Controlling Soft Robotic Arms Using Continual Learning

Francesco Piqué, Hari Teja Kalidindi, Lorenzo Fruzzetti, Cecilia Laschi, *Senior Member, IEEE*, Arianna Menciassi, *Senior Member, IEEE*, and Egidio Falotico, *Member, IEEE*

***Abstract*—Learning-based modeling and control of soft robots is advantageous due to neural network's ability to capture complex dynamical effects with low computational cost. Continual Learning techniques add further value to these methods by allowing networks to learn from continuously available data without incurring into catastrophic forgetting. In the context of soft robotic control, such capability can be exploited to design controllers able to continuously adapt to changes in robot dynamics, frequently due to material degradation or external interactions. This should be done without forgetting the control under normal working conditions which can be recovered as soon as the external interactions return to normal. In this letter elastic weight consolidation is used to continuously re-tune a neural network-based controller while changing the external loading of a soft robot. We demonstrate experimentally on a soft robot arm that this method outperforms plain stochastic gradient descent in tracking tasks, in the context of a continuously changing loading condition. We also show that the proposed control architecture can improve its performances when exposed to loading conditions already experienced. This letter represents a first step towards the introduction of continual learning methods in the soft robot control field.**

***Index Terms*—Modeling, control, and learning for soft robots, learning and adaptive systems, soft robot applications.**

I. Introduction

SOFT manipulators are typically made of a single backbone of soft and flexible material which bends and twists according to the actuation which can be either distributed, such as in pneumatic actuators, or exerted to a specific section of the backbone, such as in cable driven soft manipulators.

Unlike rigid-link robots, which have been thoroughly studied and modeled with closed form equations and with a finite number of parameters, soft and continuum robots are hard to model due to the non linearity of the materials and to the absence of joints and links that can be used to derive their kinematics in the classical sense [1]. The modeling complexity associated to soft robots makes them also difficult to control. Moreover, soft robots typically lack the richness in sensors of rigid robots, resulting in systems harder to control and making sensorization of soft robots an open research field [2]. Although analytical dynamic models of soft robot exist [3], their complexity and computational costs have led to the research of assumptions to simplify the control.

Soft robots have been most successfully and frequently controlled by employing a constant-curvature (CC) model, which assumes a constant curvature on all the length of the soft manipulator. This allows steady-state control with low computational cost [4]. By considering the soft robot backbone as multiple constant-curvature sections the piecewise constant-curvature model (PCC) is achieved [5]. In [6] the PCC model is augmented and matched to a dynamically consistent rigid robot model enabling dynamic soft robot control. However the CC and PCC models themselves are kinematic and do not consider the dynamics of a soft manipulator. Similarly, they are optimal only when the manipulator is not subject to external disturbances. These problems can be avoided by exploiting machine learning in the modeling of the soft robot. Approximating the soft robot kinematics [7] and dynamics [8] with learning approaches is a feasible and useful solution since it does not require a priori analytical knowledge of the manipulator dynamics. Supervised learning has also been proven effective in learning neural network-based controllers for soft robots [9]. In [10], an ANN is used to approximate the input-output relation of a cable driven soft robot. The obtained forward model is differentiated to compute an integral controller, with proof of robustness and stability. In [11] the authors use a machine learning approach to estimate the relationship between motor inputs and end-effector position output of a soft robot and employ trajectory optimization to perform quasi static tracking in open loop. These methods however are usually tailored to a specific task and do not take into account changes in robot dynamics or external loads. Other approaches have taken external loading conditions into account by involving the linearization of nonlinear dynamical systems

Manuscript received October 11, 2021; accepted February 8, 2022. Date of publication March 8, 2022; date of current version March 17, 2022. This letter was recommended for publication by Associate Editor Girish Krishnan and Editor K. J. Cho upon evaluation of the reviewers‘ comments. This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme through PROBOSCIS Project under Grant 863212 and in part by Human Brain Project SGA3 under Grant 945539. *Corrisponding author: Francesco Piqué.)*

Francesco Piqué, Lorenzo Fruzzetti, Arianna Menciassi, and Egidio Falotico are with the The BioRobotics Insitute, Scuola Superiore Sant'Anna, 56025 Pisa, Italy and also with the Department of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, 56127 Pisa, Italy (e-mail: f.pique@santannapisa.it; lorenzo.fruzzetti@santannapisa.it; arianna@sssup.it; e.falotico@santannapisa.it).

Hari Teja Kalidindi is with the The BioRobotics Insitute, Scuola Superiore Sant'Anna, 56025 Pisa, Italy and also with the Department of Excellence in Robotics and AI, Scuola Superiore Sant'Anna, 56127 Pisa, Italy, and also with the Institute of Communication Technologies, Electronics and Applied Mathematics, Louvain-la-Neuve, Belgium, and with the Institute of Neuroscience, Université Catholique deLouvain, Louvain-la-Neuve, Belgium (e-mail: hari.kalidindi@uclouvain.be).

Cecilia Laschi is with the Department of Mechanical Engineering, National University of Singapore Singapore 119077, Singapore (e-mail: mpeclc@nus.edu.sg).

Digital Object Identifier 10.1109/LRA.2022.3157369

by means of Koopman operator theory [12], or by reinforcement learning methods [13].

