2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Efficient Neuroevolution using Island Repopulation and Simplex Hyperparameter Optimization

Aditya Shankar Thakur*, Akshar Bajrang Awari†, Zimeng Lyu‡ and Travis Desell§
Rochester Institute of Technology
Rochester, New York, USA
Email: *at4415@rit.edu, †aba7569@rit.edu, ‡zimenglyu@mail.rit.edu, §tjdvse@rit.edu

***Abstract*—Recent studies have shown that the performance of evolutionary neural architecture search (*i.e.*, neuroevolution) algorithms can be significantly improved by the use of island based strategies which periodically experience extinction and repopulation events. Further, it has been shown that the simplex hyperparameter optimization (SHO) method can also improve neuroevolution (NE) performance by optimizing neural network training hyperparameters while the NE algorithm also trains and designs neural networks. This work provides an extensive examination of combining island repopulation events with five different island-based variations of SHO. These methods are evaluated for the evolution of recurrent neural networks for the challenging problem of multivariate time series forecasting on two real world datasets. We show with statistical significance that adding repopulation to the SHO variants in almost every case improves performance, and for those that does there is no statistical difference. In addition, we find that one variant in particular, multi-island, random island best genome (MIRIB) performs the best across all experiment types.**

***Index Terms*—Neuroevolution, Time Series Forecasting, Hyperparameter Optimization, Recurrent Neural Networks**

I. Introduction

Determining the optimal design of a neural network for a given task is a challenging problem, which can be exacerbated by the fact that what constitutes an optimal neural network for a particular dataset may change depending on how it is used – some use cases may for example sacrifice accuracy for performance or low energy consumption [1]. Additionally, the manual design of neural networks is a time consuming task which also requires significant domain expertise. Due to this, neural architecture search (NAS) methods have become a significant area of research, as they can automate the design process of neural networks [2].

Traditional NAS techniques often rely on gradient-based [2] or reinforcement learning methods [3] to explore a large space of possible neural network architectures. However, searching for optimal architectures can also be done utilizing evolutionary algorithms for NAS (*i.e.*, neuroevolution). Neuroevolution (NE) algorithms are appealing in that they can provide a more flexible and expansive framework to search less restricted NAS search spaces [4].

Another benefit provided by NE algorithms is that they tend to be easily distributed, allowing scalable performance on high performance computing systems. They can even benefit from potentially superlinear speedup by using island-based distribution strategies [5]. Recent work has shown that NE algorithms can be even further sped up by utilizing extinction and repopulation events, preventing them from being stuck at local optima [6].

Another recent work by Kini *et al.* [7] has shown that NE algorithms can not only automate the process of training and designing neural networks, but also optimize the hyperparameters with which the neural networks are trained, further increasing their performance, using a method called simplex hyperparameter optimization (SHO). This work is novel in that while there are a few existing architecture and hyperparameter search methodologies [8], [9], generally NE algorithms have focused on evolving network topology and weights, and if they do any hyperparameter tuning they use common techniques such as random search [10], Grid Search [11], Bayesian optimization [12], gradient-based optimization [13], or population-based optimization [14].

This work extends on the work by Kini *et al.* [7] by exploring a series of five variants of SHO which can be combined with island-based distribution strategies as well as extinction and repopulation events for improved performance. These methods have been implemented as part of the Evolutionary eXploration of Augmenting Memory Models (EXAMM) algorithm, which is a NE method focused on the automated design of recurrent neural networks for time series forecasting.

Time series forecasting (TSF) plays a pivotal role in forecasting future trends and patterns, which facilitates informed decision making, optimizes resource allocation, and increases efficiency. Time series forecasting could be used for financial decision making [15], promoting retail sales [16], forecasting energy demand in power industry [17], predicting spread of diseases [18], and route planning in transportation [19]. Designing TSF models for various applications manually could be time consuming and requires tuning for the optimal hyperparameters, neural architecture search (NAS) is an effective way to search for the optimal neural network for specific TSF applications [20], [21].

