Common Task Framework For a Critical Evaluation of Scientific Machine Learning Algorithms

Philippe M. Wyder [1] Judah Goldfeder [2] Alexey Yermakov [1 3] Yue Zhao [4] Stefano Riva [5] Jan Williams [6] David Zoro [3] Amy Sara Rude [1] Matteo Tomasetto [7] Joe Germany [8] Joseph Bakarji [9] Georg Maierhofer [10] Miles Cranmer [10] J. Nathan Kutz [1 3]

Abstract

To address the problem of rapid development outpacing the creation of standardized, objective benchmarks, we propose a Common Task Framework (CTF) for evaluating scientific machine learning models on dynamical systems. The CTF features a curated set of datasets and task-specific metrics spanning state forecasting, state reconstruction, and generalization under realistic constraints, including noise and limited data. Inspired by the success of CTFs in other machine learning fields like natural language processing and computer vision, our framework provides a structured, rigorous foundation for head-to-head evaluation of diverse algorithms. Our open-source framework enables researchers to rapidly implement, test, and optimize their models against our datasets to support our long-term vision to raise the bar for rigor and reproducibility in scientific ML.

[1]Department of Applied Mathematics, University of Washington, Seattle, WA 98195 [2]Department of Computer Science, Columbia University, New York, NY 10027 [3]Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195 [4]High Performance Machine Learning at SURF, Amsterdam, the Netherlands [5]Department of Energy, Nuclear Engineering Division, Politecnico di Milano, Milan, Italy [6]Department of Mechanical Engineering, University of Washington, Seattle, WA 98195 [7]Department of Mechanical Engineering, Politecnico di Milano, Milan, Italy [8]Department of Mathematics, American University in Beirut, Beirut, Lebanon [9]Department of Mechanical Engineering, American University in Beirut, Beirut, Lebanon [10]Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK. Correspondence to: J. Nathan Kutz <kutz@uw.edu>.

Proceedings of the ICML 2025 Workshop on Championing Open-source Development in Machine Learning (CODEML '25). Copyright 2025 by the author(s).

1. Introduction

Data science, especially machine learning (ML) and artificial intelligence (AI), is transforming almost every aspect of the engineering, physical, social, and biological sciences. As the body of literature on new ways to model many scientific data and systems grows, there is a lack of objective measures to adequately characterize and compare these methods. In the absence of a common standard for benchmarking new and existing approaches, the current literature is suffering from weak baselines, reporting bias, and inconsistent evaluations (McGreivy & Hakim, 2024). In this paper, we outline our contributions to objectively measure the performance of scientific ML models through our `ctf4science` package.

Our Contributions.

- The first CTF in scientific ML. We provide the scientific community with a diverse set of datasets and benchmarks for evaluating scientific ML models. Our CTF reveals method strengths and weaknesses by evaluating models on specific classes of problems and diverse objectives through multi-metric scoring.
- An open-source GitHub repository containing unit-tested code to give researchers a head start in benchmarking their own models to a growing list of already-benchmarked scientific ML models.

2. Background and Related Work

CTFs play a critical role in evaluating methodological advancements. Donoho (2017) has argued that the successful application of CTFs is a primary factor for the success of data science and machine learning. Indeed, the fields of speech recognition, natural language processing, and computer vision have developed mature CTF platforms that are progressively updated with more challenging data in order to drive progress and innovation. For instance, the industry-leading CVPR conference offers more than 30 challenge problems per year for participants to score and benchmark their ML/AI algorithms against. More broadly, classic fields of machine learning have benefited from extensive benchmark environments and common task frameworks, including ImageNet (Deng et al., 2009; Krizhevsky et al., 2012), Go and chess (Silver et al., 2018), video games such as Atari (Mnih et al., 2015) and StarCraft (Vinyals et al., 2019), and the OpenAI Gym (Ravichandiran, 2018; Dutta, 2018), among other environments for more realistic control (Deisenroth & Rasmussen, 2011; Todorov et al., 2012). Unlike these leading fields, many scientific disciplines have yet to integrate the CTF into their core infrastructure (McGreivy & Hakim, 2024). This compromises true comparative metrics between methods, algorithms, and results, and it limits the potential of ML in these areas.