In fact, typical of soft robots is that their dynamics are subject to change, by degradation and hysteresis of the materials of their bodies or of their actuators. Moreover, especially in tasks where the robot must squeeze in tiny openings, it is subject to unknown external forces and torques which, while being mathematically tractable in rigid-link robots, introduce further complications in exact soft robot modeling and also invalidate the assumptions for CC and PCC models. Such interaction forces should also be estimated which constitutes a further challenge. Importantly, the changes in dynamics may not be permanent and the robot may return to its original working conditions. To overcome these limitations, Continual Learning (CL) techniques are of great interest since they allow network-based control methods to re-update their parameters, provided a stream of data [14]. By using CL techniques it is possible for a neural network to learn continuously by means of incrementally available data without causing interference and catastrophic forgetting of previously learnt tasks [15]. CL techniques can ensure that when the robot returns to its original working conditions, the task can still be performed with low error with minimal or no retraining. In this work, we model the dynamics of a soft robot arm with a recurrent neural network by means of supervised learning and we learn a controller on top of the dynamic model with a multi-layer perceptron (MLP). Then we exploit CL techniques to update the weights of the controller, that is used to track a circular end-effector trajectory with different external loading conditions. We have chosen to use elastic weight consolidation (EWC) in this work [16] since it is a simple CL technique which does not imply neural network growth or expansive storage of data. We show that, in a continual learning context, where the robot is exposed to different loading conditions, our method performs better in terms of task space error than a simple stochastic gradient descent (SGD) method. We also show its convenience with respect to a plain controller that is only trained once on normal operating conditions (no load is attached to the robot), and not re-tuned while using loads. Moreover, we show how the proposed control architecture is able to improve its performance when exposed to already experienced loading conditions. This solution is greatly useful in the context of soft robotics, where the robot is subject to unknown interactions and unpredictable changes in its dynamics. The paper is structured as follows: in Section II-A we describe the soft robotic platform used to test the proposed controller. The data generation process, by means of motor babbling, is described in Section II-B as well as the structure and learning procedure of the forward dynamic model and of the neural network-based controller. In Section II-C we present the experiments. In Section III the results are displayed and a statistical analysis on the performances of the control method is discussed.

II. Materials and Methods

A. Soft Robot Description

The soft robot arm I-support [17] has been used in this work as a real world platform for testing the proposed control approach

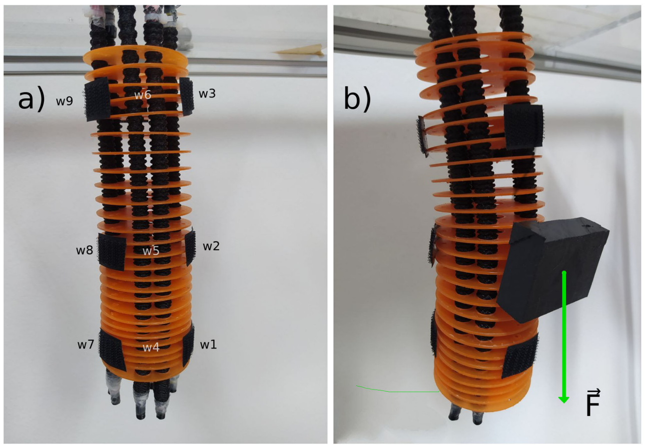

Fig. 1. (a) A single module of the I-support arm in relaxed condition. The locations where the external load will be placed are shown and labeled $w1, w2, \ldots, w9$. (b) The arm with an external load placed on location 2.

(Fig. 1). I-support is a modular soft robot intended for the assistance of elderly and disabled people in daily bathing tasks. Each module consists of three couples of extending McKibben actuators placed along its length at a 120° angle. The activation of each couple allows bending in a single direction. The actuators are kept together by equally spaced plastic discs. Only the proximal module of the I-support platform has been used in this work. The McKibben actuators are activated by pneumatic valves (Camozzi K8P) which are controlled by an Arduino Due board. The Arduino is in turn controlled by a linux PC using a serial port. The total length of the single module is 200 mm while the total weight is 160 g.

B. Model Description

In order to provide a model of the robot dynamics for offline learning of an appropriate controller, data pertaining to the mapping between the applied pressures and the resulting state of the arm must be provided. To do so, a pseudo random sequence of 12000 pressure input samples for each couple of pneumatic chambers has been generated for motor babbling. The inputs follow a random walk, meaning that each value is chosen randomly within a $\pm\epsilon$ range of the previous value. The random walk saturates at 0 bar, which is the lower range for the actuation, and at 0.83 bar, a limit imposed to prevent the damaging of the pneumatic chambers (Fig. 2(a)). To choose the ϵ range value we have considered the following trade-off: a high ϵ value would cause inputs samples to be more distant from each other, preventing the pneumatic chamber to reach the desired pressure before the next actuation sample (motor saturation). Conversely, a lower ϵ would produce a less explored workspace with the same amount of input samples [8]. In this work we have chosen $\epsilon = 20$ by trial and error. Every 100 samples the actuators are shut down (0 pressures) for 20 samples, so as to capture the dynamics of the arm in the phase from resting position to actuated position and vice versa. The inputs were sent to the soft robot for motor babbling at a rate of 10 Hz. Simultaneously, the position of the tip of the arm was measured

