Distance-based Delays in Echo State Networks*

Stefan Iacob[0000−0002−3424−0390], Matthias Freiberger[0000−0003−2101−6274], and Joni Dambre[0000−0002−9373−1210]

IDLab-AIRO, Ghent University-imec, 9052 Ghent, Belgium
{stefanteodor.iacob,matthias.freiberger,joni.dambre}@ugent.be

Abstract. Physical reservoir computing, a paradigm bearing the promise of energy-efficient high-performance computing, has raised much attention in recent years. We argue though, that the effect of signal propagation delay on reservoir task performance, one of the most central aspects of physical reservoirs, is still insufficiently understood in a more general learning context. Such physically imposed delay has been found to play a crucial role in some specific physical realizations, such as integrated photonic reservoirs. While delays at the readout layer and input of Echo State Networks (ESNs) have been successfully exploited before to improve performance, to our knowledge this feature has not been studied in a more general setting. We introduce inter-node delays, based on physical distances, into ESNs as model systems for physical reservoir computing. We propose a novel ESN design that includes variable signal delays along the connections between neurons, comparable to varying axon lengths in biological neural networks or varying length delay lines in physical systems. We study the impact of the resulting variable inter-node delays in this setup in comparison with conventional ESNs and find that incorporating variable delays significantly improves reservoir performance on the NARMA-10 benchmark task.

Keywords: Echo state networks · Bio-inspired computing · Evolutionary algorithms · Variable delays.

1 Introduction

Echo State Networks (ESNs) [12] offer a promising low-energy alternative for error backpropagation that has gained attention in recent years. Multiple layers of reservoirs can be combined in deep ESNs and have proven successful at several practical applications [7, 18, 6]. Due to ESNs essentially being recurrent neural networks (RNNs) with fixed-weight input and hidden layer, they can be approximated in a time-continuous way by many physical systems [20] which is commonly referred to as physical reservoir computing. In physical reservoirs, delays between nodes tend to vary [5] due to, amongst other reasons, design constraints and imperfections in the manufacturing process. We argue that these

* This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 860949.

variations in inter-node delays, which are often imposed by the spatial layout of the reservoir, i.e. the physical distance of reservoir nodes between each other, are not merely to be tolerated as they are intrinsic to the underlying physical processes, but can be embraced as beneficial to reservoir performance.

Similarly, despite artificial neural networks being in general far removed from biological systems, one can also draw inspiration from a simple and often overlooked aspect of biological networks: neurons have a spatial location, which means that there is physical distance between neurons. Animal axon lengths vary in the order of millimeters, and signal delays can vary in terms of milliseconds [3]. This variation in delay has been shown to have a qualitative effect on plasticity [19]. In contrast, rate-based ANNs and ESNs in particular have no such feature, meaning that all inputs to a neuron are processed simultaneously, disregarding possible variation in timing.

ESNs and physical reservoirs bear promise as a low-energy solution for time-series tasks. Therefore, the understanding of temporal processing in reservoirs is essential for improving task performance. The use of strong non-linearities in information-processing dynamical systems decreases the memory capacity of the system, whereas less non-linearity limits computational power. This observation is commonly referred to as the memory-nonlinearity-tradeoff [4]. We argue that the use of variable-length connection delays allows for a simple way to introduce linear memory at various timescales in the reservoir, mitigating the issue of too rapidily fading memory in highly nonlinear systems. In more concrete terms, delay lines of different lengths meeting at a single node allow that node to combine information from two different time points without any decay in memory.

