Your Title

Abrari Noor Hasmi
Khalifa University
Abu Dhabi, United Arab Emirates
Email: 100060615@ku.ac.ae

Hadi Susanto
Khalifa University
Abu Dhabi, United Arab Emirates
Email: hadi.susanto@ku.ac.ae

***Abstract*—This is the abstract text.**

I. Introduction

The partial differential equations are usually for modelling physical system. First principle approach to solve this equation involves discretizing the methods using various discretization scheme such as finite difference method, finite element method, finite volume method, spectral methods etc. Recently there have been attempts to solve the partial differential equations using machine learning approaches. Among the reasons to consider the machine learning to solve physical problems includes: 1) ability to incorporate measurement data in the model predictions [1] (2) Speed up computations by modelling computation expensive task and using machine learning approach as closure model to describe missing dynamics [2]. However, application of machine learning still pose many challanges namely the huge training effort, the generalibility to face problem not present in the data and interprebility of the model.

In this research, we want to contribute in investigating the effect of hyperparameters in reservoir computing in solving the Lorenz problem. Reservoir computing is one of the machine learning approaches that used in solving partial differential equations involving time and spatial variables. The reservoir computing could be seen as the recurrent neural network, which evolves a states based on input variables and previous internal states make it suitable for problems which has memory such as physical system making it more efficient in simulating such system compared to general artificial neural network. However, contrary to the recurrent neural network, only the output layer is trained in the reservoir computing, making it easier to train and more stable.

II. Spatio-Temporal Reservoir Computing

Reservoir computing can be taught as a subset of Recurrent Neural Networks. The main principle in reservoir computing is the separation between internal states (reservoir) and the readout learning. As opposed to the backward propagation method which trains both the states and output parameters, the training in reservoir computing only applies to the readout, simplifying the learning while still capable to solve tasks (almost always) better compared with RNN [3], [4]. In this research, we follows the adjustment for spatialtemporal models [1], which split the spatial data into overlapping domains and train multiple reservoirs for each data chunk. The reasoning of splitting the data is that the exchange of information between two very far seperated spatial grid is not immediate, thus creating reservoir state modelling a connection between faraway points is not physical and also a waste of computation effort. Thus by partitioning the grid while maintaining exchange of information between reservoir, render the size of reservoir and the amount of training data becomes manageable.

To be precise let measurement data consist of spatiotemporal data $\mathbf{U} = (u_{ij})$, with temporal index $1 \leq i \leq N_\tau$, and spatial index $1 \leq j < N_\xi$. The periodic spatial grid was splitted into non-overlapping regions $\mathbf{s}^{(k)} = \left(s_{ij}^{(k)}\right)$, with $1 \leq j \leq n_p, 1 \leq k \leq M$ with n_p is the number of nodes at each partition and M is the number of partition. Remark that $M \times n_p = N_\xi$. Associated with each partition $\mathbf{s}^{(k)}$,there is an overlapping domain $\mathbf{v}^{(k)}$ with overlap buffer regions l at each most left and right nodes , thus $\mathbf{v}^{(k)}$ has $n_p + 2l$ spatial grid. Furthermore, the data is temporally split into training and testing set. The overlapping domain serve as input for the each reservoir enabling change of information between neighboring reservoir, while the non-overlapping domain will act as the output.

Following the formalism in [5], we divide the steps in reservoir computing into three steps: (1) listening, (2) training, (3) prediction.