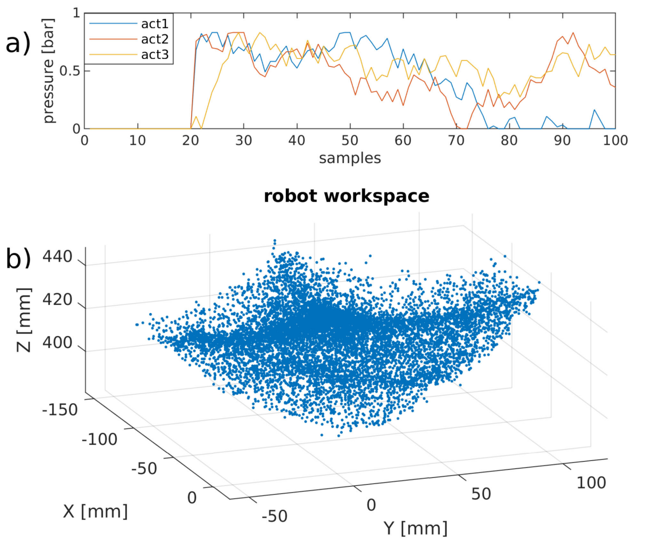

Fig. 2. (a) The first 100 samples of quasi random actuation inputs provided to the robot for motor babbling. (b) The robot workspace recorded with an NDI Aurora electromagnetic tracking system during a motor babbling session.

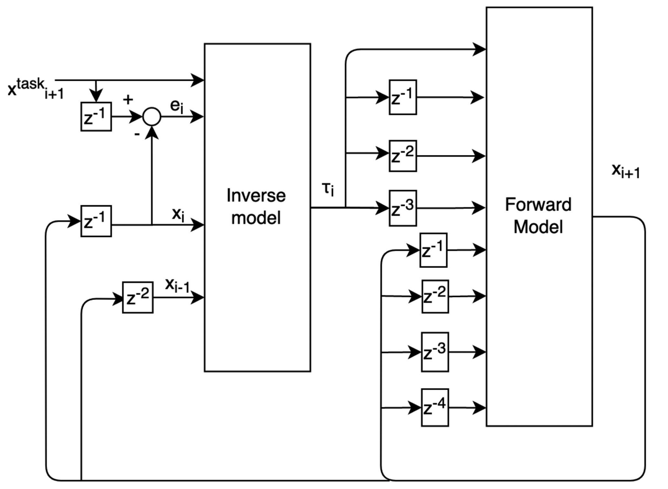

Fig. 3. The proposed adaptive control architecture. The inverse model model receives in input the current and previous end effector positions x_i, x_{i-1}, the next target x^{task}_{i+1}, the current task space error $e_i = x^{task}_i - x_i$, and outputs the actuation τ_i. The forward model takes as input a sequence of end effector positions $x_i, x_{i-1},..., x_{i-3}$, and actuation $\tau_i, \tau_{i-1},..., \tau_{i-3}$ and outputs the next end effector position x_{i+1}.

with an NDI Aurora electromagnetic tracking system (Fig. 2(b)). The obtained input/output data were split into training and test set with a 0.7 ratio for training set and a 0.3 ratio for test set, and used for supervised learning of the robot's forward model.

The ability of Recurrent Neural Networks (RNN) to learn relationships between time sequences of data has been well proven [18]. Therefore, in order to represent the robot's dynamics, an RNN layer has been used.

The model is built as the mapping between $x_i, x_{i-1},..., x_{i-3}, \tau_i, \tau_{i-1},..., \tau_{i-3}$ and x_{i+1} where $x_i, x_{i-1},..., x_{i-3}$, and x_{i+1} represent the current, previous and predicted end effector positions in Cartesian space and $\tau_i,..., \tau_{i-3}$ represent the current and previous actuator inputs. In this way the network has information of the pressures and positions in time, so as to learn the dynamics of the system. The forward model network consists of a RNN layer of 100 neurons followed by a linear layer with $tanh()$ activation. Equation (1) describes mathematically the system:

$$\begin{aligned} x_{i+i} = f_{fwd}(&x_i, x_{i-1}, x_{i-2}, x_{i-3},\\ &\tau_i, \tau_{i-1}, \tau_{i-2}, \tau_{i-3}, \theta_{fwd}) \end{aligned} \tag{1}$$

We choose the amount of delays for the feedback that result in a faster reduction of the training loss. We have experimented from one step feedback delay up to five steps feedback delay. Out of these, we observed that a memory with 4 delays results in a relatively faster reduction in model error. The model's weights θ_{fwd} are optimized on the training set by supervised learning for 1200 epochs, with a mean square error (MSE) loss function:

$$\theta_{fwd} = \arg\min_{\theta_{fwd}} L_{fwd}(\theta_{fwd})$$

$$L_{fwd}(\theta_{fwd}) = \sum_{i=0}^{N} (x^*_i - x_i)^2 \tag{2}$$

where x^*_i is the end effector position obtained by motor babbling and measured with the electromagnetic sensor, x_i is the position obtained from the forward model fed with the inputs used for motor babbling as in (1), and $N = 8400$ is the size of the training set. A learning rate of $5 * 10^{-4}$ has been set. In order to prevent overfitting, a weight decay of 10^{-4} has been introduced.