The novel contribution presented in this work is a systematic exploration of the impact of varying propagation delays on task performance in a substrate-agnostic setting. We use inter-node distance-dependent delays in ESNs, with nodes modeled as points in physical space, and distances computed as a euclidean norm of the difference of their coordinates. This begs the question of how to optimize their spatial locations. We present a novel, spatially represented ESN implementation which we refer to as distance-based delay network (DDN), where neuron locations are sampled from a Gaussian mixture distribution. We optimize the spatial structure of this network by tuning the parameters of the location distribution with a genetic algorithm. We show a strong and significant improvement in performance on the NARMA-10 benchmark task compared to conventional ESNs. Our baseline performance is in accordance with other conventional ESN implementations, whereas our best models perform better than some deep ESNs. However, showing an absolute improvement of the state of the art for time series prediction is beyond the scope of this project. We simply show in a well controlled experimental setting that variations in connection delay are exploitable by optimizing neuron location distributions in a physical space, all else being equal.

Although the use of time delays in rate-based neural networks is limited, timing is inherently considered to some extent in spiking neural network implementations due to the temporal nature of spike encoding. Jeanson et al. explored

the use of simulating axonal delays for robot control [16]. However, the added computational complexity of SNNs is an added challenge for practical applications. Therefore we study the effect of delays in a more general sense. The use of delay lines in ESNs has been previously explored in for both readout connections [11] and input connections [15]. The former work introduced learnable readout delays, which was shown to improve network performance. The latter work showed an improvement in performance of untuned reservoirs by using input delays To our knowledge, no previous work explores the use of inter-neuron signal delays in a rate-based echo state network by freely varying neuron positions.

2 Methods

In this section we describe our approach to the implementation of an ESN that includes varying signal delays in its connections (both input and recurrent). This is opposed to conventional ESNs, where the network activation at time t is instantaneously processed to produce the recurrent input at time $t+1$. These conventional ESNs can be represented by the following equation [13]

$$\mathbf{x}(n+1) = (1-a)\mathbf{x}(n) + a \cdot f(\mathbf{W}_{\text{res}}\mathbf{x}(n) + \mathbf{b}_{\text{res}} + \mathbf{W}_{\text{in}}\mathbf{v}(n)) \tag{1}$$

where $\mathbf{x}(n)$, a, $\mathbf{W}_{res}$, $\mathbf{W}_{in}$, $\mathbf{b}$, f, and $\mathbf{v}(n)$ refer to the network state at timestep n, the decay rate, the (recurrent) reservoir weight matrix, the input weight matrix, the bias weights of the reservoir, the activation function, and the input at timestep n respectively. This equation needs to be augmented for the proposed distance based delay networks (DDNs), as we do not directly take the current network state $\mathbf{x}$ as input for our activation function, but rather the sum of incoming delayed input signals to each neuron.

With DDNs, we propose an ESN whose delays are dependent on neuron locations. Any signal travelling between two neurons is delayed by an amount of time steps proportional to the physical distance between them. Given a set of 2D neuron coordinates, we can define a N by N Euclidean distance matrix $\mathbf{D}$, with N indicating the number of neurons and element $D_{i,j}$ corresponding to the distance between neuron i and neuron j. Since the simulation approach that we use requires a finite number of distance values, the elements of $\mathbf{D}$ are discretized over the set of $\{1, 2, .., D_{\max}\}$, where $D_{\max}$ corresponds to the maximum number of possible delay steps. This means that the longest connection implements a delay of D_{max} simulation steps, and the shortest connections applied only one simulation step delay. We refer to $\mathbf{D}$ as a delay matrix. In order to extend $\mathbf{W}_{\text{res}}$ from equation 1 to incorporate varying delays, instead of a single weight matrix, we define a set of "masked" weight matrices $\mathbf{W}_{D=d}$ where for each element

$$W_{i,j,D=d} = \delta_{d,D_{i,j}} \cdot W_{i,j} \tag{2}$$

where $d \in [1, D_{\max}]$ and δ is the Kronecker delta operator. As such, the weights in $\mathbf{W}_{D=d}$ corresponding to connections with a delay different from d are set to

