

Learning and extrapolation of robotic skills using task-parameterized equation learner networks

Hector Perez-Villeda^{a,*}, Justus Piater^{a,b}, Matteo Saveriano^c

^a Department of Computer Science, University of Innsbruck, Innsbruck, Austria

^b Digital Science Center (DiSC), University of Innsbruck, Innsbruck, Austria

^c Department of Industrial Engineering, University of Trento, Trento, Italy

ARTICLE INFO

Article history:

Available online 12 November 2022

Keywords:

Learning from demonstration
Learning parameterized skills
Skill generalization and extrapolation
Equation learner neural networks

ABSTRACT

Imitation learning approaches achieve good generalization within the range of the training data, but tend to generate unpredictable motions when querying outside this range. We present a novel approach to imitation learning with enhanced extrapolation capabilities that exploits the so-called Equation Learner Network (EQLN). Unlike conventional approaches, EQLNs use supervised learning to fit a set of analytical expressions that allows them to extrapolate beyond the range of the training data. We augment the task demonstrations with a set of task-dependent parameters representing spatial properties of each motion and use them to train the EQLN. At run time, the features are used to query the Task-Parameterized Equation Learner Network (TP-EQLN) and generate the corresponding robot trajectory. The set of features encodes kinematic constraints of the task such as desired height or a final point to reach. We validate the results of our approach on manipulation tasks where it is important to preserve the shape of the motion in the extrapolation domain. Our approach is also compared with existing state-of-the-art approaches, in simulation and in real setups. The experimental results show that TP-EQLN can respect the constraints of the trajectory encoded in the feature parameters, even in the extrapolation domain, while preserving the overall shape of the trajectory provided in the demonstrations.

© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Imitation Learning (IL), also known as robot programming by demonstration, is a powerful approach that allows the robot to learn new skills from human guidance [1]. The main advantage of this approach is that the user can instruct a robot with no knowledge about programming. New tasks are learned using only a set of demonstrations that the user gives as input to the learning algorithm. Task demonstrations can be provided, for instance, using kinesthetic teaching [2,3] where the user physically guides the robot towards the task execution.

Task demonstrations represent successful executions of a given task in a certain range of operative conditions. However, everyday tasks often demand good generalization capabilities in order to successfully execute the same task in different conditions. As an example, consider the pouring task in Fig. 1. In this case, the user can reasonably demonstrate how to pour into a few cups of different size and, from this data, the robot should infer how to pour into unobserved cups of different sizes.

Task generalization can be achieved by enriching the demonstrated motion trajectories with further task-dependent features that characterize the task execution [4–6]. Task-dependent features, also called *task parameters*, are given as further input to regress a feasible motion for the given executive context (e.g., a cup of different size). Task-parameterized approaches for IL, as other data-driven approaches, produce effective motions inside the range of the demonstrations. However, robotic tasks often demand good generalization capabilities outside the range of training data.

In this paper, we present a novel approach to generate task-parameterized motions that use a special type of neural networks called Equation Learner Network (EQLN) [7,8]. EQLNs combine a set of different basic functions to fit the training data with analytical expressions. This is a key difference with standard learning approaches that use one type of activation function like Gaussian, sigmoid, or ReLU. As a result, the expressions fit by the EQLN have better generalization capabilities and extrapolate well beyond the range of the training data. Following the idea of task parameters, we enrich the demonstrations with extra features describing spatial properties of the motion like the size of a box or a certain height to reach. This leads to the TP-EQLN approach proposed in this work.

* Corresponding author.

E-mail addresses: Hector.Villeda@uibk.ac.at (H. Perez-Villeda), Justus.Piater@uibk.ac.at (J. Piater), Matteo.Saveriano@unitn.it (M. Saveriano).

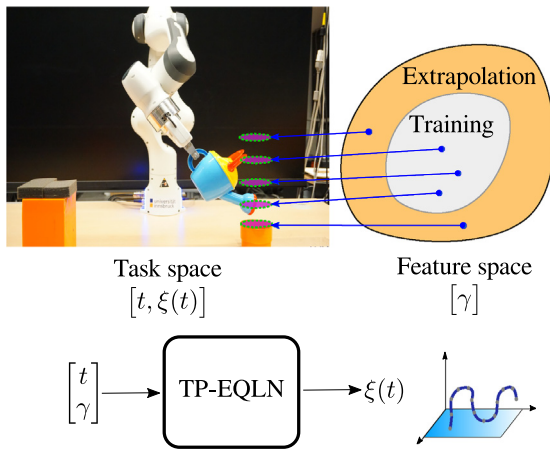


Fig. 1. Pouring task. In this example, each parameter value γ corresponds to a different size of the cup, and $\xi(t)$ corresponds to the parameterized trajectory at time t that the robot has to perform in order to pour the water.

TP-EQLNs cope with two main problems related to task-parameterized approaches: (1) They achieve good extrapolation performance over the space of task parameters while (2) preserving the overall shape of the demonstrations that defines the task by itself. In several robotic tasks, it is crucial to consider these mentioned problems in order to ensure a successful execution. We compare the performance of TP-EQLNs with existing approaches and further demonstrate the effectiveness of our approach with physics-based simulations and in a real setup on a set of manipulation tasks.

Section 2 reviews related work. Section 3 presents the proposed approach for motion extrapolation. In Section 4, we present an experimental evaluation and a comparison with existing approaches. Section 5 states the conclusions and proposes further extensions.

2. Related work

The IL formalism is built around the concept of Movement Primitive (MP) intended as a compact and flexible representation of the observed motion which allows generalization to different scenarios and reduces memory requirements [9]. A variety of approaches exist to represent MPs and each of them has distinctive features and limitations. We provide a review of existing approaches that specifically focus on task generalization.

Stable Dynamical Systems (DSs) are capable of planning converging motions from any point of the state space and have been effectively used as MP [10–13]. Among the DS-based representations, Dynamic Movement Primitives (DMPs) [14,15] are certainly the most popular and several authors have investigated their generalization capability. A DMP encodes a demonstrated trajectory into a spring-damper dynamics with a nonlinear forcing term. The forcing term acts as a disturbance on the linear dynamics and allows the DMP to generate arbitrarily complex trajectories. A set of learnable shape parameters are used to parameterize the forcing term. The shape parameters determine the overall shape of the DMP trajectories and play an important role in the generalization to new situations. Ude et al. [4] use a Gaussian Process (GP) [16] to learn and retrieve the mapping between a query point (typically a different goal) and the DMP shape parameters. Instead of learning a mapping that outputs a new set of DMP parameters, the approach in [17] embeds the task parameters directly into the forcing term. Pervaz et al. [6] follow a similar idea, but use a Gaussian Mixture Model (GMM) [18]

to represent the forcing term. Their experiments show better generalization performance compared to the approaches in [4,17]. Alternative approaches provide a probabilistic characterization of the learned MP [5,19–22]. Probabilistic MPs [19] perform generalization by statistical conditioning on the query task parameters, typically a new goal or a set of via-points. Via-point MP [22] and Kernelized MP [20,21] have better extrapolation capabilities than Probabilistic MP, but also in these approaches task parameters are new goals to reach or a set of via-points to traverse. The Task-Parameterized GMM (TP-GMM) [5] considers as task parameters the homogeneous transformations between arbitrary reference frames. By observing the human demonstrations from each of these frames the robot is able to learn the spatial relationship between start, goal, and via-points in the trajectory.

Going beyond the kinematic level, Abu-Dakka et al. [23] use DMPs to regulate the contact force during an interaction task, while Yang et al. [24] exploit DMPs to generalize a set of learned, variable-impedance robotic skills. Teaching variable impedance and force skills is also a challenge, due to the couplings between robot, environment, and human teacher [25]. Roza et al. [26] exploit teleoperation and a simplified model of the environment to extract force profiles and build training data. Kronander and Billard [27] use an artificial skin to wiggle the robot and reduce its stiffness. Electromyography (EMG) is another widely-used human-robot teaching interface, with applications in prosthetic hand interaction control [28,29], co-manipulation [30], and impedance learning [31].

Conditional Neural Processes [32] are a class of deep Neural Networks (NNs) that combines the function approximation power of NN with the data efficiency of Bayesian approaches like GP. Inspired by conditional neural processes, Seker et al. [33] developed an imitation learning framework called Conditional Neural Movement Primitive (CNMP). CNMP generates motion trajectories by sampling observations from the training data and predicting a conditional distribution over target points. Training data may include robot position, forces, and any task parameters. CNMP is promising but have limited extrapolation capabilities. A possibility to improve the extrapolation performance is to combine imitation and reinforcement learning [34]. However, we seek a solution within the imitation learning framework.

