Automating Discovery of Physics-Informed Neural State Space Models via Learning and Evolution

Elliott Skomski ELLIOTT.SKOMSKI@PNNL.GOV
Ján Drgoňa JAN.DRGONA@PNNL.GOV
Aaron Tuor AARON.TUOR@PNNL.GOV
Pacific Northwest National Laboratory
Richland, WA, USA

Editors: A. Jadbabaie, J. Lygeros, G. J. Pappas, P. A. Parrilo, B. Recht, C. J. Tomlin, M. N. Zeilinger

Abstract

Recent works exploring deep learning application to dynamical systems modeling have demonstrated that embedding physical priors into neural networks can yield more effective, physically-realistic, and data-efficient models. However, in the absence of complete prior knowledge of a dynamical system's physical characteristics, determining the optimal structure and optimization strategy for these models can be difficult. In this work, we explore methods for discovering neural state space dynamics models for system identification. Starting with a design space of block-oriented state space models and structured linear maps with strong physical priors, we encode these components into a model genome alongside network structure, penalty constraints, and optimization hyperparameters. Demonstrating the overall utility of the design space, we employ an asynchronous genetic search algorithm that alternates between model selection and optimization and obtains accurate physically consistent models of three physical systems: an aerodynamics body, a continuous stirred tank reactor, and a two tank interacting system.

Keywords: Neuroevolution, neural networks, system identification, genetic algorithms

1. Introduction

Recent work has shown that given an appropriate design space, Neural Architecture Search (NAS) (Elsken et al., 2019) using evolutionary algorithms—so-called Neuroevolution (Floreano et al., 2008)—can discover models that meet or exceed the performance of expert-designed networks for complex computer vision (Liang et al., 2018; Real et al., 2019), natural language processing (So et al., 2019), and continuous control tasks (Gaier and Ha, 2019). In this work we bring together neural architecture search and works viewing neural networks from a dynamical systems perspective. We present a new NAS design space for the discovery of accurate control-oriented systems models from a host of physics-driven neural network components.

Several recent design spaces for NAS have composed generic neural network functional components while preserving inductive priors from successful state-of-the-art computer vision or natural language processing neural network architectures (Liu et al., 2020). On the other hand, as it applies to dynamical system identification, NAS with neuroevolutionary methods have typically used lower level components such as neurons and basic function compositions applied in black-box systems modeling without the benefit of physics-modeling based priors embedded in the design space (Al-

Mahasneh et al., 2017; Subudhi and Jena, 2011a,b; Yang et al., 2017; Ferariu and Burlacu, 2011; Ayala et al., 2020; Hatanaka et al., 2006; Yang et al., 2017; Gaier and Ha, 2019). Yet outside NAS research, advantageous inductive priors for dynamics modeling have recently been proposed by several key contributions which view recurrent and residual networks through the lens of traditional dynamical systems modeling. Prominent examples are guarantees on stability by constraining the deep network's linear maps using orthogonal (Mhammedi et al., 2017; Jia et al., 2019; Wang et al., 2020), spectral (Zhang et al., 2018), symplectic (Haber and Ruthotto, 2017), anti-symmetric (Chang et al., 2019), stochastic (Tuor et al., 2020), or Schur Decomposition (Kerg et al., 2019) parametrizations.

Our current work integrates these two orthogonal yet complementary lines of research—neural architecture search and dynamics-inspired neural network components—to expedite the discovery of effective dynamical systems models. We present a neural block dynamics design space that encompasses an extensive range of time-invariant, block-oriented state space models. These models are built from a library of neural components using structured linear maps to imbue models with strong priors for physical modeling, encouraging stability and data efficiency. In addition to structural hyperparameters, coefficients for multi-objective loss terms penalizing constraints violations, block interactions, trajectory smoothing, and prediction error are also included in the search space. We evaluate our design space with two search algorithms: Random Search (RS), and an Asynchronous Genetic Algorithm (AGA). RS has no evolutionary effects, whereas AGA is designed to maximize interaction between learning and evolution, discovering performant models in a directed and expeditious manner. We conduct system identification experiments for both search methods on three non-autonomous systems representing a range of dynamic behavior and achieve highly accurate models with physically consistent open-loop response for each.

2. Methods

Our objective in this work is to develop a neural network design space for system identification and dynamics modeling that is general enough to model a large extent of known systems and leverages inductive priors specific to dynamical systems modeling. This design space is intended to serve as both a substrate for expert-built neural dynamics models and as a basis for neural architecture search to discover architectures best suited for specific systems. To this end, we introduce a family of neural state space models built from components consisting of linear map parametrizations, activation functions, and neural network block components.

