Predicting Timescales in Dynamical Systems With Explainable Machine Learning Models

Demetri Liousas[*1], **Andrew D. Simin**[1], **Aditya Kashi**[2], **Wesley H. Brewer**[2], **Stephen M. de Bruyn Kops**[1], **Muralikrishnan Gopalakrishnan Meena**[*2 †]

[1]Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst, Massachusetts, USA
[2]National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[†]To whom correspondence should be addressed: gopalakrishm@ornl.gov

Abstract

We aim to extract the timescales governing the time evolution of a dynamical system by creating physical explanations for time-series machine learning (ML) models. We employ the damped pendulum, a simple dynamical system with analytical solutions for the underlying timescales, to demonstrate this framework for explainable ML models. Our analysis reveals that by discovering the minimum information required to train a time-series ML model, we can identify the physical timescales of the underlying dynamical system.

Introduction

Understanding and modeling physical time-evolving dynamical systems is a formidable challenge encountered across various science and engineering domains. Complex examples in nature include turbulence in the ocean and atmosphere (Wyngaard 1992). Nonlinear dynamics make these systems complex and challenging to represent mathematically. Typically, these models are described by coupled nonlinear partial differential equations that can only be solved numerically with extreme computational costs (Norman et al. 2021).

Machine learning (ML) advancements have created a new class of models that can explain the behavior of time-evolving physical systems by implicitly learning their underlying dynamics (Connor, Martin, and Atlas 1994; Yu et al. 2019; Che et al. 2018; Hewamalage, Bergmeir, and Bandara 2021). In this study, we evaluate the ability of ML models to learn time series generated from a system of coupled ordinary differential equations (ODEs) and extract the timescales that define the behavior of the dynamical system. We use the damped pendulum (Nelson and Olsson 1986) as a sample problem to demonstrate the framework because it is a simple and well-studied dynamical system. It has two constant timescales, which can be derived analytically, allowing for theoretical analysis of the ML models' performance. We study this sample problem with two time-series ML models: long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) and neural ordinary differential equations (NODE) (Chen et al. 2018). We evaluate the ability of these models to extract the physical timescales by relating the physical timescales to ML hyperparameters analogous to the timescales of the time-series data.

[*]These authors contributed equally.

XAI4Sci: Explainable machine learning for sciences, AAAI-24 (xai4sci.github.io)

Time-Series Prediction with Neural Networks

We aim to create a framework for modeling a system of nonlinear ODEs, even if it has no closed-form analytical solutions. We represent the ODEs in discrete time as

$$\frac{\mathrm{d}\boldsymbol{q}(t_i)}{\mathrm{d}t} = \boldsymbol{f}\left(\boldsymbol{q}(t_i), t_i\right) \tag{1}$$

where $\boldsymbol{q}$ represents the state of the system at time t_i, and $\boldsymbol{f}$ is the vector function defining the right-hand side (RHS) of the system of ODEs; the RHS here is modeled generically as the functional form is not always known.

We would like to model the RHS of a system of ODEs using a neural network so that a generic numerical ODE solver can solve for the time evolution of the system's states, $\boldsymbol{q}(t)$. We use neural networks because they have the capacity to approximate any Borel measurable function if it has the proper architecture (Hornik, Stinchcombe, and White 1989). We also require a neural network capable of modeling the time history, as the time evolution of most dynamic systems depends on their previous states. We use neural networks with sequence-based modeling to satisfy this requirement, where a sequence refers to the time history used in the time series model or, more simply, its memory.

Generally, sequences of time-series data have two parameters analogous to timescales in physical time-evolving systems, as illustrated in Fig. 1. One timescale is the sequence length, which is the entire time span of the data in one sequence; a sequence is a subsample of a time series, not the whole time series. The second timescale is the period between samples within a single sequence, often referred to by its inverse, the sampling frequency. With respect to physical timescales, the sequence length of the time series is the longest timescale that can be modeled, and the sampling period is the shortest timescale. The timescales

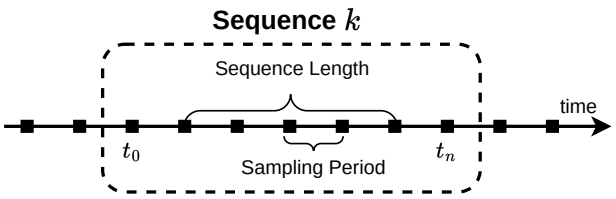


Figure 1: Illustration of the timescales defined by a sequence in a time-series data.

