CONTINUAL LEARNING OF DYNAMICAL SYSTEMS WITH COMPETITIVE FEDERATED RESERVOIR COMPUTING

Leonard Bereska
University of Amsterdam
leonard.bereska@uva.nl

Efstratios Gavves
University of Amsterdam
efstratios.gavves@uva.nl

ABSTRACT

Machine learning recently proved efficient in learning differential equations and dynamical systems from data. However, the data is commonly assumed to originate from a single never-changing system. In contrast, when modeling real-world dynamical processes, the data distribution often shifts due to changes in the underlying system dynamics. *Continual learning of these processes aims to rapidly adapt to abrupt system changes without forgetting previous dynamical regimes*. This work proposes an approach to continual learning based on reservoir computing, a state-of-the-art method for training recurrent neural networks on complex spatiotemporal dynamical systems. Reservoir computing fixes the recurrent network weights - hence these cannot be forgotten - and only updates linear projection *heads* to the output. We propose to train multiple competitive prediction heads concurrently. Inspired by neuroscience's predictive coding, only the most predictive heads activate, laterally inhibiting and thus protecting the inactive heads from forgetting induced by interfering parameter updates. We show that this multi-head reservoir minimizes interference and catastrophic forgetting on several dynamical systems, including the Van-der-Pol oscillator, the chaotic Lorenz attractor, and the high-dimensional Lorenz-96 weather model. Our results suggest that reservoir computing is a promising candidate framework for the continual learning of dynamical systems. We provide our code for data generation, method, and comparisons at https://github.com/leonardbereska/multiheadreservoir.

1 INTRODUCTION

Modeling complex spatiotemporal data is a universal problem in diverse branches of science, such as climate- (Rolnick et al., 2019b; Reichstein et al., 2019), health- (Fresca et al., 2020) or neuroscience (Pandarinath et al., 2018). The complex systems in these domains are high-dimensional *and* highly nonlinear, making it impossible to set up models from first principles (Strogatz, 1994). As a result, recent years have seen data-driven learning algorithms for identifying nonlinear dynamical systems (DS) become increasingly popular (Brunton et al., 2016; Pathak et al., 2018; Vlachas et al., 2018; Lu et al., 2018; Raissi et al., 2018; Champion, 2019; Karniadakis et al., 2021). However, these previous approaches to modeling complex data assume the data is independent and identically distributed (i.i.d.) and comes from a single, unchanging environment. In real-world settings, however, the generating distribution, as is described by the (partial) differential equations of the DS, can shift abruptly. Thus, models trained with the i.i.d. assumption need to either continue training and risk overwriting previous knowledge or restart training from scratch if the distribution's change-point is known. To this end, we propose continual learning (CL) of DS, which will allow models to learn from and adapt to such novel and time-varying dynamical environments while remembering previously observed dynamics.

Recently, learning DS beyond the i.i.d. assumption from multiple environments has been proposed in the multi-task setting (Yin et al., 2021; Wang et al., 2021; Kirchmeyer et al., 2022), where data from different environments is available. The more challenging and restrictive CL setting assumes that access to data is only granted sequentially either due to security or privacy reasons or as a realistic assumption for a real-world intelligent agent. Under such conditions, in contrast to biological agents, neural network-based models suffer from catastrophic forgetting (McCloskey & Cohen, 1989; French, 1999), destroying previously learned knowledge as they train on novel data. This trade-off between being stable enough not to forget the old and yet being plastic enough to rapidly learn the new is known as the *stability-plasticity dilemma* (Mermillod et al., 2013; Grossberg, 2013; Ditzler et al., 2015).

So far, CL research has focused heavily on minimizing forgetting on *discriminative tasks* (e.g., classification) with *static* data (e.g., images). However, neuroscientific evidence (Frölich et al., 2021) shows that memory is both *generative* and *sequential*, rather than *discriminative* and *static*. The generative nature of memory is evidenced by a neuroscientific theory called predictive coding, which suggests that the brain encodes top-down generative models

to predict the next sensory input from lower levels (Buzsáki & Tingley, 2018). Moreover, a tremendous amount of evidence from human ethology, physiology, and neuroscience shows that human memory operates via neuronal sequences as representations even in situations that seemingly provide only static sensory input (Frölich et al., 2021). Despite this evidence highlighting the need for CL in a generative sequential setting, research in this setting is scarce, with notable exceptions (Ororbia et al., 2020; Cui et al., 2016; Grewal et al., 2021; Iyer et al., 2022). Crucially, to the best of our knowledge, no studies have focused on applications to DS.

To continually learn DS, we propose reservoir computing (RC) (Jaeger, 2001; Maass et al., 2002), a state-of-the-art training scheme for recurrent neural networks on complex spatiotemporal DS' data (Pathak et al., 2018; Lu et al., 2018; Röhm et al., 2021). In addition to promising good performance for DS identification, the training mechanism promises to improve CL by fixing the recurrent network weights, preventing catastrophic overwriting by subsequent parameter updates, and training only a linear output layer (*head*). Similar ideas have been proposed for the static setting (Wortsman et al., 2020; Ramanujan et al., 2020) and also for the sequential discriminative setting (Cossu et al., 2021a; Kobayashi & Sugino, 2019). In this work, we draw inspiration from predictive coding to protect the reservoir output *head* from interference (Ororbia et al., 2021). Accordingly, we train multiple heads concurrently, each trying to predict the observed sequence. The best matching prediction heads activate sparsely and thereby laterally inhibit competing ones (Aljundi et al., 2018b). Only the active heads are updated in a continual manner with a federated RC (Bacciu et al., 2021) update scheme. Thus, the inactive heads' knowledge of previous environments stays untouched. As a result, we resolve the plasticity-stability dilemma by being simultaneously highly stable (fixed weights) and highly plastic (new head for new environment).

In summary, our contributions are:

i). We propose competitive multi-head reservoir computing as a brain-inspired framework to continually train with reservoir computing.

ii). We are, to the best of our knowledge, the first to address continual learning for dynamical systems.

iii). Thereby, we provide a use case for reservoir computing for the understudied setting of Sequential Generative continual learning more generally.

We show the effectiveness of this competitive multi-head reservoir in minimizing forgetting on several benchmark DS, namely on the Van-der-Pol oscillator, the chaotic Lorenz attractor, and the high-dimensional Lorenz-96 atmosphere model.

2 METHOD

In the following, we describe our method for training a multi-head reservoir to continuously adapt to different dynamical systems environments.

Desiderata for continually generating sequences. We start by listing several properties that we consider desirable for a sequential generative continual learning algorithm:

I. *Multiple simultaneous predictions* (Hawkins & Ahmad, 2016): For a given context, there are often multiple possible future trajectories. It is crucial to keep multiple competing hypotheses simultaneously and evaluate the likelihood of each hypothesis, especially when the context is ambiguous.

II. *Continual training and testing* (Hawkins & Ahmad, 2016): The difference between the training and testing set should be minimal to allow online adjustment to change.

III. *No external task supervision* (Lesort et al., 2020): In real-world settings, the model should determine the current task from the data alone without supervision in the form of task labels.

IV. *No storage of raw sensory data* (Lesort et al., 2020): Our goal is to develop a sequence memory that compresses high-dimensional perceptual data and extracts information for prediction. Storing raw sensory data violates that objective.

