Biologically-Inspired Homeostasis for Neuroevolution: Alternating Growth and Pruning Phases

Abhishek Singh[1], Zimeng Lyu[2], and Travis Desell[1]

[1] Golisano College of Computing and Information Sciences, Rochester Institute of Technology, Rochester, NY 14623, USA
as3485@rit.edu
[2] Kean University, 1000 Morris Ave Union, NJ 07083
zimeng.lyu@kean.edu

Abstract. This study investigates the effect of alternating growth and pruning phases on the evolution of neural architectures using the Evolutionary eXploration of Augmenting Memory Models (EXAMM). Our objective was to determine if this structured evolutionary process could guide the search towards more compact models without sacrificing predictive performance. We compared EXAMM-evolved architectures having traditional and five modern memory cells (simple, UGRNN, MGU, GRU, Delta-RNN, and LSTM) on two distinct time-series datasets: aviation flight recorder data and wind turbine sensor data. The alternating phases were controlled by enabling growth-promoting or growth-reducing mutation operations based on trigger frequencies of 50, 100, or 200 generated genomes, resulting in a 3×3 set of nine unique grow-shrink configurations. Results show that while there was no statistical difference in terms of model performance to a baseline without phases, the experiments with higher pruning to growth ratios had statistically significant decreases in model size, making this a simple but effective method for controlling network complexity.

Keywords: Time Series Forecasting · Neural Architecture Search · Neuroevolution · Neuro-inspired Computing

1 Introduction

Neural Architecture Search (NAS) and neuroevolution (NE) methods have successfully automated the design of neural networks [35, 29, 14]. Neuroevolution provides a robust, gradient-free optimization strategy that enables broad exploration of architectural topologies and weights, including parts difficult to capture with traditional NAS frameworks. In approaches such as NEAT [38], which begin from minimal networks and gradually increase structural complexity, or population-based methods that evolve genomes through iterative mutation and selection [31, 28], architectures often grow as the search favors small improvements in accuracy. This process can leads to overly complex or "bloated"

models, where minimal performance gains are obtained at the cost of substantial increases in network size, as well as extending search and evaluation times. The resulting architectures are difficult to deploy on resource-constrained devices, and their inflated structure can limit training efficiency and obscure the interpretability of the model.

In neuroevolutionary NAS, large-scale structural pruning is not typically part of the search process. Instead of removing substantial portions of a network as seen in many pruning-based NAS methods [21] [25], neuroevolution mainly relies on fitness-driven selection and standard regularization to discourage overly large or unproductive architectures. These mechanisms can limit excessive growth indirectly by eliminating weaker, more complex genomes over time, but they do not directly trim large branches or modules once they appear. As a result, structural complexity can accumulate across generations even when it provides little benefit, underscoring the need for approaches that introduce more explicit control over network size during evolution.

Biological neural systems offer a compelling perspective for addressing this challenge. Neural circuits continually balance learning capacity and resource efficiency through dynamic restructuring of connectivity [41, 40]. During wakeful learning, frequently co-active neurons strengthen their synapses to encode new information, expanding local connectivity [2, 42]. Later, global regulatory processes scale back or prune weaker and redundant synapses to maintain metabolic stability and prevent runaway growth. This refine-and-stabilize cycle is not limited to early development; recent studies show that REM sleep selectively downscales synapses potentiated while awake, effectively optimizing circuit structure on a daily basis [23, 33]. The biological pattern is thus one of expand-while-learning followed by shrink-to-stabilize—a principle that maps naturally onto the problem of architectural bloat in neuroevolution.

Inspired by this dynamic, we introduce an alternating grow–shrink schedule for neuroevolutionary NAS. Instead of allowing growth and reduction to occur in an uncoordinated, mutation-driven manner, we explicitly separate these processes into two phases: a growth phase, in which only expansion-oriented mutations are permitted, and a pruning phase, in which only size-reducing mutations occur. This on–off structuring allows evolution to explore richer architectures during growth intervals while periodically removing unnecessary components during shrink intervals, helping counter bloat without restricting the search space. Building on this idea, we introduce a grow–shrink training schedule that alternates between periods of structural expansion and periods of targeted reduction within the Evolutionary eXploration of Augmenting Memory Models (EXAMM) algorithm. In particular, we raise the following research questions of: **(RQ1) Does adding explicit growth and pruning phases affect model accuracy?** and **(RQ2) Do these growth and shrink phases lead to smaller or larger evolved networks?**

To explore these questions, we evaluate multiple grow–shrink configurations and compare them to a baseline. The alternating schedule allows evolution to explore larger architectural variations during growth intervals while periodically

removing redundant or unproductive structure during shrink intervals. This structured dynamic helps counter bloat without restricting the search space. Across two challenging time-series prediction tasks, we find that while this does not have a statistical effect on predictive performance, it does produce notably smaller and more efficient networks. These results suggest that regulating evolutionary dynamics through coordinated growth and shrink phases offers a simple yet effective way to balance exploration and complexity.

2 Related Work

Pruning Mechanisms: The removal of redundant parameters has a long history as a method for network compression and efficiency. Classic post-training approaches such as deep compression demonstrated that a large fraction of individual weights in over-parameterized networks could be removed and then retrained with minimal to no loss in accuracy [15]. This concept was extended beyond individual weights to structured pruning (e.g., removing entire channels) [25] and to dynamic methods that find sparse networks "from scratch" during training [7, 24]. This field was significantly advanced by the Lottery Ticket Hypothesis, which posited that dense networks contain sparse, trainable subnetworks ("winning tickets") from the outset [13]. Subsequent work showed these tickets could be found reliably [12], confirming that the pervasive redundancy enabling pruning is a fundamental property of over-parameterization [30].

