An Experimental Study of Weight Initialization and Lamarckian Inheritance on Neuroevolution*

Zimeng Lyu, AbdElRahman ElSaid, Joshua Karns, Mohamed Mkaouer, and Travis Desell

Rochester Institute of Technology, Rochester, NY 14623
zimenglyu@mail.rit.edu, aelsaid@mail.rit.edu, josh@mail.rit.edu, mwmvse@rit.edu, tjdvse@rit.edu

Abstract. Weight initialization is critical in being able to successfully train artificial neural networks (ANNs), and even more so for recurrent neural networks (RNNs) which can easily suffer from vanishing and exploding gradients. In neuroevolution, where evolutionary algorithms are applied to neural architecture search, weights typically need to be initialized at three different times: when the initial genomes (ANN architectures) are created, when offspring genomes are generated by crossover, and when new nodes or edges are created during mutation. This work explores the difference between the state-of-the-art Xavier and Kaiming methods, and novel Lamarckian weight inheritance for weight initialization during crossover and mutation operations. These are examined using the Evolutionary eXploration of Augmenting Memory Models (EXAMM) neuroevolution algorithm, which is capable of evolving RNNs with a variety of modern memory cells (e.g., LSTM, GRU, MGU, UGRNN and Delta-RNN cells) as well as recurrent connections with varying time skips through a high performance island based distributed evolutionary algorithm. Results show that with statistical significance, the Lamarckian strategy outperforms both Kaiming and Xavier weight initialization, can speed neuroevolution by requiring less backpropagation epochs to be evaluated per genome, and that the neuroevolutionary process provides further benefits to neural network weight optimization.

Keywords: Neuroevolution · Neural Architecture Search · Weight Inheritance · Weight Initialization · Lamarckian Evolution

1 Introduction

Neuroevolution (NE), or the use of evolutionary algorithms (EAs) for neural architecture search (NAS) and training, has seen a significant growth in popularity due to the challenges of designing deep neural networks [27, 17]. While

* This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Combustion Systems under Award Number #FE0031547 and by the Federal Aviation Administration and MITRE Corporation under the National General Aviation Flight Information Database (NGAFID) award.

some approaches to NE, such as indirect encoding, *e.g.*, HyperNEAT [28], where the genomes are used to generate the architecture and assign weights; or fitness estimation, *e.g.*, [3, 2] where genome fitness is estimated without training the networks, most modern NE algorithms involve a direct encoding approach, where a neural network's architecture and weights are directly represented as *genomes* that can be evolved by crossover and mutation operations.

In direct encoding, the initialization of network weights is critical, especially for deep neural networks (DNNs) [21], as it has been shown that poor weight initialization quickly leads to gradient vanishing and exploding problems [11]. The Xavier [11] and Kaiming weight initialization [12] methods have been a great success in reducing issues for DNNs and are now the de facto standard for training DNNs, however, these methods do not take into account extra information available during NE. For example, during mutation, a child genome is generated by randomly modifying a previously trained parent genome, and during crossover, a child genome is generated utilizing two (or more) previously trained parental genomes. These parental distributions and weight values and distributions contain valuable information that can be used to better initialize child genome weights, and this process is known as *Lamarckian* [24, 23, 6] or sometimes *Epigenetic* [7] weight initialization.

Unfortunately, many NE algorithms still use an outdated uniform random initialization for initial populations [26, 29, 9, 19], with a few exceptions that use Xavier [1, 23] or Kaiming [7] initialization. Among recent state-of-the-art NAS works, few used the Kaiming or Xavier methods, such as ENAS [22] which uses Kaiming, but others such as NSGA-NET [18] and Progressive NAS [16], still use uniform random weight initialization.

In Evolutionary Algorithms (EAs), using Lamarckian weight inheritance means that offspring inherit weights from their parents through mutation or crossover. Some studies suggest that Lamarckian weight inheritance can reduce the number of backpropagation (BP) epochs to train neural networks [7, 15] and lead to better performing neural networks, but to the best of authors' knowledge, Lamarckian weight inheritance has not been rigorously compared to the modern Xavier and Kaiming weight initialization methods.

The main contribution of this work is to provide an experimental analysis of Lamarckian weight inheritance methods (one for crossover and another for mutation) to Xavier and Kaiming initialization. This study was done in the context of evolving deep recurrent neural networks (RNNs) for time series data prediction using three challenging real world data sets. Results are promising, showing that with statistical significance the Lamarckian strategies outperform Xavier and Kaiming weight initialization, and further can reduce the amount of BP epochs used to train the neural networks, allowing more time to be spent on architectural evolution. Additionally, the neuroevolutionary process is shown to provide additional benefits to the selection of weights, as it is shown that when the best found architectures are retrained, even for a large number of epochs, in most cases, they do not surpass the performance of the networks with the evolved weights.

2 Weight Initialization and Inheritance

2.1 Xavier and Kaiming Weight Initialization

Xavier weight initialization [11] was designed for DNNs with symmetrical activation functions such as *tanh* and $softsign$. The weights in each layer are generated using a uniform distribution:

$$W \sim \mathcal{U}[-\frac{\sqrt{6}}{\sqrt{f_{in}+f_{out}}}, \frac{\sqrt{6}}{\sqrt{f_{in}+f_{out}}}] \tag{1}$$

where f_{in} and f_{out} are fan in and fan out of the layer[1].

Kaiming weight initialization [12] was designed for non-symmetrical activation functions such as ReLUs. The weights in each layer are generated with a normal distribution, and the fan in, f_{in}:

$$W \sim N(0,1) * \frac{\sqrt{2}}{f_{in}} \tag{2}$$

2.2 Lamarckian Weight Inheritance

The Lamarckian strategies investigated in this work were first introduced by Desell *et al.* for NE of convolutional neural networks (CNNs) and later used for recurrent neural networks [7, 19]. While Prellberg and Kramer also investigated Lamarckian weight inheritance for CNNs, their strategy was a simpler version where Lamarckian inheritance was only done on crossover, and mutated components were re-initialized randomly [23].

For direct encoding NE algorithms, after the initial genomes' weights are initialized, new genomes are created either via crossover, where two or more parents are recombined into a child genome, or by mutation where a single parent has one or more random modifications made.

For crossover, given a more fit and less fit parent, child genome weights are initialized as follows. When the same architectural component (*e.g.*, node, edge or layer) exists in both parents[2], the weights and biases for that component are generated using a stochastic line search recombining weights or biases from those in the parents' components. Given a random number $r \sim \mathcal{U}[-0.5, 1.5]$, a child's weight w_c is set to:

$$w_c = r(w_{p2} - w_{p1}) + w_{p1} \tag{3}$$

where w_{p1} is the weight from the more fit parent, and w_{p2} is the weight from the less fit parent (note the same r value is used for *all* child weights). This allows

[1] Fan in is the number of input signals that feed into the layer, fan out is the number of output signals that come out of the layer

[2] Components are identified as being the same by having the same *innovation number*, which is uniquely created by the neuroevolution process when an architectural component is added to a genome, and are inherited by children on crossover and mutation, as in the NEAT algorithm [26].

the child weights to be set along a gradient calculated from the weights of the two parents, performing an informed exploration of the weight space between and around the two parents. In the case where the component only exists in one parent, the same weights and biases are copied to the child.

For mutations, new components are added to the parent neural network architecture, so it is not possible to directly utilize weights from the parent. Instead, statistical information about the weight distributions of the parents can be used. Weights and biases for new components generated during mutations are instead initialized using a normal distribution around the mean μ_p and variance σ_p^2 of the parent's weights:

$$W \sim N(\mu_p, \sigma_p^2) \tag{4}$$

while the other weights are directly copied from the parent. This network-aware approach using the statistical distribution of a network's weights has also been shown to speed transfer learning, lending further credence to this approach [10].

3 Methodology

This work utilizes the Evolutionary eXploration of Augmenting Memory Models (EXAMM) neuroevolution algorithm [19] to explore the different weight initialization and inheritance strategies. EXAMM evolves progressively larger RNNs through a series of mutation and crossover (reproduction) operations. When nodes are added their type is selected uniformly at random from a suite of simple neurons and complex memory cells: Δ-RNN units [20], gated recurrent units (GRUs) [4], long short-term memory cells (LSTMs) [13], minimal gated units (MGUs) [30], and update gate RNN cells (UGRNNs) [5]. This allows EXAMM to select the best performing recurrent memory units. EXAMM also allows *deep recurrent connections*, which enable the RNN to directly use information beyond the previous time step. These deep recurrent connections have proven to offer significant improvements in model generalization, even yielding models that outperform state-of-the-art gated architectures [8]. EXAMM has both a multithreaded implementation and an MPI implementation for distributed use on high performance computing resources. To the authors' knowledge, these capabilities are not available in other neuroevolution frameworks capable of evolving RNNs, which is the primary reason EXAMM was selected for this work.