Standard regression techniques fit the training data by combining a single type of activation functions through learnable weights. For example GMM uses Gaussians while NN uses sigmoid or ReLU functions. We claim that the poor extrapolation performance of standard regression approaches arises from this design choice. Equation Learner Networks (EQLNs) [7,8] are designed to combine a set of activation functions in order to fit the training data with a richer analytical representation that possibly matches the underlying function we are training to approximate from observations. In other words, an EQLN attempts to find equations from observations. Symbolic regression approaches attempt to find such equations by searching in a certain function space by means of evolutionary algorithms [35]. This is a key difference between symbolic regression and EQLN, which uses standard backpropagation techniques. In this work, we extend the EQLN and present a novel IL approach called Task-Parameterized Equation Learner Network (TP-EQLN).

3. Methods

3.1. Problem definition

In Task-parameterized MPs, the task to be learned can be defined as a set of tuples $\Phi = \{\gamma_i, \xi(t)_i\}_i^M$, for $i = 1, \dots, M$ and $t = 0, \dots, T_i$, where M is the number of demonstrated trajectories $\xi_i(t) \in \mathbb{R}^D$ and T_i is the time duration of each trajectory. For simplicity, we assume that the demonstrations have

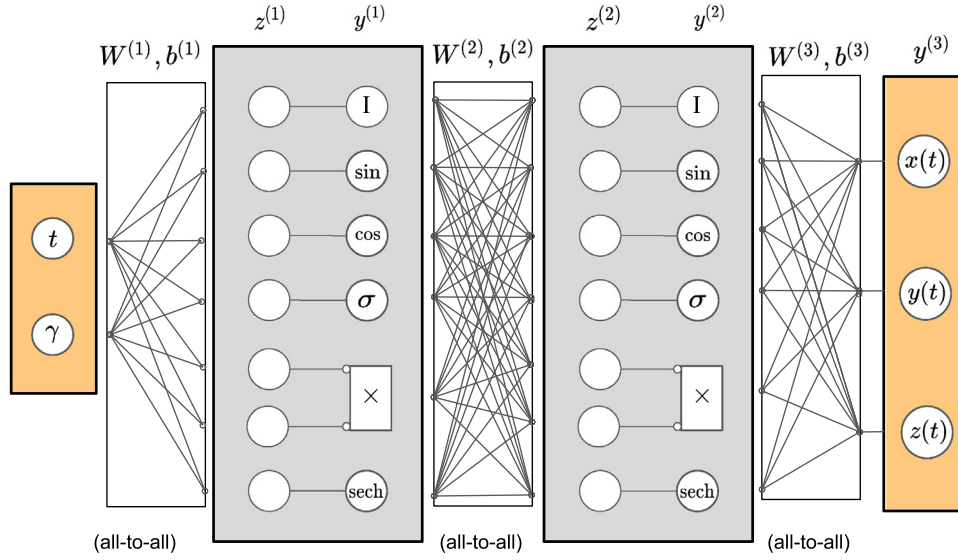


Fig. 2. TP-EQLN architecture: The network receives as run-time input the time step and the values of the feature parameters. The network outputs the pose of the end effector at each time step.

the same length, i.e., that each $T_i = T$. For each demonstrated trajectory $\xi_i(t)$, there exists a set of n feature parameters $\gamma_i \in \mathbb{R}^n$ associated with each $\xi_i(t)$. Task parameters encode information of the spatial behavior and the task's shape, e.g., a certain height to reach. As an example, in the pouring task shown in Fig. 1 the cup's size (task parameter) determines the arc length, its initial and final point as well as the orientation trajectory that the end effector of the robot has to perform in order to pour into the cup successfully. Therefore, task-parameterized MPs try to learn a mapping between the parameter space and the trajectory space.

Let us to define ζ as the whole parameter space for a given task Φ . In imitation learning, the user can only demonstrate a subset of the whole parameter space Fig. 1. When a query parameter γ is sampled from ζ , we interpolate to regress the corresponding trajectory $\xi(t)$. Traditional regression approaches perform well in interpolation. However, in real situations, we are interested in querying outside the demonstration area. In this case, we need to extrapolate to regress the corresponding trajectory. This problem is hard to solve due to the lack of training data in the extrapolation domain. The goal of this paper is to learn a mapping between task parameters and motion trajectories that achieves good inter- and extrapolation performance. As mentioned in Section 1, we build our TP-EQLN on the EQLN approach originally proposed by [7].

3.2. TP-EQLN architecture

An EQLN is a multi-layered feed-forward network with L layers; $l = \{1, \dots, L-1\}$ are hidden layers, and the last layer L is an output layer. The input $z^l \in \mathbb{R}^u$ to the l th layer is defined as

$$\begin{aligned} z^{(l)} &= W^{(l)}y^{(l-1)} + b^{(l)} \\ &= W^{(l)}f_a(z^{(l-1)}) + b^{(l)}, \end{aligned} \quad (1)$$

where $y^{(l-1)} \in \mathbb{R}^v$ is the output of the preceding layer $l-1$, $W^{(l)} \in \mathbb{R}^{u \times v}$ is a matrix of weights, and $b^{(l)} \in \mathbb{R}^u$ is a bias vector. $y^{(0)}$ and $y^{(L)}$ are defined as the network input and output respectively. W and b are the open parameters to learn. A traditional NN uses a single kind of activation function f_a , while the EQLN uses the

following activation functions:

$$\begin{aligned} f_0(z) &= I(z) = z \\ f_1(z) &= \sin(z) \\ f_2(z) &= \cos(z) \\ f_3(z) &= \sigma(z) = \frac{1}{1+e^{-z}} \\ f_4(z_0, z_1) &= z_0 \times z_1 \\ f_5(z) &= \text{sech}(z) = \frac{2}{e^z + e^{-z}} \end{aligned} \quad (2)$$

For a multidimensional input the above activation functions are applied element-wise. Note that the original formulation [7] uses only a subset of Eq. (2), namely f_0 through f_4 . The original layers are redefined by including new activation functions that contribute to better performance, as experimentally shown in the ablation study in Section 3.6. For the output $y^{(L)}$, the same configuration is kept as defined in [7], a fully-connected layer with linear activation functions. In contrast, the first layer $y^{(0)}$ is modified to include time and feature parameter(s), i.e. the time vector $\mathbf{t} = 0, \dots, 1$ is concatenated to the task parameter γ to form the input $\mathcal{I} = [\gamma, \mathbf{t}]$ of the TP-EQLN, where $\mathcal{I} \in \mathbb{R}^{n+1}$. Therefore, the output of the network $y^{(L)}$ represents the estimated trajectory $\hat{\xi}(t)$ at each time step t for a queried feature parameter \mathcal{I} . The TP-EQLN architecture used in this work is shown in Fig. 2.

3.3. Cost function

For training the TP-EQLN, we use the cost function

$$\begin{aligned} L &= \frac{1}{N} \sum_{i=1}^{|D|} \left\| \xi_i(t) - \hat{\xi}_i(t) \right\|^2 + \\ &\quad \lambda_{W,b} \sum_{l=1}^L (|W^{(l)}|_1 + |b^{(l)}|_1) + \lambda_P P_\delta \end{aligned} \quad (3)$$

where the second term is the conventional regularizer that minimizes the weights in order to achieve good generalization. $\lambda_{W,b} \geq 0$ defines a regularization constant. We additionally introduce a Penalty Epoch Term

$$\begin{aligned} P_\delta &= \sum_{i=1}^N \sum_{i=i}^D \max(\delta_{\xi_{\min}} - y_i^L(\mathcal{I}_i), 0) \\ &\quad + \max(y_i^L(\mathcal{I}_i) - \delta_{\xi_{\max}}, 0). \end{aligned} \quad (4)$$

to penalize every k epochs the predictions $\hat{\xi}_i(t)$ that are beyond a predefined threshold δ_{ξ} . $\lambda_P > 0$ defines a regularization constant for the penalty term. The introduction of a priory information in the penalty term helps to restrict the convergence space of the model by penalizing predictions $\hat{\xi}(t)$ that are beyond the boundaries defined by $[\delta_{\xi_{\min}}, \delta_{\xi_{\max}}]$. This leads to better prediction in the extrapolation domain. For robotic manipulation tasks, the restricted space $[\delta_{\xi_{\min}}, \delta_{\xi_{\max}}]$ is given by the workspace of the robot. To calculate the penalty epoch we randomly sample N inputs I without labels $\xi(t)$. The values for t are sampled from the predefined range $[0, T]$ whereas the values for γ are sampled considering only the extrapolation domain of the parameter space ζ .

Regarding the rest of the parameters in the cost function, the boundaries of the parameter space ζ depends on each task. We use $\lambda_{W,b} = 4e^{-3}$, $\lambda_P = 1e^{-6}$, $k = 50$, and $[\delta_{\xi_{\min}}, \delta_{\xi_{\max}}]$ are platform/hardware dependent and are defined as $\delta_{\xi_{\min}} = [-0.9, -0.9, -0.1]^T$, and $\delta_{\xi_{\max}} = [0.9, 0.9, 1.2]^T$.

