Delay-Sensitive Local Plasticity in Echo State Networks

Stefan Iacob
IDLab-AIRO
Ghent University-imec
Ghent, Belgium
stefanteodor.iacob@ugent.be
0000-0002-3424-0390

Spyridon Chavlis
IMBB
FORTH
Heraklion, Greece
schavlis@imbb.forth.gr
0000-0002-1046-1201

Panayiota Poirazi
IMBB
FORTH
Heraklion, Greece
poirazi@imbb.forth.gr
0000-0001-6152-595X

Joni Dambre
IDLab-AIRO
Ghent University-imec
Gent, Belgium
joni.dambre@ugent.be
0000-0002-9373-1210

***Abstract*—Time delays are inherently present in any physical or biological network. However, the role of delays in echo state networks (ESNs) has only been touched upon. In recent years, the use of local plasticity has been explored in the field of reservoir computing, and specifically in ESNs. In this paper, we investigate the role of distance dependent inter-neuron delays in adaptive reservoirs. We introduce a novel ESN design called adaptive distance-based delay network (ADDN), that combines inter-neuron delays with local synaptic plasticity in the reservoir weights using a delay sensitive version of the Bienenstock-Cooper-Munro (BCM) rule. We show that ADDNs perform better on prediction tasks compared to ESNs, regular distance-based delay networks, and ESNs with conventional BCM connections. We optimized the hyperparameters of ADDNs and each of the baseline models using covariance matrix adaptation evolution strategy (CMA-ES). We prove that with ADDNs, we can evolve a single set of hyperparameters that can generate networks which, after unsupervised adaptation, can obtain good performance on different Mackey-Glass sequences with a range of different time constants. By adapting its reservoir weights to the dynamics of the input data, ADDNs can generalize between versions of the same "class" of tasks.**

***Index Terms*—Echo state networks, Distance-based delays, Local plasticity, BCM, Delay-sensitive BCM**

I. Introduction

Echo state networks (ESNs) [1] were originally introduced as an alternative way to train recurrent neural networks. Instead of optimising all weights in a supervised way, conventional ESNs do not require any training of the reservoir weights or the weights that feed the inputs into the reservoir. These are randomly sampled from a chosen distribution and fixed throughout use. Only a task-specific readout layer is trained, which in most cases involves a linear regression between input-driven network activity and corresponding labels. Estimating the weights of a readout layer is usually quick and requires relatively few training samples.

ESNs later became part of the larger field of *reservoir computing*, which now covers theoretical *simulated* dynamical systems as well as a wide variety of physical implementations in different substrates. This *physical* reservoir computing [2], [3] uses the reservoir computing paradigm to optimally exploit the existing interactions and dynamics of hardware implementations. Provided the internal dynamics and time scales of the reservoir match the time scales of the input signals, physical reservoir systems have been argued to perform well on time series data, and could prove especially beneficial in terms of speed or power consumption.

Throughout the evolutions, ESNs have often remained the chosen model system to quantify the effect of varying system parameters ([4]–[6]) or to demonstrate new learning paradigms ([7], [8]). However, one key difference between ESNs, and both their biological and physical counter-parts, is that they lack inter-node delays. Instead, they are normally simulated in discrete time. Depending on the form of the state update equations, all delay is collapsed into either the nodes or the connections and each such delay equals one simulation time step. Instead, physical interconnection delays are usually distance-dependent and in many physical media, interconection delays are non-negligable and in some cases even dominant. The recently introduced, biologically-inspired distance-based delay networks (DDNs) [6] use such naturally distributed delays, which can in fact be optimized to improve ESN performance.

A single ESN can be viewed as a sample from a distribution of possible ESNs, defined by its hyperparameters. As such, each individual ESN can be viewed as a single sample from a distribution of possible ESNs, defined by the hyperparameters. These control, for example, the scaling of the different types of weights. These hyperparameters determine the computational properties of the reservoirs and are usually optimized for each task. Although training a readout-layer for a single network is quick, hyperparameter optimization is usually a lengthy process: each evaluation requires simulating activity for training and validation of the readout layer. The fact that the chosen hyperparameters are task specific greatly limits their practical use. For example, real problems may present variations in dynamics depending on context-dependent variables, even when sampled from slightly different versions of the same task (e.g., forecast of temperatures in different locations). Although ESNs, like any other machine learning model, generalize to unseen data, this only happens reliably when the source data

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.

and target data have similar statistics and dynamics. Trying to optimize a generic set of ESN hyperparameters results in poor generalization between different versions of the same task, as we show later in this paper.

Using synaptic plasticity in reservoir weights has been shown to improve performance by adapting to the dynamics and patterns of the input data [9]–[12]. DDNs on the other hand, improve performance by introducing delays which can be optimized for a task.

In this paper, we introduce a novel ESN model called Adaptive Distance-Based Delay Networks (ADDNs), which combines recently proposed multi-reservoir DDNs with the biologically inspired concept of local synaptic plasticity. We hypothesize that combining these two features will result in better representations of input dynamics in ADDN reservoirs, by matching both delays and reservoir weights to the task at hand. This should result in better performance on harder temporal tasks.

