Tensor Based Knowledge Transfer Across Skill Categories for Robot Control

Chenyang Zhao[*] Univ. of Edinburgh c.zhao@ed.ac.uk

Timothy M. Hospedales[*] Univ. of Edinburgh t.hospedales@ed.ac.uk

Freek Stulp[†] German Aerospace Center Freek.Stulp@dlr.de

Olivier Sigaud[‡] UPMC olivier.sigaud@isir.upmc.fr

Abstract

Advances in hardware and learning for control are enabling robots to perform increasingly dextrous and dynamic control tasks. These skills typically require a prohibitive amount of exploration for reinforcement learning, and so are commonly achieved by imitation learning from manual demonstration. The costly non-scalable nature of manual demonstration has motivated work into skill generalisation, e.g., through contextual policies and options. Despite good results, existing work along these lines is limited to generalising across variants of one skill such as throwing an object to different locations. In this paper we go significantly further and investigate generalisation across qualitatively different classes of control skills. In particular, we introduce a class of neural network controllers that can realise four distinct skill classes: reaching, object throwing, casting, and ball-in-cup. By factorising the weights of the neural network, we are able to extract transferrable latent skills that enable dramatic acceleration of learning in cross-task transfer. With a suitable curriculum, this allows us to learn challenging dextrous control tasks like ball-in-cup from scratch with pure reinforcement learning.

1 Introduction

Advances in robot hardware, policy representations and policy-search based reinforcement learning have led to a growing number of successful demonstrations of robot control involving challenging dynamics, including baseball [Peters and Schaal, 2008], ball-in-cup [Stulp *et al.*, 2014], pancake flipping [Kormushev *et al.*, 2010], table tennis [Kober *et al.*, 2010] and tetherball [Kober and Peters, 2010]. These tasks involve non-linear policies and non-convex reward functions. Thus, direct application of reinforcement learning has met with limited success as their non-linearity and large search space would require a prohibitive number of trials. A widely used strategy has therefore been to obtain an initial manual demonstration of the skill [Argall *et al.*, 2009], followed by imitation learning a suitable policy representation such as Dynamic Movement Primitives (DMPs) [Schaal *et al.*, 2005]. Based on this good initial model, policy-search based reinforcement learning is used to fine-tune the resulting control policy [Kober *et al.*, 2010]. Nevertheless, we would like robots to learn skills autonomously, because manually demonstrating each task (or possibly even variant thereof) is labor intensive, and because humans may not even know the solution to some control tasks we wish a robot to solve.

The desire to increase autonomy in this way has motivated extensive work into generalising skills. For example multi-task learning addresses sharing information across multiple skills [Deisenroth *et al.*, 2014; Parisotto *et al.*, 2016] and contextual (or parameterised) policies build skills that generalise across variants [Stulp *et al.*, 2013; Kupcsik *et al.*, 2013] within one family of tasks. Most of these studies however, address generalisation of skills that are relatively simple parametric variants of each other. For example throwing objects of different weights [Stulp *et al.*, 2014], throwing to different target locations [Kupcsik *et al.*, 2013], or going via different waypoints to avoid various obstacles [Stulp *et al.*, 2013].

In this paper, we are inspired by the vision of lifelong learning [Thrun, 1996; Ruvolo and Eaton, 2013b]. That is, the idea that new tasks should get progressively easier to learn as a wider and deeper set of prior tasks are mastered by a learner with the ability to extract task-agnostic generalisations from experience. With this in mind, we go beyond existing contextual policy work and explore transfer learning across a heterogeneous set of dynamic control tasks that do not lie in a simple parameterised family. In our setting, a robot starts with knowledge of a set of source task variants (e.g., throwing type tasks). The aim is then for it to master a different category of target task (e.g., catching type task) autonomously through reinforcement learning (RL). We explore four task families: reaching to a target position, throwing at a target [Deisenroth *et al.*, 2014; Kober *et al.*, 2010], casting at a target [Kober and Peters, 2010], and ball-in-cup [Stulp *et al.*, 2014]. Some of these are typically prohibitively difficult to learn directly with RL, so to learn one task autonomously based only on past experience

[*]University of Edinburgh, School of Informatics

[†]German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Wessling, Germany

[‡]Sorbonne Universités, UPMC Univ Paris 06, CNRS UMR 7222, Institut des Systèmes Intelligents et de Robotique, F-75005 Paris, France

of another, the robot must abstract and transfer task-agnostic generalisations. To achieve this, we define a class of neural network controllers loosely inspired by the DMPs [Schaal *et al.*, 2005] commonly used to solve these dynamic tasks. We first multi-task learn a *set* of multiple source tasks from a given family (e.g., throwing objects of various weights to various locations), and the controllers for these correspond to a stack of such neural networks. Then we *factorise* this set of tasks to obtain transferrable latent skills. Finally, by using these latent skills as a basis to construct a policy network, we are able to learn a set of target tasks (such as ball-in-cup with various string lengths) autonomously, without demonstration. Uniquely our tensor-based transfer framework enables simultaneous discovery and sharing of latent skills across *both* task categories and actuators [Luck *et al.*, 2014].