The parameters θ of the network are updated by stochastic gradient descent with mini-batches and the Adam optimizer. We use a learning rate of $\alpha = 4e^{-3}$ for all the experiments.

3.4. Training phases

We define a total of E epochs for training the model and split them into 3 phases:

Phase 1: No regularization. This phase takes place within the epochs $0 \leq E_i \leq \frac{1}{4}E$: and the cost function to be minimized is L without regularization, i.e., $\lambda_{W,b} = 0$. The reason of considering $\lambda_{W,b} = 0$ only for a few epochs at the beginning of the training is to allow the weights to freely converge towards a first approximation.

Phase 2: Lasso-like. This phase takes place within the epochs $\frac{1}{4}E < E_i \leq \frac{3}{4}E$, and the cost function includes the regularization term, with $\lambda_{W,b} = 4e^{-3}$. The goal of this phase is to enhance the generalization and prevent overfitting.

Phase 3 Weight pruning. This phase performs magnitude-based weight pruning and gradually zeroes out model weights by following the update rule

$$W_i^{(l)}, b_i^{(l)} := \begin{cases} 0 & \text{if } |W_i^{(l)}|, |b_i^{(l)}| < \delta_W, \\ W_i^{(l)}, b_i^{(l)} & \text{otherwise.} \end{cases} \quad (5)$$

This phase takes place within the range $\frac{3}{4}E < E_i \leq T$ and leads to shorter analytic expressions that still properly describe the training data. We use $\delta_W = 0.01$

The epochs range E used in each training phase was chosen based on the values proposed in [7]. We slightly modified the last range to $\frac{3}{4}E$ with the aim of increasing sparsity.

A summary of the overall training process is presented in Algorithm 1.

3.5. Extrapolation example

In this subsection we give an overview of the TP-EQLN performance with a 1D toy example. The aim is to find the set of equations via TP-EQLN that better explains the training data set, where the samples are generated as

$$f(t, \gamma) = -0.024t^2 - 0.064\gamma^2 + 0.064t - 0.112\gamma t + 0.256\gamma - 0.5 \cos(1.2t - 0.15\gamma - 1.8) - 0.2 \sin(-1.4\gamma - 0.8) + 1.3\sigma(1.4t + 0.5\gamma) \quad (6)$$

Algorithm 1 TP-EQLN training

```

1: Take a set of demonstrations  $D = \{\gamma_i, \xi_i(t)\}$ 
2: Generate the data set  $\mathcal{I} = [\gamma, \bar{t}]$  for the input and  $\xi(t)$  for the target values.
3: Train the TP-EQLN using the loss function from Eq. (3)
4: while  $T < E$  do
5:   if  $T < \frac{1}{4}E$  then
6:     Set  $\lambda_{W,b} = 0$ 
7:   else if  $T < \frac{15}{20}E$  then
8:     Set  $\lambda_{W,b} \neq 0$ 
9:   else
10:    Set  $\lambda_{W,b} = 0$  and include the update rule from Eq. (5)
11:   end if
12: end while

```

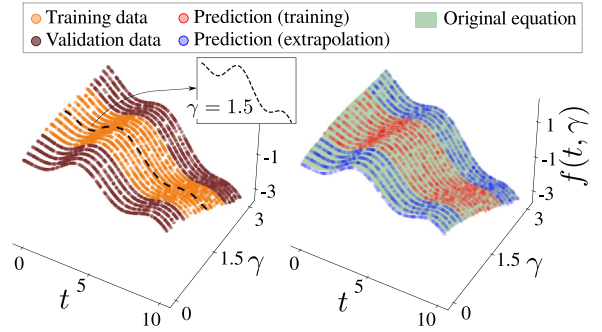


Fig. 3. A 1-D toy example that shows the extrapolation capabilities of the TP-EQLN. (Left) the training and extrapolation data sets taken from the original equation. (Right) the prediction of the TP-EQLN for each data set.

Here $f: \mathbb{R}^2 \rightarrow \mathbb{R}$, t is the time variable and γ the feature parameter. The samples are generated within the range $0 \leq \gamma \leq 3$ with 20 equally-spaced subintervals. For each value of γ , we generate 200 samples from $f(t, \gamma)$ with $t \sim U(0, 10)$. This gives a total of 4000 samples. To test the extrapolation performance, we train the model only considering 10 samples of γ generated within the range $0.789 \leq \gamma \leq 2.210$. The remaining samples are used to validate the prediction of TP-EQLN in the extrapolation domain. The training and validation data set is presented in Fig. 3 (left).

We use one hidden layer with a fully-connected layer in the output, with a batch size of 150. The equation obtained from TP-EQLN after the training process is

$$f(t, \gamma) = -0.023t^2 - 0.091\gamma^2 + 0.064t - 0.111\gamma t + 0.543\gamma - 0.349 - 0.455 \cos(-1.048\gamma - 0.064) - 0.5 \sin(-1.2t + 0.15\gamma - 3.37) + 1.299\sigma(1.402t + 0.5\gamma). \quad (7)$$

Comparing the estimated and the ground-truth equation (6), we observe that both are composed of the same trigonometric functions, and $f_5(z)$ does not play any role in either equation. The main differences of the constants are in the sin and cos functions, which could be simplified using the trigonometric identities $\sin(\theta) = \cos(\theta - \frac{\pi}{2})$ and $\cos(\theta) = \sin(\theta + \frac{\pi}{2})$, in order to obtain an expression more similar to reference equation. Small inaccuracies in the estimated constant factors give rise to mild deviations in the values of the function values that increase as the values of the arguments t and γ move away from the trained ranges. However, for the purpose addressed in this work, the error that may be produced is acceptable since the exploration area of interest is limited to the robot's workspace, which remains constant. The

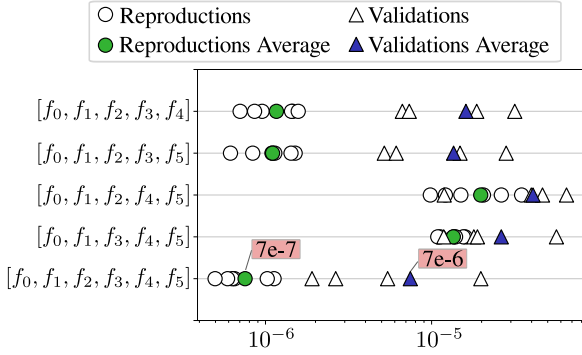


Fig. 4. Results of the ablation study.

evaluation of the equation estimated by TP-EQLN with training and validation data sets is presented in Fig. 3 (right) indicated by the red and blue data points respectively. The light-green surface represents the real equation from (6). Here, we observe that the evaluation of both training data sets remain close to the surface of the ground-truth equation.

3.6. Ablation study

To justify the use of the set of activation functions $[f_0, f_1, f_2, f_3, f_4, f_5]$ in (2) proposed for TP-EQLN, we carried out an ablation study where the performance is analyzed when some key activation functions are removed such as $[f_2, f_3, f_4, f_5]$. The study is carried out using the data set from task Φ_1 . The details of this task are presented in Section 4.1. Fig. 4 presents the results for each removed function. Each of the markers represent the MSE between the demonstrated trajectories and the predictions of each method. The circular markers refer to the training data set, whereas triangular markers refer to validation data set. The green markers represent the MSE of the demonstrations from the training data set and the blue markers represent the MSE for the extrapolation data set.

The lowest MSE obtained from this study for both the training and the extrapolation data set is highlighted in the light-red boxes, which corresponds to the use of the entire set of activation functions. Using the whole set of activation functions proposed in Section 3.2 gives the best results.

We also compared TP-EQLN against the standard EQLN to show the advantage of adding a feature parameter to the input. As for the ablation study, we used the obstacle avoidance task presented in Section 4.1. We used the same architecture for both TP-EQLN and EQLN, without considering the feature parameter in EQLN. The results are shown in Fig. 5. Fig. 5 (left) shows the trajectories used for training (green) and validation (blue). The orange trajectory is the output of the EQLN. Since it takes only the time as input, the EQLN is not able to generalize over different obstacle heights and all trajectories collapse into one. The MSE values for the training and extrapolation data set of each approach are shown in Fig. 5 (right), which validate the improvement of the TP-EQLN over the EQLN. In conclusion, it is essential to include a feature parameter in the EQLN to encode and generalize over any physical and or spatial properties of the task.

