Hyper-parameter Optimisation by Restrained Stochastic Hill Climbing

Rhys Stubbs[1], Kevin Wilson[2] and Shahin Rostami[3]

Faculty of Science & Technology, Bournemouth University, Bournemouth, BH12 5BB, United Kingdom
[1]i7433085@bournemouth.ac.uk
[2]kwilson@bournemouth.ac.uk
[3]srostami@bournemouth.ac.uk
WWW home page: https://research.bournemouth.ac.uk/project/ciri/

Abstract. Machine learning practitioners often refer to hyper-parameter optimisation (HPO) as an art form and a skill that requires intuition and experience; Neuroevolution (NE) typically employs a combination of manual and evolutionary approaches for HPO. This paper explores the integration of a stochastic hill climbing approach for HPO within a NE algorithm. We empirically show that HPO by restrained stochastic hill climbing (HORSHC) is more effective than manual and pure evolutionary HPO. Empirical evidence is derived from a comparison of: (1) a NE algorithm that solely optimises hyper-parameters through evolution and (2) a number of derived algorithms with random search optimisation integration for optimising the hyper-parameters of a Neural Network. Through statistical analysis of the experimental results it has been revealed that random initialisation of hyper-parameters does not significantly affect the final performance of the Neural Networks evolved. However, HORSHC, a novel optimisation approach proposed in this paper has been proven to significantly out-perform the NE control algorithm. HORSHC presents itself as a solution that is computationally comparable in terms of both time and complexity as well as outperforming the control algorithm.

Keywords: hyper-parameter optimisation, global optimisation, neuroevolution, artificial neural networks, random search, stochastic hill climbing

1 Introduction

Neuroevolution (NE), a sub-field of Artificial Intelligence (AI), originated in the 1980s as shown by the work of Montana & Davis [21]; NE involves evolving and adapting Artificial Neural Networks (ANN) by employing Evolutionary Algorithms (EA) such as Genetic Algorithms (GA), a sub-field of Evolutionary Computation (EC), to optimise ANNs as well as solve complex Reinforcement Learning (RL) tasks [33]. As proposed by Yao [42], the Neural Network(s) produced by NE can be described as an Evolutionary Artificial Neural Network (EANN). Traditionally, non-evolving ANNs often employ Gradient-based, Back

Propagation algorithms [22, 43, 35] to learn and optimise performance by adjusting the ANNs hyper-parameters and weights. However, in traditional NE approaches the topology is chosen prior to the employment of the learning algorithm [33] and will not evolve during training. The ANN topology is an important factor to consider when planning to employ an ANN; this is due to the fact that the topology chosen plays a fundamental role in its functionality and performance [11], and therefore it's ability to produce a meaningful output.

Prominent empirical studies [33, 26] have shown that EAs can be employed to dynamically evolve the weights as well as the overall topology of ANNs. The GAs used are often categorised depending on the type of evolution achieved. Algorithms that solely evolve the weights are known as conventional Neuroevolution algorithms [14, 33, 13, 24]; whereas, algorithms that evolve the weights and topology are known as Topology and Weight Evolving Artificial Neural Network algorithms (TWEAAN) [33]. Many popular and modern NE algorithms have adopted the TWEANN approach. However, the topics of simplification and complexification are crucial factors that algorithms should consider. The goal of a NE algorithm is to find the best performing solution for a given problem while reducing unnecessary complexity of the topology.

It should be noted that NE is not the only approach used for hyper-parameter optimisation (HPO). It is in fact one of five popular HPO approaches that can be applied to ANN evolution and learning; moreover, NE can be used for optimisation of model parameters and hyper-parameters. Alternative approaches include grid search, random search, Bayesian optimisation and Gradient-based optimisation; the latter of these is most commonly employed and has become the primary approach for Neural Network (NN) parameter optimisation by applying a Gradient Descent and Back-propagation algorithm. The research will focus on the most popular, prominent and currently available NE algorithm implementations in order to evaluate and compare them against alternative hyper-parameter optimisation approaches.

