Co-evolving Recurrent Neural Networks and their Hyperparameters with Simplex Hyperparameter Optimization

Amit Dilip Kini*
Rochester Institute of Technology
Rochester, New York, USA
ak3328@rit.edu

Swaraj Sambhaji Yadav*
Rochester Institute of Technology
Rochester, New York, USA
sy9421@rit.edu

Aditya Shankar Thakur
Rochester Institute of Technology
Rochester, New York, USA
at4415@rit.edu

Akshar Bajrang Awari
Rochester Institute of Technology
Rochester, New York, USA
aba7569@rit.edu

Zimeng Lyu
Rochester Institute of Technology
Rochester, New York, USA
zimenglyu@mail.rit.edu

Travis Desell
Rochester Institute of Technology
Rochester, New York, USA
tjdvse@rit.edu

Abstract

Designing machine learning models involves determining not only the network architecture, but also non-architectural elements such as training hyperparameters. Further confounding this problem, different architectures and datasets will perform more optimally with different hyperparameters. This problem is exacerbated for neuroevolution (NE) and neural architecture search (NAS) algorithms, which can generate and train architectures with a wide variety of architectures in order to find optimal architectures. In such algorithms, if hyperparameters are fixed, then suboptimal architectures can be found as they will be biased towards the fixed parameters. This paper evaluates the use of the simplex hyperparameter optimization (SHO) method, which allows co-evolution of hyperparameters over the course of a NE algorithm, allowing the NE algorithm to simultaneously optimize both network architectures and hyperparameters. SHO has been previously shown to be able to optimize hyperparameters for convolutional neural networks using traditional stochastic gradient descent with Nesterov momentum, and this work extends on this to evaluate SHO for evolving recurrent neural networks with additional modern weight optimizers such as RMSProp and Adam. Results show that incorporating SHO into the neuroevolution process not only enables finding better performing architectures but also faster convergence to optimal architectures across all datasets and optimization methods tested.

CCS Concepts

• **Computing methodologies** → **Online learning settings**; *Neural networks*; *Genetic algorithms*; • **Applied computing** → **Forecasting**.

Keywords

Hyperparameter Tuning, Time Series Forecasting, Neural Architecture Search, Recurrent Neural Networks, NeuroEvolution.

ACM Reference Format:
Amit Dilip Kini*, Swaraj Sambhaji Yadav*, Aditya Shankar Thakur, Akshar Bajrang Awari, Zimeng Lyu, and Travis Desell. 2023. Co-evolving Recurrent Neural Networks and their Hyperparameters with Simplex Hyperparameter Optimization. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583133.3596407

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07…$15.00
https://doi.org/10.1145/3583133.3596407

1 Introduction

Machine learning (ML) models have been widely adopted across various domains, such as computer vision [20, 26], natural language processing (NLP) [3, 10], healthcare [1], financial decision making [36], and others. However, the crucial challenges in building these models are not only how determine which neural network architectures are most appropriate for the target datasets, but also how to optimize the neural network training process for optimal performance [6].

Hyperparameter optimization typically involves tuning parameters related to training the models (*e.g.*, learning rate, velocity, etc), as opposed to the parameters (*e.g.*, neural network weights) learned during the training process. Neural architecture search (NAS) alternatively focuses on determining optimal architectural components, such as the network type, layer size, how layers are connected, activation functions, cellular compositions, etc. To find optimal network architectures and components, NAS can utilize a variety of methods, such as reinforcement learning (RL) [22], Bayesian optimization (BO) [46], gradient based methods [29], and evolutionary algorithms (EAs) [17, 30, 34]. Utilizing EAs for NAS is commonly termed neuroevolution.

Many NAS and NE algorithms involve training sampled or generated networks for a number of epochs and then evaluating their fitness as the trained network's loss on a target validation dataset. However, they commonly use fixed hyperparameters which can lead to less optimal architectures being discovered as optimal hyperparameters for networks can change during the search process, or the chosen hyperparameters may not be those which would be optimal for the optimal architecture. Surprisingly, there are very few

*These authors contributed equally to this work

works which investigate jointly learning both architecture and non-architectural hyperparameter tuning [11, 14], where in most cases the NAS algorithms only focus on architectural hyperparameters, and the training-related hyperparameters, such as learning rate, batch size, or optimization algorithm, are often set using default values or based on best practices for the given problem domain [37].

This work investigates the Simplex Hyperparameter Optimization (SHO) method, initially proposed by Desell [12] to co-evolve both convolutional neural network architectures and their training hyperparameters for stochastic gradient descent (SGD) with Nesterov momentum [9]. This work extends this prior work in determining the applicability of SHO to co-evolve training hyperparameters as an extenstion to the Evolutionary eXploration of Augmenting Memory Models (EXAMM [34]) recurrent neural network (RNN) neuroevolution algorithm. In addition to examining SHO for SGD with Nesterov momentum, we also evaluate it on the more modern training opimization methods RMSProp [41] and Adam [27]. We utilize two real world datasets, one from the aviation and the other from power systems to evolve RNNs for the challenging problem of multivariate time series forecasting.

The SHO method utilizes a modification of the venerable Simplex method, originally proposed by Nelder and Mead [33]. SHO can operate independently of a NE or NAS process, only requiring populations of candidates architectures and hyperparameters. When generating new hyperparameters for candidate networks, it selects a random group of parents and performs a randomized line search along the gradient determined by the Simplex method. Results show that SHO provides significant benefit to the NE process, allowing EXAMM to not only find better performing architectures than with fixed hyperparameters, but also improving the speed of convergence, allowing EXAMM to find those architectures in less time. SHO allows EXAMM to keep learning during the neuroevolution process, adapting to distribution shift as more evolved architectures will be larger and train better with different hyperparameters. SHO makes EXAMM more adaptable to time series datasets in differnet domains as it also learns hyperparamters better tuned to the datasets it is used on. This work further highlighting the importance of incorporating hyperparameter optimization into the NE process, as utilizing fixed hyperparameters, even when hand tuned as good defaults from other datasets led to sub-optimal results.

2 Related Work

Hyperparameter tuning is an important part of designing and training machine learning and deep learning models. Commonly used hyperparameter tuning methods include random search [8], grid search [39], Bayesian Optimization (BO) [16], gradient-based optimization [31], and evolutionary algorithms (EAs).

Grid search is still widely used in optimizing the training results for machine learning models, and while it is very easy and common, it is also highly computationally expensive, as it scales exponentially in relation to the number of hyperparameters[40]. Also, grid search has limited fidelity, as step sizes for each hyperparameter being optimized need to be selected, and other values will not be examined. However, it has shown good results for applications prior unstudied domains, such as prediction of HIV/AIDS test results [7], prediction of later occurrence of breast cancer [23], and NO2 pollution prediction [4].

