Supplementary material for "Chaos as an interpretable benchmark for forecasting and data-driven modelling"

CONTENTS

I. DATA AVAILABILITY

The database of dynamical models and precomputed time series is available on GitHub at https://github.com/williamgilpin/dysts. The `benchmarks` subdirectory contains all code needed reproduce the benchmarks, figures, and tables in this paper.

All included equations are in the public domain, and all precomputed time series datasets have been generated *de novo* from these equations. No license is required to use these equations or datasets. The repository and precomputed datasets include an Apache 2.0 license. The author attests that they bear responsibility for copyright matters associated with this dataset.

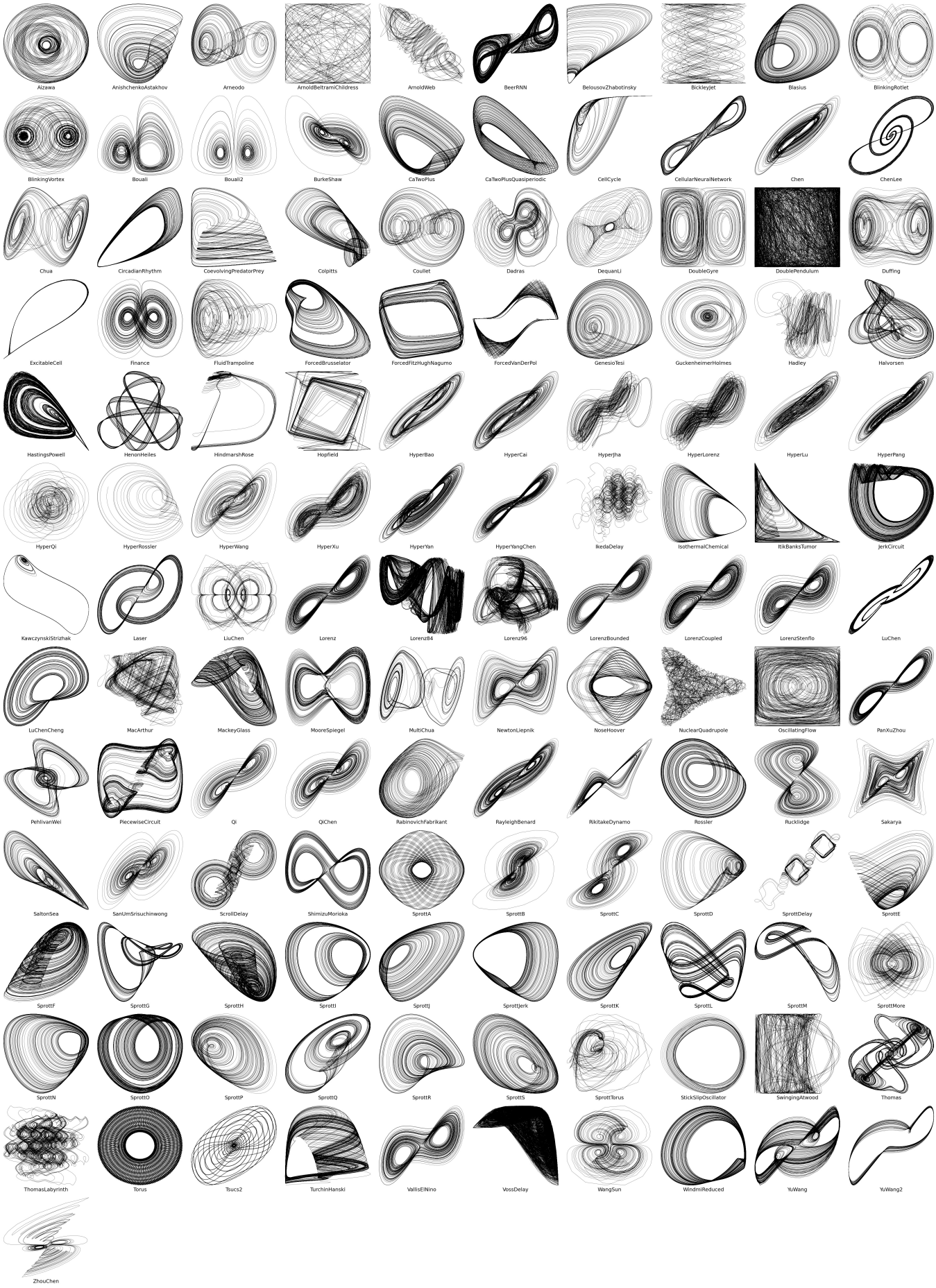

Figure S1. **All dynamical systems currently in the database.**

II. DESCRIPTIONS OF ALL SYSTEMS

Descriptions and citations for all systems are included below, and each system is visualized in Figure S1. Each system's entry in the project repository contains full records and descriptions.