Specifically, we propose a delay-sensitive version of the Bienenstock-Cooper-Munro rule [13] (delay-sensitive BCM) to update reservoir weights based on input data dynamics. This aims to learn sequence-specific temporal patterns. We hypothesize that ADDNs can be optimized to generalize between different versions of the same task.

In order to investigate this, we optimize the hyperparameters of models with and without delays and/or plasticity using covariance matrix adaptation evolution strategy (CMA-ES) [14]. This evolutionary optimization strategy, combined with a multi-reservoir architecture allows ADDNs to evolve different synaptic learning rates (as hyperparameters) for each set of weights between and within all sub-reservoirs. This flexibility results in solutions that have both adaptive and fixed reservoir weights.

Our results show that ADDNs outperform our baseline models, especially in more chaotic (and otherwise difficult to predict) versions of the well-known Mackey Glass [15] time series prediction task. Moreover, we show that distance-based delays in combination with BCM plasticity obtain better task performance than these two adaptations separately, i.e., we find an interaction effect between these two mechanisms on task performance. We find that ADDNs can generalize better than previous models by adapting reservoir weights using local unsupervised learning rules.

The remainder of this paper is organised as follows. In Sections II and III respectively, we discuss the role of delays and plasticity in ESNs. We describe the design and implementation of our ADDN and baseline models in Section IV, and the experimental setup used to validate our model in Section V. We present our findings in Section VI and conclude with VII.

TABLE I
ADDN HYPERPARAMETERS AS A FUNCTION OF NUMBER OF RESERVOIR CLUSTERS K.

Standard ESN hyperparameters			
Name	**Symbol**	**Shape**	**Description**
Weight scaling	$\mathbf{S_w}$	K by K	Element $S_{w,ij}$ is the scaling factor of the reservoir weights from cluster i to cluster j.
Bias scaling	$\mathbf{S_b}$	K	Scaling factor of the bias weights, defined per cluster.
Connectivity	$\mathbf{C}$	K by K	Element C_{ij} specifies the fraction of non-zero weights from cluster i to cluster j.
Decay	$\mathbf{a}$	K	Element a_i specifies the decay parameter from Equation 1 for cluster i.
Location-related hyperparameters			
Name	**Symbol**	**Shape**	**Description**
Component means	μ_l	K by 2	Neuron location mean of each GMM component.
Component variance	σ_l^2	K by 2	Neuron location variance along the x- and y- axis of each GMM component.
Component correlation	ρ_l	K	Neuron location correlation between x- and y-axis of each GMM component.
Mixture weights	ϕ_l	K	GMM-specific mixture weights, determine the probability that a sampled neuron belongs to a specific component.
BCM-related hyperparameters			
Name	**Symbol**	**Shape**	**Description**
Learning rate	$\mathbf{L}$	K by K	The learning rates are multiplied with $\frac{dW}{dt}$. Element L_{ij} is mulitplied with the weights from cluster i to cluster j.
θ-scaling	$\mathbf{y}_0$	K	Element $y_{0,i}$ specifies the value y_0 from Equation 5 for cluster i.

II. DISTANCE-BASED DELAY NETWORKS

An essential component of ADDNs are delayed and distributed signal propagation speeds. This mechanism was introduced in [6] as an extension to the Reservoir Computing framework, implemented using ESNs as a starting point. The resulting distance-based delay networks (DDNs) consist of neurons characterized by a physical location in either 2D or 3D space. DDNs are generated by first sampling the neuron coordinates. Based on these neuron locations, distance-based inter-neuron delays are computed. These are then discretized according to a chosen simulation timestep. As such, the delays themselves are not optimized, but instead follow from the sampled neuron locations.

The network activation of DDNs is formalized as follows:

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

$$\mathbf{y}(n) = \sum_{d=0}^{D_{\max}} \left(\mathbf{W}^{\text{res}}_{D=d}\mathbf{x}(n-d) + \mathbf{W}^{\text{in}}_{D=d}\mathbf{v}(n-d)\right) + \mathbf{b}_{\text{res}} \tag{2}$$

with $\mathbf{x}(n)$, a, $\sigma(\cdot)$, $\mathbf{v}(n)$, and $D_{\max}$ being respectively, the reservoir activity at time n, the decay rate, the sigmoid activation function, the input at time n, and the maximum delay. Lastly, $\mathbf{y}(n)$ is the pre-activation, i.e. the sum of all neuron inputs before applying the activation function. In our experiments, $D_{\max} = 23$. $\mathbf{W}_{D=d}$ refers to a *masked* weight matrix, where all elements corresponding to connections with a delay different from d are set to 0.