Rather than exhaustively evaluating all possible combinations, random search samples random values from the search space. Therefore it can be more efficient than grid search in high dimensional search spaces [8], even achieving similar performance to grid search or other meta-heuristics [32].

Bayesian optimization (BO) balances exploration and exploitation to efficiently find the best hyperparameters, and can generally provide superior results, particularly when the search space is non-convex or has complex interactions between hyperparameters [43]. Falkner *et al.*have combined using BO and bandit based methods for effective hyperparameter optimization [21]. Klein *et al.* propose BO methods for fast hyperparameter tuning for large datasets [28]. The work conducted by Ranjit *et al.* and Joy *et al.* also shows that using BO for hyperparameter tuning in machine learning could can achieve good results [25, 35].

EAs use evolutionary rules for hyperparameter tuning, EAs are generally good for black-box optimization and high dimensional optimization problems [5]. They has shown promising results in hyperparameter optimization, particularly when the objective function is noisy, discontinuous, or has multiple local optima [43]. Many related work have used EAs for hyperparameter tuning for machine learning models [42, 44]. Jinny *et al.* use GAs to optimize parameters for coronary heart disease early detection models [24]. In particular, Alibrahim *et al.* found that genetic algorithms perform better than grid search and BO for hyperparameter tuning [2].

NAS methods generally use pre-defined values for training hyperparameters, and those values usually come from experience in the specific domain [34]. However, some research has investigated the combination of NAS and hyperparamer tuning. In addition to the prior work by Desell [12] that this work extends, Zela *et al.* combined BO and Hyperband for joint NAS and hyperparameter search [45] and Egele *et al.* combine aging evolution (AE) for NAS and BO for hyperparameter tuning to adapt data-parallel training [15].

3 Evolutionary Exploration of Augmenting Memory Models

Evolutionary eXploration of Augmenting Memory Models (EXAMM) is a distributed asynchronous neuro-evolution (NE) algorithm that evolves progressively larger RNNs for large-scale, multivariate, real-world TSF [18, 19]. EXAMM utilizes an island based strategy, where n islands each maintain a population of m genomes (RNNs). Each island starts with a single minimum seed genome that only has input-to-output connections. Workers iteratively request genomes from master process, which generates new genomes from each island via mutation, inter-island or intra-island crossover in a round-robin fashion (see Ororbia *et al.* [34] an in depth presentation of the EXAMM algorithm). When workers receive a genome from the master process, they train it for a fixed number of epochs and then report the genome, including the final trained weights and fitness on validation data, back to the master. The master updates each island in a steady state fashion, where newly evaluated genomes replace the worst genome in the source island they were generated from if their fitness is higher than the (current) worst fitness value.

4 Simplex Hyperparameter Optimization

The venerable simplex method was originally proposed by Nelder and Mead [33]. It is a derivative-free optimization method which can make it very suitable to optimize training neural network hyperparameters in NE/NAS algorithms. Desell *et al.* extended this algorithm to operate as a multi-parent crossover algorithm [13] for large scale evoluationary algorithms, and then later extended it to co-optimize hyperparameters of convolutional neural networks during a neural architecture search process [12] in a method called Simplex Hyperparameter Optimization (SHO). Given the success of this prior work, we extend SHO in this work to optimize RNN hyperparameters as part of the EXAMM neuroevolution algorithm.

SHO begins with a *burn-in* phase to get an initial distributions of hyperparameters to later optimize. Parameters are initially generated uniformly at random within fixed ranges for each hyperparameter i, $\sim U[h_{i,min}, h_{i,max})]$:

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

Where the minimum and maximum values present a range typically 10% to 20% smaller than the full range which opens up once SHO begins to optimize hyperparameters. For this work, the burn-in phase lasted until each island's population was filled.

After the burn in, SHO begins optimization of hyperparameters. It chooses N distinct individuals at random from the population. Note that these N individuals are selected at random independently of parent selection for crossover and mutation, which allows SHO to operate independently from the neural architecture search utilized, making it highly generic.

Given these N individuals, the gradient is then calculated between the hyperparameters of the best selected genome and the average hyperparameters of the $N-1$ other genomes. Given r as a random point along the line bounded between two SHO hyperparameters, l_1 and l_2, which is multiplied by the gradient. All of the new hyperparameters, $h_{new,i}$ are calculated as a random point along the line between the best hyperparameters, $h_{best,i}$, and the average hyperparameters, $h_{avg,i}$, as follows:

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

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

From equation 2 we calculate the value of r by getting a random number between 0 and 1 and utilizing the suggested values of $l_1 = 2.0$ and $l_2 = 0.5$, then using equation 3 we get the values the new hyperparameters of a newly created RNN genome. These values are bounded by a minimum and maximum range for each hyperparameter and then utilized by the newly created genome.

5 Results

5.1 Datasets

We use two real-world high-dimensional datasets for all the experiments. The first dataset comes from the National General Aviation Flight Information Database (NGAFID), where flight data from 12 Cessna 172 Skyhawks (C172s) aircraft was utilized. The flight durations ranged from 1 to 3 hours, collected from 31 sensors. We selected the pitch of the plane as the output parameter to forecast for the flight dataset.

Weight Update	Hyperparameter	Initial Range	After burn in
-	η	[0.001, 0.05]	[0.00001, 0.3]
Nesterov	μ	[0.9, 0.99]	[0.9, 0.99]
RMSProp	ϵ	[1e-9, 1e-8]	[1e-9, 1e-8]
	decay	[0.85, 0.95]	[0.85, 0.95]
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 1: SHO Hyperparameter Value Ranges

Hyperparameter	Experiment Hyperparameters
η	0.0001, 0.001, 0.01, 0.02, 0.05
β_1	0.9, 0.99
β_2	0.9, 0.99
ϵ	1e-9, 1e-8
μ	0.9, 0.95, 0.99
decay	0.85, 0.9, 0.95

Table 2: Experiment Hyperparameter Combinations

Hyper-params	Nesterov		RMSProp		Adam	
	C172	Wind	C172	Wind	C172	Wind
η	0.0255	0.0236	0.0126	0.0199	0.0172	0.0185
β_1	-	-	-	-	0.9048	0.9061
β_2	-	-	-	-	0.9745	0.9742
ϵ	-	-	3.00e-8	3.20e-8	3.07e-8	2.90e-8
μ	0.9455	0.9515	-	-	-	-
decay	-	-	0.9023	0.9061	-	-

Table 3: SHO Optimized Hyperparameter Values

The second dataset was derived from ENGIE's La Haute Borne open data wind farm between 2017 and 2020. This dataset is multivariate with 22 parameters, non-seasonal, and the parameter recordings are not independent. The wind turbine data consists of readings taken every 10 minutes from 2013 to 2020, and we selected the Average Active Power as the output parameter to forecast for this wind turbine dataset.