System	Reference	Description
Aizawa	Aizawa, Yoji, and Tatsuya Uezu (1982). Topolog...	A torus-like attractor related to the forced L...
AnishchenkoAstakhov	Anishchenko, et al. Nonlinear dynamics of chao...	Stochastic resonance in forced oscillators.
Arneodo	Arneodo, A., Coullet, P. & Tresser, C. Occuren...	A modified Lotka-Volterra ecosystem, also know...
ArnoldBeltramiChildress	V. I. Arnold, Journal of Applied Mathematics a...	An exact solution of Euler's equation for invi...
ArnoldWeb	Froeschle, C., Guzzo, M. & Legga, E (2000). Gr...	A quasi-integrable system that transitions to ...
BeerRNN	Beer, R. D. (1995). On the dynamics of small c...	A two-neuron minimal model nervous system.
BelousovZhabotinsky	Gyorgyi and Field (1992). A three-variable mod...	A reduced-order model of the BZ reaction that ...
BickleyJet	Hadjighasem, Karrasch, Teramoto, Haller (2016)...	A zonal jet passing between two counter rotati...
Blasius	Blasius, Huppert, Stone. Nature 1999	A chaotic food web composed of interacting pr...
BlinkingRotlet	Meleshko & Aref. A blinking rotlet model for c...	The location of the mixer is chosen so that th...
BlinkingVortex	Aref (1984). Stirring by chaotic advection. J....	A classic minimal chaotic mixing flow. Solutio...
Bouali	Bouali (1999). Feedback loop in extended Van d...	Economic cycles with fluctuating demand. Relat...
Bouali2	Bouali (1999). Feedback loop in extended Van d...	A modified economic cycle model.
BurkeShaw	Shaw (1981). Zeitschrift fur Naturforschung.	A scroll-like attractor with unique symmetry a...
CaTwoPlus	Houart, Dupont, Goldbeter. Bull Math Biol 1999.	Intracellular calcium ion oscillations.
CaTwoPlusQuasiperiodic	Houart, Dupont, Goldbeter. Bull Math Biol 1999.	Intracellular calcium ion oscillations with qu...
CellCycle	Romond, Rustici, Gonze, Goldbeter. 1999.	A simplified model of the cell cycle. The para...
CellularNeuralNetwork	Arena, Caponetto, Fortuna, and Porto., Int J B...	Cellular neural network dynamics.
Chen	Chen (1997). Proc. First Int. Conf. Control of...	A system based on feedback anti-control in eng...
ChenLee	Chen HK, Lee CI (2004). Anti-control of chaos ...	A rigid body with feedback anti-control.
Chua	Chua, L. O. (1969) Introduction to Nonlinear N...	An electronic circuit with a diode providing n...
CircadianRhythm	Leloup, Gonze, Goldbeter. 1999. Gonze, Leloup...	The Drosophila circadian rhythm under periodic...
CoevolvingPredatorPrey	Gilpin & Feldman (2017). PLOS Comp Biol	A system of predator-prey equations with co-ev...
Colpitts	Kennedy (2007). IEEE Trans Circuits & Systems....	An electrical circuit used as a signal generator.
Coullet	Arneodo, A., Coullet, P. & Tresser, C. Occuren...	A variant of the Arneodo attractor
Dadras	S Dadras, HR Momeni (2009). A novel three-dime...	An electronic circuit capable of producing mul...
DequanLi	Li, Phys Lett A. 2008: 387-393.	Related to the Three Scroll unified attractor ...
DoubleGyre	Shadden, Lekien, Marsden (2005). Definition an...	A time-dependent fluid flow exhibiting Lagrang...
DoublePendulum	See, for example: Marion (2013). Classical dyn...	Two coupled rigid pendula without damping.
Duffing	Duffing, G. (1918), Forced oscillations with v...	A monochromatically-forced rigid pendulum, wit...
ExcitableCell	Teresa Chay. Chaos In A Three-variable Model O...	A reduced-order variant of the Hodgkin-Huxley ...
Finance	Guoliang Cai, Juanjuan Huang. International Jo...	Stock fluctuations under varying investment de...
FluidTrampoline	Gilet, Bush. The fluid trampoline: droplets bo...	A droplet bouncing on a horizontal soap film.
ForcedBrusselator	I. Prigogine, From Being to Becoming: Time and...	An autocatalytic chemical system.
ForcedFitzHughNagumo	FitzHugh, Richard (1961). Impulses and Physiol...	A driven neuron model sustaining both quiesent...
ForcedVanDerPol	B. van der Pol (1920). A theory of the amplitu...	An electronic circuit containing a triode.
GenesioTesi	Genesio, Tesi (1992). Harmonic balance methods...	A nonlinear control system with feedback.
GuckenheimerHolmes	Guckenheimer, John, and Philip Holmes (1983). ...	A nonlinear oscillator.
Hadley	G. Hadley (1735). On the cause of the general ...	An atmospheric convective cell.
Halvorsen	Sprott, Julien C (2010). Elegant chaos: algebr...	An algebraically-simple chaotic system with qu...
HastingsPowell	Hastings, Powell. Ecology 1991	A three species food web.
HenonHeiles	Henon, M.; Heiles, C. (1964). The applicabilit...	A star's motion around the galactic center.
HindmarshRose	Marhl, Perc. Chaos, Solitons, Fractals 2005.	A neuron model exhibiting spiking and bursting.
Hopfield	Lewis & Glass, Neur Comp (1992)	A neural network with frustrated connectivity
HyperBao	Bao, Liu (2008). A hyperchaotic attractor coi...	Hyperchaos in the Lu system.
HyperCai	Guoliang, Huang (2007). A New Finance Chaotic ...	A hyperchaotic variant of the Finance system.
HyperJha	Jürgen Meier (2003). Presentation of Attractor...	A hyperchaotic system.
HyperLorenz	Jürgen Meier (2003). Presentation of Attractor...	A hyperchaotic variant of the Lorenz attractor.
HyperLu	Jürgen Meier (2003). Presentation of Attractor...	A hyperchaotic variant of the Lu attractor.
HyperPang	Jürgen Meier (2003). Presentation of Attractor...	A hyperchaotic system.
HyperQi	G. Qi, M. A. van Wyk, B. J. van Wyk, and G. Ch...	A hyperchaotic variant of the Qi system.
HyperRossler	Rossler, O. E. (1979). An equation for hyperch...	A hyperchaotic variant of the Rossler system.
HyperWang	Wang, Z., Sun, Y., van Wyk, B. J., Qi, G. & va...	A hyperchaotic variant of the Wang system.
HyperXu	Letellier & Rossler (2007). Hyperchaos. Schola...	A hyperchaotic system.
HyperYan	Jürgen Meier (2003). Presentation of Attractor...	A hyperchaotic system.
HyperYangChen	Jürgen Meier (2003). Presentation of Attractor...	A hyperchaotic system.
IkedaDelay	K. Ikeda and K. Matsumoto (1987). High-dimensi...	A passive optical resonator system. A standard...

IsothermalChemical	Petrov, Scott, Showalter. Mixed-mode oscillati...	An isothermal chemical system with mixed-mode ...
ItikBanksTumor	Itik, Banks. Int J Bifurcat Chaos 2010	A model of cancer cell populations.
JerkCircuit	Sprott (2011). A new chaotic jerk circuit. IEE...	An electronic circuit with nonlinearity provid...
KawczynskiStrizhak	P. E. Strizhak and A. L. Kawczynski, J. Phys. ...	A chemical oscillator model describing mixed-m...
Laser	Abooee, Yaghini-Bonabi, Jahed-Motlagh (2013). ...	A semiconductor laser model
LiuChen	Liu, Chen. Int J Bifurcat Chaos. 2004: 1395-1403.	Derived from Sakarya.
Lorenz	Lorenz, Edward N (1963). Deterministic nonperi...	A minimal weather model based on atmospheric c...
Lorenz84	E. Lorenz (1984). Irregularity: a fundamental ...	Atmospheric circulation analogous to Hadley co...
Lorenz96	Lorenz, Edward (1996). Predictability: A probl...	A climate model containing fluid-like advectiv...
LorenzBounded	Sprott & Xiong (2015). Chaos.	The Lorenz attractor in the presence of a conf...
LorenzCoupled	Lorenz, Edward N. Deterministic nonperiodic fl...	Two coupled Lorenz attractors.
LorenzStenflo	Letellier & Rossler (2007). Hyperchaos. Schola...	Atmospheric acoustic-gravity waves.
LuChen	Lu, Chen. Int J Bifurcat Chaos. 2002: 659-661.	A system that switches shapes between the Lore...
LuChenCheng	Lu, Chen, Cheng. Int J Bifurcat Chaos. 2004: 1...	A four scroll attractor that reduces to Lorenz...
MacArthur	MacArthur, R. 1969. Species packing, and what ...	Population abundances in a plankton community,...
MackeyGlass	Glass, L. and Mackey, M. C. (1979). Pathologic...	A physiological circuit with time-delayed feed...
MooreSpiegel	Moore, Spiegel. A Thermally Excited Nonlinear ...	A thermo-mechanical oscillator.
MultiChua	Mufcstak E. Yalcin, Johan A. K. Suykens, Joos ...	Multiple interacting Chua electronic circuits.
NewtonLiepnik	Leipnik, R. B., and T. A. Newton (1981). Doubl...	Euler's equations for a rigid body, augmented ...
NoseHoover	Nose, S (1985). A unified formulation of the c...	Fixed temperature molecular dynamics for a str...
NuclearQuadrupole	Baran V. and Raduta A. A. (1998), Internationa...	A quadrupole boson Hamiltonian that produces c...
OscillatingFlow	T. H. Solomon and J. P. Gollub, Phys. Rev. A 3...	A model fluid flow that produces KAM tori. Ori...
PanXuZhou	Zhou, Wuneng, et al. On dynamics analysis of a...	A named attractor related to the DequanLi attr...
PehlivanWei	Pehlivan, Ihsan, and Wei Zhouchao (2012). Anal...	A system with quadratic nonlinearity, which un...
PiecewiseCircuit	A. Tamasevicius, G. Mykolaitis, S. Bumeliene, ...	A delay model that can be implemented as an el...
Qi	G. Qi, M. A. van Wyk, B. J. van Wyk, and G. Ch...	A hyperchaotic system with a wide power spectrum.
QiChen	Qi et al. Chaos, Solitons & Fractals 2008.	A double-wing chaotic attractor that arises fr...
RabinovichFabrikant	Rabinovich, Mikhail I.; Fabrikant, A. L. (1979...	A reduced-order model of propagating waves in ...
RayleighBenard	Yanagita, Kaneko (1995). Rayleigh-Bénard...	A reduced-order model of a convective cell.
RikitakeDynamo	Rikitake, T., Oscillations of a system of disk...	Electric current and magnetic field of two cou...
Rossler	Rossler, O. E. (1976), An Equation for Continu...	Spiral-type chaos in a simple oscillator model.
Rucklidge	Rucklidge, A.M. (1992). Chaos in models of dou...	Two-dimensional convection in a horizontal lay...
Sakarya	Li, Chunbiao, et al (2015). A novel four-wing ...	An attractor that arises due to merging of two...
SaltonSea	Upadhyay, Bairagi, Kundu, Chattopadhyay (2007)...	An eco-epidemiological model of bird and fish ...
SanUmSrisuchinwong	San-Um, Srisuchinwong. J. Comp 2012	A two-scroll attractor arising from dynamical ...
ScrollDelay	R.D. Driver, Ordinary and Delay Differential E...	A delay model that can be implemented as an el...
ShimizuMorioka	Shimizu, Morioka. Phys Lett A. 1980: 201-204	A system that bifurcates from a symmetric limi...
SprottA	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottB	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottC	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottD	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottDelay	Sprott, J. C (2007). A simple chaotic delay di...	An algebraically simple delay equation. A stan...
SprottE	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottF	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottG	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottH	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottI	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottJ	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottJerk	Sprott, J. C. Simplest dissipative chaotic flo...	An algebraidally simple flow depending on a th...
SprottK	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottL	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottM	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottMore	Sprott, J. C. (2020). Do We Need More Chaos Ex...	A multifractal system with a nearly 3D attractor
SprottN	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottO	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottP	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottQ	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottR	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottS	Sprott (1994). Some simple chaotic flows. Phys...	A member of the Sprott family of algebraically...
SprottTorus	Sprott Physics Letters A 2014	A multiattractor system that goes to a torus o...
StickSlipOscillator	Awrejcewicz, Jan, and M. M. Holicke (1999). In...	A weakly forced (quasiautonomous) oscillator w...
SwingingAtwood	Tufillaro, Nicholas B.; Abbott, Tyler A.; Grif...	A mechanical system consisting of two swinging...
Thomas	Thomas, Rene (1999). Deterministic chaos seen ...	A cyclically-symmetric attractor correspondng ...
ThomasLabyrinth	Thomas, Rene. Deterministic chaos seen in term...	A system in which trajectories seemingly under...