In [6], the neuron locations are sampled from a Gaussian Mixture Model (GMM), leaving only the GMM parameters to be optimized (i.e., mixture components, covariance matrices and component means) instead of the individual neuron locations. Although other means of encoding neuron locations are possible, using a GMM results in naturally clustered neurons. Moreover, it is known from which component/cluster each neuron is drawn. The reservoir is divided into subsets of neurons from the same cluster. These subsets are treated as separate interconnected reservoirs, thus resulting in a multi-reservoir system, similar to other presented multi-reservoirs such as [7], [8], [16]. Instead of defining ESN hyperparameters on the whole set of neurons, separate sets of hyperparameters can be defined for each cluster. Parameters related to neuron connections, such as connectivity (i.e., the fraction of non-zero weights) and weight scaling, are defined as matrices, with the element from row i and column j being the connectivity between cluster i and cluster j. In the case of connectivity, the diagonal of such a matrix denotes the intra-cluster connectivity, and the non-diagonal elements specify the inter-cluster connectivity. A summary of the DDN hyperparameters and their respective shape can be seen in the first and second section of Table I. Here, K refers to the number of Gaussian components we use for the DDN, and consequently how many neuron clusters the DDN has. It can also be interpreted as the amount of reservoirs our multi-reservoir DDN contains.

III. Local learning rules in ESNs

In contrast to using fixed weights, the benefit of using adaptive reservoir weights has been shown before [9]–[12]. In these implementations, biologically inspired neural plasticity rules are used to update the weight matrix as a pre-training step. This is usually done with learning rules that only depends on local neuronal activity (i.e., pre- and postsynaptic activity) rather than a global error function. Several different implementations of synaptic plasticity have been explored in ESNs, including the Oja rule, anti-Oja rule, and BCM rule [9]–[12].

The original BCM rule for a single post-synaptic neuron is defined by the following equations [13].

$$\begin{array}{rcl} y & = & \sum_i w_i x_i \\ \frac{dw_i}{dt} & = & y(y-\theta_M)x_i \\ \theta_M & = & E^p[(y/y_o)] \end{array} \tag{3}$$

Here, y is the activity of a neuron with a linear activation function, x_i is the ith presynaptic neuron activity, w_i is the connection weight between x_i and y, θ_M is an adaptive threshold, and y_0 is the target activity for neuron y. We choose $p = 2$. We formalize this for a population of ESN neurons rather than a single postsynaptic neuron as follows:

$$\Delta \mathbf{W} = \eta \odot \mathbf{x}(n) \odot (\mathbf{x}(n) - \theta_M(n))\mathbf{x}^{\mathrm{T}}(n) \tag{4}$$

$$\theta_M(n) = E^2[(\mathbf{x}(n)/y_o)] \approx \left(\frac{1}{T} \sum_{t=0}^{T} \frac{\mathbf{x}(n-t)}{y_o} \right)^2 \tag{5}$$

Where $\odot$ is the element-wise multiplication operator, η is the matrix of learning rates, $\mathbf{x}(n)$ is the population activity, and $E[x]$ denotes the expectation value of x. Here, instead of using the single, postsynaptic activity y, we use the vector of all neuron activities at time n: $\mathbf{x}(n)$. Instead of the presynaptic activity x_i, we use the transpose of the neuron activity vector. Hence, the multiplication of postsynaptic-derived values with presynaptic values becomes an inner product. Multiplications between different postsynaptic activity-derived values (i.e., y and $y - \theta_M$) become element-wise multiplications at population level. Furthermore, in Equation 5 we estimate $E^p[(y/y_o)]$ using an average over a fixed time window of size T. Note that we also include a learning rate, which, along with y_0 is now dependent on new hyperparameters that need to be optimized similar to how the DDN parameters are optimized. Specifically, for all weights that go from cluster i to cluster j, $\eta = \mathbf{L}_{ij}$, where $\mathbf{L}$ is a matrix of hyperparameters (see Table I). Similarly, for all neurons in cluster i, $y_0 = \mathbf{y}_{0,i}$, where $\mathbf{y}_0$ is a vector of hyperparameters. Optimizing learning rates for each cluster-pair means we can naturally arrive at reservoirs that have both fixed and plastic connections, and anything in between. As such, it is possible to obtain deep architectures, similar to [8].

IV. Models

In this section, we present the design of the different models we compare in this paper. Given our hypotheses presented in Section I, we want to study the effect of using synaptic plasticity in DDNs (our novel ADDN), compared to regular DDNs (as described in [6]), regular ESNs with synaptic plasticity (as in [10]), and a standard ESN. In the remainder of this section, we describe the design of these four models.

In the case of standard ESNs, task-specific reservoir hyperparameters need to be optimized. The most important of these parameters is the spectral radius of the reservoir (defined in [1]), which governs the scaling of reservoir weights. These weights are typically sampled from a uniform distribution, and are then scaled to achieve the desired spectral radius. After the initial scaling, the weight matrix is fixed. Spectral radius, along with other ESN hyperparameters such as input weight scaling, bias scaling and connectivity, need to be optimized for a specific task. An overview of ESN hyperparameters can be seen in the first section of Table I. It should be noted that our baseline ESN is in fact a multi-reservoir ESN. Specifically, we use the same clustered hyperparameter representation as described for DDNs in Section II. In fact, our baseline ESN is simply a DDN with all of its delays set to 1 simulation step. As a consequence, the advantage of using multiple clusters for optimizing ESN hyperparameters will be present in all models and will not interfere with the variables that we actually want to research (i.e. plasticity and delays), allowing for a fair comparison between our four models. For all models we will use 6 clusters (i.e., $K = 6$ in Table I) and 300 neurons per reservoir.