4. Experimental results

We perform six sets of experiments in different scenarios. The first four were carried out in simulation and the last two on a real Franka Emika Panda robot. In each experiment, we evaluate the inter- and extrapolation performance of the proposed

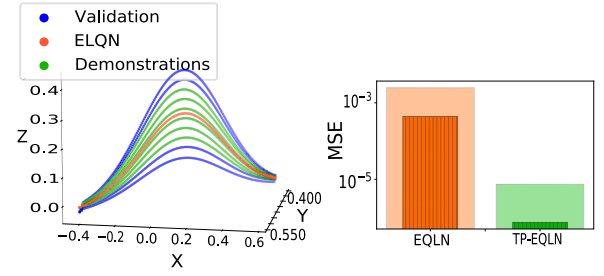


Fig. 5. Comparison between EQLN and TP-EQLN. (Left) The training and validation data set and the predicted trajectory from EQLN. (Right) The MSE comparison obtained from EQLN and TP-EQLN respectively. Solid bars correspond to the MSE from the interpolation data set whereas light bars correspond to the MSE from the extrapolation data set. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

TP-EQLN. Our approach is compared against two prominent approaches, namely CNMP and TP-GMM. The estimated trajectory in all the experiments is defined in the Cartesian space. First, a brief introduction to the six experiments is given. Afterwards, each experiment is discussed in detail in their respective subsection.

The first three experiments resemble a pick and place task where the robot picks an object from point A and places it at point B. For each experiment different task parameters are used. For the first experiment Φ_1 , we varied only the obstacle height. In the second experiment Φ_2 , we varied the height of the obstacle and the endpoint B. In the third experiment Φ_3 , we vary the height and position of the obstacle and the endpoint B. Demonstrations for Φ_1 to Φ_3 are synthetic data sampled from the parametric equation:

$$\begin{bmatrix} x_r(t) \\ y_r(t) \\ z_r(t) \end{bmatrix} = \begin{bmatrix} x_0 + t \cos \theta \\ y_0 + t \sin \theta \\ z_0 + z_d(t) \end{bmatrix}, \quad (8)$$

where the initial position is $[x_0, y_0, z_0]^T = [0.57, -0.41, 0.0]^T$ m, $\theta = 100$ deg is the rotation between the trajectory and the global frame, and t and $z_d(t)$ are vectors that parameterize the motion of each task; t varies linearly (200 samples) in the range reported in Table 1, while $z_d(t)$ is sampled from the Gaussian distribution $z_d(t) = \gamma_1 (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right)$. The parameter γ_1 varies the height of the Gaussian. The ranges of γ_1 , μ and σ are listed in Table 1.

The fourth experiment Φ_4 consists of opening boxes of different sizes by holding their cover and executing an arch trajectory until the box is entirely open. The fifth experiment Φ_5 is carried out in a real setup with a Panda robot and consists of picking and placing an object into different containers while avoiding some fixed obstacles in the environment. The last experiment Φ_6 is also carried out in a real setup and consists of pouring water into cups of varying size.

TP-EQLN hyperparameters used in each experiments are shown in Table 2. The values were selected empirically using the following rationale. Selecting a small number of hidden layers leads to better generalization in extrapolation but loses details in fitting the trajectory. On the other hand, selecting a large number of hidden layers fits the training data better but leads to highly nonlinear curves in the extrapolation domain, which impedes generalization in the extrapolation in some cases. As a general rule, one can select a high number of hidden layers if the trajectories in the extrapolation domain are expected to be highly nonlinear. Otherwise, a low number of hidden layers is preferable. We found that 1 to 3 hidden layers are enough to capture the trajectories of the performed experiments. The batch sizes are

Table 1

Feature parameters used for each task.

Task	Feature parameter		Feature Space	Data set		Motion parameters		
				Training	Extrapolation	μ	σ	t
Φ_1	$\vec{\gamma} \in \mathbb{R}$	Ranges	$0.085 \leq \gamma_1 \leq 0.4$	$0.155 \leq \gamma_1 \leq 0.33$	Remaining	0.5	0.21	$0 \leq t \leq 1$
		samples	10	6	4			
Φ_2	$\vec{\gamma} \in \mathbb{R}^2$	Ranges	$0.085 \leq \gamma_1 \leq 0.26,$ $-0.6 \leq \gamma_2 \leq -0.3$	$0.12 \leq \gamma_1 \leq 0.225,$ $-0.54 \leq \gamma_2 \leq -0.36$	Remaining	0.5	0.12	$0 \leq t \leq 0.5 - \gamma_2$
		samples	36	16	20			
Φ_3	$\vec{\gamma} \in \mathbb{R}^3$	Ranges	$0.085 \leq \gamma_1 \leq 0.365,$ $-0.6 \leq \gamma_2 \leq -0.35,$ $-0.53 \leq \gamma_2 \leq -0.41,$	$0.15 \leq \gamma_1 \leq 0.29,$ $-0.2 \leq \gamma_3 \leq 0$ $-0.15 \leq \gamma_3 \leq -0.05$	Remaining	$0.5 + \gamma_3$	0.12	$0 \leq t \leq 0.5 - \gamma_2$
		samples	125	98	27			
Φ_4	$\vec{\gamma} \in \mathbb{R}$	Ranges	$0.06 \leq \gamma \leq 0.6$	$0.24 \leq \gamma \leq 0.48$	Remaining	-	-	-
		samples	10	5	5			
Φ_5	$\vec{\gamma} \in \mathbb{R}^6$	Ranges	$\gamma_1 = 0.37,$ $\gamma_2 = -0.52,$ $\gamma_3 = 0.12$ $0.28 \leq \gamma_4 \leq 0.59,$ $0.0 \leq \gamma_5 \leq 52,$ $0.15 \leq \gamma_6 \leq 0.19$	$C_1 = [\gamma_1, \gamma_2, \gamma_3, \gamma_4 = 0.56,$ $\gamma_5 = 0.0, \gamma_6 = 0.17]$ $C_2 = [\gamma_1, \gamma_2, \gamma_3, \gamma_4 = 0.56,$ $\gamma_5 = 0.17, \gamma_6 = 0.17]$ $C_3 = [\gamma_1, \gamma_2, \gamma_3, \gamma_4 = 0.29,$ $\gamma_5 = 0.03, \gamma_6 = 0.17]$ $C_4 = [\gamma_1, \gamma_2, \gamma_3, \gamma_4 = 0.29,$ $\gamma_5 = 0.21, \gamma_6 = 0.17]$	$C_5 = [\gamma_1, \gamma_2, \gamma_3, \gamma_4 = 0.59,$ $\gamma_5 = 0.32, \gamma_6 = 0.19]$ $C_6 = [\gamma_1, \gamma_2, \gamma_3, \gamma_4 = 0.58,$ $\gamma_5 = 0.50, \gamma_6 = 0.15]$ $C_7 = [\gamma_1, \gamma_2, \gamma_3, \gamma_4 = 0.28,$ $\gamma_5 = 0.35, \gamma_6 = 0.17]$ $C_8 = [\gamma_1, \gamma_2, \gamma_3, \gamma_4 = 0.29,$ $\gamma_5 = 0.52, \gamma_6 = 0.16]$	-	-	-
		samples	8	4	4			
Φ_6	$\vec{\gamma} \in \mathbb{R}$	Ranges	$0.075 \leq \gamma \leq 0.245$	$0.12 \leq \gamma \leq 0.19$	Remaining	-	-	-
		samples	5	3	2			

Table 2

TP-EQLN hyperparameters used for each task.

Task	Layers/Batch/Epochs	Functions per layer
Φ_1	3/150/20 000	$\{l, \sin, \cos, \sigma, \times, \text{sech}\}$
Φ_2	3/150/20 000	$\{l, \sin, \cos, \sigma, \times, \text{sech}\}$
Φ_3	3/150/20 000	$\{l, \sin, \cos, \sigma, \times, \text{sech}\}$
Φ_4	1/50/5 000	$2 \times \{l, \sin, \cos, \sigma, \times, \text{sech}\}$
Φ_5	2/150/20 000	$2 \times \{l, \sin, \cos, \sigma, \times, \text{sech}\}$
Φ_6	1/100/20 000	$3 \times \{l, \sin, \cos, \sigma, \times, \text{sech}\}$

normally low for simple trajectories that do not demand large amounts of training data. The number of epochs is directly related to the size of the data set and the number of feature parameters. We found that after 20.000 epochs the prediction accuracy does not change significantly. A small number of epochs could be enough for simple trajectories that present low nonlinear variations between the feature parameters as in the case of Φ_4 .

For TP-GMM we used 30 Gaussian components. This number was chosen empirically by varying the number of components between 3 and 40 and checking the reproduction accuracy. After 30 components the estimated trajectory showed some overfitting. Moreover, we use 3 reference frames (beginning, middle, and end of the trajectory) for Φ_1 to Φ_3 , and 2 reference frames (beginning and end of the trajectory) for Φ_5 and Φ_6 . For CNMP we used 4 layers with 128 units per layer for both the encoder and decoder parts of the network, 3×10^6 training steps, and an Adam optimizer with $l_r = 10^{-4}$. For TP-EQLN and CNMP the same input data format \mathcal{I} was used for the training in all the experiments.