This understanding led to dynamic methods that prune *during* or *before* training. These approaches find sparse subnetworks "from scratch" [7] or use evolutionary-inspired dynamics to manage sparsity throughout training [24]. Modern pruning criteria have become more sophisticated, moving beyond simple magnitude to include information-theoretic measures [44] or hardware-aware objectives like latency-saliency budgets [37].

Growth Mechanisms: Methods for growing networks are split between two dominant paradigms: population-based neuroevolution (NE) and single-network dynamic sparse training (DST). Historically, NE has modeled architectural design as a growth-oriented process, starting with minimal networks and expanding them. This is inspired by the biological principle of Hebbian plasticity, where connections strengthen with use [17]. The most notable example, NEAT, demonstrated that networks could evolve by starting minimally and progressively adding structural complexity (nodes and connections) through mutations over generations [38]. This growth-centric paradigm persists in modern evolutionary Neural Architecture Search (NAS), where mutation operators expand architectures by adding new pathways, blocks, or depth [28, 31]. While effective for exploration, this unconstrained growth often leads to "bloat," where architectures accumulate complexity without corresponding performance gains.

In contrast, DST methods "sculpt" a single network's sparse topology *during* a single training run or search within a large scale network superstructure. These methods use an iterative cycle of pruning and growing. For example, Sparse

Evolutionary Training (SET) prunes the smallest-magnitude connections and then *grows* an equal number of new connections randomly [24]. Other methods use more complex criteria, such as growing new connections based on gradient momentum to explore promising areas of the topology [7] or nature inspired strategies such as those based on ant colonies [10, 9, 11]. In these methods, growth is not an evolutionary operator but a mechanism to escape local minima and refine the sparse connectivity.

Integrating Pruning and Growth: Given the rapid expansion of search spaces, many NAS methods now integrate pruning as a control mechanism rather than just a post-processing step. Some use pruning to remove low-value operations during the search, allowing the process to focus on more promising variations [5]. Others dynamically prune the sampling distribution of operators to tighten the search as it progresses [45]. Pruning and NAS are also combined to meet specific hardware or latency constraints [43, 21] and have been extended to new domains like large language models [20].

Within neuroevolution, some work treats pruning as the central objective, using co-evolution to prune filters layer-by-layer [36] or evolving progressively smaller subnetworks [22]. Other hybrid approaches reformulate the problem entirely. For instance, Transformable Architecture Search (TAS) recasts pruning itself as a learnable architecture search problem, where a controller learns the optimal pruning transformation for each layer to meet a target constraint [8]. More direct hybrids, like G-EvoNAS, first perform a global expansion and then perform pruning [47]. STEERAGE provides a different two-stage methodology, first using a global search to find a base architecture, and then applying a local *grow-and-prune* algorithm to simultaneously optimize the final model's performance and efficiency [16].

This tension between growth and pruning mirrors biological neural systems, which must balance Hebbian-style expansion with homeostatic plasticity to maintain stability [2, 41]. Biological systems often resolve this temporal paradox by separating phases of plasticity (growth) from phases of synaptic renormalization (shrinkage), such as during sleep [42, 33, 23]. The method introduced in this work connects these directions by arranging growth and shrinkage into a scheduled cycle inside neuroevolution. Growth mutations widen the design space at controlled intervals, and pruning mutations apply targeted reduction when the search becomes oversaturated with structure. This alternation keeps exploration intact but reins in the complexity that typically accumulates in evolutionary NAS, allowing the search to maintain expressiveness without drifting toward unnecessary size.

3 Neuroevolution with Synaptic Growth and Pruning Phases

EXAMM [**?**] was chosen as the base method to examine utilizing synaptic growth and pruning phases as it is a modern augmenting topology neuroevolution algorithm (similar to NEAT [38]) which has built in support for easily specifying

how and when to perform mutation and crossover operations. Further, EXAMM has been widely studied as a method for designing efficient recurrent neural networks, with a number of advanced optimizations. It utilizes Lamarckian weight inheritance to significantly reduce the amount of backpropagation required for every generated neural network [26], can evolve deep recurrent connections which improve on or outperform memory cell architectures [6], utilizes a distributed island based strategy with extinction and repopulation events to prevent stagnation of the search process [27], and can co-evolve training hyperparameters while perfroming neural architecture search and training [19, 39].

Similar to NEAT, EXAMM provides a set of edge level mutations to aid in the architecture search process: *add edge*, *add recurrent edge*, *split edge*, *enable edge* and *disable edge*. In addition to these basic mutations, EXAMM also provides novel node level mutations which significantly speed up the search process [**?**]: *add node*, *split node*, *merge node*, *enable node* and *disable node*; as well as a *clone* operation to allow a child genome to further refine its weights starting from the parents weights (as EXAMM is memetic, unlike NEAT). Lastly, as EXAMM utilizes a distributed island approach [27], it also has two crossover operations: *intra-island crossover*, where both parents are selected from the same island, and *inter-island crossover*, where both parents are selected from different islands. By default, the standard baseline EXAMM method performs inter-island crossover 10% of the time, intra-island crossover 20% of the time, and mutation the remaining 70% of the time, where the mutation operation used is selected uniformly at random from the above 11 mutation methods.