5.2 Experimental Setup

In contrast to the original work by Desell [12], which only optimized hyperparameters related to stochastic gradient descent (SGD) *e.g.*, the learning rate and velocity for Nesterov momentum [9] for convolutional neural networks, this work investigates optimizing hyperparameters not only for SGD with momentum, but also Adam [27], and RMSProp [41] while evolving recurrent neural networks. Equation 4 to 11 shows the weight update equation for three methods. For Nesterov, the learning rate η and momentum μ were tuned, and for RMSProp, the learning rate η, decay rate, and ϵ were tuned. For Adam, the hyperparameters tuned were learning rate η, β_1, β_2, and ϵ. Each hyperparameter was optimized from the value range shown in Table 1 for burn-in and post-burn-in.

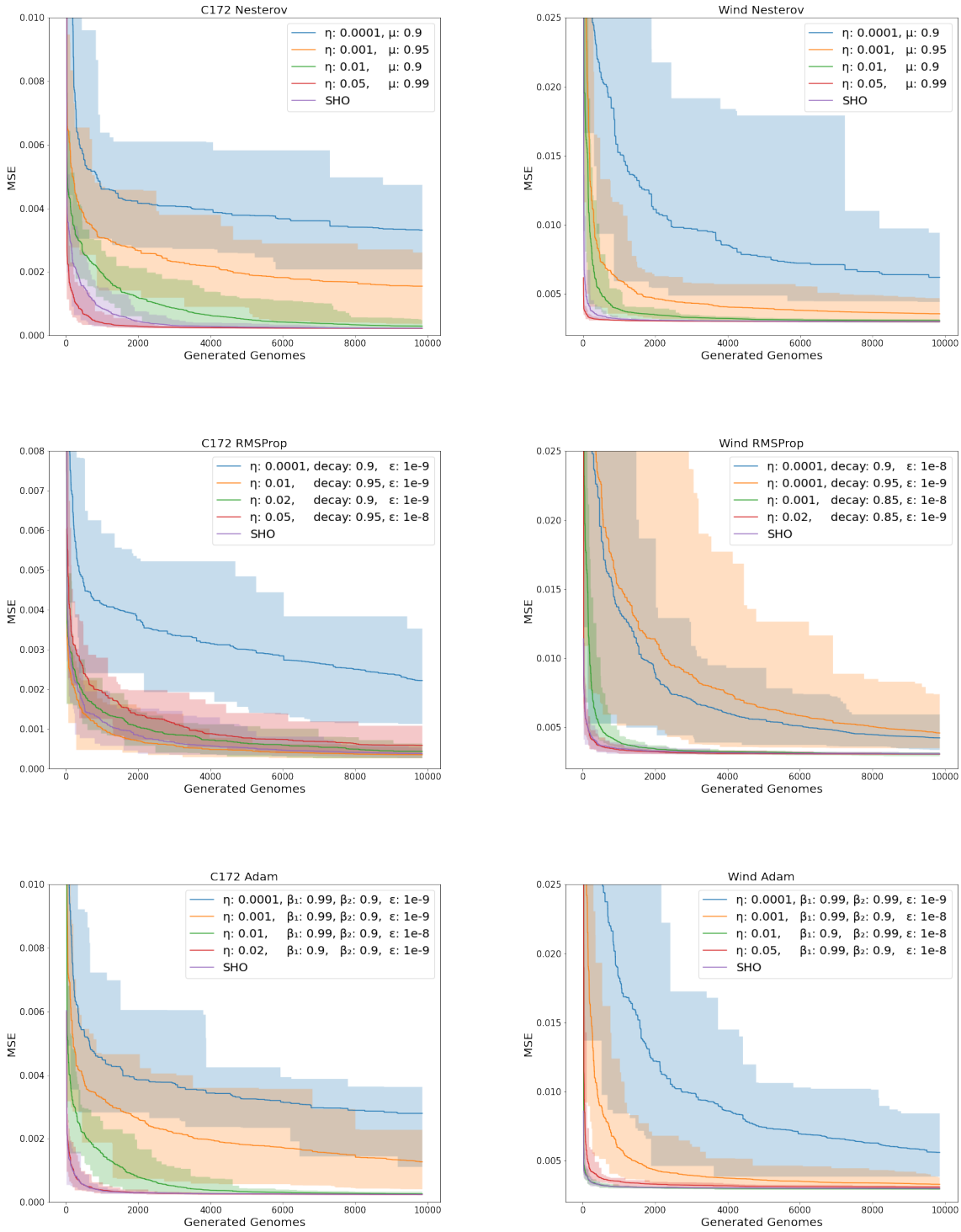

Figure 1: Search fitness over time for SHO tuned EXAMM compared to EXAMM with fixed hyperparameters for both datasets and varying weight update methods. SHO tuned EXAMM converges nearly as fast or faster as the best fixed parameters while still finding genomes with the best fitness.

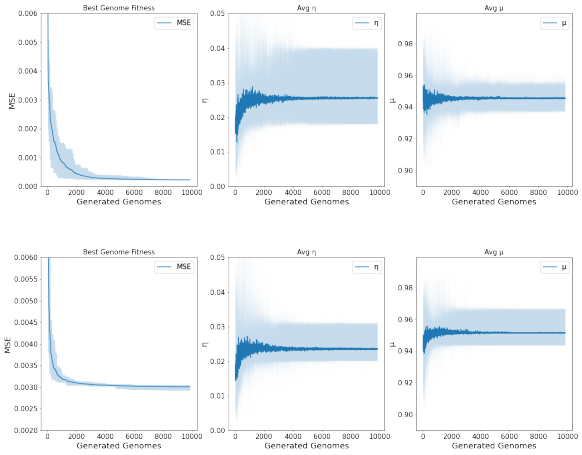

Figure 2: SHO Generated Hyperparameter Values for Nesterov Momentum.