Torus	See, for example, Strogatz (1994). Nonlinear D...	A minimal quasiperiodic flow on a torus. All l...
Tsucs2	Pan, Zhou, Li (2013). Synchronization of Three...	A named attractor related to the DequanLi attr...
TurchinHanski	Turchin, Hanski. The American Naturalist 1997....	A chaotic three species food web. The species...
VallisElNino	Vallis GK. Conceptual models of El Nio and the...	Atmospheric temperature fluctuations with annu...
VossDelay	Voss (2002). Real-time anticipation of chaotic...	An electronic circuit with delayed feedback. A...
WangSun	Wang, Z., Sun, Y., van Wyk, B. J., Qi, G. & va...	A four-scroll attractor
WindmiReduced	Smith, Thiffeault, Horton. J Geophys Res. 2000...	Energy transfer into the ionosphere and magnet...
YuWang	Yu, Wang (2012). A novel three dimension auton...	A temperature-compensation circuit with an ope...
YuWang2	Yu, Wang (2012). A novel three dimension auton...	An alternative temperature-compensation circui...
ZhouChen	Zhou, Chen (2004). A simple smooth chaotic sys...	A feedback circuit model.

Table S2. Properties recorded for each chaotic system in the dataset

System Name	
Reference	A citation to published work or original source where available.
Description	A brief description of domain area, or original motivation for publication
Parameters	Parameters governing the differential equation (e.g for bifurcations)
Embedding Dimension	The number of dynamical variables, or the number set by default for delay equations
Unbounded Indices	Indices of dynamical variables that grow without bound (e.g. time for nonautonomous systems)
dt	The integration timestep, determined by surrogate testing of the power spectrum
Initial Conditions	Initial conditions on the attractor, determined by a long simulation discarding a transient
Period	The dominant timescale in the system, determined by surrogate testing of the power spectrum
Lyapunov Spectrum	The spectrum of Lyapunov exponents, measure of trajectory dispersion
Largest Lyapunov Exponent	The largest Lyapunov exponent, a measure of chaoticity
Correlation Dimension	The fractal dimension, a measure of geometric complexity
Kaplan-Yorke Dimension	An alternative fractal dimension, a measure of geometric complexity
Multiscale Entropy	A measure of signal complexity
Pesin Entropy	An upper bound on the entropy under discretized measurements
Delay	Whether the system is a delay differential equation
Hamiltonian	Whether the dynamics are Hamiltonian
Non-autonomous	Whether the dynamics depend explicitly on time

III. DATASET STRUCTURE AND FORMAT

All systems are primarily represented as Python objects, with names matching those in Figure S1 and the accompanying table. Underlying mathematical properties, parameters of the governing differential equation, recommended integration timestep and period, and default initial conditions are accessed as instance attributes. A callable implementation of the right hand side of the differential equation, a function for loading precomputed trajectories, and a function for re-integrating with default initial conditions and timescales, are included as instance methods. Additionally, we include a separate submodule for loading precomputed time series in bulk, or re-integrating all systems, which are useful for benchmarking tasks.

Our object representation abstracts the underlying records and metadata for each system, which are stored in a `JSON` file. The attributes recorded in the database file for each system are listed in Table S2.

For each dynamical system, we include 16 precomputed time series corresponding to all combinations of the following: coarse and fine sampling granularity, train and test splits emanating from different initial conditions, multivariate and univariate views, and trajectories with and without Brownian noise influencing the dynamics. The precomputed granularities correspond to a coarse granularity sampled at 15 points per period (the dominant timescale determined by surrogate testing on the power spectrum), and a fine granularity sampled at 100 points per period. The stochastically-forced trajectories correspond to adding a Langevin forcing term to the right hand side of each term in the dynamical equation. We used a scaled force with amplitude equal to to 1/40 the standard deviation of the values the dynamical variable takes on the attractor in the absence of noise. When integrating these trajectories, we use variant of the Runge-Kutta algorithm for stochastic differential equations [1], as implemented in the Python package `sdeint`.

IV. GLOSSARY

Here, we provide a glossary of several terms as they appear in the work presented here. More detailed treatments can be found in several references [2–5].

Attractor. A set of points within the state space of a dynamical system that most initial conditions approach over time. These points usually represent a subset of the full state space. In the work presented here, "attractor" and "dynamical attractor" are used interchangeably.

Bifurcation. A qualitative change in the dynamics exhibited by a dynamical system, as one or more system parameters is varied. For example, strange attractor can become a periodic orbit or fixed point as one of the parameters of the underlying dynamical equations is varied. Importantly, bifurcations occur as the result of changes to the underlying dynamical system, and do not in themselves result from the dynamics.

Dynamical System. A set of rules describing how points within a space evolve over time. Dynamical systems usually appear either as (1) systems of coupled ordinary differential equations, which can be integrated to produce continuous-time trajectories, or (2) discrete-time maps that send points at one timepoint to new points a fixed

interval Δt later. In the context of the work presented here, a dynamical system is a single set of deterministic ordinary differential equations (e.g. the Lorenz system).

Entropy. A statistical property of a dynamical system corresponding to the gain of information over time as the system is observed. A highly regular and predictable process will have low entropy, while a stochastic process will have high entropy. Unlike dimensionality, the entropy of a system typically does not require a notion of distance on the state space. For example, if different regions of an attractor are colored with discrete labels, it is possible to define the entropy of a trajectory based on the sequence of symbols it passes through—without referencing the precise locations visited, or the distance among the symbols.