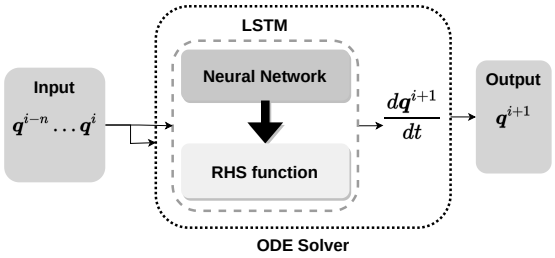


Figure 2: Sample portrayal of the LSTM modeling framework used in the current study for predicting the time evolution of a system of ODEs.

of the time-series data are hyperparameters for sequence-based ML models. Our aim is to reveal a relationship between these ML and physical timescales, thus creating a physics-based explanation for the ML hyperparameters. We analyze two neural network architectures for sequence modeling: Long Short-Term Memory (LSTM) and Neural Ordinary Differential Equations (NODE).

Long Short-Term Memory (LSTM)

The recurrent neural network (RNN) architecture allows for processing sequences of inputs, which theoretically allows it to model time-series data (Connor, Martin, and Atlas 1994; Che et al. 2018; Hewamalage, Bergmeir, and Bandara 2021). The LSTM architecture improves the generic RNN by increasing stability and predictive performance, making it more attractive to use (Hochreiter and Schmidhuber 1997). We use an LSTM model to represent the RHS of Eq. 1. Our modeling framework is depicted in Fig. 2. The input of the LSTM model is a full sequence, or time history, of time-series data, $\boldsymbol{q}(t_{i-n})$, $\boldsymbol{q}(t_{i-(n-1)})$, $\dots$, $\boldsymbol{q}(t_i)$ where n is the sequence length. The output of the LSTM model is the RHS of the system of ODEs, $\boldsymbol{f}\left(\boldsymbol{q}(t_i), t_i\right)$. This is fed into an ODE solver (such as RK4) to obtain the next state $\boldsymbol{q}_{i+1}$.

In practice, we find that the LSTM modeling framework lacks the desired stability and accuracy for modeling the time evolution of $\boldsymbol{q}$, which will be demonstrated later in this paper.

Neural Ordinary Differential Equations (NODE)

NODE is a neural network architecture devised by Chen et al. for modeling time-series data from systems of ODEs. Rather than just modeling the RHS of a system of ODEs, NODE integrates the ODE solver into the ML model (Chen et al. 2018). The hidden state of NODE itself is an ODE

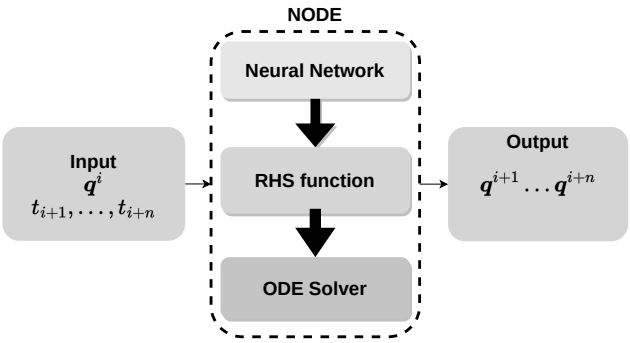


Figure 3: Sample portrayal of the NODE modeling framework used in the current study for predicting the time evolution of a system of ODEs.

$$\frac{\mathrm{d}\boldsymbol{q}(t)}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{q}(t), t, \theta), \tag{2}$$

where θ is the weight tensor which the optimization process varies. The optimization is performed on a black-box ODE solver for Eq. 2. The loss $\mathcal{L}\left(\boldsymbol{q}(t_1)\right)$ is given by

$$\mathcal{L}\left(\boldsymbol{q}(t_0) + \int_{t_0}^{t_1} \boldsymbol{f}(\boldsymbol{q}(t), t, \theta)dt\right) \tag{3}$$

which is discretized to use a numerical ODE solver

$$\mathcal{L}\left(ODESolve(\boldsymbol{q}(t_0), \boldsymbol{f}, t_0, t_1, \theta)\right). \tag{4}$$