Our contributions are two-fold: (i) We introduce a DMP-inspired neural network that can represent a variety of dynamic skills, and show how multiple networks can be factored to obtain transferrable latent skills, (ii) We evaluate this idea with four challenging dynamic tasks, and show how transferring latent skills can dramatically speed up learning target tasks, ultimately allowing challenging new skills to be mastered autonomously by policy-search reinforcement learning.

2 Related Work

Learning challenging robot control skills is usually achieved in practice by supervised learning from demonstration. Nevertheless, the vision of learning-based robot control is for robots to learn new skills autonomously. Direct application of RL however requires prohibitive amounts of training, risking physical damage to the robot. This has motivated a fruitful line of research into contextual (parameterised) policies and options where a robot learns to solve a *related family* of tasks, such as throwing objects of different weights [Stulp *et al.*, 2014]. By contrast, we aim to achieve autonomous learning of a new task *category* through transfer learning.

Transfer & Multi-task Learning Transfer (TL) and multi-task (MTL) learning aim to improve and accelerate learning by sharing knowledge across different tasks, in a uni- and multi-directional way respectively. These are well studied topics in supervised machine learning. One of the earliest applications to RL showed that for classic pole balance problems, it was faster to learn a policy for a novel system with knowledge transferred from other similar systems [Selfridge *et al.*, 1985]. TL has since been widely applied to accelerate reinforcement learning [Taylor and Stone, 2009]. MTL has been used to jointly optimise multiple RL tasks including inverted pendulum problems with various mass or length [Lazaric and Ghavamzadeh, 2010] and stacking various numbers of blocks [Deisenroth *et al.*, 2014]. However, in our terminology these are members of task family, rather than distinct task categories. Very recently, multi-task RL has been applied in learning to play multiple video games [Parisotto *et al.*, 2016].

A related work to ours is PG-ELLA [Ammar *et al.*, 2014], which adapts the successful GO-MTL algorithm [Kumar and Daume III, 2012] from supervised learning to the policy gradient based RL setting. Their idea is to apply low-rank matrix factorisation to a stack of linear models and share information through the resulting subspace, and PG-ELLA applies this to control tasks such as pole-balancing. We go significantly beyond this: PG-ELLA/GO-MTL deal with *linear* policies only. This policy representation means they can only address simple linear control tasks, and not the dynamic tasks we study. These require dynamic trajectory planning that is highly non-linear in time. Secondly PG-ELLA shares latent knowledge across tasks only. Our tensor-based TL framework represents policy for each actuator as a slice of tensor and therefore, discovers and shares latent skill across both tasks [Ammar *et al.*, 2014] and actuators [Luck *et al.*, 2014] simultaneously.

Lifelong & Curriculum Learning Lifelong learning takes TL/MTL ideas further with the vision that as more tasks are learned, better task-agnostic abstract knowledge can be extracted. The resulting more humanlike "*learning to learn*" should make each new task easier to progressively easier to master [Thrun, 1996]. These ideas have also been studied in robot control, for example by treating different environments [Ring, 1998], or robot hardware platforms [Isele *et al.*, 2016] as multiple tasks to be mastered in a lifelong learning manner. The choice of task *sequence* is important to the outcome in lifelong learning [Bengio *et al.*, 2009], but despite some work [Ruvolo and Eaton, 2013a], how to predict a good curriculum in advance is still an open question. In this paper we demonstrate that with an intuitive choice of curriculum, our framework can ultimately learn a challenging dynamic control skill autonomously through RL.

Low-Rank Tensors and Factorisation Low-rank tensor models are often studied for missing data imputation, and compressing/speeding up large neural networks [Lebedev *et al.*, 2015]. A few MTL studies used low-rank tensors for knowledge sharing across tasks with a structured description, rather than a simple index [Wimalawarne *et al.*, 2014]. However, in these studies the underlying per-task representation in each case is still a linear model (single vector) that predicts a single output. We address tensor factorisation based transfer to share knowledge across both tasks and multiple outputs (actuators) with non-linear neural network-based models.

3 Methodology

3.1 Policy Representation

The challenging dynamic control tasks we address in this paper are commonly solved by DMP-based approaches. DMPs combines a non-linear open-loop term (which depends on time) with a linear closed-loop term (which depends on state). Inspired by this idea, we design a policy network $a_{t,i} = \pi(t, \mathbf{x}^o_{t,i})$ which produces the ith actuator's force $a_{t,i}$ given the current time t and observed state $\mathbf{x}^o_{t,i}$. The network has inputs for t and $\mathbf{x}^o_{t,i}$, and includes two learnable layers. The first is a radial-basis function (RBF) layer that functions as a trajectory planner, allowing temporally extended and highly non-linear movements to be generated. The second is a linear fully-connected (FC) layer that encodes learned controller parameters that will ensure the trajectory is followed. The roles of these roughly correspond to the two components in a DMP.