This work evaluates the five SHO variants with and without repopulation events to optimize training hyperparameters

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

for the Adam and Nesterov momentum optimizers as part of EXAMM for the evolution of recurrent neural networks to perform TSF on two real world multivariate datasets. These datasets are challenging in that they are non-seasonal and noisy, one consisting of aviation sensor data coming from multiple flights of Cessna 172 (C172) aircraft, and the other from 7 years of sensor data from multiple wind turbines. Results show that with statistical significance repopulation events improve performance across almost all experiments, and do not have a statistical difference when they do not. Further, we show that a SHO variant which combines global island selection for diversity with local selection of best genomes hyperparmeters within islands performed the best across all experiments.

II. EXAMM - Evolutionary Exploration of Augmenting Memory Models

The Evolutionary eXploration of Augmenting Memory Models (EXAMM) algorithm is a novel asynchronous and distributed neuro-evolution strategy. EXAMM automates the design and training of recurrent neural networks (RNNs) with a particular focus on multivariate time series forecasting (TSF). EXAMM employs an island-based strategy RNN where instead of a utilizing single population in the evolution process, n islands with a capacity of m genomes evolve independently with the occasional transfer of information between islands via an inter-island crossover operation in genome (RNN) generation. EXAMM typically initializes itself with a single minimal seed genome, which is a RNN that is just a fully connected input to output nodes. It is also possible to utilize pre-designed and trained RNN in the case of transfer learning [22]. Following this, EXAMM evolves progressively more complex RNNs via a set of mutation, inter-island and intra-island crossover operations.

EXAMM's distributed algorithm is comprised of single master process and several worker processes which asynchronously request genomes, which are trained for a selected amount of epochs, and after which the performance, architecture, hyperparameters and weights of those genomes are reported back to the master process. If a newly evaluated genome is more fit (*i.e.*, has a lower mean squared error (MSE) on the validation datset) than the worst genome in its target island, the worst genome will be removed from that island and the new more fit genome will replace it. Due to space restraints, we refer the reader to Ororbia *et al.* [23] which provides an in depth description of the EXAMM algorithm.

III. Simplex Hyperparameter Optimization (SHO)

Simplex Hyperparameter Optimization (SHO) is a method created by Desell *et al.* [24], which built upon the initial notion of the Nelder-Mead simplex technique [25]. SHO can utilize multiple parent values to estimate a gradient and perform crossover in highly scalable distributed evolutionary algorithms. Desell *et al.* later extended SHO as a method for optimizing Convolutional Neural Network (CNN) hyperparameters [26]. More recently, Kini *et al.* [7] utilized SHO to optimize RNN-specific hyperparameters in the EXAMM algorithm with promising results.

SHO begins with a burn-in phase to reduce initial bias of the method. The burn-in phase typically lasts until all islands (or a single population) are full, or until a given number of genomes have been generated. In the burn in phase, for each of the parameters p, initial values are randomly assigned within predefined bounds, $\sim U[h_{p,min}, h_{p,max})]$:

$$h_i = (rand(0,1) * (h_{p,max} - h_{p,min})) + h_{p,min} \quad (1)$$

After the burn-in phase, the main phase of SHO performs hyperparameter optimization. In this phase, to generate hyperparameters N distinct genomes are selected from within the available population(s). Amongst the selected genomes, SHO calculates the gradient between the hyperparameters of the best genome $h_{best,p}$ and the mean of the hyperparameters of the other $N-1$ genomes $h_{avg,p}$. It then finds a random point along the line made between the average and best hyperparameters $l1$ and $l2$ and multiplies the gradient by this point to generate the new simplex hyperparameters $h_{new,p}$:

$$r = (rand(0,1) * l_1) - l_2 \quad (2)$$

$$h_{new,p} = h_{avg,p} + r * (h_{best,p} - h_{avg,p}) \quad (3)$$

IV. Simplex Hyperparameter Optimization (SHO) Variants

The previous work by Desell [26] only utilized a single island without repopulation, and while the more recent work by Kini *et al.* [7] utilized islands, it did not utilize repopulation and only had a preliminary implementation of island SHO which randomly selected parent genomes from a randomly selected island (described as the MIRI method in the next Section). This work expands on prior work by providing an in depth examination of multiple methodologies for utilizing SHO given islands of populations, and also incorporates repopulation into the process. Note for all methodologies, the N parental genomes for SHO are selected without replacement. In particular, we investigate the five following variants on SHO (each with and without repopulation):

A. Single Island, Full Population SHO (SIFP)

To provide a baseline methodology without islands, SIFP utilizes a single population (single island) which is used for both genome generation and hyperparameter optimization with SHO. This is the same method as utilized in Desell [26].