The input of NODE is the current state of the system $\boldsymbol{q}(t_i)$ and the desired time sequence t_i, t_{i+1}, $\dots$, t_{i+n} at which to evaluate the state, where n is the sequence length. The output is the state at those time instances $\boldsymbol{q}(t_i)$, $\boldsymbol{q}(t_{i+1})$, $\dots$, $\boldsymbol{q}(t_{i+n})$. This framework is depicted in Fig. 3.

The differences between the time dependencies of LSTM and NODE are summarized in Fig. 4. While LSTM depends on the time history of the data, the NODE model only needs the current state to predict the time evolution of a system. Nonetheless, the two timescales – sequence length and sampling frequency – are hyperparameters to both models. We investigate these hyperparameters to find the minimum information needed to model a system of ODEs accurately.

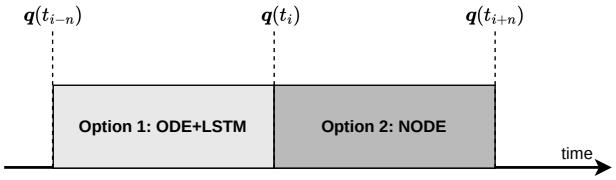


Figure 4: Illustration of the difference in time-dependence between the LSTM and NODE modeling frameworks.

Damped Pendulum Sample Problem

A canonical problem to test the capability of our framework is the simple damped pendulum (Nelson and Olsson 1986).

This nonlinear system can be written as a system of coupled nonlinear ODEs and has analytical solutions using the small-angle approximation.

Dynamics Model

The governing equations of the pendulum are typically written in terms of its angular position θ and angular velocity ω. However, many dynamical systems in engineering and nature are best described in terms of energy. We write the system of ODEs governing the time evolution of the pendulum in terms of its kinetic energy K, potential energy P, and the flux F between them. It is given by

$$\frac{\mathrm{d}K}{\mathrm{d}t} = -\frac{2b}{m}K - F \tag{5}$$

$$\frac{\mathrm{d}P}{\mathrm{d}t} = F \tag{6}$$

$$\frac{\mathrm{d}F}{\mathrm{d}t} = 2\frac{g}{l}(K - P) + \frac{P}{ml^2}(2K - P) - \frac{b}{m}F, \tag{7}$$

where b is the damping coefficient, m is the mass of the pendulum bob, g is the acceleration due to gravity, and l is the length of the pendulum. Thus, the system's state is defined as $\boldsymbol{q} = [K \quad P \quad F]^T$. The system of ODEs is solved with the initial conditions K_0, P_0, and F_0.

Timescale Analysis

The pendulum has two timescales that govern its time evolution. The shorter timescale is the oscillating timescale τ_o and the longer one is the damping timescale τ_d. They can be derived analytically using the small-angle approximation (Nelson and Olsson 1986) and are given by

$$\tau_o = \sqrt{\frac{\ell}{g}} \tag{8}$$

$$\tau_d = \frac{m}{b}. \tag{9}$$

We are only interested in cases where the oscillating timescale is much shorter because this gives periodic motion. This happens when the pendulum is underdamped such that

$$\tau_o \ll \tau_d. \tag{10}$$

We can use these timescales to determine the theoretical minimum information that a data-driven model would need to learn the dynamics of a pendulum effectively. An accurate model must capture the information corresponding to the damping and oscillating timescales.

Results

We perform a parametric study in the ML timescale hyperparameter space comprising the sequence length and sampling frequency. We study sequence lengths that span time periods of 0.02 to 5.5, and sampling frequencies from 8 to 256.