A. Delay-Sensitive BCM

We implement synaptic plasticity in ADDNs using the BCM rule. We have to make some additional adaptations to Equation 4 in order to fit within the DDN framework. First, we have to formalize BCM on a reservoir of sigmoid neurons with delayed connections (as shown in Equations 1 and 2). Secondly, note that the change in weight is dependent on current pre- and postsynaptic firing rate. Although we have this firing rate readily available, the original BCM rule aplies only for network simulations without delays. However, if we would use BCM in DDNs without further change, we would disregard the delay that is applied to the pre-synaptic activity. In that case, we actually consider synaptic activity that has not yet arrived at the "synapse" that we are modelling. To account for this, we introduce a delay-sensitive BCM rule, which we formalize similarly to Equation 4:

$$\Delta \mathbf{W}_{D=d} = \eta \odot \mathbf{x}(n) \odot (\mathbf{x}(n) - \theta_M(n))\mathbf{x}^{\mathrm{T}}(n-d) \qquad (6)$$

Where $\mathbf{x}(n)$ is the population activity as obtained from Equation 1. We thus select the pre-synaptic activity of d_i timesteps ago, where d_i is the delay applied by connection i. This is explained in more detail in Figure 1. Hence, our delay-sensitive BCM rule leads to increased weights when high pre-synaptic activity is followed by high post-synaptic activity, exactly d_i steps later. It should be noted that in our implementation, the historical network activities are saved in a buffer up to $\mathbf{x}(n - D_{\max})$, so the term $\mathbf{x}(n-d)$ is readily available for computing the delay-sensitive weight update.

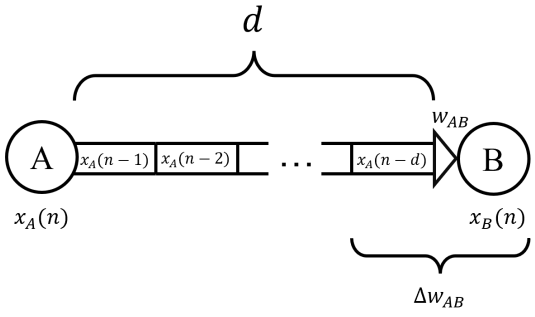

Fig. 1. Diagram showing the flow of activity through the connection between neuron A and neuron B, with a connection that applies d simulation steps of delay. Here, $x_{\mathrm{A}}(n)$ and $x_{\mathrm{B}}(n)$ are the neuron activity at time n of neuron A and B respectively. Assuming that the synapse with weight w_{AB} is located on the "body" of neuron B, the weight update for ΔW should depend on local activity. For the postsynaptic activity, we simply use $x_{\mathrm{B}}(n)$. However, the presynaptic activity that is present at this location has already been delayed by d simulation steps. Hence, as described in Equation 6, for the presynaptic activation we use $x_{\mathrm{A}}(n-d)$.

For the ADDNs, we need to optimize standard ESN hyperparameters, location related (DDN) hyperparameters, and BCM-related hyperparameters (i.e., all rows from Table I)

V. Experimental Setup

A. Task

To validate our ADDNs, we use them to predict the state of Mackey-Glass series, described by Equation 7.

$$\frac{dx}{dt} = \beta \frac{x(t-\tau)}{1 + x(t-\tau)^n} - \gamma x(t) \qquad (7)$$

The parameters β, τ, n, γ are real, positive numbers and $x(t)$ is the state of the system at time t. Here, the goal is to predict the future states x of such a series based on the current and previous states.

The Mackey-Glass equations are often used as a benchmark task for ESN evaluation [10], [17]. The equation can display different periodic or chaotic behaviours, depending on the parameters. As such, varying these parameters can result in qualitatively different problems, presumably with different modelling requirements. Notably, when validating our ADDNs, we intentionally train and test our models with a range of Mackey-Glass series with different parameter settings to show potential to generalize over a class of tasks rather than a single task.

After training the readout, we use our ADDNs to perform *blind prediction* on Mackey-Glass series (see for example [17]). After a warm-up phase, the first input to the ADDN will be the first value from a Mackey-Glass sequence. The ADDN predicts the following state. This prediction is then used as the new input. Due to the fact that any prediction error is fed into the input, this error will compound over time. Our performance measure is the prediction horizon, defined as the amount of steps that the ADDN can predict using the previous prediction as input (i.e., blind predictions), while maintaining an absolute error lower than a chosen error margin. In this paper, we chose an error margin of $0.1 \cdot \sigma_l^2$, where σ_l^2 is the variance of the labels (i.e. the Mackey-Glass sequence to be predicted).