B. Multi Island, Full Population SHO (MIFP)

This method is used to investigate the performance of islands while still utilizing the baseline SHO method in SIFP. Genomes are generated from multiple islands in a round-robin fashion (as is standard in EXAMM), however hyperparameters to train those genomes are generated by SHO selecting the N parents at random from across all islands (as if they were a single population).

C. Multi Island, Random Island SHO (MIRI)

This method represents strategy used in prior work by Kini *et al.* [7]. This also uses round-robin genome generation from multiple islands, however for hyperparameter generation with SHO, it selects a random island (which may not be the island used for genome generation) and then the N randomly selected individuals for SHO from that random island.

D. Multi Island, Current Island SHO (MICI)

This is similar to MIRI except that the N individuals for SHO are randomly selected from the same (current) island which is generating the RNN genome instead of a random island.

E. Multi Island, Random Island Best Genome SHO (MIRIB)

The last method utilizes a more selective but global approach to SHO. MIRIB uses round-robin genome generation from multiple islands as in the previous strategies, however for SHO, it selects N islands at random and then from each of the selected islands, it selects the hyperparameters from best individual genome on that island in terms of its fitness scores.

V. Island Repopulation

To further improve the performance of the EXAMM neuroevolution process, this work also investigates combining these variants of SHO (MIFP, MIRI, MIRIB and MICI) with the island repopulation procedure performed originally by Lyu *et al.* [6]. This approach reduces suboptimal performance of EXAMM algorithm, which can occur due to premature convergence of the island-based strategies. This strategy uses periodic extinction and repopulation events which occur with a specified frequency (in this work either every 150 or 300 generated genomes). When one of these events occur, the island whose best genome is the worst compared to all other islands is repopulated by removing all genomes and repopulating the island with new genomes generated by mutation and crossover operations from genome with the overall best fitness across all other islands. This both encourages diversity but also prevents islands from getting prematurely "stuck" in a local minima.

VI. Results

A. Datasets

Similar to the work done by Kini *et al.* [7], this work was evaluated using two large scale, real world, multivariate time series benchmark datasets provided as part of the EXAMM github repository[1]. The first dataset consists of flight data from the National General Aviation Flight Information Database (NGAFID). This dataset contains 12 files each representing a flight from a Cessna 172 Skyhawk (C172). The duration of each flight ranges between from 1 to 3 hours, with per second recordings from 31 sensors. Recorded values from the pitch sensor of the plane were considered as the response variable to predict for the flight dataset experiments.

The second dataset was derived from ENGIE's La Haute Borne open data wind farm. It utilized multivariate time series data collected from 5 wind turbines between 2013 and 2022, with 22 sensors being recorded with a 10 minute frequency. The response variable predicted for the experiments was Average Active Power.

[1]https://github.com/travisdesell/exact

Weight Update	Hyperparamer	Initial Range	After Burn In
-	η	[0.001, 0.05]	[0.00001, 0.3]
Nesterov	μ	[0.9, 0.99]	[0.9, 0.99]
Adam	β_1	[0.9, 0.99]	[0.9, 0.99]
	β_2	[0.9, 0.99]	[0.9, 0.99]
	ϵ	[1e-9, 1e-8]	[1e-9, 1e-8]

TABLE I: SHO Hyperparameter Value Ranges

B. Experimental Setup

Each dataset was evaluated using each of the five variations on SHO to optimize hyperparameters for Nesterov momentum and Adam as neural network training optimizers. All methods except for SIFP were evaluated using no repopulation, repopulation every 150 generated genomes, and repopulation every 300 generated genomes. It is not possible to utilize repopulation on SIFP as there is only a single population. The experiments with islands utilized 10 islands with each having an island capacity of 10, SIFP utilized a single island with a capacity of 100. Each EXAMM run generated and evaluated 10,000 genomes (RNNs). Table I shows the pre and post burn-in hyperparameter value ranges. Each of the 54 experimental setups (9 SHO variations across 2 optimizers across 3 repopulation strategies) were repeated 20 times to gather enough performance results for tests of statistical significance. These experiments were run on Rochester Insitute of Technology's research computing cluster [27]. All experiments were run using 16 CPU cores and 8GB RAM.

