Reservoir Computing for Chaotic Dynamical Systems

Filip Batur Stipic Fatjon Zogaj Patrik Okanovic Rafael Sterzinger
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Chaotic dynamical systems continue to puzzle and amaze practitioners due to their inherent unpredictability, despite their finite and concise representations. In spite of its simplicity, Reservoir Computing [1] has been demonstrated to be well-equipped at the task of predicting the trajectories of chaotic systems where more intricate and computationally intensive Deep Learning methods have failed, but it has so far only been evaluated on a small and selected set of chaotic systems [2]. We build and evaluate the performance of a Reservoir Computing model known as the Echo State Network [1] on a large collection of chaotic systems recently published by Gilpin [3] and show that ESN does in fact beat all but the top approach out of the 16 forecasting baselines reported by the author.**

I. Introduction

Ever since the invention of calculus by Newton and Leibniz, researchers in both science and industry have sought to model many different real-world phenomena as systems whose state evolves through time using differential equations. The field of dynamical systems concerns the analysis, prediction and understanding of the behavior of such systems, encompassing a wide range of phenomena observed in classical mechanics, climate science, and ecology [4].

Chaos Theory is the study of dynamical systems whose key property is exponential sensitivity on initial conditions [5]. This property renders the systems inherently unpredictable over a long enough time horizon, since small differences in the measurement of the initial conditions would cause the predictions of any model to eventually diverge from the true system trajectory [6].

Thus, chaotic systems pose a unique challenge to modern statistical learning methods; recently the work of Gilpin [3] has provided a database of low-dimensional chaotic systems drawn from published work in diverse domains, where each system is paired with pre-computed time series of system trajectories.[1] Moreover, the author has benchmarked a dozen of well-known time series forecasting models such as ARIMA [7], N-BEATS [8] and LSTM [9] on the task of predicting trajectories for different initial conditions, in order to systematically quantify and interpret algorithm performance relative to the mathematical properties of the system. In our work, we challenge these baselines by showing it is possible to obtain competitive performance by using a Recurrent Neural Network (RNN) whose internal weights are randomly fixed and only the output layer of the network is trained, that is with substantially better computational and memory efficiency. This architecture paradigm is known as Reservoir Computing (RC) [1]; we consider a special type of RC known as the Echo State Network, a model in which additionally a linear constraint is imposed on the spectral radius of the model weight matrix, which has been shown in literature to enable great performance on a small, selected set of chaotic time series prediction tasks, but has never before been evaluated at large-scale [10, 11, 12].

[1] https://github.com/williamgilpin/dysts

II. Model

Formally, a reservoir can be thought of as a discrete time non-linear dynamical system, with the state update equation at time $n+1$ given by:

$$x(n+1) = f(Wx(n) + W^{in}u(n+1)) \tag{1}$$

where $x(n)$ is the N-dimensional reservoir state, f is the non-linear activation function, e.g. hyperbolic tangent or sigmoid, W is the $N \times N$ reservoir weight matrix, and W^{in} is the $N \times K$ input weight matrix where K is the dimensionality of the input signal $u(n)$. The output $y(n)$ of the model is then obtained from the extended system state:

$$y(n) = g(W^{out}[x(n); u(n)]), \tag{2}$$

where g is an output activation function and W^{out} is an $L \times (K+N)$ dimensional matrix of output weights, with L being the dimensionality of $y(n)$. Both W^{in} and the internal reservoir matrix W need to be chosen at random whilst ensuring that W satisfies the echo state property (ESP), which informally states that the current network state $x(k)$ is uniquely determined by any left input sequence $u(0), u(1), ..., u(k)$ [13]. To construct such a matrix W, Yildiz et al. [13] provide a simple procedure:

1) Sample W uniformly at random with all non-negative entries
2) Scale W so that the spectral radius $\rho(W)$ is < 1
3) Change the signs of a desired number of entries on W to get negative connection weights as well