The purpose of the forward model is to serve as a decent approximation of the real robot forward dynamics mapping for the training phase of the controller. The inverse model is learned by a multilayer perceptron (MLP) with 2 hidden layers of 150 neurons with $tanh()$ activation. The inverse model is built as the mapping between x^{task}_{i+1}, x_{i-1}, x_i, e_i and τ_i where x^{task} represents the desired end effector position target in Cartesian space and $e = x^{task} - x$.

The training of the weights is performed while using the controller in the control loop shown in Fig. 3. The weight update occurs by minimization of the MSE error between the desired task x^{task}, which is an input of the inverse model, and the output of the forward model x, for 1200 epochs. The controller outputs the actuation of the soft robot:

$$\tau_i = f_{inv}\left(x_i, x_{i-1}, x^{task}_{i+1}, e_i, \theta_{inv}\right)$$
$$i = \left\lfloor \frac{t}{dt} \right\rfloor \qquad \forall t = 0 \ldots t_f \tag{3}$$

where $dt = 10\,ms$ and $t_f = 5\,s$. The vector θ_{inv} represents the weights of the inverse model which are optimized by supervised learning with the following MSE loss function:

$$L(\theta_{inv}) = \sum_i (x^{task}_i - x_i)^2$$
$$i = \left\lfloor \frac{t}{dt} \right\rfloor \qquad \forall t = 0 \ldots t_f \tag{4}$$

where x^{task} are the points in cartesian space of the desired task, and x are the end effector positions given by the forward model.

Future work may include training the inverse model directly on the real robot.

Once the inverse model is learnt we have an evaluation of the performance of the controller in simulation, with respect to the desired task. After having optimized the controller it is possible to test it on the physical platform by substituting the robot to its forward model.

After this phase, the controller is optimized to let the robot execute a single task. However if the robot is subject to an unknown interaction, which we induce here by adding external loads, the control of the desired task fails. For this reason we propose to use continual learning which modifies the loss function in the optimization of the inverse model, such that an explicit term penalizes forgetting the older tasks. The goal is to sequentially learn the modification of the dynamics that the robot undergoes and simultaneously to maintain a good performance in terms of task error, without incurring into catastrophic forgetting.

C. Soft Robot Control

First, to establish the necessity of a CL algorithm in this type of problem, we evaluate the performance of the controller with different loading conditions by keeping the parameters optimized for the unloaded robot, without retraining. In this work we consider ten loading conditions for the robotic platform. First the robot is considered without any load, then with a load of 50 g, corresponding to 31% of the robot's mass, applied to different locations of a single module of the I-support arm. The weights are placed at a 120° angle at the tip of the module, in median position, and in proximal position and are numbered from $w1$ to $w9$ (Fig. 1). We consider the performance of the robot in this round of experiments as the baseline which should be improved by the proposed method. We therefore call this non-retrained controller 'null' method. In fact, when we test the same controller on the arm with an additional load the error will be higher because the dynamics of the robot are modified by adding the external load. However, the input-output data gathered during the failed test can be used to retrain the forward model so as to learn its dynamics with the new loading condition, as in [19]. Therefore, in the proposed method the forward model network's weights are re-updated by supervised learning for 100 epochs, minimizing the MSE error between the target circular trajectory and the real robot tip position acquired during this testing phase. This retraining phase takes considerably less time than the first training of the forward model. By doing so we obtain a new forward model which is updated on the recent modification of the robot's dynamics, introduced by the external load. Similarly, the inverse model is updated on the newly trained forward model, obtaining a controller tailored on the new dynamics of the robot.

During the retraining phase of the inverse model however, an elastic weight consolidation (EWC) penalty is introduced. EWC is a method which, by imitating the synaptic plasticity of mammalian brains, allows continual learning in a supervised learning context [16]. EWC consists in adding a penalty term to the loss function which constrains the network parameters to stay in an area of low error around the optimal parameters of the previous task A, so as to prevent catastrophic forgetting while learning the next task B:

$$L(\theta) = L_b(\theta) + \sum \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2 \tag{5}$$

where L is the loss function, in this case the MSE error between the model output and the real Cartesian position, L_b is the loss only for task B, and the remaining term on the right-side is the regularization term that ensures continual learning without forgetting. F is the diagonal of the Fisher information matrix, θ is the parameter vector to be optimized, θ^*_A are the optimized parameters for the first task. The Fisher matrix weighs the importance of each neural network parameter θ in creating a memory for the older tasks, and is the crucial term in the EWC method. Ignoring the Fisher term makes the regularization term in (5) a pure L2-norm on the network parameters, and it is shown to be not very useful for continual learning without forgetting older tasks [16]. λ is the importance parameter which determines how much the memory of the old task is preserved. A higher λ ensures the best memory recall of old tasks but impairs the capability of learning new tasks while a lower λ degrades the memory of previous tasks bringing the method near to a plain Stochastic Gradient Descent (SGD) supervised learning. In this work the hyper-parameter λ has been tuned by trial and error to 10^5. The retraining is done also without EWC for comparison: in this case plain SGD is used for retraining, therefore we call this method 'naive'.