Sequential Generative Continual Learning. The learning algorithm is trained on sequences $\boldsymbol{X}_{1:T} = \{\boldsymbol{x}_t | t = 1, \ldots, T\}$ of d-dimensional vectors $\boldsymbol{x}_t \in \mathbb{R}^d$. First, sequences stem from the same distribution, and at some point (unknown to the algorithm), the distribution shifts abruptly to a different environment, *e.g.,* other underlying system equations. Given a sequence fragment $\boldsymbol{x}_{1:\tau}$ as cue, the task is to remember the future trajectory of that sequence $\boldsymbol{x}_{\tau:T}$. The challenge is to not forget sequences from the previous environment after training only on sequences from the

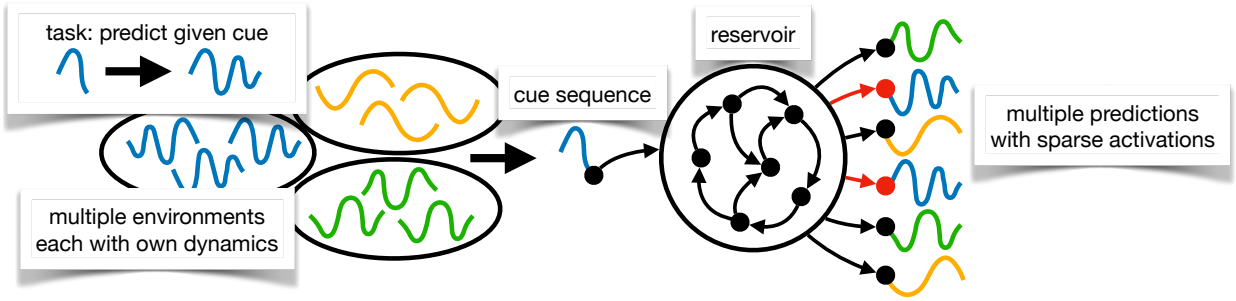


Figure 1: Overview of task setting and method: The model is trained on sequences drawn from multiple environments. Each environment is encountered only once, without replay. Given a fragment of a training sequence, the model should associate/predict the full training sequence. Since it is trained successively on the environments, the challenge is not to forget dynamics from previous environments when encountering new ones. A fixed-weight reservoir with multiple prediction heads of which only a subset sparsely activates and is updated achieves this.

novel environment. Accordingly, the learning algorithm is trained on K environments $\{\mathcal{E}_k\}_{k=1,\dots,K}$ in succession, the number of which is unknown to the algorithm. The training data as a whole is thus an ordered collection of sequences $\{\boldsymbol{X}_{1:\tau}^{(n)}\}_{n=1,\dots,N}$, where the algorithm is agnostic to the origin-environment of each sequence $\boldsymbol{X}_{1:\tau}^{(n)}$.

Reservoir Computing starts with a randomly connected recurrent neural network. By fixing the recurrent network weights, RC forgoes the notorious vanishing gradient problem arising from backpropagation through time. The state of the recurrent neural network of M neurons is given by its hidden activations $\boldsymbol{h} \in \mathbb{R}^M$, which are connected by *randomly initialized and fixed* matrix $\boldsymbol{W}_\mathrm{h}$. An input sequence $\boldsymbol{X}_{1:\tau}$ is embedded by a linear map $\boldsymbol{W}_\mathrm{i}$ to the state $\boldsymbol{h}_\tau$:

$$\begin{aligned} \boldsymbol{h}_1 &= \sigma(\boldsymbol{W}_\mathrm{h}\boldsymbol{h}_0 + \boldsymbol{W}_\mathrm{i}\boldsymbol{x}_1) \\ \vdots \qquad & \qquad \vdots \\ \boldsymbol{h}_\tau &= \sigma(\boldsymbol{W}_\mathrm{h}\boldsymbol{h}_{\tau-1} + \boldsymbol{W}_\mathrm{i}\boldsymbol{x}_\tau). \end{aligned} \tag{1}$$

The reservoir linearly predicts the next input $\boldsymbol{x}_{\tau+1}$:

$$\hat{\boldsymbol{x}}_{\tau+1} = \boldsymbol{W}_\mathrm{o}\boldsymbol{h}_\tau, \tag{2}$$

with $\boldsymbol{W}_\mathrm{o}$ mapping from hidden activation to output. Only the parameters of this output matrix $\boldsymbol{W}_\mathrm{o}$ are trained. Hence, the optimal values can be obtained analytically via ridge regression (with regularization parameter λ) as:

$$\boldsymbol{W}_\mathrm{o} = (\boldsymbol{H}^T\boldsymbol{H} + \lambda\boldsymbol{I})^{-1}\boldsymbol{H}^T\boldsymbol{X}, \tag{3}$$

where $\boldsymbol{H}$ are the hidden states, $\boldsymbol{I}$ the identity matrix, and $\boldsymbol{X}$ the targets of the regression.

Federated Reservoir Computing. When training is decentralized onto multiple clients for data security or privacy reasons, federated learning provides a framework to integrate the learning progress of these clients into one shared machine learning model. In our case, the training data is not scattered in space but in time: continual learning prohibits accessing all data points at once. Therefore, we propose transferring the machinery of federated learning to continual learning by interpreting different clients as different points in time. Following Bacciu et al. (2021), we break up the analytic (and thus instantaneous) calculation of the optimal output weights in Eq. 3 into components:

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{H}^T\boldsymbol{H} \\ \boldsymbol{B} &= \boldsymbol{H}^T\boldsymbol{X}, \end{aligned} \tag{4}$$

such that Eq. 3 turns into $\boldsymbol{W}_\mathrm{o} = (\boldsymbol{A} + \lambda\boldsymbol{I})^{-1}\boldsymbol{B}$. These terms $\boldsymbol{A}$ and $\boldsymbol{B}$ are then additive *w.r.t.* successively incoming data, as multiple iterative data fragments $\boldsymbol{H} = \begin{pmatrix}\boldsymbol{h}_1 \\ \boldsymbol{h}_2\end{pmatrix}$, $\boldsymbol{X} = \begin{pmatrix}\boldsymbol{x}_1 \\ \boldsymbol{x}_2\end{pmatrix}$ decompose into sums: $\boldsymbol{A} = \boldsymbol{h}_1^T\boldsymbol{h}_1 + \boldsymbol{h}_2^T\boldsymbol{h}_2$ and $\boldsymbol{B} = \boldsymbol{h}_1^T\boldsymbol{x}_1 + \boldsymbol{h}_2^T\boldsymbol{x}_2$. Hence, for training successively at time step t, it suffices to keep a copy of $\boldsymbol{A}^{(t-1)}$ and $\boldsymbol{B}^{(t-1)}$:

$$\begin{aligned} \boldsymbol{A}^{(t)} &= \boldsymbol{A}^{(t-1)} + \tilde{\boldsymbol{A}} \\ \boldsymbol{B}^{(t)} &= \boldsymbol{B}^{(t-1)} + \tilde{\boldsymbol{B}}, \end{aligned} \tag{5}$$

where $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$ are calculated on incoming data at time step t according to definition in Eq. 4. Output weights are again calculated as:

$$\boldsymbol{W}_\mathrm{o}^{(t)} = (\boldsymbol{A}^{(t)} + \lambda\boldsymbol{I})^{-1}\boldsymbol{B}^{(t)}. \tag{6}$$