3. Common Task Framework for Science and Engineering

We propose a CTF for science and engineering that is primarily focused on evaluating machine learning and AI models for dynamical systems: systems whose underlying evolution is determined by physical or biophysical principles or governing equations. The CTF will evaluate how well scientific ML models perform inference – such as state forecasting and state reconstruction – under defined constraints. We call these common, yet challenging scenarios “tasks”, which may involve noisy measurements, limited data, or varying system parameters. Given a training dataset and a range of timesteps to predict, users will produce approximations for a hidden test dataset. The predictions are evaluated and scored on a diverse set of metrics by an independent referee and posted on a leaderboard.

This scoring system prevents a winner-takes-all framework, since different modeling approaches will excel on different tasks. Some will do well with noise, others will not. Others might excel in the limited data regime, while performing poorly under parametric generalization. These profiles are important to provide a comprehensive and well-rounded performance metric and help guide scientists in selecting a suitable method.

Once the `ctf4science` is launched[1], we invite researchers to benchmark their methods on the CTF for Science by taking the following steps:

1. Sign-up and sign-in on Kaggle
2. Train your model with our training data and generate predictions for each benchmark case
3. Submit prediction files to the competition platform
4. See your score on the leaderboard

[1]We are proposing a launch date of November 1, 2025 on Kaggle

To interact with `ctf4science` before the competition launch, visit our GitHub repository[2], install the `ctf4science` package, and evaluate your method on our demo datasets *ODE_Lorenz* and *PDE_KS*. Our datasets and our ctf4science Python package do not require high-performance hardware and can be run on a laptop computer.

4. Datasets & Evaluation Metrics

We launch the CTF platform with two canonical problems in scientific machine learning: the Kuramoto-Sivashinsky (KS) equation, and the Lorenz equations. Both exhibit complex and challenging behavior for the science and engineering tasks of reconstruction and forecasting under the constraints of noise, limited data, and parametric dependence. While these datasets and benchmarks serve as a starting point, the CTF will evolve to include both more complex data and more challenging tasks. The CTF framework is a sustainable platform that evolves and grows as the community develops more sophisticated methods and algorithms and faces new challenges.

4.1. Spatio-Temporal System: Kuramoto-Sivashinsky

The first dataset we consider is the spatio-temporal Kuramoto-Sivashinsky equation. The KS equation is a fourth-order, nonlinear partial differential equation (PDE). It is considered a canonical example of spatio-temporal chaos in a one-dimensional PDE and is therefore commonly used as a test problem for data-driven algorithms. The KS equation is a particularly challenging case for fitting algorithms due to its combination of high dimensionality, nonlinearity, and sensitivity to initial conditions (chaotic behavior):

$$u_t + uu_x + u_{xx} + \mu u_{xxxx} = 0. \tag{1}$$

The solutions of Eq. (1) are defined on a grid across the domain of $[0, 32\pi]$ with periodic boundary conditions. A numerical integrator with a time step Δt evolves the solution m steps. An illustration of KS is provided in Fig. 1(b).

4.2. Dynamical System: Lorenz

The second dataset comes from one of the most influential dynamical systems in history, the Lorenz dynamical system, which is given by the ordinary differential equation (ODE):

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = rx - xz - y, \quad \frac{dz}{dt} = xy - bz.$$

where the parameters $b = 8/3$ and $\sigma = 10$ are typically fixed at these values while r is explored as a bifurcation parameter. For specific values of r, including our choice