Using the proposed method the controller preserves the memory of the previously learned dynamics and avoids catastrophic forgetting. In fact, if a plain stochastic gradient descent (SGD) method is used, the network will learn the task only for that particular loading condition, forgetting all the other configurations. Moreover, by using plain SGD, successive update of the same weights on different tasks causes interference which degrades the intended performance of the network. This procedure is repeated while changing the loading conditions of the arm in random order, until all loading conditions are tested twice on the robot. Note that the forward model is retrained using the data from the most recent task, always starting from forward model 0 (meaning the model trained from the original motor babbling data), so as to have an estimate of the current robot dynamics altered by the external load. We assume in fact that a retraining with data from a control trial on top of a fully trained forward model of the unloaded robot is sufficient to have a good approximation of the new dynamics. We want to stress that the retraining of the inverse model with EWC and the one of the inverse model with SGD are performed independently, meaning that two retrained forward models are always built from data obtained from the trial done using the EWC-controller and from the one using the SGD-controller. The two retrained models are used separately for the retraining phase of the EWC and SGD controller respectively. The loading conditions are presented randomly and are named $w1, w2, \ldots, w9$ (see Fig. 1) while the conditions with the subscript $w1_2, w2_2, \ldots, w9_2$ represent that the condition is encountered for the second time. Considering also the no load condition, a total of 20 trials has been performed for each round of experiments. The flowchart for the described experiment is shown in Fig. 4 and has been repeated five times.

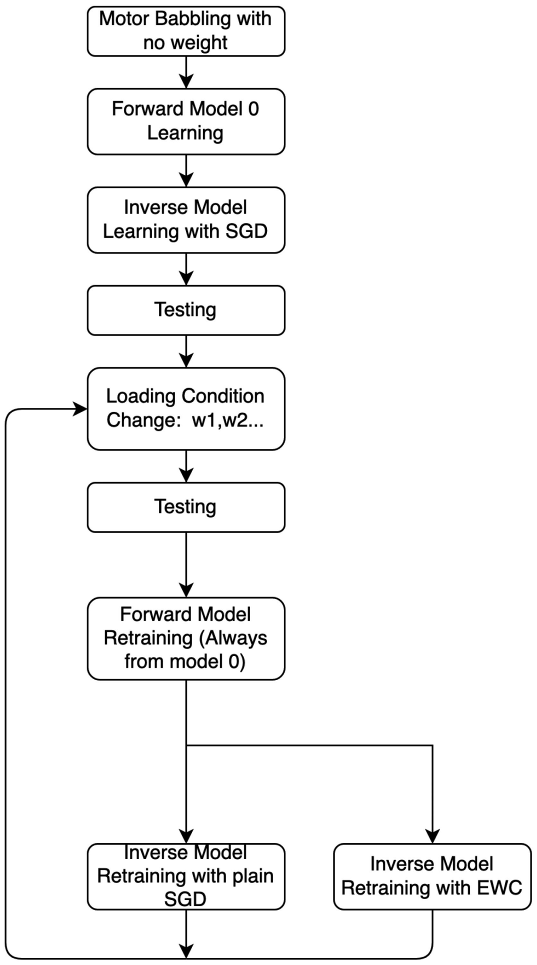
Motor Babbling with no weight
Forward Model 0 Learning
Inverse Model Learning with SGD
Testing
Loading Condition Change: w1,w2...
Testing
Forward Model Retraining (Always from model 0)
Inverse Model Retraining with plain SGD
Inverse Model Retraining with EWC

Fig. 4. Flowchart of the performed experiments.

The experiment without retraining the inverse model has been repeated five times as well, for every loading condition. The first training of the forward model takes about 68 minutes on an Intel Core$^{\text{TM}}$ i7-4790 CPU @ 3.60 GHz $\times$ 8. The training/retraining of the controller takes about 48 seconds, while the retraining of the forward model takes 1.3 seconds. This is because the retraining phase of the dynamic model updates the weights of an already trained model using a small amount of data.

III. Results and Discussion

In Fig. 5 the comparison between the mean RMS error in Cartesian space with the EWC method, the naive (SGD) method, and the null method (no retraining) is shown. The performance metric is calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i^{task} - x_i)^2}{N}} \tag{6}$$