This work extends EXAMM by providing support for alternating phases of growth and pruning by dynamically changing which mutation operations are performed during the search process. Operations are classified as either growth, pruning or neutral operations as follows:

- *Add Edge* **(growth)**: adds an edge between two nodes that do not have an edge.
- *Add Recurrent Edge* **(growth)**: adds a recurrent edge with a randomly selected depth between two nodes that do not have an edge.
- *Split Edge* **(growth)**: selects an edge at random, disables it, and adds a node in between the edge's input and output, with two edges connecting the new node to the previous input and output. This increases network size by one enabled edge and one enabled node.
- *Enable Edge* **(growth)**: enables a disabled edge.
- *Disable Edge* **(prune)**: disables an enabled edge.
- *Add Node* **(growth)**: Adds a node at a random bisection of the neural networks graph, and adds a randomly selected number of input edges to that node from nodes deeper in the network, and a randomly selected number of output edges to nodes shallower in the network. This increases the network size by one enabled node and at least two enabled edges.
- *Split Node* **(growth)**: Disables a node and all of its input and output edges, adding two nodes at the same depth in the network, and randomly divides that nodes input and output nodes between the two created nodes, adding

edges between them. If there is only one input or output to the split node, an edge to that node will be added to both new nodes. This increases the number of enabled nodes by one and the number of enabled edges by at 1 or 2 if there is only a single output and/or input node to the split node.
- *Enable Node* **(growth)**: Enables a node and all of its input and output edges.
- *Disable Node* **(prune)**: Disables a node and all of its input and output edges.
- *Merge Node* **(prune)**: Disables two nodes, creating a new node at a depth halfway between the disabled nodes, with edges from the new node to all input and output nodes of the two merged nodes. This reduces the number of enabled nodes by one.
- *Clone* **(neutral)**: Creates a copy of the parent neural network.
- *Intra- and Inter-Island Crossover* **(neutral):** are classified as neural because new genomes generated by crossover keep edges and nodes found in both parents, and randomly keeps edges and nodes from either parent at specified least-fit and best crossover rates. This results in child genomes which are approximately an average size of both parents, which on average leads to neither growth nor pruning.

As EXAMM utilizes an asynchronous distributed computing methodology with steady-state populations (no explicit population generations) for each island, EXAMM was extended with two additional hyperparameters, one which specifies the number of genomes to generate within a growth phase, and the other which specifies the number of genomes to generate within its pruning phase. EXAMM then alternates between growth and pruning phases until a total specified number of genomes have been evaluated. During the growth phase, only growth and neutral operations are allowed, and during the pruning phase only pruning and neutral operations are allowed. Similarly to baseline EXAMM, inter-island crossover is performed at 10%, intra-island crossover is performed at 20% and the mutations allowed during the particular phase are selected uniformly at random the other 70% of the time. By evaluating EXAMM's performance with varying growth and pruning phase lengths, we are able to investigate this work's research questions.

4 Data Sets

This work utilizes two real-world time series benchmark datasets to evaluate how utilizing growth and pruning phases effects EXAMM's neuroevolution process. These datasets have been openly provided through the EXAMM GitHub repository for the purposes of reproducibility and to serve as a valuable resource for the time-series prediction community. Both datasets are multivariate, exhibit non-seasonal characteristics, and feature interdependent parameter recordings, making them challenging targets for prediction.

Aviation Flight Recorder Data (Aviation) The first dataset is sourced from the National General Aviation Flight Information Database (NGAFID) [1]. It comprises recordings from 10 separate flights, with each flight consisting of over an

hour of per-second data recordings across 31 sensor parameters, with identifying information (such as tail number, date/time, and geo-coordinates) removed to protect pilot privacy. The data is provided in its raw, unnormalized state. *Pitch* was selected as the target forecasting parameter, as it is a primary component of the aircraft's attitude and critical for modeling flight stability and control.

Wind Turbine Data (wind) The second dataset is wind turbine engine data collected and made available by ENGIE's La Haute Borne open data windfarm, which was gathered from 2017 to January 2018. This wind dataset is multivariate (containing 22 parameters), non-seasonal, and the parameter recordings are not independent. The wind turbine data consists of readings every 10 minutes from 2017 to January 2018. Average Active Power was selected as the output parameter to forecast for the wind turbine dataset.

5 Results

All experiments for this work utilized standard EXAMM settings, including nodes being generated by random selection from a library of simple neurons with a *tanh* activation function, Δ-RNN units [32], gated recurrent units (GRUs) [3], long short-term memory cells (LSTMs) [18], minimal gated units (MGUs) [46], and update-gate RNN cells (UGRNNs) [4]. Across all experiments, a consistent set of hyperparameters were used, which were selected from prior literature on EXAMM which had manually tuned them. The evolutionary process was configured with 10 islands, each with a population size of 10. Each run had maximum of 10,000 genomes evaluated. For network training, backpropagation (BP) was performed for 10 iterations using the Adam optimizer, with a learning rate of 0.001 and standard library defaults for all other parameters ($\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 1 \times 10^{-8}$). All input data was normalized using the min-max method.

To establish a baseline for each dataset, an experiment with standard EXAMM without growth and pruning phases was performed, where any of the 11 mutations were selected uniformly at random for the entire search. Following this, experiments were performed for nine different configurations, representing a variety of growth and shrink phase lengths. The frequencies to trigger a change between growth and shrink phases were every 50, 100, or 200 generated genomes. This 3x3 design (e.g., G:50/S:50, G:50/S:100, G:50/S:200 ..., G:200/S:200, where G:X represents a growth phase of X generated genomes and S:X represents a shrink phase of X generated genomes) resulted in nine unique configurations. The baseline and each of these nine configurations was repeated 10 times for each of the two datasets to measure statistical significance.