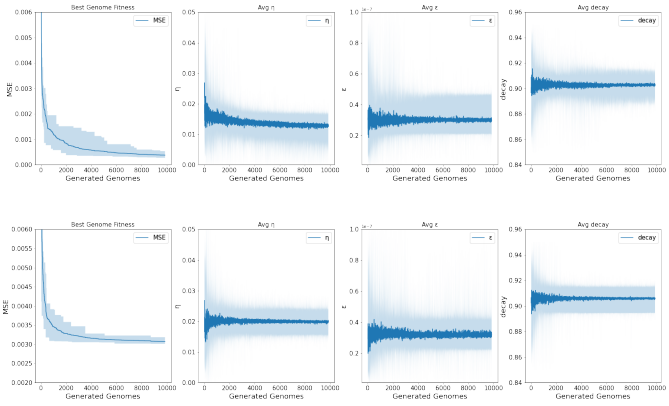

Figure 3: SHO Generated Hyperparameter Values for RMSProp.

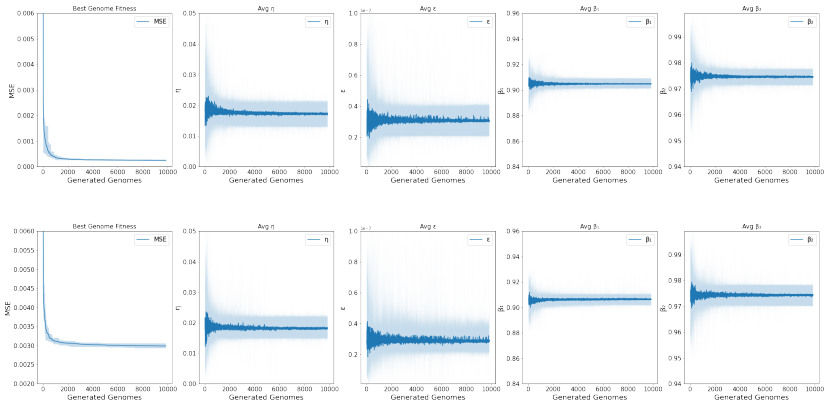

Figure 4: SHO Generated Hyperparameter Values for Adam.

Nesterov:

$$velocity\_pre = velocity, \tag{4}$$

$$velocity = (\mu \times velocity) - (\eta \times gradient), \tag{5}$$

$$weights+ = (\mu \times velocity\_pre) + ((1 + \mu) \times velocity) \tag{6}$$

RMSProp:

$$cache = (decay \times cache) + (1 - decay) \times gradient^2 \tag{7}$$

$$weight- = (\eta/(\sqrt{cache} + \epsilon)) \times gradient \tag{8}$$

Adam:

$$term\_1 = (\beta_1 \times term\_1) + ((1 - \beta_1) \times gradient), \tag{9}$$

$$term\_2 = (\beta_2 \times term\_2) + ((1 - \beta_2) \times gradient^2), \tag{10}$$

$$weights- = \eta \times term\_1/(\sqrt{(term\_2 + \epsilon)}) \tag{11}$$

The experiments were set up to compare the performance of SHO-tuned EXAMM with baseline EXAMM, which utilizes static values of training hyperparameters. Additionally, the results of SHO-tuned EXAMM were compared to EXAMM with hand-tuned best-found hyperparameters to determine how effective the method is. The experiments were conducted using three weight update methods: Adam, SGD with Nesterov momentum, and RMSProp, on c172 flight and wind turbine datasets. A range of diverse hyperparameter values were tested, resulting in a total of 30 experiments for Nesterov, 60 experiments for RMSProp, and 80 experiments for Adam. The specific values tested are presented in Table 2.

All other hyperparameters were held fixed the same among experiment groups and control groups. EXAMM uses 10 islands with each having an island capacity of 10. Each EXAMM run generates 10,000 genomes. And each experiment is run for 20 repeats.

These experiments were run on the research computing cluster of Rochester Institute of Technology [38]. All experiments are run using 16 CPU cores and 8GB RAM.

5.3 SHO Tuned EXAMM Performance

Figure 1 shows the convergence process of the evolutionary process on all three weight update methods and two datasets. For each of the weight update methods, we tested different combinations of pre-defined hyperparameter values. The plots visualize a range of selected experiments, which include the optimal fixed parameters compared to results from SHO-tuned EXAMM. In each plot, the colored range shows the minimum and maximum validation mean squared error (MSE) over 20 repeated runs, and the line in the middle shows the average validation MSE over 20 repeated runs. From the plots, we can see that SHO-tuned EXAMM generally converges faster and also has better performance.

Tables 4, 5 and 6 show the comparison using pre-defined hyperparameters versus SHO for Nastrov, RMSProp and Adam weight update methods on the C172 and wind datasets across all experiments. The average global best MSE column in these tables presents the average global best validation MSE at the end of evolution, across the 20 repeated runs. The values marked as bold indicate where SHO global best MSE outperforms the control groups. The tables also show the Mann–Whitney U test p-values comparing using pre-defined hyperparameter values to using SHO for optimizing the hyperparameters, comparing the best global best validation MSE between SHO and fixed hyperparameter EXAMM. p-values in bold represent statistically significant differences with $\alpha = 0.1$, showing that using SHO for training hyperparameter optimization performs statistically different than using pre-defined values.

η	μ	C172		Wind	
		p-values	Avg Global Best MSE	p-values	Avg Global Best MSE
0.0001	0.9	**7E-8**	**0.003314**	**7E-8**	**0.006205**
	0.95	**7E-8**	**0.003117**	**7E-8**	**0.005942**
	0.99	**7E-8**	**0.002544**	**7E-8**	**0.004384**
0.001	0.9	**7E-8**	**0.001865**	**7E-8**	**0.003856**
	0.95	**7E-8**	**0.001547**	**7E-8**	**0.003558**
	0.99	**7E-8**	**0.001374**	**7E-8**	**0.003447**
0.01	0.9	**2E-7**	**0.000287**	**7E-8**	**0.003070**
	0.95	**1E-5**	**0.000264**	**1E-7**	**0.003029**
	0.99	**0.0001**	**0.000228**	**0.0066**	**0.003026**
0.02	0.9	**0.0004**	**0.000244**	**4E-7**	**0.003025**
	0.95	0.1719	0.000222	**0.0256**	**0.003009**
	0.99	0.3941	0.000220	**2E-5**	0.002994
0.05	0.9	**0.0239**	0.000225	0.3941	0.002998
	0.95	**3E-6**	0.000216	**1E-5**	0.002988
	0.99	**2E-5**	0.000220	**1E-6**	0.002995

Table 4: SHO vs. Fixed Hyperparameters Significance Testing results and Avg Global Best Performance for Nestrov Momentum