[2]The package is available at:
`github.com/CTF-for-Science/ctf4science`

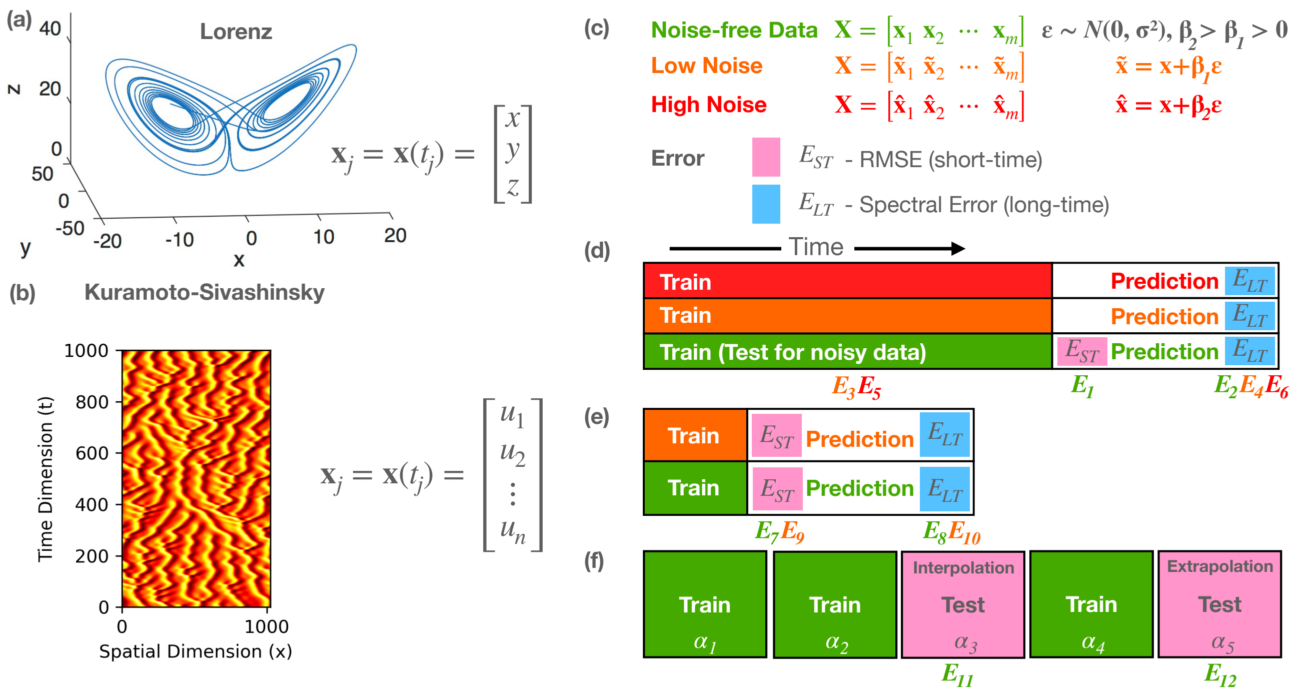
(a)
Lorenz
$\mathbf{x}_j = \mathbf{x}(t_j) = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$
(b)
Kuramoto-Sivashinsky
Time Dimension (t)
Spatial Dimension (x)
$\mathbf{x}_j = \mathbf{x}(t_j) = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$
(c)
Noise-free Data $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_m]$ $\varepsilon \sim N(0, \sigma^2)$, $\beta_2 > \beta_1 > 0$
Low Noise $\mathbf{X} = [\tilde{\mathbf{x}}_1\ \tilde{\mathbf{x}}_2\ \cdots\ \tilde{\mathbf{x}}_m]$ $\tilde{\mathbf{x}} = \mathbf{x} + \beta_1\varepsilon$
High Noise $\mathbf{X} = [\hat{\mathbf{x}}_1\ \hat{\mathbf{x}}_2\ \cdots\ \hat{\mathbf{x}}_m]$ $\hat{\mathbf{x}} = \mathbf{x} + \beta_2\varepsilon$
Error
E_{ST} - RMSE (short-time)
E_{LT} - Spectral Error (long-time)
(d)
Time
Train
Prediction E_{LT}
Train
Prediction E_{LT}
Train (Test for noisy data)
E_{ST} Prediction E_{LT}
$E_3 E_5$
E_1
$E_2 E_4 E_6$
(e)
Train
E_{ST} Prediction E_{LT}
Train
E_{ST} Prediction E_{LT}
$E_7 E_9$
$E_8 E_{10}$
(f)
Train α_1
Train α_2
Interpolation Test α_3
Train α_4
Extrapolation Test α_5
E_{11}
E_{12}

Figure 1. The CTF Evaluation framework scores the performance of methods on (a) the Lorenz dynamical system and (b) the Kuramoto-Sivashinsky partial differential equation. (c) Data is collected and organized into matrices which is then split into testing and training sets. RMSE is computed for reconstruction and short-time forecasting, while the spectral error computes the statistics of long-time forecasting (spatial or temporal). (d) Forecasting and reconstruction tasks are evaluated on noise-free, low-noise and high-noise data. Methods are also evaluated when (e) only limited data is available and (f) for reconstruction of parametrically dependent data.

$r = 28$, the system exhibits chaotic behavior as shown in Fig. 1(a).

4.3. Evaluation Metrics

For each dataset, we quantitatively assess the performance of each model on an ensemble of 12 classes of problems.

