Reservoir Computing for Chaotic Dynamical Systems

Filip Batur Stipic
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 (ESN) [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 RNN-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 given by:

$$\tilde{x}(n) = f(Wx(n-1) + W^{in}[1; u(n)]) \tag{1}$$
$$x(n) = (1-\alpha)\tilde{x}(n-1) + \alpha\tilde{x}(n) \tag{2}$$

where $x(n)$ is the N-dimensional reservoir state, f is the non-linear activation function applied element-wise (e.g. hyperbolic tangent or sigmoid), W is the $N \times N$ reservoir weight matrix and W^{in} is the $N \times K + 1$ input weight matrix where K is the dimensionality of the input signal $u(n) \in \mathbb{R}^K$. The term $\alpha \in (0, 1]$ is known as the leaky rate, which can be useful for stabilizing the reservoir dynamics but can also be ignored by setting $\alpha = 1$ and hence $\tilde{x}(n) = x(n)$ [13]. The linear readout layer is defined as:

$$y(n) = W^{out}x'(n), \; x'(n) = [1; u(n); x(n)] \tag{3}$$

where $x'(n)$ is the augmented reservoir state obtained by concatenating the bias, input signal and the reservoir state in a single vector, $y(n) \in \mathbb{R}^L$ is the network output and W^{out} is an $L \times (1+K+N)$ dimensional matrix of output weights. The reservoir map (1) in conjunction with the linear readout layer (3) is common to any reservoir computing architecture, however it can be slightly adapted depending on the nature of the task. For the purposes of our project, the input signal $u(t)$ at time t is the measurement of the dynamical system's trajectory at time t, given some predetermined initial conditions, whereas the target output is the same trajectory shifted one timestep into the future. To learn the output weights W^{out} from the input signal $[u(1), \ldots, u(T)]$, where T is the total length of the input timeseries, we collect the augmented reservoir states as columns of the matrix $X = [x'(1), \dots, x'(T-1)]$ and fit W^{out} using ridge regression, i.e. by minimizing the objective [14]:

$$W^{out} = \underset{W \in \mathbb{R}^{N \times N}}{\operatorname{argmin}} \left(\sum_{k=1}^{T-1} \|W^T x'(k) - u(k+1)\|^2 + \lambda \|W\|^2 \right) \tag{4}$$

where λ is the regularization parameter. Moreover, both W^{in} and the internal reservoir matrix W need to be chosen at random whilst ensuring that the reservoir map in equation (1) satisfies the echo state property (ESP). Formally, a reservoir map $F : \mathbb{R}^N \times \mathbb{R}^d \to \mathbb{R}^N$ satisfies the ESP if given any sequence of inputs $(z_k \in \mathbb{R}^d)_{k \in \mathbb{N}}$ and initial reservoir states $x_0, y_0 \in \mathbb{R}^N$ the sequences:

$$x_{k+1} = F(x_k, z_k) \quad \text{and} \quad y_{k+1} = F(y_k, z_k) \tag{5}$$

satisfy $\|x_{k+1} - y_{k+1}\| \to 0$ as $k \to \infty$ [15]. Intuitively, this means that the dynamics of the ESN should not be asymptotically dependent on the reservoir state initialization, but rather in the limit be driven only by the input signal. To construct a matrix W such that the ESP holds with high probability, Yildiz et al. [15] 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 [15]. 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 $dim(W)$ [16]. 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 reservoir computing model that satisfies the ESP is called the Echo State Network (ESN). In order to predict the target system's trajectory for an arbitrary number of timesteps after fitting the readout map, previous predictions are fed back into the ESN, i.e. the input to the readout map in equation (3) at time n is changed from the input signal $u(n)$ to the model's prediction at time $n-1$, that is $y(n-1)$. The dynamics generated in this way are labelled as the ESN autonomous phase dynamics, which have been successfully used in some dynamical systems trajectory forecasting tasks [17]. Hence, in our project we focus on the ESN architecture.

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. 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). The average rank across all systems on the trajectory forecasting tasks is based solely on the sMAPE score. 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.