η	decay	ϵ	C172		Wind	
			p-values	Avg Global Best MSE	p-values	Avg Global Best MSE
0.0001	0.85	1E-9	**7E-8**	**0.002221**	**7E-8**	**0.004083**
		1E-8	**7E-8**	**0.002025**	**7E-8**	**0.004202**
	0.9	1E-9	**7E-8**	**0.002222**	**7E-8**	**0.004233**
		1E-8	**7E-8**	**0.001971**	**7E-8**	**0.004566**
	0.95	1E-9	**7E-8**	**0.002111**	**7E-8**	**0.004586**
		1E-8	**7E-8**	**0.001872**	**7E-8**	**0.004598**
0.001	0.85	1E-9	0.2977	0.000328	**9E-7**	**0.003099**
		1E-8	0.2732	0.000306	**1E-7**	**0.003076**
	0.9	1E-9	0.9461	0.000288	**2E-7**	**0.003104**
		1E-8	**0.0909**	0.000302	**7E-7**	**0.003101**
	0.95	1E-9	0.8392	0.000316	**5E-6**	**0.003099**
		1E-8	**0.0962**	0.000265	**3E-7**	0.003071
0.01	0.85	1E-9	0.6168	**0.000408**	0.1478	0.003074
		1E-8	0.1404	**0.000400**	0.5979	0.003071
	0.9	1E-9	0.5249	**0.000405**	**0.0361**	**0.003086**
		1E-8	**0.0721**	**0.000414**	0.3369	0.003073
	0.95	1E-9	**0.0123**	0.000353	**0.0531**	**0.003097**
		1E-8	0.2616	0.000370	0.3235	0.003073
0.02	0.85	1E-9	**0.0721**	**0.000444**	**0.0679**	0.003064
		1E-8	0.3942	**0.000421**	0.1636	0.003073
	0.9	1E-9	**0.0026**	**0.000437**	0.3793	0.003069
		1E-8	**0.0315**	**0.000473**	0.9461	**0.003090**
	0.95	1E-9	**0.0721**	**0.000469**	0.3507	0.003053
		1E-8	**0.0721**	**0.000446**	0.5250	0.003064
0.05	0.85	1E-9	**0.0002**	**0.000550**	0.1988	0.003075
		1E-8	**2E-5**	**0.000560**	0.9031	**0.003077**
	0.9	1E-9	**0.0003**	**0.000487**	0.4093	0.003069
		1E-8	**0.0003**	**0.000476**	0.7972	**0.003081**
	0.95	1E-9	**1E-5**	**0.000602**	0.1556	**0.003080**
		1E-8	**1E-6**	**0.000590**	0.7557	**0.003079**

Table 5: SHO vs. Control Group Significance Testing results and Avg Global Best Performance for RMSProp

η	β_1	β_1	ϵ	C172		Wind	
				p-values	Avg Global Best MSE	p-values	Avg Global Best MSE
0.0001	0.9	0.9	1E-9	**7E-8**	**0.002504**	**7E-8**	**0.004046**
			1E-8	**7E-8**	**0.002221**	**7E-8**	**0.004146**
		0.99	1E-9	**7E-8**	**0.001830**	**7E-8**	**0.003837**
			1E-8	**7E-8**	**0.001876**	**7E-8**	**0.004049**
	0.99	0.9	1E-9	**7E-8**	**0.002794**	**7E-8**	**0.004923**
			1E-8	**7E-8**	**0.003015**	**7E-8**	**0.005537**
		0.99	1E-9	**7E-8**	**0.002832**	**7E-8**	**0.005597**
			1E-8	**7E-8**	**0.002841**	**7E-8**	**0.005029**
0.001	0.9	0.9	1E-9	**7E-8**	**0.000277**	**7E-8**	**0.003031**
			1E-8	**7E-8**	**0.000246**	**7E-8**	**0.003033**
		0.99	1E-9	**7E-8**	0.000235	**7E-8**	**0.003016**
			1E-8	**7E-8**	**0.000276**	**7E-8**	**0.003012**
	0.99	0.9	1E-9	**7E-8**	**0.001269**	**7E-8**	**0.003312**
			1E-8	**7E-8**	**0.001107**	**7E-8**	**0.003292**
		0.99	1E-9	**7E-8**	**0.000480**	**7E-8**	**0.003181**
			1E-8	**7E-8**	**0.000542**	**7E-8**	**0.003205**
0.01	0.9	0.9	1E-9	0.6168	0.000233	0.5249	0.002973
			1E-8	0.6949	0.000236	0.1895	0.002971
		0.99	1E-9	**0.0033**	0.000236	0.6359	0.002961
			1E-8	0.4094	0.000238	0.1988	0.002954
	0.99	0.9	1E-9	**7E-8**	**0.000246**	**7E-8**	**0.003036**
			1E-8	**7E-8**	**0.000259**	**7E-8**	**0.003038**
		0.99	1E-9	**7E-8**	0.000237	**1E-5**	**0.003009**
			1E-8	**7E-8**	**0.000245**	**1E-7**	**0.003013**
0.02	0.9	0.9	1E-9	0.9246	0.000242	0.2393	0.002990
			1E-8	0.9031	**0.000248**	0.7764	0.002985
		0.99	1E-9	0.5979	**0.000252**	0.6168	**0.002994**
			1E-8	0.6750	**0.000249**	0.3235	0.002982
	0.99	0.9	1E-9	**1E-7**	0.000241	**7E-8**	**0.003043**
			1E-8	**7E-8**	0.000237	**7E-8**	**0.003042**
		0.99	1E-9	**7E-8**	**0.000248**	**1E-6**	**0.003023**
			1E-8	**1E-6**	**0.000246**	**5E-7**	**0.002998**
0.05	0.9	0.9	1E-9	**1E-6**	**0.000274**	**0.0123**	**0.003008**
			1E-8	**1E-6**	**0.000280**	**0.0411**	**0.002999**
		0.99	1E-9	**7E-8**	**0.000299**	0.9899	0.002985
			1E-8	**7E-8**	**0.000309**	0.7557	**0.002999**
	0.99	0.9	1E-9	**1E-7**	**0.000250**	**9E-8**	**0.003065**
			1E-8	**7E-8**	**0.000256**	**7E-8**	**0.003078**
		0.99	1E-9	**2E-7**	**0.000263**	**7E-8**	**0.003061**
			1E-8	**2E-7**	**0.000267**	**2E-7**	**0.003034**

Table 6: SHO vs. Control Group Significance Testing results and Avg Global Best Performance for Adam

Table 3 shows best-found hyperparameters from SHO-tuned EXAMM for the different weight update methods on the C172 and the wind datasets. From these results, we can see that while there are some fixed hyperparameter settings that perform slightly faster or close to SHO, SHO still converges to the same best-found RNN performance. Additionally, they show that SHO can statistically improve the performance of EXAMM over using pre-defined values. The results generally show overall statistical significance for improved performance except for the experiments the fixed hyperparameters were close to the optimal hyperparameters.