Computationally, the most expensive part of this procedure is computing $\rho(W)$, achieved by performing eigendecomposition of W, with complexity of $O(N^3)$. Since the randomly sampled matrix W is not guaranteed to be diagonalizable, in practice it is necessary to re-sample the matrix if eigendecomposition does not converge. Fortunately,

we have found that on average not more than 2-3 repetitions of the sampling procedure are needed to get a diagonalizable matrix. Hence, the complexity of the whole procedure is $O(N^3)$. Usually, after computing the spectral radius, one also re-scales the matrix to set the spectral radius to a predetermined value, which has in literature been found to have a substantial impact on performance of RC and is treated as a hyperparameter [13]. Other important hyperparameters that we tune are the sparsity of W, defined as the percentage of non-zero entries of W, and the reservoir size, defined as $dim(W)$ [14]. Also note that 3) is not a necessary condition; in practice allowing negative weights might or might not improve performance, so we treat satisfying this condition as a Boolean hyperparameter. A network that satisfies the ESP is defined as the Echo State Network (ESN), which has in literature been applied successfully on a variety of supervised learning tasks [15] and hence, in our project we focus on the ESN architecture.

We compare the performance of the ESN with two baseline approaches; namely, we attempt to learn the weight matrices W^{in} and W of the RC via backpropagation through time after random initialization and we train a vanilla RNN with different cell types such as LSTM [16] and GRU [17] from scratch. More experimentally, we also consider variations of randomly connected RNNs with the LSTM and GRU architecture, however note that in this setting the ESP is not guaranteed to hold.

Algorithm 1: Backpropagation Through Time for Learning a Reservoir

Data: update_steps, learning rate α, train data set u
Result: Find matrices W^{in}, W

Standardize train data set u
Create splits u_{tr}, u_{ts}
Initialize matrices W^{in}, W
for $i = 0$ *to update_steps* **do**
  Reset reservoir states x
  for $j = 0$ *to len*$(u) - 1$ **do**
    $x(j+1) \leftarrow f(Wx(j) + W^{in}u(j+1))$
  Create splits for reservoir states x_{tr}, x_{ts}
  Fit linear model using x_{tr}, u_{tr} to obtain W^{out}
  Reset predictions y
  for $j = 0$ *to len*(x_{ts}) **do**
    $y(j) \leftarrow W^{out}x_{ts}(j)$
  Calculate loss $\mathcal{L}$ using y, u_{ts}
  $W^{in} \leftarrow W^{in} - \alpha\nabla_{W^{in}}\mathcal{L}$
  $W \leftarrow W - \alpha\nabla_{W}\mathcal{L}$

Finally, we investigate the performance of an ensemble of multiple RC instances where we approximate the expectation over multiple predictions and also look into restarting the RC model multiple times by re-sampling the reservoir matrices W and W^{in} to find a better initialization.

III. Experiments

A. Dataset

The project by Gilpin has for the first time collected, annotated, and made available the equations driving the dynamics of more than 130 different chaotic systems known in the literature (see Figure 3 and exemplary Figure 5). In addition, the author provides a dataset where each chaotic system is paired with pre-computed train and test trajectories for different initial conditions at both coarse and fine granularity, multivariate and univariate views, and with and without Brownian noise. In our project we focus on the system trajectory forecasting task over all chaotic systems independently, in the setting with univariate timeseries with both coarse and fine granularity (15 and 100 points per period), and without noise.

B. Experimental Setup and Evaluation Metrics

Gilpin reports that the ranking of the benchmarks remains relatively consistent across different canonical regression performance metrics such as Mean Squared Error (MSE), Mean Absolute Scaled Error (MASE), Mean Absolute Error (MAE), and Symmetric Mean Absolute Percentage Error (sMAPE). However, the average rank across all systems on the trajectory forecasting tasks is based solely on the sMAPE score, but in order to have a meaningful comparison with the predictions of the forecasting models reported by the author, we exactly replicate this setup. More specifically, we train and tune the hyperparameters of our models for each system independently, and we average both the sMAPE and the the rank based on the sMAPE score obtained on the test set over all systems.