C. Effects of Repopulation on SHO Performance

Figure 1 demonstrates the convergence of the varying SHO variants, with the solid line denoting the performance of the mean of the best found RNN genome mean squared error (MSE) across the 20 repeats for the experiments. The shaded in area presents the range between the best found RNN of the best performing experiment (overall lowest MSE) and the best found RNN of the worst performing experiment after generating that many genomes. These results highlight how well the variants which utilize islands perform over the single population SIFP variant. Additionally, for the C172 data, we see an interesting effect where the MIRIB variant performs worse than other island variants without repopulation, but the best when repopulation is added. MIRIB also shows the best performance across the wind dataset for Adam optimization, however not for the wind dataset with Nesterov momentum.

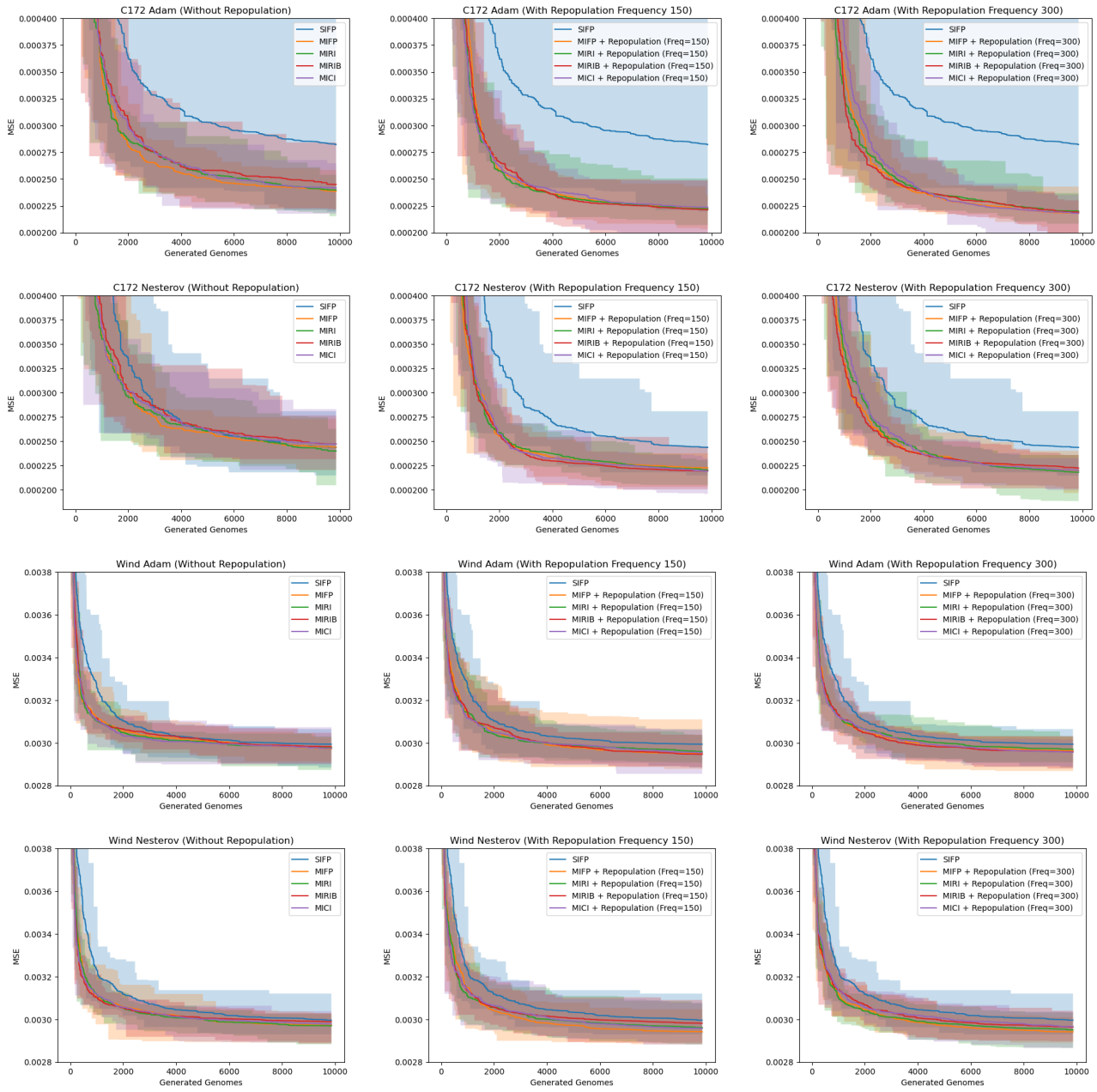

Fig. 1: Search fitness over time corresponding to different variants of SHO tuned EXAMM compared with and without repopulation with different frequencies for both datasets and varying weight update methods.

Tables II, III, IV and V further summarize the final results of the experiments with a comparison of SHO with and without repopulation for the Adam and Nesterov optimizers on the C172 and wind datasets across all experiments with (RP) and without (No RP) repopulation. The Average Best MSE column in these tables presents the average global best fitness at the end of evolution, across the entire set of 20 repeated runs for each experiment. The Global Best MSE column presents the best (minimum) MSE value across all the 20 runs for each experiment. The no repopulation (No RP) and repopulation (RP) columns compare the methods with and without repopulation, given a repopulation frequency of 150. MSE values in italic show the best MSE for the variant, and