0. Using the same notational conventions as in equation 1, we can formalize the state update mechanism of a DDN as

$$\mathbf{x}(n+1) = (1-a)\mathbf{x}(n) + a\mathbf{y}(n) \tag{3}$$

$$\mathbf{y}(n) = f\left(\sum_{d=0}^{D_{\max}} \left(\mathbf{W}^{\text{res}}_{D=d}\mathbf{x}(n-d) + \mathbf{W}^{\text{in}}_{D=d}\mathbf{v}(n-d)\right) + \mathbf{b}_{\text{res}}\right) \tag{4}$$

In concrete terms, this means that, in order to compute the current reservoir activation, we consider the historical activation from up to $D_{\max}$ earlier timesteps. We multiply the activation of each of these previous timesteps with their corresponding masked weight matrix, such that we obtain the corresponding delayed neuron input.

Notably, we are now left with the challenge of selecting optimal neuron coordinates. The coordinates could be treated as hyperparameters to be optimized. However, this is inconvenient due to the large amount of neuron coordinates ($2N$). Moreover, for many physical implementations, it is likely unrealistic that location can be assigned precisely. Therefore, we sample the neuron locations from a 2D Gaussian Mixture Model (GMM) distribution, with a fixed number of Gaussians, described by

$$p(\theta) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\mu_{\mathbf{i}}, \mathbf{\Sigma_i}) \tag{5}$$

These Gaussians represent K clusters of neurons. Instead of optimizing each neuron location, we optimize the parameters of this distribution. These consist of the mixture parameters ϕ, means μ, correlations, and variances (each existing for the x and y coordinates), which are used to compute the covariance matrices $\mathbf{\Sigma}$. As such, with K being the number of Gaussians in our GMM and N the number of neurons in our reservoir, we have a drastic reduction from $2N$ location parameters to K values for the mixture parameters, $2K$ values for the means, K values for the correlations and $2K$ values for the variances, resulting in $6K$ parameters. Note that in ESNs, N is usually in the order of hundreds, whereas in our experiments we use $K \leq 4$.

Additionally, network architecture is dependent on reservoir connectivity, which is the fraction of non-zero weights. Using the cluster representation of neurons, connectivity can be defined within and between each cluster rather than a single connectivity parameter. We define a $K+1$ by $K+1$ connectivity matrix. Each element (i, j) in this matrix indicates the fraction of non-zero weights (with respect to full connectivity) from cluster i to cluster j (the last row and column correspond to the input neuron). As such, the diagonal elements indicate the recurrent connectivity of each cluster. We performed experiments with single-cluster, as well as four-cluster networks, in both DDNs and standard ESNs (i.e. $K = 1$ and $K = 4$). Note that even though standard ESNs do not take into account neuron locations, the presence of multiple clusters still influences the size of the connectivity matrix. This allows us to study the effect of

optimizing a multi-cluster connectivity matrix and optimizing neuron locations as two independent factors.

Additionally, like other ESN implementations, optimization of several other hyperparameters is necessary. Specifically, we optimize input weight scaling, reservoir weight scaling, bias scaling, the decay parameter, the fraction of inhibitory neurons, next to the previously mentioned between- and within-cluster connectivity, and location distribution parameters. The scaling parameters are scalars ranging between 0 and 1 that are multiplied with the corresponding subset of weights. The decay parameter, referred to in equation 3 as a, also ranges between 0 and 1.

We make use of a hyperbolic tangent activation function. The values of W_{res}, W_{in} and bias weights $\mathbf{b}_{\text{res}}$ are uniformly sampled between -0.5 and 0.5. We use 300 neurons for our reservoirs. For our readout layer we use ridge regression [10] with 5-fold cross-validation.

To optimize the hyperparameters we make use of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [8, 9]. At each generation, a population of 20 parameter sets (i.e., individuals) is generated. From each parameter set, five networks are generated and evaluated. A fitness score is determined for each parameter set based on the average performance of these five networks, which is fed back to the evolutionary algorithm. In turn, a new population of parameters is returned based on the best performing individual. We run CMA-ES for 99 generations.