EXAMM uses an asynchronous island based evolution strategy with a fixed number of islands n, each with an island capacity m. During the evolution process, islands go through two phases: *initialization*, and *filled*. During the *initialization* phase, each island starts with one seed genome, which is the minimal possible feed-forward neural network structure with no hidden layers, with the input layer fully connected to the output layer. Worker processes repeatedly request genomes to evaluate from the master process using a work-stealing approach.

On receiving a genome the worker then evaluates its *fitness*, calculated as mean squared error (MSE) on a validation data set after stochastic back propagation training. When reported back to the master process, if the island is

not full, it is inserted into the island; otherwise, if the *fitness* is better than the worst genome on that island, it will replace the worst genome. The master generates new genomes from islands in a round-robin manner, by doing a random mutation on randomly selected genomes from an island until that island reaches maximum capacity m, and its status becomes *filled*. When all islands are *filled*, they repopulate through inter-island crossover, intra-island crossover and mutation operations. *Intra-island crossover* selects two random genomes from the same island, and the child gets inserted back to where its parents come from. *Inter-island crossover* selects the first parent at random from the island the child will be inserted into, and the second parent is the best genome from another randomly selected island. As islands are distinct sub-populations and otherwise evolve independently, the only chance for the islands to exchange genes is through *inter-island crossover*.

4 Results

Data Sets: This work utilized three real-world data sets for predicting time series data with RNNs[3]. The first comes from data collected from 12 burners of a coal-fired power plant, the second is the wind turbine engine data from 2013 to 2020, collected and made available by ENGIE's La Haute Borne open data windfarm[4], and the third comes from a selection of 10 flights worth of data from the National General Aviation Flight Information Database (NGAFID). All of the datasets are multivariate (with 12, 88, and 31 parameters, respectively), non-seasonal, and the parameter recordings are not independent. Furthermore, they are very long. The power plant data consists of 10-days worth of per-minute data, the wind turbine data consists of readings every 10 minutes from 2013 to 2020, and the aviation time series range from 1 to 3 hours worth of per-second data. *Main flame intensity* was chosen as the output parameter for the coal dataset and *average active power* was selected as output parameter for the wind turbine data set. The aviation dataset was used to predict 4 engine output parameters, *E1 CHT1*, *E1 CHT2*, *E1 CHT3*, *E1 CHT4*.

Results were gathered using Rochester Institute of Technology's research computing systems. This system consists of 2304 Intel® Xeon® Gold 6150 CPU 2.70GHz cores and 24 TB RAM, with compute nodes running the RedHat Enterprise Linux 7 system. Each experiment utilized 72 cores.

Each EXAMM run used 10 islands, each with a maximum capacity of 10 genomes. New RNNs were generated via mutation at a rate of 70%, intra-island crossover at a rate of 20%, and inter-island crossover at a rate of 10%. 10 out of EXAMM's 11 mutation operations were utilized (all except for *split edge*), and each was chosen with a uniform 10% chance. EXAMM generated new nodes by selecting from simple neurons, Δ-RNN, GRU, LSTM, MGU, and UGRNN

[3] These data sets are made publicly available at EXAMM GitHub repository: https://github.com/travisdesell/exact/tree/master/datasets/

[4] https://opendata-renewables.engie.com

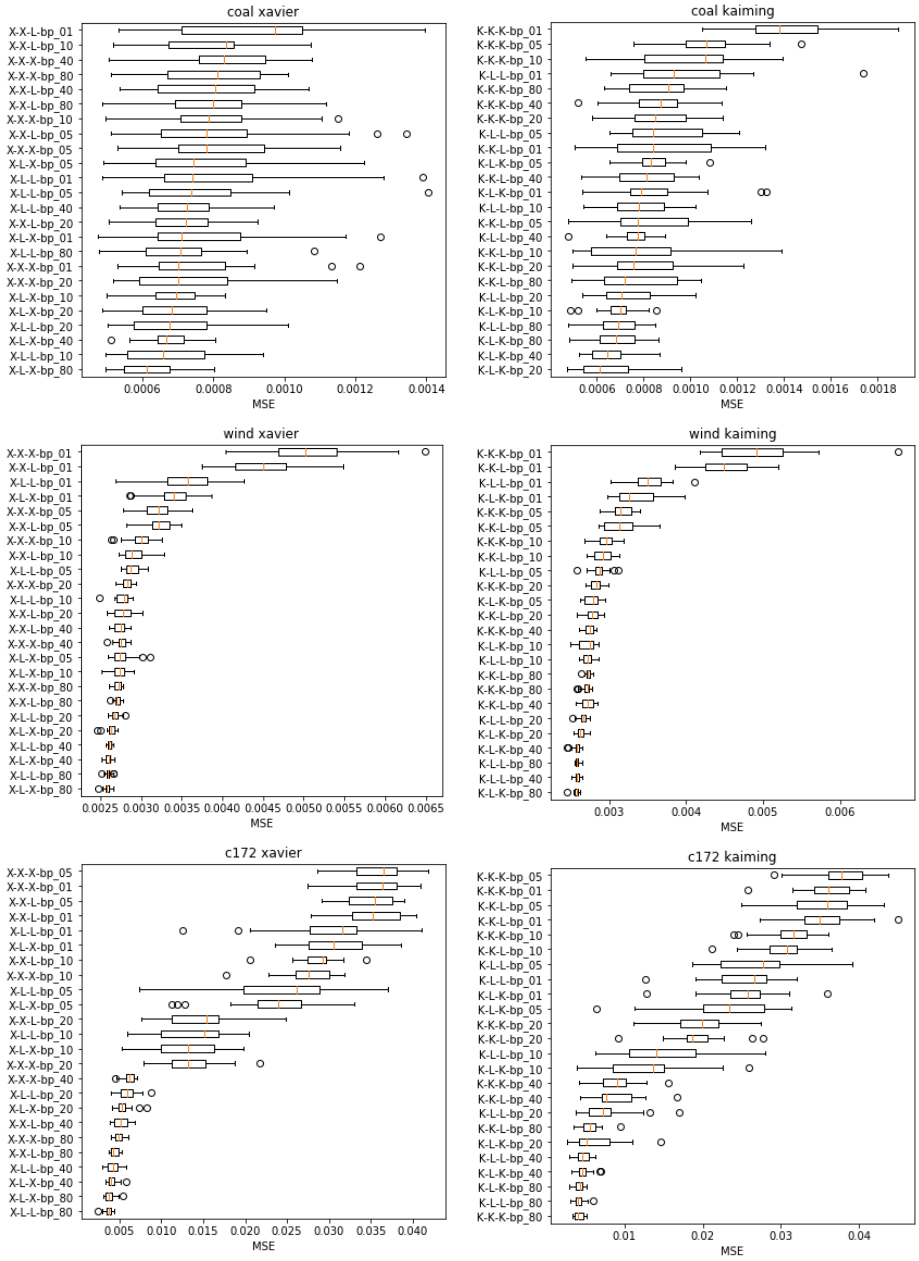

Fig. 1: Convergence rates (in terms of best MSE on validation data) with Xavier, Kaiming weight initialization predicting *main flame intensity* from the coal fired power plant dataset, *average active power* for the wind turbine dataset, and *E1 CHT1*, *E1 CHT2*, *E1 CHT3*, *E1 CHT4* for the aviation dataset. Weight initialization and inheritance type are labeled by *initial genome strategy-crossover strategy-mutation strategy*, so *e.g.*, *K-L-K* would use Kaiming for the initial genomes, Lamarckian on crossover operations, and Kaiming for components generated by mutation.