Competitive Multi-Head Reservoir. We want the learning algorithm to make multiple predictions when presented with a sequence (desideratum 1.). Hence, we train a reservoir with L linear heads. The number of prediction heads L should be much higher than the number of environments K that the algorithm is expected to learn continually: $L \gg K$. Although K is assumed unknown, in principle, we can dynamically add (with linearly rising memory cost) new heads when shifts in environments are detected.
To iteratively learn with federated reservoir computing, we then simultaneously maintain L prediction heads $\{\boldsymbol{W}_o^{(l)} | l = 1, \ldots, L\}$, corresponding to L respective matrices $\boldsymbol{A}^{(l)}, \boldsymbol{B}^{(l)}$. All heads compete for predicting the sequence most accurately, as measured by the Mean Squared Error (MSE). Given a sequence $\boldsymbol{X}_{1:T}$, the reservoir state $\boldsymbol{h}_t$ for time step t is obtained by embedding $\boldsymbol{X}_{1:t}$ via Eq. 1. The Squared Error (SE) for head l is then:

$$\mathrm{SE}^{(l)} = (\hat{\boldsymbol{x}}_{t+1}^{(l)} - \boldsymbol{x}_{t+1})^2 = (\boldsymbol{W}_o^{(l)} \boldsymbol{h}_t - \boldsymbol{x}_{t+1})^2. \tag{7}$$

For each sequence we take the average over all time steps t to obtain $\mathrm{MSE}^{(l)}$ from $\mathrm{SE}^{(l)}$. Heads with the lowest MSE activate unless a new environment is detected, in which case new (previously never active) heads become activated. Only active heads receive parameter updates, which protects inactive heads from catastrophic forgetting. We visualize the competitive multi-head reservoir and its application to continual learning of dynamical systems in Fig. 1. The complete algorithm in pseudocode is revealed in Algo. 1.

Algorithm 1 Competitive Multi-Head Reservoir Computing

Require: Ordered collection of input sequences $\{\boldsymbol{X}_{1:\tau}^{(n)}\}_{n=1,\ldots,N}$.
Require: Novel environment detection threshold $\theta_{\text{new env}}$.
  Initialize (sparse-random) reservoir $\boldsymbol{W}_h, \boldsymbol{W}_i$.
  Initialize (randomly) L heads $\{\boldsymbol{W}_o^{(l)}\}_{l=1,\ldots,L}$.
  Initialize $\boldsymbol{A}^{(l)}, \boldsymbol{B}^{(l)}$ to zero (matrices).
  Initialize *never-before-active heads* as set that contains all heads.
  Initialize *previous active heads* as empty set.
  for $n = 1$ to N **do**
    Embed sequence $\boldsymbol{X}_{1:\tau}^{(n)}$ according to Eq. 1 to obtain reservoir state $\boldsymbol{h}_\tau$.
    Calculate ahead prediction $\mathrm{MSE}^{(l)}$ for each head $\boldsymbol{W}_o^{(l)}$.

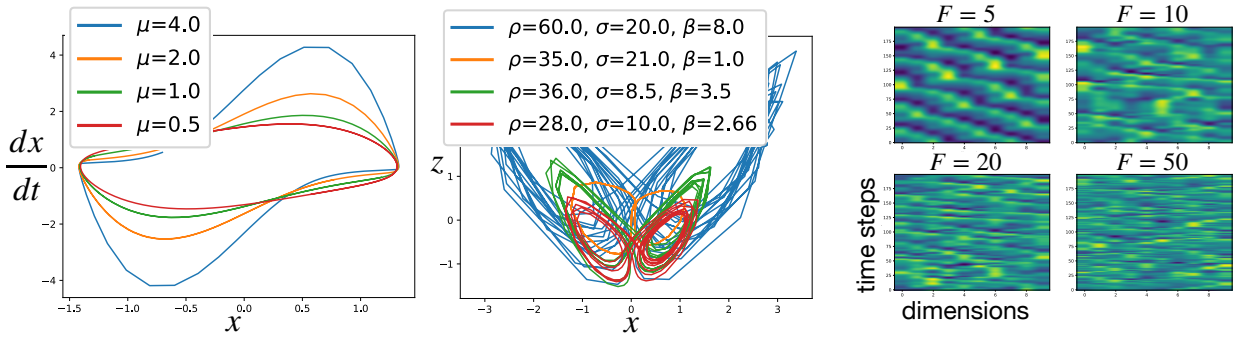
    if detect novel environment with $\mathrm{MSE}^{(l)}/\mathrm{MSE}_{\text{last}}^{(l)} > \theta_{\text{new env}}$ for any l of *previous active heads* **then**
      active heads $\leftarrow$ random heads from *never-before-active heads*.
    else
      active heads $\leftarrow$ heads with lowest MSE.
    end if

    Update $\boldsymbol{A}^{(l)}, \boldsymbol{B}^{(l)}$ and $\boldsymbol{W}_o^{(l)}$ for l in *active heads* according to Eq. 5 and 6.

    $\mathrm{MSE}_{\text{last}}^{(l)} \leftarrow \mathrm{MSE}^{(l)}$, for all l.
    previous active heads $\leftarrow$ *active heads*.
    Remove *active heads* from *never-before-active heads*.
  end for

3 Experiments

We experiment on three families of dynamical systems that exhibit a diverse set of dynamical regimes: The nonlinear Van-der-Pol (VdP) oscillator is a representative of limit cycle behavior, the famous chaotic Lorenz-63 (L63) attractor is a simple chaotic system and the Lorenz-96 (L96) atmosphere model is a high-dimensional chaotic system. We instantiate the respective differential equations with varying sets of parameter values to obtain four different environments per system. Thus, different environments share the same functional form of the underlying equations but differ in particular values for the parameters. *Therefore, a shift from one environment to the next can be interpreted as an abrupt bifurcation.*



(a) (b) (c)

Figure 2: Environments for the (a) Van-der-Pol oscillator, the (b) Lorenz-63 attractor, and the (c) Lorenz-96 model.

3.1 Datasets and environments

Van-der-Pol (VdP). The VdP oscillator is a non-conservative oscillator with nonlinear damping. It is defined through the second-order differential equation:

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} - \mu(1-x^2)\frac{\mathrm{d}x}{\mathrm{d}t} + x = 0, \tag{8}$$

and evolves on a limit cycle. The damping parameter μ characterizes the shape of this cycle. Lower values of μ result in a more circular shape; for $\mu = 0$ (no damping), the system becomes the conservative simple harmonic oscillator. Our environments defined by $\mu \in \{0.5, 1, 2, 4\}$ are shown in Fig. 2a.

Lorenz-63 (L63). The classical three-dimensional Lorenz attractor (Lorenz, 1963) was the first example of a low-dimensional system with chaotic solutions. The L63 system is described by the differential equations:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y-x), \frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y, \frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z. \tag{9}$$

It is frequently used as a benchmark system for DS reconstruction algorithms (Pathak et al., 2018). We use the following parameter instantiations as environments: $(\rho, \sigma, \beta) \in \{(28, 10, 2.66), (36, 8.5, 3.5), (35, 21, 1), (60, 20, 8)\}$ (visualized in Fig. 2b).

Lorenz-96 (L96). The Lorenz-96 model (Lorenz, 1996) was originally constructed by Edward Lorenz as a model for the laminar nature of the earth's atmosphere. The dimensionality D can be freely chosen, as the system is recursively defined, by

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F,\ i = 1, \ldots, D; \tag{10}$$

with the forcing term F as only parameter, defining the environments by $F \in \{5, 10, 20, 50\}$ (Fig. 2c). It is characterized by nearest-neighbor interactions (x_i interacting with neighboring x_{i-2}, x_{i-1}, and x_{i+1}, as directly seen from Eq. 10) and is thus easily visualized as a 1D image. Like the L63, it is a common benchmark for DS reconstruction.

