Exploration of Different Graph Reservoir Structure in Reservoir Computers for Dynamic Systems Forecasting

Abstract

Reservoir computing, a relatively new algorithm, has been observed to be well suited for modeling dynamic systems. However, it has also been observed that reservoir computing does not perform well in the long term and still cannot be compared to deep learning neural networks. Therefore, in this paper we explore 3 different randomness based reservoir networks on the Lorentz system. We then compare our results with ESN, and normal RC and use the Forecast Horizon and time metrics for comparison. While the results show that ESN performs well in all test cases, our models outperform the normal RC in both long (20000 timesteps) and short term (5000 timesteps) forecasting. Nonetheless, there are still many variables left to explore and much room for improvement.

Introduction

Dynamic systems have always been an integral part of our world. Computational models are being explored for their potential of modeling chaotic systems ranging from storm development and wildfire behavior to natural language understanding and stock market prediction. The development of an RNN subfield known as Reservoir Computing (RC) has received attention and is observed to be well suited for handling dynamic system forecasting. Proposed quite recently in 2004 by Jaeger and Haas, RC uses a sparsely connected dynamic network to model the given systems. The reservoir is instantiated with fixed connections and fixed random weights. However, unlike other networks, the reservoir also has memory. By using past inputs, it is able to forecast the future in discrete time steps. However, it is seen that despite its apparent improvement in time complexity, RC falls short in the accuracy obtained by deep learning neural networks (Bianchi et al., 2018). In our algorithm, we explore how the structure of the network and how order of randomness while instantiating the network affects the performance of the model. We explore the potential of complete graphs, power law trees, and random edges, and use the Lorentz system as our test system.

Existing Work

A paper by Xue et al., (2017) explored the effect of circular reservoir topology and leaky integrator neurons in the performance of Echo State Networks (ESN), a type of reservoir network. The results showed that the combined ESN performed the best, demonstrating the importance of the type of activation function and reservoir topology. Another study by Ma et al., (2023) explored the effect of predetermined diagonal block matrix structure of the reservoir and found that the predetermined reservoir structure performed better than the traditional Erdős-Rényi random network. These two studies have inspired our own novel explorations of the predetermined Fully Connected Graphs in forecasting the Lorentz system.

The plasticity of reservoir networks has also been explored. While traditional RC have fixed weight connections, plasticity allows those connections to change and, therefore, add a level of dynamicity to the algorithm. Morales et al., (2021) analyzed how plasticity affects the performance of a RC and found there to be an increase in performance because the number of pairwise correlations decreased. Although we were not able to implement this level of dynamicity, we added another level of randomness as our Random Edge Probability reservoir system.

The application of RC networks in fields from multivariate time series (Bianchi et al., 2018) forecasting to calculating Lyapunov exponents (Pathak et al., 2017) is also being explored. These two papers served as our inspiration and foundation, respectively. Bianchi et al., recognized that MTS classifiers based on normal RC architectures lack the effectiveness and accuracy of fully

trained neural networks, leading us to explore different reservoir systems. Secondly, we based our main RC network on Pathak et al's (2017) implementation.

Dataset

We used the Lorentz 1963 system, a popular test system for studies involved in Reservoir computing or dynamic systems modeling. This system, initially proposed for modeling atmospheric convection, is a perfect example of a dynamic environment. The Lorentz system is governed by the following differential equations:

$$\frac{dx}{dt} = \sigma \cdot (y - x)$$
$$\frac{dy}{dt} = x \cdot (\rho - z) - y$$
$$\frac{dz}{dt} = x \cdot y - \beta \cdot z,$$

By convention σ=10, ρ=28, and β=8/3 [6]. The dataset itself was imported from the `reservoir.py` library [10]. No data preprocessing was required since the data was artificially generated using the Lorentz differential equations. The data was given in time steps with each time step containing the x,y, and z coordinates of the system.

	x	y	z
0	1.000000	1.000000	1.000000
1	1.048843	1.524634	0.973098
2	1.187246	2.089728	0.961757
3	1.410074	2.741063	0.972760
4	1.722680	3.520906	1.017465
...	...	...	...
995	3.836816	7.164134	9.558573
996	4.563417	8.554135	9.702186
997	5.432535	10.175903	10.110336
998	6.464004	12.036800	10.872472
999	7.666668	14.109721	12.108835