MSE values in bold show the best MSE value across all variants, with and without repopulation. The tables also show the Mann–Whitney U test p-values comparing the SHO global best validation MSE with and without repopulation. Values in bold show the variants where repopulation performed better with a statistically significant difference with $\alpha = 0.1$.

For the C172 data, results show that with statistical significance across all variants, the variants with repopulation

SHO	C172 Adam				
	p-values	Avg Best MSE		Global Best MSE	
		No RP	RP	No RP	RP
SIFP	-	0.0003342	-	0.000234	-
MIFP	**0.000047**	0.0003165	*0.0002818*	0.000258	*0.000234*
MIRI	**0.000066**	0.0003154	*0.0002911*	0.000253	*0.000239*
MIRIB	**0.000029**	0.0003208	***0.0002773***	0.000247	***0.000227***
MICI	**0.000104**	0.0003061	*0.0002847*	0.000243	*0.000231*

TABLE II: SHO variant performance significance testing results, with and without repopulation, for the flight dataset and Adam optimizer.

SHO	C172 Nesterov				
	p-values	Avg Best MSE		Global Best MSE	
		No RP	RP	No RP	RP
SIFP	-	0.0003349	-	0.000230	-
MIFP	**0.0000052**	0.0003004	*0.0002591*	0.000254	*0.000219*
MIRI	**0.000029**	0.0002909	*0.0002525*	0.000228	*0.000215*
MIRIB	**0.0000006**	0.0003112	***0.0002327***	0.000252	***0.000208***
MICI	**0.000087**	0.0002957	*0.0002578*	0.000257	*0.000224*

TABLE III: SHO variant performance significance testing results, with and without repopulation, for the flight dataset and Nesterov optimizer.

SHO	Wind Adam				
	p-values	Avg Best MSE		Global Best MSE	
		No RP	RP	No RP	RP
SIFP	-	0.00313	-	0.00303	-
MIFP	**0.053**	0.003051	*0.003040*	0.00300	*0.00295*
MIRI	0.119	*0.003076*	0.003088	*0.00301*	0.00303
MIRIB	**0.009**	0.003039	***0.002938***	0.00296	***0.00238***
MICI	0.133	*0.002992*	0.002997	*0.00285*	0.00293

TABLE IV: SHO variant performance significance testing results, with and without repopulation, for the wind dataset and Adam optimizer.