In the remainder of this paper we refer to the different parameter sets in a population from one generation as different *candidates*. We refer to the generation of networks based on a set of parameters as *sampling networks from a candidate*, due to the fact that most network aspects are random.

We optimize the hyperparameters of both standard ESNs and DDNs with CMA-ES in order to compare performance. In case of a standard ESN with one cluster (our baseline models), location parameters are not used and the connectivity parameter is just one scalar defining the percentage of non-zero connections in the whole network.

To validate the benefit of variable delays in ESNs, we use a 10th-order Nonlinear Auto-Regressive Moving Average (NARMA) task [2]. This system is described by the following equation.

$$y(t+1) = 0.3y(t) + 0.05y(t)\sum_{i=0}^{9} y(t-i) + 1.5u(t-9)u(t) + 0.1 \qquad (6)$$

Here $y(t)$ and $u(t)$ are respectively the output and input at time t. We generate a training sequence of 8000 samples, a validation sequence of 4000 samples and a test sequence of 10000 samples. We only use the test sequence after evolution, to evaluate our best parameter settings.

NARMA-10 is a commonly used benchmark for ESNs, hence validating our model on this task allows us to understand how DDNs compare to state-of-the-art ESNs. We generate our training, validation and test data with an input sequence that is uniformly distributed between 0 and 0.5 as input. These inputs

are sequentially fed to the NARMA-10 system to generate the task labels. To train the model, the same input sequence is used as input for the reservoir. A linear regression is performed with the reservoir activity as independent variables and the task labels as dependent variables. However, the first 400 samples are discarded before regression, to account for initialization of the reservoir (see [14]).

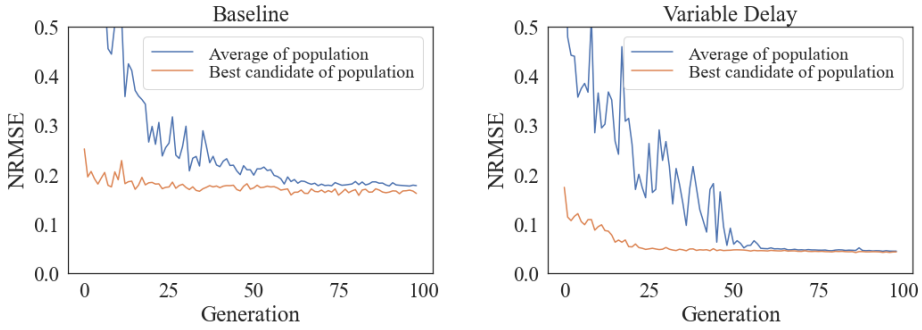
In the case of conventional ESNs, the input at time t corresponds with the network activity at time t and the label at time t. However, this only makes sense if there are no connection delays. Due to the introduction of delays in the proposed DDN reservoir, it takes an unknown amount of additional simulation steps until the information that is relevant for a particular label is sufficiently present in the reservoir activity. Therefore, it is necessary to shift the input values in time with relation to the labels. For this purpose, we introduce a new lag parameter l (indicating the lag of relevant information), which controls how many time steps the labels are shifted. We optimize l by doing a grid search for each network evaluation, i.e., instead of evaluating a network once, we evaluate it multiple times using different positive integer values for l and pick the one that performs best. We measure performance using normalized root mean squared error (NRMSE) [5], considering the lag parameter by pairing every network activation vector $\mathbf{x}(n)$ with label $y(n-l)$.