1000 rows × 3 columns

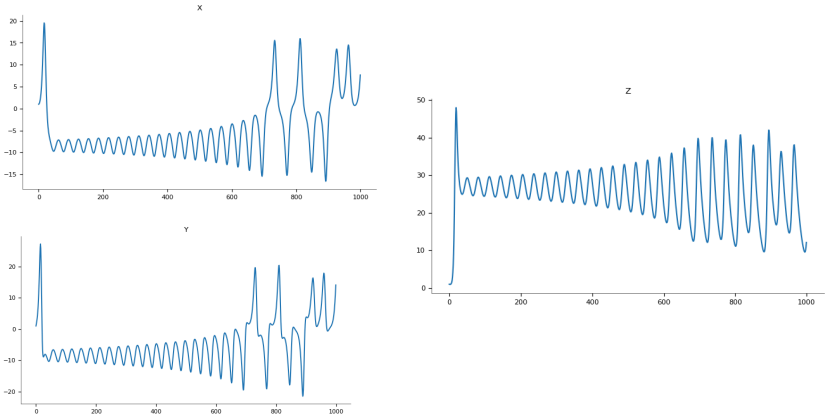


Figure 1: Pandas generated tabular formatted data, time series graph of X, Z, and Y (clockwise from left).

Methodology

Five different models were created and explored: Pathak et al., RC structure, Echo State Network (ESN), Fully Connected Graph RC, Power Law Random Tree graph RC, and Random Edge Connectivity RC.

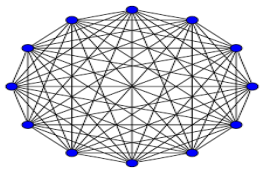
Pathak et al., RC

The RC structure provided by Pathak et al., was used as a foundation for our novelties. Pathak et al., has 3 components to their RC: an Input Reservoir Coupler, Reservoir, and a Reservoir Output Coupler. During the training phase where the model is trained on times -T<t<0, the input is represented by matrix U(t) which is multiplied by the fixed input weights and fed into the

reservoir. The reservoir has two components: the fixed adjacency matrix of the network, A, and current reservoir state, r(t). r(t + dt) is calculated as tanh[A*r(t) + W_{in}*U(t)]. Once the reservoir has evolved into its final r(0) state, the weights of the R/O coupler, W_{out}, is calculated by minimizing the error using the Tikhonov regularized regression procedure shown left (Ma et al., and Morales et al.,). The prediction phase couples the output and the input. The transformed equation for calculating the r(t+dt) = tanh[A*r(t) + W_{in}*W_{out}*r(t-1)]. This is continued until the required time step, and all the predictions are then returned as a matrix.

$$W^{out} = Y^{target} X^T \left(XX^T + \beta I \right)^{-1}$$

Fully connected graph RC



The fully connected graph is a graph of N nodes where each node is connected with all nodes, including itself. This connection was predetermined and thus removed the randomness of the reservoir connections.

Random Power Law Tree RC

The random power law tree uses the power law distribution, where one variable is proportional to a power of the other. In graph theory, a degree sequence is a list of the degrees of each vertex for a graph, or the number of edges which meet at that vertex. To build the tree, a trial degree sequence is randomly generated using a power law distribution. Elements are then swapped with new elements generated using a power law distribution. This process is repeated until the sequence makes a tree, or in other words the number of edges is one smaller than the number of nodes.

Random connectivity RC

For each potential edge in the graph (2 nodes chosen randomly), we generated a distinct, random probability *p*. Based on *p*, an edge was created. This process continued until the graph had (*n* choose 2) * 0.5 edges, where *n* is the number of nodes. We decided to use this as our number of edges, as the Erdős–Rényi model converges to (*n* choose 2) * *u*, where *u* is the fixed probability of the model. We used 0.5 as the average of *u* will converge to 0.5 with an infinite number of random generations.

Echo State Networks (ESNs)

ESNs are a type of recurrent neural network with a sparsely hidden reservoir layer, usually consisting of about 10% connectivity. This model was used to compare our models results

against an industry level model. The main difference between Pathak et al., and ESN proposed by Jaeger and Haas is the state update equation:

$$\mathbf{r}(t+\Delta t) = \tanh[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)] \qquad \mathbf{s}_{t+1} = (1-\gamma)h(\mathbf{W}\mathbf{s}_t + \mathbf{W}_i\mathbf{x}_{t+1} + \mathbf{W}_b\mathbf{y}_t) + \gamma\mathbf{s}_t$$