Experiments were run on a high performance computing cluster with a total of 2,304 Intel Skylake CPU cores and 24 TB of RAM running Red Hat Enterprise Linux 9.7. Each experiment utilized 16 compute cores and MPI processes to run the extended version of EXAMM, requiring less than 5GB total RAM.

Table 1: Summary of Mean Square Error(MSE) results for Aviation Flight Recorder and Wind Turbine datasets.

Dataset	Experiment	Min	Avg	Max	Std	P-value
Aviation	Baseline	**0.000203**	0.0002230	0.000236	0.00000931	1
Aviation	Grow 50 Shrink 50	0.000211↑	0.0002211↓	0.000231↓	0.00000627↓	0.5681
Aviation	Grow 50 Shrink 100	0.000213↑	0.0002281↑	0.000291↑	0.00002180↑	0.8203
Aviation	Grow 50 Shrink 200	0.000214↑	0.0002190↓	**0.000224**↓	**0.00000344**↓	0.1715
Aviation	Grow 100 Shrink 50	0.000209↑	0.0002200↓	0.000228↓	0.00000533↓	0.3239
Aviation	Grow 100 Shrink 100	0.000204↑	0.0002218↓	0.000269↑	0.00001720↑	0.4478
Aviation	Grow 100 Shrink 200	0.000206↑	0.0002229↓	0.000247↑	0.00001190↑	0.7912
Aviation	Grow 200 Shrink 50	0.000216↑	0.0002293↑	0.000252↑	0.00001300↑	0.6225
Aviation	Grow 200 Shrink 100	0.000209↑	0.0002195↓	0.000235↓	0.00000749↓	0.2721
Aviation	Grow 200 Shrink 200	0.000205↑	**0.0002176**↓	0.000235↓	0.00000795↓	0.1394
Wind	Baseline	0.002933	0.0030155	0.003073	0.0000532	1
Wind	Grow 50 Shrink 50	0.002940↑	0.0030049↓	0.003070↓	0.0000509↓	0.3843
Wind	Grow 50 Shrink 100	0.002941↑	0.0030112↓	0.003099↑	0.0000473↓	0.7337
Wind	Grow 50 Shrink 200	0.002954↑	0.0030257↑	0.003106↑	0.0000434↓	0.7913
Wind	Grow 100 Shrink 50	0.002936↑	0.0030183↑	0.003068↓	0.0000539↑	0.6499
Wind	Grow 100 Shrink 100	0.002922↓	**0.0029669**↓	0.003082↑	0.0000444↓	0.1123
Wind	Grow 100 Shrink 200	0.002967↑	0.0030253↑	**0.003051**↓	**0.0000253**↓	0.9698
Wind	Grow 200 Shrink 50	0.002923↓	0.0030125↓	0.003087↑	0.0000490↓	0.9097
Wind	Grow 200 Shrink 100	0.002926↓	0.0030028↓	0.003064↓	0.0000374↓	0.5205
Wind	Grow 200 Shrink 200	**0.002903**↓	0.0030011↓	0.003074↑	0.0000594↑	0.6230

5.1 Comparative Analysis of Evolved Architectures

Table 1 summarizes the validation mean squared error (MSE) for the 10 final global best architecture from each of the 10 experimental repeats for both of the datasets. It presents descriptive statistics (average, standard deviation, minimum, and maximum) for all nine 3x3 grow-shrink configurations and the baseline. The final column provides the p-value from a statistical comparison of each configuration's results against the baseline. Similarly, Table 2 reports the same descriptive statistics and p-value comparisons for the total number of enabled weights for the global best architectures found for both of the datasets, providing insight into the evolved model complexity.

For both datasets, none of the grow-shrink configurations (G:50/S:50 through G:200/S:200) yielded a statistically significant difference at a 95% confidence interval in final validation MSE compared to the baseline (all p-values ≥ 0.05). This indicates that while the evolutionary process was subjected to alternating growth and shrinkage phases, the overall predictive performance, as measured by validation MSE, converged to comparable levels across all tested configurations and the baseline on both datasets.

While the alternating grow-shrink phases did not have a statistically significant impact model accuracy, they did have a statistically significant impact on the final model complexity. For the aviation dataset, the G:50/S:200 configuration produced models with an average of 67.9 parameters, a significant reduction from the baseline's average of 75.7 (p=0.0040). This effect was even

Table 2: Summary of Enabled Weights results for Aviation Flight Recorder and Wind Turbine datasets.

