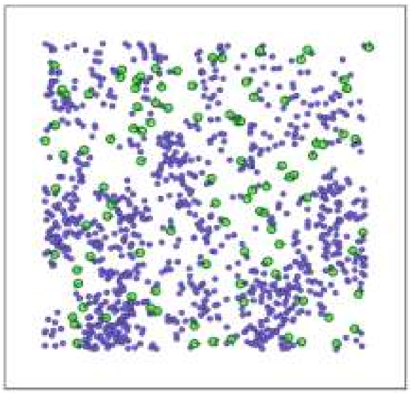
Gen: 0, Time_Step:0



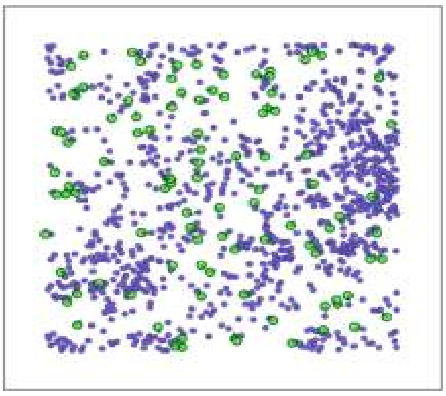
Gen 10, Time_Step:23



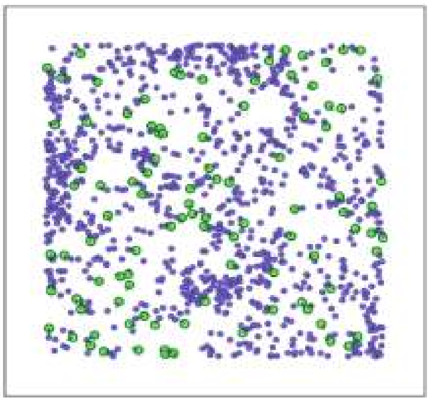
Gen:15 Time_Step:100



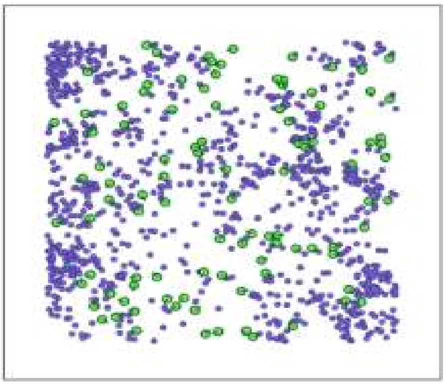
Gen:17, Time_Step:79



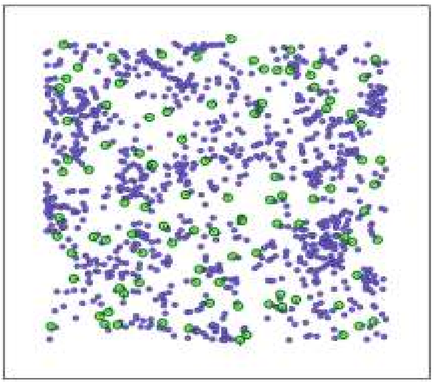
Gen:20, Time_Step: 79



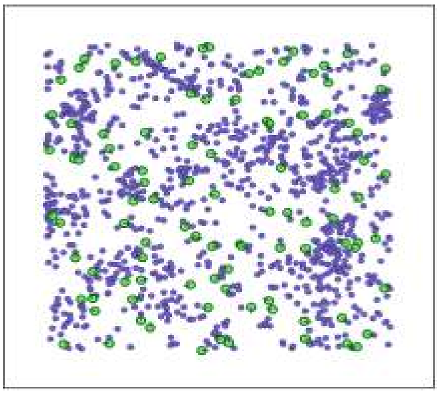
Gen:26, Time_Step:117



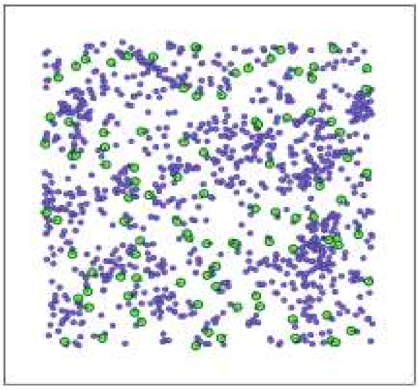
Gen: 38, Time_Step: 85



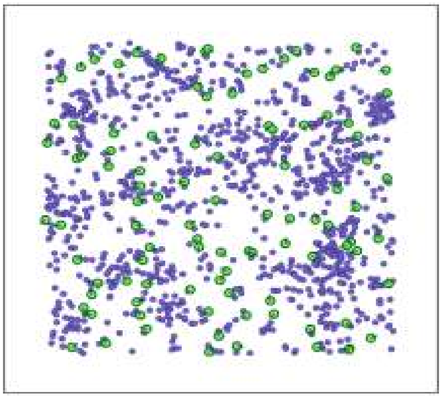
Gen: 65, Time_Step:114



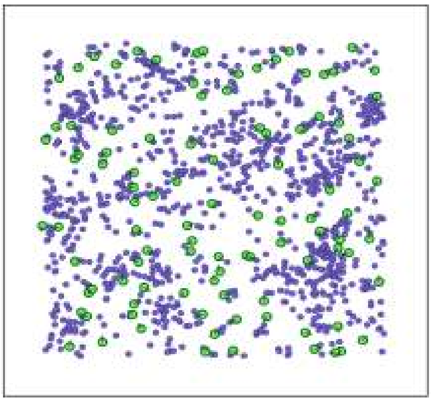
Gen:70, Time_Step:121



Gen:92, Time_Step:50



Gen:95, Time_Step:75



Gen:97, Time_Step:124

Fig.2. Simulation results of GA based deep learning network in different generations

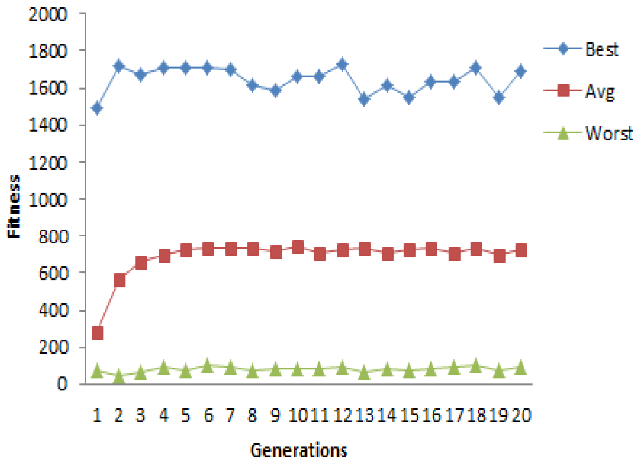


Fig. 3. Average fitness value Vs generation (first 20 iterations) chart for the best, average and worst fitness values recorded for 30 independent runs with total time step 130.

5 Concluding Remarks and Future Plans

This paper has presented a GA-based approach to applying evolution to a deep learning network problem. Initial results suggest that GAs can be utilized for the training of deep learning networks not just an alternative to backpropagation methods as in previous work, but can rather work in conjunction with backpropagation effectively solve the deep learning optimization problem. Our experiments utilizes an auto-encoder, we believe that the same method can be generalized to other forms of deep learning network architectures.

In regards to future work, we aim to compare the performance of GA-based training methods with other meta-heuristic approaches and gradient descent methods, and to extend the method for de-noising auto-encoders and implement a similar system for training deep Boltzmann machines. In addition, we aim to develop a GA based deep learning system for autonomous driving.

Acknowledgment. Authors would like to acknowledge financial support from the Horizon 2020 European Research project DREAM4CARS (#731593).

References

1. Cheney N.,MacCurdy R., Clune J. and LipsonH. "*Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding*." Proceedings of the 15th annual conference on Genetic and evolutionary computation, pp. 167-174, ACM, 2013.
2. Holland J. H. "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence". MIT press, 1992.