Ergodic. A property of a dynamical system specifying that, over sufficiently long timescales, the system will visit all parts of its state space. A dissipative dynamical system will not be ergodic over its full state space, but it may be ergodic once it settles onto an attractor. In the context of time series analysis, ergodicity implies that a forecasting model trained on many short trajectories initialized at different points on an attractor will have the same properties as a model trained on subsections of a single long trajectory.

Fractal. A set of points that appears self-similar over all length scales. Fractals have dimensionality intermediate to traditional mathematical objects like lines and surfaces, resulting in a diffuse appearance.

Initial Conditions. A point within the state space of a dynamical system. As time passes, the rules specifying the dynamical system will transmit this point to other points within the system's state space. An initial condition does not necessarily lie on an attractor of the dynamical system.

Limit Cycle. A type of attractor in which trajectories undergo recurring periodic motion. A swinging, frictionless pendulum exhibits a limit cycle.

Lyapunov Exponent. The initial growth rate of an infinitesimal perturbation to a point within a dynamical system's state space. If two initial conditions are chosen with infinitesimal initial separation, then as time passes the two points will spread apart exponentially. The logarithm of the rate of change in their separation equals the Lyapunov exponent. For non-chaotic systems (such as systems evolving along regular limit cycles), neighboring points do not diverge, and so the Lyapunov exponent is zero. When used in reference to an entire attractor, the Lyapunov exponent corresponds to an average over all points on the attractor.

Quasiperiodic Motion. A type of attractor corresponding to non-repeating continuous motion, which does not exhibit fractal structure. The dynamics contain at least two frequencies that are incommensurate with one another. Quasiperiodic attractors have integer fractal dimension and a surface-like appearance, in contrast to the diffuse appearance of strange attractors.

Stable Fixed Point. A type of attractor in which trajectories converge to a single location within the state space.

State Space. The set of all possible states of a dynamical system. Initial conditions, trajectories, and attractors are all subsets of this space.

Strange Attractor. An attractor in which trajectories continuously wander over a bounded region in state space, but never stop at a fixed point or settle into a repeating limit cycle. The dynamics are therefore globally stable, but locally unstable: the attractor contains a dense set of unstable periodic orbits, and trajectories briefly shadow individual orbits before escaping onto others. These unstable orbits span a continuous range of frequencies, producing motion at a range of length scales—and resulting in the fractal appearance of strange attractors.

Trajectory. A set of points corresponding to the locations to which a given initial condition is mapped by a dynamical system. Trajectories are continuous curves for continuous-time systems, and isolated points for discrete-time maps.

V. CALCULATION OF MATHEMATICAL PROPERTIES

For all mathematical properties we perform 20 replicate computations from different initial conditions, and record the average in our database. To ensure high-quality estimates, we compute trajectories at high granularity of 500 points per period (as determined by the dominant frequency in the power spectrum), and we use trajectories with length 2500, corresponding to five complete periods.

Timescale alignment. All systems in our database have been timescale-aligned, allowing them to be re-integrated at equivalent dominant timescales and sampling rates. This feature differentiates our approach from other time series collections, as well as previous applications of data-driven models to ordinary differential equations, and it allows easier comparison among systems. In order to align timescales, for each system we calculate the optimal integration timestep by computing the power spectrum, and then using random phase surrogates in order to identify the smallest and dominant significant significant frequencies [6]. The smallest frequency determines the integration timestep when re-integrating each system, while the highest amplitude peak in the power spectrum determines the dominant significant frequency, and thus the governing timescale. We use the dominant timescale to downsample integrated dynamics, ensuring consistency across systems. We record both fields in our database.

Lyapunov Exponents. We implement standard techniques for computing Lyapunov exponents [7–9]. Our basic approach consists of following a bundle of vectors along a trajectory, and at each timestep using the Gram-Schmidt procedure to re-orthonormalize the bundle. The stretching rates of the principal axes provide estimates of the Lyapunov exponents in each direction.

When determining the Lyapunov exponents, for each initial condition we continue integration until the smallest-magnitude Lyapunov exponent drops below our tolerance level of 10^{-8}, because all continuous time systems have at least one zero-magnitude exponent. Our replicate spectrum estimates across initial conditions are averaged with weighting proportional to the distance between the smallest magnitude exponent and zero, in order to produce a final estimate.

Fractal Dimension. We compute the fractal dimension using the Grassberger-Procaccia algorithm for the correlation dimension, a robust nonparametric estimator of the fractal dimension that can be calculated deterministically from finite point sets [10].

Entropy. The multiscale entropy was used to estimate the intrinsic complexity of each trajectory [11]. While a multivariate generalization of the multiscale entropy has recently been proposed [12], due to convergence issues we calculate the entropy separately for each dynamical variable, and then record the median across all coordinates. Because this approach fails to take into account common motifs across multiple dimensions, we expect that our calculations overestimate the true entropy of the underlying systems. A similar effect occurs when mutual information is computed among subsets of correlated variables.

Additional mathematical properties. We derive and record in our database several properties derived from the spectrum of Lyapunov exponents, including the Pesin's upper bound on the entropy (the sum of all positive Lyapunov exponents) and the Kaplan-Yorke fractal dimension (an alternative estimator of the fractal dimension) [5, 6].

VI. STATISTICAL FEATURES AND EMBEDDING

For each dynamical system, we generate 40 trajectories of length 2000 originating from random initial conditions on the attractor. We use the default granularity of 100 points per dominant period as determined by Fourier transform. For each system and replicate, we compute 787 standard common time series features using standard methods [13]. For each dynamical system and replicate, we drop all null features, and then use an inner join operation to retain only features that appear across all dynamical systems and replicates. We then retain only the 100 features with the highest variance relative to their mean values across all dynamical systems.

We use these features to generate an embedding with UMAP [14]. We repeat this procedure for each of the 40 random initial conditions that were featurized for each dynamical system, and we report the median across replicates as the embedding of the dynamical system. We use affinity propagation with default hyperparameters in order to identify eight clusters within the embedding [15].

VII. FORECASTING EXPERIMENTS

Benchmarks are computed on the Harvard FAS Cannon cluster, using two Tesla V100-PCIE-32GB GPU and 32 GB RAM per node. Benchmarks are implemented with the aid of the `darts`, `GluonTS`, and `sktime` libraries [16–18].

Models. We include forecasting models from several domains: deep learning methods (NBEATS, Transformer, LSTM, and Temporal Convolutional Network), statistical methods (Prophet, Exponential Smoothing, Theta, 4Theta), common machine learning techniques (Random Forest), classical forecasting methods (ARIMA, AutoARIMA, Fourier transform regression), and standard naive baselines (naive mean, naive seasonal, naive drift) [17, 19–21]. All non-tuned hyperparameters (e.g. training epochs, number of layers, etc) are kept at default values used in reference implementations included in the `darts`, `GluonTS`, and `sktime` libraries [16–18].

Hyperparameter tuning. Hyperparameter tuning is performed separately for each forecasting model, dynamical system, and sampling granularity. The training set for each attractor consists of a single train time series comprising a trajectory emanating from a random location on the chaotic attractor. For each trajectory, 10 full periods are used to train the model, and 2 periods are used to generate forecast mean-squared-errors to evaluate combinations of hyperparameters. These splits correspond to 150 and 30 timepoints for the coarse granularity datasets, and 1000 and 200 timepoints for the fine granularity datasets.