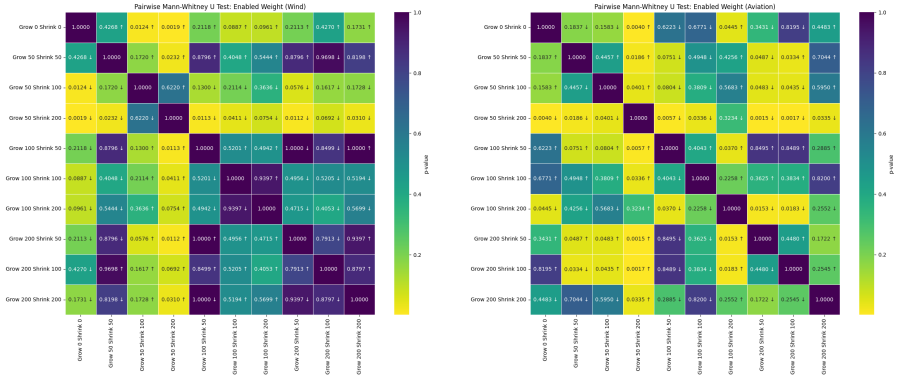
Dataset	Experiment	Min	Avg	Max	Std	P-value
Aviation	Baseline	67	75.7	83	4.9	1
Aviation	Grow 50 Shrink 50	66↓	77.8↑	136↑	19.5948972949592↑	0.1837
Aviation	Grow 50 Shrink 100	65↓	76.1↑	122↑	15.90880259479↑	0.1583
Aviation	Grow 50 Shrink 200	**64**↓	**67.9**↓	79↓	4.25323406362734↓	**0.0040**
Aviation	Grow 100 Shrink 50	65↓	85.5↑	166↑	27.5581204003466↑	0.6223
Aviation	Grow 100 Shrink 100	**64**↓	82.5↑	146↑	23.6103790736193↑	0.6771
Aviation	Grow 100 Shrink 200	**64**↓	70.3↓	**77**↓	4.77598157450382↓	**0.0445**
Aviation	Grow 200 Shrink 50	69↑	78.2↑	87↑	5.67097875150313↑	0.3431
Aviation	Grow 200 Shrink 100	71↑	76.4↑	82↓	**3.26190128606001**↓	0.8195
Aviation	Grow 200 Shrink 200	66↓	74.8↓	95↑	8.41189633792523↑	0.4483
Wind	Baseline	79	103.1	124	14.4391828	1
Wind	Grow 50 Shrink 50	65↓	95.9 ↓	121↓	15.87104281↑	0.4268
Wind	Grow 50 Shrink 100	60↓	82.6↓	**97**↓	12.4675579↓	**0.0124**
Wind	Grow 50 Shrink 200	63↓	**79**↓	100 ↓	10.02995513↓	**0.0019**
Wind	Grow 100 Shrink 50	57↓	93↓	109↓	13.61616686↓	0.2118
Wind	Grow 100 Shrink 100	63↓	90↓	105↓	12.19836055↓	0.0887
Wind	Grow 100 Shrink 200	**56**↓	88↓	101↓	14.35966573↓	0.0961
Wind	Grow 200 Shrink 50	79↓	94.3↓	109↓	**9.86965045**↓	0.2113
Wind	Grow 200 Shrink 100	66↓	96.5↓	141↑	21.58355856↑	0.4270
Wind	Grow 200 Shrink 200	69↓	92.8↓	111↓	14.04136745↓	0.1731

more pronounced on the wind turbine dataset, where the G:50/S:200 configuration resulted in models averaging 79.0 parameters, compared to the baseline's 103.1 (p=0.0019). This indicates that specific grow-shrink configurations, particularly those with a Shrink 200 but with low Growth configurations (Growth:50) component, successfully guided the evolutionary search toward more smaller architectures. The fact that the aviation dataset has more input parameters than the wind turbine dataset (31 vs 26) may also lead to the stronger statistical significance for model size, as more parameters can lead to more complex models or variance in potential model size.

5.2 Pairwise Statistical Comparison of Model Complexity

To provide a comprehensive statistical understanding of how the various grow-shrink phase configurations impact final model complexity, we performed a pairwise Mann-Whitney U test on the "Enabled Weights" (enabled parameters). This non-parametric test was applied to the combined set of the final 10 global best genomes from all 10 independent runs for every possible configuration, including the baseline (Grow 0 Shrink 0).

The results are presented as heatmaps for the wind turbine and aviation datasets, as shown in Figure 1. Each cell in the heatmap represents the p-value comparing the final model complexity distributions of the intersecting row and column configurations. A p-value less than 0.05 (highlighted in yellow) indicates a statistically significant difference in model size between the two configurations.



(a) Wind Turbine Dataset (b) Aviation Flight Recorder Dataset

Fig. 1: Pairwise Mann-Whitney U test results for final model enabled weights. Each cell in the p-value heamaps displays the uncorrected p-value for the comparison between the row experiment and the column experiment.

The arrow in each cell indicates the direction of the mean difference: ↑ signifies the row experiment's average was higher than the column's, while ↓ signifies it was lower. The color intensity corresponds to the p-value, with darker shades indicating lesser statistical significance (higher p-value).

The heatmaps powerfully reinforce the quantitative findings from Tables 1 and 2, and confirm the effectiveness of the G:50/S:200 configuration in achieving model compression:

- For the aviation dataset, the G:50/S:200 configuration shows a highly significant difference from the baseline ($p = 0.0040$), as previously noted. The heatmaps also reveal other configuration, G:100/S:50 ($p = 0.0445$), that similarly resulted in statistically smaller architectures than the baseline.
- For the wind turbine dataset, G:50/S:200 ($p = 0.0019$) again shows a highly significant difference from the baseline, which is the most pronounced reduction observed. Other configurations showing significant reduction include G:50/S:100 ($p = 0.0124$).

Superiority of G:50/S:200:

- The G:50/S:200 configuration (fourth row/column) is statistically the most effective model reducer. On the aviation dataset, it yields a statistically significant difference (smaller models) against six of the nine tested grow-shrink configurations. On the wind turbine dataset, G:50/S:200 maintains a statistically significant reduction against four other configurations.
- This analysis solidifies the conclusion that G:50/S:200 does not just beat the baseline, but is one of the best complexity-reducing phases overall, especially when combined with a low growth frequency. The low p-values associated with high shrinkage rates (e.g., Shrink 200 compared to Shrink 50) confirm

the design hypothesis that increasing the ratio of the shrink phase is the dominant factor in successfully guiding the evolutionary process toward more smaller neural architectures.

