Preliminary Findings in the Development of DERC: Distributed Edge Reservoir Computation

David Kaauwai[1], Felix Grote[2], Antje Neve[2], Bryan Donyanavard[1]

[1] San Diego State University, [2] Universität der Bundeswehr München

Abstract :

Reservoir computers have several properties that lend themselves to implementation in resource constrained computational situations. They train quickly, are computationally efficient, and take up very little memory real estate. Also, because reservoir computers implemented on von Neumann architecture are a simulacrum of a physical dynamical system, they may be used as an analog to test the ability of new computational substrates before attempting to apply these computers in hardware. Finally, the same randomly initialized model may be tuned for and applied to many similar tasks because any training occurs only in the final layer of the architecture. This report shows the preliminary findings that establish the ability of a randomly initialized reservoir to accurately predict the next step of two different chaotic time series without further hyperparameter tuning.

1 Introduction

Time series computation is an important part of decision making processes in autonomous and self-aware computational systems. The ability of a computational system to accurately infer and predict possible future internal and external states directly informs that system's ability to perform the tasks that it may be designed to do. Reservoir computers are exceptionally lightweight, space efficient, computationally efficient, and quick to train in contrast to transformers and other neural networking models. Another property that has not been explored as of yet is the reservoir model's generalizable accuracy across similar tasks with the ability to separate the static reservoir portion of computation from the trained readout layer and how these properties may be leveraged in edge computation.

Research into computational applications between edge and terminal devices on the network has grown commensurate with the growth of ubiquitous computation and automation. A generalizable model that may be instantiated in resource constrained contexts and can be trained with very few data points is a valuable resource to have at the edge of the network for autonomous system decision-making and general time-series prediction. This initial work begins to evaluate how reservoir computers may fit into computational applications at the edge of the network between edge and terminal devices. Figure 1 shows a diagram of the architecture of the proposed and evaluated system.

2 Methodology

2.1 System Architecture

This project investigates a distributed reservoir computing framework in which the data generation process and training of the readout layers and the reservoir state updates are executed on separate ROS 2 nodes each housed in separate docker containers on a simulated network. As illustrated, the architecture of the system is constructed from two primary components:

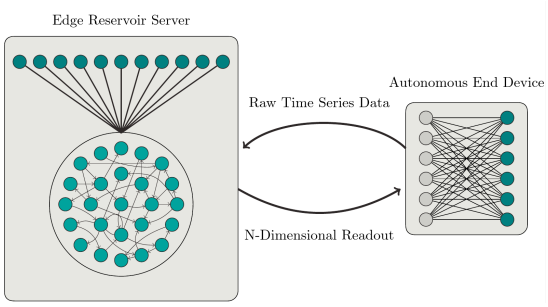

Figure 1. A diagram of a distributed reservoir computer. The edge node instantiates and processes data with the large and comparatively costly reservoir, and the end node only instantiates the linear readout layer.

- **Agent Node** — Generates time-series data by numerically integrating a chosen nonlinear dynamical system. It publishes batches of state values to the network and receives processed reservoir outputs.
- **Edge Node** — Implements the reservoir computer. It receives state data from the agent, performs reservoir state evolution using a fixed recurrent weight matrix, and transmits the updated reservoir state back to the agent.

2.2 Reservoir Configuration

The reservoir implementation used in this project follows the classical Echo State Network formulation. An initial hyperparameter search established that that the mean square error of prediction would not improve with an increased number of parameters in the reservoir while increasing the number of parameter correlated with an increase in round trip time of each message between the reservoir and end device container. Given the results of our hyper-parameter search, the reservoir dimension is set to 1024, the leak rate to 0.1, and a spectral radius to 1.1. Each reservoir is instantiated with the same seed (42) and random number generation algorithm to ensure that all evaluations are instantiated with the same parameter weights. The weight matrix is initialized using a power-law distribution to create a spectrum of weights with a heavy-tailed structure. We choose to square the value of the output of the activation function on at random as the reservoir is iterated forward. This initialization method and activation function modification increase the dynamical richness of the recursive algorithm and is consistent with reservoir designs used in recent literature. Only the readout layer is trained using a linear ridge optimization, consistent with standard reservoir computing practice [1, 2, 3].

Two continuous chaotic system of differential equations, the Röestler and Lorenz- systems, integrated with the fourth-order ac-

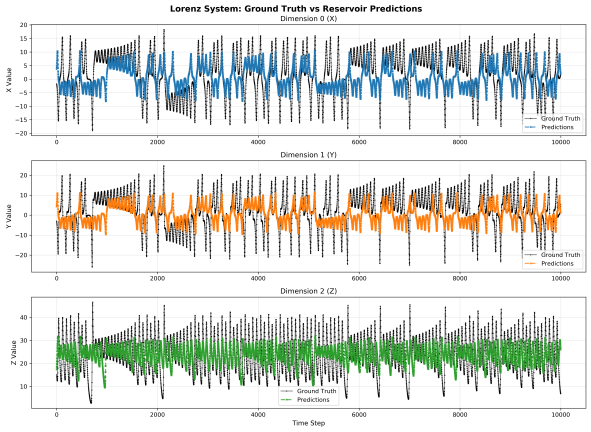

Figure 2. Predictions vs. GST Lorenz-96 computation with RC.