We select a damped pendulum with damping coefficient $b = 0.5$, mass $m = 1$, gravity $g = 25$, length $\ell = l$, and initial conditions $K_0 = 0$, $P_0 = 1$ and $F_0 = 0$. Note that for these parameters, $\tau_d >> \tau_o$. Thus, the pendulum is underdamped. The raw data was generated numerically using these settings of the pendulum. The sequenced training and testing data for the ML models was derived from the raw data using the various sequencing parameters of the study. Training and testing of the models were performed over different time intervals.

LSTM Results

We used an LSTM model with two layers with a hidden size of five. We also use a multi-layer perceptron (MLP) with one hidden layer of ten neurons before the output layer. A *Leaky* Rectified Linear Unit (LeakyReLU) activation function with a slope of 0.1 is applied for both the LSTM and MLP layers. We measure the mean squared error as the loss function for optimizing the network weights during the training process. We select a learning rate of 0.001 and train the model for 4000 epochs, by which the model training converges. Towards the end of this training process, we use ensemble learning to improve the stability of the LSTM model. These hyperparameters are used for every LSTM model in the parametric study.

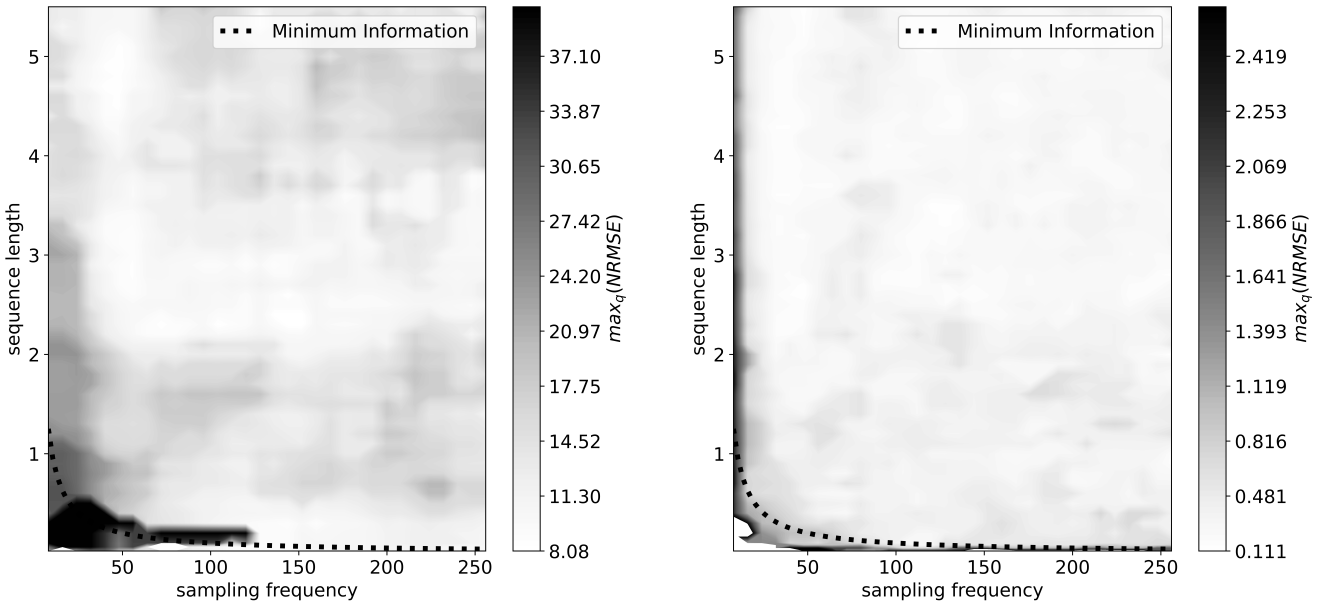
The parametric study results are shown in Fig. 5a. Small sampling frequency and sequence length values have very high errors because too little information is provided to the model. This lives in the region below the minimum information line. The minimum information line corresponds to when the sequence length to sampling period ratio equals the analytically computed damping to oscillating timescale ratio. The models behave much more accurately above this line despite the local regions of high model error. This distinction shows that the LSTM framework can explain the damped pendulum system dynamics by extracting its physical timescales.

NODE Results

Through experimentation, we selected the other hyperparameters for the NODE model. We chose an absolute tolerance of 10^{-3} and a relative tolerance of 10^{-4} for the integrated ODE solver. Furthermore, an MLP with three layers and 20 neurons per layer was used to represent the RHS of the ODE. We use a Sigmoid Linear Unit (SiLU) activation function for these layers. We apply a learning rate of 0.05 and 1000 epochs for training due to the faster convergence of NODE compared to LSTM.

The results of the parametric study for NODE are shown in Fig. 5b. Similar to LSTM, the NODE model error is very high below the minimum information line, and the performance is much better above it. Again, these results demonstrate that analyzing the inherent timescales of the input data to the time-series ML model, NODE for the current discussion, can enable one to extract the timescales of the physical system. Furthermore, we identify that these timescales correspond to the minimum information required by the time-series ML models to predict the system dynamics accurately.

We observe that the NODE model performs notably better than LSTM in this parametric study. The maximum and minimum errors observed in the NODE model are an order



(a) LSTM Model

(b) NODE Model

Figure 5: Shaded contour plot of the parametric study with both ML models. The lighter-colored regions correspond to lower model error. The error metric used is the maximum normalized rms error (NRMSE) among the state variables. The minimum information line is denoted by the black dashed lines.

of magnitude smaller than those of the LSTM model. Furthermore, NODE is much more numerically stable above the minimum information line demonstrated by the much smoother plateau than LSTM, which has many local minima and maxima.

Concluding Remarks

We develop physical explanations for the information required by time-series machine learning (ML) models by analyzing the timescales of the input data. We broadly identify the timescales of the input sequence data for time-series ML models as the sequence length and sampling frequency of the data. We perform a parametric analysis of the effect of these ML timescales on the performance of two time-series ML models, namely LSTM and NODE, to effectively model the time evolution of dynamical systems. By employing the analysis on a simple dynamical system, a damped pendulum, we showed that the minimum information (ML timescales) required by the time-series ML models to effectively model the time evolution of the pendulum corresponds to the physical timescale of the pendulum.

Specifically for the damped pendulum, we observe that the minimum information needed for an ML model is when the ratio of the ML timescales is equal to that of the oscillating and damping timescales of the pendulum. Intuitively, this result means that the sampling frequency must be large enough to pick up on the oscillations of the pendulum, and the ML model needs to see a long enough sequence of data to model the damping of the pendulum. The current framework of timescale extraction from time-series ML models is a valuable tool for creating physical explanations of such models' behavior. Moreover, such data-driven techniques can be extended to physical problems where experimental or numerical data is available, but limited information of the physical timescales is known.

Acknowledgments

This research was supported by the Office of Naval Research via grant N00014-19-1-2152. High performance computing resources were provided via the U.S. Department of Energy (DOE) INCITE program by the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. Computing resources were also provided through the U.S. Department of Defense High Performance Computing Modernization Program. This project was supported by an Honors Research Grant from the University of Massachusetts Amherst Commonwealth Honors College.

References

Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; and Liu, Y. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1): 6085.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Connor, J. T.; Martin, R. D.; and Atlas, L. E. 1994. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2): 240–254.

Hewamalage, H.; Bergmeir, C.; and Bandara, K. 2021. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1): 388–427.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-term Memory. *Neural computation*, 9: 1735–80.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks*, 2(5): 359–366.

Nelson, R. A.; and Olsson, M. G. 1986. The Pendulum—Rich Physics from a Simple System. *American Journal of Physics*, 54(2): 112–121.

Norman, M. R.; Bader, D. A.; Eldred, C.; Hannah, W. M.; Hillman, B. R.; Jones, C. R.; Lee, J. M.; Leung, L. R.; Lyngaas, I.; Pressel, K. G.; Sreepathi, S.; Taylor, M. A.; and Yuan, X. 2021. Unprecedented cloud resolution in a GPU-enabled full-physics atmospheric climate simulation on OLCF's summit supercomputer. *International Journal of High Performance Computing Applications*, 36(1): 93–105.

Wyngaard, J. C. 1992. Atmospheric turbulence. *Annual Review of Fluid Mechanics*, 24(1): 205–234.

Yu, Y.; Si, X.; Hu, C.; and Zhang, J. 2019. A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation*, 31(7): 1235–1270.