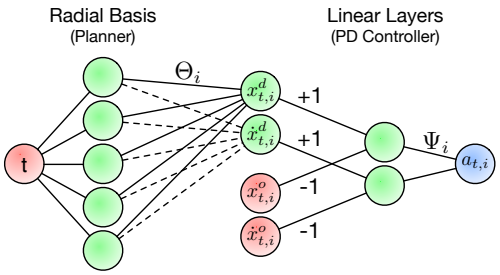


Figure 1: Network defining our policy $\mathbf{a}_i = \pi(\mathbf{x}^o_{t,i}, t)$ from time-step t and observed state $\mathbf{x}^o_{t,1}$ to action $a_{t,i}$.

Specifically, for actuator i the policy is parameterised as:

$$\phi_n(t;\mu,\Sigma) = \frac{1}{\sqrt{2\pi}\sigma_n}\exp\left(\frac{(t-\mu_n)^2}{2\sigma_n^2}\right) \quad (1)$$

$$x^d_{t,i} = \Theta_i \Phi(t;\mu,\Sigma) \quad (2)$$

$$\dot{x^d_{t,i}} = \frac{1}{2\Delta t}(x^d_{t+\Delta t,i} - x^d_{t-\Delta t,i}) \quad (3)$$

$$a_{t,i} = \Psi_i(\mathbf{x}^d_{t,i} - \mathbf{x}^o_{t,i}) \quad (4)$$

Here the learnable parameters are the matrices Ψ_i that implement the controller, and Θ_i containing the weights of N basis functions describing the non-linear trajectory. Fig. 1 illustrates the network for the ith actuator. The dash lines indicate that desired velocity is not generated from the feed forward network directly, but via the Euler method as in Eq. (3). In contrast to the conventional DMP-based pipeline, our approach jointly learns the inter-related problems of trajectory planning and PD controller.

Single Task Learning The network can be trained in a supervised way via learning from demonstration, or by RL via policy-search. For RL, we use Covariance Matrix Adaptation - Evolutionary Strategy (CMA-ES) [Kern *et al.*, 2004], a direct-policy search method for optimisation of non-linear non-convex functions in continuous space. CMA-ES has connections to policy-gradient and has been shown to outperform other RL algorithms for our type of policy [Stulp and Sigaud, 2013]. For an actuator i, both layers' parameters can be summarised as a vector $\mathbf{w}_i$ containing both vectorised RBF weights $\Theta_i \in \mathbb{R}^N$ and controller parameters $\Psi_i \in \mathbb{R}^{\dim(\mathbf{x})}$. Thus, for a given task, the parameters are represented as a matrix $\mathbf{W} \in \mathbb{R}^{D\times A}$ where $D = (N+\dim(\mathbf{x}))$ and $A = \dim(\mathbf{a})$, where $\dim(\mathbf{x}) = 2$ in the case of a PD controller.

3.2 Low-Rank Tensor Factorisation: Latent Skills

Given the above policy representation, we aim to discover latent skills that can be shared across tasks and also actuators. As summarised above, the policy for each task is represented as a matrix, so multiple tasks stack into a 3-way tensor. We will achieve knowledge sharing through low-rank modelling of this tensor. Unlike for matrices, there are many definitions of low-rank tensor factorisation, and we use one of the most general, Tucker decomposition [Tucker, 1966]. Tucker decomposition factors an N-way tensor into a lower-rank N-way core tensor and N matrices along each mode. A 3-way tensor $\mathcal{X} \in \mathbb{R}^{D_1\times D_2\times D_3}$ is assumed to be composed as:

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 \quad (5)$$

where $\mathcal{G} \in \mathbb{R}^{K_1\times K_2\times K_3}$ is the lower-rank core tensor, and $\mathbf{U}_n \in \mathbb{R}^{K_n\times D_n}$ are the factor matrices and can be regarded as the principal components in each mode, and $K_n \leq D_n$. This can be efficiently solved [Lathauwer *et al.*, 2000] as a higher-order singular value decomposition problem and obtaining $\mathbf{U}_n$ as the $\mathbf{U}$ matrix from SVD of mode-n flattening of $\mathcal{X}$, after which the core tensor is obtained by

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}_1^T \times_2 \mathbf{U}_2^T \times_3 \mathbf{U}_3^T. \quad (6)$$

3.3 Multi-Task and Transfer Learning Strategy

Given a set of tasks/skills to learn, we aim to use the above tensor factorisation strategy to extract task-agnostic information to share between them (multi-task learning), and to transfer to benefit the learning of new tasks (transfer learning). A schematic overview of the procedure is given in Fig. 2.

