Task-Specific Loss: A Teacher-Centered Approach to Transfer Learning Between Distinctly Structured Robotic Agents

Mehdi Mounsif[1(✉)], Sébastien Lengagne[1], Benoit Thuilot[1], and Lounis Adouane[2]

[1] Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, 63000 Clermont-Ferrand, France
mehdi.mounsif@uca.fr

[2] Université de Technologie de Compiègne, CNRS, Heudiasyc, 60200 Compiègne, France

Abstract. Recent progress in robotics and artificial intelligence let us envision a future where robot presence and activity will be ubiquitous. Fueled by economics, cultural background, and design choices, human creativity will most likely design robots of various forms and shapes, using a wide range of sensors and actuators to accomplish their tasks. Consequently, it is highly probable that differently structured robots will be required to perform the same task. As many of these tasks may require learning-based control, which still relies on millions of examples to perform correctly, it would be eminently useful to be able to transfer skills from one agent to another, notwithstanding their distinct physical structure. As such, we propose a new method for the fast transfer of skills using a family of differentiable task-specific distance metrics called Task-Specific Losses (TSL). After highlighting the main shared concepts and differences with the closest existing state-of-the-art method, we demonstrate this technique on two different assistive control tasks, showing that we can indeed transfer the realization of a task learned from an expert/teacher to an agent with no previous interaction with the environment.

Keywords: Transfer learning · Variational autoencoders · Generative adversarial networks · Control · Differentiable models

1 Introduction

Knowledge sharing has always held a particularly important place within human societies, from the first educational tales transmitted vocally to modern knowledge instances such as universities, the internet or social media. When engaged in physical activities, humans can also benefit from the knowledge transfer. Indeed, in environments such as sport classes, relying on an expert/teacher to learn a movement can greatly decrease the time needed to master these skills.

Given the immense benefits of this human feature, considerable efforts have been dedicated to adapt and translate this transfer process for a pool of robots. Within the scope of learning-based methods, Reinforcement Learning (RL) methods present the closest paradigm to the human traits introduced above and have been observing

O. Gusikhin et al. (Eds.): ICINCO 2020, LNEE 793, pp. 166–186, 2022.
https://doi.org/10.1007/978-3-030-92442-3_10

a strong popularity increase, due to recent advances [33,36,37,47,52]. Nevertheless, these methods are also known for their very poor sample efficiency and their extreme sensibility to state distribution. Indeed, even when trained, transferring a successful RL policy from a simulated environment to the real world is very challenging. Consequently, the added complexity of having to additionally make the policy adapt to a new robot thus appears to be beyond the scope of current RL capabilities. As a supplemental motivation, let us consider a case of everyday life usage of robots such as assistive robots.

As the use of assistive robots spreads among the elderly and people with physical disabilities, it appears crucial that their usage is as intuitive and easy as possible. However, as the agility of these robots is linked to their degrees of freedom, their increased dexterity de factor makes them severely arduous to control through a joystick. As painful as it is to get used to this type of control, this process shouldn't be replicated every time the assistive robot is changed.

Following the concept of teacher-student transfer proposed in [28], this current work proposes an alternative way to envision the transfer of knowledge between differently structured agents. More precisely, using a non-linear encoding of a task realization for a given robot, it is possible to create an intuitive controller that pilots a robot with a high number of degrees of freedom with low-dimensional inputs from a user. Then, we want to enable the transfer of the latent space of the skills needed to solve the task to another robot featuring a different body configuration. To do so, we will introduce the concept of Task-Specific Loss (TSL). Formally, the TSL is a differentiable, task-specific distance metric that aims at inducing a similar state variation, in both the student agent and the teacher agent. As a result, this allows us to quickly distill the teacher skills within the student agent without it having to access to the task environment.

2 Related Works

Currently, the concept of transfer learning is essential to an important part of deep learning research, particularly in supervised and unsupervised tasks. Originally motivated by the increasing number of parameters in Computer Vision models [14,20,56,61], it has proved very useful even for downstream tasks such as detection or segmentation [11,12,41–44,46,54]. Since the introduction of Language Models (LM), a similar trend has been observed in Natural Language Processing [6,7,15,21,24,40] and it is very common for deep learning practicioners to initialize a model with pre-trained parameters.