IV. Results

The main results can be found in Table I, where we compare the ESN with the top performing baselines by Gilpin [3] across all chosen metrics and all chaotic systems. We first examine the performance of ESN in Section IV-A, following which we elaborate on model selection and investigate the possibility of choosing a single hyperparameter setting for the ESN across all systems (Section IV-B). Finally, we look into the performance of an ensemble of ESNs (Section IV-C).

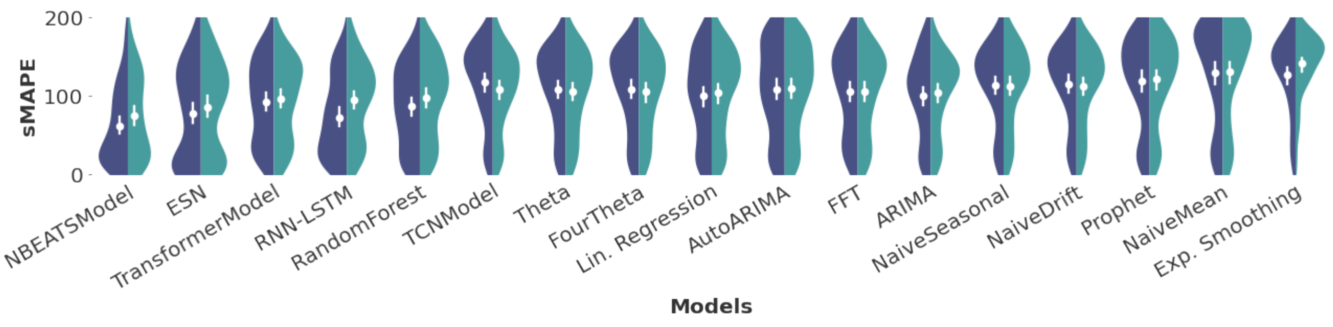


Figure 1: A comparison of the distribution of forecast errors for all dynamical systems and for all forecasting models, sorted by increasing median error. The hues correspond to coarse (15 points per period, dark) and fine (100 points per period, light) timeseries granularities.

Granularity	Model	Rank / sMAPE
Fine	N-BEATS	3.76 / 67.70
	ESN	5.12 / 94.51
	Transformer	6.93 / 101.43
	RNN-LSTM	6.53 / 102.03
Coarse	N-BEATS	3.12 / 49.21
	RNN-LSTM	4.13 / 66.72
	ESN	4.95 / 80.55
	Transformer	7.04 / 93.39

Table I: The top four models in terms of median sMAPE in both the fine and coarse univariate prediction setting. The median sMAPE and average rank reported are computed over all dynamical systems in the dataset.

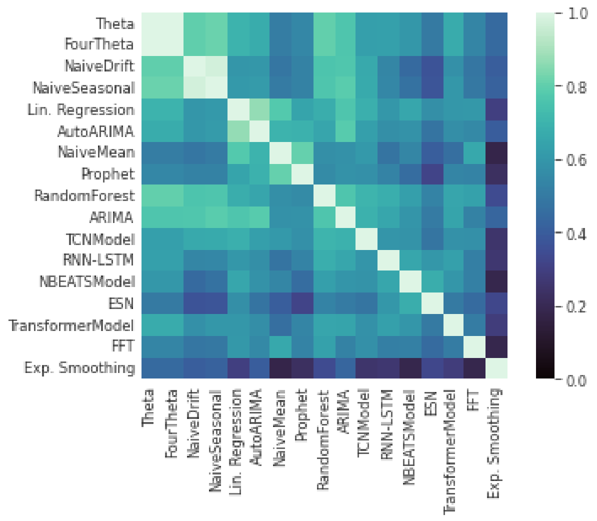


Figure 2: Illustration of the Spearman correlation among all forecasting models, based on the sMAPE metric, computed across all dynamical systems at fine granularity. Columns are sorted by descending maximum cross-correlation in order to group models whose predictions are similar. We can see from the plot that the ESN is highly correlated with other high performing benchmarks reported by the author of the dataset, namely N-BEATS, RNN with LSTM and the Transformer model.