5.3 Impact of Grow-Shrink Configurations OR Temporal Analysis of Model Complexity

To visually demonstrate the impact of the alternating grow-shrink phases on model complexity, Figures 2 and 4 compare the Enabled Weight Count (total parameters) during the evolutionary run for the baseline (Grow 0 Shrink 0) against the most successful pruning configuration (Grow 50 Shrink 200) across both datasets where points show the enabled parameter counts for each generated model during the search.

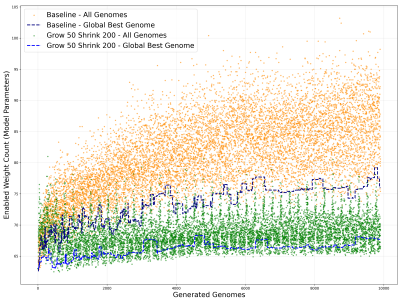


Fig. 2: Enabled weight count (total parameters) during evolutionary runs for the aviation dataset with G:50/S:200.

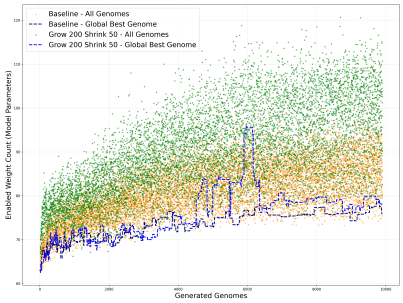


Fig. 3: Enabled weight count (total parameters) during evolutionary runs for the aviation dataset with G:200/S:50.

On the aviation dataset (Figure 2), the Grow 50 Shrink 200 configuration consistently generates genomes with a lower parameter count (green scatter) compared to the baseline (orange scatter). Furthermore, the global best complexity maintained by Grow 50 Shrink 200 (dark blue solid line) is significantly lower than the baseline's (dark blue dashed line) throughout the 10,000 generated genomes. This visual evidence supports the finding in Table 2 that this specific configuration yields a statistically significant reduction in final model size.

This pattern is replicated and even more pronounced on the wind turbine dataset (Figure 4). Here, the Grow 50 Shrink 200 configuration successfully suppresses model growth much more effectively than the baseline throughout the run. The maximum complexity of genomes generated by Grow 50 Shrink 200 is consistently lower, and its global best complexity trace remains substantially below the baseline's from approximately 2,000 genomes onward. The clear separation between the two complexity trends in Figure 4 further validates that the

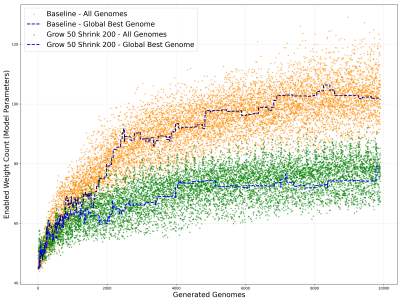


Fig. 4: Enabled weight counts (total parameters) during evolutionary runs on the wind turbine dataset with G:50/S:200.

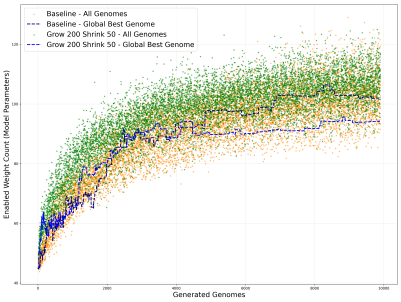
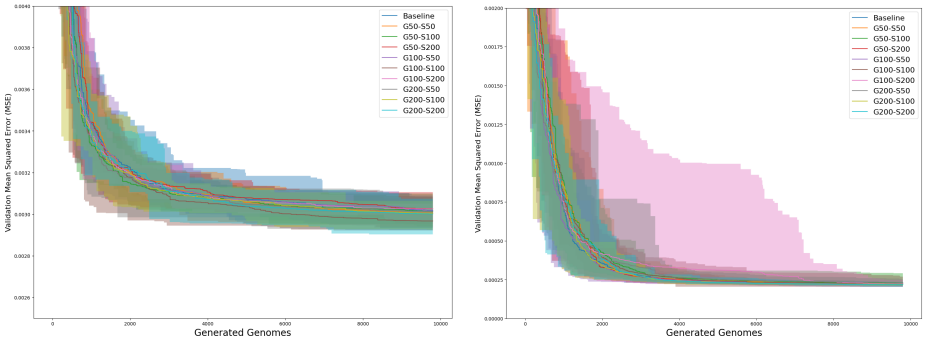


Fig. 5: Enabled weight counts (total parameters) during evolutionary runs on the wind turbine dataset with G:200/S:50.

Grow 50 Shrink 200 configuration is the most effective combination tested at guiding the evolutionary search towards finding smaller architectures.



(a) Wind Turbine Dataset (b) Aviation Dataset

Fig. 6: Average validation MSE convergence comparing the baseline to the nine Grow-Shrink (G-S) configurations. Abbreviations "G" and "S" refer to the phase length in generations (e.g., G50-S100 represents a 50-generation grow phase followed by a 100-generation shrink phase). Shaded areas represent the minimum and maximum MSE across 10 independent runs.