Where $N = 50$ is the number of points in the circular trajectory, x^{task} is the desired trajectory and x is the real robot's end effector position. The bar represents the RMSE averaged on the five repetitions of the experiment, for the particular trial, while the standard deviation on the five trials is chosen as confidence interval and shown in the plot. During the first two trials (without weight and with weight in location 9, noted as w9), only the error of the null trials is shown since there is no retraining in between the trials (see Fig. 4). Therefore the EWC penalty does not come into play. The error with no load is comparable to the performance present in the literature with similar neural network based control methods [8], in this case 12.92 ± 0.9 mm, corresponding to 6.46% of the robot's length. In the immediately subsequent trial, w3, where the EWC penalty has been introduced in the retraining phase in between the trials, we can immediately appreciate a lower error with the controller retrained using EWC. This is because in this trial the controller has been trained only to minimize the task error for the previous loading condition w9, in the case of the plain SGD method, and it is asked to control a manipulator with a different loading condition. In the case of the EWC instead the controller has been also trained on the previous loading condition, but by means of the EWC penalty it also maintains the memory of the loading conditions previously encountered, in this case the no weight condition. This allows for a better performance, especially as the encountered tasks succeed one another. Moreover, the plain SGD method undergoes catastrophic forgetting and after successive retraining erases the memory of the previously learnt robot dynamics. We note that the more that the successive loading conditions are differently affecting the robot's dynamics (eg. the weight are placed on opposite locations) the more the difference in performance between SGD and EWC is in favour of EWC (See for example $w2_2$ in Fig. 5). Conversely, if the two loading conditions are similar it is possible that the two methods perform equally or that the SGD method performs slightly better. This is because the training performed on the first loading condition remains valid for the subsequent loading condition, since the dynamics have not changed significantly (See for example $w4$ in Fig. 5). However the performance in terms of task error is overall better for the EWC method as we can verify using a cumulative density function plot (Fig. 6).

In Fig. 5 the comparison between the mean RMS error with EWC and the error resulting from not retraining is shown as well. Here, since the controller in the 'null' method never undergoes a retraining phase it is always tailored on the no load condition. Therefore once any load is added the controller performance will degrade. We observe that the worst performance is achieved when the weight is placed closer to the tip of the robot, since it exerts a higher momentum.

In Fig. 6 we plotted the cumulative density function of the task errors (on the x-axis) resulting from different learning strategies across the task conditions (of loading and unloading). Note that here the task errors for each strategy (ewc, naive, and null) correspond to the test errors obtained on a new task (new loading condition), while the network was not yet retrained to reduce the error on the current loading condition and has been trained only until the preceding loading condition. We see clearly in Fig. 6 that the EWC method results in an overall lower mean error (blue line) on new loading conditions when compared with the null and naive strategies, thus justifying the need of such a controller.

See Fig. 8 for a comparison between the trajectories in task space with and without EWC. In Fig. 7 we show the best achieved

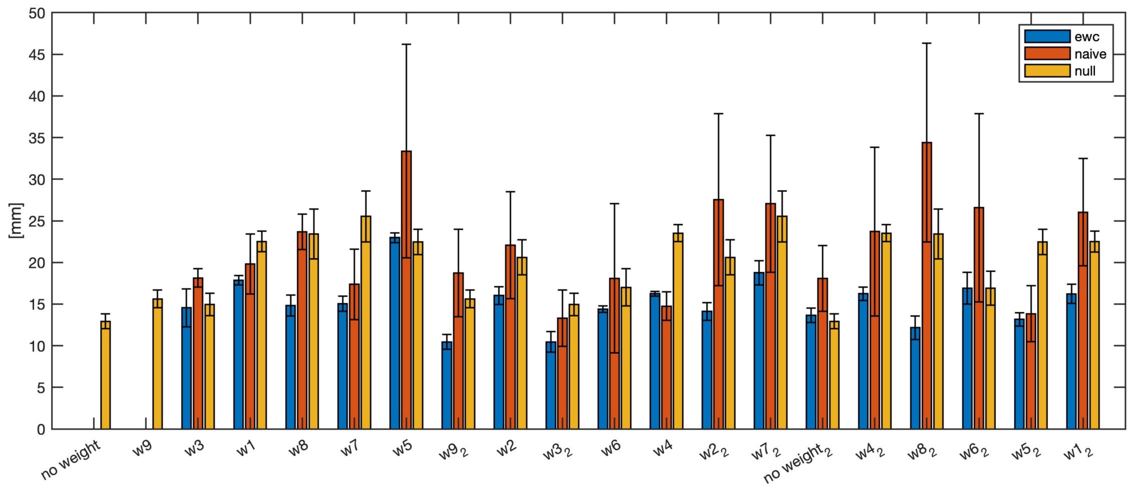

Fig. 5. The mean task space error computed for each trial shown for the EWC, SGD (naive) and null methods after training only on previous loading conditions, and before retraining on the current condition. On the x-axis the conditions $w1, w2, \ldots, w9$ are indicated, while the subscript $w1_2, w2_2, \ldots, w9_2$ indicates that the condition is encountered the second time.

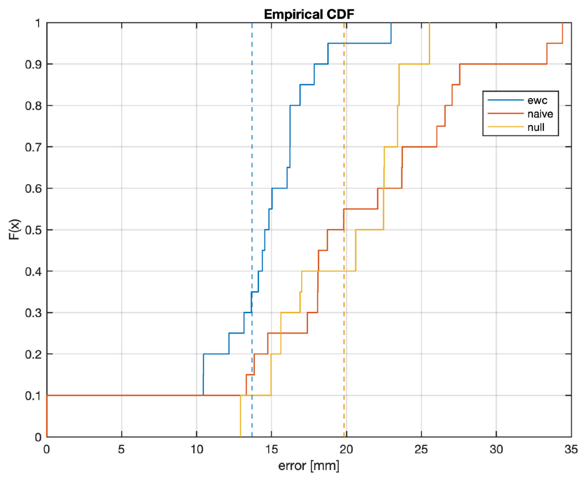

Fig. 6. The cumulative density function plots of the Cartesian kinematic errors from change in loading before network retraining for the new load. The dashed lines represent the mean of each distribution.