2.1. Structured Linear Maps

In addition to unstructured linear maps, our design space includes three structured linear map parametrizations which can introduce strong inductive biases and provide guarantees suitable for a large extent of known systems models. For sparsity-inducing priors, we employ a Lasso variant implemented with gradient descent as described in Bottou (2010). We include another matrix parametrization, Soft SVD, which enforces singular value constraints on linear maps. This is accomplished by parametrizing the matrix as a product $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$, initializing $\mathbf{U}$ and $\mathbf{V}$ as random orthogonal matrices, and introducing a regularization term to enforce orthogonality as parameters are updated. Bounds λ_{min} and λ_{max} are placed on the nonzero elements of the diagonal matrix $\mathbf{\Sigma}$ where:

$$\mathbf{\Sigma} = \text{diag}(\lambda_{\text{max}} - (\lambda_{\text{max}} - \lambda_{\text{min}}) \cdot \sigma(\mathbf{p})) \tag{1}$$

with $\mathbf{p}$ a randomly initialized vector and σ the elementwise logistic sigmoid. The final structured parametrization in the design space is the Perron-Frobenius (PF) map proposed by Tuor et al. (2020), which bounds the dominant eigenvalue of the matrix to guarantee stability of the learned system and global dynamic properties such as dissipation.

2.2. Activation Functions

The design space contains a library of two non-parametric and two parametric activation functions: Rectified Linear Units (ReLU), a common activation which clamps negative values to zero (Nair and Hinton, 2010); Gaussian Error Linear Units (GELU), which approximates the expected value of stochastic regularization (Hendrycks and Gimpel, 2016); Bendable Linear Units (BLU), which learns a continuous approximation of two piecewise linear functions (Godfrey, 2019); and Soft Exponential (SoftExp), which learns a function that interpolates between exponential and logarithmic functions (Godfrey and Gashler, 2015). For parametric activation functions, we use independent activations at each layer to allow models to learn activations for each phase of computation.

2.3. Neural Time-Invariant Block Dynamics Models

The general form of state space model we consider is composed of a state estimator f_o, state transition dynamics f, and observation dynamics f_y with learnable parameters θ. With $\mathbf{u}_k$ as control input at time k, and $\mathbf{y}_{k-N_p}, ..., \mathbf{y}_{k-1}, \mathbf{y}_k$ a sequence of initial observed variables of the system the model is of the form:

$$\hat{\mathbf{x}}^{\mathrm{e}}_k = f_o(\hat{\mathbf{y}}_{k-N_p}, ..., \hat{\mathbf{y}}_k; \theta_o) \tag{2a}$$
$$\hat{\mathbf{x}}_{k+1} = f(\hat{\mathbf{x}}^{\mathrm{e}}_k, \mathbf{u}_k; \theta_{xu}) \tag{2b}$$
$$\hat{\mathbf{y}}_{k+1} = f_y(\hat{\mathbf{x}}_{k+1}; \theta_y) \tag{2c}$$

It is common for the state dimension of the main transition dynamics to be unknown. From the deep learning perspective we can view $\hat{\mathbf{x}}$ as the hidden state of an RNN with cell function f, input $\mathbf{u}$, and output $\hat{\mathbf{y}}$. From a dynamics modeling perspective, when f is linear, f_o plays the role of a finite approximate lifting function from Koopman operator theory as suggested by Yeung et al. (2019). The library of block components for f, f_o, and f_y consists of linear maps (LMs), multi-layer perceptrons (MLPs), residual networks (rMLPs), and recurrent neural networks (RNNs) built from the parametrized linear maps and activation functions discussed in the previous sections. An additional factor of variation is introduced in f_o to account for time lag using a window of N_p past observations.

Within the general class of models, f may be a single neural network which takes as input the concatenated $\mathbf{u}$ and $\hat{\mathbf{x}}$ vectors. Or alternatively, lending more structure to the problem, interactions between the the control inputs, and principal state transition dynamics are modeled as a composition component blocks so that Equation 2b becomes:

$$\hat{\mathbf{x}}_{k+1} = f_x(\hat{\mathbf{x}}^{\mathrm{e}}_k; \theta_x) \circ f_u(\mathbf{u}_k; \theta_u) \tag{3}$$

where f_x and f_u are drawn from the library of block components and $\circ$ is an elementwise operator modeling the influence of exogenous inputs upon the estimated system state. Three operators

are considered: addition ($+$), multiplication ($\times$), and a learnable interpolation of addition and multiplication ($+/\times$) as given in Equation 5 of Godfrey and Gashler (2015).