Figure 6a shows the average validation MSE convergence on the wind turbine dataset. The baseline (Grow 0 Shrink 0, in blue) and the G:50/S:200 configuration (in orange) exhibit similar convergence trends. Both configurations rapidly decrease the validation MSE within the first 2,000 generated genomes before settling into a relatively stable state. The shaded areas around the average lines represent the minimum and maximum validation MSE values observed across the

ten independent runs. This range illustrates the full spectrum of results obtained from the evolutionary process. Notably, the G:50/S:200 configuration achieves a final validation MSE that is comparable to the baseline, demonstrating that the structural evolution process does not compromise the final predictive accuracy.

Figure 6b presents the average validation MSE convergence on the aviation dataset. Similar to the wind turbine dataset, both the baseline and the G:50/S:200 configuration rapidly decrease the validation MSE. The convergence is quicker on this dataset, with the error stabilizing below MSE=0.0001 after approximately 2,000 generated genomes. The variability is slightly more pronounced for the G:50/S:200 configuration during the early stages of evolution (500 to 2,000 genomes) compared to the baseline. However, the final convergence point for the G:50/S:200 configuration remains comparable to the baseline, reinforcing the finding that the periodic application of the shrink phase allows for complexity reduction without significant performance degradation.

6 Conclusion

This work introduces the use of alternating growth and pruning phases in neuroevolution, mimicking the refine-and-stabilize cycles found in natural biological systems such as during early development and REM sleep. To our knowledge, this work presents the first integration of these as alternating phases in a neuroevolution algorithm. Results provide a clear answer to both research questions. For RQ1 (Accuracy), our findings show that no grow-shrink configuration produced a statistically significant difference in final validation MSE compared to the baseline including the most aggressive pruning schedules. This is a critical finding – imposing a structured pruning phase does *not* hinder the algorithm's ability to find accurate models. For RQ2 (Parameter Size), the schedules had a significant and desirable effect on final model complexity. Configurations with long shrink phases, particularly Grow 50/Shrink 200, produced networks that were statistically and practically smaller than the baseline. This was most pronounced on the wind dataset (which had a larger parameter space), where it reduced the average model size from 103.1 to 79.0 enabled weights ($p = 0.0019$). The aviation dataset also still had a significant reduction from 75.7 to 67.9 weights ($p = 0.0040$). This study demonstrates that separating growth and pruning into distinct, alternating phases is a simple yet powerful method for controlling complexity in neuroevolution. It also leaves open a number of avenues for future work, such as investigating on more (and larger scale) datasets, and investigating if dynamic adaptation or longer length phases/larger ratios between pruning and growth provide even further effect.

7 Acknowledgments

This research is partially supported by a gift from Lenovo, and used CPU resources provided by Research Computing at Rochester Institute of Technology, and we thank the team for their support [34].

References

1. The National General Aviation Flight Information Database (NGAFID). https://ngafid.org, accessed: 2025-11-12
2. Abbott, L.F., Nelson, S.B.: Synaptic plasticity: taming the beast. Nature neuroscience **3**(11), 1178–1183 (2000)
3. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
4. Collins, J., Sohl-Dickstein, J., Sussillo, D.: Capacity and trainability in recurrent neural networks. arXiv preprint arXiv:1611.09913 (2016)
5. Dai, X., Chen, D., Liu, M., Chen, Y., Yuan, L.: Da-nas: Data adapted pruning for efficient neural architecture search. In: European conference on computer vision. pp. 584–600. Springer (2020)
6. Desell, T., ElSaid, A., Ororbia, A.G.: An empirical exploration of deep recurrent connections using neuro-evolution. In: The 23nd International Conference on the Applications of Evolutionary Computation (EvoStar: EvoApps 2020). Seville, Spain (April 2020)
7. Dettmers, T., Zettlemoyer, L.: Sparse networks from scratch: Faster training without losing performance (2019)
8. Dong, X., Yang, Y.: Network pruning via transformable architecture search. vol. 32 (2019)
9. ElSaid, A., Karns, J., Lyu, Z., Ororbia, A.G., Desell, T.: Continuous ant-based neural topology search. In: International Conference on the Applications of Evolutionary Computation (Part of EvoStar). pp. 291–306. Springer (2021)
10. ElSaid, A., Ororbia, A.G., Desell, T.J.: Ant-based neural topology search (ants) for optimizing recurrent networks. In: International Conference on the Applications of Evolutionary Computation (Part of EvoStar). pp. 626–641. Springer (2020)
11. ElSaid, A., Ricanek, K., Lyu, Z., Ororbia, A., Desell, T.: Backpropagation-free 4d continuous ant-based neural topology search. Applied Soft Computing **147**, 110737 (2023)
12. Evci, U., Gale, T., Menick, J., Castro, P.S., Elsen, E.: Rigging the lottery: Making all tickets winners. In: International conference on machine learning. pp. 2943–2952. PMLR (2020)
13. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arxiv 2018. arXiv preprint arXiv:1803.03635 (1810)
14. Galván, E., Mooney, P.: Neuroevolution in deep neural networks: Current trends and future challenges. IEEE Transactions on Artificial Intelligence **2**(6), 476–493 (2021)
15. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
16. Hassantabar, S., Dai, X., Jha, N.K.: Steerage: Synthesis of neural networks using architecture search and grow-and-prune methods. arXiv preprint arXiv:1912.05831 (2019)
17. Hebb, D.O.: The organization of behavior: A neuropsychological theory. Psychology press (2005)
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

19. Kini, A.D., Yadav, S.S., Thakur, A.S., Awari, A.B., Lyu, Z., Desell, T.: Co-evolving recurrent neural networks and their hyperparameters with simplex hyperparameter optimization. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation. pp. 1639–1647 (2023)
20. Klein, A., Golebiowski, J., Ma, X., Perrone, V., Archambeau, C.: Structural pruning of pre-trained language models via neural architecture search. arXiv preprint arXiv:2405.02267 (2024)
21. Li, G., Xu, M., Giancola, S., Thabet, A., Ghanem, B.: Lc-nas: Latency constrained neural architecture search for point cloud networks. In: 2022 International Conference on 3D Vision (3DV). pp. 1–11. IEEE (2022)
22. Li, Q., Zhang, B., Chu, X.: Eapruning: evolutionary pruning for vision transformers and cnns. arXiv preprint arXiv:2210.00181 (2022)
23. Li, W., Ma, L., Yang, G., Gan, W.B.: Rem sleep selectively prunes and maintains new synapses in development and learning. Nature neuroscience **20**(3), 427–437 (2017)
24. Liu, T., Zenke, F.: Finding trainable sparse networks through neural tangent transfer pp. 6336–6347 (2020)
25. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE international conference on computer vision. pp. 2736–2744 (2017)
26. Lyu, Z., ElSaid, A., Karns, J., Mkaouer, M., Desell, T.: An experimental study of weight initialization and lamarckian inheritance on neuroevolution. The 24th International Conference on the Applications of Evolutionary Computation (EvoStar: EvoApps) (2021)
27. Lyu, Z., Karnas, J., ElSaid, A., Mkaouer, M., Desell, T.: Improving distributed neuroevolution using island extinction and repopulation. The 24th International Conference on the Applications of Evolutionary Computation (EvoStar: EvoApps) (2021)
28. Lyu, Z., Ororbia, A., Desell, T.: Online evolutionary neural architecture search for multivariate non-stationary time series forecasting. Applied Soft Computing **145**, 110522 (2023)
29. Miikkulainen, R.: Neuroevolution insights into biological neural computation. Science **387**(6735), eadp7478 (2025)
30. Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., Srebro, N.: Towards understanding the role of over-parametrization in generalization of neural networks. arXiv preprint arXiv:1805.12076 (2018)
31. Ororbia, A., ElSaid, A., Desell, T.: Investigating recurrent neural network memory structures using neuro-evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 446–455. GECCO '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3321707.3321795, http://doi.acm.org/10.1145/3321707.3321795
32. Ororbia II, A.G., Mikolov, T., Reitter, D.: Learning simpler language models with the differential state framework. Neural Computation **0**(0), 1–26 (2017). https://doi.org/10.1162/neco_a_01017, https://doi.org/10.1162/neco_a_01017, pMID: 28957029
33. Rasch, B., Born, J.: About sleep's role in memory. Physiological reviews (2013)
34. Rochester Institute of Technology: Research Computing Services (2019). https://doi.org/10.34788/0S3G-QD15, https://doi.org/10.34788/0S3G-QD15, accessed 2026-01-23
35. Salmani Pour Avval, S., Eskue, N.D., Groves, R.M., Yaghoubi, V.: Systematic review on neural architecture search. Artificial Intelligence Review **58**(3), 73 (2025)

36. Shang, H., Wu, J.L., Hong, W., Qian, C.: Neural network pruning by cooperative coevolution. arXiv preprint arXiv:2204.05639 (2022)
37. Shen, M., Yin, H., Molchanov, P., Mao, L., Liu, J., Alvarez, J.M.: Structural pruning via latency-saliency knapsack. Advances in Neural Information Processing Systems **35**, 12894–12908 (2022)
38. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
39. Thakur, A.S., Awari, A.B., Lyu, Z., Desell, T.: Efficient neuroevolution using island repopulation and simplex hyperparameter optimization. In: 2023 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1837–1842. IEEE (2023)
40. Turrigiano, G.: Homeostatic synaptic plasticity: local and global mechanisms for stabilizing neuronal function. Cold Spring Harbor perspectives in biology **4**(1), a005736 (2012)
41. Turrigiano, G.G., Nelson, S.B.: Hebb and homeostasis in neuronal plasticity. Current opinion in neurobiology **10**(3), 358–364 (2000)
42. Zenke, F., Gerstner, W., Ganguli, S.: The temporal paradox of hebbian learning and homeostatic plasticity. Current opinion in neurobiology **43**, 166–176 (2017)
43. Zhan, Z., Gong, Y., Zhao, P., Yuan, G., Niu, W., Wu, Y., Zhang, T., Jayaweera, M., Kaeli, D., Ren, B., et al.: Achieving on-mobile real-time super-resolution with neural architecture and pruning search. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4821–4831 (2021)
44. Zheng, X., Ma, Y., Xi, T., Zhang, G., Ding, E., Li, Y., Chen, J., Tian, Y., Ji, R.: An information theory-inspired strategy for automated network pruning. International Journal of Computer Vision pp. 1–28 (2025)
45. Zheng, X., Yang, C., Zhang, S., Wang, Y., Zhang, B., Wu, Y., Wu, Y., Shao, L., Ji, R.: Ddpnas: Efficient neural architecture search via dynamic distribution pruning. International Journal of Computer Vision **131**(5), 1234–1249 (2023)
46. Zhou, G.B., Wu, J., Zhang, C.L., Zhou, Z.H.: Minimal gated unit for recurrent neural networks. International Journal of Automation and Computing **13**(3), 226–234 (2016)
47. Zou, J., Jiang, W., Xia, Y., Liu, Y., Hou, Z.: G-evonas: Evolutionary neural architecture search based on network growth. arXiv preprint arXiv:2403.02667 (2024)