Because benchmarks are computed on both coarse and fine granularities, different value ranges are searched for the two granularities: 1 timepoint, 5 timepoints, half of a period (8 timepoints for the coarse granularity, 50 timepoints for the fine granularity), and one full period (15 timepoints / 100 timepoints). For forecast models that accept a seasonality hyperparameter, the presence of additive seasonality (such as monochromatic forcing) is treated as an

additional hyperparameter. A standard grid search is used to find the best sets of hyperparameters separately for each model, system, and granularity.

Scoring. The testing dataset consists of a single time series emanating from another point on the same attractor. On this trajectory, a model is trained on the first 10 periods using the best hyperparameters the train dataset, and the forecast score is generated on the remaining 2 periods of the testing time series. Several standard time series similarity metrics are recorded for each dynamical system and forecasting model: mean absolute percentage error (MAPE), symmetric mean absolute percentage error SMAPE, coefficient of variation (CV), mean absolute error (MAE), mean absolute ranged relative error (MARRE), mean squared error (MSE), root mean squared error (RMSE), coefficient of determination (r^2), and mean absolute scaled error (MASE).

A. The effect of noise on forecasting results.

In order to determine the robustness of our experimental results to the presence of non-deterministic noise in the dataset, we perform a full replication of our experiments above on a modified dataset that includes noise. For each dynamical system, the scale of each dynamical variable is determined by generating a reference trajectory without noise, and calculating the standard deviation along each dimension. A new trajectory is then generated with noise of amplitude equal to 20% of the scale of each dynamical variable. Figure S2 shows the result of our benchmarks with noise, compared to our benchmarks in the absence of noise.

As expected, the median forecasting performance degrades for all methods in the presence of noise. Noise only weakly affects the naive baselines, because the range of values present in the data remains the same in the presence of noise. The deep learning models continue to perform very well, consistent with general intuition that large, overparametrized models effectively filter low-information content from complex signals [22]. Interestingly, the performance of the random forest model noticeably degrades with noise, suggesting that the representation learned by the model is fragile in the presence of extraneous information from noise. Conversely, the simple Fourier transform regression performs better than several more sophisticated models in the presence of noise. We hypothesize that high-frequency noise disproportionately obfuscates phase information within the signal, and so forecasting models that project time series onto periodic basis functions (e.g., Fourier and N-BEATS) are least impacted.

VIII. FORECASTING EXPERIMENTS AS GRANULARITY AND NOISE ARE VARIED

In order to better understand how the performance of different forecasting models depends on properties of the time series, we perform a set of experiments in which we re-train all forecasting models on datasets with a range of granularities and noise levels. We define noise level the same way as in our forecasting experiments: a noise level of 0.2 corresponds to a noise amplitude equal to 20% of the normal standard deviation of the signal. Granularity refers to the number of points sampled per period, as defined by the dominant significant frequency in the power spectrum. For these experiments, the same hyperparameters are used as for the original forecasting experiments. However, for the granularity sweep, hyperparameters that have units equivalent to timescale (e.g. number of time lags, or input chunk size) are rescaled by the granularity.

The results are shown in Figure S3. We find that forecasting models are most strongly differentiated at low noise levels, and that as the noise level exceeds the average amplitude of the signal the performance of models converges. This effect arises because there is less useable information in the signal for forecasting. However, the relative ranking of the different models remains somewhat stable as noise intensity increases, suggesting that the deep learning models remain effective at extracting relevant information even in the presence of dominant noise.

The granularity results show that the relative performance of different forecasting models is stable across granularities, and that the deep learning models (and particularly NBEATS) continue to perform well across a range of granularities. However, unlike the statistical methods, the performance of the deep learning models fluctuates widely across granularities, and in a systematic manner that cannot be attributed to sampling error—all points and rankings are averages over all 131 systems. These results suggest that more complex models may have timescale bias in their default architectures. However, we caution that exhaustive (albeit computationally expensive) hyperparameter tuning is needed to further understand this effect.

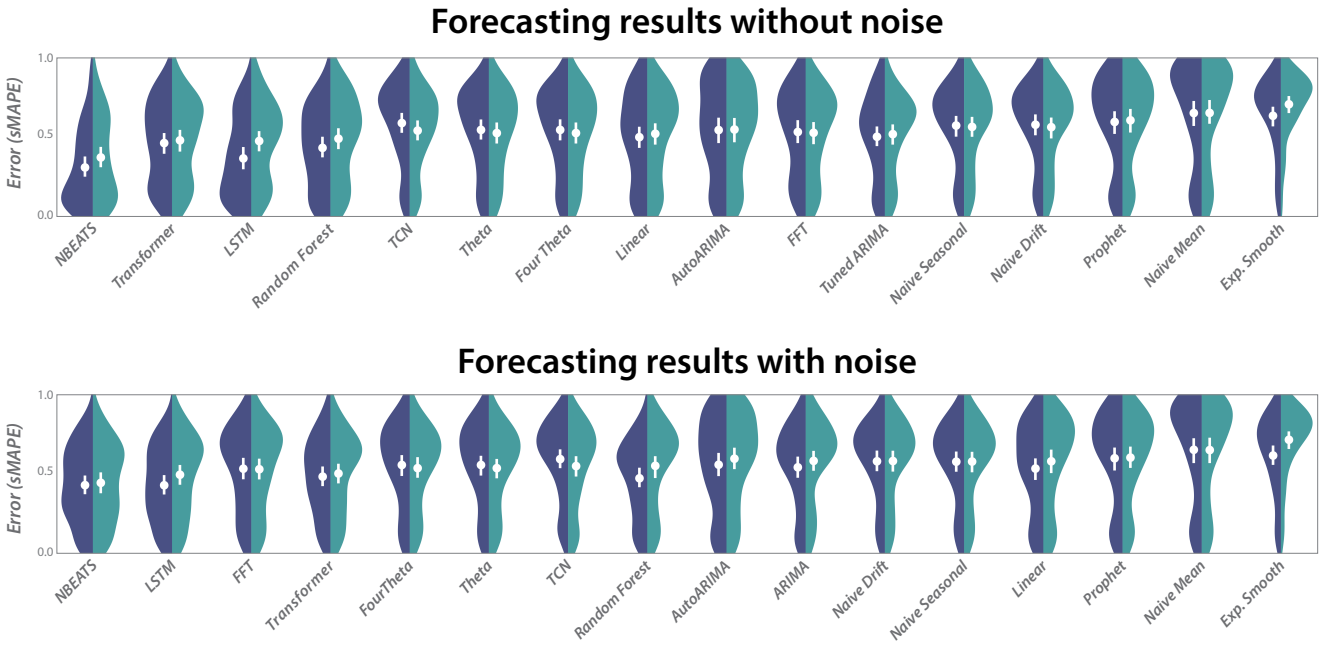

Figure S2. **Forecasting results with and without noise.** Each panel shows the distribution of forecast errors for all dynamical systems across different forecasting models, sorted by increasing median error. Dark and light hues correspond to coarse and fine time series sampling granularities. Upper panel corresponds to results for the full chaotic systems collection without noise, and lower panel corresponds to results from replicate experiments in which noise is present. Note that the model order along the horizontal axis differs between the two panels, because the relative performance of the different forecasting methods changes in the presence of noise.