Simulation details. We simulate the systems with `scipy.odeint`, with step size 0.05. The initial state is randomly (uniform distribution for each dimension) assigned. To only obtain the limit distribution, we cut transient behavior by dropping the first 100 time steps. We simulate 10 sequences per environment, each for 200 time steps. And split the data into 7 sequences for training and 3 sequences for testing. We standardize the data (subtract mean and divide by standard deviation) jointly over all four environments per system.

3.2 Results

We consider the VdP, L63, and L96 datasets, where we train on 4 environments with 7 trajectories per environment. For evaluation, we create a separate test set of 3 trajectories, where the multi-head reservoir's active head is chosen according to the 10-time step cue (out of a total of 200 time steps per sequence). To quantify the performance, we choose the one-step-ahead prediction MSE (in units of 10^{-3}), averaging over 10 experiments for each setting. The MSE is given in absolute terms, but as the data are normalized, they can also be interpreted as relative to the standard deviation of the data.

Comparing to methods for continual learning. For lack of comparison methods in the domain of generative sequential continual learning, we compare our method, the competitive multi-head reservoir, to two constructed baselines that unfortunately do not satisfy our desiderata completely:

i). Similarly to Cossu et al. (2021a), we combine a Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) model with a general-purpose continual learning regularization technique, namely Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017). This method does not make multiple simultaneous predictions (contrary to our first desideratum, Sec. 2) and needs to be supplied with task labels (disobeying our third desideratum).

ii). In addition, we compare to a simple replay baseline again using a simple LSTM. Replay is considered one of the most effective continual learning methods (van de Ven et al., 2020) if one allows for storing raw data (we do not consider that desirable: see our fourth desideratum). Following Lopez-Paz & Ranzato (2017) we save the last sequence of each dataset in a buffer and randomly intersperse the regular training with buffer training steps with a frequency of 1%.

The comparison to these baselines is shown in Tab. 1.

Table 1: Comparing to other continual learning methods. We give the one-step-ahead prediction mean squared error (in units of 10^{-3}), and the standard error of the mean in brackets. Note that the fourth environment is not indicative of continual learning performance as it is trained last and thus not subject to catastrophic forgetting.

Dataset	Desiderata	Method	Environment 1	2	3	4
VdP	II, IV	LSTM+EWC	10.1 ± 0.2	7.9 ± 0.2	4.3 ± 0.1	$0.000\,306 \pm 0.000\,008$
	II	LSTM+Replay	0.54 ± 0.01	0.286 ± 0.008	0.59 ± 0.02	5.9 ± 0.3
	I, II, III, IV	Ours	1.3 ± 0.1	1.8 ± 0.2	3.5 ± 0.4	20 ± 5
L63	II, IV	LSTM+EWC	463 ± 6	424 ± 5	238 ± 3	0.053 ± 0.003
	II	LSTM+Replay	19.4 ± 0.2	19.7 ± 0.3	9.0 ± 0.1	454 ± 5
	I, II, III, IV	Ours	7.7 ± 0.8	60 ± 4	3.6 ± 0.3	26 ± 2
L96	II, IV	LSTM+EWC	162.0 ± 0.6	133.0 ± 0.5	111.0 ± 0.6	332 ± 3
	II	LSTM+Replay	1.78 ± 0.02	5.44 ± 0.04	27.7 ± 0.2	430 ± 4
	I, II, III, IV	Ours	2.93 ± 0.06	15.9 ± 0.3	197 ± 7	2470 ± 60

Comparing the continual, single-task and multi-task setting. To provide some context, we also compare to two relaxations of continual learning, namely the:

i). *Single-task* setting: Training and evaluating each environment separately. This constitutes an upper bound on performance (*i.e.* lower bound on MSE).

ii). *Multi-task* setting: Joint training, with access to all environments at once. While accessing all data at once may benefit training an LSTM, for the RC training, this setting can be seen as a lower bound on performance, as a single head needs to cater to different environments and is thus forced to interpolate across environments with merely a linear layer.

Note that replay can be seen as an interpolation between the continual and the multi-task setting, but we still list it under the continual learning methods in Tab. 1. For these non-continual settings, we compare to RC, a standard LSTM, and Sparse Identification of Nonlinear Dynamical systems (SINDy) (Brunton et al., 2016), a state-of-the-art method for identification of dynamical system equations. SINDy sparsely regresses coefficients for a library of terms and hence directly learns the governing equations of the dynamical system. We use the publicly available `PySINDy` package (de Silva et al., 2020) with default parameters (which are chosen to work very well on the popular DS benchmarks that we work with by default).

3.3 Discussion

Continual learning. The LSTM seems to be the more powerful approximation method in comparison to RC, as becomes clear from the orders-of-magnitudes-better performance in the single- and multi-task settings, cf. Tab. 2.

Despite being generally more powerful, in the continual learning setting, the LSTM forgets previous environments catastrophically, and the regularization by elastic weight consolidation does not prevent this. If trained with replay, on the other hand, the LSTM does not suffer from catastrophic forgetting, as can be expected from interpolation between the continual and the multi-task setting. In fact, it performs on par with the multi-task setting (cf. Tab. 2). Comparing our competitive multi-head RC to these two methods shows that catastrophic forgetting is indeed mitigated. As shown in Tab. 1, our competitive multi-head reservoir remembers how to predict previous environments in the continual setting.

Single- and multi-task setting. The LSTM performs well both in the single-task and the multi-task setting. In contrast, for RC, multi-tasking overstretches the capacity of a single prediction head. Note that the competitive multi-head reservoir keeps the performance close to its upper bound, the single-task RC. SINDy works best in the single-task setting (for which it is designed) and naturally struggles with integrating multiple environments into one dynamical system equation.

Table 2: Comparing to single- and multi-task settings. We give the one-step-ahead prediction mean squared error (in units of 10^{-3}), and the standard error of the mean in brackets.

Dataset	Setting	Method	Environment 1	2	3	4
VdP	Single-task	RC	4.5 ± 0.2	2.8 ± 0.3	8.5 ± 0.7	10 ± 10
		LSTM	0.0195 ± 0.0003	0.0328 ± 0.0004	0.0554 ± 0.0006	0.054 ± 0.002
		SINDy	0.273 ± 0.004	1.18 ± 0.04	5.0 ± 0.2	19.0 ± 0.8
	Multi-task	RC	20 ± 2	15 ± 1	3.3 ± 0.3	7 ± 1
		LSTM	0.409 ± 0.007	0.162 ± 0.004	0.372 ± 0.009	1.23 ± 0.05
		SINDy	0.45 ± 0.01	1.28 ± 0.03	5.6 ± 0.1	19.4 ± 0.9
	Continual	Ours	1.3 ± 0.1	1.8 ± 0.2	3.5 ± 0.4	20 ± 5
L63	Single-task	RC	8.5 ± 0.9	31 ± 2	4.0 ± 0.4	24 ± 2
		LSTM	0.0358 ± 0.0009	0.0553 ± 0.0008	$0.002\,94 \pm 0.000\,03$	0.261 ± 0.005
		SINDy	0.385 ± 0.007	2.33 ± 0.04	0.354 ± 0.007	262 ± 4
	Multi-task	RC	7.5 ± 0.3	34 ± 2	4.7 ± 0.3	180 ± 10
		LSTM	9.8 ± 0.2	40.6 ± 0.7	19.9 ± 0.3	101 ± 2
		SINDy	34.7 ± 0.9	61 ± 1	35.7 ± 0.6	diverging
	Continual	Ours	7.7 ± 0.8	60 ± 4	3.6 ± 0.3	26 ± 2
L96	Single-task	RC	0.99 ± 0.03	7.2 ± 0.2	189 ± 4	4180 ± 80
		LSTM	0.142 ± 0.001	1.39 ± 0.01	20.4 ± 0.2	764 ± 5
		SINDy	0.0239 ± 0.0004	0.523 ± 0.005	14.2 ± 0.1	783 ± 9
	Multi-task	RC	71 ± 2	203 ± 4	650 ± 10	2580 ± 60
		LSTM	0.359 ± 0.002	1.49 ± 0.01	9.30 ± 0.07	169 ± 2
		SINDy	41.0 ± 0.3	36.1 ± 0.3	44.9 ± 0.4	404 ± 5
	Continual	Ours	2.93 ± 0.06	15.9 ± 0.3	197 ± 7	2470 ± 60