2 Background study

2.1 Neuroevolution

Neuroevolution is the process of evolving ANNs through evolutionary algorithms [17] and is inspired by the evolution of biological brains [32, 33]. Various research papers have proven that a GA can be applied and used to find ANNs that consistently display improved learning speeds in comparison to a typical Feed Forward ANN [40]. NE establishes a fundamentally different approach to learning tasks in contrast to alternative techniques such as Back Propagation with Gradient Descent. NE is a phylogenetic learning approach that focuses on evolving a whole population of solutions, whereas conventional ontogenetic approaches focus on training a single solution [15, 36]. Moreover, GAs have been proven to successfully perform tasks such as: connection weight training, architecture design, learning rule adaptation, input feature selection, connection weight initialisation and rule extraction [42].

Neuroevolution can have a number of evolutionary taxonomies, including weights, topology, and learning rules [42]. Moreover, there have been numerous discussions on the methods that evolve topologies; specifically, whether the algorithm should complexify or simplify a NN topology, also known as constructive and destructive algorithms [30, 42, 44, 1]. This would entail adding or removing connections and/or neurons from the NN. In recent years, there has been an increasing amount of literature on NE, specifically, on how to improve the performance of the NNs evolved against increasingly difficult multi-objective problems such as Atari game playing, and complex real-world control/automation problems. Differentiable plasticity, a concept proposed by [20] seeks to address the problem of "learning to learn". It aims to decouple training and learning, as currently agents must be retrained for a different task than the one initially learnt. Despite the initial usage of GAs for NE, recent research has moved towards using Evolution Strategies (ES) for NN training [23], a class of black box optimisation algorithms whose performance, when applied to complex RL problems, rivals that of algorithms such as Q-Learning (DQN) and Policy Gradients (A3C), and which are also highly parallelisable [5, 29]. A primary differentiating factor of ES is that solutions are encoded using real numbers [25]. Also, ES uses mutation in a fundamentally different way to GAs and it is achieved through self-adaptation or Covariance Matrix Adaptation (CMA) [38]. However, despite these alternative approaches, researchers at Uber AI Labs [35] have developed Deep GA, a GA for *Deep Neuroevolution*; moreover, their findings show that a simple GA is competitive at completing modern problems that were originally thought of as extremely challenging [38]

2.2 Hyper-parameter Optimisation

Hyper-parameter optimisation is a crucial step of applying a ML algorithm and finding optimal hyper-parameter values manually is often a time consuming and tedious task [10] and it is often referred to as an art form [12]. ML algorithms are rarely hyper-parameter free, and indeed, they are often considered nuisances, however, the process of automatically determining optimal values can be seen as a process of optimisation [31]. HPO is the process of optimising a loss function over a graph-structured configuration space [4]; the aim is to maximise or minimise a given function [39]. Determining appropriate values for the hyper-parameters is fundamental in finding an optimal solution, however, it is a frustratingly difficult task [18, 10, 8, 9] and the performance of NNs crucially depend on the hyper-parameters used [9, 6]. For example, the internal structure is a key factor in determining the efficiency of the NN [27]. Moreover, a major challenge when designing and building a NN is determining the optimal hyper-parameters for the network given the data for the problem at hand [7]. There are a number of popular and widely employed HPO approaches; each has been developed with a different aim and generally improves upon the previously developed approach. However, grid search and manual search are presently the most widely used strategies for HPO [3].

Grid and random search Grid search, also known as a parameter sweep, is the traditional approach used for finding the optimal set of hyper-parameters for a given function; the approach is an exhaustive search that tries all possible combinations to find the optimal value(s) [3]. Due to the fact that all possible values are explored, the approach can guarantee reliable optimisation in low dimensional spaces [3]. However, not necessarily efficiently because it exhaustively tries all possible combinations and suffers from the curse of dimensionality as the number of values grows exponentially with the number of hyper-parameters [2, 3].

The random search approach was developed and aimed to reduce the cost of computation and find an alternative to grid search. In principle, the random search approach is very similar to that of grid search, however, instead of all possible value combinations tested, the algorithm stochastically tries values within the search space. A number of empirical studies have shown random search to outperform and be computationally more efficient than grid search at finding an optimal combination of hyper-parameters [3].