In modern data-based approaches, control tasks, such as controlling a robot in a manipulation setting, is usually addressed with RL techniques, rather than supervised learning [31,32]. Impressive progresses in the field have made model-free RL methods very attractive [1,2,16,25,37,38,53,55]. Nevertheless, these techniques still suffer from a low sample efficiency, which in turn generates interests for transfer learning. However, in this setting. it is not clear how to apply transfer learning partly due to the important modification between environments [57]. Consequently, there exists multiple ways to conceive transfer learning in the RL scope. Various works focus on transferring knowledge between simulation and reality [31,34,39] where the author implement strategies to make the agent less sensitive to state distribution modification. Another important research axis is concerned with transfer of skills between tasks [1,2,35] where the objective is to repurposed pretrained agents on a different task, for instance through the modification of the reward function. While the state space and action space stay the same, the authors have shown that some policies can benefit from their past experience to improve their performances in the new task. An alternative strategy for similar results is the Curriculum based transfer [5,31,45,59]: in this context, a complex task is broken down in multiple stages which help the agent learn faster while tackling the full task initially would likely have resulted into a failure to learn. Alternatively, multiples works consider broadening the agent's skillset through the use of intrinsic motivation [3,9,22,29,48,49,51,63]. These strategy, either relying on a curiosity-based reward or an entropy-maximization scheme incites the agent to maximize its state-space exploration which in turn is bound to develop a more thorough control, ultimately offering a better preparation for potentially new environments. This paradigm can also be encountered in most works relying on Meta-Learning [8,10,17,30,50,58,64] where the goal is to increase the agent understanding and adaptability by training it on a span of tasks. While extremely appealing, in practice, the task distribution considered in these settings are usually very narrow [62]. For instance, a common evaluation of Meta-RL algorithm involves changing the maximal joint velocities of a serial robot on a manipulation task while another one consists in choosing different running directions for simulated legged robots [10].

As can be understood from the above paragraphs, transfer learning and fine-tuning are becoming increasingly present in modern deep learning. Nevertheless, there is a clear contrast when considering the application of transfer learning in supervised learning and reinforcement learning. Indeed, supervised tasks and common models architecture are particularly well suited to preserved the embedded knowledge and pass it on to for downstream tasks. From this point of view, the absence of dimensional bottleneck in RL models and the diversity of agents considered makes transfer less straightforward. Although very diverse and powerful frameworks are investigated, each one of these is also focuses on a single agent with a fixed morphology. However, there exist numerous real-world cases that would rather benefit from transferring knowledge from one entity to a distinct one. For instance, the authors of [26,27] propose to create a task model, independent from the agent's body, thus allowing them to transfer it from one agent to another. Although this technique is theoretically powerful, it is in practice constrained by the information bottleneck between the task model and the agents, limiting its flexibility. In this dynamic, the Task-Specific Loss (TSL) function can cope with a broad scope of situations and is not limited by the task model expressiveness. The specific objective considered in this paper has been studied in [28], where the author developed the CoachGAN method. The next section focuses on the main characteristics of this method and provides motivations for the TSL.

3 From CoachGAN to TSL

In order to fully appreciate alternative paradigm and the enhancements brought by the TSL framework to the transfer of task knowledge between kinematically distinct agent, this section goes over the most important ideas and concepts of the CoachGAN methods and highlights its relationship with the TSL framework.

3.1 The CoachGAN Principle

The CoachGAN principle is best understood with an analogy: Consider a crowd assisting to a sports event, such as a boxing match. It is likely that the many individuals among the assistance are not genuine experts of the sports. Nevertheless, having assisted to several matches has endowed them with enough experience to be able to detect *good* moves (at least some) and distinguish them from actions that are unlikely to yield desirable results. As such, they can provide a feedback signal to the current fighter but are also able to translate their overall understanding to a different training athlete.