BP Epochs	Type	Avg Node	Avg Edge	Avg Rec Edge	Worst MSE	Avg MSE	Best MSE
1	K-K-K	17.1	41.6	38.0	1.89e-3	1.42e-3	1.05e-3
	K-L-K	29.8	118.2	55.6	1.74e-3	9.84e-4	6.60e-4
	K-L-L	23.3	90.0	64.0	1.33e-3	8.40e-4	5.36e-4
	K-K-L	16.7	41.0	49.8	1.32e-3	8.83e-4	5.03e-4
	X-X-X	16.6	39.4	47.0	***1.21e-3***	***7.51e-4***	5.30e-4
	X-L-X	23.4	92.3	65.8	1.39e-3	8.18e-4	4.87e-4
	X-L-L	21.8	81.7	75.1	1.27e-3	7.85e-4	***4.75e-4***
	X-X-L	16.4	39.6	54.9	1.40e-3	9.11e-4	5.32e-4
5	K-K-K	16.1	34.5	28.3	1.47e-3	1.07e-3	7.58e-4
	K-L-K	21.9	73.2	47.6	1.21e-3	8.88e-4	6.56e-4
	K-L-L	20.0	62.5	35.5	***1.08e-3***	8.36e-4	6.52e-4
	K-K-L	15.9	31.9	22.0	1.26e-3	8.16e-4	***4.75e-4***
	X-X-X	15.6	31.7	19.9	1.16e-3	8.21e-4	5.29e-4
	X-L-X	18.4	52.0	27.6	1.41e-3	***7.70e-4***	5.42e-4
	X-L-L	18.9	56.6	36.9	1.23e-3	7.73e-4	4.89e-4
	X-X-L	16.2	33.4	26.6	1.34e-3	8.22e-4	5.11e-4
10	K-K-K	16.1	34.0	20.9	1.39e-3	9.89e-4	5.51e-4
	K-L-K	22.9	71.0	33.6	1.02e-3	7.80e-4	5.45e-4
	K-L-L	21.6	66.1	32.0	8.54e-4	6.90e-4	***4.87e-4***
	K-K-L	15.7	30.1	17.7	1.39e-3	7.78e-4	4.95e-4
	X-X-X	16.4	32.0	15.8	1.15e-3	8.06e-4	4.97e-4
	X-L-X	18.4	51.2	23.1	9.41e-4	***6.78e-4***	4.96e-4
	X-L-L	19.9	58.0	34.5	***8.33e-4***	6.90e-4	5.01e-4
	X-X-L	16.0	30.3	15.5	1.08e-3	7.79e-4	5.18e-4
20	K-K-K	16.4	30.9	11.5	1.14e-3	8.62e-4	5.80e-4
	K-L-K	20.9	58.0	25.0	1.02e-3	7.37e-4	5.38e-4
	K-L-L	18.2	43.6	17.4	9.60e-4	***6.57e-4***	**4.72e-4**
	K-K-L	15.8	28.1	10.2	1.22e-3	8.11e-4	4.95e-4
	X-X-X	16.5	30.6	11.6	1.15e-3	7.27e-4	5.19e-4
	X-L-X	18.3	42.3	18.8	1.01e-3	6.96e-4	5.04e-4
	X-L-L	21.6	61.0	25.2	9.50e-4	6.93e-4	4.88e-4
	X-X-L	16.4	30.9	9.7	***9.25e-4***	7.12e-4	5.06e-4
40	K-K-K	15.9	25.9	7.5	1.13e-3	8.47e-4	5.18e-4
	K-L-K	19.4	46.9	15.5	8.90e-4	7.57e-4	***4.77e-4***
	K-L-L	19.1	43.7	15.1	8.67e-4	***6.55e-4***	5.26e-4
	K-K-L	15.8	26.2	7.6	1.04e-3	8.09e-4	5.34e-4
	X-X-X	15.9	26.6	6.5	1.08e-3	8.34e-4	5.06e-4
	X-L-X	18.1	37.1	12.1	9.69e-4	7.25e-4	5.37e-4
	X-L-L	17.8	34.9	13.4	***8.05e-4***	6.72e-4	5.12e-4
	X-X-L	15.6	26.2	6.5	1.07e-3	7.79e-4	5.36e-4
80	K-K-K	15.3	22.4	4.3	1.15e-3	8.71e-4	6.32e-4
	K-L-K	17.8	34.8	9.1	8.48e-4	6.78e-4	***4.75e-4***
	K-L-L	17.7	35.0	8.8	8.62e-4	6.76e-4	4.81e-4
	K-K-L	15.7	23.7	3.1	1.05e-3	7.64e-4	4.93e-4
	X-X-X	14.9	20.9	3.9	1.01e-3	7.93e-4	5.13e-4
	X-L-X	17.0	30.8	7.0	1.08e-3	6.97e-4	4.79e-4
	X-L-L	16.8	30.4	6.5	**8.02e-4**	**6.19e-4**	4.95e-4
	X-X-L	15.4	23.0	5.7	1.12e-3	7.82e-4	4.88e-4

Table 1: Statistics of best genomes over 20 repeats on the coal dataset. Weight initialization and inheritance type are labeled by *initial genome strategy-crossover strategy-mutation strategy*, so *e.g.*, *K-L-K* would use Kaiming for the initial genomes, Lamarckian on crossover operations, and Kaiming for components generated by mutation.

BP Epochs	Type	Avg Node	Avg Edge	Avg Rec Edge	Worst MSE	Avg MSE	Best MSE
1	K-K-K	95.0	123.0	7.2	6.76e-3	4.95e-3	4.19e-3
	K-L-K	101.3	170.2	24.2	4.11e-3	3.51e-3	3.02e-3
	K-L-L	95.0	151.8	33.3	3.98e-3	***3.37e-3***	2.98e-3
	K-K-L	93.7	121.1	5.5	5.20e-3	4.55e-3	3.85e-3
	X-X-X	95.1	126.9	6.6	6.50e-3	5.07e-3	4.04e-3
	X-L-X	99.2	170.2	27.6	4.26e-3	3.57e-3	***2.69e-3***
	X-L-L	94.3	154.1	33.5	***3.86e-3***	3.38e-3	2.86e-3
	X-X-L	96.0	132.1	8.6	5.48e-3	4.50e-3	3.75e-3
5	K-K-K	94.0	121.1	9.5	3.40e-3	3.17e-3	2.88e-3
	K-L-K	98.7	158.1	24.9	3.12e-3	2.87e-3	***2.58e-3***
	K-L-L	96.2	139.8	25.5	***2.94e-3***	***2.77e-3***	2.62e-3
	K-K-L	94.5	124.8	10.8	3.66e-3	3.16e-3	2.87e-3
	X-X-X	94.5	126.0	11.2	3.62e-3	3.21e-3	2.78e-3
	X-L-X	98.7	154.8	20.5	3.09e-3	2.90e-3	2.75e-3
	X-L-L	95.3	140.3	23.4	3.12e-3	***2.77e-3***	2.60e-3
	X-X-L	93.6	119.3	10.9	3.50e-3	3.23e-3	2.82e-3
10	K-K-K	93.7	119.4	10.2	3.18e-3	2.94e-3	2.68e-3
	K-L-K	100.3	159.6	22.2	***2.86e-3***	2.72e-3	2.61e-3
	K-L-L	96.4	140.8	22.6	***2.86e-3***	***2.71e-3***	***2.49e-3***
	K-K-L	93.5	116.7	10.8	3.13e-3	2.91e-3	2.71e-3
	X-X-X	94.0	121.2	9.4	3.26e-3	2.98e-3	2.63e-3
	X-L-X	97.7	147.1	16.4	2.90e-3	2.77e-3	***2.49e-3***
	X-L-L	95.5	132.9	22.2	2.91e-3	2.72e-3	2.51e-3
	X-X-L	93.2	114.9	9.9	3.28e-3	2.92e-3	2.72e-3
20	K-K-K	93.3	115.2	7.3	3.00e-3	2.83e-3	2.69e-3
	K-L-K	100.3	156.0	18.1	2.75e-3	2.65e-3	2.52e-3
	K-L-L	95.9	130.8	18.9	2.74e-3	***2.63e-3***	2.53e-3
	K-K-L	92.5	110.5	9.2	2.93e-3	2.78e-3	2.58e-3
	X-X-X	93.7	117.0	7.3	2.94e-3	2.82e-3	2.68e-3
	X-L-X	98.5	145.1	13.5	2.81e-3	2.68e-3	2.59e-3
	X-L-L	95.6	128.2	16.9	***2.71e-3***	***2.63e-3***	***2.46e-3***
	X-X-L	92.9	112.4	6.8	3.02e-3	2.79e-3	2.58e-3
40	K-K-K	93.3	115.0	5.2	2.84e-3	2.74e-3	2.61e-3
	K-L-K	101.5	155.2	18.9	***2.65e-3***	2.59e-3	2.51e-3
	K-L-L	94.5	118.2	13.4	***2.65e-3***	***2.58e-3***	***2.46e-3***
	K-K-L	91.8	104.5	5.7	2.85e-3	2.73e-3	2.57e-3
	X-X-X	92.8	110.8	4.8	2.87e-3	2.76e-3	2.58e-3
	X-L-X	98.2	138.8	12.2	2.66e-3	2.61e-3	2.57e-3
	X-L-L	95.4	124.7	14.6	2.67e-3	2.60e-3	2.51e-3
	X-X-L	92.3	108.1	4.9	2.87e-3	2.74e-3	2.60e-3
80	K-K-K	91.5	101.5	4.0	2.78e-3	2.70e-3	2.58e-3
	K-L-K	97.4	131.6	11.2	2.65e-3	2.59e-3	2.55e-3
	K-L-L	94.2	114.8	9.3	**2.62e-3**	**2.57e-3**	**2.45e-3**
	K-K-L	90.8	98.7	4.4	2.80e-3	2.73e-3	2.64e-3
	X-X-X	92.3	106.4	3.0	2.78e-3	2.71e-3	2.60e-3
	X-L-X	97.4	131.1	8.9	2.66e-3	2.59e-3	2.51e-3
	X-L-L	94.0	112.7	10.2	2.66e-3	2.58e-3	2.48e-3
	X-X-L	91.0	98.8	4.0	2.78e-3	2.71e-3	2.62e-3