B. Hyperparameter optimization using CMA-ES

Similar to the approach presented in [6], [12], we optimize these hyperparamters using CMA-ES [14]. This is an evolutionary algorithm that is well suited for non-linear and non-convex optimization problems such as the one at hand. All parameters shown in Table I are first scaled to the same range and serialized/flattened into one vector. These are then optimized through CMA-ES using the prediction horizon as the fitness measure. Since we want to compare the performance of different types of networks, we perform a CMA-ES run for each model type, namely, standard ESNs, DDNs, standard ESNs with BCM connections, and ADDNs. We use a population size of 20, and run the evolution for 100 generations. For each of these four models, we measure the performance of 20 candidate hyperparameter solutions provided by CMA-ES. Five networks are sampled from each candidate. For the adaptive models (i.e., the ADDNs and regular ESNs with BCM connections), the reservoir weights

are learned (unsupervised) based on five Mackey-Glass sequences of 500 timesteps. For each sequence, this unsupervised learning phase is done by first presenting 400 warm-up samples while keeping the reservoir weights fixed. Then the following 500 samples are presented serially, while updating the reservoir weights at every simulation step, according to Equation 4 or 6 (for respectively BCM networks or ADDNs). After this unsupervised phase, the reservoir weights are fixed again. The readout weights are trained by teacher forcing, using five sequences of 1000 samples, with the teacher signal being the same as the input signal, but shifted ahead by one timestep. In the case of the non-adaptive models (i.e., the DDNs and regular ESNs) the readout weights are trained using five samples of 1500 timesteps, such that the total amount of training samples (supervised samples plus unsupervised samples) are equal between adaptive and non-adaptive models. All training sequences are generated using Equation 7, for each network separately, with a randomly selected starting value $x(0)$ (sampled from a uniform distribution between 0.5 and 1.2), $\beta = 0.2, n = 10, \gamma = 0.1$, and a random value for τ, sampled from a uniform distribution between 12 and 22. In the case of the adaptive models, the same value for τ is used in the unsupervised stage as in the teacher-forcing stage.

After training, the prediction horizon of each network is measured on five validation sequences of 500 samples (each preceded by 400 warm-up samples, with the same Mackey-Glass parameters as the sequences used for training (except for the starting value, which is picked randomly). We define the performance of this network as the average prediction horizon over the five validation sequences. Note that, due to the size of the validation sequences, in our experiments the theoretical maximum prediction horizon is limited to 500 steps. To determine the fitness of each candidate in a CMA-ES generation's population, we take the average performance of the five networks sampled from the respective hyperparameter set.

C. Evaluation

We select the hyperparameters resulting in the highest validation performance achieved during evolution. Based on these best parameters, 100 different networks are sampled, which are evaluated on 10 different test datasets. Each test set contains five Mackey-Glass sequences with random start value between 0.5 and 1.2, consisting of 500 samples each. Each test set is generated from a Mackey-Glass equation with a different (integer) τ, ranging from 12 to 22. We report the average performance on all 10 test sets, as well as the performance on each test set separately. For each τ, the network is first retrained using the same procedure as during hyperparameter optimization. Then, the trained network is evaluated using the corresponding test set (i.e., with the same Mackey-Glass parameters).

VI. Results

In the following subsections, we present the experimental results for each of the four model types separately and compare their performances. The average prediction horizons of each model type can be seen in Table II. We present the test performances of our models in Figures 3, 4, and 5.

TABLE II
AVERAGE PREDICTION HORIZON PER MODEL, TESTED ON 100 NETWORKS.

BCM	Variable Delay	Prediction Horizon
False	False	54.25
	True	168.75
True	False	91.00
	True	291.21

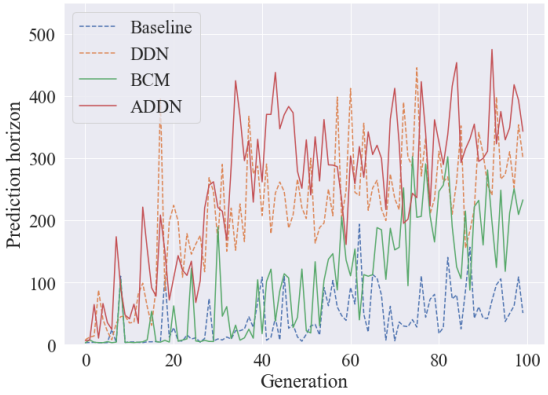

Fig. 2. Validation performance throughout evolution. The graphs show the prediction horizon on the validation set of the best performing candidate for each generation of the CMA-ES evolution run, for each of the four models.

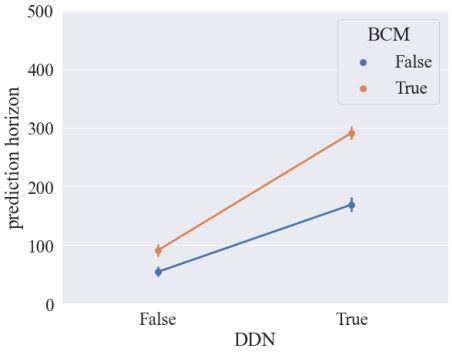

Fig. 3. Test performance of best candidates averaged over all test sets. The x-axis indicates whether distance-based delays were used, and the color of the graphs shows whether BCM connections were used. The y-axis shows the number of time steps the models were able to predict while remaining within the chosen error margin of $.1\sigma_l^2$. The error bars represent the confidence intervals.

It should be noted that the error bars in our figures denote the confidence intervals. However, the standard deviation of the prediction horizons was much larger. This can be explained by the fact that, due to the inherent randomness of reservoirs, some of the initialized networks perform considerably below average. Hence, none of the models had normally distributed test performances, instead having multiple modes. We found standard deviations of 125.0, 189.3, 152.0, and 164.2 steps for respectively baseline, DDN, BCM, and ADDN networks.