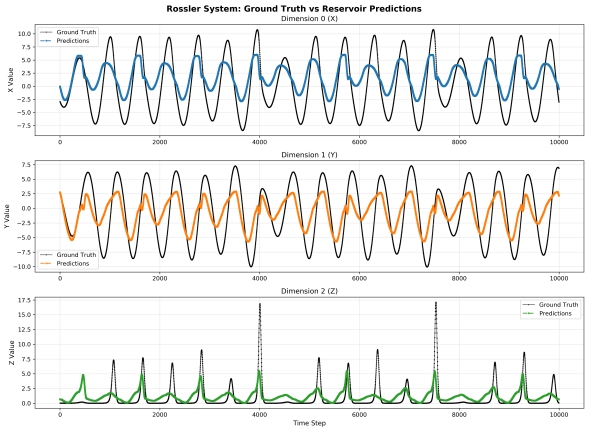

Figure 4. Predictions vs. GST Rossler computation with RC.

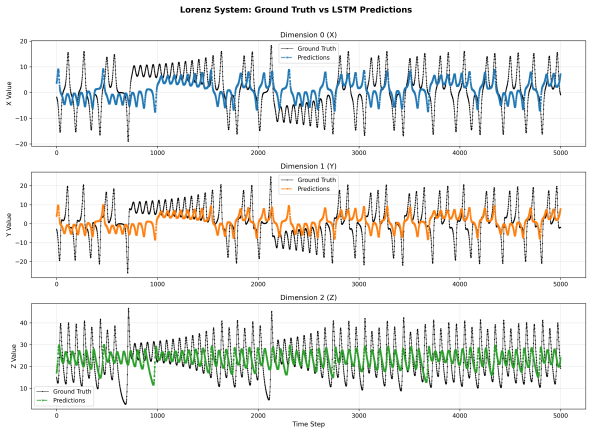

Figure 3. Predictions vs. GST Lorenz-96 computation with LSTM.

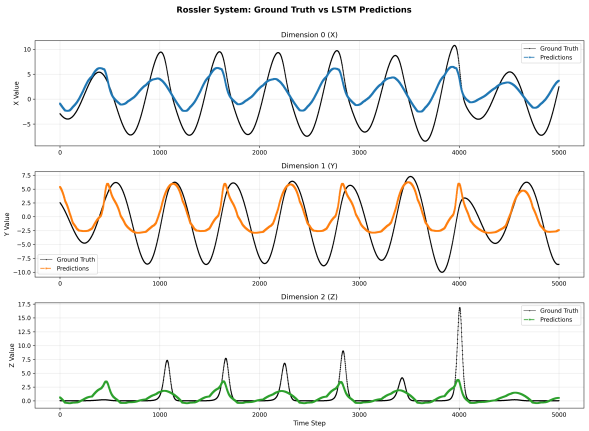

Figure 5. Predictions vs. GST Rossler computation with LSTM.

curate Runge-Kutta 45 integrator simulate a live stream of data. Each system is solved with parameters and initial conditions that are known to generate chaotic, pseudo-random attractors.

3 Results

Figures 2 and 4 plot the results of 10000 time step predictions of the Lorenz and Rössler systems by a reservoir predictor with a readout layer trained for the respective system. Figures 3 and 5 plot the results of 5000 time step predictions of the Lorenz and Rössler systems by a reservoir predictor with a readout layer trained for the respective system. Note that both reservoir predictors have a low RMSE (see Table 1). The reservoir preserves the characteristic chaotic behavior of both systems without exceeding the bounds of either system, and the predictors achieve accuracy on par with state of the art reservoir computer implementations [1, 2].

4 Discussion and Future Work

These preliminary results results introduce the utility and efficacy of distributing the layers of a reservoir computer between an edge and terminal device in a network and confirm the hypothesis that reservoir computer architecture my be distributed over a network across more than one device. Current work is to modify the current code to allow many agents to simultaneously connect to the edge server while running different computational workloads similar to the experiments conducted in this report. This system framework will then be applied to the computation and processing of streaming image data in the context of simultaneous localization and mapping.

References

[1] Andrea Ceni and Claudio Gallicchio. "Edge of stability echo state networks". In: (Aug. 2023). DOI: 10.48550/ARXIV.2308.02902. arXiv: 2308.02902 [cs.LG].

[2] Daniel J. Gauthier et al. "Next generation reservoir computing". In: *Nature Communications* 12.1 (Sept. 2021). ISSN: 2041-1723. DOI: 10.1038/s41467-021-25801-2.

[3] Grigoryeva Lyudmila and Juan-Pablo Ortega. "Echo state networks are universal". In: (June 2018). DOI: 10.48550/ARXIV.1806.00797. arXiv: 1806.00797 [cs.NE].

Table 1
Root mean square error of reservoir computers predicting the trajectory of the Lorenz-96 and Rössler systems by dimension for reservoir and LSTM.

	Reservoir		LSTM	
	Lorenz-96	Rössler	Lorenz-96	Rössler
x-dimension	7.833409	3.852907	8.101246	3.756750
y-dimension	8.854183	3.738567	9.137643	3.365552
z-dimension	8.973881	1.860858	9.173077	1.757927