A. *Competitiveness of the ESN*

Looking at Table I and the comparison of the results of the existing benchmark depicted in Figure 1, one can observe that in the task of forecasting univariate chaotic timeseries with fine granularity, our ESN model outperforms all the baselines reported by the author, except the N-BEATS model. More specifically, the ESN model achieves an average rank of 5.12 whereas N-BEATS achieves an average rank of 3.76 in the fine univariate setting. However, we note that our model takes significantly less time to train and is a lot less memory intensive than N-BEATS. We also want to point the reader to Figure 2 which captures similarities in performance of different models.

B. *Model Selection and Hyperparameter Tuning*

Since in the paradigm of Reservoir Computing only the weights in the readout layer are trainable, model selection amounts to performing hyperparameter selection, which is crucial for achieving good performance on any task. In line with the suggestions of Lukoševičius [13], we find that the performance of ESN is most sensitive to changes in spectral radius and the leaky rate. More specifically, we find that the closer the spectral radius is to 1 the better the performance, although there are some systems that pose exceptions to this rule. This supports the common knowledge in the community that ESN performs best at the ”edge of chaos” [18]. We also find that is helpful to set $\alpha \approx 0.5$ as it helps stabilize the learning dynamics, but also in some cases an α close to 1.0 can exhibit strong performance. As for the other hyperparameters, we find that it is important to set the sparsity, reservoir size and regularization to reasonable values, but after doing do so these will not have a very substantial effect on performance. A larger reservoir size gives the ESN more expressivity which can be beneficial if the measurements are coming from a very high-dimensional system, but conversely if the underlying system is low-dimensional it can increase the probability of overfitting, meaning that a higher regularization parameter should be

Hyperparameter	Value
Reservoir Size	1000
Sparsity	0.1
Spectral Radius	0.9
Leaky rate	1.0
Tikhonov Regularization	1e-7

Table II: Empirically and heuristically chosen single best hyperparameter setting

employed for larger reservoir sizes. We only mildly regularize ($\lambda \in [1 \times 10^{-7}, 1 \times 10^{-5}]$) so as not to run in to numerical issues when solving the ridge optimization problem in equation (4). Finally, sparsity of around 0.1 to 0.2 seems to work well in almost all instances.

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 ridge regression. In order not to rely on a specific draw of the reservoir and the input matrices W and W^{in}, we resample the weights before fitting the readout layer for each system independently, as we have done in our original experiment. The single fixed hyperparameters were chosen using a combination of literature recommendation and knowledge of the set of frequently occurring hyperparameters after grid search; these can be found in Table II.

After fixing these hyperparameters and computing the average rank over all systems multiple times, we find that the average performance is very similar to the case where we use best hyperparameters for each setting, however, the variance is much higher, as the average rank can vary from 5.0 all the way up to 5.8. This is contrast to the case where the hyperparameters are chosen for each system independently, in which case the variance in rank is less than 0.1. Furthermore, changing the most sensitive hyperparameters by, for example, lowering the spectral radius, can cause the average rank to drop all the way up to 7, and introduce even more variance in model predictions. 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.

C. Ensembles of Echo State Networks

Since the predictions of the ESN heavily rely on the sample of matrices W and W^{in}, we were interested in whether creating an ensemble of ESNs by resampling both the reservoir and the input matrix could improve average rank or reduce variance. For each chaotic system trajectory, we fix the best hyperparameters found during grid search and we instantiate 10 ESNs, whose weight matrices were drawn i.i.d, and average their predictions. Strangely enough, we observe a slight dip in performance in both the coarse and the fine setting, with the average rank falling to 5.31 and 5.82 respectively. A possible explanation for this is that, looking at Figure 1, we can see that the violin plot for ESN has a very strong bi-modal shape, with 2 modes at both a relatively low and high sMAPE score, meaning that averaging is not likely to increase performance. Moreover, it appears that sampling a reservoir matrix is not an act of performing statistical estimation but rather that the linear random projection via the reservoir map in equation (1) should be thought of as one-shot random search through the space of high-dimensional non-linear feature maps. We are not always guaranteed to find a good feature map, however if we re-sample several times with high probability we are likely to find very useful feature maps for the readout layer to use; this raises the question of whether one could determine whether the sampled reservoir matrix has enough expressivity in order to be useful for the prediction task before fitting the readout layer, which to the best of our knowledge has been unexplored in current literature.