The unstructured and structured model classes, which are high-level options in the design space, are denoted ***black-box*** and ***block*** respectively. The block-oriented formulation allows the representation of several classes of models commonly used in system identification. Choosing the addition operator, when both are f_x and f_u are linear, we have a basic linear time-invariant model. When only f_u is nonlinear, we have a Hammerstein model with neural network nonlinearity. When both f_u and f_y are nonlinear but not f_x, we have a Hammerstein-Weiner model. Making all blocks nonlinear provides a general neural block nonlinear model with nonlinear state transition dynamics.

2.4. Multi-Objective Loss

The state space models are trained with multi-objective loss functions informed by best practices in control-oriented data-driven system identification. Expressions for constituent objective terms are found in Equation 4. The principal loss function is mean squared error between predicted trajectory and ground-truth measurements over an N-step time horizon, $\mathcal{L}_y$ (Equation 4a). The model is given a sequence of N_p initial previous ground-truth measurements and future sequence of control inputs $\mathbf{U}$, then generates a series of predictions $\hat{\mathbf{y}}_1, ..., \hat{\mathbf{y}}_N$. Although not necessary, for simplicity we align the N_p past measurements given to the state estimator, and the N-step prediction horizon so that $N_p = N$. To promote alignment between the state estimator, f_o, and dynamics, f, we incorporate an additional arrival cost penalty, $\mathcal{L}_\text{e}$ (Equation 4b). We include an additional term, $\mathcal{L}_{dx}$, to ensure smooth state transitions regularizing the distance between successive states (Equation 4c). We employ the penalty method such that predictions remain within realistic bounds, enforcing this property by defining lower and upper bounds $\underline{y}$ and $\overline{y}$, then apply inequality constraints via a loss term $\mathcal{L}_y^\text{con}$ (Equation 4d). We can further constrain the influence of input map f_u on predicted states for block-structured models, defining lower and upper bounds $\underline{f_u}$ and $\overline{f_u}$ and using the same inequality constraint formulation as in $\mathcal{L}_y^\text{con}$ to create another loss term, $\mathcal{L}_{f_u}^\text{con}$. In this work, we set $\underline{y} = -0.2$, $\overline{y} = 1.2$, $\underline{f_u} = -0.02$, and $\overline{f_u} = 0.02$. Finally, the term $\mathcal{L}_\text{reg}$ is included for structured linear maps which include constraints that must be enforced via optimization—in this case, the Soft SVD parametrization.

$$\mathcal{L}_y = \frac{1}{N}\sum_{k=1}^{N} ||\hat{\mathbf{y}}_k - \mathbf{y}_k||_2^2 \tag{4a}$$

$$\mathcal{L}_\text{e} = ||\hat{\mathbf{x}}^\text{e} - \hat{\mathbf{x}}||_2^2 \tag{4b}$$

$$\mathcal{L}_{dx} = \frac{1}{N-1}\sum_{k=1}^{N-1} ||\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_{k+1}||_2^2 \tag{4c}$$

$$\mathcal{L}_{f_u}^\text{con} = \frac{1}{N}\sum_{k=1}^{N}\big(\max(0,\ -f_u(\mathbf{u}_k) + \underline{f_u}) + \max(0,\ f_u(\mathbf{u}_k) - \overline{f_u})\big) \tag{4d}$$

$$\mathcal{L}_y^\text{con} = \frac{1}{N}\sum_{k=1}^{N}\big(\max(0,\ -\hat{\mathbf{y}}_k + \underline{y}) + \max(0,\ \hat{\mathbf{y}}_k - \overline{y})\big) \tag{4e}$$

$$\mathcal{L} = Q_y\mathcal{L}_y + Q_\text{e}\mathcal{L}_\text{e} + Q_{dx}\mathcal{L}_{dx} + Q_y^\text{con}\mathcal{L}_y^\text{con} + Q_{f_u}^\text{con}\mathcal{L}_{f_u}^\text{con} + Q_\text{reg}\mathcal{L}_\text{reg} \tag{4f}$$