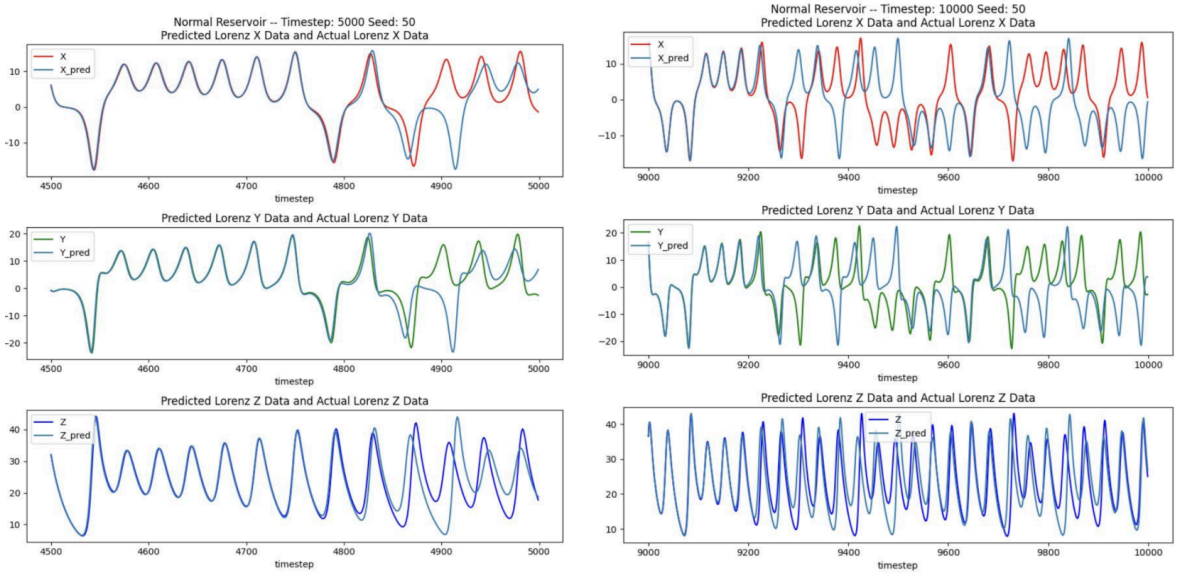
Figure 2: Left: Pathak et al s state update equation; Right: Jaeger and Haas -- s_t is the state of the reservoir units at time t, x_t is the input, $h(\cdot)$is the activation function, y_t is the desired output, W is reservoir weight matrix, W_i is the input weight matrix, W_b is the output feedback weight matrix, and γ is the leak (or retainment) rate.

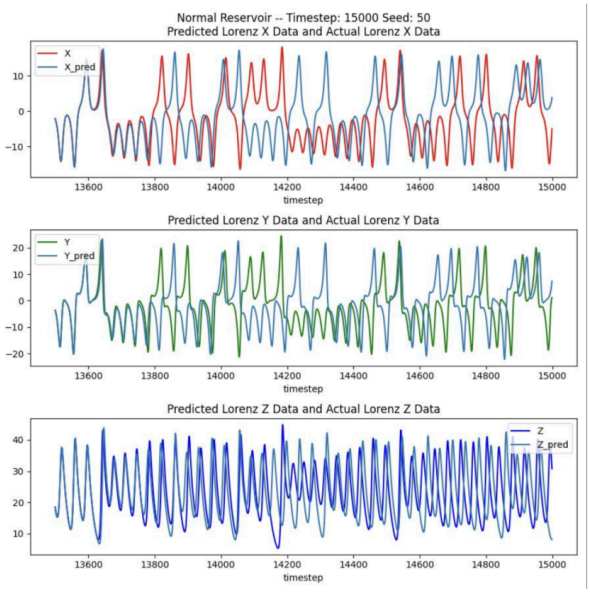
Lorentz forecasting and Accuracy Measurements

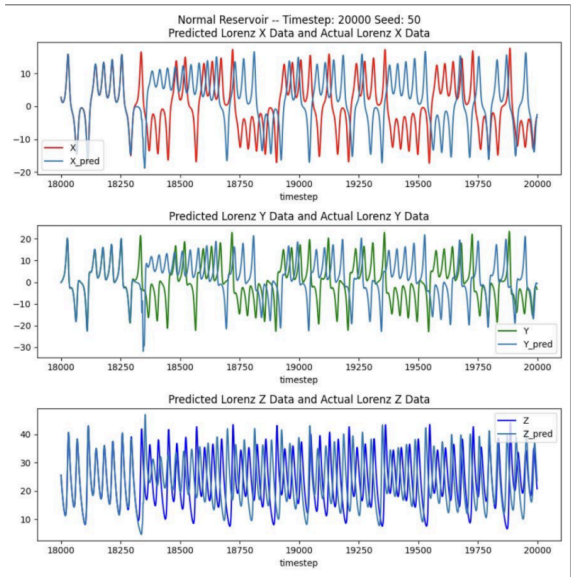
Exactly making predictions can be quite difficult in chaotic modeling; as for this, the general method for calculating accuracy cannot be used. There we used the forecast horizon which counts the number of timesteps in which the difference in between the prediction and testing datapoint is less than the standard deviation of the testing set (Ma et al., 2023).

Results

We created time forecasting graphs such as the ones displayed below for timesteps 5000, 10000, 15000, 20000. For each time step, we ran 3 seeds for each model and calculated the time and forecast horizon. The averages for the 5000 and 2000 are shown below. An extensive list of graphs and metrics can be found in the results folder.