Additionally, it is worth noting that according to our results, for the same experiment configuration, and using the same weight update method, the optimal training hyperparameters are different for different datasets. This becomes more obvious when the weight update methods have multiple hyperparameters to optimize, such as RMSProp or Adam. This highlights the importance of incorporating SHO into EXAMM or other NE/NAS algorithms, as hyperparameters need to be individually tuned to different datasets, and SHO can accomplish this automatically.

5.4 SHO Convergence

	85%		90%		95%	
	C172	Wind	C172	Wind	C172	Wind
Nesterov	4395	434	5213	594	6452	1306
RMSProp	6604	888	6853	1194	7337	2986
Adam	2848	215	4342	396	5636	837

Table 7: SHO Convergence Table

η	μ	C172			Wind		
		85%	90%	95%	85%	90%	95%
0.0001	0.9	-	-	-	-	-	-
	0.95	-	-	-	-	-	-
	0.99	-	-	-	-	-	-
0.001	0.9	-	-	-	**9132**	**9763**	**9951**
	0.95	-	-	-	**7064**	**8143**	**9301**
	0.99	-	-	-	**7286**	**8584**	**9424**
0.01	0.9	**8535**	**8976**	**9405**	**2138**	**2955**	**4984**
	0.95	**7794**	**8276**	**8843**	**987**	**1366**	**2674**
	0.99	**5812**	**6769**	**7509**	**444**	**785**	**1875**
0.02	0.9	**6819**	**7436**	**8391**	**824**	**1258**	**2392**
	0.95	4345	4987	6036	**508**	**838**	**1562**
	0.99	4044	4522	5540	191	352	775
0.05	0.9	3163	3905	5847	388	**660**	1153
	0.95	2494	3055	4026	219	329	668
	0.99	2479	2936	3888	107	169	429

Table 8: Nesterov Control Group Converge Table

When we investigate how well hyperparameter optimization works, besides looking at the overall performance of the algorithm, we also need to determine if each of the hyperparameters that is being optimized is actually converging to optimal values. Figures 2, 3 and 4 show the convergence progress of each training hyperparameter. The plots show the value range of between minimum and maximum parameters across all repeated EXAMM run populations (fitness or hyperparameter value), and the solid line in the middle shows the average value. From these plots, we can observe that all hyperparameters are converging, as well as the EXAMM best validation MSE.

From Figure 1 we observe that SHO not only has better performance in finding the global best genome, it generally converges faster than most of the fixed parameter experiments. To quantify this observation, we calculated how many generated genomes it takes to reach 85%, 90%, and 95% of the overall best-performing genome's performance. Table 7 shows for different weight update methods and datasets, how many generated genomes it takes to reach 85%, 90%, and 95% of the best performance on average over 20 repeated runs. To compare to the fixed hyperparameter experiments, Tables 8, 9 and 10 shows how long each fixed hyperparameter experiment took to reach 85%, 90%, and 95% of SHO's performance. As shown in Figure 1, not all the fixed hyperparameter experiments resulted in well-performing best genomes, getting stuck in local optima. For the experiments that could not make it close to the best performance, results are marked as "-" in the tables. Overall comparing the speed of convergence of the fixed hyperparameter

η	decay	ϵ	C172			Wind		
			85%	90%	95%	85%	90%	95%
0.0001	0.85	1E-9	-	-	-	**7832**	**8142**	**8142**
		1E-8	-	-	-	**7563**	**7969**	**8160**
	0.9	1E-9	-	-	-	**8696**	**8750**	**8771**
		1E-8	-	-	-	**7603**	**7791**	**7849**
	0.95	1E-9	-	-	-	**7783**	**7980**	**7980**
		1E-8	-	-	-	**7916**	**8245**	**8303**
0.001	0.85	1E-9	4372	4597	4799	**1347**	**2000**	**3354**
		1E-8	4279	4510	4749	**1478**	**2128**	**4130**
	0.9	1E-9	4086	4226	4411	**1353**	**2042**	**3751**
		1E-8	4658	5037	5361	**1489**	**2254**	**4357**
	0.95	1E-9	4189	4409	4636	**1260**	**2013**	**4029**
		1E-8	2942	3141	3379	**1354**	**1832**	**3262**
0.01	0.85	1E-9	**6652**	**6996**	**7919**	**938**	**1613**	2928
		1E-8	**7211**	**7672**	**8159**	816	1089	2485
	0.9	1E-9	**7661**	**8064**	**8445**	**1020**	**1506**	2821
		1E-8	**7861**	**8094**	**8387**	**973**	**1625**	**3233**
	0.95	1E-9	5252	5816	6555	**1100**	**1700**	**3202**
		1E-8	5621	5995	6165	**1087**	**1819**	**3023**
0.02	0.85	1E-9	**7541**	**7740**	**8436**	611	936	2194
		1E-8	**7311**	**7897**	**8154**	706	1060	2302
	0.9	1E-9	**8394**	**8518**	**8666**	699	**1283**	2549
		1E-8	**7780**	**7957**	**8902**	702	1089	2789
	0.95	1E-9	**8183**	**8471**	**8612**	717	899	2395
		1E-8	**7872**	**8151**	**8331**	755	1025	2222
0.05	0.85	1E-9	**8471**	**8579**	**8579**	746	971	1910
		1E-8	**8871**	**9240**	**9278**	802	**1449**	2454
	0.9	1E-9	**8432**	**8814**	**8865**	705	1093	1943
		1E-8	**8980**	**9003**	**9242**	656	1120	2704
	0.95	1E-9	**8618**	**8986**	**9586**	**957**	**1653**	**3213**
		1E-8	**9042**	**9390**	**9576**	**988**	**1476**	2705