Moreover, to show our model did not overfit to sMAPE, we will also report MAE. Examining other metrics such as MSE, we note that averaging MSE over all dynamical systems can often be misleading, e.g. N-BEATS performs the best on most systems except on a few where it achieves a much worse MSE score, which when averaged indicate that our model performs better than N-BEATS, which on sMAPE is not the case.

IV. Results

The main results can be found in Table I, where we compare both our proposed models and the baselines across all metrics that we had chosen across all chaotic systems. We first examine the performance of our best model, namely ESN, in Section IV-A. Following that, we examine the performance of our baselines in Section IV-B and Section IV-C and RC with LSTM and GRU in Section IV-D. Finally, we investigate the possibility of choosing a single hyperparameter setting for the ESN across all systems (Section IV-E) and look into the performance of an ensemble of ESNs (Section IV-F).

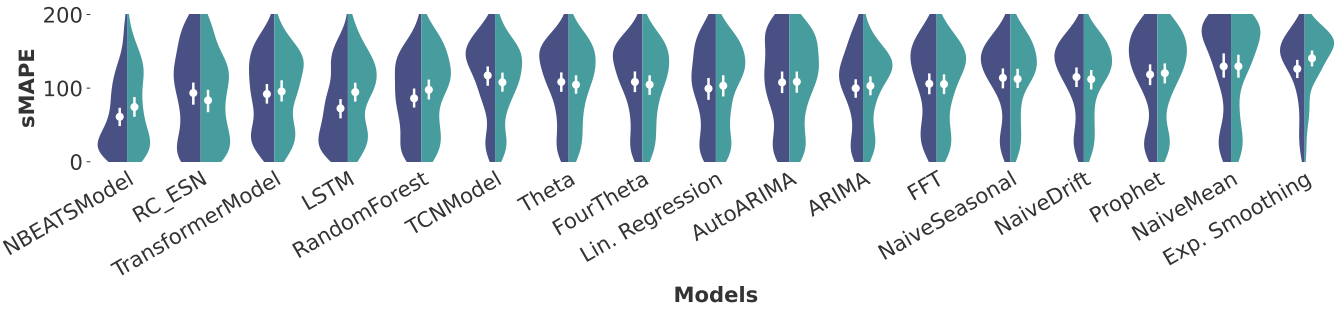


Figure 1: A comparison of the average rank of different models when evaluated on 130 chaotic systems sorted by increasing median error. The hues correspond to coarse (15 points per period, dark) and fine (100 points per period, light) time series granularities. Our proposed model, reservoir computing with an echo state cell, beats most of the existing models benchmarked by Gilpin, with only the N-BEATS model performing better (3.79 versus 5.24 average rank).

Models	15 points per period Rank / sMAPE / MAE	100 points per period Rank / sMAPE / MAE
RC_ESN	6.36 / 93.43 / **0.67**	**5.23 / 83.93 / 1.05**
RC_RNN	9.86 / 127.56 / 6958.24	6.71 / 103.79 / 3.20
RC_LSTM	7.06 / 100.57 / 3.75	6.74 / 98.33 / 2.72
RC_GRU	7.37 / 103.50 / 4.39	6.89 / 99.97 / 4.49
RNN	5.25 / 83.03 / 0.85	8.27 / 111.63 / 1.48
LSTM	**4.31 / 72.97 / 0.67**	6.44 / 94.81 / 1.17
GRU	4.99 / 77.98 / 0.82	8.60 / 114.66 / 1.50

Table I: Results of evaluating average rank, average sMAPE and MAE for our conducted experiments on the dynamical systems with 15 and 100 points per period.