	SSM Type	f_*	f_* Map	f_* Act.	f_* Layers	f_* Nodes	f_* λ_{min}	f_* λ_{max}	$f_x \circ f_u$	N	$Q_{f_u}^{con}$	Q_y^{con}	Q_{reg}	Q_{dx}	Q_e
1	Block	Linear	Linear	ReLU	1	4	0.0	0.5	+	4	0.0	0.0	0.1	0.0	0.0
2	Black-box	MLP	Soft SVD	GELU	2	8	0.1	0.6	×	8	0.1	0.1	1.0	0.1	0.1
3		Res. MLP	PF	BLU	3	16	0.2	0.7	+ / ×	16	1.0	1.0	10.0	1.0	1.0
4		RNN	Lasso	SoftExp	4	32	0.3	0.8		32	10.0	10.0		10.0	10.0
5					5		0.4	0.9		64					
6							0.5	1.0							
7								1.1							
8								1.2							

Figure 1: The SSM genome.

As shown in Equation 4f, the combined multi-objective loss $\mathcal{L}$ contains the terms from above weighted by factors Q—these loss terms may be optimally weighted with varying importance for particular systems and architectures.

2.5. State Space Model Genome

Figure 1 illustrates the complete space of model configurations possible in our proposed neural dynamics design space. Columns indicate the attributes (genes) of constituent components while rows indicate the possible values each attribute can take. Columns with f_* represent attributes that model components f_x, f_u, f_y, and f_o share but whose values may vary across components. For the purposes of our experiments, we represent models sampled from the search space as a vector of traits, each with a discrete set of possible values. These sets are represented as ordered ring buffers which allows wrapping of position-wise mutations selecting the nearest higher or lower value for attributes with a natural ordering.

In addition to the full granularity of the design space with all potential combinations of attribute values available (dubbed ***XL***), we introduce a restricted design space (dubbed ***standard***) that captures typical heuristics commonly found in expert designed models. The standard design space couples the linear parametrizations and activation functions of all block components f_* in the SSM. It further imposes some additional hierarchy on the design of discovered architectures through the addition of SSM types Hammerstein, Hammerstein-Weiner, Linear, and Block Nonlinear. With these additional values the attribute SSM Type completely determines which block components are linear (see Table 1) for the standard design space. The standard genome additionally omits $f_x \circ f_u$ from the search, using addition to model state and input component interactions by default. Following a common pattern in neural network architecture design practice, values for number of layers, number of nodes, activation function, and linear map parametrization are the same across block components in the standard design space. While still a vast search space, these restrictions reduce the total number of possible architectures by orders of magnitude from $\sim$3 trillion for the XL design space to $\sim$3.5 billion for the standard design space.

2.6. Asynchronous Genetic Search

We introduce an asynchronous genetic algorithm designed to maximize utilization of a fixed computation budget while searching the design space in a directed manner. This is accomplished by

Block	Model class			
	Block Nonlinear	Hammerstein-Wiener	Hammerstein	Linear
f_x	N	Y	Y	Y
f_u	N	N	N	Y
f_y	Y	N	Y	Y

Table 1: Linear components for standard design space block-oriented model classes.

maintaining a fixed number of actively training individuals—initially generated at random—and at fixed-duration intervals (5 minutes for our experiments) spawning new individuals as models finish training. Models are ranked and selected according to best-observed n-step prediction mean squared error $\mathcal{L}_y$ (Equation 4a) on the validation set.

The algorithm is initialized with a random population of state space models from the design space. The number of new state space models dispatched during the periodic spawning phases equals the number of models which have terminated training since the last spawning, thereby effectively leveraging but not exceeding a fixed computational budget.

New population members are generated via one of three operators: random, mutation, or crossover. The random operator selects all traits at random. The mutation operator randomly selects a trait and randomly steps to the next higher or lower value according to the trait's prescribed or natural ordering. The crossover operator implements crossover weighted by fitness, taking two individuals A and B and for each trait randomly selecting to use A's trait with probability $\text{fitness}(A)/(\text{fitness}(A) + \text{fitness}(B))$.

The proportion of each generation's new births via mutation versus crossover is a hyperparameter: mutation probability p^{mut} is given, and crossover probability p^{cross} is derived simply as $1 - p^{\text{mut}}$. Further, in order to promote more exploration at the beginning of search, each new generation may spawn a randomly-selected model with probability p_i^{birth}, which is annealed by a constant factor $0 < k < 1$ prior to each iteration, i, so that $p_{i+1}^{\text{birth}} = k p_i^{\text{birth}}$.

We use random search as an alternative method for exploring the design space. Random search selects options for each position in the model vector representation at random without the fitness-based guidance of genetic operators.