3.4 Experimental details

For initializing the reservoir, we use a radius of 0.6, sparsity of 0.01, and a reservoir size of $M = 1000$, which are default hyperparameters we take from Pathak et al. (2018). We take the first 10 time steps of each sequence as a cue for initializing the hidden state, and when predicting the future (on test data), the first 5 steps, while the following 5 steps are used for determining the prediction head. We obtain the regularization parameter λ for the reservoir by validation on a separate validation set (cf. Appendix, Fig. 4), resulting in $\lambda_{\text{L63}} = 10^{-6}$, $\lambda_{\text{L96}} = 5$, and $\lambda_{\text{VdP}} = 10^{-6}$. We use the same hyperparameters for all settings. For the multi-head architecture, we use ten heads, with one head active at any given time. We determine the drift detection threshold for detecting a new environment on the validation set (cf. Appendix, Fig. 3), where we exclude values below 2 due to too many false-positive detections. In practice, an order (or even multiple orders) of magnitude increase in MSE is typical as the environment changes. Hence, the drift detection works for an extensive range of threshold values between 2 and 100, as an analysis (on a separate validation

set) of the accuracy shows (cf. Appendix, Fig. 5). We choose a doubling in error (factor 2 increase, thus a value of 2 for the threshold) as a drift detection threshold. We run all experiments on an Apple M1. Running all experiments related to RC (three datasets, three methods, four environments, hence 36 settings (as in Tab. 2 with 10 data points per setting) takes only a couple of minutes, in contrast to the LSTM training that takes a couple of hours. This shows the computational efficiency of the one-shot linear regressions used by RC in comparison to the gradient-based LSTM training.

4 Related Work

4.1 Multi-Environment Dynamical Systems

Most data-driven approaches to identify differential equations (Pathak et al., 2018; Vlachas et al., 2018; Brunton et al., 2016; Lu et al., 2018; Raissi et al., 2018; Champion, 2019; Chang et al., 2021; Li et al., 2020b; Chen et al., 2019) assume the data to be i.i.d., *i.e.,* from one (single-environment) dynamical system. To integrate multiple dynamical environments into one shared model, several works (Yin et al., 2021; Kirchmeyer et al., 2022; Wang et al., 2021) propose to learn from several dynamical systems in a multi-task setting. These methods generalize over environments by deconstructing the learned function into a general and an environment-specific term. In contrast, our setting for continual learning prohibits access to all data at once. Another line of work considers sequential access to data with abrupt distributional shifts (Quade et al., 2018; Li et al., 2020a). While these methods focus on adaptation to the novel distribution, they do not remember the system before the shift, in contrast to our method. To date, the CL setting has not been considered for the case of DS.

4.2 Continual Learning Beyond the Stability-Plasticity Dilemma

As described by the *stability-plasticity dilemma*, CL is characterized by a trade-off between being stable enough not to forget the old and yet being plastic enough to rapidly learn the new. Attempts to continually integrate new observations without interfering with existing representations can be grouped into three strands (Parisi et al., 2019): *i)* Replay of previous data needs, relaxing the memory constraint (from our desiderata in Sec. 2) to store a buffer of representative data points (Sprechmann et al., 2018; Aljundi et al., 2019) or training a generative model from which to sample previous experience (Shin et al., 2017; Kemker & Kanan, 2018; Ostapenko et al., 2019; Wu et al., 2018; van de Ven et al., 2020; Rolnick et al., 2019a). *ii)* Expanding the model capacity to prevent overwriting previous memory. (Rusu et al., 2016; Yoon et al., 2018; Coop et al., 2013) *iii)* Ensuring that subsequent training does not interfere by regularizing task-relevant weights (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018a), or isolating parameters of different tasks into orthogonal subspaces (Duncker et al., 2020), subnetworks (Masse et al., 2018; von Oswald et al., 2021), submasks (Mallya & Lazebnik, 2018; Mallya et al., 2018; Serra et al., 2018; Ramanujan et al., 2020; Fernando et al., 2017) or sparse representations (Ahmad & Hawkins, 2016; Ororbia et al., 2020; Wortsman et al., 2020; Iyer et al., 2022).

While the plastic model components must be non-interfering for different tasks, any component shared among tasks minimizes forgetting by maximizing stability. This consideration leads to the idea of fixing weights in advance (Frankle & Carbin, 2019; Ramanujan et al., 2020; Wortsman et al., 2020). For sequential data, fixing recurrent network weights is known as Reservoir Computing (RC) (Jaeger, 2001; Maass et al., 2002), whereby a recurrent neural network is initialized randomly and kept fixed, while only a linear output layer is trained. Recent neuroscientific evidence supports evidence for RC properties in the human cortex (Enel et al., 2016; Buzsáki & Tingley, 2018). As fixed weights do not suffer from forgetting, Cossu et al. (2021a) and Kobayashi & Sugino (2019) applied RC to CL. However, in contrast to our method, both works are *i)* on discriminative tasks and *ii)* require task labels.

4.3 Sequential Generative Continual Learning

Although neuroscientific evidence shows that biological CL in the human memory is based on generating neuronal sequences, most research on CL focuses on discriminative tasks such as classification on static data (e.g., images). While this work applies CL to *generative* tasks on *sequences*, past work has investigated the application of CL for both 1) *discriminative* tasks on *sequences* (Sodhani et al., 2020; Duncker et al., 2020; Cossu et al., 2021b), (see (Cossu et al., 2021b; Ehret et al., 2021) for reviews) and 2) *generative* tasks on *static* data using generative adversarial networks (Seff et al., 2017) or variational autoencoders (Nguyen et al., 2018). Currently, the space for *generative sequential* CL is still a niche occupied by neuroscientifically inspired methods, forgoing standard backpropagation (Ororbia et al., 2020; Ororbia, 2021), or employing dendrite-inspired complex activation functions for CL (Cui et al., 2016; Iyer et al., 2022; Grewal et al., 2021). Although, like us, these works consider *generative* tasks on *sequences*, our work is novel as *i)* we apply our method to DS and *ii)* we propse a CL framework based on RC.

5 CONCLUSION

We propose a novel framework based on reservoir computing to continually learn dynamical systems from multiple environments. We compare favorably against baselines, showing promising results on several benchmark dynamical systems. With this study on reservoir computing for continual learning, we hope to inspire further research into:

i). Sequential generative continual learning, which is currently overshadowed by the easier-to-quantify classification of static images.

ii). Reservoir computing as a training scheme for continual learning of sequences.