Table 9: RMSProp Control Group Converge Table

η	β_1	β_1	ϵ	C172			Wind		
				85%	90%	95%	85%	90%	95%
0.0001	0.9	0.9	1E-9	-	-	-	**7352**	**7684**	**7684**
			1E-8	-	-	-	**7515**	**7604**	**7687**
		0.99	1E-9	-	-	-	**7028**	**7651**	**7778**
			1E-8	-	-	-	**6665**	**7036**	**7257**
	0.99	0.9	1E-9	-	-	-	**7337**	**7659**	**7659**
			1E-8	-	-	-	**7573**	**7613**	**7613**
		0.99	1E-9	-	-	-	**6146**	**6146**	**6146**
			1E-8	-	-	-	**7141**	**7141**	**7141**
0.001	0.9	0.9	1E-9	**7016**	**7493**	**8066**	**1280**	**2056**	**3893**
			1E-8	**5688**	**6201**	**7092**	**1326**	**2300**	**4547**
		0.99	1E-9	**4647**	**5102**	**5835**	**1280**	**1893**	**3747**
			1E-8	**5010**	**5553**	**6341**	**1430**	**2398**	**3893**
	0.99	0.9	1E-9	**9858**	**9858**	**9858**	**6524**	**7715**	**8992**
			1E-8	**9893**	**9893**	**9893**	**5692**	**7480**	**9193**
		0.99	1E-9	**9876**	**9941**	**9941**	**4091**	**5681**	**8275**
			1E-8	**9917**	**9917**	**9917**	**4654**	**6566**	**8445**
0.01	0.9	0.9	1E-9	2122	2766	3827	**240**	**437**	**1060**
			1E-8	2175	2917	4163	**231**	**399**	**1120**
		0.99	1E-9	1847	2500	3709	**281**	**400**	**910**
			1E-8	2056	2773	4772	**233**	**434**	801
	0.99	0.9	1E-9	**5250**	**6048**	**7192**	**1073**	**1949**	**4287**
			1E-8	**6763**	**7797**	**8705**	**1055**	**1798**	**4805**
		0.99	1E-9	**3992**	**4726**	**5775**	**769**	**1225**	**3359**
			1E-8	**4952**	**5960**	**7359**	**658**	**1240**	**3369**
0.02	0.9	0.9	1E-9	2364	3479	5471	**225**	374	**1026**
			1E-8	**3393**	**4390**	**6313**	181	341	804
		0.99	1E-9	**3335**	**4940**	**7080**	**230**	**397**	828
			1E-8	**3227**	4191	**7434**	**225**	392	**864**
	0.99	0.9	1E-9	**4595**	**5269**	**6247**	**833**	**1688**	**4040**
			1E-8	**4441**	**4982**	**6369**	**971**	**1875**	**4749**
		0.99	1E-9	**4871**	**5752**	**6844**	**500**	**1064**	**2401**
			1E-8	**4083**	**5372**	**6940**	**603**	**1213**	**2369**
0.05	0.9	0.9	1E-9	**7316**	**8228**	**9534**	**327**	**522**	**1061**
			1E-8	**8719**	**9260**	**9707**	**260**	**535**	**1323**
		0.99	1E-9	**8306**	**8515**	**9368**	193	308	**967**
			1E-8	**8807**	**9267**	**9388**	196	376	**889**
	0.99	0.9	1E-9	**5292**	**6204**	**6883**	**1034**	**2146**	**4898**
			1E-8	**5654**	**6930**	**7993**	**1270**	**2347**	**5105**
		0.99	1E-9	**5429**	**7190**	**8067**	**510**	**1216**	**4522**
			1E-8	**6639**	**7785**	**8744**	**644**	**1391**	**3722**

Table 10: Adam Control Group Converge Table

experiments to SHO, we found that SHO generally converges faster than using pre-defined hyperparameter values.

6 Discussion

This paper investigates using simplex hyperparameter optimization (SHO) for online adaptation of neural network training hyperparameters as part of the Evolutionary eXploration of Augmenting Memory Models (EXAMM) neuroevolution algorithm. Prior work evaluated SHO for evolution of convolutional neural networks and traditional stochastic gradient descent (SGD) with nesterov momentum, and this work expands on that by applying SHO to evolution of recurrent neural networks and the hyperparameters of more modern optimization processes such as Adam and RMSProp.

SHO was implemented with EXAMM and was tested on the problem of evolving RNNs for time series forecasting on two real-world large scale high dimensional datasets, one in aviation, Cessna 172 flight recorder data, and another in power systems, wind turbine data from ENGIE's La Haute Borne open data wind farm. Experiments show that across all data sets and optimization methods evaluated, SHO could find the optimal training hyperparameters, outperforming holding hyperparameters fixed. Results also show that SHO converges faster than using pre-defined hyperparameter values. It speeds up the neuroevolution process while also improving the validation MSE of the global best genomes (RNNs) that EXAMM evolved.

SHO allows EXAMM to be more easily adaptable to different datasets, as it can optimize both neural network architectures while finding optimal hyperparameters, with SHO adding an additional benefit of being able to adapt to distribution shift (as more evolved neural networks require different hyperparameters), allowing EXAMM to keep learning during the neuroevolution process. SHO is also a generic strategy, as it independently co-evolves hyperparameters based on a neuroevolution algorithm's population – so it can easily be utilized for any population based neural architecture search (NAS) method. We find these results show that SHO is a strong algorithm for automating the process of determining hyperparameters in neural architecture search.

Acknowledgements

This work has been partially supported by the Federal Aviation Administration National General Aviation Flight Information Database (NGAFID) award and by the National Science Foundation under Grant Number 2225354. 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 National Science Foundation or the Federal Aviation Administration. We also thank RIT's Research Computing team for their assistance and support.

References

[1] Abdullah Alanazi. 2022. Using machine learning for healthcare challenges and opportunities. *Informatics in Medicine Unlocked* (2022), 100924.

[2] Hussain Alibrahim and Simone A Ludwig. 2021. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1551–1559.

[3] Moriah Ariely, Tanya Nazaretsky, and Giora Alexandron. 2022. Machine learning and Hebrew NLP for automated assessment of open-ended questions in biology. *International journal of artificial intelligence in education* (2022), 1–34.

[4] Habeeb Balogun, Hafiz Alaka, and Christian Nnaemeka Egwim. 2021. Boruta-grid-search least square support vector machine for NO2 pollution prediction using big data analytics and IoT emission sensors. *Applied Computing and Informatics* ahead-of-print (2021).

[5] Thomas Bartz-Beielstein, Jürgen Branke, Jörn Mehnen, and Olaf Mersmann. 2014. Evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4, 3 (2014), 178–195.

[6] Dilyara Baymurzina, Eugene Golikov, and Mikhail Burtsev. 2022. A review of neural architecture search. *Neurocomputing* 474 (2022), 82–93.

[7] Daniel Mesafint Belete and Manjaiah D Huchaiah. 2022. Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results. *International Journal of Computers and Applications* 44, 9 (2022), 875–886.

[8] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

[9] Aleksandar Botev, Guy Lever, and David Barber. 2017. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In *2017 International joint conference on neural networks (IJCNN)*. IEEE, 1899–1903.

[10] Alebachew Chiche and Betselot Yitagesu. 2022. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *Journal of Big Data* 9, 1 (2022), 1–25.