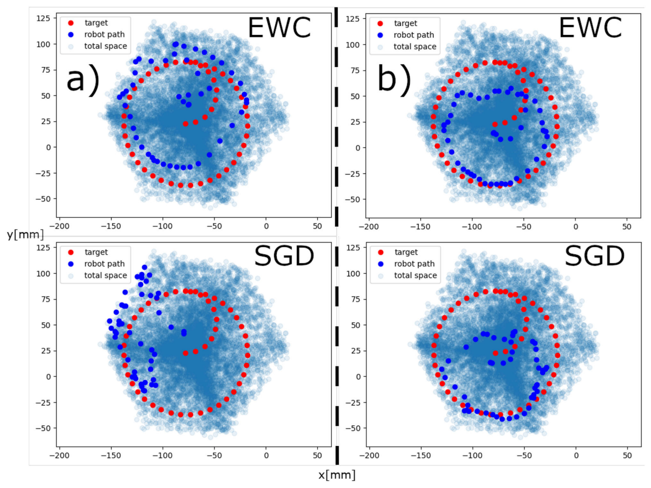

Fig. 8. (a) Task $w2_2$ and (b) task $w7_2$, (Top = EWC, bottom = SGD).

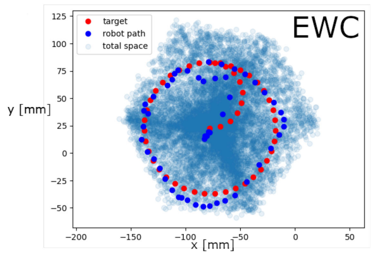

Fig. 7. The best performance achieved by the EWC method on task $w9_2$.

performance of the EWC method (10.44 ± 0.89 mm) achieved in a trial on task $w9_2$, corresponding to 5.22% of the robot's length. In Fig. 8(a) and (b) two particular cases are shown where the difference in performance between the two methods is highest in Fig. 8(a) and lowest in Fig. 8(b).

We have averaged the cartesian error across all the different weights for each of the five repetitions of the experiment to perform a one way ANOVA. First we have verified the normality of the data with a Kolmogorov-Smirnov (KS) test. The KS test shows that the data from the EWC, naive and null groups come from a normal distribution. The ANOVA test shows that there is a statistically significant difference between all groups ($p = 1.3180 * 10^{-5}$). A multiple comparison test shows that there is a statistically significant difference between the EWC method and the naive method, and between the EWC method and the null method ($p = 1.4239 * 10^{-5}$ and $p = 1.7539 * 10^{-4}$, respectively). There was no statistically significant difference between naive and null methods ($p = 0.2179$), implying that the naive controller retrained without any EWC regularization (see (5)) does not perform significantly different compared to the use of a controller that is fixed in the beginning and never retrained for changing loads. Both methods do not have memory of newly encountered tasks, therefore their error is not significantly different. We display the statistical significance of the differences between groups in Fig. 10(b).

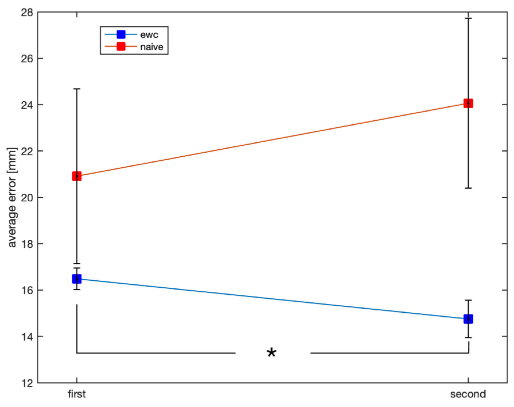

Fig. 9. The average cartesian error on the first round of weights compared to the average error for the second round of weights, for the EWC and naive methods.

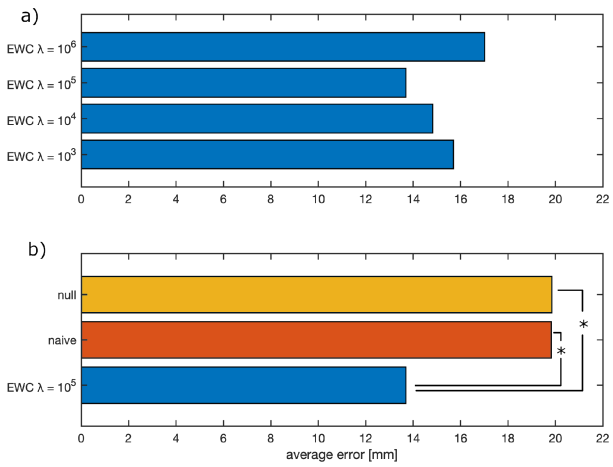

Fig. 10. Average error across all trials for the null method, the SGD (naive) method, and the EWC method with importance factor $\lambda = 10^3$, $\lambda = 10^4$, $\lambda = 10^5$, $\lambda = 10^6$. The statistical significance of the difference between the EWC method with $\lambda = 10^5$ method and the naive and null methods is shown.