A. *Competitiveness of Reservoir Computing with ESN Cells*

Looking at Table I and the comparison of the results of the existing benchmark depicted in Figure 1, one can observe that forecasting univariate chaotic timeseries with fine granularity using reservoir computing with an echo state cell outperforms all models except the best model reported by Gilpin, N-BEATS. Regarding the ranking between our proposed model and N-BEATS compared to all the others, reservoir computing achieves an average rank of 5.24 whereas N-BEATS is placed slightly better as it obtained an average rank of 3.79. Our approach was able to outperform 15 models some of which take significantly more time to train, one of those being the first ranked N-BEATS. We also want to point the reader to Figure 4 which captures similarities in performance of different models.

B. *Learning a Reservoir via Backpropagation Through Time*

In order to illustrate the performance one could obtain from learning the reservoir weight matrices W and W^{in} versus sampling and leaving them fixed, we propose Algorithm 1. In theory, this algorithm should allow a reservoir to learn a good feature embedding via the reservoir states by means of gradient descent. Here, the key idea is that after each gradient update, the matrices W^{in} and W are able to embed the input in a more meaningful way such that after the linear model has been fitted, the prediction error would decrease. Note however that Algorithm 1 illustrates the learning procedure on a high abstraction level omitting implementation details originally employed by Vlachas et al. During the evaluation, using five updates steps, a learning rate of 0.01, and a reservoir size of 1.000, we noticed that the loss fluctuates a lot, achieving an average rank of 10.05. We hypothesize that this performance drop resulting from our proposed algorithm is mainly due to two reasons: 1) Changing too many weights at once causes oscillatory behavior which leads to instabilities during training; 2) Performing gradient descent on the reservoir breaks the ESP which is crucial for capturing chaotic dynamical behavior. However, enforcing ESP after each gradient update would be infeasible, since it requires performing eigendecomposition of W which is of complexity $O(N^3)$.

C. *Comparison of Different Recurrent Neural Networks versus Reservoir Computing with Echo State Cells*

We report the performance of training an RNN versus training the closely related ESN. RNNs are typically successful in solving problems with sequential input data such as time series [18]. However, they often suffer from the problem of vanishing or exploding gradients [19]. Hence, we additionally examined models that alleviate these problems such as RNN with LSTM or GRU cells. Table I and Figure 7 illustrate the performance of these types of models and allow for a direct comparison to RC. The obtained results support our hypothesis that for dynamical systems training only the last layer via fitting a linear model seems beneficial when compared to the training procedure performed in a standard RNN with different cell types. As LSTM and GRU cells suffer less from vanishing/exploding gradients, these models rank better than the standard RNN. Interestingly, even though RC with an echo state cell outperforms RNN variations as expected in at least the finer 100 points per period granularity prediction task, it yields worse performance than the RNN with an LSTM cell when evaluated on the coarser 15 points per period chaotic time series.

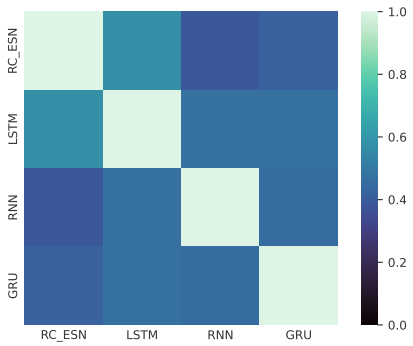


Figure 2: Illustrating the Spearman correlation among reservoir computing with an echo state cell and standard variations of recurrent neural networks comparing the correlation of sMAPE computed across the 130 dynamical systems at fine granularity (100 points per period). In order to better visualize similarities, columns were sorted by descending maximum cross-correlation. As expected reservoir computing with an echo state cell is closest correlated to LSTM given it is the closest to match the performance of reservoir computing.