3 Results

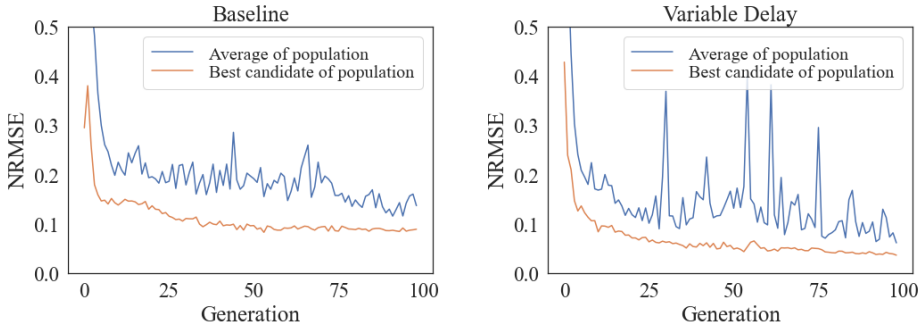
The purpose of our experiments is to establish if the introduction of variable connection delays improves ESN performance. Therefore, we compare the test performance on a fixed dataset of a baseline network (i.e. a network without delays), and a variable delay DDN. Both are tuned using CMA-ES as described in section 2. Our variable delay model has delays that can range between 1 and 20 simulation steps. The actual range of delays will depend on the sampled neuron positions, which in turn depend on location distribution parameters selected by the evolutionary strategy.

We present the results of tuning single cluster and four-cluster networks based on the baseline ESNs and variable delay DDNs with CMA-ES in Figure 1a and Figure 1b respectively. We show the average achieved validation score for the best found set of hyperparameters in a generation as well as the average validation scores achieved by the whole population for 99 generations. The validation performance of a single set of hyperparameters was obtained by training 5 networks previously generated using the set in question. Subsequently, we have evaluated the performance of each network on the validation set, and computed the average of all resulting errors.

We can see that in both experiments, the variable delay models achieve a lower validation NRMSE than the baselines. This difference is especially pronounced in the single cluster experiment. However note that the best candidates of the variable delay model in the four-cluster experiment still show signs of improvement during the final generations. In both experiments we see a larger



(a) Validation performance per generation for single-cluster experiment. We notice that both conditions in this experiment converged to a stable NRMSE. The best performance and average performance are close, and the variance within a population is low (not reported here). We see that the average baseline and variable delay validation NRMSE converge around 0.18 and 0.05 respectively.



(b) Validation performance per generation for four-cluster experiment. Although it appears that the best baseline model has converged to a stable NRMSE, we see that the average NRMSE of both conditions have not yet stabilized. Moreover, in the variable delay condition, both average and best performances are still improving during the final generations. The average validation scores over the last 10 generations for the baseline and variable delay models are 0.14 and 0.09.

Fig. 1: NRMSE on validation set throughout the CMA-ES optimization. We show the average performance per generation averaged over all candidate solutions, as well as the average performance of the best candidate within each generation. The best performing candidate solution of the single-cluster baseline networks achieves an average NRMSE of 0.1579 in generation 81 when evaluated on the NARMA-10 validation set. For the best performing single-cluster variable delay network, we get an average NRMSE of 0.04234 at generation 86. Analogously, the best four-cluster baseline candidate has a validation NRMSE of 0.08319 in generation 53 and the best four-cluster variable delay candidate a validation NRMSE of 0.03696 in generation 98. All four network types consist of 300 reservoir units and all four reported validation scores were averages of 5 networks sampled from the candidate.

drop in NRMSE for the variable delay models. Furthermore, in the single cluster experiment we observe that good scores for the best candidates are obtained after few generations, with already adequate best performances at the start of evolution. Obtaining a good average performance takes additional optimization.

To fairly compare the variable delay models with the baselines, we use the best baseline and variable delay parameter sets to randomly generate 40 networks for both conditions. We train these networks using the same NARMA-10 training set as used during evolution, and evaluate them on a test set. We show the results in performance in Table 1. These test scores are in line with our previous observations, as both DDNs perform consistently better than their respective baselines, but also similarly or better compared what is reported in several recent novel ESN implementations [5, 17, 1].