[11] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. 2020. Fbnetv3: Joint architecture-recipe search using neural acquisition function. *arXiv preprint arXiv:2006.02049* 1, 2 (2020), 3.

[12] Travis Desell. 2017. 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, 19–28.

[13] Travis Desell, Boleslaw Szymanski, and Carlos Varela. 2008. 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*. 921–928.

[14] Xuanyi Dong, Mingxing Tan, Adams Wei Yu, Daiyi Peng, Bogdan Gabrys, and Quoc V Le. 2020. Autohas: Differentiable hyper-parameter and architecture search. *arXiv preprint arXiv:2006.03656* 4, 5 (2020).

[15] Romain Egele, Prasanna Balaprakash, Isabelle Guyon, Venkatram Vishwanath, Fangfang Xia, Rick Stevens, and Zhengying Liu. 2021. AgEBO-tabular: joint neural architecture and hyperparameter search with autotuned data-parallel training for tabular data. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–14.

[16] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, Kevin Leyton-Brown, et al. 2013. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, Vol. 10.

[17] AbdElRahman ElSaid, Steven Benson, Shuchita Patwardhan, David Stadem, and Travis Desell. 2019. Evolving recurrent neural networks for time series data prediction of coal plant parameters. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 488–503.

[18] AbdElRahman ElSaid, Joshua Karnas, Zimeng Lyu, Daniel Krutz, Alexander G Ororbia, and Travis Desell. 2020. Neuro-Evolutionary Transfer Learning through Structural Adaptation. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 610–625.

[19] AbdElRahman ElSaid, Joshua Karns, Zimeng Lyu, Daniel Krutz, Alexander Ororbia, and Travis Desell. 2020. Improving neuroevolutionary transfer learning of deep recurrent neural networks through network-aware adaptation. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 315–323.

[20] Eyad Elyan, Pattaramon Vuttipittayamongkol, Pamela Johnston, Kyle Martin, Kyle McPherson, Chrisina Jayne, Mostafa Kamal Sarker, et al. 2022. Computer vision and machine learning for medical image analysis: recent advances, challenges, and way forward. *Artificial Intelligence Surgery* 2 (2022).

[21] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*. PMLR, 1437–1446.

[22] Yesmina Jaafra, Jean Luc Laurent, Aline Deruyver, and Mohamed Saber Naceur. 2019. Reinforcement learning for neural architecture search: A review. *Image and Vision Computing* 89 (2019), 57–66.

[23] Xia Jiang and Chuhan Xu. 2022. Deep Learning and Machine Learning with Grid Search to Predict Later Occurrence of Breast Cancer Metastasis Using Clinical Data. *Journal of Clinical Medicine* 11, 19 (2022), 5772.

[24] S Vinila Jinny and Yash Vijay Mate. 2021. Early prediction model for coronary heart disease using genetic algorithms, hyper-parameter optimization and machine learning techniques. *Health and Technology* 11 (2021), 63–73.

[25] Tinu Theckel Joy, Santu Rana, Sunil Gupta, and Svetha Venkatesh. 2016. Hyperparameter tuning for big data using Bayesian optimisation. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2574–2579.

[26] Asharul Islam Khan and Salim Al-Habsi. 2020. Machine learning in computer vision. *Procedia Computer Science* 167 (2020), 1444–1451.

[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. 2017. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial intelligence and statistics*. PMLR, 528–536.

[29] Liam Li, Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. 2020. Geometry-aware gradient algorithms for neural architecture search. *arXiv preprint arXiv:2004.07802* (2020).

[30] Zimeng Lyu, Shuchita Patwardhan, David Stadem, James Langfeld, Steve Benson, Seth Thoelke, and Travis Desell. 2021. Neuroevolution of recurrent neural networks for time series forecasting of coal-fired power plant operating parameters. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1735–1743.

[31] Dougal Maclaurin, David Duvenaud, and Ryan Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*. PMLR, 2113–2122.

[32] Rafael G Mantovani, André LD Rossi, Joaquin Vanschoren, Bernd Bischl, and André CPLF De Carvalho. 2015. Effectiveness of random search in SVM hyper-parameter tuning. In *2015 International Joint Conference on Neural Networks (IJCNN)*. Ieee, 1–8.

[33] John A Nelder and Roger Mead. 1965. A simplex method for function minimization. *The computer journal* 7, 4 (1965), 308–313.

[34] Alexander Ororbia, AbdElRahman ElSaid, and Travis Desell. 2019. Investigating recurrent neural network memory structures using neuro-evolution. In *Proceedings of the genetic and evolutionary computation conference*. 446–455.

[35] Mercy Prasanna Ranjit, Gopinath Ganapathy, Kalaivani Sridhar, and Vikram Arumugham. 2019. Efficient deep learning hyperparameter tuning using cloud infrastructure: Intelligent distributed hyperparameter tuning with Bayesian optimization in the cloud. In *2019 IEEE 12th international conference on cloud computing (CLOUD)*. IEEE, 520–522.

[36] Romil Rawat, Yagya Nath Rimal, P William, Snehil Dahima, Sonali Gupta, and K Sakthidasan Sankaran. 2022. Malware Threat Affecting Financial Organization Analysis Using Machine Learning Approach. *International Journal of Information Technology and Web Engineering (IJITWE)* 17, 1 (2022), 1–20.

[37] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2021. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–34.

[38] Rochester Institute of Technology. 2022. Research Computing Services. https://doi.org/10.34788/0S3G-QD15

[39] BH Shekar and Guesh Dagnew. 2019. Grid search-based hyperparameter tuning and classification of microarray cancer data. In *2019 second international conference on advanced computational and communication paradigms (ICACCP)*. IEEE, 1–8.

[40] Iwan Syarif, Adam Prugel-Bennett, and Gary Wills. 2016. SVM parameter optimization using grid search and genetic algorithm to improve classification performance. *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 14, 4 (2016), 1502–1509.

[41] Tijmen Tieleman and Geoffrey Hinton. 2012. Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *COURSERA Neural Networks Mach. Learn* 17 (2012).

[42] Ananto Setyo Wicaksono and Ahmad Afif Supianto. 2018. Hyper parameter optimization using genetic algorithm on machine learning methods for online news popularity prediction. *International Journal of Advanced Computer Science and Applications* 9, 12 (2018).

[43] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.

[44] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. 2015. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the workshop on machine learning in high-performance computing environments*. 1–5.

[45] Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. 2018. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. *arXiv preprint arXiv:1807.06906* (2018).

[46] Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. 2019. Bayesnas: A bayesian approach for neural architecture search. In *International conference on machine learning*. PMLR, 7603–7613.