Table 2: Statistics of best genomes over 20 repeats on the wind dataset. Weight initialization and inheritance type are labeled by *initial genome strategy-crossover strategy-mutation strategy*, so *e.g.*, *K-L-K* would use Kaiming for the initial genomes, Lamarckian on crossover operations, and Kaiming for components generated by mutation.

BP Epochs	Type	Avg Node	Avg Edge	Avg Rec Edge	Worst MSE	Avg MSE	Best MSE
1	K-K-K	46.1	243.7	8.9	4.08e-2	3.60e-2	2.58e-2
	K-L-K	43.0	206.5	8.9	***3.20e-2***	***2.53e-2***	***1.26e-2***
	K-L-L	41.8	196.7	12.3	3.60e-2	2.54e-2	1.28e-2
	K-K-L	42.7	205.3	6.9	4.50e-2	3.52e-2	2.73e-2
	X-X-X	53.8	324.4	16.3	4.09e-2	3.54e-2	2.74e-2
	X-L-X	40.2	181.8	9.2	4.10e-2	2.97e-2	***1.26e-2***
	X-L-L	46.9	249.7	11.2	3.87e-2	3.08e-2	2.36e-2
	X-X-L	53.2	319.4	16.4	4.04e-2	3.56e-2	2.79e-2
5	K-K-K	39.2	167.8	3.1	4.37e-2	3.75e-2	2.91e-2
	K-L-K	41.0	185.3	10.1	3.91e-2	2.69e-2	1.87e-2
	K-L-L	39.7	172.4	8.4	***3.13e-2***	***2.28e-2***	***6.45e-3***
	K-K-L	39.1	162.8	4.2	4.32e-2	3.54e-2	2.50e-2
	X-X-X	39.1	161.2	3.4	4.19e-2	3.60e-2	2.86e-2
	X-L-X	39.6	172.8	10.7	3.70e-2	2.41e-2	7.35e-3
	X-L-L	40.1	182.1	11.1	3.30e-2	2.36e-2	1.13e-2
	X-X-L	40.5	175.2	3.6	3.90e-2	3.48e-2	2.91e-2
10	K-K-K	39.0	167.0	4.7	3.61e-2	3.11e-2	2.40e-2
	K-L-K	40.7	190.3	17.1	2.80e-2	1.52e-2	6.21e-3
	K-L-L	39.7	177.2	16.4	2.59e-2	1.33e-2	***3.82e-3***
	K-K-L	38.4	157.7	3.8	3.65e-2	3.02e-2	2.12e-2
	X-X-X	38.9	163.9	5.2	3.19e-2	2.74e-2	1.77e-2
	X-L-X	40.0	183.2	17.9	2.05e-2	1.35e-2	5.90e-3
	X-L-L	40.5	193.9	20.4	***1.99e-2***	***1.30e-2***	5.27e-3
	X-X-L	38.8	161.9	3.5	3.45e-2	2.86e-2	2.05e-2
20	K-K-K	38.8	164.0	7.2	2.75e-2	1.96e-2	1.12e-2
	K-L-K	40.3	184.2	14.9	1.69e-2	7.69e-3	3.69e-3
	K-L-L	40.2	182.3	15.2	1.46e-2	6.53e-3	***2.61e-3***
	K-K-L	38.5	160.7	6.0	2.77e-2	1.92e-2	9.17e-3
	X-X-X	38.8	165.8	8.9	2.18e-2	1.37e-2	7.83e-3
	X-L-X	40.0	180.3	16.4	8.78e-3	5.92e-3	3.95e-3
	X-L-L	40.1	181.9	17.9	***8.33e-3***	***5.46e-3***	4.18e-3
	X-X-L	40.0	178.7	8.6	2.49e-2	1.51e-2	7.68e-3
40	K-K-K	38.5	161.1	7.0	1.56e-2	8.87e-3	4.15e-3
	K-L-K	39.3	171.6	10.7	6.31e-3	4.66e-3	***2.92e-3***
	K-L-L	39.4	171.3	11.5	6.93e-3	4.68e-3	3.22e-3
	K-K-L	38.2	158.2	7.0	1.68e-2	8.69e-3	4.23e-3
	X-X-X	39.1	167.6	7.8	7.15e-3	6.13e-3	4.47e-3
	X-L-X	40.0	178.7	11.4	5.86e-3	4.21e-3	2.97e-3
	X-L-L	40.0	179.5	9.1	***5.81e-3***	***4.17e-3***	3.40e-3
	X-X-L	39.0	168.3	7.9	6.83e-3	5.24e-3	3.92e-3
80	K-K-K	37.1	146.2	4.8	5.12e-3	4.18e-3	3.28e-3
	K-L-K	38.4	159.5	6.5	5.94e-3	4.17e-3	3.00e-3
	K-L-L	37.7	152.2	6.7	5.09e-3	4.22e-3	2.87e-3
	K-K-L	36.9	142.7	4.7	9.40e-3	5.63e-3	3.50e-3
	X-X-X	36.9	144.2	4.5	6.08e-3	4.91e-3	4.05e-3
	X-L-X	38.0	155.5	6.5	**4.33e-3**	**3.68e-3**	**2.45e-3**
	X-L-L	38.9	164.4	5.8	5.38e-3	3.83e-3	3.03e-3
	X-X-L	37.8	152.6	5.0	5.32e-3	4.44e-3	3.76e-3

Table 3: Statistics of best genomes over 20 repeats on the c172 dataset. Weight initialization and inheritance type are labeled by *initial genome strategy-crossover strategy-mutation strategy*, so *e.g.*, *K-L-K* would use Kaiming for the initial genomes, Lamarckian on crossover operations, and Kaiming for components generated by mutation.

memory cells uniformly at random. Recurrent connections could span any time-skip generated randomly between $\mathcal{U}(1, 10)$. Backpropagation (BP) through time was run with a learning rate of $\eta = 0.001$ and used Nesterov momentum with $\mu = 0.9$. For the memory cells with forget gates, the forget gate bias had a value of 1.0 added to it (motivated by [14]). To prevent exploding gradients, gradient scaling [21] was used when the norm of the gradient exceeded a threshold of 1.0. To combat vanishing gradients, gradient boosting (the opposite of scaling) was used when the gradient norm was below 0.05. These parameters have been selected by hand-tuning during the prior experience.

Effect of Weight Inheritance on RNN Training Time: Our hypotheses were that *i)*, utilizing Lamarckian weight inheritance would provide performance improvements over Xavier and Kaiming initialization, and *ii)*, it could potentially allow for networks to be effectively evolved using fewer BP epochs per genome. To provide a comprehensive exploration, we set up experiments where the initial genomes initialized weights with the Xavier and Kaiming strategies (as the Lamarckian strategies could not yet be used). In these experiments, we tested the combinations of the two Lamarckian strategies with the initial weight inheritance strategy. Note that when Lamarckian weight inheritance is used for crossover and Xavier for weight inheritance, this is identical to the Lamarckain strategy used by [23], so their strategy was also investigated.