D. Comparison of Different Cell Types versus Echo State Cells in Reservoir Computing

Looking at the results from Table I and Figure 6, we observe that employing an echo state cell in RC outperforms other cell types in every case. When comparing LSTM, RNN, and GRU cells for RC one can notice that the LSTM cell generally tends to perform better than both the GRU and the RNN cell when it comes to average sMAPE and MAE. Furthermore, we noticed that the GRU cell beats the RNN cell when trained on coarser chaotic time series for which the RNN performs significantly worse. During this evaluation we questioned whether random feature maps produced by one type of cell tend to be better than the others.

Parameter	**Value**
Cell Type	ESN Torch
Reservoir Size	1000
Sparsity	0.01
Radius	0.5
Dynamics Fit Ratio	2/7

Table II: Most occurring parameters for the echo state cell found after performing an extensive hyperparameter search.

E. General Purpose Reservoir for Reservoir Computing

Additionally, we examine whether similar performance to our best approach is achievable if hyperparameters of the reservoir are not tuned specifically for each chaotic time series but rather fixed for all systems, meaning only the output layer is trained per system via linear regression. Therefore, we train another ESN model where all the hyperparameters are fixed to be the most commonly occurring ones found whilst doing grid search over all systems, displayed in Table II. Here our performance slightly drops when compared to our best model, yielding an average rank of 5.28. This highlights another benefit of using RC especially with ESN cells for chaotic systems as tuning the hyperparameters individually obtains only slight improvements.

F. Ensembles of Reservoirs with Echo State Cells

Concurrently to this, we looked into the idea of ensembles, i.e. averaging predictions of multiple RCs. For this, we randomly initialized five RCs with ESN cells which achieved an average rank of 6.97 over three runs. We expected this decrease in performance because intuitively, sampling a reservoir matrix is not an act of performing statistical estimation. Pushing the input through the dynamical system update equation 1 can rather be thought of as a random non-linear projection or one-shot random search through the space of high-dimensional non-linear feature maps. Hence, averaging reservoirs will not have any convergence properties nor performance guarantees. Furthermore, we hypothesized that performing multiple random restarts by re-sampling W, W^{in} and refitting W^{out} on the reservoir output would substantially increase performance, since then the model would not rely on the feature maps derived from one particular sample. Therefore, we sampled 15 reservoirs and chose the one used for prediction on the test set by comparing the scores on the validation set, taken to be the last 200 points of the train set. Surprisingly enough, we find that re-sampling has a negative impact on performance, causing the average rank over all systems to drop from 5.23 to 6.31 in the fine univariate setting. We claim this is because the performance on the validation set may not be indicative of the performance on the test set, due to the properties of the chaotic systems, but more investigation is needed in this direction to corroborate the claim. We suspected that increasing the size of the validation set or that fitting the output layer by using regularization such as ridge regression over the output of the reservoir and the target y would produce improvements, but this was also not the case. Finally, we also note that we were unable to obtain better performance by switching the activation function from tanh to either ReLU or the sigmoid.

V. Discussion

Although we were unable to match the performance of the top approach reported by the author, namely N-BEATS [8], we demonstrate that a substantially simpler and computationally more efficient model can achieve the second best average rank out of 16 well-known timeseries forecasting models in the fine univariate setting. Among the top 3 approaches, our model is faster to train by several orders of magnitude, since the only bottleneck is computing the eigendecomposition of a random matrix and fitting a linear regression model on the output of the reservoir and

the target, both of which have complexity cubic in the size of the reservoir weight matrix N. Moreover, we note that our model can be efficiently parallelized, although we were unable to attain benefit from it in this application area. Deeper investigation is needed into the ESP inductive bias, and its relationship to chaotic dynamical systems, which has, to the best of our knowledge, been unexplored in the literature. Since randomness is crucial for the expressivity of the random feature maps and thus for reservoir computing performance, it would be interesting to see whether one could improve the score by learning the distribution over W, W^{in} from the input signal and the target, rather than sampling the reservoir weights uniformly at random. Future projects could also explore richer RC architectures such as the Deep ESN [20] where multiple distinct reservoirs are sequentially chained so that the output of one reservoir is fed as the input to the next one, and the W^{out} matrix is fitted on the augmented state of all the reservoir states in the chain. Finally, it would be worth investigating whether unrelated timeseries forecasting models can be improved by pre-processing the input via an ESN that is optimized for chaotic dynamical systems.