Note that in our single cluster experiment, sampling neuron location from a GMM with $K = 1$ means that we are in fact sampling from a simple Gaussian distribution, so no extra constraints are added to the connectivity of the weight matrix. As such, we can isolate the use of variable delays as the only alteration compared to conventional ESNs. We can interpret the improvement seen in the $K = 1$ DNNs compared to baseline ESNs as solely the result of variable delays. Although our single-cluster networks achieve the reported scores with random connectivity, in our four-cluster experiment, the within- and between-cluster connectivity is specified and optimized. As such networks are free to evolve into more specific architectures, including deep architectures. Hence, we cannot isolate variable delays as the only cause of improvement. However, we can see that the four-cluster baseline performs better than the single-cluster baseline, but worse than the single cluster DDN. Furthermore, the best four-cluster variable delay candidate achieved the best NRMSE that we report.

Table 1: Average NRMSE on unseen test set of 40 networks sampled from the best candidates from the evolution. Bold font indicates best found approach.

Type	**K**	**NRMSE (test)**
Baseline	1	0.1588 ± 0.0124
DDN	1	0.0639 ± 0.0018
Baseline	4	0.0848 ± 0.0056
DDN	**4**	$\mathbf{0.0391 \pm 0.0025}$

As discussed in Section 2, inter-neuron delays cause an implicit shift in time between the network activation (which can be seen as features for the linear readout) and the task labels. The exact number of simulation steps this shift amounts to can not be determined beforehand, as it is unclear after how many steps the relevant information is present in the reservoir. Therefore, we introduced the lag parameter. The regression is performed repeatedly, with a lag parameter ranging from 0 to 15 for both the baseline as well as our proposed DDN networks. We can use the performance of DDNs with different lag param-

eters to gain insight in the effect of the delays. In Figure 2 we show the effect of

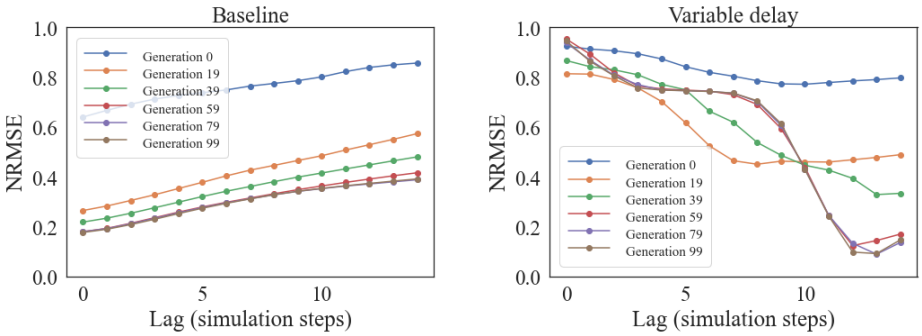


Fig. 2: Best network's average NRMSE on NARMA-10 validation data using different lag parameters, measured for different generations. The left and right plots show performances for baseline candidates and variable delay candidates respectively. The shape of the baseline lag profile remains the same throughout evolution, and only changes in terms of absolute NRMSE. Conversely, the lag profiles of the DDNs do not only change in absolute terms, but the optimal lags are shifted further into the future as evolution progresses.

the lag parameter with respect to performance on the NARMA-10 task. Shown NRMSE validation scores are averaged across an entire generation as a function of lag for various generations throughout the CMA-ES evolution process. The evolution of this lag profile gives insight in the temporal range where the relevant information is located. Note that Figure 2 only shows the performance for the conducted single-cluster experiments. Four-cluster lag profiles were qualitatively similar and differed only in terms of absolute NRMSE or absolute lags. In the variable delay case we see that the lag parameter resulting in the lowest NRMSE grows throughout the evolution. This means that processing of task inputs tends to take longer. It is to be expected that baseline networks consistently find an optimum using a lag parameter of 0. Any other value would have suggested that baseline ESNs are superior at predicting future NARMA-10 states without any knowledge of the corresponding input, compared to predicting simply the next value based on known input. On the other hand, Figure 2 begs the question as to why DDNs always find a non-zero optimal lag, and moreover why the lag parameter grows during evolution. We propose that non-zero delays allow for more variability in delay, and as such we interpret the consistency of non-zero optimal lag throughout evolution as additional support for our hypothesis, namely, that variation in propagation delays can be exploited in echo state networks. To further validate this claim, we plot the variance in delay line length among 10 DDNs generated using the best set of hyperparameters in a single CMA-ES generation in Figure 3. As the reader can see, the the variance of delays indeed grows as the number of generations in the CMA-ES optimisation process increases.