Formalisation We assume that we have P source tasks where $\mathbf{W}_p$ represents the policy parameter matrix for the pth task's network (as described in Sec. 3.1). The policy parameters for all source tasks can be stacked as slices of a 3-way tensor $\mathcal{W}$ of size $D \times A \times P$. The latent task assumption is to factorise the weight tensor as per Eq. (5). In this case $\mathcal{G}$ is the core tensor of size $K_1 \times K_2 \times K_3$ that contains knowledge abstracting both skills and actuators. $\mathbf{U}_2 \in \mathbb{R}^{K_2\times A}$ contains actuator specific knowledge, and $\mathbf{U}_3 \in \mathbb{R}^{K_3\times P}$ encodes task specific knowledge. Based on this decomposition, we consider both multi-task and transfer learning.

Multi-task Learning To jointly learn several tasks, we exploit 'constructive' multi-task learning [Yang and Hospedales, 2015; 2017]. That is, the parameters we actually train with CMA-ES are the factors $\mathcal{G}$ and $\mathbf{U}_{1,\ldots,3}$, which are then multiplied out (Eq. 5) to obtain the tensor parameterising the policies for all tasks (neural network RBF and FC parameters) in order to perform a rollout of a given task. For regularisation we place a L_1-norm on $\mathbf{U}_3$ and L_2-norm on the others. Simultaneously training the parameters in this way shares knowledge across actuators and across tasks.

Learning a Target Task We next consider transferring knowledge to a target task, given a set of source tasks (modelled by $\mathcal{G}$ and $\mathbf{U}_{1,\ldots,3}$ above). To achieve this, we gather the task-agnostic knowledge (latent skills) by contracting the task independent factors into a tensor $\mathcal{L} \in \mathbb{R}^{D\times A\times K}$ as $\mathcal{L} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2$. The tensor $\mathcal{L}$ is then transferred to the target task. This provides a good initial subspace, so that a challenging target task in a different category can in practice now be learned autonomously with RL. The policy parameters for each target task are initialised as a random point in the transferred subspace $\mathbf{W}_{target} = \mathcal{L} \times_3 \mathbf{s}_{target}$ by initialising the weight vector $\mathbf{s}_{target}$ randomly. The learner then searches for the policy $\mathbf{W}_{target}$ by direct policy search in the space of $\mathbf{s}_{target}$ and $\mathcal{L}$ with a L_2-norm regulariser on output weights. Algorithm 1 summarises the overall procedure.

Discussion In contrast to some other one-to-one transfer learning approaches [Taylor and Stone, 2009], an important difference is that our algorithm exploits transfer from *multiple* source tasks in order to extract task-agnostic information

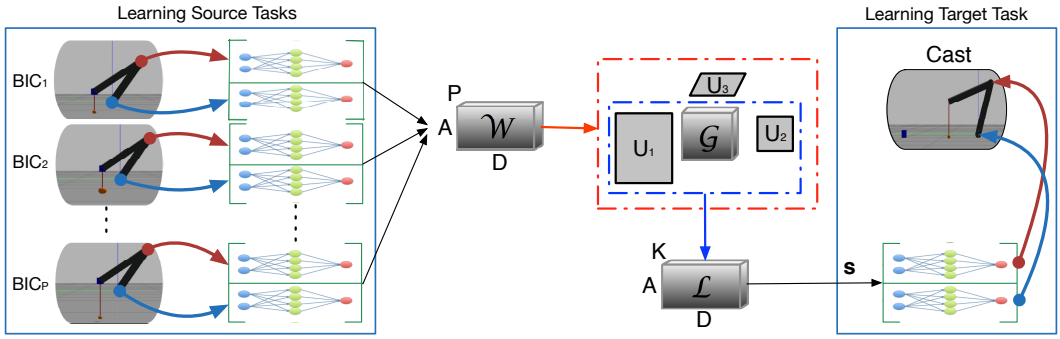


Figure 2: Transfer Learning Schematic. A set of P source tasks are learned and the matrices representing each source policy are stacked in to a tensor $\mathcal{W}$. Factorising $\mathcal{W}$ separates task-specific, actuator-specific and shared knowledge. Task-independent latent skills are gathered in tensor $\mathcal{L}$. For the target task, the agent learns the weights $\mathbf{s}$ that reconstructs $\mathcal{L}$ into policy parameters.

that is likely to be transferable. The subspace/latent task set $\mathcal{L}$ is updated when learning the target, so we can see that transfer learning here is about providing a good *initial condition* for the subspace before performing RL.