The listening process correspond to constructing reservoir and evolve them based on input and previous states. Specifically let $r_i^{(k)} \in \mathbb{R}^M$ with $M \gg n_p + 2l$ denote the state of reservoir k at time index i. The reservoir dynamics is given by:

$$r_{i+1}^{(k)} = \tanh\left(W^{(k)} r_i^{(k)} + W_{in}^{(k)} v_i^{(k)}\right) \tag{1}$$

The matrix $W^{(k)}$ is an adjecency matrix which usually set to ensure "Echo States Property"[6]. In a simplified term, this condition express that the effect of previous state and previous input at future state should vanish gradually at future time. Practically, even though not always neccessary but the echo state property property is satisfied by scaling $W^{(k)}$, such that $W^{(k)}$ are contractive i.e. the spectral radius of matrix $W^{(k)}$ given by ρ should satisfies $\rho < 1$. Furthermore, the adjecency matrix is set to be sparsely connected and random to ensure that the reservoir is rich enough for learning phase. The input coupling matrix $W_{in}^{(k)}$ is a sparse matrix whose entries magnitude not more than σ. Furthermore, the approximate number of non-zero element at each row is fixed.

The learning phase involve in finding output matrix $W_{out}^{(k)}$ such that the readout defined by:

$$y_i^{(k)} = W_{out}^{(k)} \hat{r}_i^{(k)} \tag{2}$$

minimized a cost function. The modified state $\hat{r}_i^{(k)}$ is a nonlinear modification aimed to futher enhance the nonlinearity feature of the reservoir[4], [1]. Let $r_{ij}^{(k)}$ is the j-th element of vector $r_{ij}^{(k)}$, then the j-th element of vector $\hat{r}_{ij}^{(k)}$ is defined as $\left(r_{ij}^{(k)}\right)^2$ if j is even and $r_{ij}^{(k)}$ otherwise. The cost function is defined as

$$E^{(k)} = \min_{W_{out}^{k}} \sum_{i=1}^{N_{train}} \left\| W_{out}^{(k)} \hat{r}_i^{(k)} - s_i^{(k)} \right\|_2^2 + \beta \left\| W_{out}^{(k)} \right\|^2 \tag{3}$$

In essential, the training can be done by using any method to solve linear least square problem. By defining the $\mathbf{R}^{(k)}$ and $\mathbf{S}^{(k)}$ as a matrix whose i-th column is vector $\hat{r}_i^{(k)}$ and $s_i^{(k)}$ respectively.

During the prediction phase, the input vector $v_i^{(k)}$ in equation 1 is replaced by the recent available prediction from the readout (equation 2) using the output matrix obtained during the training phase. However, as the vector $v_i^{(k)}$ consist of overlapping domain, then the output should be constructed by combining information from the neighbouring reservoir $y_i^{(k-1)}, y_i^{(k)} y_i^{(k+1)}$. Calling this prediction as $\tilde{v}_i^{(k)}$, then the steps in prediction phase are:

1) Compute $y_i^{(k)} = W_{out}^{k} \hat{r}_i^{(k)}$
2) Construct $\tilde{v}_i^{(k)}$ based on $y_i^{(k-1)}, y_i^{(k)} y_i^{(k+1)}$
3) Update the reservoir states: $r_{i+1}^{(k)} = \tanh\left(W^{(k)} r_i^{(k)} + W_{in}^{(k)} \tilde{v}_i^{(k)}\right)$

The implementation of the reservoir computing is based on the code [7] which provide implementations of the reservoir computing including the non-linear transformation.

III. Lorenz 96 Equation

The Lorentz 96 system is a coupled ordinary differential euqation which is used as initial model for atmospheric dynamics or testing a machine learning algorithm for spatiotemporal chaotic system [8], [4]. The model is defined as coupled ordinary differential equations:

$$\frac{dX_j}{dt} = -X_j + X_{j-1}X_{j+1} - X_{j-1}X_{j-2} + F$$

with the spatial variables X_j, $1 \leq j \leq N$,$X_{j+N} = X_j$.