Scores E_1 and E_2 evaluate how effective a model is at generating future states of a dynamical system – a process that we call state "forecasting". E_1 evaluates short-time forecasting, measuring trajectory accuracy since deterministic predictions are feasible, while E_2 evaluates long-time forecasting, measuring statistical fidelity because only broad statistical properties are recoverable. For short-time forecasting we use the Root Mean Square Error (RMSE) metric, while for long-time forecasting we use the Spectral Error metric. Both E_1 and E_2 assume there is no noise in the training data. Scores E_3 and E_5 evaluate the model's ability to de-noise training data – a process we call state "reconstruction" – in the low-noise and high-noise regimes respectively. Noise is defined as random Gaussian perturbations with mean 0 and variance σ^2 of the training data. E_4 and E_6 evaluate long-time forecasting with training data in the low-noise and high-noise regimes respectively. Low- and high-noise refers to the magnitude of the random perturbations to the training data. These scores are illustrated in Fig. 1(d).

E_7 and E_8 evaluate short-time and long-time forecasting in the limited training data regime without noise in the training data. E_9 and E_{10} evaluate short-time and long-time forecasting in the limited training data regime when there is noise in the training data. These scores are illustrated in Fig. 1(e).

E_{11} evaluates a model's capacity to interpolate within parameter regimes without noise. Specifically, the testing data is generated from a set of parameters within the parameter space used to generate the training data. E_{12} evaluates a model's capacity to extrapolate outside parameter regimes without noise. In this case, the testing data is generated from a set of parameters outside the parameters used to generate the training data. For evaluation, a burn-in matrix is provided to the model from the first few time-steps of the interpolated and extrapolated testing set. These scores are illustrated in Fig. 1(f).

5. ctf4science

We develop the open-source `ctf4science` repository for streamlining the adoption of our CTF. Our repository gives developers an API for performing all of the core functionality needed to train an existing model on our datasets, provides several examples of scientific ML models using our package, has our competition training data for the Lorenz and KS systems, and hyperparameter optimization configurations for all example models.

The core package provides four modules: `data_module`, `eval_module`, `tune_module`, and `visualization_module`. `data_module` and `eval_module` are unit tested, ensuring that the core data loading and evaluation functionality of the package is working properly.

`data_module` provides an interface for loading data from the simulated Lorenz and KS systems. We make it easy to generate validation splits for hyperparameter optimization, provide time steps for the data for models that expect it, and provide metadata such as spatial dimensions and the Δt value used to generate the data.

`eval_module` gives users the ability to evaluate their model directly on our metrics after training. We implement short- and long-time forecasting evaluation metrics and reconstruction scores. We also provide an interface for storing model scores in yaml files.

`tune_module` automates hyperparameter tuning using Ray Tune (Liaw et al., 2018). It handles individual or multiple model tuning with functionalities such as automatic resource allocation, ASHA (Asynchronous Successive Halving Algorithm) scheduling for early stopping, and configurable time or number of trial budgets (Li et al., 2020). Parallel trial execution is managed by Ray Tune and achieved by automatic detection of local CPU and GPU resources and their distribution based on defaults or user preference.

`visualization_module` gives users the ability to visualize the predictions and scores of their models. This simultaneously provides a way to debug their models as well as understand their model's strengths and limitations by visualizing the model's scores.

In addition to a simple-to-use API, we showcase fourteen already implemented models making full use of our framework from data loading to hyperparameter tuning. The models we provide are a baseline model predicting zeros, a baseline model predicting the average of the training data, DeepONet (Lu et al., 2021), SINDy (Brunton et al., 2016; Fasel et al., 2022), Reservoir models (Jaeger, 2001; Maass & Markram, 2004; Pathak et al., 2018), PyKoopman (Brunton et al., 2022; Pan et al., 2024), FNO (Li et al., 2021), KAN (Liu et al., 2025), OptDMD (Askham & Kutz, 2018), Spacetime (Zhang et al., 2023), HighOrder DMD (Le Clainche & Vega, 2017), PINN (Raissi et al., 2019), ODE-LSTM (Coelho et al., 2024), and LSTM (Hochreiter & Schmidhuber, 1997).

Combined, our modules and implemented models facilitate the rapid integration of new models to our CTF, paving the way for the scientific community to objectively and efficiently benchmark the performance of scientific ML models against one another.