Algorithm 1 Tensor-based Transfer Learning

{Source Task Learning}
for $p = 1$ **to** P source tasks **do**
  Learn source task policy Θ_p and Ψ_p.
end for
Initialise tensor $\mathcal{W}$ containing weights as slices.
MTL initial condition: $\mathcal{W} \rightarrow \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$.
Multi-task learn source tasks.
{Target Task Policy Search}
Task-agnostic knowledge tensor: $\mathcal{W}' = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2$.
Initialise $\mathcal{L}$ with slices of tensor $\mathcal{W}'$.
Initialise $\mathbf{s}$ as one random source task.
while not converged **do**
  Fix $\mathcal{L}$ and update $\mathbf{s}$ with CMA-ES
  Fix $\mathbf{s}$ and update $\mathcal{L}$ with CMA-ES
end while

4 Experiments

4.1 Environment and Tasks

We exploit a simulated robot arm whose end-effector operates in a 2-dimensional space. In the dynamical simulator, we assume an ideal inverse kinematic system which is able to convert the movement of end-effector into angular movement of each joint. Our policy network outputs the acceleration of end-effector in both horizontal and vertical directions at each time step t. In detail, the agent observes the positions and velocities, and outputs the accelerations for two actuators ($A = 2$). Each actuator trajectory is modelled by $N = 26$ weighted RBFs (centers between 0 and 5, with variances $\sigma = 0.6$) in the first layer of network and each actuator controller is modelled with 2 PD controlling parameters. For most experiments, we assume $P = 16$ task instances per category, and thus the policy tensor is $W \in \mathbb{R}^{28\times2\times16}$. For transfer learning, we assume $K = 8$ latent tasks throughout. The task categories used in our experiments are as follows:

Target Reaching (Reach) The agent learns to move the end-effector, reaching specific target position with time discounted reward for each time step and penalties on large velocities and accelerations. Tasks within the category are to reach different positions, so task parameters are the 2-dimensional positions of goal points.

Object Throwing (Throw) The learner aims to throw a ball into a basket by holding it with the end-effector and then releasing it during dynamic arm motion [Kober *et al.*, 2010]. We fix the ball release time. The task parameters only includes the position of target basket in 2-dimensional space.

Ball-In-Cup (BIC) In this task [Stulp *et al.*, 2014; Kober and Peters, 2010], the robot holds a cup in its end-effector and the cup has a string attached, from which a ball hangs. Initially, the ball is hanging at rest vertically below the cup. The task is to induce motion in the ball through the string: swinging the ball up and catching the ball with the cup. This is illustrated in Figure 3, the red line shows the position of the ball when the string is taunt and the green lines show the ball position when the string is loose. The task parameters include the length of the string and the mass of the ball.

Casting (Cast) In casting [Kober and Peters, 2010], a ball is attached to the end-effector by a string. The task is to get the ball into a small cup, placed in front of the robot. Because the end-effector controls the ball only indirectly via the string (which may change between loose and taunt depending on the dynamics), the dynamics are very different to throwing. We fixed the position of target cup and the task parameters include the length of the string and the mass of the ball only.

Cost Function The cost for reaching tasks is defined with action penalties and a time discounted linear summation of quadratic functions with respective to the distance to target state $J = \sum_{t=0}^{T}[\gamma^t((\mathbf{x} - \mathbf{x}_{target})^2 - b) + \alpha\|\mathbf{a}_t\|]$, where constant b is a pre-defined baseline. The other three tasks use a cost function of the same parametric form. The cost (cf. reward) J of each episode is based on the difference between horizontal position of the ball x_b and the cup x_c at t_m when they are at the same vertical height. The cost is assigned as 0 if the ball fails to reach the same height as the cup. L_2-norms on actions are added to penalise extreme accelerations. For

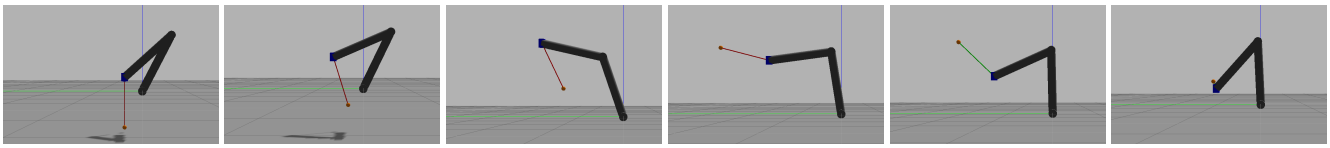


Figure 3: Successful Movement Trajectory of Ball In Cup. The arm starts from static state and moves rightward firstly, generating the ball a momentum toward right, then moves leftward fast, generating the ball a large momentum toward upper left. After the ball is pulled back through the string, the arm moves the cup towards the ball's dropping point and catches the ball.

our experiments, we choose $\gamma = 0.99, \alpha = 0.001$.