Timesteps	Model	Avg FH X	Avg FH Y	Avg FH Z	Average Time
5000	0	0.960667	0.941333	0.916	0.161938
5000	1	1	1	1	0.682665
5000	2	0.970667	0.950667	0.930667	0.267794
5000	3	0.942667	0.944	0.959333	0.181134
5000	4	0.963333	0.941333	0.930667	0.171952

20000	0	0.8885	0.862667	0.879	0.627953
20000	1	1	1	1	1.18523
20000	2	0.896833	0.881667	0.931333	0.834071
20000	3	0.889167	0.872667	0.922333	0.761023
20000	4	0.927833	0.9165	0.920167	0.674067

Discussion

Firstly, the ESN performed well in all cases, with FH equal to 1. However, it also had the greatest amount of time. It was interesting to see that all other model's performance dropped off in long

term forecasting (timesteps). It was also interesting to see that the normal RC scored the lowest across both time steps. This shows that while our novelty's FH changed depending on the type of forecasting (long or short), they were all better than the normal network (our implementation of the algorithm proposed by Pathak et al). Notably, the Fully Connected graph structure performed best in short term with X,Y, Z FH of 97%, 95%, and 93%, while Random Probabilities performed best in the long term with X,Y, Z FH of 93%, 92%, and 92%. This can signify that predetermined connections fare well for short term prediction while more random or dynamic connections perform well in the long run. This is an area which requires further investigation.

Another interesting observation was that FH only understands how close our prediction is to the true value and not the general or even local trend. In dynamic systems, where trend is a potent signifier of the future state, the inability of FH to include that in its accuracy measure is problematic to say the least. Therefore, a better metric needs to be explored which takes the trend of a state into account.

Future Work

Obtaining a better understanding of our models performance is going to be one of our main goals for the future of this project. By understanding and presenting our results in relation to the test system's Lyapunov time constant like Pathak et al., we will be able to better judge the performance of our models. Other methods such as root mean square error, and mean squared error are also areas of further investigation to obtain a better accuracy score for time series forecasting. Another interesting RC structure would be stacking. This structure was proposed by Moon et al., and was observed to have better performance compared to simply increasing the size of the reservoir. Overall, reservoir computing is a very recent innovation. There are many factors of this algorithm which have not been explored by the scientific community. The intersection of graph theory and randomness and statistics makes reservoir computing a very potent field for exploration.

References

[1] Berner, R. (2023, April 12). *Adaptive dynamical networks*. arXiv.org. https://arxiv.org/abs/2304.05652

[2] Bianchi, F. M. (2018, March 21). *Reservoir computing approaches for representation and classification of multivariate time series*. arXiv.org. https://arxiv.org/abs/1803.07870

[3] Cucchi, M., Abreu, S., Ciccone, G., Brunner, D., & Kleemann, H. (2022). Hands-on reservoir computing: a tutorial for practical implementation. *Neuromorphic Computing and Engineering*, *2*(3), 032002. https://doi.org/10.1088/2634-4386/ac7db7

[4] Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science*, *304*(5667), 78–80. https://doi.org/10.1126/science.1091277

[5] Li, G., Li, B., Yu, X., & Cheng, C. (2015). Echo State Network with Bayesian Regularization for Forecasting Short-Term Power Production of Small Hydropower Plants. *Energies*, *8*(10), 12228–12241. https://doi.org/10.3390/en81012228

[6] Ma, H., Prosperino, D., & Räth, C. (2023). A novel approach to minimal reservoir computing. *Scientific Reports*, *13*(1). https://doi.org/10.1038/s41598-023-39886-w

[7] Moon, J. W., Wu, Y., & Lü, W. (2021). Hierarchical architectures in reservoir computing systems. *Neuromorphic Computing and Engineering*, *1*(1), 014006. https://doi.org/10.1088/2634-4386/ac1b75

[8] Morales, G. B., Mirasso, C. R., & Soriano, M. C. (2021). Unveiling the role of plasticity rules in reservoir computing. *Neurocomputing*, *461*, 705–715. https://doi.org/10.1016/j.neucom.2020.05.127

[9] Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., & Ott, E. (2017). Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *27*(12). https://doi.org/10.1063/1.5010300

[10] Trouvain, N. (2020, September 15). *ReservoirPy: an Efficient and User-Friendly Library to Design Echo State Networks*. https://inria.hal.science/hal-02595026

[11] Xue, F., Li, Q., & Li, X. (2017). The combination of circle topology and leaky integrator neurons remarkably improves the performance of echo state network on time series prediction. *PLOS ONE*, *12*(7), e0181816. https://doi.org/10.1371/journal.pone.0181816