3 Experimental Design

The design of this experiment draws on existing research, most notably: [3, 4, 19, 45]. A considerable amount of literature has been published on grid search, it is widely accepted as a computationally expensive approach. However, in problems where the intrinsic dimensionality is low, it may be an appropriate approach. NE and EAs are stochastic algorithms that often employ manual and evolutionary HPO, however, there are some unanswered questions about the validity of this approach and its ability find optimal hyper-parameters. Therefore, this research aims to integrate and utilise a random search approach for HPO. The question that then naturally arises is whether this approach is reliable, as randomising the optimisation process may arbitrarily produce optimal solutions, which may represent a long-winded process as the computation time required to find an optimal solution increases at each step an optimal solution is not found.

This experiment involves taking a standard NE algorithm and producing five distinct modified versions, as listed in Table 1, and applying them to an Unsupervised Learning problem that involves the NNs learning to target seek. The modifications included in the experiment can be categorised as either an initialisation modification (alias prefix IN) or a run-time optimisation modification (alias prefix RT). The NE algorithm used is *Neataptic* [37], a JavaScript implementation based on NEAT [33]. The random search optimisation modifications will be integrated into multiple distinct copies of the original Neataptic algorithm.

In order to carry out the experiment, each algorithm must be configured as shown in Table 2. Each algorithm will undergo a total of 10000 function evaluations with a sample size of 30 executions for each algorithm. The population size has been determined using a method developed in [34], specifically, the population size $P = N \times 10$ where N is the number of objectives (i.e. 1 in our experiment).

Algorithm 1 is the pseudo-code for the experiment and outlines the fundamental flow of execution for the experiment and the algorithms.

Table 1: Included algorithm descriptions and alias definitions

Algorithm	**Alias**
Neataptic (no modifications)	Vanilla
Neataptic with initial neuron activation function modification	INAFM
Neataptic with initial neuron bias modification	INBM
Neataptic with initial network topology modification	INTM
Neataptic with run-time activation function optimisation	RTAFO
Neataptic with run-time network topology optimisation	RTNTO

Table 2: Parameter configurations for the GA

Parameter	**Value**		
Mutation rate	0.3		
Elitism	0.1		
Selection method	Tournament		
Crossover method	Uniform		
Mutation methods	Add neuron	Remove neuron	Add self-connection
	Add connection	Remove connection	Remove self-connection
	Modify weight	Modify bias	Remove recurrent connection
	Add gate	Remove gate	Add recurrent connection
Population size	100		
Generation count	100		

4 Hyper-parameter Optimisation by Restrained Stochastic Hill Climbing

Stochastic hill climbing chooses it's next value at random from the available search-space [28]. The approach introduced in this section, named hyper-parameter optimisation by restrained stochastic hill climbing (HORSHC), proposes an approach to HPO that rivals the manual and evolutionary approach found in the NE algorithm used in our experiment.

The HORSHC process outlined in Algorithm 2 begins by defining a limit, this is the *restrainment* applied to the algorithm. This can be considered another hyper-parameter for optimisation. The algorithm then goes on to stochastically increase or decrease a network's size, doing so until either the limit is reached or performance no longer improves.

Algorithm 1 Hyper-parameter optimisation experiment execution cycle.

```
1: $g \leftarrow 0$
2: $p \leftarrow [size]$                          ▷ in our experiment size = 100
3: $p$ = InitialisePopulation($size$)
4: while $g \leq max$ do                          ▷ in our experiment max = 100
5:     for $network \leftarrow 0$ to $size$ do
6:         ApplyToLossFunction($network$)
7:         $f$ = evaluateFitness($network$)
8:         $start$ = now
9:         optimise($network$)
10:        $finish$ = now
11:        storeTimes($finish$ - $start$)
12:        storeFitness($f$)
13:    end for
14:    $g \leftarrow g + 1$
15: end while
```