$$J = \begin{cases} \sum_{t=0}^{T} \alpha \|\mathbf{a}_t\| & \text{if vertical heights never match} \\ \gamma^{t_m} \min(b_1(x_b - x_c)^2 - b, 0) + \sum_{t=0}^{T} \alpha \|\mathbf{a}_t\| & \text{otherwise} \end{cases}$$

4.2 Transfer Learning

Setting In the first experiment we investigate autonomously learning a new category of target task with RL, given a set of known source tasks from a different category. We use CMA-ES as base learner with the initial parameter $\sigma = 0.01$. Our focus is on comparing the impact of different types of knowledge transfer, assuming the source tasks are well learned. For simplicity, we therefore learn the source with supervised demonstration followed by MTL RL refinement. There are multiple ways to evaluate transfer learning performance [Taylor and Stone, 2009]. We report (i) the total reward/cost during learning and (ii) target task success rate – the percentage of experiments that the robot successfully completes (e.g., gets the ball in the cup) when learning terminates. For parsimony, we adopt an experimental design where each task category is considered in turn as both a source and a target. Therefore 4 task categories entail 16 transfer experiments. Each experiment considers families of 16 source and 50 target tasks.

Alternatives We compare our tensor-based transfer approach with: *Scratch*: learning the target from scratch. *Direct:* A simple direct transfer learning baseline of initialising the target task to that of a (randomly chosen) learned source task. This is a common strategy of transfer by 'warm start' followed by fine-tuning [Taylor and Stone, 2009]. *Matrix:* Matrix-based transfer approach, where the policy parameters for all tasks are structured as a matrix of size $DA \times P$, knowledge sharing is achieved by SVD – thus only transferring knowledge across tasks and not also across actuators. These transfer approaches are thus about providing a good initial condition for the policy-search based RL of the target task, but ours transfers the latent tasks in a tensor structure.

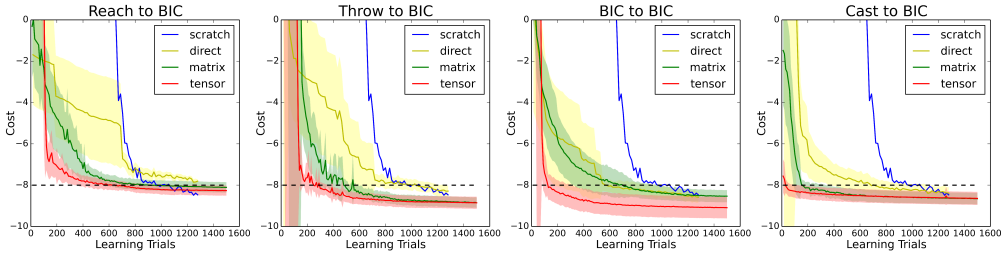
Results Fig. 4a shows illustrative learning curves of 4 of the 16 experiments. Note that only successful learned experiments are counted for the learning curves. Fig. 4b summarises learning success rate and total cost for all 16 experiments. We observe that: (i) Learning from scratch is feasible (it achieves success) although slow/costly for Reaching and Throwing. However it mostly fails to succeed at the harder BIC and Casting tasks. (ii) Among the transfer learning approaches, we see that our tensor-based approach is best in terms of total cost/learning speed, followed by matrix-based transfer, and direct's warm-start approach. (iii) Our tensor-based transfer is the only one to solve (high success rate) most tasks given most sources. Overall the results show that with a suitably designed policy and TL strategy, it is possible to learn challenging dynamic control tasks autonomously, even when transferring from very different and much easier source tasks (e.g., Reach→BIC). Previous solutions to BIC have required demonstration [Stulp *et al.*, 2014].

4.3 Comparative Analysis

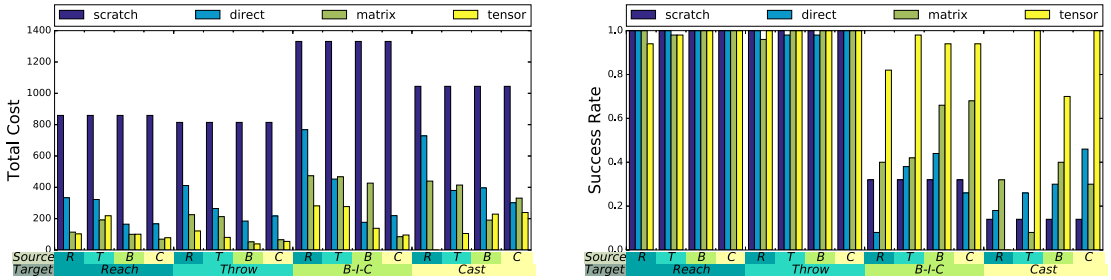
Competitors We next compare our framework against three competitors with alternative policy representations, and transfer strategies. **ELLA:** Our implementation of [Ruvolo and Eaton, 2013b], a state of the art framework for matrix-based transfer of linear models. We adapt to our purpose by providing it the inputs $[\mathbf{x}_t, t]$ our model uses. **RBF-ELLA:** As ELLA is designed for linear models, we generalise it for non-linear tasks through RBF-tiling the input space. **FCNN:** A fully connected multi-layer NN policy mapping $[\mathbf{x}_t, t]$ to actions via one RELU hidden layer of 8 units is a conventional alternative to our DMP-inspired policy.