6. Limitations & Future Work

We are launching `ctf4science` in a limited scope with fourteen models and two datasets: a dynamical system (Lorenz) and a spatio-temporal system (KS). The evaluation metrics test short- and long-time state forecasting and state reconstruction under the challenges of Gaussian noise, limited data, and parametric dependency. There are many more datasets and tasks that could and should be considered for science and engineering, most notably tasks in control. This CTF is an important first step to establish fair comparisons among modeling methods on truly withheld test sets. In future versions, more challenging datasets, real world datasets, more tasks, and more models will be integrated.

7. Conclusion

We developed a CTF that scores modeling approaches on a diversity of tasks that are prototypical in science and engineering. The canonical Lorenz and KS systems form an accepted testbench for demonstrating the effectiveness of modeling methods in scientific machine learning literature and act as the starting point of our framework. Our work builds a fair and multi-dimensional comparison between methods that is based on a true hidden testset—limiting the risk of "hacked" scores.

To enable the rapid adoption of our CTF, we provide the scientific community with a maintained, unit-tested, open-source framework for jump-starting the model evaluation process. In our repository, we provide a collection of popular scientific ML models that have already been ported to our framework. These implementations provide examples of how anyone can port their model into our framework to start comparing their models on our benchmarks. Our modules are unit-tested, provide an intuitive API for streamlining the development process and even includes an interface for hyperparameter optimization.

Our CTF invites researchers in the community to refine architectures and to co-create a truly comprehensive benchmarking suite for scientific machine learning, enabling the discovery of scientific breakthroughs and foundational world models.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Askham, T. and Kutz, J. N. Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 17(1):380–416, 2018.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113. URL https://www.pnas.org/doi/abs/10.1073/pnas.1517384113.

Brunton, S. L., Budišić, M., Kaiser, E., and Kutz, J. N. Modern Koopman Theory for Dynamical Systems. *SIAM Review*, 64(2):229–340, 2022. doi: 10.1137/21M1401243. URL https://doi.org/10.1137/21M1401243.

Coelho, C., Costa, M. F. P., and Ferrás, L. L. Enhancing continuous time series modelling with a latent ode-lstm approach. *Applied Mathematics and Computation*, 475:128727, 2024.

Deisenroth, M. P. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 465–472, 2011.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.

Donoho, D. 50 years of data science. *Journal of Computational and Graphical Statistics*, 26(4):745–766, 2017.

Dutta, S. *Reinforcement Learning with TensorFlow*. Packt Publishing Ltd, 2018.

Fasel, U., Kutz, J. N., Brunton, B. W., and Brunton, S. L. Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 478 (2260):20210904, 2022. doi: 10.1098/rspa.2021.0904. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2021.0904.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jaeger, H. "the 'echo state' approach to analyzing and training recurrent neural networks". Technical report, German National Research Center for Information Technology, Technical Report GMD 148, 2001.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Le Clainche, S. and Vega, J. M. Higher order dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 16(2):882–925, 2017. doi: 10.1137/15M1054924. URL https://doi.org/10.1137/15M1054924.

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., and Talwalkar, A. A system for massively parallel hyperparameter tuning, 2020. URL https://arxiv.org/abs/1810.05934.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E., and Stoica, I. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., and Tegmark, M. Kan: Kolmogorov-arnold networks, 2025. URL https://arxiv.org/abs/2404.19756.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Maass, W. and Markram, H. On the computational power of circuits of spiking neurons. *Journal of Computer and System Sciences*, 69(4):593–616, December 2004. ISSN 0022-0000. doi: 10.1016/j.jcss.2004.04.001. URL https://www.sciencedirect.com/science/article/pii/S0022000004000406.

McGreivy, N. and Hakim, A. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, 6(10):1256–1269, Oct 2024. ISSN 2522-5839. doi: 10.1038/s42256-024-00897-5. URL https://doi.org/10.1038/s42256-024-00897-5.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Pan, S., Kaiser, E., de Silva, B. M., Kutz, J. N., and Brunton, S. L. PyKoopman: A Python Package for Data-Driven Approximation of the Koopman Operator. *Journal of Open Source Software*, 9(94):5881, 2024. doi: 10.21105/joss.05881. URL https://doi.org/10.21105/joss.05881.