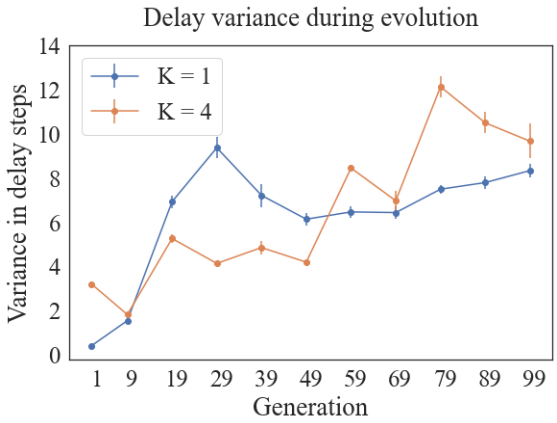


Fig. 3: DDN delay variance in terms of simulation steps throughout evolution. 10 networks are sampled form the best candidate every 10^{th} generation. An average is taken of the variance competed for each of these sample networks.

4 Conclusion

We proposed an extension of echo state networks (ESNs), wherein neurons have a stochastically varying physical location. Proximity between neurons guides the variable propagation delay applied to neuron activation, similar to how variations in axon length and propagation speed causes variable delays in biological brains. Similarly, propagation delay is inherent in all physical systems. We hypothesized that this could be exploited to improve the processing of timeseries. We have shown that the resulting distance delay networks (DDNs) proposed in such a way reach much lower validation and test errors on the NARMA-10 task than baseline ESNs. When testing 40 randomly sampled networks based on the best selected parameters on an unseen test set, we observed that the DDNs performed significantly better. As such, we can conclude that the addition of variable delays in simulated rate-based reservoirs improves performance on the NARMA-10 task. Furthermore, inspired by approaches such as DeepESN [7] we have shown that using multiple clusters in a single DDN can be exploited to enhance performance with other means than just the use of delays. However, our single cluster experiments also achieve similar scores with fully random connectivity, hence isolating variable delays as the cause of improvement.

5 Future work

We see that in our four-cluster experiment, the variable delay models are still improving in performance at the end of evolution, suggesting that there is more performance to be gained, therefore future exploration in this direction seems promising.

Yet, while our findings support our hypothesis, to confirm that this improvement generalises well, our DDNs need to be tested on additional tasks. An especially important question to answer in future work is whether DDNs offer an equally significant improvement in tasks based on natural data (as opposed to artificial tasks such as NARMA-10 task). Additionally, in order to provide better insight into the consistency in results of our methods, results of many evolution runs should be analyzed.

Finally, it should be mentioned that our variable delay implementation runs significantly slower than our baseline. However, as the main objective of this work is to show that variation in propagation speed can be exploited to our benefit, we leave the optimisation of our implementation in order to narrow this performance gap for future work. For newly proposed physical reservoirs, this additional performance cost can usually be avoided since natural substrates implement this varying delay implicitly.