BP epochs of 1, 5, 10, 20, 40 and 80 per genome generated were examined. To have a fair comparison between the test cases, the total number of BP epochs for each search was held fixed at $200k$ for each test. This resulted in the total number of genomes generated during evolution for the tests being $200k$, $40k$, $20k$, $10k$, $5k$ and $2.5k$ respectively. In total, for each initial weight strategy (Kaiming and Xavier) there were 72 different experiments done with different BP epochs and strategies for crossover and mutation weight initialization for the coal fired power plant, wind turbine, and aviation datasets. All the experiments were repeated 20 times, allowing the Mann–Whitney U-test for statistical significance.

Figure 1 presents box plots of the best genome fitness from 20 repeats of each of the experiments using Xavier and Kaiming weight initialization and their combinations with the Lamarckian strategies. These results are summarized in Tables 1, 2 and 3, which present the best, average, and worst global best genome mean average error (MAE) at the end of the 20 repeated tests for each experiment performed on coal, wind, and aviation datasets. The best performing experiments are highlighted in bold and italics for each number of BP epochs, and the overall best experiment is highlighted in bold.

In the average case, for all three datasets, all but one best performing genomes were found using Lamarckian weight inheritance on crossover, and further all but one of the best cases utilized Lamarckian weight inheritance for crossover. Only one case used Xavier for weight initialization, inheritance for crossover and mutation in coal dataset that achieved the average best when the BP epochs was 1. All other best and average cases used Lamarckian weight initialize for crossover, mutation or both. Generally, the more epochs used in BP training in EXAMM, the better the validation MSE. In all of the three datasets the average

Cessna 172 - Kaiming

BP Epochs	Type	K-K-K	K-L-K	K-L-L	K-K-L
1	K-K-K	/	**0.0000**	**0.0000**	0.1427
	K-L-K	**0.0000**	/	0.4302	**0.0000**
	K-L-L	**0.0000**	0.4302	/	**0.0000**
	K-K-L	0.1427	**0.0000**	**0.0000**	/
5	K-K-K	/	**0.0000**	**0.0000**	0.0702
	K-L-K	**0.0000**	/	**0.0283**	**0.0000**
	K-L-L	**0.0000**	**0.0283**	/	**0.0000**
	K-K-L	0.0702	**0.0000**	**0.0000**	/
10	K-K-K	/	**0.0000**	**0.0000**	0.1897
	K-L-K	**0.0000**	/	0.2285	**0.0000**
	K-L-L	**0.0000**	0.2285	/	**0.0000**
	K-K-L	0.1897	**0.0000**	**0.0000**	/
20	K-K-K	/	**0.0000**	**0.0000**	0.3676
	K-L-K	**0.0000**	/	0.1042	**0.0000**
	K-L-L	**0.0000**	0.1042	/	**0.0000**
	K-K-L	0.3676	**0.0000**	**0.0000**	/
40	K-K-K	/	**0.0000**	**0.0000**	0.3474
	K-L-K	**0.0000**	/	0.4516	**0.0000**
	K-L-L	**0.0000**	0.4516	/	**0.0000**
	K-K-L	0.3474	**0.0000**	**0.0000**	/
80	K-K-K	/	0.4946	0.4946	**0.0001**
	K-L-K	0.4946	/	0.2714	**0.0000**
	K-L-L	0.4946	0.2714	/	**0.0000**
	K-K-L	**0.0001**	**0.0000**	**0.0000**	/

Cessna 172 - Xavier

BP Epochs	Type	X-X-X	X-L-X	X-L-L	X-X-L
1	X-X-X	/	**0.0010**	**0.0008**	0.4516
	X-L-X	**0.0010**	/	0.4409	**0.0006**
	X-L-L	**0.0008**	0.4409	/	**0.0006**
	X-X-L	0.4516	**0.0006**	**0.0006**	/
5	X-X-X	/	**0.0000**	**0.0000**	0.1824
	X-L-X	**0.0000**	/	0.2452	**0.0000**
	X-L-L	**0.0000**	0.2452	/	**0.0000**
	X-X-L	0.1824	**0.0000**	**0.0000**	/
10	X-X-X	/	**0.0000**	**0.0000**	0.1971
	X-L-X	**0.0000**	/	0.3375	**0.0000**
	X-L-L	**0.0000**	0.3375	/	**0.0000**
	X-X-L	0.1971	**0.0000**	**0.0000**	/
20	X-X-X	/	**0.0000**	**0.0000**	0.1552
	X-L-X	**0.0000**	/	0.0739	**0.0000**
	X-L-L	**0.0000**	0.0739	/	**0.0000**
	X-X-L	0.1552	**0.0000**	**0.0000**	/
40	X-X-X	/	**0.0000**	**0.0000**	**0.0015**
	X-L-X	**0.0000**	/	0.3474	**0.0024**
	X-L-L	**0.0000**	0.3474	/	**0.0002**
	X-X-L	**0.0015**	**0.0024**	**0.0002**	/
80	X-X-X	/	**0.0000**	**0.0000**	**0.0090**
	X-L-X	**0.0000**	/	0.3676	**0.0000**
	X-L-L	**0.0000**	0.3676	/	**0.0006**
	X-X-L	**0.0090**	**0.0000**	**0.0006**	/

Coal - Kaiming

BP Epochs	Type	K-K-K	K-L-K	K-L-L	K-K-L
1	K-K-K	/	**0.0000**	**0.0000**	**0.0000**
	K-L-K	**0.0000**	/	**0.0360**	0.1366
	K-L-L	**0.0000**	**0.0360**	/	0.3575
	K-K-L	**0.0000**	0.1366	0.3575	/
5	K-K-K	/	**0.0036**	**0.0000**	**0.0429**
	K-L-K	**0.0036**	/	**0.0057**	0.1308
	K-L-L	**0.0000**	**0.0057**	/	**0.0010**
	K-K-L	**0.0429**	0.1308	**0.0010**	/
10	K-K-K	/	**0.0024**	**0.0001**	**0.0033**
	K-L-K	**0.0024**	/	**0.0120**	0.3779
	K-L-L	**0.0001**	**0.0120**	/	0.1427
	K-K-L	**0.0033**	0.3779	0.1427	/
20	K-K-K	/	**0.0137**	**0.0006**	0.1552
	K-L-K	**0.0137**	/	**0.0234**	0.1427
	K-L-L	**0.0006**	**0.0234**	/	**0.0077**
	K-K-L	0.1552	0.1427	**0.0077**	/
40	K-K-K	/	**0.0072**	**0.0001**	0.2047
	K-L-K	**0.0072**	/	**0.0004**	0.0994
	K-L-L	**0.0001**	**0.0004**	/	**0.0005**
	K-K-L	0.2047	0.0994	**0.0005**	/
80	K-K-K	/	**0.0003**	**0.0001**	**0.0360**
	K-L-K	**0.0003**	/	0.4302	0.1366
	K-L-L	**0.0001**	0.4302	/	0.0903
	K-K-L	**0.0360**	0.1366	0.0903	/

Coal - Xavier

BP Epochs	Type	X-X-X	X-L-X	X-L-L	X-X-L
1	X-X-X	/	0.1754	0.3779	**0.0158**
	X-L-X	0.1754	/	0.2896	0.0666
	X-L-L	0.3779	0.2896	/	**0.0481**
	X-X-L	**0.0158**	0.0666	**0.0481**	/
5	X-X-X	/	0.1617	0.1824	0.3474
	X-L-X	0.1617	/	0.4409	0.2538
	X-L-L	0.1824	0.4409	/	0.2896
	X-X-L	0.3474	0.2538	0.2896	/
10	X-X-X	/	**0.0077**	**0.0077**	0.3474
	X-L-X	**0.0077**	/	0.2367	**0.0249**
	X-L-L	**0.0077**	0.2367	/	**0.0158**
	X-X-L	0.3474	**0.0249**	**0.0158**	/
20	X-X-X	/	0.2625	0.3375	0.4730
	X-L-X	0.2625	/	0.4409	0.2896
	X-L-L	0.3375	0.4409	/	0.3575
	X-X-L	0.4730	0.2896	0.3575	/
40	X-X-X	/	**0.0053**	**0.0002**	0.1552
	X-L-X	**0.0053**	/	**0.0455**	0.1617
	X-L-L	**0.0002**	**0.0455**	/	**0.0266**
	X-X-L	0.1552	0.1617	**0.0266**	/
80	X-X-X	/	**0.0283**	**0.0004**	0.3277
	X-L-X	**0.0283**	/	**0.0283**	**0.0266**
	X-L-L	**0.0004**	**0.0283**	/	**0.0002**
	X-X-L	0.3277	**0.0266**	**0.0002**	/