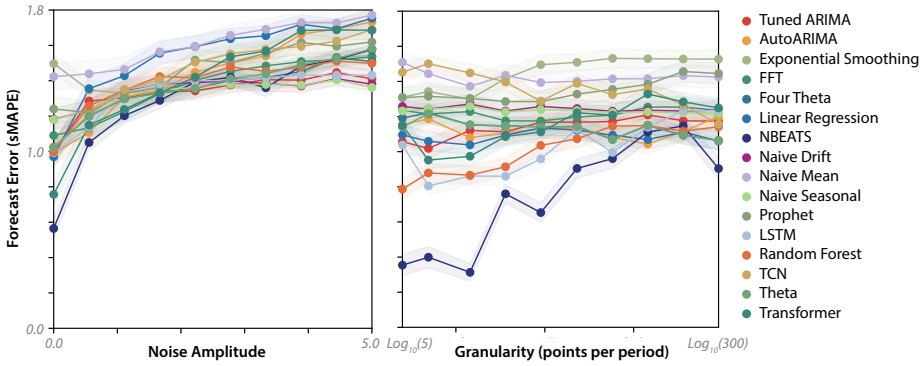

Figure S3. **Variation in forecasting model performance as noise level and granularity are varied.** Points and shaded ranges correspond to medians and standard errors across dynamical systems.

IX. RELATIVE PERFORMANCE OF FORECASTING MODELS ACROSS DIFFERENT MATHEMATICAL PROPERTIES

In order to determine whether different forecasting models are better suited to different types of dynamical system, we analyze our forecasting benchmarks striated by different mathematical properties of the dynamical systems. For a given mathematical property (such as Lyapunov exponent), we select only the dynamical systems among the bottom 20% of systems (i.e. the least chaotic systems), and we compute the average forecast error for each forecasting model on just this group. We repeat the analysis for the dynamical systems in the quantile $10-30\%$, then $20-40\%$, and so forth in order to determine how forecasting performance of each model type varies with level of chaoticity. We repeat the analysis for the correlation dimension and multiscale entropy. Our results are shown in Figure S4

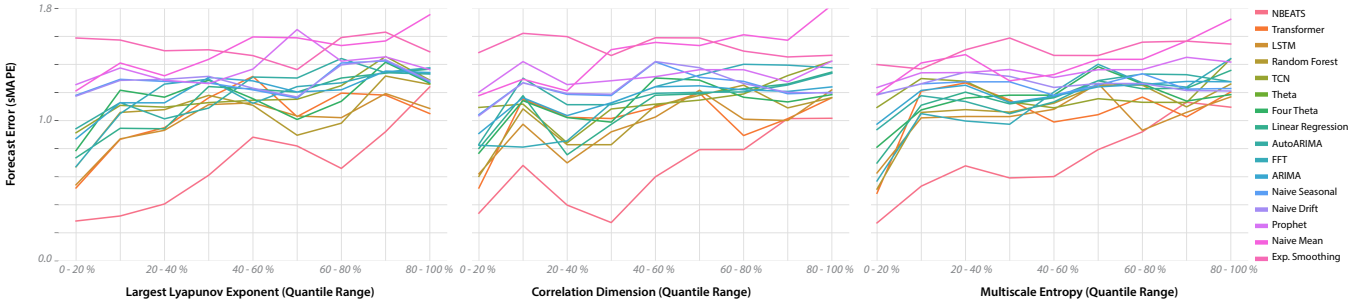

Figure S4. **Variation in forecasting model performance across different mathematical properties.** The horizontal axis of each plot corresponds to a sliding window comprising a 20% quantile in the property across all systems. Points correspond to medians across all dynamical systems in that quantile.

X. IMPORTANCE SAMPLING EXPERIMENTS

Our importance sampling experiment consists of a modified version of our forecasting task. We choose a single model, the LSTM, and alter its training procedure in order to determine how it is affected by alternative sampling strategies. In order to control for unintended interactions, we use a single set of hyperparameters for models trained on all chaotic systems, corresponding to the most common values from our forecasting benchmark. As a result, the baseline forecast error is higher across the chaotic systems dataset compared to our forecasting experiments, in which the LSTM was tuned separately for each chaotic system.

Our procedure consists of the following: (1) We halt training every few epochs and compute historical forecasts (backtests) on the training trajectory. (2) We randomly sample timepoints proportionately to their error in the historical forecast, and then generate a set of initial conditions corresponding to random perturbations away from each sampled attractor point. (3) We simulate the full dynamical system for $\tau = 150$ timesteps for each of these initial conditions, and we use these new trajectories as the training set for the next $b = 30$ epochs. We repeat this procedure for $\nu = 5$ meta-epochs. For the original training procedure, the training time scales as $\sim B = 400$, the number of training epochs times the number of timepoints in a full trajectory.

For the control "full epoch" baseline, we use the standard training procedure. For our "random batch" control experiments, we repeat the importance sampling procedure, but randomly sample timepoints, rather than weighting points by their backtest error. We include this control in order to account for the possibility of forecast error decreasing with total training data, an effect that would lead the importance sampling procedure to perform well spuriously.

XI. TRANSFER LEARNING EXPERIMENTS

For our classification experiments, we start with the 128 tasks currently within the UCR time series classification archive, and we narrow the set to the 96 datasets that contain at least 100 valid timepoints [23].

Our autoencoder is based on a causal dilated architecture recently shown to provide competitive performance among unsupervised embedding methods on the UCR archive [24]. Following previous work, our encoder comprises a single causal convolutional block [25], containing two causal convolutions with kernel size 3 and dilations of 2. A convolutional residual connection bridges the input layer and the latent layer, and leaky ReLU activations are used throughout. Unlike previous studies that learned embeddings using a triplet loss (thereby eliminating the need for a decoder) [24], we use a standard decoder similar to our previous study on chaotic system embedding [26], consisting of a three-layer standard convolutional network with ELU activation functions. We train our models using the Adam optimizer with mean squared error loss and a learning rate of 10^{-3} [27]. Our PyTorch network implementations are included in the project repository.

We train separate encoders for each classification task in the UCR archive. Briefly, we retrieve the training dataset for a given classifation task, and we use phase surrogate testing to determine the dominant frequency in the training data. We then convert this timescale into an effective granularity (in points per dominant period) for the training data. We then re-integrate all 131 dynamical systems within our dataset, with a granularity setting set to match the training data. We train the autoencoder on these trajectories, and we then apply the encoder to the training data of the classification task, in order to generate a featurized time series. For our "random timescale" ablation experiment, we select random granularities unrelated to the training data, and otherwise repeat the procedure above.

Having obtained encoded representations of the classification task training data, we then convert the training data

into a featurized representation using `tsfresh`, a suite that generates 787 standard time series features (such as number of peaks, average power, wavelet coefficients) [13]. We then pass these features to a standard ridge regression classifier, which we set to search for α values over a range 10^{-3} – 10^{3} via cross-validation [15]. Our approach to classifying time series is based upon recent methods for generating classification results from features learned from time series in an unsupervised setting, which found that complex unsupervised feature extractors followed by supervised linear classification yield competitive performance [28]. For our "no transfer learning" baseline, we apply the featurization and regression to the bare original training data for the classification problem.

Our reported scores correspond to accuracy on the test partition of the UCR archive. The timescale extraction, surrogate data generation, autoencoder, `tsfresh` featurization, and ridge classifier cross-validation steps are all trained only on the training data, and the trained encoder, `tsfresh` featurization, and ridge classifer are applied to the test data.

XII. SYMBOLIC REGRESSION EXPERIMENTS

Our symbolic regression dataset consists of input values corresponding to points along a trajectory, and target values corresponding to the value of the right hand side of the governing differential equation at those points. For our benchmark, we generate train and test datasets corresponding to trajectories originating from different locations on the attractor. Because we are interested in performance using information sampled across the attractor, we generate long trajectories (10 full periods, as determined by dominant timescale in power spectrum) at low sampling granularity (15 points per period), for a total of 150 datapoints in each of the train and test trajectories. This number of points is comparable to existing benchmarks [29]. While, in principle, random inputs could be generated and used to produce output values for our differential equations, because our target formulae correspond to dynamical systems, we favor using trajectories—which would best simulate observations from a real-world system. As we note in the main text, the accuracy of the target formulae will likely be reduced in regions of the attractor with lower measure.