The training data is generated by using 4th order Runge Kutta time integrator with fixed time step $\Delta t = 0.01$ and choosing $F = 8$. The number of spatial grid N is 40 with the initial condition at every grid being a random number between 0 and 1. The largest Lyapunov exponent for this system is $\Lambda = 1.4$. The number of training data is N_{train}= 80000. The hyperparameter used in reservoir computing is given in Table I which are quite similar with the hyperparameter in similar problem used in [8]. The spatially averaged error at time index i is given by: $e_i = \frac{\|\tilde{u}_i - u_i\|}{\|u_i\|}$

Parameter	Explanation	Value
ρ	Spectral Radius of W	0.6
σ	maximum elementwise magnitude of W_{in}	0.1
β	Regularization Parameter	10^{-4}
M	Number of reservoirs	20
l	Number of overlap grid	2
D	Node in reservoir	5000

TABLE I
Hyperparameter for Lorenz 96

Figure 1 shows a comparison between the test data and the prediction by Reservoir Computing. The second panel shows the prediction by reservoir computing, while the third panel shows the difference between the prediction and the "real" test data (shown in first panel). The fourth panel plots the error of the solutions over time. Comparing our result with the result presented in [8](Figure 7) , reveals that our result lies between the realizable RMS error in the computation. A more robust comparison can be achieved by doing predictions starting from different states of attractor, which might be our future work.

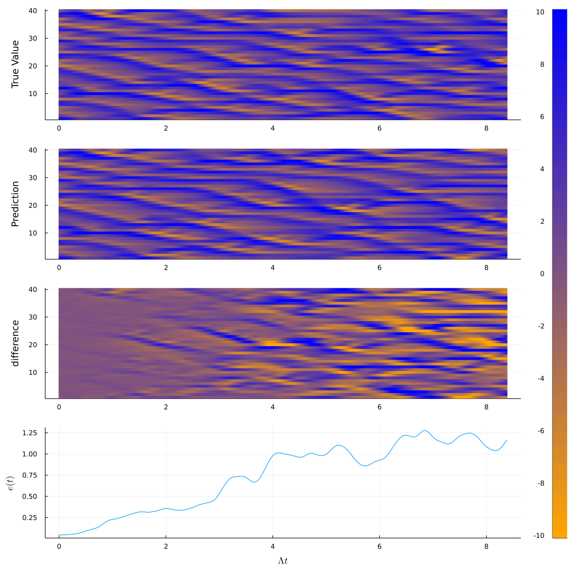

Fig. 1. Prediction of Lorentz 96

IV. Conclusions

The prediction of reservoir computing for solving 5th order breather of NLS with a hyperparameter shows that the (provisional) prediction time is about 0.1 seconds. Further research needs to be done by varying the hyperparameter used in reservoir computing and closely observing the impact of these to the prediction result.

Acknowlegment

The authors wish to acknowledge the contribution of Khalifa University's high-performance computing and research computing facilities to the results of this research.

References

[1] 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,” *Physical Review Letters*, vol. 120, no. 2, p. 24102, 2018. [Online]. Available: https://doi.org/10.1103/PhysRevLett.120.024102

[2] R. Vinuesa and S. L. Brunton, “The Potential of Machine Learning to Enhance Computational Fluid Dynamics,” pp. 1–13, 2021. [Online]. Available: http://arxiv.org/abs/2110.02085

[3] M. Lukoševičius and H. Jaeger, “Reservoir computing approaches to recurrent neural network training,” *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[4] A. Chattopadhyay, P. Hassanzadeh, and D. Subramanian, “Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: Reservoir computing, artificial neural network, and long short-term memory network,” *Nonlinear Processes in Geophysics*, vol. 27, no. 3, pp. 373–389, 2020.

[5] Z. Lu, B. R. Hunt, and E. Ott, “Attractor reconstruction by machine learning,” *Chaos*, vol. 28, no. 6, 2018.

[6] H. Jaeger and H. Haas, “Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication,” *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[7] F. Martinuzzi, “ReservoirComputing.jl,” 2020. [Online]. Available: https://github.com/SciML/ReservoirComputing.jl.git

[8] J. Pathak and E. Ott, “Reservoir Computing for Forecasting Large Spatiotemporal Dynamical Systems,” in *Reservoir Computing*. Springer Nature Singapore, 2021, pp. 117–138.