3. Experiments

To measure the efficacy of our dynamics model design space, we perform model search using the two search algorithms and three datasets.

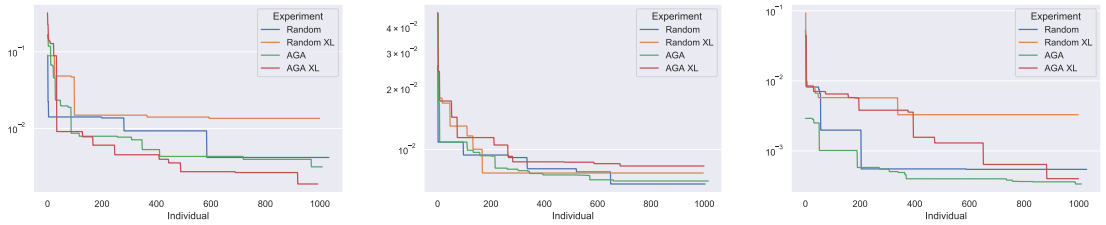
3.1. Model Training

Models are trained via full-batch gradient descent with the AdamW optimizer (Loshchilov and Hutter, 2019) for a fixed number of epochs and a learning rate of 2×10^{-3}. Early stopping is used to terminate training when a model has not improved on its best validation set performance for a set number of epochs. We train models for 1,000 epochs at most, and allow models 100 epochs of training without improvement before termination.

We initialize each search algorithm with 50 individuals, maintain a fixed-size pool of 50 active individuals, and check the active population and spawn any new individuals every 5 minutes. For

Dataset		AGA	Random	AGA XL	Random XL
Aero	Val.	3.16×10^{-3}	4.18×10^{-3}	$\mathbf{1.89 \times 10^{-3}}$	1.35×10^{-2}
	Test	1.23×10^{-2}	1.04×10^{-2}	$\mathbf{6.87 \times 10^{-4}}$	1.44×10^{-2}
CSTR	Val.	7.00×10^{-3}	$\mathbf{6.76 \times 10^{-3}}$	8.30×10^{-3}	7.66×10^{-3}
	Test	8.32×10^{-3}	$\mathbf{8.00 \times 10^{-3}}$	9.14×10^{-3}	1.12×10^{-2}
Two Tank	Val.	$\mathbf{3.37 \times 10^{-4}}$	5.45×10^{-4}	4.01×10^{-4}	3.29×10^{-3}
	Test	$\mathbf{1.00 \times 10^{-3}}$	2.21×10^{-3}	2.56×10^{-3}	1.11×10^{-2}

Table 2: Validation and test set open-loop MSE for each dataset and algorithm's best model.



(*a*) Aerodynamics (*b*) CSTR (*c*) Two Tank

Figure 2: Best open-loop validation set MSE as optimization progresses for each search algorithm, model genome, and dataset.

the AGA, we let initial random birth probability $p_0^{\text{birth}} = 1$ with annealing rate $k = 0.5$, and we let $p^{\text{mut}} = 0.2$ and $p^{\text{cross}} = 0.8$.

3.2. Datasets

We evaluate our design spaces using three non-autonomous systems with different properties (observed variables $\mathbf{y}_k$ and control inputs $\mathbf{u}_k$ dimensionality given in parentheses):

- **Aerodynamics** (MathWorks, a): models y and z acceleration and angular velocity in all dimensions of an aerodynamic body using ten control inputs ($\mathbf{y}_k \in \mathbb{R}^5$, $\mathbf{u}_k \in \mathbb{R}^{10}$).
- **CSTR** (Hedengren, a; MathWorks, b): models temperature and chemical concentration of a non-adiabatic continuous stirred tank reactor using a single control input ($\mathbf{y}_k \in \mathbb{R}^2$, $\mathbf{u}_k \in \mathbb{R}$).
- **Two Tank** (Hedengren, b): models water levels in two tanks governed by inputs for pump speed and valve opening ($\mathbf{y}_k \in \mathbb{R}^2$, $\mathbf{u}_k \in \mathbb{R}^2$).