Wind - Kaiming

BP Epochs	Type	K-K-K	K-L-K	K-L-L	K-K-L
1	K-K-K	/	**0.0000**	**0.0000**	**0.0158**
	K-L-K	**0.0000**	/	0.0509	**0.0000**
	K-L-L	**0.0000**	0.0509	/	**0.0000**
	K-K-L	**0.0158**	**0.0000**	**0.0000**	/
5	K-K-K	/	**0.0000**	**0.0000**	0.2538
	K-L-K	**0.0000**	/	**0.0072**	**0.0000**
	K-L-L	**0.0000**	**0.0072**	/	**0.0000**
	K-K-L	0.2538	**0.0000**	**0.0000**	/
10	K-K-K	/	**0.0000**	**0.0000**	0.2896
	K-L-K	**0.0000**	/	0.4623	**0.0000**
	K-L-L	**0.0000**	0.4623	/	**0.0000**
	K-K-L	0.2896	**0.0000**	**0.0000**	/
20	K-K-K	/	**0.0000**	**0.0000**	0.0739
	K-L-K	**0.0000**	/	**0.0429**	**0.0005**
	K-L-L	**0.0000**	**0.0429**	/	**0.0001**
	K-K-L	0.0739	**0.0005**	**0.0001**	/
40	K-K-K	/	**0.0000**	**0.0000**	0.3882
	K-L-K	**0.0000**	/	0.4946	**0.0000**
	K-L-L	**0.0000**	0.4946	/	**0.0000**
	K-K-L	0.3882	**0.0000**	**0.0000**	/
80	K-K-K	/	**0.0000**	**0.0000**	0.1824
	K-L-K	**0.0000**	/	0.2285	**0.0000**
	K-L-L	**0.0000**	0.2285	/	**0.0000**
	K-K-L	0.1824	**0.0000**	**0.0000**	/

Wind - Xavier

BP Epochs	Type	X-X-X	X-L-X	X-L-L	X-X-L
1	X-X-X	/	**0.0000**	**0.0000**	**0.0024**
	X-L-X	**0.0000**	/	**0.0266**	**0.0000**
	X-L-L	**0.0000**	**0.0266**	/	**0.0000**
	X-X-L	**0.0024**	**0.0000**	**0.0000**	/
5	X-X-X	/	**0.0000**	**0.0000**	0.3575
	X-L-X	**0.0000**	/	**0.0008**	**0.0000**
	X-L-L	**0.0000**	**0.0008**	/	**0.0000**
	X-X-L	0.3575	**0.0000**	**0.0000**	/
10	X-X-X	/	**0.0000**	**0.0000**	**0.0405**
	X-L-X	**0.0000**	/	0.0818	**0.0008**
	X-L-L	**0.0000**	0.0818	/	**0.0001**
	X-X-L	**0.0405**	**0.0008**	**0.0001**	/
20	X-X-X	/	**0.0000**	**0.0000**	0.1197
	X-L-X	**0.0000**	/	**0.0133**	**0.0057**
	X-L-L	**0.0000**	**0.0133**	/	**0.0001**
	X-X-L	0.1197	**0.0057**	**0.0001**	/
40	X-X-X	/	**0.0000**	**0.0000**	0.3084
	X-L-X	**0.0000**	/	0.2452	**0.0000**
	X-L-L	**0.0000**	0.2452	/	**0.0000**
	X-X-L	0.3084	**0.0000**	**0.0000**	/
80	X-X-X	/	**0.0000**	**0.0000**	0.2124
	X-L-X	**0.0000**	/	0.1366	**0.0000**
	X-L-L	**0.0000**	0.1366	/	**0.0000**
	X-X-L	0.2124	**0.0000**	**0.0000**	/

Table 4: Mann–Whitney U test p-values comparing Kaiming and Xavier weight initialization and inheritance strategies for the three datasets. p-values in bold indicate a statistically significant difference with $\alpha = 0.05$. Weight initialization and inheritance type are labeled by *initial genome strategy-crossover strategy-mutation strategy*, so *e.g.*, *K-L-K* would use Kaiming for the initial genomes, Lamarckian on crossover operations, and Kaiming for components generated by mutation.

global best results come from BP epoch 80. The global best genome in coal dataset is found when using 20 BP epochs, whereas the global best genomes in the other two datasets are all found using 80 epochs.

Further strengthening these results on the benefit of the Lamarckian strategy, Table 4 presents Mann-Whitney U tests of statistical significance comparing the varying weight initialization tests against each other for each number of BP epochs. For the wind turbine data, we see very strong statistical significance in most cases, highlighting that the improvements from the Lamarckian strategy. While the statistical significance is less strong in some cases on the coal dataset, interestingly the statistical significance increases with the number of BP epochs utilized perhaps due to the fact more training time enables quicker convergence to local or global minima.

For the coal and wind turbine dataset, increasing the number of BP epochs does not significantly improve the validation MSE. It means utilizing Lamarckian weight inheritance provides the ability to reduce the number of BP epochs required for training, which in turn allows for more time to be spent evolving the RNN architectures. However, the C172 dataset results generally improve when using more BP epochs, perhaps due to the added complexity of having multiple output parameters.

Effect of Evolution on Genome's Weight Performance: Further tests were done to see if Lamarckian weight inheritance and the neuroevolutionary process provided any other benefits over training RNNs for long periods of time. The best RNN architectures from each of the 20 repeats from all the experiments were retrained with their weights reinitialized with both the Kaiming and Xavier methods. Those architectures were then retrained for a long time (3000 epochs), and the best and average validation MSE of the retrained results were compared with the best and average validation MSE results.

Table 5 shows the best and average EXAMM validation results compared to the best and average retrained results for all the experiment types across all three datasets. In the best case, for the coal and wind dataset, almost all of the results at the end of EXAMM outperform the retrained RNNs, and in the cases where it does not the results are quite close. The C172 data proved more challenging here and outperformed the final EXAMM results, which may indicate that it would benefit from even longer training epochs when generating genomes. Additionally, when the BP epochs were 40, EXAMM performed the best on both the coal and wind dataset, which shows that utilizing Lamarckian strategies can provide strong results using fewer epochs, and also that the neuroevolutionary process is providing additional benefits, as even when these networks are retrained for a very long period of time, they still perform worse than the ones with evolved weights.

In the average case, EXAMM strongly outperforms the retrained networks for all the datasets in all but one case. This shows that EXAMM's Lamarckian weight inheritance method typically evolves genomes that do not require further training, as opposed to other neural architecture search strategies which first find the architecture and then need to train it for a significant period of time.