For tasks Φ_1 to Φ_5 , the orientation of the end-effector was not considered since it was not relevant for the tasks. Therefore, the estimated trajectory for these tasks is $\hat{\xi}(t) \in \mathbb{R}^3$ (end-effector position). The orientation (4D unit quaternion) is considered in task Φ_6 , resulting in an estimated trajectory $\hat{\xi}(t) \in \mathbb{R}^7$. The feature parameters and the training and extrapolation data set ranges of the experiments are summarized in Table 1.

4.1. Φ_1 - Variable height

This task consists of avoiding an obstacle of varying height h , with only one feature parameter $\gamma_1 = h$, Fig. 6 (top-left). The range of γ_1 and the training and extrapolation data sets are reported in Table 1. The position of the obstacle is fixed at $[0.5, 0.07, 0.0]$ m for all the demonstrations.

We show three different trajectories for each approach in Fig. 6 (top-right), two for $\gamma_1 = [0.08, 0.4]$ (extrapolation set) and one for $\gamma_1 = [0.23]$ (training set). The results show that TP-EQLN is able to correctly predict the trajectory for unobserved obstacle heights while keeping the desired shape and reaching the desired highest point of the curve. On the other hand, CNMP accurately reproduces the trajectories from the training data set but generates larger distortions in the extrapolation domain, especially for the highest point of the trajectory. This can be seen in Fig. 6 (middle), where the MSE for each feature value of each method is visualized. Here, it is observed that our method is the one that presents the smallest MSE for each value of γ_1 . Finally, Fig. 6 (bottom) presents the MSE of the highest point of the curve. This point is of importance to avoid the collision with the obstacle. The plots show that TP-EQLN outperforms both TP-GMM and CNMP in both training and extrapolation sets.

It is worth mentioning that the two frames used by the TP-GMM at the beginning/end of the motion are constant for each value of the feature parameter, while the frame in the middle varies depending on the height of the obstacle. Otherwise the estimated trajectory from TP-GMM will be constant along the z-axis. In other words, for TP-GMM we assume that the highest point to reach is always known, even for the extrapolation domain. In practice, this point is unknown and has to be estimated since it is directly correlated with the height of the obstacle. For the TP-EQLN and CNMP methods, this extra information is not provided. This assumption reveals a first clear advantage of our method over TP-GMM, i.e., it can be more practical to specify a set of high-level feature parameters rather than a reference frame.

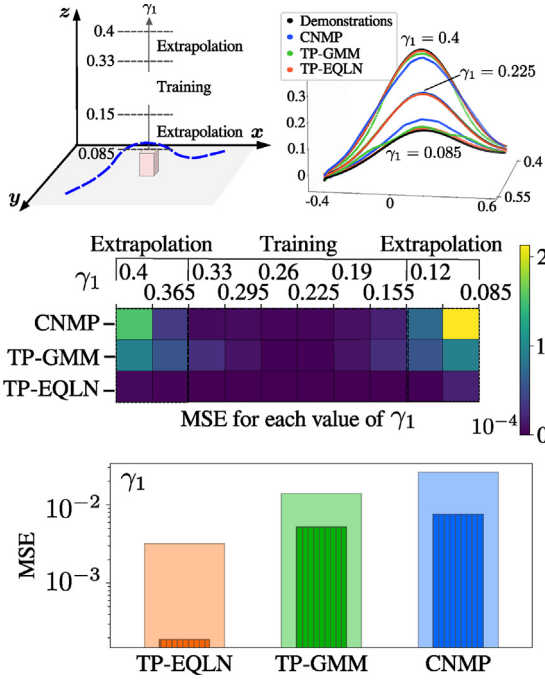


Fig. 6. Definition of the task Φ_1 and qualitative results. (Top-left) The feature space considered for the task. (Top-right) Three trajectories, one from the training set and two for the validation set, obtained for each method. (Bottom) MSE obtained for each method considering all the demonstrations in the Φ_1 task. Shaded (non-shaded) areas indicate the extrapolation (training) domain.

4.2. Φ_2 - Variable height and goal

Similarly to the previous task, we vary the height of the obstacle encoded in γ_1 . Additionally, we introduce a new task parameter γ_2 to encode the final position (goal). The feature parameter vector is formed as $\gamma = [\gamma_1, \gamma_2]$. The demonstrations were sampled from (8), with the parameters defined in Table 1. An overview of this task is depicted in Fig. 7 (top-left), while 3 trajectories retrieved with each approach are shown in Fig. 7 (top-right). In particular, we show two trajectories from the extrapolation set with $[\gamma_1, \gamma_2] = [[0.085, -0.3], [0.26, -0.6]]$ and one from the training set with $[\gamma_1, \gamma_2] = [0.155, -0.48]$. For these cases, the results show that our approach presents less distortion in the shape of the trajectory than the other methods (Fig. 7, bottom).

4.3. Φ_3 - Variable height, obstacle position, and goal

In this task we change the obstacle height, its position, and the final point of the trajectory, which are embedded in three feature parameters $\gamma = [\gamma_1, \gamma_2, \gamma_3]$. Demonstrations are generated from (8), with the parameters defined in Table 1. The position of the obstacle is also varied linearly by shifting the Gaussian function with μ , where μ depends on γ_3 . The results of this procedure are shown in Fig. 8 (top-left).

Fig. 8 (top-right) shows 3 predicted trajectories for each method with three different sets of feature parameter values, i.e. $\gamma = [0.225, -0.475, -0.1]$ (training set) and $\gamma = [[0.085, -0.35, -0.2], [0.365, -0.6, 0.0]]$ (extrapolation set). For TP-GMM, we have provided again 3 reference frames. The frame at the beginning remains constant, while the others vary depending on the desired obstacle's position and the goal of the trajectory. As done in Φ_1 and Φ_2 , the position of the frames is provided also for parameter values in the extrapolation domain.

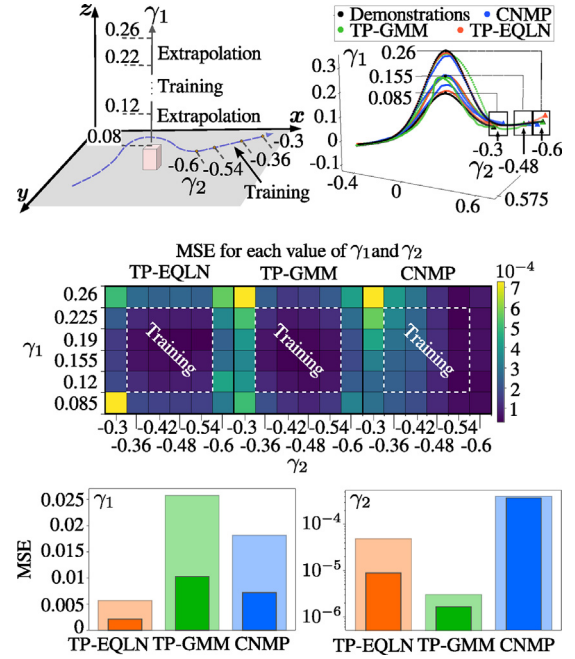


Fig. 7. Definition of the task Φ_2 and qualitative results. (Top-left) The feature space considered for the task. (Top-right) Three trajectories, one from the training set and two for the validation set, obtained for each method. (Bottom) MSE estimation for each approach for both reproduction and validation set.

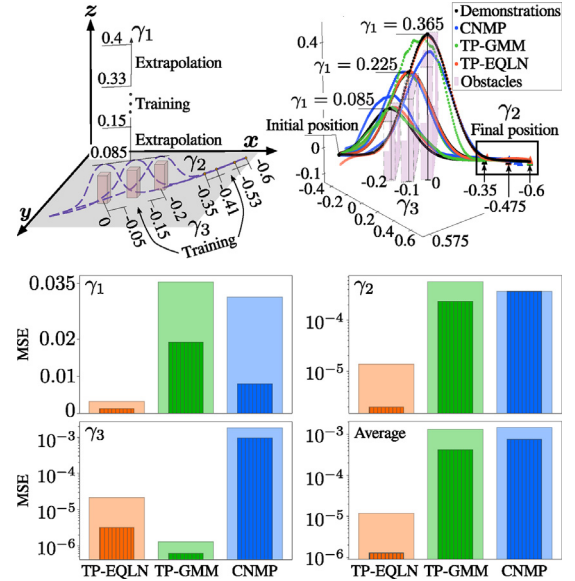


Fig. 8. Definition of the task Φ_3 and obtained results. (Top-left) The feature space considered for the task. (Top-right) Three trajectories, one from the training set and two for the validation set, obtained for each method. (Bottom) MSE obtained for each method considering all the demonstrations in the Φ_3 task. Shaded (non-shaded) areas indicate the extrapolation (training) domain.