References

1. Akiyama, T., Tanaka, G.: Computational efficiency of multi-step learning echo state networks for nonlinear time series prediction. IEEE Access **10**, 28535–28544 (2022). https://doi.org/10.1109/ACCESS.2022.3158755
2. Atiya, A.F., Parlos, A.G.: New results on recurrent network training: unifying the algorithms and accelerating convergence. IEEE transactions on neural networks **11**(3), 697–709 (2000)
3. Caminiti, R., Carducci, F., Piervincenzi, C., Battaglia-Mayer, A., Confalone, G., Visco-Comandini, F., Pantano, P., Innocenti, G.M.: Diameter, length, speed, and conduction delay of callosal axons in macaque monkeys and humans: Comparing data from histology and magnetic resonance imaging diffusion tractography. Journal of Neuroscience **33**(36), 14501–14511 (2013). https://doi.org/10.1523/JNEUROSCI.0761-13.2013, https://www.jneurosci.org/content/33/36/14501
4. Dambre, J., Verstraeten, D., Schrauwen, B., Massar, S.: Information processing capacity of dynamical systems. Scientific reports **2**(1), 1–7 (2012)
5. Freiberger, M., Bienstman, P., Dambre, J.: A training algorithm for networks of high-variability reservoirs. Scientific reports **10**(1), 1–11 (2020)
6. Gallicchio, C., Micheli, A., Pedrelli, L.: Deep echo state networks for diagnosis of parkinson's disease. arXiv preprint arXiv:1802.06708 (2018)
7. Gallicchio, C., Micheli, A., Pedrelli, L.: Design of deep echo state networks. Neural Networks **108**, 33–47 (2018). https://doi.org/https://doi.org/10.1016/j.neunet.2018.08.002, https://www.sciencedirect.com/science/article/pii/S0893608018302223
8. Hansen, N.: The cma evolution strategy: a comparing review. Towards a new evolutionary computation pp. 75–102 (2006)
9. Hansen, N., Akimoto, Y., Baudis, P.: CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634 (Feb 2019). https://doi.org/10.5281/zenodo.2559634, https://doi.org/10.5281/zenodo.2559634
10. Hoerl, A.E., Kennard, R.W.: Ridge regression: Biased estimation for nonorthogonal problems. Technometrics **12**(1),

55–67 (1970). https://doi.org/10.1080/00401706.1970.10488634, https://www.tandfonline.com/doi/abs/10.1080/00401706.1970.10488634
11. Holzmann, G., Hauser, H.: Echo state networks with filter neurons and a delaysum readout. Neural Networks **23**(2), 244–256 (2010). https://doi.org/https://doi.org/10.1016/j.neunet.2009.07.004, https://www.sciencedirect.com/science/article/pii/S0893608009001580
12. Jaeger, H.: 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 **148**(34), 13 (2001)
13. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach, vol. 5. GMD-Forschungszentrum Informationstechnik Bonn (2002)
14. Jaeger, H.: Echo state network. scholarpedia **2**(9), 2330 (2007)
15. Jaurigue, L., Robertson, E., Wolters, J., Lüdge, K.: Reservoir computing with delayed input for fast and easy optimisation. Entropy **23**(12) (2021). https://doi.org/10.3390/e23121560, https://www.mdpi.com/1099-4300/23/12/1560
16. Jeanson, F., White, A.: Evolving axonal delay neural networks for robot control. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. p. 121–128. GECCO '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2330163.2330181, https://doi.org/10.1145/2330163.2330181
17. Li, Z., Tanaka, G.: Deep echo state networks with multi-span features for nonlinear time series prediction. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–9 (2020). https://doi.org/10.1109/IJCNN48605.2020.9207401
18. Long, J., Zhang, S., Li, C.: Evolving deep echo state networks for intelligent fault diagnosis. IEEE Transactions on Industrial Informatics **16**(7), 4928–4937 (2020). https://doi.org/10.1109/TII.2019.2938884
19. Madadi Asl, M., Valizadeh, A., Tass, P.A.: Dendritic and axonal propagation delays determine emergent structures of neuronal networks with plastic synapses. Scientific reports **7**(1), 1–12 (2017)
20. Tanaka, G., Yamane, T., Héroux, J.B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., Hirose, A.: Recent advances in physical reservoir computing: A review. Neural Networks **115**, 100–123 (2019). https://doi.org/https://doi.org/10.1016/j.neunet.2019.03.005, https://www.sciencedirect.com/science/article/pii/S0893608019300784