4. Results and Analysis

Table 2 lists open-loop validation and test set MSE for models obtained by each search algorithm, and Table 3 gives the attribute values for each of these models. For the Aerodynamics and Two Tank datasets, AGA discovered models from the XL genome that outpaced those found by RS from the same genome, with a considerable performance gap emerging as optimization progressed. These results underscore the AGA's robustness with high-dimensional search spaces. A similar result can be seen for the Two Tank datasets with the standard model genome, though the performance gap between AGA and RS isn't as pronounced. Although the Aerodynamics model obtained from the

Dataset	Algorithm	SSM Type	State Est.	Linear Map	Nonlin. Map	Activation	Layers	Nodes	λ_{min}	λ_{max}	N	Q_{fu}^{con}	Q_{y}^{con}	Q_{reg}	Q_{dx}	Q_{e}
Aero	AGA	Blk. Nonlin.	Res. MLP	Soft SVD	RNN	BLU	2	32	0.0	1.2	64	0.0	0.1	0.1	0.0	1.0
CSTR	RS	HW	RNN	Linear	RNN	GELU	3	32	—	—	64	10.0	0.0	1.0	10.0	0.0
Two Tank	AGA	Hammerstein	Linear	Linear	RNN	GELU	2	32	—	—	8	1.0	0.0	0.1	1.0	10.0

Dataset	Algorithm	Model	f_*	Nonlin. Map	Linear Map	Activation	Layers	Nodes	λ_{min}	λ_{max}	$f_x \circ f_u$	N	Q_{fu}^{con}	Q_{y}^{con}	Q_{reg}	Q_{dx}	Q_{e}
Aero	AGA	Block	f_x	rMLP	Lasso	ReLU	1	32	—	—	$+/\times$	4	0.0	0.0	10.0	1.0	0.1
			f_u	Linear	Soft SVD	SoftExp	4	4	0.0	0.8							
			f_y	rMLP	Lasso	ReLU	1	32	—	—							
			f_o	RNN	Lasso	BLU	2	8	—	—							
CSTR	RS	Black-box	f	rMLP	Soft SVD	ReLU	5	32	0.5	0.9	N/A	16	—	0.1	10.0	10.0	0.1
			f_y	MLP	Soft SVD	BLU	2	4	0.3	1.2							
			f_o	Linear	PF	GELU	5	4	0.0	1.2							
Two Tank	AGA	Black-box	f	rMLP	Linear	GELU	1	16	—	—	N/A	8	—	10.0	—	0.1	0.1
			f_y	MLP	Linear	BLU	1	16	—	—							
			f_o	Linear	Linear	BLU	5	32	—	—							

Table 3: Attributes of best-observed models according to validation set MSE, obtained from standard (top) and XL (bottom) SSM genomes.

standard genome outperformed the model obtained by RS, it didn't quite generalize to the test set as effectively as the latter, a possible indication of the AGA's potential to overfit the validation set. For CSTR, RS prevailed over AGA for the standard genome in both validation and test set MSE; however, AGA search over the XL genome obtained a model with better generalization to the test set than the model found via RS despite lower validation set performance. By all accounts, AGA search was able to find models that yielded performance competitive or superior to those found by RS.

The XL genome results noticeably demonstrate the efficacy of our AGA implementation when performing directed searches in high-dimensional optimization landscapes. For the Aerodynamics and Two Tank datasets, AGA is able to pull ahead of RS and converge to more performant models. These results further suggest that decoupling the structure of model components in neural SSMs can improve performance on certain systems—the Aerodynamics model in particular benefits from this decoupling, though other systems seem to perform well without this decoupling.

To assess the effectiveness of structured architectures and linear maps for neural SSMs, Figure 3 shows the distribution of deviation from the average log-scaled test set loss for models obtained by RS over the standard model genome, grouped by SSM type and linear map choices. For brevity, we focus our analysis on these attributes and their behavior in the standard genome.

One might expect that each system would converge toward a structure that lends itself to the true physical properties of the system; however, we find that a broad set of structural configurations can produce good models overall. Looking at SSM types, it appears that block-structured SSMs generally perform better than linear or black-box SSMs for all systems, though linear and black-box can be competitive at their best. Models with purely linear components can perform well, but exhibit high variance in performance—a reasonable result since the systems being modeled are nonlinear. As for linear maps, adding structure can be effective toward producing consistent models with lower variance in performance. However, this structure is not always effective for all systems—often linear maps without structural priors yield models that perform well. For certain systems, however, models with structured maps on average perform better than the average over all linear maps and may exceed the performance of unstructured maps with further tuning. It remains to be seen whether more careful tuning of constraints and loss coefficients will give better performance to the structured maps that rely on these attributes.