Algorithm 2 HORSHC execution cycle

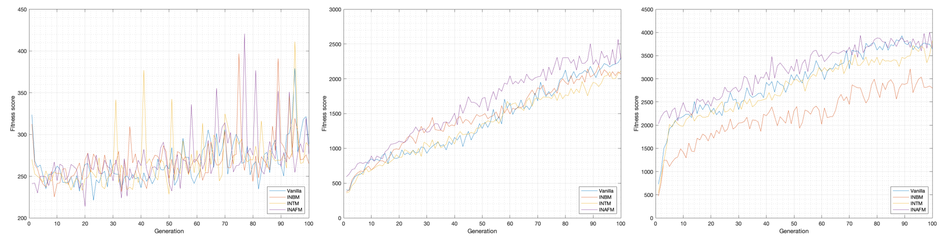
```
1: $limit \leftarrow 3$
2: $h \leftarrow$network.score
3: $i \leftarrow 0$
4: do
5:     $h \leftarrow$network.score
6:     if $p >= 0.5$ then                      ▷ p is assigned a random number 0-1
7:         $network$ = simpiflyNetwork
8:     else
9:         $network$ = complexifyNetwork
10:    end if
11:    ApplyToLossFunction($network$)
12:    $network.score \leftarrow$ FitnessFunction
13:    $i \leftarrow i + 1$
14: while $network.score > h$ and $i$ != $limit$
```

5 Numerical Results

Figure 1 depicts the overall performance of each of the initialisation modifications (INAFM, INBM, INTM), compared with the unmodified vanilla version; specifically, the average fitness scores as well as the average of the worst and best performing solutions throughout the 100 generations. Figure 2 does the same for the run-time activation function modification (RTAFO), and Figure 3 shows the results for the run-time network topology modification (RTNTO).

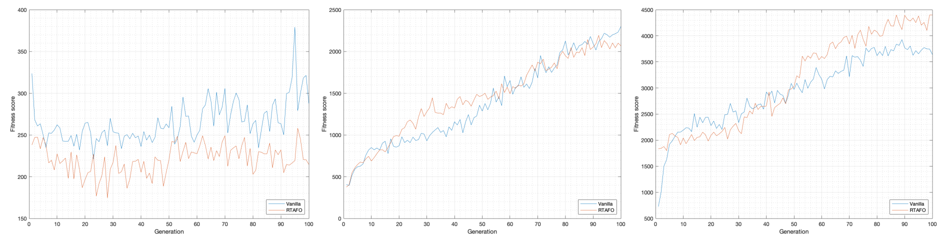
In order to complement the other findings, the Wilcoxon signed-rank test [41] has been performed using a significant value of 0.05. The test will determine if there is a significant statistical difference between the results obtained from the vanilla algorithm and the other algorithms, hypothesis values with a = indicates equal performance, - indicates inferior performance and a + indicates superior performance.

Table 3 shows the average of the worst, mean, and best performing solutions for all algorithm variations across 30 independent samples of 10000 function evaluations in comparison to the vanilla. Despite the results illustrated in Figure 1 that depicted potential performance increases, none of the results for the initialisation modifications were significantly different to the results produced by



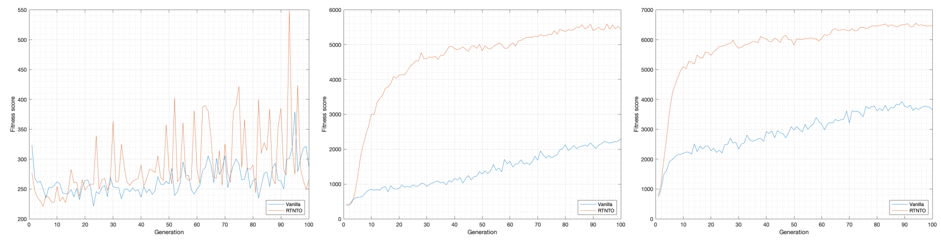
(a) Lows (b) Averages (c) Highs

Fig. 1: Experimental results for the initialisation modifications



(a) Lows (b) Averages (c) Highs

Fig. 2: Experimental results for the activation function modifications



(a) Lows (b) Averages (c) Highs

Fig. 3: Experimental results for the topology optimisation modifications

the vanilla algorithm. However, results for the run-time modifications show that both RTAFO and RTNTO significantly outperform the vanilla algorithm.