		Coal		Wind		C172		Coal		Wind		C172	
BP Epochs	Type	EXAMM Best MSE	Retrain Best MSE	EXAMM Best MSE	Retrain Best MSE	EXAMM Best MSE	Retrain Best MSE	EXAMM Avg MSE	Retrain Avg MSE	EXAMM Avg MSE	Retrain Avg MSE	EXAMM Avg MSE	Retrain Avg MSE
1	K-K-K	**1.050e-3**	2.023e-3	4.185e-3	**2.765e-3**	2.579e-2	**1.061e-3**	**1.423e-3**	1.957e-2	**4.950e-3**	5.535e-2	**3.600e-2**	5.545e-2
	K-L-K	**6.601e-4**	1.106e-3	3.016e-3	**2.769e-3**	1.264e-2	**1.285e-3**	**9.839e-4**	5.197e-2	**3.515e-3**	6.745e-2	**2.528e-2**	7.750e-2
	K-L-L	5.358e-4	**5.170e-4**	2.982e-3	**2.645e-3**	1.279e-2	**1.055e-3**	**8.395e-4**	6.032e-2	**3.371e-3**	4.936e-2	2.538e-2	**1.389e-2**
	K-K-L	**5.033e-4**	7.330e-4	3.850e-3	**2.837e-3**	2.726e-2	**1.085e-3**	**8.828e-4**	1.667e-2	**4.554e-3**	4.265e-2	**3.525e-2**	5.049e-2
	X-X-X	**5.300e-4**	7.230e-4	4.040e-3	**2.775e-3**	2.739e-2	**1.377e-3**	**7.509e-4**	9.739e-2	**5.074e-3**	5.757e-2	**3.544e-2**	1.383e-1
	X-L-X	**4.875e-4**	1.159e-3	**2.687e-3**	2.730e-3	1.257e-2	**1.482e-3**	**8.184e-4**	4.842e-2	**3.573e-3**	5.481e-2	**2.965e-2**	1.186e-1
	X-L-L	**4.753e-4**	1.208e-3	**2.858e-3**	3.081e-3	2.361e-2	**1.569e-3**	**7.845e-4**	2.124e-1	**3.381e-3**	1.012e-1	**3.082e-2**	1.063e-1
	X-X-L	**5.321e-4**	1.494e-3	3.750e-3	**2.590e-3**	2.788e-2	**1.839e-3**	**9.112e-4**	3.213e-2	**4.502e-3**	4.077e-2	**3.562e-2**	1.497e-1
5	K-K-K	7.578e-4	**5.670e-4**	2.876e-3	**2.691e-3**	2.915e-2	**2.020e-3**	**1.068e-3**	3.366e-2	**3.166e-3**	8.689e-3	**3.754e-2**	1.097e-1
	K-L-K	6.561e-4	**5.190e-4**	**2.581e-3**	2.644e-3	1.865e-2	**1.492e-3**	**8.882e-4**	5.872e-3	**2.869e-3**	1.033e-2	**2.693e-2**	4.403e-2
	K-L-L	**6.524e-4**	7.460e-4	2.624e-3	**2.547e-3**	6.451e-3	**1.451e-3**	**8.361e-4**	1.816e-2	**2.774e-3**	1.114e-2	**2.277e-2**	5.902e-2
	K-K-L	**4.754e-4**	5.130e-4	2.869e-3	**2.680e-3**	2.497e-2	**1.648e-3**	**8.165e-4**	3.798e-2	**3.155e-3**	3.662e-3	**3.544e-2**	6.933e-2
	X-X-X	**5.292e-4**	7.540e-4	2.778e-3	**2.606e-3**	2.855e-2	**1.605e-3**	**8.212e-4**	1.293e-2	**3.208e-3**	6.376e-3	**3.599e-2**	1.186e-1
	X-L-X	**5.415e-4**	8.040e-4	2.752e-3	**2.730e-3**	7.350e-3	**1.255e-3**	**7.704e-4**	5.240e-2	**2.898e-3**	1.876e-2	**2.411e-2**	5.466e-2
	X-L-L	**4.890e-4**	5.520e-4	**2.598e-3**	2.675e-3	1.128e-2	**1.354e-3**	**7.728e-4**	8.748e-2	**2.770e-3**	1.674e-2	**2.355e-2**	9.598e-2
	X-X-L	**5.107e-4**	6.400e-4	2.823e-3	**2.693e-3**	2.913e-2	**1.773e-3**	**8.217e-4**	4.371e-3	**3.229e-3**	1.831e-2	**3.478e-2**	1.426e-1
10	K-K-K	**5.510e-4**	5.850e-4	2.680e-3	**2.671e-3**	2.403e-2	**1.345e-3**	**9.886e-4**	7.516e-2	**2.942e-3**	8.848e-3	**3.111e-2**	7.538e-2
	K-L-K	5.447e-4	**4.860e-4**	**2.608e-3**	2.631e-3	6.208e-3	**1.389e-3**	**7.801e-4**	9.670e-3	**2.720e-3**	8.456e-3	**1.516e-2**	8.258e-2
	K-L-L	**4.871e-4**	5.650e-4	**2.492e-3**	2.596e-3	3.818e-3	**1.049e-3**	**6.903e-4**	4.617e-3	**2.706e-3**	8.780e-3	**1.328e-2**	7.331e-2
	K-K-L	**4.950e-4**	5.880e-4	2.711e-3	**2.628e-3**	2.121e-2	**1.710e-3**	**7.784e-4**	5.204e-3	**2.913e-3**	6.111e-3	**3.022e-2**	5.077e-2
	X-X-X	**4.965e-4**	5.140e-4	**2.631e-3**	2.688e-3	1.774e-2	**2.034e-3**	**8.063e-4**	4.659e-3	**2.984e-3**	1.973e-2	**2.741e-2**	6.479e-2
	X-L-X	**4.958e-4**	5.210e-4	**2.491e-3**	2.640e-3	5.895e-3	**1.737e-3**	**6.777e-4**	3.834e-3	**2.769e-3**	6.102e-3	**1.354e-2**	1.335e-1
	X-L-L	**5.008e-4**	5.500e-4	**2.514e-3**	2.642e-3	5.270e-3	**1.785e-3**	**6.900e-4**	2.797e-2	**2.725e-3**	4.701e-3	**1.295e-2**	1.565e-1
	X-X-L	**5.180e-4**	5.970e-4	2.724e-3	**2.650e-3**	2.055e-2	**1.720e-3**	**7.787e-4**	3.139e-3	**2.916e-3**	3.883e-3	**2.856e-2**	1.019e-1
20	K-K-K	**5.796e-4**	8.090e-4	2.695e-3	**2.685e-3**	1.116e-2	**2.078e-3**	**8.617e-4**	3.057e-3	**2.831e-3**	3.366e-3	**1.957e-2**	4.263e-2
	K-L-K	5.381e-4	**4.880e-4**	**2.517e-3**	2.593e-3	3.691e-3	**1.175e-3**	**7.372e-4**	3.532e-3	**2.653e-3**	2.970e-3	**7.690e-3**	8.499e-2
	K-L-L	**4.720e-4**	5.610e-4	**2.532e-3**	2.611e-3	2.607e-3	**1.181e-3**	**6.574e-4**	4.018e-2	**2.626e-3**	3.144e-3	**6.533e-3**	7.914e-2
	K-K-L	**4.947e-4**	6.200e-4	**2.576e-3**	2.655e-3	9.174e-3	**1.649e-3**	**8.109e-4**	5.459e-3	**2.775e-3**	3.727e-3	**1.922e-2**	4.841e-2
	X-X-X	**5.189e-4**	6.160e-4	2.684e-3	**2.611e-3**	7.828e-3	**1.679e-3**	**7.268e-4**	2.572e-3	**2.824e-3**	3.152e-3	**1.371e-2**	6.488e-2
	X-L-X	**5.040e-4**	5.190e-4	**2.589e-3**	2.630e-3	3.947e-3	**2.138e-3**	**6.956e-4**	5.919e-2	**2.681e-3**	2.978e-3	**5.919e-3**	4.296e-2
	X-L-L	**4.877e-4**	5.060e-4	**2.462e-3**	2.634e-3	4.185e-3	**1.114e-3**	**6.929e-4**	3.719e-3	**2.626e-3**	3.376e-3	**5.465e-3**	2.547e-2
	X-X-L	**5.056e-4**	7.820e-4	**2.577e-3**	2.641e-3	7.675e-3	**1.870e-3**	**7.118e-4**	5.389e-2	**2.785e-3**	3.336e-3	**1.508e-2**	5.384e-2
40	K-K-K	5.181e-4	**4.780e-4**	**2.611e-3**	2.625e-3	4.152e-3	**2.289e-3**	**8.470e-4**	2.576e-3	**2.741e-3**	2.934e-3	**8.872e-3**	6.760e-2
	K-L-K	**4.772e-4**	4.810e-4	**2.513e-3**	2.630e-3	2.915e-3	**1.919e-3**	**7.569e-4**	3.306e-3	**2.586e-3**	3.099e-3	**4.663e-3**	4.319e-2
	K-L-L	**5.261e-4**	5.770e-4	**2.456e-3**	2.597e-3	3.221e-3	**1.521e-3**	**6.550e-4**	5.689e-3	**2.577e-3**	6.016e-3	**4.679e-3**	8.081e-2
	K-K-L	5.340e-4	**5.120e-4**	**2.566e-3**	2.626e-3	4.230e-3	**1.497e-3**	**8.090e-4**	1.831e-2	**2.727e-3**	3.921e-3	**8.687e-3**	6.318e-2
	X-X-X	5.060e-4	**4.770e-4**	**2.579e-3**	2.627e-3	4.471e-3	**1.570e-3**	**8.342e-4**	2.633e-3	**2.756e-3**	3.281e-3	**6.127e-3**	2.971e-2
	X-L-X	5.374e-4	**4.940e-4**	**2.565e-3**	2.647e-3	2.973e-3	**1.812e-3**	**7.254e-4**	2.680e-3	**2.612e-3**	3.139e-3	**4.211e-3**	3.815e-2
	X-L-L	**5.120e-4**	5.130e-4	**2.513e-3**	2.630e-3	3.400e-3	**1.468e-3**	**6.718e-4**	1.878e-3	**2.602e-3**	3.142e-3	**4.167e-3**	3.480e-2
	X-X-L	**5.361e-4**	5.660e-4	**2.603e-3**	2.637e-3	3.917e-3	**1.399e-3**	**7.795e-4**	3.615e-3	**2.745e-3**	5.542e-3	**5.235e-3**	3.657e-2
80	K-K-K	**6.316e-4**	6.710e-4	**2.578e-3**	2.603e-3	3.284e-3	**1.830e-3**	**8.708e-4**	2.491e-3	**2.703e-3**	3.676e-3	**4.184e-3**	1.662e-2
	K-L-K	**4.749e-4**	5.650e-4	**2.553e-3**	2.615e-3	3.001e-3	**1.213e-3**	**6.778e-4**	2.883e-3	**2.586e-3**	3.066e-3	**4.172e-3**	3.241e-2
	K-L-L	4.812e-4	**4.800e-4**	**2.452e-3**	2.616e-3	2.867e-3	**1.599e-3**	**6.758e-4**	3.917e-3	**2.574e-3**	2.889e-3	**4.219e-3**	3.761e-2
	K-K-L	**4.928e-4**	5.820e-4	2.644e-3	**2.608e-3**	3.495e-3	**1.776e-3**	**7.639e-4**	8.406e-2	**2.725e-3**	2.911e-3	**5.633e-3**	3.731e-2
	X-X-X	**5.127e-4**	6.750e-4	2.603e-3	**2.542e-3**	4.051e-3	**1.441e-3**	**7.926e-4**	3.806e-3	**2.715e-3**	2.928e-3	**4.908e-3**	2.897e-2
	X-L-X	**4.790e-4**	4.990e-4	**2.510e-3**	2.621e-3	2.450e-3	**1.695e-3**	**6.974e-4**	3.025e-3	**2.595e-3**	3.044e-3	**3.683e-3**	3.050e-2
	X-L-L	**4.954e-4**	5.010e-4	**2.478e-3**	2.641e-3	3.033e-3	**1.196e-3**	**6.186e-4**	2.436e-3	**2.580e-3**	3.307e-3	**3.832e-3**	4.158e-2
	X-X-L	**4.883e-4**	6.460e-4	2.623e-3	**2.619e-3**	3.765e-3	**1.604e-3**	**7.825e-4**	3.064e-3	**2.707e-3**	3.016e-3	**4.443e-3**	2.450e-2