VI. Summary

We conduct for the first time a large scale evaluation of both the ESN and the RC model with various cell types such as LSTM or GRU on the task of predicting the trajectories of more than 130 chaotic dynamical systems. We train and tune the hyperparameters of our models on each system separately and compare the performance with 16 different timeseries forecasting baselines reported by Gilpin and find that our model achieves the second best ranking on sMAPE score in the fine setting and 4th best in the coarse setting when averaged over all systems. Additionally, we find that fixing the most frequently best found hyperparameter setting on our ESN model over all systems decreases the performance only marginally. Finally, we evaluate and show that Reservoir Computing with the LSTM and GRU cells performs substantially worse than the ESN, which supports the claim that the ESP is crucial for achieving good performance on this task.

References

[1] H. Jaeger, "The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, 01 2001.

[2] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation Algorithms and Reservoir Computing in Recurrent Neural Networks for the Forecasting of Complex Spatiotemporal Dynamics," *arXiv:1910.05266 [physics]*, Feb. 2020, arXiv: 1910.05266. [Online]. Available: http://arxiv.org/abs/1910.05266

[3] W. Gilpin, "Chaos as an interpretable benchmark for forecasting and data-driven modelling," 2021.

[4] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[5] E. Ott, *Chaos in Dynamical Systems*, 2nd ed. Cambridge University Press, 2002.

[6] S. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*, ser. Studies in nonlinearity. Westview, 2000. [Online]. Available: https://books.google.ch/books?id=NZZDnQEACAAJ

[7] G. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[8] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: neural basis expansion analysis for interpretable time series forecasting," *CoRR*, vol. abs/1905.10437, 2019. [Online]. Available: http://arxiv.org/abs/1905.10437

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] P. Vlachas, J. Pathak, B. Hunt, T. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," *Neural Networks*, vol. 126, pp. 191–217, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608020300708

[11] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," *Phys. Rev. Lett.*, vol. 120, p. 024102, Jan 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.120.024102

[12] Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 28, no. 6, p. 061104, Jun 2018. [Online]. Available: http://dx.doi.org/10.1063/1.5039508

[13] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," *Neural Networks*, vol. 35, pp. 1–9, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608012001852

[14] C. Gallicchio, "Sparsity in reservoir computing neural networks," *CoRR*, vol. abs/2006.02957, 2020. [Online]. Available: https://arxiv.org/abs/2006.02957

[15] B. Schrauwen, D. Verstraeten, and J. Campenhout, "An overview of reservoir computing: Theory, applications and implementations," 01 2007, pp. 471–482.

[16] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM)

network," *CoRR*, vol. abs/1808.03314, 2018. [Online]. Available: http://arxiv.org/abs/1808.03314

[17] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[18] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *CoRR*, vol. abs/1909.00590, 2019. [Online]. Available: http://arxiv.org/abs/1909.00590

[19] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," 2013.

[20] C. Gallicchio and A. Micheli, "Deep echo state network (deepesn): A brief survey," *CoRR*, vol. abs/1712.04323, 2017. [Online]. Available: http://arxiv.org/abs/1712.04323