We consider these two strands of research understudied and critically relevant in light of neuroscientific insights into human memory.

6 ETHICAL CONCERNS

A brain-like continually learning memory is a crucial step towards general artificial intelligence (AI). While general AI can have unimaginably positive consequences, it also poses an existential risk to humanity. Apart from these general concerns, we do not foresee direct adverse effects of the presented research. On a positive note, a continually adapting learning algorithm for dynamical systems can be applied to all sciences that deal with complex spatiotemporal data and propel research on the frontier of high-dimensional nonlinear systems.

REFERENCES

Subutai Ahmad and Jeff Hawkins. How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites. *ArXiv*, January 2016. URL https://arxiv.org/abs/1601.00720.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning what (not) to forget. *ECCV*, 2018a. URL https://arxiv.org/abs/1711.09601.

Rahaf Aljundi, Marcus Rohrbach, and Tinne Tuytelaars. Selfless Sequential Learning. *ArXiv*, June 2018b. URL https://arxiv.org/abs/1806.05421.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-Free Continual Learning. *CVPR*, August 2019. URL http://arxiv.org/abs/1812.03596.

Davide Bacciu, Daniele Di Sarli, Pouria Faraji, Claudio Gallicchio, and Alessio Micheli. Federated Reservoir Computing Neural Networks. *IJCNN*, July 2021. URL https://ieeexplore.ieee.org/document/9534035/.

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS*, 113(15):3932–3937, April 2016. doi: 10.1073/pnas.1517384113. URL https://www.pnas.org/doi/10.1073/pnas.1517384113.

György Buzsáki and David Tingley. Space and Time: The Hippocampus as a Sequence Generator. *Trends in Cognitive Sciences*, 22(10):853–869, October 2018. URL https://www.cell.com/trends/cognitive-sciences/abstract/S1364-6613(18)30166-9.

Kathleen P. Champion. From data to dynamics: discovering governing equations from data. *Dissertation*, 2019. URL https://digital.lib.washington.edu/researchworks/bitstream/handle/1773/44709/Champion_washington_0250E_20692.pdf.

Yifan Chang, Wenbo Li, Jian Peng, Bo Tang, Yu Kang, Yinjie Lei, Yuanmiao Gui, Qing Zhu, Yu Liu, and Haifeng Li. Reviewing continual learning from the perspective of human-level intelligence. *ArXiv*, November 2021. URL http://arxiv.org/abs/2111.11964.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. *NeurIPS*, December 2019. URL http://arxiv.org/abs/1806.07366.

Robert Coop, Aaron Mishtal, and Itamar Arel. Ensemble Learning in Fixed Expansion Layer Networks for Mitigating Catastrophic Forgetting. *IEEE Transactions on Neural Networks and Learning Systems*, 24:1623–1634, 2013. URL https://ieeexplore.ieee.org/document/6544273.

Andrea Cossu, Davide Bacciu, Antonio Carta, Claudio Gallicchio, and Vincenzo Lomonaco. Continual Learning with Echo State Networks. *ArXiv*, August 2021a. URL http://arxiv.org/abs/2105.07674.

Andrea Cossu, Antonio Carta, Vincenzo Lomonaco, and Davide Bacciu. Continual learning for recurrent neural networks: An empirical evaluation. *Neural Networks*, 143, November 2021b. URL https://www.sciencedirect.com/science/article/pii/S0893608021002847.

Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*, 28(11), November 2016. URL http://arxiv.org/abs/1512.05463.

Brian M. de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. PySINDy: A Python package for the Sparse Identification of Nonlinear Dynamics from Data. *ArXiv*, April 2020. URL http://arxiv.org/abs/2004.08424.

Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in Nonstationary Environments: A Survey. *IEEE Computational Intelligence Magazine*, 10(4), November 2015. URL https://ieeexplore.ieee.org/document/7296710.

Lea Duncker, Laura Driscoll, Krishna V. Shenoy, Maneesh Sahani, and David Sussillo. Organizing recurrent network dynamics by task-computation to enable continual learning. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/a576eafbce762079f7d1f77fca1c5cc2-Abstract.html.

Benjamin Ehret, Christian Henning, Maria R. Cervera, Alexander Meulemans, Johannes von Oswald, and Benjamin F. Grewe. Continual Learning in Recurrent Neural Networks. *ArXiv*, March 2021. URL http://arxiv.org/abs/2006.12109.

Pierre Enel, Emmanuel Procyk, René Quilodran, and Peter Ford Dominey. Reservoir Computing Properties of Neural Dynamics in Prefrontal Cortex. *PLOS Computational Biology*, 12(6), June 2016. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004967.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. *ArXiv*, January 2017. URL http://arxiv.org/abs/1701.08734.

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *ICLR*, March 2019. URL http://arxiv.org/abs/1803.03635.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4), April 1999. URL https://www.sciencedirect.com/science/article/pii/S1364661399012942.

Stefania Fresca, Andrea Manzoni, Luca Dedè, and Alfio Quarteroni. Deep learning-based reduced order models in cardiac electrophysiology. *PLoS ONE*, 15(10), October 2020. URL https://dx.plos.org/10.1371/journal.pone.0239416.

Sascha Frölich, Dimitrije Marković, and Stefan J. Kiebel. Neuronal Sequence Models for Bayesian Online Inference. *Front. Artif. Intell.*, 4, May 2021. URL https://www.frontiersin.org/articles/10.3389/frai.2021.530937/full.

Karan Grewal, Jeremy Forest, Benjamin P. Cohen, and Subutai Ahmad. Going Beyond the Point Neuron: Active Dendrites and Sparse Representations for Continual Learning. *BioRxiv*, October 2021. URL http://biorxiv.org/lookup/doi/10.1101/2021.10.25.465651.

Stephen Grossberg. Adaptive Resonance Theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Networks*, 37, January 2013. URL https://www.sciencedirect.com/science/article/pii/S0893608012002584.

Jeff Hawkins and Subutai Ahmad. Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. *Front. Neural Circuits*, 10, March 2016. URL http://journal.frontiersin.org/Article/10.3389/fncir.2016.00023/abstract.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8), 1997. URL https://direct.mit.edu/neco/article-abstract/9/8/1735/6109/Long-Short-Term-Memory.

Abhiram Iyer, Karan Grewal, Akash Velu, Lucas Oliveira Souza, Jeremy Forest, and Subutai Ahmad. Avoiding Catastrophe: Active Dendrites Enable Multi-Task Learning in Dynamic Environments. *Frontiers in Neurorobotics*, 16, 2022. ISSN 1662-5218. URL https://www.frontiersin.org/article/10.3389/fnbot.2022.846219.

Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001. URL https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nat Rev Phys*, 3(6), June 2021. URL http://www.nature.com/articles/s42254-021-00314-5.

Ronald Kemker and Christopher Kanan. FearNet: Brain-Inspired Model for Incremental Learning. *ICLR*, 2018. URL https://arxiv.org/abs/1711.10563v2.

Matthieu Kirchmeyer, Yuan Yin, Jérémie Donà, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to New Physical Systems via Context-Informed Dynamics Model. *ArXiv*, February 2022. URL http://arxiv.org/abs/2202.01889.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13), March 2017. URL https://www.pnas.org/content/114/13/3521.

Taisuke Kobayashi and Toshiki Sugino. Continual Learning Exploiting Structure of Fractal Reservoir Computing. *ICANN*, 11731, 2019. URL http://link.springer.com/10.1007/978-3-030-30493-5_4.