Results We perform the same experiment as the previous section, considering 16 task pairs for transfer. Fig. 5 shows an illustrative example of the learning curve of each method along with summary statistics (i) Average learning cost, and (ii) % of total cost 'wins' (which method had the smallest cost in a given experiment). Fig. 5a illustrates our tensor-transfer approach starting off better and converging quicker and to better asymptote than alternatives. Fig. 5b shows that we achieve much better total cost on average, and considering each experiment achieve the best total cost in most cases.

Discussion Why does our method perform so much better than ELLA which has previously shown convincing results on benchmarks like cart-pole, quadrotor, etc? The linear policy of ELLA can solve these classic benchmarks as they have a specific target state (e.g., balance/hover position) and are linearly controllable closed loop systems [Mokhtari and Benallegue, 2004] that do not require extended dynamic trajectory planning. However in our dynamic manipulation tasks, the objective is not a simple target state: it is an extended movement trajectory. Our policy class including a non-linear open loop layer as trajectory planner, and closed-loop controller can address this. ELLA can be seen as implementing only the linear feedback controller in the final layer of our network. RBF-ELLA is much more flexible and could potentially solve our tasks. But it suffers from the needing to tile RBFs in 3/5D (P/PD control), which means either an under-fitting policy, or very many parameters to train (we used $5 \times 5 \times 5$ tiling to give it a similar number of parameters to ours).

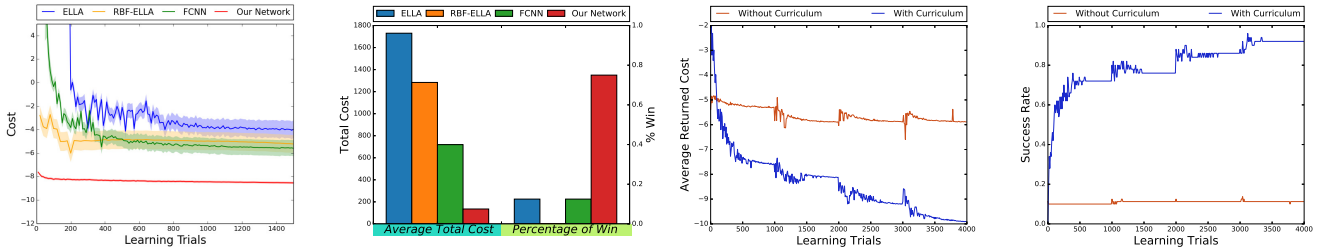


(a) Learning curves of transfer learning from 4 source tasks to BIC tasks



(b) Total Cost of Learning and Success Rate at Convergence

Figure 4: Learning curves (a) and summary statistics (b) of different transfer strategies.



(a) Casting to BIC transfer (b) Summary of all transfer pairs (c) Learning Curve of BIC by CL (d) Success Rate of BIC by CL

Figure 5: (a,b) Comparison vs State of the Art. (c,d) Autonomously learning ball-in-cup from scratch via curriculum.

4.4 Curriculum Learning

The previous experiments showed that with pre-learned source tasks, we can autonomously learn a new task category with transfer and RL. In this section we explore whether it is possible to learn dynamic manipulation tasks autonomously with no supervision anywhere in the pipeline, by constructing an appropriate training curriculum. The procedure starts with learning the easiest task-category from scratch with multi-task RL, and the task-agnostic tensor is extracted and transferred to bootstrap learning the next task-category with transfer multi-task RL. We consider the task category curriculum Reach-Throw-BIC. We compare following this curriculum (CL) using tensor transfer against no curriculum (NC). We control for the total cost in rollouts giving both conditions 6000 trials of pre-training. In NC these iterations are used on MTL learning of BIC family tasks. In CL, they are spent on MTL learning of reaching then throwing task categories.

Based on these initial conditions, Fig. 5c, 5d shows the average cost and success rate of the following 4000 trials of MTL training on BIC for both approaches. The periodic bumps are caused by the alternating optimisation of $\mathcal{L}$ and $\mathbf{s}$. The results show that the challenging BIC task can be autonomously solved by knowledge transfer from a curriculum of autonomously learned source tasks, thus achieving BIC without any demonstrating supervision.

5 Conclusions

We explored transfer learning to enable autonomous reinforcement learning of non-linear dynamic control tasks. Through our effective policy network representation and tensor based transfer of the latent task subspace, the speed and asymptotic success rate of autonomous RL of target tasks is significantly improved. With curriculum transfer learning, we were ultimately able to learn hard tasks such as BIC from scratch autonomously without any demonstration. In future work we will explore lifelong MTL within and across task families, and extend our framework to model tasks within families as contextual policies rather than discrete tasks.