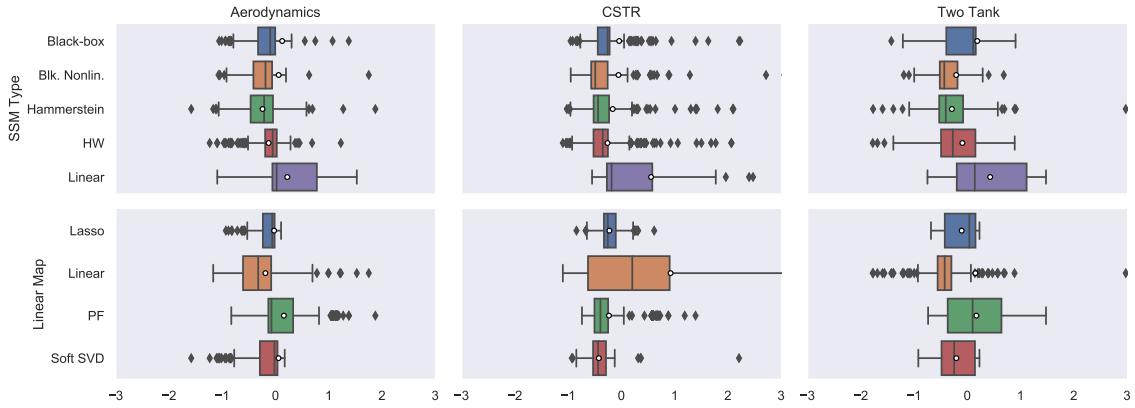
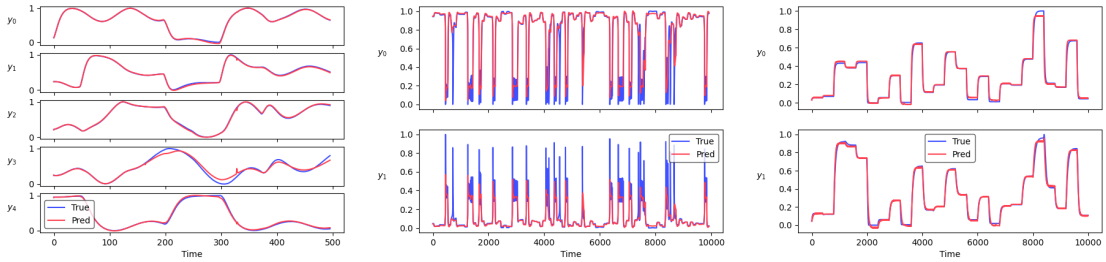


Figure 3: Boxplots indicating distributions of deviation from average log-scaled test set loss for SSM type and linear map choices used in all random search runs over the standard model genome for each dataset. White circles indicate mean of deviations.



(*a*) Aerodynamics (AGA XL) (*b*) CSTR (Random) (*c*) Two Tank (AGA)

Figure 4: Open-loop traces for each system's best-observed model, with search algorithm and space indicated in parentheses.

5. Conclusion

We present design space variations for neural SSMs, which include a rich selection of structured components and optimization constraints for the automated discovery of performant dynamical system identification models. We evaluate our two design spaces with two search algorithms—random and asynchronous genetic search—and provide an analysis of the architectures discovered for models trained and evaluated on three non-autonomous systems. We find that adding block structure and nonlinearity to neural SSMs can yield a greater likelihood of obtaining good models. Blackbox and/or linear variations can also perform well, though with larger variance across runs.

For future work, we would like to extend our analysis to how AGA parameters affect optimization progress and results, as well as a larger selection of structural configurations. We also plan to continue exploring the properties of our genetic algorithm. Thus far, our analysis has revealed that the asynchronicity of our genetic algorithm results in a complex interaction between learning and evolution. We hope to examine these properties in greater detail and find ways to exploit them to

improve the AGA's efficacy. Additionally, the AGA currently uses a naive selection approach, which may limit the diversity of selected models. We believe this could be improved using techniques such as non-dominated sorting with multiple objectives or tournament selection. Finally, we plan to extend the AGA to evolve network topology at a lower level.

6. Acknowledgments

This research was supported by the Laboratory Directed Research and Development (LDRD) at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL0-1830.

References

A. J. Al-Mahasneh, S. G. Anavatti, and M. Garratt. Nonlinear multi-input multi-output system identification using neuro-evolutionary methods for a quadcopter. In *2017 Ninth International Conference on Advanced Computational Intelligence (ICACI)*, pages 217–222, 2017.