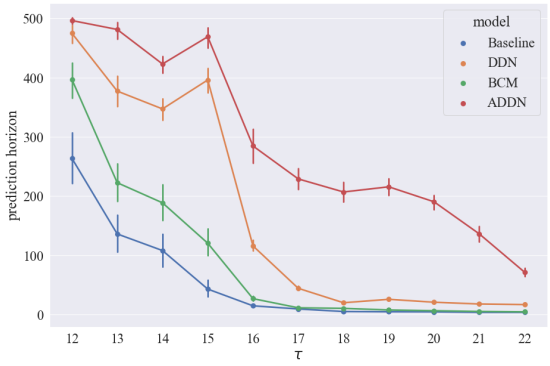

Fig. 4. Test performance of best candidates on separate Mackey-Glass sequences with varying values for τ. The x-axis represents the value chosen for τ in Equation 7 to generate the training and test sequences. The y-axis is the amount of blind prediction the models were able to predict while remaining within the chosen error margin of $.1\sigma_l^2$. The error bars represent the confidence intervals.

A. Baseline Multi-ESN

We present the validation performance of our baseline model, averaged over each generation's population of candidate hyperparameters in Figure 2, represented by the blue dashed line. Note that this validation performance is an average over networks that have been trained and validated on sequences with different, randomly selected values for τ. Given the earlier discussed additional difficulty this introduces, our baseline model cannot be compared against Mackey-Glass performance of other ESNs presented in the literature and as such, we will focus on comparisons between the different models presented in this paper. Indeed, we note that the population average of the validation performance for the baseline ESN remains low throughout evolution. This shows that finding a common set of hyperparameters which generalizes well to different time constants poses a significant challenge for regular ESNs. For the baseline ESN, the highest validation score was achieved in generation 62, with an average prediction horizon of 193.9 timesteps for the best individual of this generation. The 100 networks sampled from the best hyperparameters achieve an average prediction horizon over all test sets of 54.3 timesteps. We present the average prediction horizon for each test set separately in Figure 4, as the blue line. We observe a downward trend in prediction horizon when increasing τ. This is to be expected, because in general, smaller time constants result in more predictable and periodic behavior, whereas larger time constants can result in more erratic and chaotic behavior [15].

B. BCM model

We refer again to Figure 2 for the validation performance of the BCM model. Compared to the baseline, we observe a faster increase and a higher final validation performance, becoming especially prominent after generation 60. The highest validation score was achieved in generation 74, with an average prediction horizon of 302.8 steps for the best candidate. As such, this confirms earlier findings from [9]–[12]: evolving local learning rules in reservoirs benefits the task performance compared to only learning a readout layer using teacher forcing. Moreover, we prove this while also keeping the total amount of training samples constant between the adaptive and non-adaptive models, which suggests that using part of the available training data for unsupervised reservoir adaptation is a more efficient use of training data than teacher forcing with the entire training set.

In Figure 4, we see that the BCM networks (in green) score higher than the baseline networks for $\tau \in [12, 16]$. This suggests that, as sequences grow more chaotic and the task becomes more difficult, the benefits of local learning are diminishing. It could however also be that more samples for unsupervised learning are required to see improvement from using BCM connections, hence further experimentation is required in this regard.

C. DDN

The next jump in performance can be seen in the DDN models. In Figure 2, we see that the validation performance of the DDNs grows quickly from the first generations, with a higher maximum validation performance compared to previously discussed models. The best performing parameter set was found in generation 75, with an average prediction horizon of 445.80 steps. Average DDN performance on the test sets was 168.6 steps. DDNs score consistently higher on the test sets across all τ values compared to baseline and BCM models (see Figure 4). However, the largest increase in test performance is seen in lower τ values. This further strengthens the claims made in [6]: distance-based delays can be optimized to improve performance on temporally difficult tasks.

D. ADDN

From the four models we have tested, the newly introduced ADDNs perform best. During evolution it achieves the highest validation score of 475.2 in generation 92. Although the best performing candidate is found in generation 92, in Figure 2 we see that the validation performance grows quickly until approximately generation 40, after which growth continues but slows down.

For the best candidate hyperparameters, we achieved an average test performance of 291.2 steps averaged across all test sets, making this the best-performing model. Looking at Figure 4, we again notice a consistent increase in test performance compared to previously presented models. However, this time the highest benefit (compared to the DDN model) can be seen for higher values of τ. This suggests that ADDNs are better suited for harder, more chaotic tasks. Hence, we confirm our hypothesis that combining Hebbian learning rules in reservoirs (specifically BCM) with distance-dependent delays, results in a model that is better suited to learn chaotic dynamics at larger timescales. By adding BCM to the baseline model, it becomes possible for the network to adapt its reservoir weights to better capture the input dynamics. Adding variable distance-based delays to the baseline model also improves task performance, possibly explained by an increase in memory capacity (as

suggested by [6]). However, we find the largest benefit when combining these two features. By using our newly introduced delay-sensitive BCM rule, the networks can co-adapt weights with the available delays to better represent input dynamics during the unsupervised stage, while also co-evolving learning rates and neuron distances during the hyperparameter optimization stage. However, the exact mechanism leading to this substantial increase in performance requires further research.