Pathak, J., Hunt, B., Girvan, M., Lu, Z., and Ott, E. Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. *Physical Review Letters*, 120(2): 024102, January 2018. doi: 10.1103/PhysRevLett.120.024102. URL https://link.aps.org/doi/10.1103/PhysRevLett.120.024102. Publisher: American Physical Society.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Ravichandiran, S. *Hands-On Reinforcement Learning with Python*. Packt Publishing Ltd, 2018.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362 (6419):1140–1144, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Zhang, M., Saab, K. K., Poli, M., Dao, T., Goel, K., and Ré, C. Effectively modeling time series with simple discrete state spaces. *arXiv preprint arXiv:2303.09489*, 2023.

A. Evaluation Metrics

The scores E_1 through E_{12} for the Lorenz and KS systems are presented formally.

A.1. Spatio-Temporal System: Kuramoto-Sivashinsky

For the KS system, the scores E_1 through E_{12} are computed using the following tests.

A.1.1. Forecasting (2 scores)

The first set of tasks, shown in Fig. 1-d, involve the approximation of the future state of the system. Thus, given a data matrix representing the dynamics over $t \in [0, 10T]$ ($\mathbf{X}_1 \in \mathbb{R}^{10m \times n}$), a generated forecast is requested from the model being tested for $t \in [10T, 11T]$ ($\mathbf{X}_{1pred} \in \mathbb{R}^{m \times n}$) to compare with the ground-truth ($\mathbf{X}_{1test} \in \mathbb{R}^{m \times n}$), with n being the dimension of the system and m being the number of forecasted time steps. The forecasting score is composed of two scores evaluating both the short-time forecast E_{ST}, which is computed using Root Mean Square Error (RMSE) between the test set and the model's approximation, and the long-time forecast E_{LT}, which is computed using the spectral error based upon the power spectral density, see Fig. 1-c. Short-time forecasting measures trajectory accuracy where deterministic prediction is feasible, while long-time forecasting measures statistical fidelity where only the system's broad statistical properties are recoverable.

For the challenge dynamics of interest, sensitivity of initial conditions is common, making long range forecasting to match the test set an unreasonable task given fundamental mathematical limitations with Lyapunov times. Thus, the long-time error is computed by least-squares fitting of the power spectrum $\mathbf{P}(\mathbf{X}, \mathbf{k}, k) = \ln(|\text{FFT}(\mathbf{X}[-k : -1, \mathbf{k}])|^2)$, where the **fftshift** has been used to model the data in the wavenumber domain and $\mathbf{k} = n/2 - k_{max} : n/2 + (k_{max} + 1)$ with $k_{max} = 100$. This means that we look at the match in the first 100 wavenumbers of the power spectrum over a long time simulation. Let $\hat{\mathbf{X}}$ be the ground-truth matrix, $\tilde{\mathbf{X}}$ be the prediction matrix, and $k \in (0, T)$ an integer specifying how to split the matrices for the short-time and long-time scores. The following two error scores are then computed:

$$S_{\text{ST}}(\tilde{\mathbf{X}}, \hat{\mathbf{X}}) = \frac{||\hat{\mathbf{X}}[1:k,:] - \tilde{\mathbf{X}}[1:k,:]||}{||\hat{\mathbf{X}}[1:k,:]||}, \tag{2}$$

$$S_{\text{LT}}(\tilde{\mathbf{X}}, \hat{\mathbf{X}}) = \frac{||\mathbf{P}(\hat{\mathbf{X}}, \mathbf{k}, k) - \mathbf{P}(\tilde{\mathbf{X}}, \mathbf{k}, k)||}{||\mathbf{P}(\hat{\mathbf{X}}, \mathbf{k}, k)||}. \tag{3}$$

It is clear that there are many ways to evaluate the long-range forecasting capabilities. We chose a simple and transparent metric fully understanding that more nuanced scoring could be used. To provide a reasonable range we then compute the two scores

$$E_1 = 100(1 - S_{\text{ST}}(\mathbf{X}_{1pred}, \mathbf{X}_{1test})), \quad E_2 = 100(1 - S_{\text{LT}}(\mathbf{X}_{1pred}, \mathbf{X}_{1test})), \tag{4}$$

meaning in each case a score of $E_i = 100$ corresponds to a perfect match. Note that, as a baseline, a solution guess of zeros $\tilde{\mathbf{X}}[1:k,:] = \mathbf{0}$ (corresponding also to $\mathbf{P}(\tilde{\mathbf{X}}, \mathbf{k}, k) = \mathbf{0}$) gives a score of $E_1 = E_2 = 0$.