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 and third best in the coarse 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 ridge 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. 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 [19] 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 the ESN 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 based on sMAPE score in the fine setting and third best in the coarse setting when averaged over all systems.

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] M. Lukoševičius, "A practical guide to applying echo state networks," 01 2012.

[14] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, no. 1, pp. 80–86, 2000. [Online]. Available: http://www.jstor.org/stable/1271436

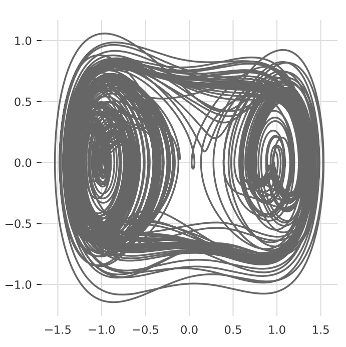
[15] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," *Neural Networks*, vol. 35, pp. 1–9, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608012001852

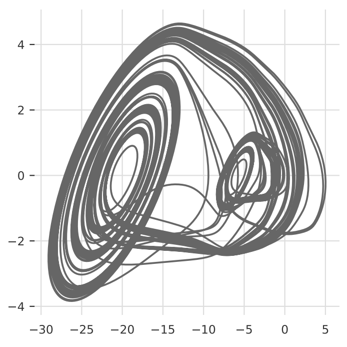
[16] C. Gallicchio, "Sparsity in reservoir computing neural networks," *CoRR*, vol. abs/2006.02957, 2020. [Online]. Available: https://arxiv.org/abs/2006.02957

[17] A. Hart, J. Hook, and J. Dawes, "Embedding and approximation theorems for echo state networks," *Neural Networks*, vol. 128, pp. 234–247, aug 2020. [Online]. Available: https://doi.org/10.1016%2Fj.neunet.2020.05.013

[18] R. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural circuit models," *Neural networks*, vol. 20, no. 3, pp. 323–333, 2007.

[19] 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


(a) Duffing


(b) MultiChua

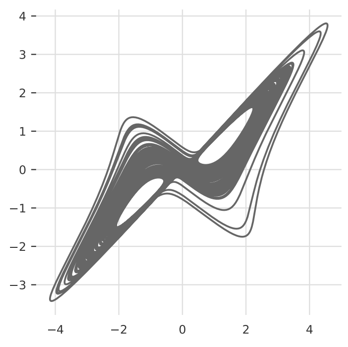

(c) RikitakeDynamo

Figure 3: An example of phase diagrams of three different chaotic systems with their corresponding trajectories for randomly chosen initial conditions.

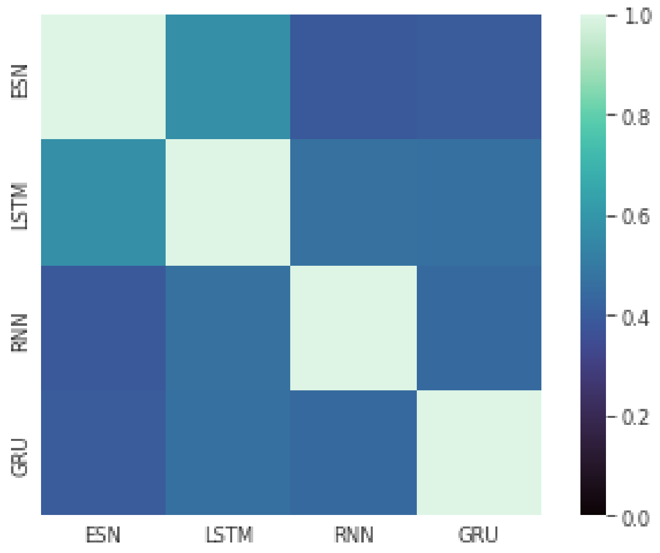


Figure 4: Illustrating the Spearman correlation of ESN and standard variations of recurrent neural networks at fine granularity. In order to better visualize similarities, columns were sorted by descending maximum cross-correlation. As expected, the predictions of ESN are most closely correlated with RNN-LSTM given it most closely matches the performance of ESN in terms of average rank/sMAPE.

Appendix