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2019.12.004. URL https://www.sciencedirect.com/science/article/pii/S1566253519307377.

Chunjiang Li, Zhilong Huang, Yong Wang, and Hanqing Jiang. Rapid identification of switched systems: A data-driven method in variational framework. *Science China Technological Sciences*, 64, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. *ICLR*, October 2020b. URL http://arxiv.org/abs/2010.08895.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. *NeurIPS*, June 2017. URL https://arxiv.org/abs/1706.08840.

Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2), 1963.

Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996.

Zhixin Lu, Brian R. Hunt, and Edward Ott. Attractor reconstruction by machine learning. *Chaos*, 28(6):061104, June 2018. URL https://arxiv.org/abs/1805.03362.

Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation*, 14(11), November 2002. URL https://ieeexplore.ieee.org/document/6789852.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. *CVPR*, May 2018. URL http://arxiv.org/abs/1711.05769.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. *ECCV*, March 2018. URL http://arxiv.org/abs/1801.06519.

Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *PNAS*, 115(44), October 2018. URL https://www.pnas.org/content/115/44/E10467.

Michael McCloskey and Neal J. Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In Gordon H. Bower (ed.), *Psychology of Learning and Motivation*, volume 24. Academic Press, January 1989. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Martial Mermillod, Aurélia Bugaiska, and Patrick BONIN. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, 4, 2013. URL https://www.frontiersin.org/article/10.3389/fpsyg.2013.00504.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational Continual Learning. *ArXiv*, May 2018. URL http://arxiv.org/abs/1710.10628.

Alexander Ororbia, Ankur Mali, C. Lee Giles, and Daniel Kifer. Continual Learning of Recurrent Neural Networks by Locally Aligning Distributed Representations. *IEEE Trans. Neural Netw. Learning Syst.*, 31(10), October 2020. URL https://ieeexplore.ieee.org/document/8963851/.

Alexander Ororbia, Ankur Mali, Daniel Kifer, and C. Lee Giles. Lifelong Neural Predictive Coding: Learning Cumulatively Online without Forgetting. *ArXiv*, June 2021. URL http://arxiv.org/abs/1905.10696.

Alexander G. Ororbia. Continual Competitive Memory: A Neural System for Online Task-Free Lifelong Learning. *ArXiv*, June 2021. URL http://arxiv.org/abs/2106.13300.

Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to Remember: A Synaptic Plasticity Driven Framework for Continual Learning. *CVPR*, 2019. URL https://arxiv.org/abs/1904.03137.

Chethan Pandarinath, Daniel J. O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, Matthew T. Kaufman, Stephen I. Ryu, Leigh R. Hochberg, Jaimie M. Henderson, Krishna V. Shenoy, L. F. Abbott, and David Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nat Methods*, 15(10), October 2018. URL http://www.nature.com/articles/s41592-018-0109-9.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A Review. *Neural Networks*, 113, May 2019. URL http://arxiv.org/abs/1802.07569.

Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. *Phys Rev Lett*, 120(2), January 2018. URL https://link.aps.org/accepted/10.1103/PhysRevLett.120.024102.

Markus Quade, Markus Abel, J. Nathan Kutz, and Steven L. Brunton. Sparse identification of nonlinear dynamics for rapid model recovery. *Chaos*, 28(6), June 2018. URL http://dx.doi.org/10.1063/1.5027470.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems. *ArXiv*, January 2018. URL http://arxiv.org/abs/1801.01236.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's Hidden in a Randomly Weighted Neural Network? *CVPR*, 2020. URL https://arxiv.org/abs/1911.13299.

Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, and Prabhat. Deep learning and process understanding for data-driven Earth system science. *Nature*, 566(7743), February 2019. URL http://www.nature.com/articles/s41586-019-0912-1.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience Replay for Continual Learning. *NeurIPS*, 2019a. doi: p. URL https://arxiv.org/abs/1811.11682.

David Rolnick, Priya L. Donti, Lynn H. Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, Alexandra Luccioni, Tegan Maharaj, Evan D. Sherwin, S. Karthik Mukkavilli, Konrad P. Kording, Carla Gomes, Andrew Y. Ng, Demis Hassabis, John C. Platt, Felix Creutzig, Jennifer Chayes, and Yoshua Bengio. Tackling Climate Change with Machine Learning. *ArXiv*, November 2019b. URL http://arxiv.org/abs/1906.05433.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *ArXiv*, June 2016. URL https://arxiv.org/abs/1606.04671v3.

André Röhm, D. Gauthier, and Ingo Fischer. Model-free inference of unseen attractors: Reconstructing phase space features from a single noisy trajectory using reservoir computing. *Chaos*, 2021. URL https://arxiv.org/abs/2108.04074.

Ari Seff, Alex Beatson, Daniel Suo, and Han Liu. Continual Learning in Generative Adversarial Nets. *ArXiv*, May 2017. URL https://arxiv.org/abs/1705.08395.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming Catastrophic Forgetting with Hard Attention to the Task. *PMLR*, July 2018. URL https://proceedings.mlr.press/v80/serra18a.html.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. *NeurIPS*, May 2017. URL https://arxiv.org/abs/1705.08690.

Shagun Sodhani, Sarath Chandar, and Yoshua Bengio. Toward Training Recurrent Neural Networks for Lifelong Learning. *Neural Computation*, 32:1–35, 2020.

Pablo Sprechmann, Siddhant M. Jayakumar, Jack W. Rae, Alexander Pritzel, Adrià Puigdomènech Badia, Benigno Uria, Oriol Vinyals, Demis Hassabis, Razvan Pascanu, and Charles Blundell. Memory-based Parameter Adaptation. *ICLR*, February 2018. URL https://arxiv.org/abs/1802.10542v1.

Steven H. Strogatz. *Nonlinear Dynamics And Chaos*. 1994.

Gido M. van de Ven, Hava T. Siegelmann, and Andreas S. Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nat Commun*, 11(1), December 2020. URL https://www.nature.com/articles/s41467-020-17866-2.

Pantelis R. Vlachas, Wonmin Byeon, Zhong Y. Wan, Themistoklis P. Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proc Math Phys Eng Sci*, 474(2213), May 2018. URL https://arxiv.org/abs/1802.07486.

Johannes von Oswald, Dominic Zhao, Seijin Kobayashi, Simon Schug, Massimo Caccia, Nicolas Zucchet, and João Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. *NeurIPS*, October 2021. URL http://arxiv.org/abs/2110.14402.

Rui Wang, Robin Walters, and Rose Yu. Meta-Learning Dynamics Forecasting Using Task Inference. *ArXiv*, August 2021. URL http://arxiv.org/abs/2102.10271sup.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *NeurIPS*, 2020. URL https://arxiv.org/abs/2006.14769.

Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory Replay GANs: learning to generate images from new categories without forgetting. *NeurIPS*, September 2018. URL https://arxiv.org/abs/1809.02058.