For PySINDY, we fit separate models with purely polynomial and purely trigonometric bases. For DSR and pySR, we use default hyperparameters, and allow a fixed library of binary and unary expressions, $\{+, -, \times, \div\}$, $\{\sin, \cos, \exp, \log, \tanh\}$ [30]. Because our dynamical systems are multivariate, we fit separate expressions to each dynamical variable, and record the median across dynamical variables as the overall error for the system.

We apply the expressions generated by symbolic regression to the unseen test trajectory, and we treat the resulting values as forecasts. We therefore record the same error metrics as for our forecasting benchmark above.

XIII. NEURAL ORDINARY DIFFERENTIAL EQUATION EXPERIMENTS

We perform a preliminary neural ordinary differential equation (nODE) experiment, in order to evaluate whether mathematical properties of a dynamical system influence the properties of a fitted nODE. We design our experiment identically to our fine-granularity forecasting benchmark above: for each system, a multivariate training trajectory consisting of 1000 timepoints is used to train a nODE model [31]. An unseen "test" initial condition is then randomly chosen, and 200 timepoint trajectories are generated using both the true dynamical system, and the trained neural ODE. The quality of the resulting trajectory is evaluated using the sMAPE error between the predicted and true trajectory.

Our results are shown in Figure S5. Overall, the forecasting performance of the nODE model is competitive with other time series forecasting techniques, with the advantage of producing a differentiable representation of the underlying process that can potentially be used for downstream analysis. Qualitatively, we observe that the nODE dynamics frequently become trapped near unstable periodic orbits over long durations, suggesting that shadowing events observed in the training data dominate the learned representation [2].

Unlike our symbolic regression experiments, we find that there is no significant correlation between the quality of a nODE model and any underlying properties of the differential equations. Among the various mathematical properties (Lyapunov exponents, fractal dimension, etc) the largest observed Spearman correlation was not significantly different from zero (0.072 ± 0.003, median with standard error determined by bootstrapping),

XIV. DATASHEET: DATASET DOCUMENTATION AND INTENDED USES

The primary inclusion criteria for dynamical systems is appearance in published work with explicit equations and parameter values provided that created chaotic dynamics. While there are infinite possible chaotic attractors, our collection surveys systems as they appear in the literature—which primarily comprises particular domain-area

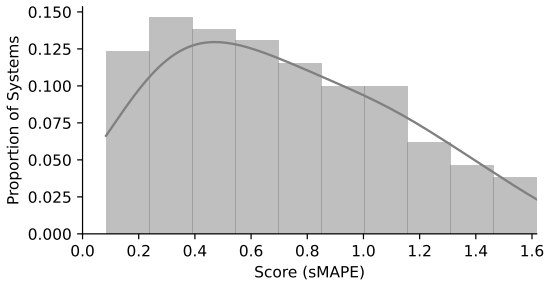

Figure S5. Distribution of error scores for the neural ordinary differential equations benchmark.

applications, as well as systems with particular mathematical properties. Below, we address the questions included in an existing dataset datasheet guide [32].

1. Motivation

Purpose This dataset was created for the purpose of providing a generative benchmark for time series mining applications, in which arbitrary synthetic data can be generated using a deterministic process.

Unintended Uses To our knowledge, there are no pressing uses for this data that could cause unintended harm. However, insofar as our dataset can be used to improve existing time series models (illustrated by our time series classification benchmark), there is a possibility of our dataset contributing to privacy concerns with time series analysis—particularly by making it possible for large models to identify latent factors that could, for example, de-anonymize physiological recordings [33]. In our project repository, we include instructions asking users who become aware of any unintended harms to submit an issue on GitHub.

Previous Uses Some time series analysis utilities and specific systems in this repository were used in our previous work [26], but the full dataset and benchmarks are all new.

Creator and Funding. This repository was created by William Gilpin, with support from the NSF-Simons Center for Quantitative Biology at Harvard University, as well as the University of Texas at Austin. No special funding was solicited for this project.

2. Composition

Instances. Each instance in this dataset comprises a set of nonlinear differential equations describing a chaotic process, a set of standard parameter values and initial conditions, a set of default timescales and integration timesteps, a set of characteristic mathematica properties, a citation to a published source (where available), a brief description of the system, and 16 precomputed trajectories from the system under various granularities and initial conditions.

Instance Relationships. Each instance corresponds to a different dynamical system.

Instance Count. At time of writing, there are 131 continuous-time dynamical systems (126 ordinary differential equations, and 5 delay equations). There are also 30 discrete-time chaotic maps, however we do not include these in any analyses or discussion presented here.

Instance Scope. Each instance corresponds to a particular realization of a dynamical system, based on previously-published parameter values and initial conditions. In principle, an infinite number of additional chaotic systems exists; our dataset seeks to provide a representative sample of published systems.

Labels. Each trajectory and system contains metadata describing its provenance, however there is not a particular label associated with each trajectory. However, all systems are labelled a variety of annotations that can, in principle, be used as labels (see Table S2).

External Dependencies. The data itself has no external dependencies. Simulating each system requires several standard scientific Python packages (enumerated in the repository README file). Running the benchmarks requires several additional dependencies, which are also listed in the README.

Data Splits. No splits are baked-in, because (in principle) arbitrary amounts of training, validation, and testing data can be generated for each dynamical system. Splits can either be performed by holding out some timepoints, or (for multivariate systems) by splitting the set of dynamical variables. For the purpose of benchmarking experiments, splits corresponding to 10 periods of training data, and 2 periods of unseen prediction/validation data, were used for both the train and test datasets (the test dataset corresponds to an unseen initial condition). For the fine granularity

time series, this corresponds to splits of 1000/200 for both the train and test initial conditions. For the coarse granularity time series, this corresponds to a split of 150/30. The data loader utilities included in the Python library use the 10 period / 2 period split by default.

Experiments. All benchmark experiments are described at length in our preprint. They primarily consist of forecasting benchmarks, generative experiments (importance sampling and model pretraining), and data-driven model inference experiments.

3. Collection

Collection. ISI Web of Science was used to identify papers claiming novel low-dimensional chaotic systems published after 1963 (the year of Lorenz's original paper). Papers were sorted by citations in order to determine priority for re-implementation, and systems were only included that had (1) explicit analytical expressions and (2) published parameter values and initial conditions leading to chaos. All systems were re-implemented in Python and checked to verify that the reported dynamics were chaotic. Additionally, several previous collections and galleries of chaos were checked, to ensure that all entries are included [9, 34–36].

Workers. All individuals involved in data collection and curation are authors on the paper.

Timeframe. Data was collected from 2018 – 2021.

Instance Acquisition. Each dynamical system required implementation in Python of the stated dynamical equations, as well as all parameter values and initial conditions leading to chaos. Each system was then numerically integrated in order to ensure that the observed dynamics matched those claimed in the original publication. Once chaos was validated, the integration timestep and the trajectory sampling rate were determined using the power spectrum, with time series surrogate analysis used to identify significant frequencies. Once the correct timescales were known, properties such as the Lyapunov exponents and entropy were calculated. For all trajectory data and initial conditions, a long transient was discarded in order to ensure that the dynamics settled onto the attractor.

Instance Scope. There are effectively an infinite number of possible chaotic dynamical systems, even in low dimensions. However, our collection represents a sample of named and published chaotic systems, and it includes most well-known systems.

Sampling. Because our dataset comprises only named and published chaotic systems, it does not comprise a representative sample of the larger space of all low-dimensional chaotic systems. Therefore, our database should not be used to compute any quantities that depend on the measure of chaotic systems within the broader space of all possible dynamical systems. For example, a study that seeks to identify the most common features or motifs of chaotic systems cannot use our database as representative sample. However, our database does comprise a representative sample of chaotic dynamics as they appear in the literature.