Input: $\mathbf{X}_{1train} \in \mathbb{R}^{10m \times n}$; **Output:** $\mathbf{X}_{1pred} \in \mathbb{R}^{m \times n}$; **Scores:** E_1, E_2.

A.1.2. Noisy Data (4 scores)

The ability to handle noise is critical in all data-driven applications as sensors and measurement technologies are by default embedded with varying levels of noise. Methods that work with numerically accurate data, for example data points that are 10^{-6} accurate, may be useful for model reduction, but are rarely suitable for discovery and engineering design from real-world data. Both strong and weak noise are considered as these represent realistic challenges to be addressed in practice.

This task is very similar to Task 1, but now with noise added to the data. Specifically, the model is provided a data matrix $\mathbf{X}_{2train} \in \mathbb{R}^{10m \times n}$ and $\mathbf{X}_{3train} \in \mathbb{R}^{10m \times n}$ representing the evolution with low or high noise respectively. The objective is to first produce a reconstruction of the data itself, i.e. denoise the data to produce an estimate of the true state of the dynamics, $\mathbf{X}_{2pred}, \mathbf{X}_{4pred} \in \mathbb{R}^{10m \times n}$ for $\mathbf{X}_{2train}, \mathbf{X}_{3train}$ respectively, and the second objective is to then forecast the future state, matrices $\mathbf{X}_{3pred}, \mathbf{X}_{5pred} \in \mathbb{R}^{m \times n}$ for $\mathbf{X}_{2train}, \mathbf{X}_{3train}$ respectively. For the first task, a least-square fit is used between the approximation of the denoised data and the truth, and for the forecasting a long-time evaluation is computed leading to the following scores:

$$E_3 = 100(1 - S_{\text{ST}}(\mathbf{X}_{2pred}, \mathbf{X}_{2test})), \quad E_4 = 100(1 - S_{\text{LT}}(\mathbf{X}_{3pred}, \mathbf{X}_{3test})),$$
$$E_5 = 100(1 - S_{\text{ST}}(\mathbf{X}_{4pred}, \mathbf{X}_{4test})), \quad E_6 = 100(1 - S_{\text{LT}}(\mathbf{X}_{5pred}, \mathbf{X}_{5test})).$$

Input: $\mathbf{X}_{2train}, \mathbf{X}_{3train} \in \mathbb{R}^{10m\times n}$; **Output:** $\mathbf{X}_{2pred}, \mathbf{X}_{4pred} \in \mathbb{R}^{10m\times n}$, $\mathbf{X}_{3pred}, \mathbf{X}_{5pred} \in \mathbb{R}^{m\times n}$; **Scores:** E_3, E_4, E_5, E_6.

A.1.3. Limited Data (4 scores)

Data limitations are common in real world physical systems and often affect the success of data-driven methods. Thus, testing for model performance on low-data is critically important and provides important insight to potential users.

Figure 1-e demonstrates the nature of the task. In this case only a limited number of snapshots M on numerically accurate data are given $\mathbf{X}_{4train} \in \mathbb{R}^{M\times n}$. From this limited data, a forecast must be made which is evaluated with both error metrics (2) & (3) on the approximated future $\mathbf{X}_{6pred} \in \mathbb{R}^{m\times n}$. The experiment is repeated with noise on the measurements using the training matrix $\mathbf{X}_{5train} \in \mathbb{R}^{M\times n}$ for which a forecasting prediction matrix is produced $\mathbf{X}_{7pred} \in \mathbb{R}^{m\times n}$. The performance is evaluated on the following scores representing short and long-time metrics for both noise-free and noisy data respectively.

$$E_7 = 100(1 - S_{\text{ST}}(\mathbf{X}_{6pred}, \mathbf{X}_{6test})), \quad E_8 = 100(1 - S_{\text{LT}}(\mathbf{X}_{6pred}, \mathbf{X}_{6test})),$$
$$E_9 = 100(1 - S_{\text{ST}}(\mathbf{X}_{7pred}, \mathbf{X}_{7test})), \quad E_{10} = 100(1 - S_{\text{LT}}(\mathbf{X}_{7pred}, \mathbf{X}_{7test})).$$

Two error scores (analogous to E_1 and E_2) are produced for the noise-free and noisy limited data. These scores are E_7 (short) and E_8 (long) for the noise free case and E_9 (short) and E_{10} (long) for the noisy case.