Table 5: EXAMM best MSE and best genome retrain best MSE, the better result is marked in bold for the same dataset and same weight initialization and weight inheritance type. Weight initialization and inheritance type are labeled by *initial genome strategy-crossover strategy-mutation strategy*, so *e.g.*, *K-L-K* would use Kaiming for the initial genomes, Lamarckian on crossover operations, and Kaiming for components generated by mutation.

5 Conclusions

This work is an experimental study on the effects of weight initialization and weight inheritance in neuroevolution. It compares the well known Kaiming and Xavier weight initialization strategies to two Lamarckian weight inheritance strategies, once based on recombining parental weights during crossover, and another using statistical information of parental weights to assign new weights in mutation operations. This is done in the context of the Evolutionary eXploration of Augmenting Memory Models (EXAMM) neuroevolution algorithm, which progressively evolves and trains RNNs for time series data prediction using a direct encoding strategy. Experiments were done using three large scale real world time series data sets, one generated from a coal fired power plant, one from a wind turbine, and one from aviation flight.

A comprehensive suite of tests was run, finding with statistical significance that the Lamarckian strategies outperform Xavier and Kaiming weight initialization for generating new RNNs through EXAMM. Further, these Lamarckian strategies are also shown to be able to reduce the number of backpropagation epochs required to train the generated neural networks, allowing the neuroevolution algorithm to be able to perform more architectural evolution. These results validate a commonly held view that Lamarckian weight inheritance strategies can improve the performance of neuroevolution algorithms [23, 7, 19], which to the authors knowledge has not been rigorously compared to state-of-the-art Xavier and Kaiming weight initialization.

Further, the weights found during the neuroevolution process with Lamarckian weight inheritance were found to provide better results than when the best found evolved architectures were retrained from random Xavier or Kaiming initializations for a large number of epochs. This highlights that the Lamarckian evolutionary process is providing additional benefit to the selection of weights and how the weight search space is traversed, showing that in most cases, the networks and weights generated by EXAMM and Lamarckian weight inheritance do not require further training, unlike many other neural architecture search strategies which first find an architecture and then need to continue to train it for a longer period of time.

The Lamarckian strategies presented are generic and can be applied to any direct encoding neuroevolution algorithm. Future work will expand these results to convolutional neural networks as well as recurrent neural networks used for natural language processing tasks (which tend to have wider but shallower architectures). Furthermore, this weight initialization and inheritance strategy can also be applied to and tested on other NE algorithms. Given these results as motivation, investigating new Lamarckian strategies to further enhance performance will also be done.

Acknowledgements

Most of the computation of this research was done on the high performance computing clusters of Research Computing at Rochester Institute of Technology [25].

We would like to thank the Research Computing team for their assistance and the support they generously offered to ensure that the heavy computation this study required was available.

References

1. Aly, A., Weikersdorfer, D., Delaunay, C.: Optimizing deep neural networks with multiple search neuroevolution. arXiv preprint arXiv:1901.05988 (2019)
2. Camero, A., Toutouh, J., Alba, E.: Low-cost recurrent neural network expected performance evaluation. arXiv preprint arXiv:1805.07159 (2018)
3. Camero, A., Toutouh, J., Alba, E.: A specialized evolutionary strategy using mean absolute error random sampling to design recurrent neural networks. arXiv preprint arXiv:1909.02425 (2019)
4. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
5. Collins, J., Sohl-Dickstein, J., Sussillo, D.: Capacity and trainability in recurrent neural networks. arXiv preprint arXiv:1611.09913 (2016)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation **6**(2), 182–197 (2002)
7. Desell, T.: Accelerating the evolution of convolutional neural networks with node-level mutations and epigenetic weight initialization. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 157–158. ACM (2018)
8. 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)
9. ElSaid, A., El Jamiy, F., Higgins, J., Wild, B., Desell, T.: Optimizing long short-term memory recurrent neural networks using ant colony optimization to predict turbine engine vibration. Applied Soft Computing (2018)
10. ElSaid, A., Karns, J., Lyu, Z., Krutz, D., Ororbia, A., Desell, T.: Improving neuroevolutionary transfer learning of deep recurrent neural networks through network-aware adaptation. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. pp. 315–323 (2020)
11. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256 (2010)
12. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
14. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: International Conference on Machine Learning. pp. 2342–2350 (2015)
15. Ku, K.W., Mak, M.W.: Exploring the effects of lamarckian and baldwinian learning in evolving recurrent neural networks. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97). pp. 617–621. IEEE (1997)

16. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 19–34 (2018)
17. Liu, Y., Sun, Y., Xue, B., Zhang, M., Yen, G.: A survey on evolutionary neural architecture search. arXiv preprint arXiv:2008.10937 (2020)
18. Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., Banzhaf, W.: Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 419–427 (2019)
19. 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
20. 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
21. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning. pp. 1310–1318 (2013)
22. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268 (2018)
23. Prellberg, J., Kramer, O.: Lamarckian evolution of convolutional neural networks. In: International Conference on Parallel Problem Solving from Nature. pp. 424–435. Springer (2018)
24. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q., Kurakin, A.: Large-scale evolution of image classifiers. arXiv preprint arXiv:1703.01041 (2017)
25. Rochester Institute of Technology: Research computing services (2019). https://doi.org/10.34788/0S3G-QD15, https://www.rit.edu/researchcomputing/
26. Stanley, K., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
27. Stanley, K.O., Clune, J., Lehman, J., Miikkulainen, R.: Designing neural networks through neuroevolution. Nature Machine Intelligence **1**(1), 24–35 (2019)
28. Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. Artificial life **15**(2), 185–212 (2009)
29. Zhang, Q., Li, H.: Moea/d: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on evolutionary computation **11**(6), 712–731 (2007)
30. 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)