The CoachGAN method is inspired by the situation described above and implements it within the Generative Adversarial Network (GAN) framework [13]. Specifically, to transfer knowledge from an expert agent to a potentially kinematically different student agent, the CoachGAN method proceeds in two-steps:

1. Trains a teacher adversarial pair, such as presented in Fig. 1, green rectangle: Using the expert dataset, a GAN is trained to produce states that are likely to be the result of the expert policy. The goal in this step is to create a reliable discriminator (an educated boxer observer)
2. Once the teacher discriminator is ready, it is repurposed to train a student generator to produce experts-like states, orange rectangle in Fig. 1

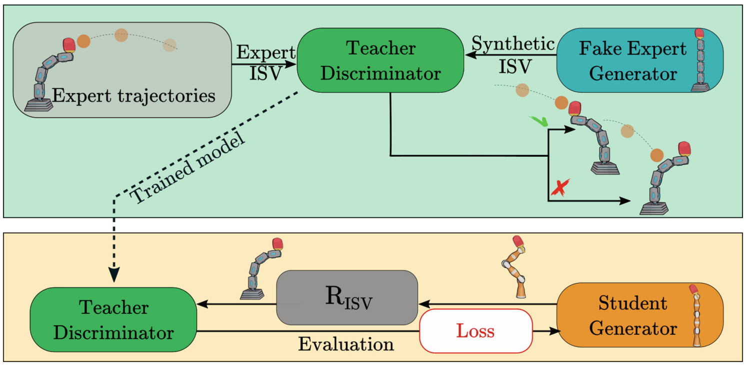

Fig. 1. CoachGAN high-level principle [28]. The green rectangle depicts the teacher adversarial pair training process. Once trained, the teacher discriminator is used to train the student, in the orange rectangle. The concept of ISV is introduced in Sect. 3.2.

While interesting conceptually, this approach faces a non-negligible issue due to the fact that the expert and the student have probably different state spaces. To circumvent this obstacle and allow the discriminator to provide a suitable learning signal to the student generator the Intermediate State Variable (ISV) concept is introduced.

3.2 Intermediate State Variable

The interesting aspect of CoachGAN is to be able to transfer knowledge from one agent to another one with different morphology. However, the geometry differences between the expert and the student usually result in different state spaces, which would render unusable the teacher discriminator knowledge if it were applied directly to the student states distribution. Consequently, the CoachGAN method first requires the user to define a state transformation function (noted R_{ISV} in Fig. 1) that will produce an intermediate state representation, the ISV, on which the teacher discriminator is trained. The student generated states will then be transformed accordingly to enable the discriminator knowledge to be applied blindly to both agents. As an example, consider the Tennis environment in which an agent is trained to bounce the ball against a wall. Specifically, let us define the expert agent as a 5 DoF, planar serial manipulator and the student as a 4 DoF robot. In this case, a possible ISV could be the effector position on impact: the teacher discriminator would be trained to evaluate effector positions given the initial ball configuration (position and speed), based on the expert dataset. It is then possible to transfer this knowledge to the student agent. As all agent training relies on backpropagation, it is crucial for the state transformation function to be differentiable: failing to meet this condition would effectively prevent the student error, as defined by the discriminator, to be used to optimize the students weights.

3.3 CoachGAN Results and Analysis

Relying on an adversarial framework and a simulated Tennis task as experimental environment, the CoachGAN technique was evaluated using various Intermediate State Variables (ISV), such as the effector position or the rebound ball speed that demonstrates the flexibility of this framework. Figure 2 shows the teacher discriminator evaluation of effector positions for a given initial ball configuration (position and speed) as well as student generated joints positions and their resulting effector position. As the teacher discriminator is trained on the expert dataset, it learns to recognize suitable effector positions to hit the ball. In order to evaluate the teacher discriminator reliability a vector of sample effector positions is fed to the discriminator, for a given ball configuration, which then return an associated scalar for each position that represents the desirability of this position. This measure is represented by the colored background, where yellow color indicates highly desired effector positions and blue colors symbolize positions unlikely under the expert dataset. Once the student generator is trained, it is similarly evaluated for a given ball configuration. As can be observed, the student distribution of effector positions is located in an area considered suitable by the teacher discriminator and the best rated position is close to the ground truth (expert) position. Additionally, it can be noted that the untrained student generator states propositions result in a wide distribution, far from the discriminator preferences. These results suggest that CoachGAN is indeed a suitable approach for transferring task knowledge to a student agent without interaction of this agent with the task.