E. Interaction

In Figures 3 and 5, we present interaction plots for, respectively, the test performance averaged over all values of τ and for each τ separately. We see in Figure 3 that the difference in performance between models with adaptive reservoirs and models with fixed reservoirs is greater when using distance-based delays, which suggests an interaction effect between these two features. However, in Figure 5 we note that the size and direction of this interaction effect changes drastically across the different τ values. We observe a negative interaction for $\tau = 12$, followed by no interaction from 13 to 15, and positive interaction for $\tau > 15$. This suggests that the use of delay sensitive BCM is especially useful for harder, more chaotic tasks, while not offering as much benefit for easier, more periodic sequences.

F. Generalization

Because Mackey-Glass task with random time constants is a task that inherently requires generalization across datasets with different dynamics, we can conclude that any performance increase consistent across different time constants (see Figure 4) implies better generalization capacities. As such, we can confirm our second hypothesis, namely that the addition of BCM to reservoirs allows us to generalize to different datasets, by first adapting the reservoir to the dynamics of that dataset through an unsupervised learning phase prior to teacher forcing. More specifically, this adaptation phase is more effective when combined with evolved delays.

VII. Conclusion

In this paper we introduced ADDN, a novel approach to multi ESN systems that combines distance-based delays and delay-sensitive BCM connections in a multi-reservoir architecture. We optimize the hyperparameters of ADDN, along with a baseline ESN, an ESN using normal BCM connections, and DDN with fixed reservoir weights, using CMA-ES. While evolving these hyperparameters, we estimate the fitness of each candidate solution of these four models by measuring system approximation performance on Mackey-Glass sequences with a randomly sampled time constant τ. Here, the aim was to find a single set of hyperparameters for each model type, capable of producing networks that can generalize well between Mackey-Glass sequences with different time constants. We show that, of the four tested model types, ADDN performs best, having the highest overall prediction horizon, and the highest prediction horizon for each time constant separately. This consistency in performance

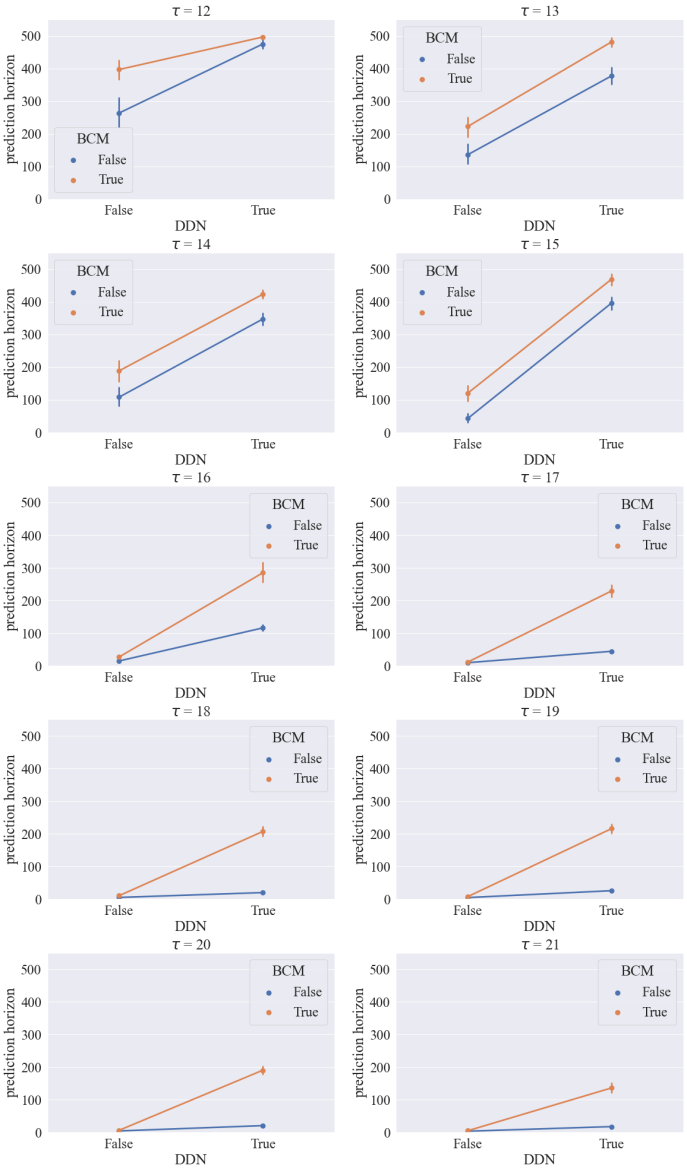

Fig. 5. Alternative representation of Figure 4, using interaction plots. The x-axis indicates whether distance-based delays were used, and the color of the graphs shows whether BCM connections were used. The y-axis shows the performance in terms of prediction horizon within the chosen error margin. The error bars represent the confidence intervals.

across time constants suggests that ADDNs have superior generalization capabilities. This confirms our hypothesis that the delay sensitive BCM allows us to generalize to different datasets, by first adapting the reservoir to the dynamics of that dataset through an unsupervised learning phase prior to teacher forcing. Moreover, we show that the addition of delay-sensitive BCM to DDNs leads to a larger increase in performance compared to adding regular BCM to conventional ESNs. This suggests an interaction effect between the two mechanisms, confirming our second hypothesis.