Input: $\mathbf{X}_{4train}, \mathbf{X}_{5train} \in \mathbb{R}^{M\times n}$; **Output:** $\mathbf{X}_{6pred}, \mathbf{X}_{7pred} \in \mathbb{R}^{m\times n}$; **Scores:** E_7, E_8, E_9, E_{10}.

A.1.4. Parametric Generalization (2 scores)

Finally, the ability of a model to generalize to different parameter values is evaluated. For this case, the model's ability to interpolate and extrapolate to new parameter regimes is considered with noise-free data. The interpolation and extrapolation are each their own score, resulting in two scores that evaluate parametric dependence.

Figure 1-f shows the basic architecture of the task. Three training data sets are provided with three different (unknown) parameter values $\mathbf{X}_{6train}, \mathbf{X}_{7train}, \mathbf{X}_{8train} \in \mathbb{R}^{10m\times n}$. Construction of the dynamics in parametric regimes that are interpolatory $\mathbf{X}_{8pred} \in \mathbb{R}^{m\times n}$ and extrapolatory $\mathbf{X}_{9pred} \in \mathbb{R}^{m\times n}$ are required. For both of the tasks, a burn in matrix $\mathbf{X}_{9train}$ and $\mathbf{X}_{9train}$ respectively of size $M \times n$ (where $M < m$) is given and the performance is evaluated using the short-time metric (2).

$$E_{11} = 100(1 - S_{\text{ST}}(\mathbf{X}_{8pred}, \mathbf{X}_{8test})), \quad E_{12} = 100(1 - S_{\text{ST}}(\mathbf{X}_{9pred}, \mathbf{X}_{9test})).$$

Input: $\mathbf{X}_{6train}, \mathbf{X}_{7train}, \mathbf{X}_{8train} \in \mathbb{R}^{10m\times n}, \mathbf{X}_{9train}, \mathbf{X}_{10train} \in \mathbb{R}^{M\times n}$;

Output: $\mathbf{X}_{8pred}, \mathbf{X}_{9pred} \in \mathbb{R}^{m\times n}$; **Scores:** E_{11}, E_{12}.

A.2. Dynamical System: Lorenz

For the Lorenz system, the training and testing are identical as for the spatio-temporal KS system described above aside from the long-time forecast score. Data matrices for testing and training are of the same form as in Section A.1 with $n = 3$ being the dimension of the dynamical system. Since in this case there is no spatial coordinate it is no longer possible to use the power spectral density of the differential equation to evaluate the long-time performance. Instead, for this system, we evaluate the long-time forecasting based on the distribution of values in the state-space over the last k time steps (e.g. $k = 500$). For this we compare the histograms of the distribution of predicted and true solution trajectories in the following way. The histogram for a time series is computed using the histogram command (`histogram` in MATLAB and `numpy.histogram` in Python using the `numpy` package) with a set number of bins (e.g., `bins` $= 41$ for our current

Lorenz evaluation). The difference of the histogram between the truth (x, y and z) and prediction ($\tilde{x}, \tilde{y}$ and $\tilde{z}$) for each variable is measured in an ℓ_1-sense:

$$s_{\text{LT}}(x, \tilde{x}) = \frac{\|\texttt{histogram}(x, \texttt{bins}) - \texttt{histogram}(\tilde{x}, \texttt{bins})\|_1}{\|\texttt{histogram}(x, \texttt{bins})\|_1}.$$

From this the long-time error score for the Lorenz system is composed of the distributional error in each coordinate:

$$S_{\text{LT}}^{(\text{Lorenz})}(\mathbf{X}, \tilde{\mathbf{X}}) = \frac{s_{\text{LT}}(x, \tilde{x}) + s_{\text{LT}}(y, \tilde{y}) + s_{\text{LT}}(z, \tilde{z})}{3}.$$

As with the spatio-temporal system and the power spectral density, this gives a simple measure of the accuracy of the prediction from a statistical viewpoint since long-time prediction is well beyond the Lyapunov time which would not allow for a least-square match between trajectories of the truth and prediction.

A.3. Composite Score

We compute a composite score ($AvgScore$) per dataset from metrics E_1 through E_{12} by averaging the resulting scores for each method. This score is evaluated per method, not per model. Thus, each method can fit a model for each task and produce the best possible score. All scores are clipped such that $E_i \in [-100, 100]$, thus $AvgScore \in [-100, 100]$. Methods that cannot produce a result for a given task receive the minimum score -100.