Furthermore, this method allows training student within minutes, while most RL tasks require several hours of training, as displayed in Fig. 3. Thus, the applications of CoachGAN are numerous, especially given the increasing availability of approximation models that could result in very diverse and flexible ISV.

This method, in its current form does however present some limitations. Specifically, the student generator outputs correspond to final target states (for instance, the expected configuration for touching the ball). This output definition can in some cases

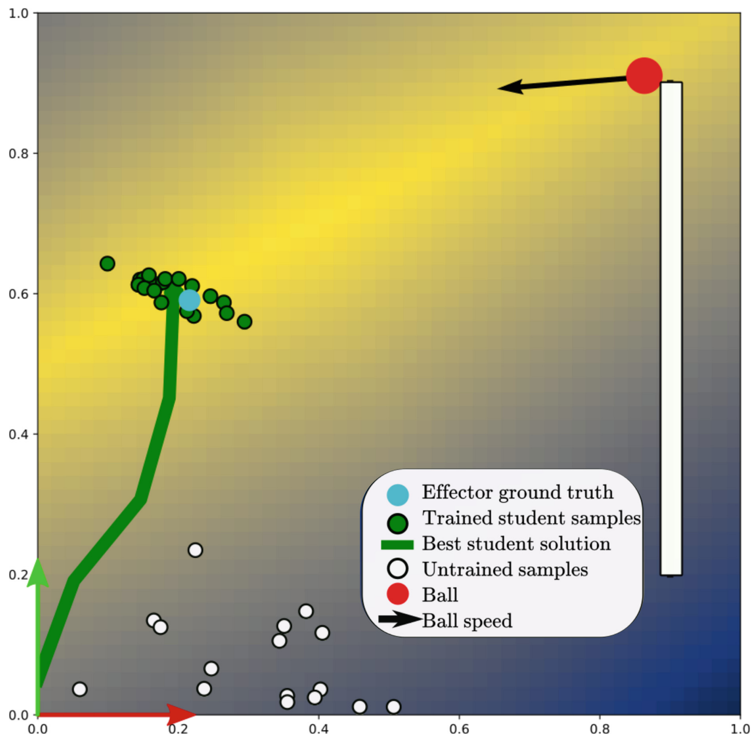

Fig. 2. CoachGAN results: [28]. Student generator proposed configurations and their resulting effector position. Best position according to the teacher discriminator is represented by the full robot. Background color represents teacher discriminator evaluation of ISV (effector positions) in the task-space.

be detrimental to the application spectrum, particularly in cases where unexpected event must be handled. This limitation stems directly from the learning process. Indeed, as the learning loss is established based on the ISV, the discriminator learns to evaluate states. As the student generator is trained using the discriminator, predicting intermediate state conditioned on current state makes training more difficult.

Taking into account this limitation, various alternative ways could offer a way out. A straightforward approach would be to adapt the generator to output several intermediate key state and sum the teacher discriminator evaluation of these to compute the error. Although appealing, this formulation could introduce stabilization issues related to the number of intermediate steps and the distance between them. Another paradigm would be to find a way to evaluate the results of a student action with respect to the action taken by the teacher for a similar environment configuration. The next section introduces the Task-Specific Loss (TSL), which implements this strategy to propose a reactive agent formulation.