APPENDIX
OVERVIEW ON DIFFERENT CHAOS SYSTEMS

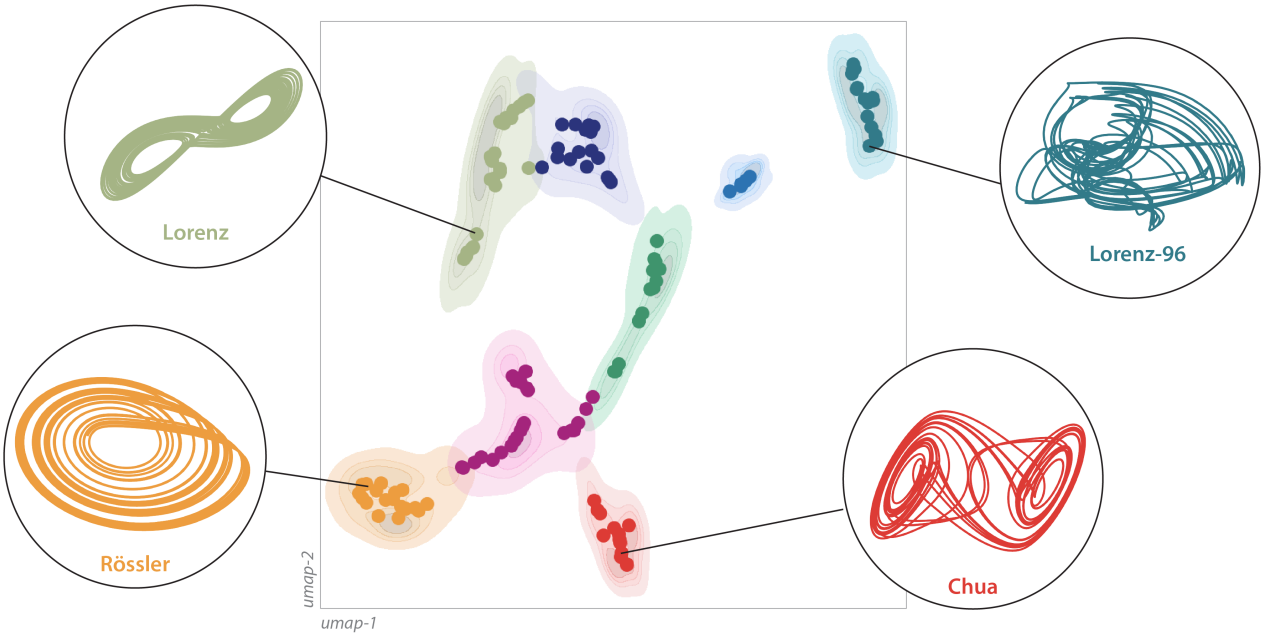
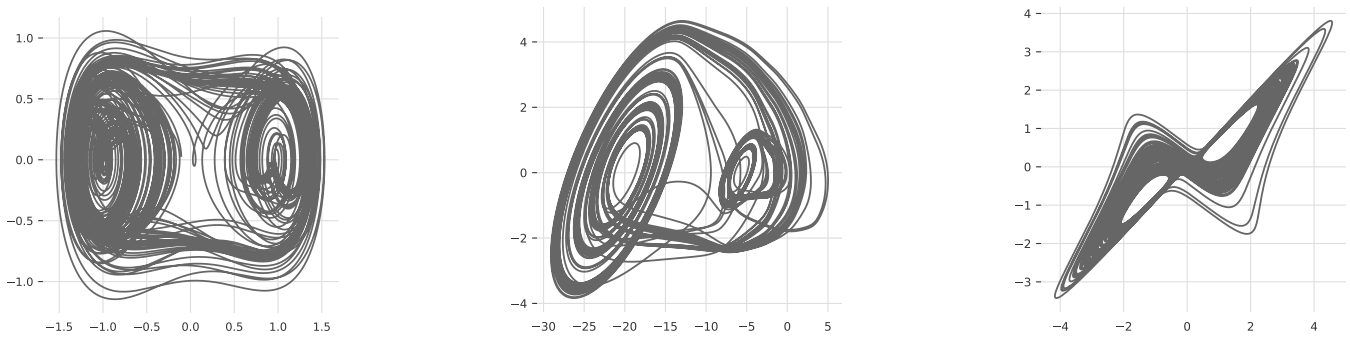


Figure 3: Embeddings of trajectories of chaotic systems clustered in the embedding space as reported by Gilpin.