3. Schaffer J.D., Whitley D. and Eshelman L. J. "*Combinations of genetic algorithms and neural networks: A survey of the state of the art*." Combinations of Genetic Algorithms and Neural Networks, 1992, COGANN-92. International Workshop on. IEEE, 1992.
4. David O.E. and Greental I."*Genetic algorithms for evolving deep neural networks*." In Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation (pp. 1451-1452). ACM, 2014.
5. Larochelle H., Bengio Y., Louradour J. and Lamblin P."*Exploring strategies for training deep neural networks*". Journal of machine learning research, 10(Jan), pp. 1-40, 2009.
6. Pandey H.M. and Windridge D. "*A comprehensive classification of deep learning libraries*." In: International Congress on Information and Communication Technology, Feb 2018, London, UK.
7. Kuremoto, T., Kimura, S., Kobayashi, K., and Obayashi, M. "*Time series forecasting using restricted boltzmann machine.*" In International Conference on Intelligent Computing (pp. 17-22), 2012, Springer, Berlin, Heidelberg.
8. Liu, K., Zhang, L. M., and Sun, Y. W. "*Deep Boltzmann machines aided design based on genetic algorithms.*" In Applied Mechanics and Materials (Vol. 568, pp. 848-851), 2014, Trans Tech Publications.
9. Levy, E., David, O. E., & Netanyahu, N. S. "*Genetic algorithms and deep learning for automatic painter classification*." In proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 1143-1150, 2014, ACM.
10. Rodrigues, D., Yang, X. S., and Papa, J. P. "*Fine-tuning deep belief networks using cuckoo search.*" In Bio-Inspired Computation and Applications in Image Processing, pp. 47-59, 2017.
11. Rosa, G., Papa, J., Costa, K., Passos, L., Pereira, C., and Yang, X. S. "*Learning parameters in deep belief networks through firefly algorithm.*" In IAPR Workshop on Artificial Neural Networks in Pattern Recognition, pp. 138-149, 2016. Springer, Cham.
12. Papa, J. P., Scheirer, W., & Cox, D. D. "*Fine-tuning deep belief networks using harmony search.*" Applied Soft Computing, 46 (2016), pp. 875-885.
13. Horng, M. H."*Fine-Tuning Parameters of Deep Belief Networks Using Artificial Bee Colony Algorithm.*" DEStech Transactions on Computer Science and Engineering, (aita), 2017.
14. Pandey, H.M., Chaudhary A, Mehrotra D. and Kemdall G. "*Maintaining regularity and generalization in data using the minimum description length principle and genetic algorithm: Case of grammatical inference",* Swarm and Evolutionary Computation, Vol. 31, pp. 11-23, Dec. 2016.
15. Pandey, Hari Mohan. "Natural language grammar induction using genetic and parallel genetic algorithms and load balancing." *Int. J. Comput. Sci. Technol* 1.1 (2012): 28.
16. Pandey, Hari Mohan, Ankit Chaudhary, and Deepti Mehrotra. "Grammar induction using bit masking oriented genetic algorithm and comparative analysis." *Applied Soft Computing* 38 (2016): 453-468.
17. Choubey, Nitin Surajkishor, Hari Mohan Pandey, and M. U. Kharat. "Developing genetic algorithm library using Java for CFG induction." *International Journal of Advancement in Technology* (2011).
18. Pandey, Hari Mohan, Manjul Rajput, and Varun Mishra. "Performance comparison of pattern search, simulated annealing, genetic algorithm and jaya algorithm." *Data Engineering and Intelligent Computing*. Springer, Singapore, 2018. 377-384.
19. Pandey, Hari Mohan. "Solving lecture time tabling problem using GA." Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference. IEEE, 2016.

20. Pandey, Hari Mohan, Ankit Chaudhary, and Deepti Mehrotra. "Bit mask-oriented genetic algorithm for grammatical inference and premature convergence." *International Journal of Bio-Inspired Computation* 12.1 (2018): 54-69.
21. Cervantes, Leticia, et al. "Fuzzy Dynamic Adaptation of Gap Generation and Mutation in Genetic Optimization of Type 2 Fuzzy Controllers." *Advances in Operations Research* 2018 (2018).
22. Lagunes, Marylu L., et al. "Parameter Optimization for Membership Functions of Type-2 Fuzzy Controllers for Autonomous Mobile Robots Using the Firefly Algorithm." *North American Fuzzy Information Processing Society Annual Conference*. Springer, Cham, 2018.