Fig. 8 (bottom) shows the MSE for γ . The MSE for γ_1 represents error of the highest point, the MSE for γ_2 represents error of the obstacle's position, which is measured with respect to the position of the highest point of the trajectory in the plane xz , and the MSE for γ_3 represents error of the final goal position. These results show that our approach reaches closer to the highest point of the trajectory than the other methods. Regarding the MSE for γ_2 , our method is slightly outperformed by TP-GMM. The reason can be the extra information provided by the middle

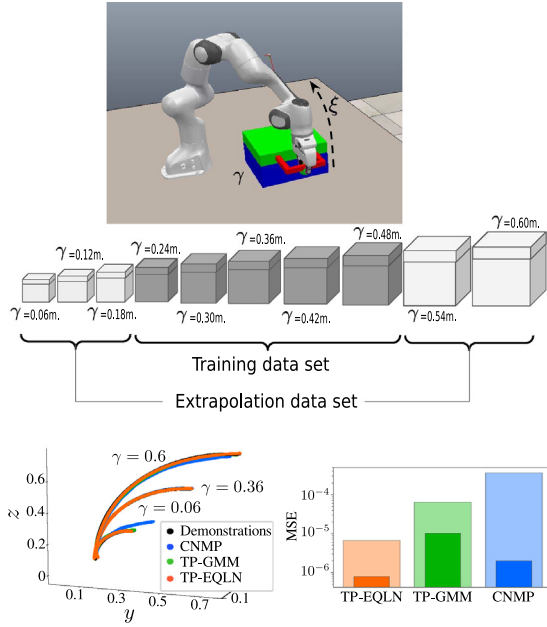


Fig. 9. Definition of Task Φ_4 . (Top) Each feature parameter encodes the trajectory to be performed depending on the box' size. (Bottom-left) Three Predicted trajectories one from the training set and two for extrapolation set. (Bottom-right) MSE obtained for each method. Shaded (non-shaded) areas indicate the extrapolation (training) domain.

frame. Regarding the MSE for γ_3 the TP-EQLN outperforms other methods. This means that our method is able to predict better the desired highest point of the trajectory at the right place to prevent collisions.

Finally, the average MSE over all the trajectories (for both training and extrapolation set) show that TP-EQLN is able to preserve the shape of the trajectory even for feature parameter values in the extrapolation domain.

The presented results show that our method predicts the spatial features linked to the features parameters accurately, while maintaining the shape of the trajectory even for feature parameters values from the extrapolation domain.

4.4. Φ_4 - Opening a box

This task consists in opening boxes of different sizes by holding their cover and executing an arch trajectory until the box is entirely open (Fig. 9, top). The box's size is given as a feature parameter γ , which affects both the trajectory's arc that must be followed as well as its endpoint. The trajectories are expressed with respect to the center of the handle and are generated as:

$$x = 0$$

$$y = (0.15 + \gamma) \left(1 - \cos \left(\frac{t\pi}{400} \right) \right), \quad (9)$$

$$z = (0.15 + \gamma) \sin \left(\frac{t\pi}{400} \right)$$

where the value of the parameters is defined in Table 1. We provide two frames for TP-GMM, one at the beginning and one at the end of the trajectory. The frame at the beginning remains constant, whereas the frame at the end varies depending on the feature parameter. The frame value is considered as extra information since in practice the end point of the trajectory is unknown in the extrapolation domain. Fig. 9 (bottom-left) shows the predicted trajectory for $\gamma = 0.06, 0.6$ (extrapolation set) and

for $\gamma = 0.36$ (training set). In the presented cases, CNMP is the one that shows larger deviations from the desired trajectories in extrapolation whereas TP-EQLN and TP-GMM predict accurately the shape of the trajectory. This can be seen in Fig. 9 (bottom-right), where the MSE for both training and extrapolation set are shown.

4.5. Φ_5 - Pick and place with obstacle avoidance

This experiment evaluates the prediction capabilities of our approach in a real pick and place task. The experiment consists of releasing an object in a container while avoiding some fixed obstacles in the environment (see Fig. 10, top-left). We used 8 containers for this experiment denoted as C_i with $i \in \{1, \dots, 8\}$. However, we demonstrate the pick and place task only for the containers $i \in \{1, 2, 3, 4\}$. The initial position of the object to be transported is fixed whereas the release point is defined right above of each container's position. The demonstrations were collected using a Panda robot, where the user demonstrated kinesthetically the trajectories, as shown in Fig. 10, top-right. Each trajectory was stored with a resolution of 700 data-points and smoothed using Cubic Splines from the SciPy library. The preservation of the trajectories' shape in this task is relevant to guarantee the avoidance of the obstacles in the setup.

The feature parameter is defined with the initial and the release point, i.e. $\gamma_i \in \mathbb{R}^6$. Although the initial point remains constant, we found out that TP-EQLN performs better by including this initial point in the feature parameter. The feature parameter space and the training and extrapolation data sets considered for this task can be found in Table 1.

For TP-GMM, we have defined two frames, at the beginning and at the end of each trajectory. In Fig. 10 (bottom-left and middle), we show a trajectory in training and 2 in extrapolation domain executed by the robot with TP-GMM, CNMP, and TP-EQLN. The 3 methods produce correct trajectories for avoiding the obstacles successfully in both data-sets. For the release point, the three methods present a gap with respect to the goal. This gap is mainly present in the z axis.

Fig. 10 (bottom-right) shows the MSE of the release point (goal) for each data set. Results show that TP-EQLN produces the lowest gap in both training and extrapolation sets. This means that TP-EQLN releases the object in the container successfully, whereas the TP-GMM and CNMP release the object outside of the container. The shape of the trajectory is also important for executing the task successfully. In Fig. 10 (bottom-right), we show the average MSE for each data set which tell us how well the shape of the trajectory is preserved. The results show that TP-EQLN significantly reduces the total error in reproducing the demonstrations. Fig. 10 (middle-right) shows some snapshots of the executed trajectory for the Container C_8 using TP-EQLN.

4.6. Φ_6 - Pouring into different cups

This experiment demonstrates the generalization capabilities of TP-EQLN in a pouring task (see Fig. 11, top-left), where both position and orientation of the end-effector have to be controlled. In particular, the position trajectory is mainly affected by the size of the cup and therefore varies in the xy plane. The orientation plays an important role as it gives the correct inclination of the end-effector during the pouring, but it does not present a significant variation with respect to the size of the cup. Also the correct coupling between position and orientation trajectories is important for the successful execution of the task. The estimated trajectory is $\xi \in \mathbb{R}^7$, where $[x, y, z]$ defines the position of the end-effector in Cartesian space and $[q_x, q_y, q_z, q_w]$ the orientation as a unit quaternion. The initial position of the cup is constant

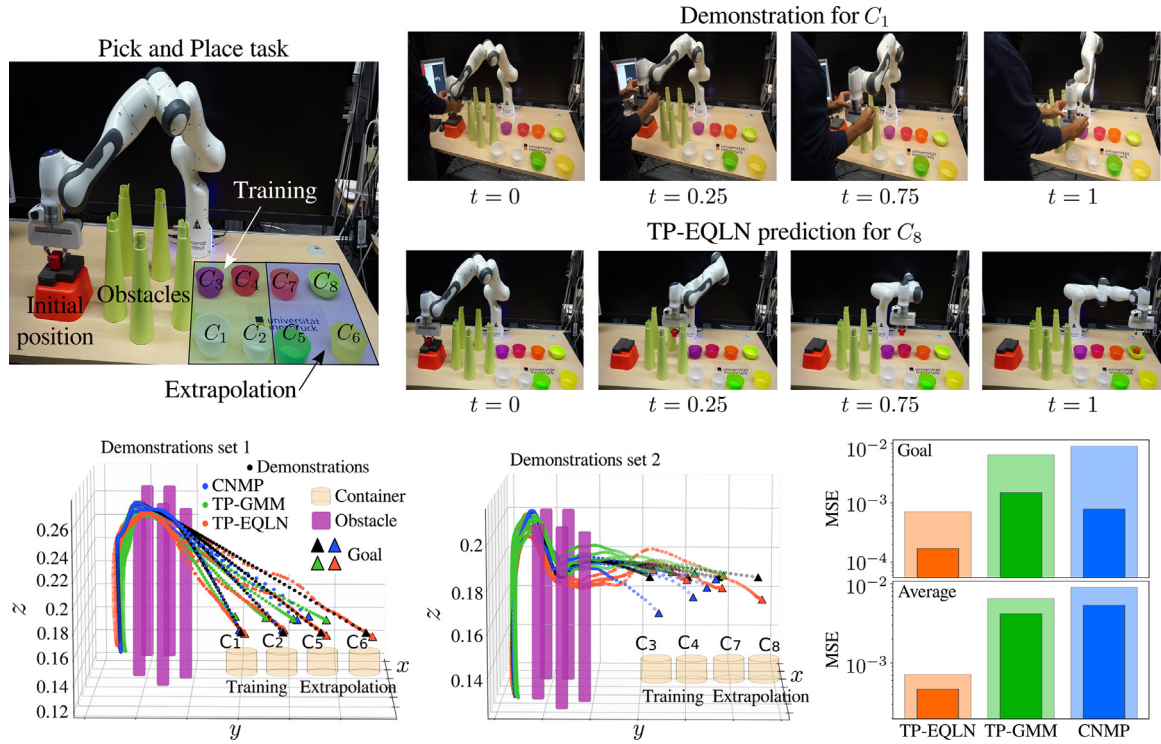


Fig. 10. Definition of task Φ_5 and qualitative results. (Top-left) The experiment setup. (Bottom-left) Trajectories obtained from each method in the training (containers C_1, C_2) and extrapolation data sets (containers C_5, C_6). (Bottom-center) Trajectories obtained from each method in the training (containers C_3, C_4) and extrapolation data sets (containers C_7, C_8). (Bottom-right) MSE estimation for each approach. Shaded (non-shaded) areas indicate the extrapolation (training) domain.

for all the demonstrations. The variation of the feature parameter $\gamma \in \mathbb{R}$ is linked to the cup's size and the ranges are listed in Table 1.