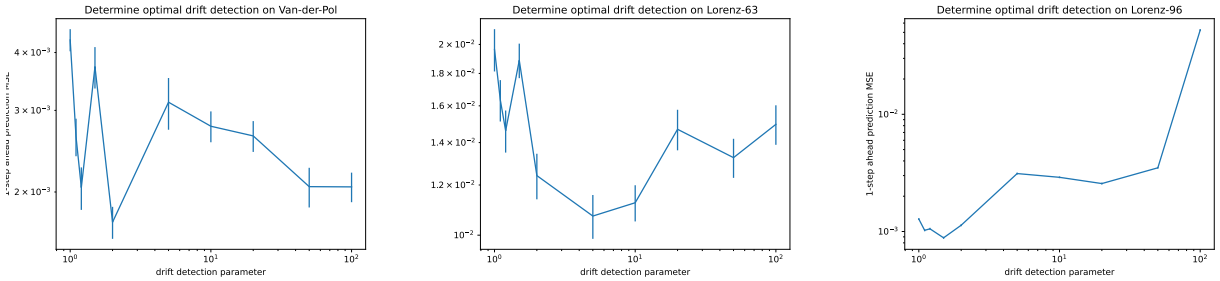
Yuan Yin, Ibrahim Ayed, Emmanuel de Bézenac, Nicolas Baskiotis, and Patrick Gallinari. LEADS: Learning Dynamical Systems that Generalize Across Environments. *NeurIPS*, June 2021. URL http://arxiv.org/abs/2106.04546.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong Learning with Dynamically Expandable Networks. *ICLR*, 2018. URL https://arxiv.org/abs/1708.01547.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. *PMLR*, July 2017. URL https://proceedings.mlr.press/v70/zenke17a.html.

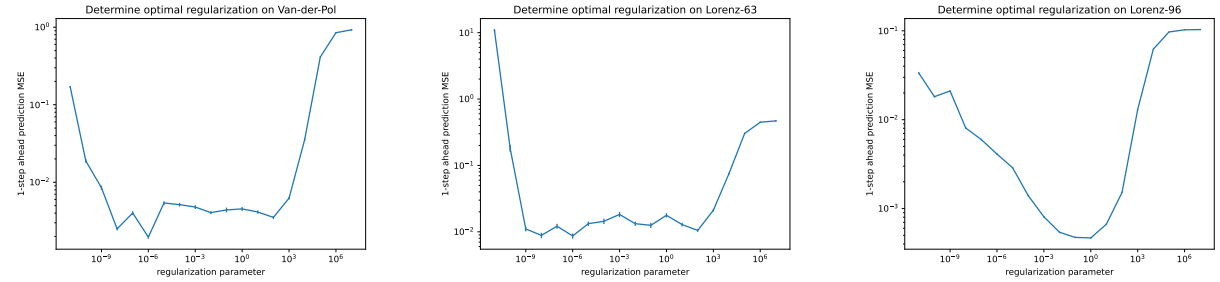
A Appendix

A.1 Hyperparameter selection and evaluation



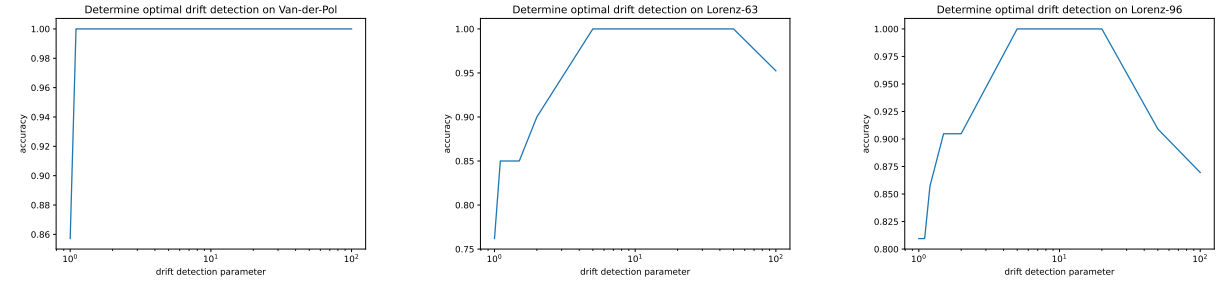
(a) (b) (c)

Figure 3: Selecting the drift detection threshold for the (a) Van-der-Pol oscillator, the (b) Lorenz-63 attractor, and the (c) Lorenz-96 model.



(a) (b) (c)

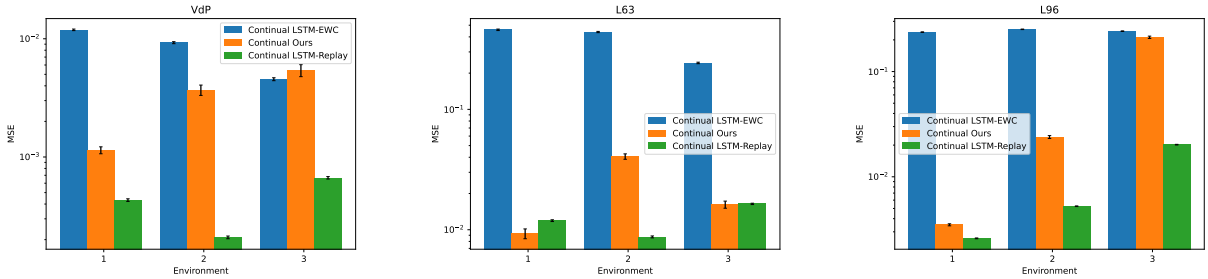
Figure 4: Selecting the regularization parameter for the (a) Van-der-Pol oscillator, the (b) Lorenz-63 attractor, and the (c) Lorenz-96 model.



(a) (b) (c)

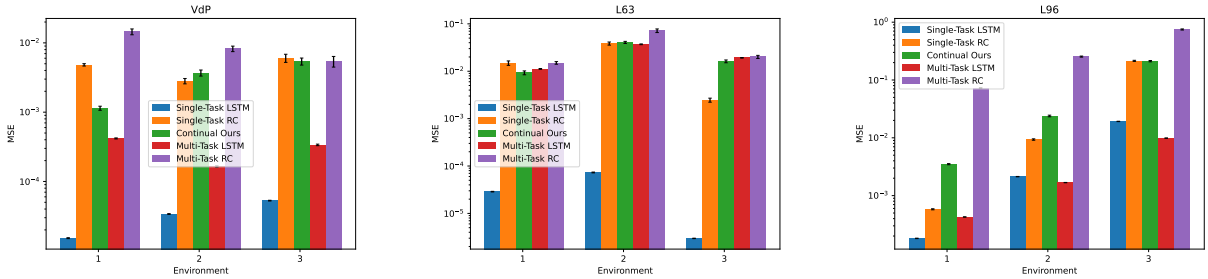
Figure 5: Evaluating the accuracy for different drift detection thresholds for the (a) Van-der-Pol oscillator, the (b) Lorenz-63 attractor, and the (c) Lorenz-96 model.

A.2 PERFORMANCE COMPARISON



(a) (b) (c)

Figure 6: Comparing our method to continual learning baselines for the (a) Van-der-Pol oscillator, the (b) Lorenz-63 attractor, and the (c) Lorenz-96 model.



(a) (b) (c)

Figure 7: Comparing our method to the single-task and multi-task settings for the (a) Van-der-Pol oscillator, the (b) Lorenz-63 attractor, and the (c) Lorenz-96 model.

A.3 MEMORY FOOTPRINT

Reservoir computing is not cheap on memory. Given a reservoir size of N_{res}, the number of prediction heads N_{heads}, and the number of dataset dimensions N_{data}, the memory footprint of our model (to store the matrices A and B) is $N_{\text{heads}} * N_{\text{res}} * N_{\text{res}} + N_{\text{heads}} * N_{\text{res}} * N_{\text{data}}$. Thus, it grows linear with the number of prediction heads, similar to a simple replay baseline. It depends on the length and amount of sequences that are stored in the buffer which method would require more memory in terms of number of parameters. However, we find the crucial difference here is that the parameters of our model do not store raw sensory data, while replay does.