Missing Information. For systems in which a reference citation or additional context is unavailable, the corresponding field in the metadata file is left blank. However, all systems have sufficient information to be integrated.

Errors. If any errors or redundancies are identified, we encourage users to submit an issue via GitHub.

Noise. Noise can be added to the trajectories either by adding random values to each observed timepoint (measurement noise), or performing a stochastic simulation (stochastic dynamics). A stochastic integration function is included in the Python library. The precomputed trajectories associated with each system include trajectories with noise.

4. Preprocessing

Cleaning. Dynamical systems may be numerically integrated with arbitrary precision, and their dynamics can be recorded at arbitrarily small intervals. In order to report all systems consistently, we use time series phase surrogate testing to identify the highest significant frequency in the power spectrum of each system's dynamics. We then set the numerical integration timestep to be proportional to this timescale. We then re-integrate, and use surrogates to identify the dominant significant frequency in each system's dynamics. We use this timescale to determine the sampling rate. This process ensures overall that all systems exhibit dynamical variation over comparable timescales, and that the integration timestep is sufficiently small to accurately resolve the dynamics.

Having determined the appropriate integration timescales, we then determine the Lyapunov exponents, average period, and other ensemble-level properties of each dynamical system. We compute these quantities for replicate trajectories originating from different initial conditions on the attractor, and record the average.

For each fixed univariate time series dataset, the first ordinal component of the system's dynamics is included.

Raw data. New time series data can be generated as needed via the `make_trajectory()` method of each dynamical system.

Preprocessing Software. All analysis software is included in the repository.
Motivation. To our knowledge, dataset processing is consistent with the underlying motivation of the dataset.

5. Distribution

Distribution. The dataset is distributed on GitHub.
First Distribution. A private fork may be distributed with the paper for review in order to maintain anonymity for certain venues. The updated repository will be distributed with the final paper.
License. We include an Apache 2.0 License in the project repository.
Fees. None.

6. Legal

People. No individuals are included in this dataset.
Protected Subjects. No ethically-protected subjects are included in this dataset.
Institutional Approval. No institutional approval is required for this dataset
Consent. No individual data is included in this dataset.
Harm. No individual data is included in this dataset. However, the README file of the dataset repository includes instructions to submit an issue if an unintended harm is detected in the process of using this dataset.
Disadvantages. No individual data is included in this dataset.
Privacy. None of the data contains personal information.
GDPR. To our knowledge, this dataset complies with GDPR and equivalent foreign standards.
Sensitivity. To our knowledge, this dataset contains no sensitive or confidential information
Inappropriate. This dataset contains no inappropriate or offensive content.

XV. AUTHOR STATEMENT AND HOSTING PLAN

The authors bear all responsibility in case of rights violations. The data license has been included elsewhere in this appendix. The authors have full control of the data repository on GitHub, and will ensure its continued accessibility.

[1] Rößler, A. Runge–kutta methods for the strong approximation of solutions of stochastic differential equations. *SIAM Journal on Numerical Analysis* **48**, 922–952 (2010).

[2] Guckenheimer, J. & Holmes, P. *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, vol. 42 (Springer Science & Business Media, 2013).

[3] Kuznetsov, Y. A. *Elements of applied bifurcation theory*, vol. 112 (Springer Science & Business Media, 2013).

[4] Strogatz, S. H. *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering* (CRC press, 2018).

[5] Ott, E. *Chaos in Dynamical Systems* (Cambridge University Press, 2002).

[6] Kantz, H. & Schreiber, T. *Nonlinear time series analysis*, vol. 7 (Cambridge university press, 2004).

[7] Wolf, A., Swift, J. B., Swinney, H. L. & Vastano, J. A. Determining lyapunov exponents from a time series. *Physica D: nonlinear phenomena* **16**, 285–317 (1985).

[8] Holzfuss, J. & Parlitz, U. Lyapunov exponents from time series. In *Lyapunov exponents*, 263–270 (Springer, 1991).

[9] Datseris, G. DynamicalSystems.jl: A Julia software library for chaos and nonlinear dynamics. *Journal of Open Source Software* **3**, 598 (2018).

[10] Grassberger, P. & Procaccia, I. Characterization of strange attractors. *Physical review letters* **50**, 346 (1983).

[11] Costa, M., Goldberger, A. L. & Peng, C.-K. Multiscale entropy analysis of complex physiologic time series. *Physical review letters* **89**, 068102 (2002).

[12] Ahmed, M. U. & Mandic, D. P. Multivariate multiscale entropy: A tool for complexity analysis of multichannel data. *Physical Review E* **84**, 061918 (2011).

[13] Christ, M., Braun, N., Neuffer, J. & Kempa-Liehr, A. W. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing* **307**, 72–77 (2018).

[14] McInnes, L., Healy, J. & Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[15] Pedregosa, F. *et al.* Scikit-learn: Machine learning in python. *the Journal of machine Learning research* **12**, 2825–2830 (2011).

[16] Löning, M. *et al.* sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872* (2019).

[17] Alexandrov, A. *et al.* GluonTS: Probabilistic and Neural Time Series Modeling in Python. *J. Mach. Learn. Res.* **21**, 1–6 (2020).

[18] Herzen, J. *et al.* Darts: User-friendly modern machine learning for time series. *arXiv preprint arXiv:2110.03224* (2021). URL https://arxiv.org/abs/2110.03224.

[19] Oreshkin, B. N., Carpov, D., Chapados, N. & Bengio, Y. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437* (2019).

[20] Lea, C., Vidal, R., Reiter, A. & Hager, G. D. Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision*, 47–54 (Springer, 2016).

[21] Godahewa, R., Bergmeir, C., Webb, G. I., Hyndman, R. J. & Montero-Manso, P. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643* (2021).

[22] Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics (Springer, 2009). URL https://books.google.com/books?id=eBSgoAEACAAJ.

[23] Dau, H. A. *et al.* The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica* **6**, 1293–1305 (2019).

[24] Franceschi, J.-Y., Dieuleveut, A. & Jaggi, M. Unsupervised scalable representation learning for multivariate time series. *Advances in Neural Information Processing Systems* **32**, 4650–4661 (2019).

[25] Bai, S., Kolter, J. Z. & Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[26] Gilpin, W. Deep reconstruction of strange attractors from time series. *Advances in Neural Information Processing Systems* **33** (2020).

[27] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] Dempster, A., Petitjean, F. & Webb, G. I. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery* **34**, 1454–1495 (2020).

[29] La Cava, W. *et al.* Contemporary symbolic regression methods and their relative performance. *arXiv preprint arXiv:2107.14351* (2021).

[30] Petersen, B. K. *et al.* Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871* (2019).

[31] Chen, R. T., Rubanova, Y., Bettencourt, J. & Duvenaud, D. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366* (2018).

[32] Gebru, T. *et al.* Datasheets for datasets. *arXiv preprint arXiv:1803.09010* (2018).

[33] Shi, E., Chan, T. H., Rieffel, E., Chow, R. & Song, D. Privacy-preserving aggregation of time-series data. In *Proc. NDSS*, vol. 2, 1–17 (Citeseer, 2011).

[34] Sprott, J. C. *Elegant chaos: algebraically simple chaotic flows* (World Scientific, 2010).

[35] Meier, J. Presentation of attractors with cinema. http://www.3d-meier.de/tut19/Seite1.html (2003). [Online; accessed 19-March-2020].

[36] Myers, A. D., Yesilli, M., Tymochko, S., Khasawneh, F. & Munch, E. Teaspoon: A comprehensive python package for topological signal processing. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond* (2020).