The demonstrations were taken with a Panda robot where the user demonstrated kinesthetically the trajectories as shown in the snapshots of Fig. 10 (top-right). Each trajectory was stored with a resolution of 700 data points. We estimate position and orientation using separate TP-EQLNs or CNMP that share the feature parameter as input. Both estimated trajectories are aligned in time to form the estimated trajectory $\hat{\xi} \in \mathbb{R}^7$.

As done in the other task for comparison purposes, we provide to TP-GMM information about the unknown (final) point by specifying the value of the end frame also in extrapolation.

In Fig. 11 (bottom-left), we show the predicted trajectories for position and orientation for 3 different cup sizes, i.e. $\gamma = 0.16$ (training set) and $\gamma = [0.075, 0.245]$ (extrapolation set). For this task, there are three relevant points of the trajectory (see the markers in Fig. 11, bottom-left). These points are used as a reference to compare the TP-EQLN against TP-GMM and CNMP. The first point is the initial pose of the robot. All the approaches manage to keep the initial pose for different feature parameters. The second point represents the approach position between the tip of the watering can and the cup. The last point defines the pose where the robot ends the pouring process. The last one is the most important considering that a deviation in this point may cause the liquid to spill outside the cup.

In Fig. 11 (bottom-right), we show the MSE value for each key point in both training and extrapolation domains. Regarding the initial point, our method is the one that presents the lowest MSE. A large MSE in the initial point of the trajectory would generate large control actions and a non-smooth start of the robot motion. Regarding the approach and final point, our method is slightly outperformed by TP-GMM in the training domain. However, in the extrapolation domain, our method produces better results. In Fig. 11 (bottom-right), we show the average MSE per data set.

In the training domain, TP-EQLN and CNMP perform similarly and are slightly outperformed by TP-GMM. TP-EQLN outperforms other approaches in extrapolation.

5. Conclusions and future work

We presented the Task-Parameterized Equation Learner Network (TP-EQLN), a novel approach to imitation learning with enhanced extrapolation capabilities for tasks where maintaining the shape of the trajectory is important. The key idea of TP-EQLN is to combine different types of activation functions in order to find an analytical expression that fits the training data. This leads to a better approximation of the desired motion in both inter- and extrapolation domains. We enrich our network with task-dependent parameters and use them alongside a time vector to query the desired path. We showed it is possible to encode physical or spatial properties of the task in this feature parameter and to generalize the task over the feature parameter space.

Our TP-EQLN was empirically evaluated and compared against two prominent existing approaches in a set of tasks of increasing complexity. The obtained results show that, in most of the cases, TP-EQLN outperforms state-of-the-art approaches, especially in extrapolation, and most importantly, preserves the shape of the trajectory. Whereas CNMP also use a feature parameter to encode some properties of the task, it is not possible to generalize beyond the data distribution of the training data since this method is based on probability distributions. On the other hand, TP-GMM require the explicit definition of frames provided by the user to be able to adapt the trajectory for any change in the environment. TP-EQLN encode those changes in the environment using high-level feature parameters. With this it is possible to generalize the task beyond the data distribution of the feature parameter used for training. The definition of high-level feature parameters allows us to define and generalize the tasks more intuitively.

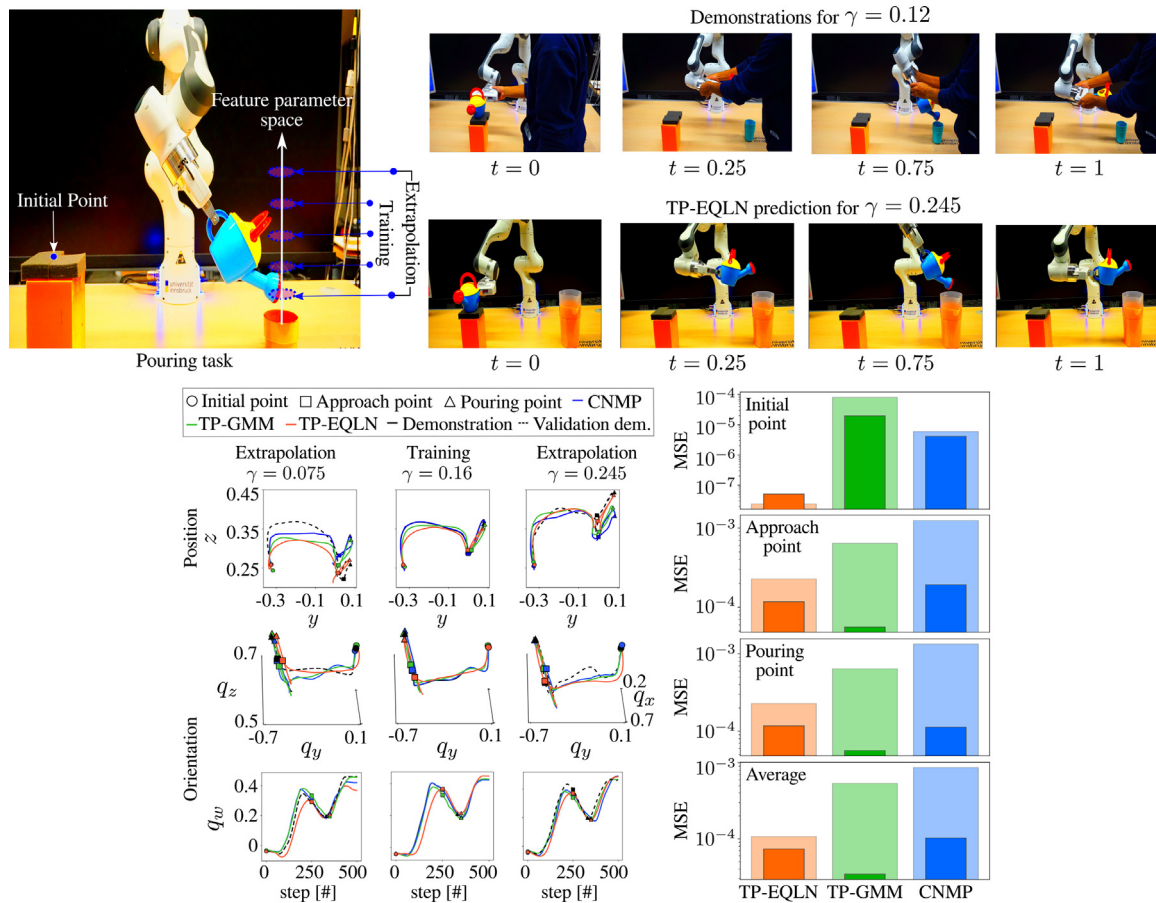


Fig. 11. Definition of the task Φ_6 and results. (Top-left) the experiment setup. (Top-right) Snapshots of a task demonstration and reproduction (in extrapolation) using TP-EQLN. (Bottom-left) Three trajectories for position and orientation obtained for each method, where we highlight the most important (key) points of each demonstration. (Bottom-right) MSE obtained for each key points and considering the whole trajectory for training/extrapolation (non-shaded/shaded areas) sets.

Our future research will follow different directions. We will explore the possibility to combine TP-EQLN with control techniques to ensure convergence to a given target while keeping the overall shape of the motion. We also plan to endow TP-EQLN with stochastic information to characterize the variability and the uncertainty in the demonstrations. Finally, we will investigate the possibility to use more abstract concepts as task parameters, like the property of containing liquids, in combination with a task ontology to generalize learned skills across different objects with similar properties.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research has received funding from the Austrian Research Foundation (Euregio IPN 86-N30, OLIVER). The authors thank M. Yunus Seker and E. Ugur for sharing the CNMP code, which is now publicly available [36].

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.robot.2022.104309>.