SHO	Wind Nesterov				
	p-values	Avg Best MSE		Global Best MSE	
		No RP	RP	No RP	RP
SIFP	-	0.00308	-	0.00299	-
MIFP	**0.029**	0.002937	*0.002916*	0.00295	*0.002891*
MIRI	0.457	*0.003047*	0.003095	*0.002958*	0.00303
MIRIB	**0.006**	0.003101	***0.002867***	0.003	***0.00244***
MICI	0.243	*0.003053*	0.003071	*0.00293*	0.002978

TABLE V: SHO variant performance significance testing results, with and without repopulation, for the wind dataset and Nesterov optimizer.

perform better. Additionally, the MIRIB variant for both the Nesterov and Adam optimizers performed the best. On the wind dataset, MIRI and MICI did not have a statistically significant difference with and without repopulation, however MIFP did, and MIRIB with repopulation, similar to the flight dataset found the best genomes in the average and overall cases, again with statistical significance.

Overall, these results are interesting in first confirming that repopulation of island strategies provides a statistically significant improvement in performance, or at the very least does not reduce performance. Additionally, the MIRIB variant with repopulation was shown to be highly reliable, performing best across all experiments. With the MIFP variant performing second best, these results suggest that utilizing a more global method for SHO optimization, when coupled with repopulation allows for faster convergence to better performing hyperparameters, with the MIFIB variant allowing for the best combination of exploration (using hyperparameters from across all islands) and exploitation (using the best hyperparameters from those islands). The variants which use hyperparameters from a single island (MIRI and MICI) most likely lack good exploitation in the case of MIRI, and exploration in the case of MICI.

VII. Discussion

The use of islands in distributed evolutionary algorithms has been a long used method for improving their performance, as they have been shown to potentially allow superlinear speedup [5]. More recent work has shown that utilizing islands with repopulation events can provide additional significant performance improvement for evolutionary neural architecture search (neuroevolution) algorithms [6]. This work extends on this with an investigation of how to combine a recent method for combining hyperparamater optimization for training of neural networks generated in a neuroevolution algorithm called simplex hyperparameter optimization (SHO) [7] with repopulating islands.

Four island based variants of SHO were investigated with and without repopulation, and compared to a baseline single population methodology on the challenging task of time series forecasting using two real world datasets. Results show that for almost all experiments, utilizing repopulation provides a statistically significant improvement in performance, and when it does not the difference is not statistically significant. Additionally, we show that a variant which combines a global search of hyperparameters with an exploitative selection process where the best set of hyperparameters from each island is selected for SHO to generate new candidate hyperparameters performed the best across all datasets and optimization problems.

The methodologies presented in this paper are generic and can be applied to any population based neuroevolution algorithm. In particular, they provide effective methodologies for the automatic design and training of neural networks, while additionally optimizing the hyperparameters to train those neural networks for further performance benefit. These methods can significantly simplify automated design of neural networks, as in combination they effectively automates most of the design process.

VIII. Acknowledgements

This work has been partially supported by the Federal Aviation Administration National General Aviation Flight Information Database (NGAFID) award, the National Science Foundation under Grant Number #2225354, and the U.S. Department of Energy, Office of Science, Office of Advanced Combustion Systems under Award Number #FE00031750. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Department of Energy, National Science Foundation or the Federal Aviation Administration. We also thank RIT's Research Computing team for their assistance and support.

References

[1] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, pp. 1–34, 2021.

[2] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[3] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[4] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE transactions on neural networks and learning systems*, 2021.

[5] E. Alba, "Parallel evolutionary algorithms can achieve super-linear performance," *Information Processing Letters*, vol. 82, no. 1, pp. 7–13, 2002.

[6] Z. Lyu, J. Karns, A. ElSaid, M. Mkaouer, and T. Desell, "Improving distributed neuroevolution using island extinction and repopulation," in *Applications of Evolutionary Computation*, P. A. Castillo and J. L. Jiménez Laredo, Eds. Cham: Springer International Publishing, 2021, pp. 568–583.