Fig. 9 shows a comparison between the average errors across the loading conditions applied in the first and in the second round of experiments. We have averaged all the cartesian errors across the trials where a loading condition is experienced the first time, and where it is experienced the second time respectively. The standard deviation computed on the five repetitions of the experiment is assumed as confidence interval and shown as a error bar. We can appreciate a decrease in the mean error for the EWC method when the weights are applied the second time. We have performed a paired t-test of the hypothesis showing that the two groups of average errors come from distributions with equal means. The null hypothesis is rejected for the ewc groups ($p = 0.2924$) confirming that there is a statistically significant change in average error when the loading conditions are repeated, showing the capability of the proposed method to incrementally learn from repetitions without forgetting. The average error for the naive groups, besides being consistently higher, is not found to change in a statistically significant manner.

So far the results are for an importance factor of $\lambda = 10^5$. The previous experiments have been repeated for $\lambda = 10^6$, $\lambda = 10^4$ and $\lambda = 10^3$. In Fig. 10(a) the average error across all trials for the EWC method at varying values of lambda is shown. Since we do not appreciate a significant change in the performance of the EWC method we have chosen $\lambda = 10^5$ so as to maximize the memorization capability of the method. A higher lambda invalidates the capability of the controller to learn new tasks as we can deduce from the increasing error for $\lambda = 10^6$. In this work we applied the weights randomly without any prior knowledge of all loading conditions. In future studies, it will be interesting to compare how the EWC method performs relative to a batch training where all loading conditions are trained by considering them to be known apriori.

IV. Conclusion

In this paper we have explored the application of continual learning methods to soft robot control. The advantages of such methods are in that a continuously re-updatable controller can adapt itself to changes in the soft robot dynamics, which can be due to material degradation or to the external disturbances of unstructured environments. We show that EWC can be used to successfully re-tune the parameters of a neural network-based controller to adapt it to changing robot dynamics without forgetting the previously learnt dynamics. In our experiments with successive trials with different weights we show that our method is able to track the same circular task outperforming plain SGD, which incurs in interference. Additionally it is also capable of improving its performance when exposed to loading conditions already experienced.

This method is also scalable meaning that it could be applied to different soft manipulators of different sizes due to the generalization capabilities of neural networks. Our results put the basis for the study of more advanced continual learning methods on this type of platforms.

References

[1] T. G. Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft Robot.*, vol. 5, no. 2, pp. 149–163, 2018.

[2] G. Gerboni, A. Diodato, G. Ciuti, M. Cianchetti, and A. Menciassi, "Feedback control of soft robot actuators via commercial flex bend sensors," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 4, pp. 1881–1888, Aug. 2017.

[3] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi, "Dynamic model of a multibending soft robot arm driven by cables," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1109–1122, Oct. 2014.

[4] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 43–55, Feb. 2006.

[5] R. J. Webster III and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *Int. J. Robot. Res.*, vol. 29, no. 13, pp. 1661–1683, 2010.

[6] C. Della Santina, R. K. Katzschmann, A. Biechi, and D. Rus, "Dynamic control of soft robots interacting with the environment," in *Proc. IEEE Int. Conf. Soft Robot.*, 2018, pp. 46–53.

[7] A. Melingui, O. Lakhal, B. Daachi, J. B. Mbede, and R. Merzouki, "Adaptive neural network control of a compact bionic handling arm," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 6, pp. 2862–2875, Dec. 2015.

[8] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Learning dynamic models for open loop predictive control of soft robotic manipulators," *Bioinspiration Biomimetics*, vol. 12, no. 6, 2017, Art. no. 066003.

[9] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, “Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators,” *IEEE Trans. Robot.*, vol. 35, no. 1, pp. 124–134, Feb. 2019.

[10] G. Zheng, Y. Zhou, and M. Ju, “Robust control of a silicone soft robot using neural networks,” *ISA Trans.*, vol. 100, pp. 38–45, 2020.

[11] J. M. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros, “Soft robot control with a learned differentiable model,” in *Proc. 3 rd IEEE Int. Conf. Soft Robot.*, 2020, pp. 417–423.

[12] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, “Koopman-based control of a soft continuum manipulator under variable loading conditions,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 6852–6859, Oct. 2021.

[13] S. Satheeshbabu, N. K. Uppalapati, G. Chowdhary, and G. Krishnan, “Open loop position control of soft continuum arm using deep reinforcement learning,” in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 5133–5139.

[14] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Dıaz-Rodrıguez, “Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges,” *Inf. Fusion*, vol. 58, pp. 52–68, 2020.

[15] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural Netw.*, vol. 113, pp. 54–71, 2019.

[16] J. Kirkpatrick *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proc. Nat. Acad. Sci.*, vol. 114, no. 13, pp. 3521–3526, 2017.

[17] M. Manti, A. Pratesi, E. Falotico, M. Cianchetti, and C. Laschi, “Soft assistive robot for personal care of elderly people,” in *Proc. 6th IEEE Int. Conf. Biomed. Robot. Biomechatronics*, 2016, pp. 833–838.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[19] F. Piqué, H. T. Kalidindi, A. Menciassi, C. Laschi, and E. Falotico, “A learning-based approach for adaptive closed-loop control of a soft robotic arm,” in *Proc. I-RIM 3D Conf.*, Online, 2020, pp. 10–12.