Table 4 shows the Wilcoxon test results for the total times of the full experiment execution, RTNTO represents a significantly inferior algorithm compared to the vanilla algorithm in terms of overall time taken to execute. Whereas, RTAFO had equal performance of that shown by the vanilla algorithm.

Table 3: Wilcoxon test results for the modified algorithms.

	Worst		**Mean**		**Best**	
Algorithm	p-value	Hypothesis	p-values	Hypothesis	p-value	Hypothesis
INAFM	0.65176	0 (=)	0.65544	0 (=)	0.51585	0 (=)
INBM	0.26876	0 (=)	0.77657	0 (=)	0.19093	0 (=)
INTM	0.85942	0 (=)	0.95153	0 (=)	0.37074	0 (=)
RTAFO	0.00078147	1 (-)	0.33874	0 (=)	0.00078147	1 (+)
RTNTO	0.56638	0 (=)	1.8626e-08	1 (+)	9.3132e-09	1 (+)

Table 4: Wilcoxon test results for the total time of RTAFO and RTNTO in comparison to the vanilla algorithm.

	Total time		
Algorithm	**Mean time (mins)**	**p-value**	**Hypothesis**
RTAFO	9.1542	0.55611	0 (=)
RTNTO	10.9112	5.7183e-07	1 (-)

6 Conclusion

The experiment carried out during this research examined 3 hyper-parameter initialisation approaches and 2 run-time HPO approaches. The 3 initialisation approaches aimed to better improve the initialisation performance of solutions; whereas, the run-time optimisation approaches aimed to optimise existing solutions in the population. The novel run-time optimisation approaches employed have performed remarkably well in comparison to the vanilla algorithm. It should be remarked that RTNTO has been shown to significantly outperform all other algorithms and was able to do so within a third of the total function evaluations allocated.

With that said, the initialisation algorithms do present some interesting results. Despite neither INAFM, INBM, or INTM displaying superior performance to that of the vanilla algorithm, all 3 were able to produce results that were of equal performance. However, the results show that INAFM has consistently improved the initial performance of the best performing solutions; therefore, it suggests that the selection and crossover mechanisms used by the algorithm are not subsequently producing better performing solutions. Instead, INAFM is accelerating the initial optimisation process that the other approaches were unable to achieve.

An important question to answer is whether performance can be improved by different permutations of the initialisation approaches; however, this could also result in a further acceleration of the optimisation and not result in higher performing solutions at the end of the process. The method of complexification employed by many modern algorithms ensures that the initial size of the NNs are small, however, INTM disregards this and allows solutions to have potentially larger initial topologies. Depending on the requirements, it may not be advantageous to produce high performing solutions whose topologies are potentially over-complex in comparison to their counterparts.

Moreover, the results of RTNTO have revealed an unanticipated superior performance in comparison to the vanilla algorithm and all other algorithms used in this research. As previously mentioned, RTNTO was able to outperform the other algorithms within a third of the allocated function evaluations. As with INAFM, the RTNTO algorithm was able to not only accelerate the optimisation of solutions, but also increase overall solution performance by 56 percent. However, despite the significant performance improvement, as with all of the algorithms, RTNTO was unable to increase performance of the whole population as average low scores were significantly lower and did not show signs of improvement during execution.

Similarly, due to the increase in computation time required, RTNTO takes significantly longer to complete than both the vanilla and RTAFO algorithms. Generally speaking, a single, optimal solution is what a researcher/practitioner requires and this inability to remove weak performing solutions and/or execute in the fastest time may not pose a problem. Interestingly, RTNTO was a by-product of another algorithm and was similar in its approach but was unrestricted in terms of how may hidden layers/neurons could be added during a single function evaluation. Despite a lack of statistical evidence, this former algorithm displayed similar performance to that of RTNTO.