[7] A. D. Kini, S. S. Yadav, A. S. Thakur, A. B. Awari, Z. Lyu, and T. Desell, "Co-evolving recurrent neural networks and their hyperparameters with simplex hyperparameter optimization," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO '23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 1639–1647. [Online]. Available: https://doi.org/10.1145/3583133.3596407

[8] A. Zela, A. Klein, S. Falkner, and F. Hutter, "Towards automated deep learning: Efficient joint neural architecture and hyperparameter search," *arXiv preprint arXiv:1807.06906*, 2018.

[9] X. Dong, M. Tan, A. W. Yu, D. Peng, B. Gabrys, and Q. V. Le, "Autohas: Differentiable hyper-parameter and architecture search," *arXiv preprint arXiv:2006.03656*, vol. 4, no. 5, 2020.

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[11] B. Shekar and G. Dagnew, "Grid search-based hyperparameter tuning and classification of microarray cancer data," in *2019 second international conference on advanced computational and communication paradigms (ICACCP)*. IEEE, 2019, pp. 1–8.

[12] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, K. Leyton-Brown *et al.*, "Towards an empirical foundation for assessing bayesian optimization of hyperparameters," in *NIPS workshop on Bayesian Optimization in Theory and Practice*, vol. 10, no. 3, 2013.

[13] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *International conference on machine learning*. PMLR, 2015, pp. 2113–2122.

[14] J. Parker-Holder, V. Nguyen, and S. J. Roberts, "Provably efficient online hyperparameter optimization with population-based bandits," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 200–17 211, 2020.

[15] B. Krollner, B. Vanstone, and G. Finnie, "Financial time series forecasting with machine learning techniques: A survey," in *European Symposium on Artificial Neural Networks: Computational Intelligence and Machine Learning*, 2010, pp. 25–30.

[16] Y. Ensafi, S. H. Amin, G. Zhang, and B. Shah, "Time-series forecasting of seasonal items sales using machine learning–a comparative analysis," *International Journal of Information Management Data Insights*, vol. 2, no. 1, p. 100058, 2022.

[17] C. Garcia-Ascanio and C. Maté, "Electric power demand forecasting using interval time series: A comparison between var and imlp," *Energy policy*, vol. 38, no. 2, pp. 715–725, 2010.

[18] V. K. R. Chimmula and L. Zhang, "Time series forecasting of covid-19 transmission in canada using lstm networks," *Chaos, solitons & fractals*, vol. 135, p. 109864, 2020.

[19] Z. Karami and R. Kashef, "Smart transportation planning: Data, models, and algorithms," *Transportation Engineering*, vol. 2, p. 100013, 2020.

[20] Z. Lyu, A. ElSaid, J. Karns, M. Mkaouer, and T. Desell, "An experimental study of weight initialization and lamarckian inheritance on neuroevolution," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2021, pp. 584–600.

[21] Z. Lyu, A. Ororbia, and T. Desell, "Online evolutionary neural architecture search for multivariate non-stationary time series forecasting," *Applied Soft Computing*, p. 110522, 2023.

[22] A. ElSaid, J. Karns, Z. Lyu, D. Krutz, A. Ororbia, and T. Desell, "Improving neuroevolutionary transfer learning of deep recurrent neural networks through network-aware adaptation," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 315–323.

[23] A. Ororbia, A. ElSaid, and T. Desell, "Investigating recurrent neural network memory structures using neuro-evolution," in *Proceedings of the genetic and evolutionary computation conference*, 2019, pp. 446–455.

[24] T. Desell, B. Szymanski, and C. Varela, "An asynchronous hybrid genetic-simplex search for modeling the milky way galaxy using volunteer computing," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, 2008, pp. 921–928.

[25] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[26] T. Desell, "Developing a volunteer computing project to evolve convolutional neural networks and their hyperparameters," in *2017 IEEE 13th International Conference on e-Science (e-Science)*. IEEE, 2017, pp. 19–28.

[27] Rochester Institute of Technology, "Research computing services," 2022. [Online]. Available: https://www.rit.edu/researchcomputing/