(a) Duffing (b) MultiChua (c) RikitakeDynamo

Figure 5: An example of phase diagrams of three different chaotic systems with their corresponding trajectories for randomly chosen initial conditions.

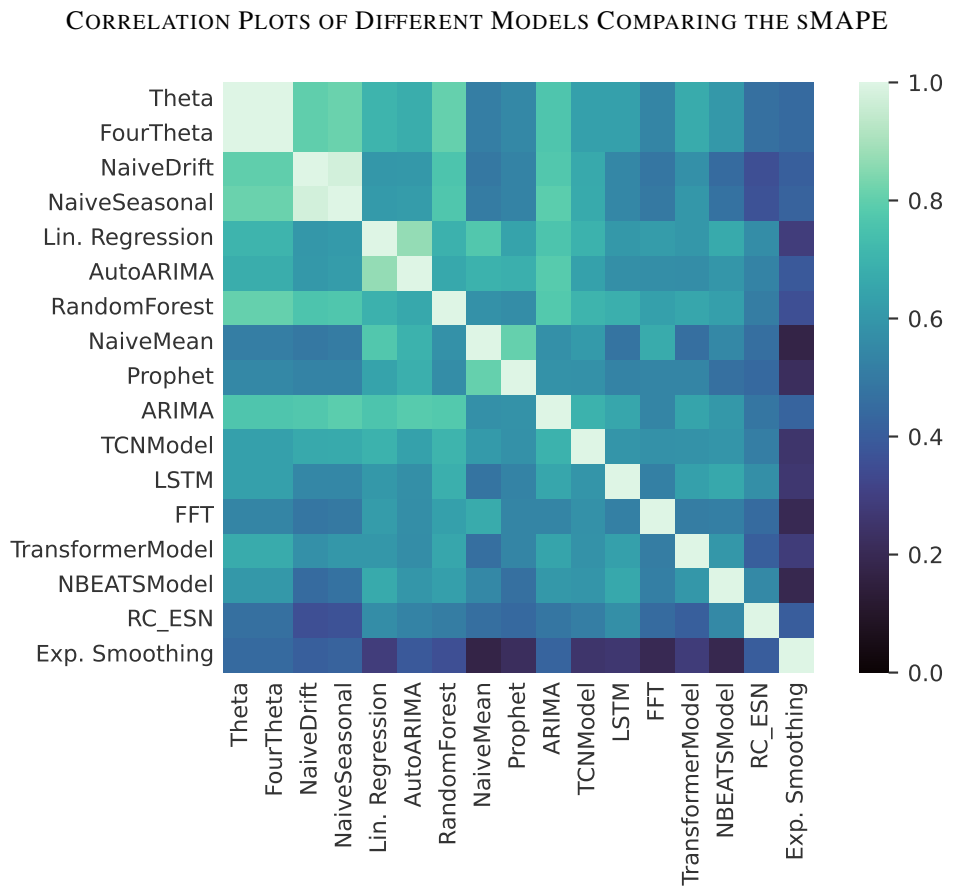


Figure 4: Illustrating the Spearman correlation among different models comparing the correlation of sMAPE computed across the 130 dynamical systems at fine granularity (100 points per period). In order to better visualize similarities, columns were sorted by descending maximum cross-correlation. Visible in this plot is the rather high correlation between reservoir computing with an echo state cell, N-BEATS, LSTM and Linear Regression.

Figure 6: Comparison between reservoir computing with an echo state cell versus reservoir computing with other cells. Models are sorted by increasing median error. The hues correspond to coarse (15 points per period, dark) and fine (100 points per period, light) time series granularities.

Figure 7: Comparison between reservoir computing with an echo state cell versus different types of recurrent neural networks. Models are sorted by increasing median error. The hues correspond to coarse (15 points per period, dark) and fine (100 points per period, light) time series granularities.