Note that during hyperparameter optimization of ADDNs, we optimize the learning rates of each inter-cluster weight

layer, and intra-cluster recurrent weight layer. We also optimize the spatial cluster positions and shapes, through the optimization of cluster means and covariance matrices. Hence, a possible explanation for our findings is that distances and BCM learning rates can co-evolve. Moreover, the presence of delay-sensitive BCM connections might allow ADDNs to "select" the needed delays from a range of connections with different delays during the unsupervised learning phase. Connections that capture temporal relations in the input data will be strengthened, whereas other connections might be pruned. However, confirming this explanation requires a better understanding of the exact mechanism of delay-sensitive BCM.

The possibilities of optimizing delays in reservoir computing has important implications for the field of physical reservoirs. All physical systems inherently contain some delay. However, delays should be matched to the task at hand. With the delay-sensitive BCM rule and distributed reservoir delays, it is possible to achieve this.

References

[1] H. Jaeger, "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*, vol. 148, no. 34, p. 13, 2001.

[2] K. Nakajima, "Physical reservoir computing—an introductory perspective," *Japanese Journal of Applied Physics*, vol. 59, no. 6, p. 060 501, May 2020. DOI: 10.35848/1347-4065/ab8d4f. [Online]. Available: https://dx.doi.org/10.35848/1347-4065/ab8d4f.

[3] J. Dambre, A. Katumba, C. Ma, *et al.*, "Computing with integrated photonic reservoirs," in *Reservoir Computing: Theory, Physical Implementations, and Applications*, K. Nakajima and I. Fischer, Eds. Singapore: Springer Singapore, 2021, pp. 397–419, ISBN: 978-981-13-1687-6. DOI: 10.1007/978-981-13-1687-6_17. [Online]. Available: https://doi.org/10.1007/978-981-13-1687-6_17.

[4] B. Vettelschoss, M. Freiberger, and J. Dambre, "Self-organized dynamic attractors in recurrent neural networks," Oct. 2020.

[5] B. Vettelschoss, A. Röhm, and M. C. Soriano, "Information processing capacity of a single-node reservoir computer: An experimental evaluation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2714–2725, 2022. DOI: 10.1109/TNNLS.2021.3116709.

[6] S. Iacob, M. Freiberger, and J. Dambre, "Distance-Based Delays in Echo State Networks," pp. 211–222, 2022. DOI: 10.1007/978-3-031-21753-1_21. [Online]. Available: https://link.springer.com/10.1007/978-3-031-21753-1_21.

[7] M. Freiberger, P. Bienstman, and J. Dambre, "A training algorithm for networks of high-variability reservoirs," *Scientific reports*, vol. 10, no. 1, p. 14 451, 2020.

[8] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep reservoir computing: A critical experimental analysis," *Neurocomputing*, vol. 268, pp. 87–99, 2017, Advances in artificial neural networks, machine learning and computational intelligence, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2016.12.089. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217307567.

[9] Š. Babinec and J. Pospíchal, "Improving the prediction accuracy of echo state neural networks by anti-oja's learning," in *Artificial Neural Networks – ICANN 2007*, J. M. de Sá, L. A. Alexandre, W. Duch, and D. Mandic, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 19–28, ISBN: 978-3-540-74690-4.

[10] M.-H. Yusoff, J. Chrol-Cannon, and Y. Jin, "Modeling neural plasticity in echo state networks for classification and regression," *Information Sciences*, vol. 364-365, pp. 184–196, 2016, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2015.11.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025515008270.

[11] X. Wang, Y. Jin, and K. Hao, "Synergies between synaptic and intrinsic plasticity in echo state networks," *Neurocomputing*, vol. 432, pp. 32–43, 2021, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.12.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220318889.

[12] X. Wang, Y. Jin, and K. Hao, "Evolving Local Plasticity Rules for Synergistic Learning in Echo State Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1363–1374, Apr. 2020, ISSN: 2162-237X. DOI: 10.1109/TNNLS.2019.2919903. [Online]. Available: https://ieeexplore.ieee.org/document/8744482/.

[13] E. Bienenstock, L. N. Cooper, and P. W. Munro, "Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex," in *Journal of Neuroscience*, 1982.

[14] N. Hansen, "The cma evolution strategy: A comparing review," *Towards a new evolutionary computation*, pp. 75–102, 2006.

[15] L. Glass and M. Mackey, "Mackey-Glass equation," *Scholarpedia*, vol. 5, no. 3, p. 6908, 2010, revision #186443. DOI: 10.4249/scholarpedia.6908.

[16] Z. K. Malik, A. Hussain, and Q. J. Wu, "Multilayered echo state machine: A novel architecture and algorithm," *IEEE Transactions on Cybernetics*, vol. 47, no. 4, pp. 946–959, 2017. DOI: 10.1109/TCYB.2016.2533545.

[17] F. Wyffels, B. Schrauwen, D. Verstraeten, and D. Stroobandt, "Band-pass reservoir computing," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 3204–3209. DOI: 10.1109/IJCNN.2008.4634252.