References

- [1] A. Billard, S. Calinon, R. Dillmann, Learning from humans, in: B. Siciliano, O. Khatib (Eds.), *Handbook of Robotics*, 2nd Edition, Springer, Secaucus, NJ, USA, 2016, pp. 1995–2014.
- [2] M. Saveriano, S. An, D. Lee, Incremental kinesthetic teaching of end-effector and null-space motion primitives, in: *IEEE International Conference on Robotics and Automation*, 2015, pp. 3570–3575.
- [3] R. Caccavale, M. Saveriano, A. Finzi, D. Lee, Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction, *Auton. Robots* 43 (6) (2019) 1291–1307.
- [4] A. Ude, A. Gams, T. Asfour, J. Morimoto, Task-specific generalization of discrete and periodic dynamic movement primitives, *IEEE Trans. Robot.* 26 (5) (2010) 800–815.
- [5] S. Calinon, A tutorial on task-parameterized movement learning and retrieval, *Intell. Serv. Robot.* 9 (1) (2016) 1–29.
- [6] A. Pervez, D. Lee, Learning task-parameterized dynamic movement primitives using mixture of GMMs, *Intell. Serv. Robot.* 11 (1) (2018) 61–78.
- [7] G. Martius, C.H. Lampert, Extrapolation and learning equations, 2016, CoRR abs/1610.02995, arXiv:1610.02995.
- [8] S. Sahoo, C. Lampert, G. Martius, Learning equations for extrapolation and control, in: *International Conference on Machine Learning*, 2018, pp. 4442–4450.
- [9] S. Calinon, F. Guenter, A. Billard, On learning, representing, and generalizing a task in a humanoid robot, *IEEE Trans. Syst. Man Cybern. B* 37 (2) (2007) 286–298.
- [10] S.M. Khansari-Zadeh, A. Billard, Learning stable non-linear dynamical systems with Gaussian mixture models, *Trans. Robot.* 27 (5) (2011) 943–957.

- [11] N. Perrin, P. Schlehuter-Caissier, Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems, *Systems Control Lett.* 96 (2016) 51–59.
- [12] M. Saveriano, D. Lee, Incremental skill learning of stable dynamical systems, in: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2018, pp. 6574–6581.
- [13] M. Saveriano, An energy-based approach to ensure the stability of learned dynamical systems, in: *IEEE International Conference on Robotics and Automation*, IEEE, 2020, pp. 4407–4413.
- [14] A.J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors, *Neural Comput.* 25 (2) (2013) 328–373.
- [15] M. Saveriano, F.J. Abu-Dakka, A. Kramberger, L. Peternel, Dynamic movement primitives in robotics: A tutorial survey, 2021, *CoRR* abs/2102.03861.
- [16] C.E. Rasmussen, C.K.I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, Cambridge, Massachusetts, 2006.
- [17] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, O. Sigaud, Learning compact parameterized skills with a single regression, in: *IEEE-RAS International Conference on Humanoid Robots*, Atlanta, GA, USA, 2013, pp. 417–422.
- [18] D.A. Cohn, Z. Ghahramani, M.I. Jordan, Active learning with statistical models, *J. Artificial Intelligence Res.* 4 (1996) 129–145.
- [19] A. Paraschos, C. Daniel, J. Peters, G. Neumann, Probabilistic movement primitives, in: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger (Eds.), *Advances in Neural Information Processing Systems* 26, Curran Associates, Inc., Lake Tahoe, Nevada, US, 2013, pp. 2616–2624.
- [20] Y. Huang, L. Rozo, J. Silvério, D.G. Caldwell, Kernelized movement primitives, *Int. J. Robot. Res.* 38 (7) (2019) 833–852.
- [21] Y. Huang, F.J. Abu-Dakka, J. Silvério, D.G. Caldwell, Toward orientation learning and adaptation in cartesian space, *IEEE Trans. Robot.* 37 (1) (2021) 82–98.
- [22] Y. Zhou, J. Gao, T. Asfour, Learning via-point movement primitives with inter- and extrapolation capabilities, in: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Macau, China, 2019, pp. 4301–4308.
- [23] F.J. Abu-Dakka, B. Nemec, J.A. Jørgensen, T.R. Savarimuthu, N. Krüger, A. Ude, Adaptation of manipulation skills in physical contact with the environment to reference force profiles, *Auton. Robots* 39 (2) (2015) 199–217.
- [24] C. Yang, C. Zeng, C. Fang, W. He, Z. Li, A DMPs-based framework for robot learning and generalization of humanlike variable impedance skills, *IEEE/ASME Trans. Mechatronics* 23 (3) (2018) 1193–1203.
- [25] F.J. Abu-Dakka, M. Saveriano, Variable impedance control and learning—a review, *Front. Robot. AI* 7 (2020) 590681.
- [26] L. Rozo, S. Calinon, D.G. Caldwell, P. Jiménez, C. Torras, Learning physical collaborative robot behaviors from human demonstrations, *IEEE Trans. Robot.* 32 (3) (2016) 513–527.
- [27] K. Kronander, A. Billard, Learning compliant manipulation through kinesthetic and tactile human-robot interaction, *IEEE Trans. Haptics* 7 (3) (2013) 367–380.
- [28] N. Jaquier, R. Haschke, S. Calinon, Tensor-variate mixture of experts for proportional myographic control of a robotic hand, *Robot. Auton. Syst.* 142 (2021) 103812.
- [29] B. Fang, C. Wang, F. Sun, Z. Chen, J. Shan, H. Liu, W. Ding, W. Liang, Simultaneous sEMG recognition of gestures and force levels for interaction with prosthetic hand, *IEEE Trans. Neural Syst. Rehabil. Eng.* 30 (2022) 2426–2436.
- [30] L. Peternel, N. Tsarakakis, D. Caldwell, A. Ajoudani, Robot adaptation to human physical fatigue in human-robot co-manipulation, *Auton. Robots* 42 (5) (2018) 1011–1021.
- [31] C. Zeng, C. Yang, H. Cheng, Y. Li, S.-L. Dai, Simultaneously encoding movement and sEMG-based stiffness for robotic skill learning, *IEEE Trans. Ind. Inform.* 17 (2) (2021) 1244–1252.
- [32] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y.W. Teh, D. Rezende, S.A. Eslami, Conditional neural processes, in: *International Conference on Machine Learning*, 2018, pp. 1704–1713.
- [33] M.Y. Seker, M. Imre, J.H. Piater, E. Ugur, Conditional neural movement primitives, in: *Robotics: Science and Systems*, 2019.
- [34] M.T. Akbulut, E. Oztup, M.Y. Seker, H. Xue, A.E. Tekden, E. Ugur, ACNMP: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing, in: *Conference on Robot Learning*, 2020.
- [35] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, *Science* 324 (5923) (2009) 81–85.
- [36] M. Yunus Seker, Conditional Neural Movement Primitives (CNMP) - Source code, URL <https://github.com/myunusseker/CNMP>.



Hector Pérez Villeda received the M.Sc. degree in Robotics and Advanced Manufacturing at CINVESTAV, Mexico in 2015. In the same year he co-founded Introid Inc., a company focused on A.I. and computer vision. In parallel, he continued collaborating with researchers from CINVESTAV, where he authored and co-authored different conference and journal papers focused on multi-robot coordination systems. In 2019 he was a research engineer at the Institute of Plasma and Nuclear Fusion (IPFN) in Lisboa, Portugal.

He is currently a Ph.D. candidate at the University of Innsbruck, Austria, and a member of the Intelligent and Interactive Systems group in the computer science department. His research is focused on Imitation learning, Programming by demonstrations, Machine learning for robot tasks generalization and optimization.



Justus Piater is a professor of computer science at the University of Innsbruck, Austria, where he leads the Intelligent and Interactive Systems group. He holds a M.Sc. degree from the University of Magdeburg, Germany, and M.Sc. and Ph.D. degrees from the University of Massachusetts Amherst, USA, all in computer science. Before joining the University of Innsbruck in 2010, he was a visiting researcher at the Max Planck Institute for Biological Cybernetics in Tübingen, Germany, a professor of computer science at the University of Liège, Belgium, and a Marie-Curie research fellow

at GRAVIR-IMAG, INRIA Rhône-Alpes, France. His research interests focus on learning and inference in sensorimotor systems. He has published more than 200 papers in international journals and conferences, several of which have received best-paper awards. Currently he serves as the founding director of the interdisciplinary Digital Science Center of the University of Innsbruck.



Matteo Saveriano received his B.Sc. and M.Sc. degree in automatic control engineering from University of Naples, Italy, in 2008 and 2011, respectively. He received his Ph.D. from the Technical University of Munich in 2017. Currently, he is an assistant professor at the Department of Industrial Engineering (DII), University of Trento, Italy. Previously, he was an assistant professor at the University of Innsbruck and a post-doctoral researcher at the German Aerospace Center (DLR). He is an Associate Editor for RA-L. His research activities include robot learning, human-robot interaction, understanding and interpreting human activities. Webpage: <https://matteosaveriano.weebly.com/>.