Acknowledgements

This research was supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 640891.

References

[Ammar *et al.*, 2014] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew Taylor. Online multi-task learning for policy gradient methods. In *ICML*, 2014.

[Argall *et al.*, 2009] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, May 2009.

[Bengio *et al.*, 2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009.

[Deisenroth *et al.*, 2014] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox. Multi-task policy search for robotics. In *ICRA*, pages 3876–3881, May 2014.

[Isele *et al.*, 2016] David Isele, Jose Marcio Luna, Eric Eaton, Gabriel V. de la Cruz, James Irwin, Brandon Kallaher, and Matthew E. Taylor. Lifelong learning for disturbance rejection on mobile robots. In *IROS*, 2016.

[Kern *et al.*, 2004] Stefan Kern, Sibylle D Müller, Nikolaus Hansen, Dirk Büche, Jiri Ocenasek, and Petros Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms–a comparative review. *Natural Computing*, 3(1):77–112, 2004.

[Kober and Peters, 2010] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1):171–203, 2010.

[Kober *et al.*, 2010] J. Kober, E. Oztop, and J. Peters. Reinforcement learning to adjust robot movements to new situations. In *Robotics: Science and Systems (R:SS)*, 2010.

[Kormushev *et al.*, 2010] P. Kormushev, S. Calinon, and D. G. Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *IROS*, 2010.

[Kumar and Daume III, 2012] Abhishek Kumar and Hal Daume III. Learning task grouping and overlap in multi-task learning. In *ICML*, 2012.

[Kupcsik *et al.*, 2013] Andras Gabor Kupcsik, Marc Peter Deisenroth, Jan Peters, and Gerhard Neumann. Data-efficient generalization of robot skills with contextual policy search. In *AAAI*, 2013.

[Lathauwer *et al.*, 2000] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. In *SIAM Journal on Matrix Analysis and Applications*, 2000.

[Lazaric and Ghavamzadeh, 2010] Alessandro Lazaric and Mohammad Ghavamzadeh. Bayesian multi-task reinforcement learning. In *ICML*, 2010.

[Lebedev *et al.*, 2015] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using finetuned cp-decomposition. In *ICLR*, 2015.

[Luck *et al.*, 2014] K. S. Luck, G. Neumann, E. Berger, J. Peters, and H. B. Amor. Latent space policy search for robotics. In *IROS*, 2014.

[Mokhtari and Benallegue, 2004] Abdellah Mokhtari and Abdelaziz Benallegue. Dynamic feedback controller of euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle. In *ICRA*, 2004.

[Parisotto *et al.*, 2016] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR*, 2016.

[Peters and Schaal, 2008] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682 – 697, 2008.

[Ring, 1998] Mark B Ring. Child: A first step towards continual learning. *Machine Learning*, pages 261–292, 1998.

[Ruvolo and Eaton, 2013a] Paul Ruvolo and Eric Eaton. Active task selection for lifelong machine learning. In *AAAI*, 2013.

[Ruvolo and Eaton, 2013b] Paul Ruvolo and Eric Eaton. Ella: An efficient lifelong learning algorithm. In *ICML*, 2013.

[Schaal *et al.*, 2005] Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. In *International Symposium on Robotics Research*, 2005.

[Selfridge *et al.*, 1985] Oliver G Selfridge, Richard S Sutton, and Andrew G Barto. Training and tracking in robotics. In *IJCAI*, pages 670–672, 1985.

[Stulp and Sigaud, 2013] F. Stulp and O. Sigaud. Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn, Journal of Behavioral Robotics*, 4(1):49–61, 2013.

[Stulp *et al.*, 2013] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, and O. Sigaud. Learning compact parameterized skills with a single regression. In *Humanoids*, 2013.

[Stulp *et al.*, 2014] Freek Stulp, Laura Herlant, Antoine Hoarau, and Gennaro Raiola. Simultaneous on-line discovery and improvement of robotic skill options. In *IROS*, 2014.

[Taylor and Stone, 2009] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[Thrun, 1996] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS*, 1996.

[Tucker, 1966] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[Wimalawarne *et al.*, 2014] Kishan Wimalawarne, Masashi Sugiyama, and Ryota Tomioka. Multitask learning meets tensor factorization: task imputation via convex optimization. In *NIPS*, 2014.

[Yang and Hospedales, 2015] Yongxin Yang and Timothy M. Hospedales. A unified perspective on multi-domain and multi-task learning. In *ICLR*, 2015.

[Yang and Hospedales, 2017] Yongxin Yang and Timothy M. Hospedales. Deep multi-task representation learning: A tensor factorisation approach. *ICLR*, 2017.