Contrasting RTAFO and RTNTO, despite the initial promising performance of INAFM during the initialisation experiments, RTAFO was unable to achieve similar results to that of RTNTO. Comparing RTAFO to the vanilla algorithm, it was able to significantly outperform in terms of average highest fitness for candidate solutions. However, it has become apparent throughout the research that all of the algorithms are unable to improve the population as a whole and RTAFO is no exception to this; in fact, it is the only algorithm that was significantly outperformed by the vanilla algorithm on this basis. Due to this consistency, it opens a question to whether the surrounding components are to blame or whether additional logic is required to eliminate the issue and allow the whole population to improve.

7 Future Work

The results uncovered during this research on the integration of alternative HPO approaches within a Neuroevolution algorithm have revealed 2 probable hypotheses: (1) random initialisation of hyper-parameters has little significance on the final performance of solutions; (2) HORSHC performed significantly better than pure evolutionary and/or manual search strategies for finding high performing solutions. HORSHC is a competitive HPO algorithm and it is proposed that it should be employed for optimising NNs solving single objective problems such as the target seeking problem used in our experiment. However, several questions remain unanswered: (1) how well do the approaches demonstrated in this research perform against a problem with multi/many objectives?; (2) how well do the approaches demonstrated in this research perform on a different set of problems?; (3) Are there permutations of initialisation methods that provide better optimisation results? A natural progression of this work would be to explore the application of the approaches proposed in this research according to questions 1 and 2 as well as the combination of approaches to see if they reveal further performance increases. Furthermore, a further study could assess the effect of activation function optimisation and its significance; [16] performed a similar experiment and introduced HA-NEAT that is analogous to RNAFO. Furthermore, as RTNTO was restricted to 3 modifications, it could be argued that this is another hyper-parameter to tune and increases/decreases may yield better results. Finally, as ES lead the latest research and have fewer hyper-parameters [29], a further study of traditional HPO with ES would be a complementary contribution.

References

1. Angeline, P.J., Saunders, G.M., Pollack, J.B.: An evolutionary algorithm that constructs recurrent neural networks. IEEE transactions on Neural Networks **5**(1), 54–65 (1994)
2. Bellman, R.E.: Adaptive control processes: a guided tour, vol. 2045. Princeton university press (2015)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**(Feb), 281–305 (2012)
4. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in neural information processing systems, pp. 2546–2554 (2011)
5. Conti, E., Madhavan, V., Such, F.P., Lehman, J., Stanley, K., Clune, J.: Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In: Advances in Neural Information Processing Systems, pp. 5027–5038 (2018)
6. Dernoncourt, F., Lee, J.Y.: Optimizing neural network hyperparameters with gaussian processes for dialog act classification. In: 2016 IEEE Spoken Language Technology Workshop (SLT), pp. 406–413. IEEE (2016)
7. Diaz, G.I., Fokoue-Nkoutche, A., Nannicini, G., Samulowitz, H.: An effective algorithm for hyperparameter optimization of neural networks. IBM Journal of Research and Development **61**(4/5), 9–1 (2017)

8. Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., Leyton-Brown, K.: Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In: NIPS workshop on Bayesian Optimization in Theory and Practice, vol. 10, p. 3 (2013)
9. Eggensperger, K., Hutter, F., Hoos, H., Leyton-Brown, K.: Efficient benchmarking of hyperparameter optimizers via surrogates. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
10. Feurer, M., Springenberg, J.T., Hutter, F.: Initializing bayesian hyperparameter optimization via meta-learning. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
11. Fiesler, E.: Neural network topologies (1996)
12. G, Y.: The 7 steps of machine learning (2017). URL https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e
13. Gomez, F., Schmidhuber, J., Miikkulainen, R.: Efficient non-linear control through neuroevolution. In: European Conference on Machine Learning, pp. 654–662. Springer (2006)
14. Gomez, F., Schmidhuber, J., Miikkulainen, R.: Accelerated neural evolution through cooperatively coevolved synapses. Journal of Machine Learning Research **9**(May), 937–965 (2008)
15. Gomez, F.J.: Robust non-linear control through neuroevolution. Ph.D. thesis (2003)
16. Hagg, A., Mensing, M., Asteroth, A.: Evolving parsimonious networks by mixing activation functions. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 425–432. ACM (2017)
17. Heidrich-Meisner, V., Igel, C.: Neuroevolution strategies for episodic reinforcement learning. Journal of Algorithms **64**(4), 152–168 (2009)
18. Ilievski, I., Akhtar, T., Feng, J., Shoemaker, C.A.: Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
19. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: Proceedings of the 24th international conference on Machine learning, pp. 473–480. ACM (2007)
20. Miconi, T., Clune, J., Stanley, K.O.: Differentiable plasticity: training plastic neural networks with backpropagation. arXiv preprint arXiv:1804.02464 (2018)
21. Montana, D.J., Davis, L.: Training feedforward neural networks using genetic algorithms. In: IJCAI, vol. 89, pp. 762–767 (1989)
22. Morse, G., Stanley, K.O.: Simple evolutionary optimization can rival stochastic gradient descent in neural networks. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, pp. 477–484. ACM (2016)
23. Rechenber, I.: Optimierung technischer systeme nach prinzipien der biologischen evolution. Ph.D. thesis, Verlag nicht ermittelbar (1970)
24. Risi, S., Togelius, J.: Neuroevolution in games: State of the art and open challenges. CoRR **abs/1410.7326** (2014). URL http://arxiv.org/abs/1410.7326
25. Rostami, S.: Preference focussed many-objective evolutionary computation. Ph.D. thesis, Manchester Metropolitan University (2014)
26. Rostami, S., Neri, F.: Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm. Integrated Computer-Aided Engineering **23**(4), 313–329 (2016)

27. Rostami, S., OReilly, D., Shenfield, A., Bowring, N.: A novel preference articulation operator for the evolutionary multi-objective optimisation of classifiers in concealed weapons detection. Information Sciences **295**, 494–520 (2015)
28. Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, (2016)
29. Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning. preprint arXiv:1703.03864 (2017)
30. Siebel, N.T., Botel, J., Sommer, G.: Efficient neural network pruning during neuro-evolution. In: 2009 International Joint Conference on Neural Networks, pp. 2920–2927. IEEE (2009)
31. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, pp. 2951–2959 (2012)
32. Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. Artificial life **15**(2), 185–212 (2009)
33. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
34. Storn, R.: On the usage of differential evolution for function optimization. In: Proc of North American Fuzzy Information Processing, pp. 519–523. IEEE (1996)
35. Such, F.P., Madhavan, V., Conti, E., Lehman, J., Stanley, K.O., Clune, J.: Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567 (2017)
36. Togelius, J., Schaul, T., Wierstra, D., Igel, C., Gomez, F., Schmidhuber, J.: Ontogenetic and phylogenetic reinforcement learning. Künstliche Intelligenz **23**(3), 30–33 (2009)
37. Wagenaartje, T.: wagenaartje/neataptic (2018). URL https://github.com/wagenaartje/neataptic
38. Wang, L., Feng, M., Zhou, B., Xiang, B., Mahadevan, S.: Efficient hyper-parameter optimization for nlp applications. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2112–2117 (2015)
39. White Jr, R.: A survey of random methods for parameter optimization. Simulation **17**(5), 197–205 (1971)
40. Whitley, D., Starkweather, T., Bogart, C.: Genetic algorithms and neural networks: Optimizing connections and connectivity. Parallel computing **14**(3), 347–361 (1990)
41. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics bulletin **1**(6), 80–83 (1945)
42. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE **87**(9), 1423–1447 (1999)
43. Yao, X., Liu, Y.: A new evolutionary system for evolving artificial neural networks. Trans. Neur. Netw. **8**(3), 694–713 (1997)
44. Yao, X., Liu, Y.: A new evolutionary system for evolving artificial neural networks. IEEE transactions on neural networks **8**(3), 694–713 (1997)
45. Young, S.R., Rose, D.C., Karnowski, T.P., Lim, S.H., Patton, R.M.: Optimizing deep learning hyper-parameters through an evolutionary algorithm. In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, p. 4. ACM (2015)