Helon Vicente Hultmann Ayala, Didace Habineza, Micky Rakotondrabe, and Leandro dos Santos Coelho. Nonlinear black-box system identification through coevolutionary algorithms and radial basis function artificial neural networks. *Applied Soft Computing*, 87:105990, 2020. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2019.105990. URL http://www.sciencedirect.com/science/article/pii/S1568494619307719.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. AntisymmetricRNN: A dynamical system view on recurrent neural networks. *arXiv preprint arXiv:1902.09689*, 2019.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research 20 (2019) 1-21*, 2019.

L. Ferariu and B. Burlacu. Multiobjective graph genetic programming with encapsulation applied to neural system identification. In *15th International Conference on System Theory, Control and Computing*, pages 1–6, 2011.

Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary intelligence*, 1(1):47–62, 2008.

Adam Gaier and David Ha. Weight agnostic neural networks. In *Advances in Neural Information Processing Systems*, pages 5364–5378, 2019.

Luke B Godfrey. An evaluation of parametric activation functions for deep learning. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3006–3011. IEEE, 2019.

Luke B Godfrey and Michael S Gashler. A continuum among logarithmic, linear, and exponential functions, and its potential to improve generalization in neural networks. In *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, volume 1, pages 481–486. IEEE, 2015.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34 (1):014004, 2017.

Toshiharu Hatanaka, Nobuhiko Kondo, and Katsuji Uosaki. Multi-objective structure selection for RBF networks and its application to nonlinear system identification. In *Multi-Objective Machine Learning*, pages 491–505. Springer, 2006.

John D. Hedengren. Nonlinear Model Predictive Control. http://apmonitor.com/do/index.php/Main/NonlinearControl, a. Accessed: 2020-11-05.

John D. Hedengren. Level Regulation with MPC. https://apmonitor.com/do/index.php/Main/LevelControl, b. Accessed: 2020-11-05.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units. *arXiv preprint arXiv:1606.08415*, 2016.

Kui Jia, Shuai Li, Yuxin Wen, Tongliang Liu, and Dacheng Tao. Orthogonal deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

Giancarlo Kerg, Kyle Goyette, Maximilian Puelma Touzel, Gauthier Gidel, Eugene Vorontsov, Yoshua Bengio, and Guillaume Lajoie. Non-normal recurrent neural network (nnRNN): learning long time dependencies while improving expressivity with transient dynamics. In *Advances in Neural Information Processing Systems*, pages 13613–13623, 2019.

Jason Liang, Elliot Meyerson, and Risto Miikkulainen. Evolutionary architecture search for deep multitask networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 466–473, 2018.

Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, and Gary Yen. A survey on evolutionary neural architecture search. *arXiv preprint arXiv:2008.10937*, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019.

MathWorks. Modeling an Aerodynamic Body. https://www.mathworks.com/help/ident/ug/modeling-an-aerodynamic-body.html, a. Accessed: 2020-10-06.

MathWorks. Non-Adiabatic Continuous Stirred Tank Reactor: MATLAB File Modeling with Simulations in Simulink. https://www.mathworks.com/help/ident/ug/non-adiabatic-continuous-stirred-tank-reactor-matlab-file-modeling-with-simulations-in-simulink.html, b. Accessed: 2020-10-06.

Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *International Conference on Machine Learning*, pages 2401–2409, 2017.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

David R So, Chen Liang, and Quoc V Le. The evolved transformer. *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Bidyadhar Subudhi and Debashisha Jena. A differential evolution based neural network approach to nonlinear system identification. *Applied Soft Computing*, 11(1):861–871, 2011a.

Bidyadhar Subudhi and Debashisha Jena. Nonlinear system identification using memetic differential evolution trained neural networks. *Neurocomputing*, 74(10):1696–1709, 2011b.

Aaron Tuor, Jan Drgona, and Draguna Vrabie. Constrained neural ordinary differential equations with stability guarantees. *arXiv preprint arXiv:2004.10883*, 2020.

Jiayun Wang, Yubei Chen, Rudrasis Chakraborty, and Stella X Yu. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11505–11515, 2020.

C. Yang, J. Qiao, and L. Wang. A novel echo state network design method based on differential evolution algorithm. In *2017 36th Chinese Control Conference (CCC)*, pages 3977–3982, 2017.

Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839. IEEE, 2019.

Jiong Zhang, Qi Lei, and Inderjit S Dhillon. Stabilizing gradients for deep neural networks via efficient SVD parameterization. *Proceedings of the 35th International Conference on Machine Learning*, 2018.