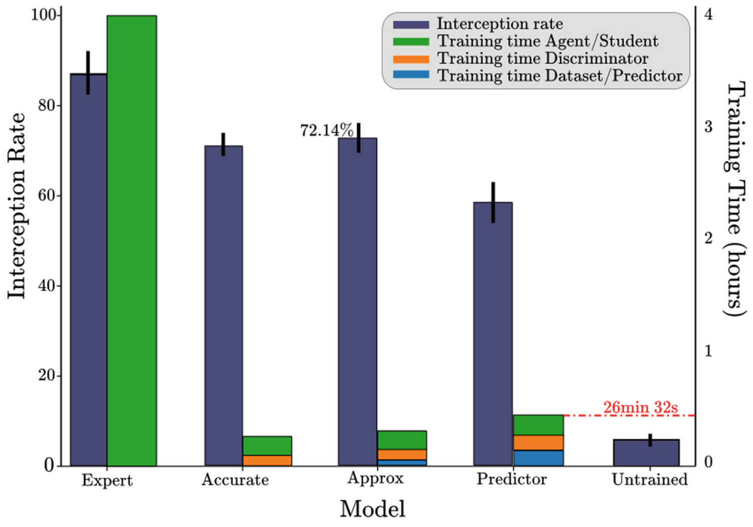

Fig. 3. CoachGAN results: Interception performance and training time [28].

4 The Task-Specific Loss Method

As mentioned above, the Task-Specific Loss (TSL) approach introduced in this paper distills the task knowledge of a teacher agent within a student agent of a different structure through the use of a loss function relying on a differentiable task-specific metric. In this section, we first present the original learning process and pipeline, proposed in [23] for the assistive control of robots. It relies on an expert dataset, which is a set of trajectories composed of a succession of state and action tuples. These trajectories can be generated with a trained RL process or a state-of-the-art robotic method, if available. Afterward, focus is set on the proposed transfer process and the various losses used to reach this goal.

4.1 Learning in the TSL Framework

As explained, our goal is to transfer a skill learned with a given robot to another one with a different body structure. As opposed to [26], where a manually-defined vector of physical values is used as an interface between the robot and the task, we would like to avoid to **explicitly** restrict the state representation.

Thus, the first step of this method is to create a latent space suitable for solving the task. We do so through a Variational AutoEncoder (VAE) [19], which is a common generative type of neural networks. A classic VAE architecture is shown in Fig. 4. The main principles behind a VAE are rather close to those behind a classic AutoEncoder (AE). Indeed, they both compress high-dimensional inputs to a much smaller representation, sometimes called *code* and denoted z, with a loss function that encourages the network to reconstruct the input as closely as possible. However, VAE are also controlled through an additional loss, the KL loss (Kullback-Leibler divergence loss [19]) that focuses on the low-dimensional representation and penalizes the network shall the *code* distribution be too far from a multidimensional Gaussian. This allows for a smooth and exploitable latent space structure, thus giving the VAE its generative capacity. Practically, the total loss L_{VAE} to be minimized by the model can be put as follows:

$$L_{\text{VAE}} = L_r + \alpha L_{\text{KL}} \tag{1}$$

with $\alpha \in \mathbb{R}^+$ a weighting parameter for the distribution regularization, L_r and L_{KL} the reconstruction and KL loss, respectively.

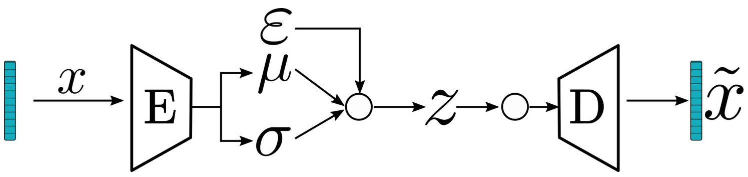

Fig. 4. Classic state-of-the-art VAE architecture [19].

Specifically, for a given batch of inputs vectors x, D being the decoding part of the VAE model, the reconstruction loss is equal to the mean distance between the input x and the reconstruction $\tilde{x} = D(z)$. As the sampling operation is not differentiable, the vector z used by the decoder relies on a reparametrization trick that involves using two heads after the encoder E, one for the mean vector μ and the other for the standard deviation vector σ. These vectors have the same dimensionality as z. Based on the input batch x, we have $\mu, \sigma = E(x)$. