Reservoir Computing: An investigation of biophysical neuron modeling

Ashley Chen[1], Jennifer Xia[2], Ananya Yammanuru[2], and Nancy M. Amato[2]

[1]University of Minnesota Twin-Cities, Department of Computer Science and Engineering

[2]University of Illinois Urbana-Champaign, Department of Computer Science

Summer 2023

The Mind in Vitro (MiV) investigation's overarching goal is to study the potential for computation using living neurons as an alternative to von Neumann architectures. Our project is an investigation into reservoir computing (RC), a potential architecture which can use a neuron culture as a "reservoir." RC is a machine learning strategy most commonly used for temporal classification and prediction tasks. In our summer research, we explored implementing RC using the biophysical neuron simulator Nengo, while adjusting a variety of parameters, to assess the potential of RC for neurological computation. Results show that while RC isn't the most precise model, it is better at identifying temporal attributes as compared to linear regression.

1 Introduction

The overarching goal of the NSF-funded Mind in Vitro (MiV) investigation is to study the potential for computation using living neurons. This form of computation acts as an alternative to our current digital system based on von Neumann architectures and boolean logic. While digital computing offers high speed and accuracy, shifting our focus to neural substrates opens new possibilities for computational systems capable of cognitive behaviors such as learning, adaptability, and capacity under uncertain conditions [1]. Our project involves reservoir computing (RC), a potential avenue for neurological computation for the MiV investigation.

1.1 Reservoir Computing

RC is a machine learning architecture in which an input is fed into a reservoir of recurrently connected nodes. The recurrent connections between nodes introduces short-term memory capabilities to the reservoir [2] and non-linearly projects the input into a higher-dimensional state which makes it easier to linearly separate the temporal patterns within the input [3]. The higher-dimensional state is known as the readout and is then trained on the target output using some machine learning method, typically linear regression [4]. The reservoir can be any type of network as long as the network exhibits dynamic internal states. RC has even been shown to work with a bucket of water used as the reservoir [5]. However, Recurrent Neural Networks (RNNs) are the most common reservoir choice [4], which is why we use RNNs as our reservoir for this paper. Figure 1 shows the basic schema of the RC architecture.

1.2 Motivation

The RC framework of having a large, fixed RNN with random weights and a training method that is only applied to the readout has been independently discovered across multiple relevant fields including cognitive neuroscience, computational neuroscience, and machine learning [6]. Across these interdisciplinary areas, there has been various works [7][8][9] focused on constructing RC-based frameworks capable of cognitive functions, which align with what we hope to achieve in the MiV investigation. Notably, the MiV NSF proposal mentions that the MiV team was able to create an in vitro chip capable of Morse code communications using RC further supporting the potential of exploring RC as a model for more advanced neurological computations.

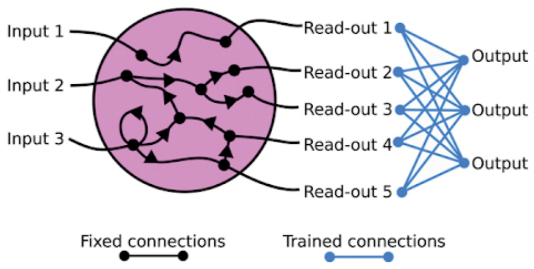

Figure 1: Basic RC Schema, only the connections between the readout and the output layer is trained. The reservoir connections remain fixed [4].

In addition, the unique characteristics of the RC architecture gives it advantages. RNNs and its different variants require gradient descent due to having to train all the recurrent connections [2]. In comparison, RC only requires training on the readout. RC doesn't have to worry about a common issue with gradient descent, exploding/vanishing gradients, which means faster training and more reliable convergence. Finally, RC has advantages over non-recurrent machine learning models. Projection and short-term memory capabilities induced by the recurrent connections within the reservoir is why RC is especially useful for temporal classification and prediction tasks [4].

2 Methods

2.1 Data

For our testing, we decided to use temporal prediction because RC is especially suited for these types of tasks as explained in the previous section. To avoid complications such as insufficient, noisy, or inconsistent data, we generated our own data points. The data points were based on a sine wave model, described by the equation: 50 * sin((pi / 2) * x) + 0.4 * x, and was composed of 50 periods with 20 data points per

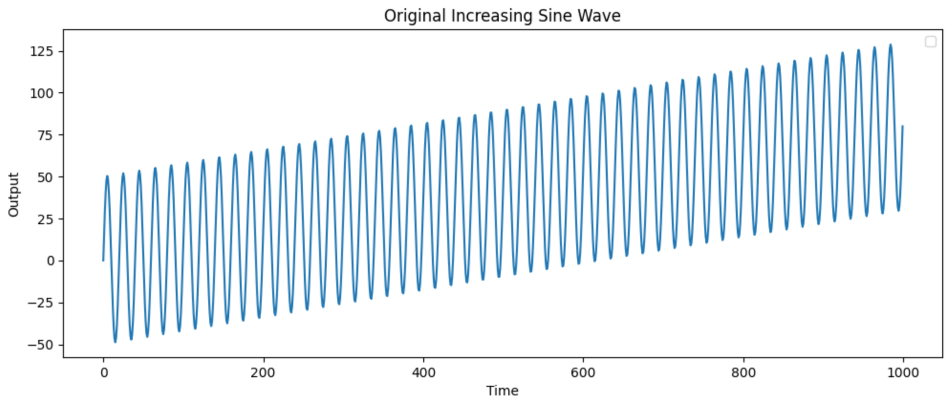

Figure 2: Input signal that we used train and test the reservoir.

periodic cycle. A graph of our sine wave can be seen in Figure 2. Using these data points, we set the features to the original data points and the target to the data points shifted by one position.

2.2 Nengo RC Model

Nengo [10] is a commonly used Python library for simulating biophysical neurons and the tool that we used to construct our reservoir. We set the neuron type of our Nengo reservoir to Leaky Integrate-and-Fire (LIF) neurons, a popularly used, simplified computational model that mimics the firing patterns of real neurons [11]. We also initialized the reservoir weights randomly. Our reservoir was greatly inspired by this tutorial [12]. We use standard linear regression to train the readout from the Nengo reservoir and measure the mean squared error (MSE).

2.3 Reservoir Parameters

We tested two key [13] reservoir parameters: reservoir size and spectral radius. Reservoir size indicates the number of neurons within the reservoir. For our experiments, we set our reservoir size to 100, 200, 300, 400, 500, and 1000 neurons (n).

The spectral radius is the largest absolute eigenvalue of the reservoir weight matrix. Spectral radius values close to 1 are best for tasks that require long memory. Larger spectral radius values cause the reservoir to be more chaotic and introduce more non-linearity. Smaller spectral radius values cause the reservoir to resist change and is beneficial when long memory may be harmful [14][15]. We originally set the spectral radius to one and then tuned the spectral radius value (sr) from there.

2.4 Baseline

We also wanted some standard to test the effectiveness of our Nengo reservoir. Instead of feeding the input through a reservoir and then training the readout with linear regression like we do for our Nengo RC model, we decided to just directly train the input using linear regression. This direct training is our baseline to compare against our Nengo RC model. We performed comparisons both qualitatively with observation and quantitatively with MSE.

3 Results

Starting with a reservoir of size n = 100 and sr = 1, the results were qualitatively and quantitatively quite poor in comparison to baseline. While increasing the reservoir size did improve the shape of the prediction and lowered the MSE, overall the RC Nengo model was still significantly underperforming. Then, we started to tune the spectral radius. While the spectral radius was very volatile, we were eventually able to tune it with positive results. With a reservoir of size n = 1000 and sr = 0.36, we managed to greatly improve the training MSE, however, the overfitting started to occur on the testing data. To combat this overfitting, we added a regularization parameter to our linear regression model to decrease the influence of individual features from the readout. A reservoir size of n = 1000, sr = 0.36, and regularization parameter (alpha) of 50,000 gave the best results so far as can be seen in figure 3.

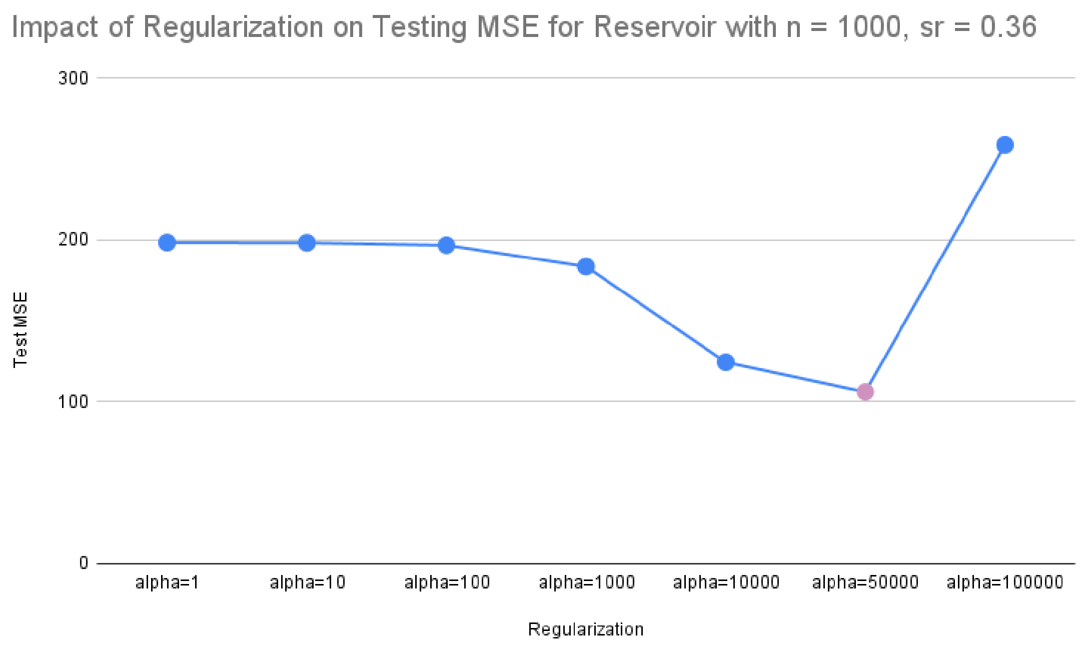

Figure 3: The impact of regularization on preventing overfitting after tuning the reservoir parameters.

We then compared the results of our best tuned Nengo RC model to baseline. While the MSE between the two was relatively similar, our Nengo RC model did a better job of learning the temporal parts of the increasing sine wave. Baseline was unable to learn the shift in the target which is why its prediction is consistently offset as seen on the left side of figure 4. On the other hand, while our Nengo RC model wasn't able to as accurately learn the shape of the target as baseline, our Nengo RC model was able to learn the shift as seen on the right side of figure 4.

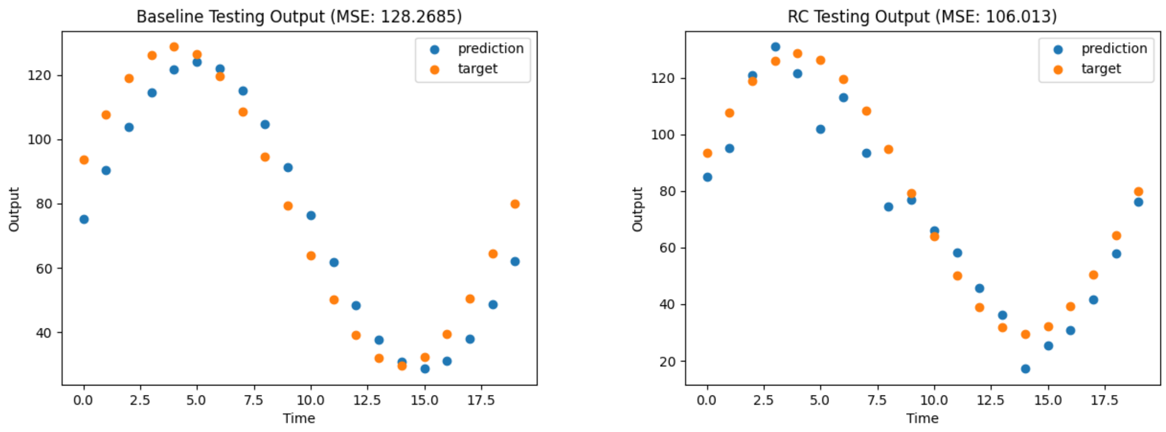

Figure 4: Comparing the results of baseline linear regression (left) with our tuned Nengo RC model (right).

4 Discussion

This paper describes some of our initial work and results for our investigation of RC for computation with living neurons. Tuning both the reservoir size and the spectral radius greatly improved the results of our RC model, but caused overfitting. We were able to counteract the overfitting using regularization, and our final tuned Nengo RC model yielded positive results. Overall, our Nengo RC model was able to better learn the temporal components of our input data than baseline. There is still a lot of work to be done in exploring RC and improving our Nengo RC model, however, these results demonstrate the potential of RC for neurological computation.

In the future, we will continue to adjust the Nengo RC model. Currently, there are some irregularities. For instance, the spectral radius is extremely volatile even around one which is inconsistent with the literature [14][15][16]. Therefore, these irregularities need to be investigated. We plan to start by testing simple polynomial functions rather than a sine curve to simplify our data as much as possible. We will also continue to investigate different recommended [14] reservoir parameters including reservoir weight sparsity and distribution, along with the leaking rate of the neurons. Then, we plan to expand our study by implementing different reservoir architectures [17]. So far, we have mainly used linear regression as the readout training method, performed minimal preprocessing of the input, and fed the input through one reservoir. These are all architecture factors that we plan to adapt and test in the future. Finally, we eventually want to start comparing the results of our Nengo RC model to the MiV team's main simulator [18]. From there, we can start doing our own personalized tests with the MiV Simulator.

5 Acknowledgements

I worked on this project with my fellow intern Jennifer Xia and graduate mentor Ananya Yammanuru who were both tremendously helpful and amazing to work with. I would like to thank Prof. Nancy M. Amato for being a great mentor and for giving me this opportunity. Additional thanks to Associate Prof. Marco Morales Aguirre, Courtney McBeth, Rachel Ann Moan, Seung Hyun Kim, Frithjof Gressmann, the Parasol Lab, UIUC CS-SRP, and CRA-WP for their assistance this summer.

References

[1] I. Schuller and R. Stevens, "Neuromorphic Computing: From Materials to Systems Architecture," Report of a Roundtable Convened to Consider Neuromorphic Computing Basic Research Needs, 2015. 1

[2] Jaeger, Herbert. "Echo State Network." Scholarpedia, 2007, www.scholarpedia.org/article/Echo_state _network. 1, 2

[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, doi:10.1016/j.cosrev.2009.03.005. 1

[4] Matteo Cucchi et al. "Hands-on Reservoir Computing: A Tutorial for Practical Implementation." IOPScience, 2022, iopscience.iop.org/article/10.1088/2634-4386/ac7db7/meta. 2

[5] Fernando, Chrisantha, and Sampsa Sojakka. "Pattern Recognition in a Bucket." SpringerLink, 2003, link.springer.com/chapter/10.1007/978-3-540-39432-7 _63. 2

[6] Matteo Cucchi et al. "Hands-on Reservoir Computing: A Tutorial for Practical Implementation." IOPScience, 2022, journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004967. 2

[7] Pascanu, Razvan, and Herbert Jaeger. "A Neurodynamical Model for Working Memory." Neural Networks, vol. 24, no. 2, 2011, pp. 199–207, doi:10.1016/j.neunet.2010.10.003. 2

[8] Hinaut, Xavier, and Peter Ford Dominey. "Real-Time Parallel Processing of Grammatical Structure in the Fronto-Striatal System: A Recurrent Network Simulation Study Using Reservoir Computing." PLOS ONE, 2013, journals.plos.org/plosone/article?id=10.1371/journal.pone.0052946. 2

[9] Hinaut, Xavier, et al. "Corticostriatal Response Selection in Sentence Production: Insights from Neural Network Simulation with Reservoir Computing." Brain and Language, vol. 150, 2015, pp. 54–68, doi:10.1016/j.bandl.2015.08.002. 2

[10] https://www.nengo.ai/ 3

[11] Lu, Sijia, and Feng Xu. "Linear Leaky-Integrate-and-Fire Neuron Model Based Spiking Neural Networks and Its Mapping Relationship to Deep Neural Networks." Frontiers, 27 July 2022, frontiersin.org/articles/10.3389/fnins.2022.857513/full. 3

[12] https://arvoelke.github.io/nengolib-docs/master/notebooks/examples/full_force_learning.htm 3

[13] Lukoševičius, Mantas. "A Practical Guide to Applying Echo State Networks." SpringerLink, 2012, link.springer.com/chapter/10.1007/978-3-642-35289-8_36. 3

[14] Herbert, Jaeger. "The "Echo State" approach to Analysing and Training Recurrent Neural Networks." German National Research Institute for Computer Science, 2001, www.bibsonomy.org/bibtex/23d434b04cf1479acf45be9af65f8bc78/idsia. 3, 5

[15] Verstraeten, David, et al. "Memory versus Non-Linearity in Reservoirs." The 2010 International Joint Conference on Neural Networks (IJCNN), 2010, doi:10.1109/ijcnn.2010.5596492. 3, 5

[16] Morales, Guillermo B., and Miguel A. Muñoz. "Optimal Input Representation in Neural Systems at the Edge of Chaos." arXiv.Org, 12 July 2021, arxiv.org/abs/2107.05709. 5

[17] Sun, Chenxi, et al. "A Review of Designs and Applications of Echo State Networks." arXiv.Org, 5 Dec. 2020, arxiv.org/abs/2012.02974. 5

[18] https://github.com/GazzolaLab/MiV-Simulator 5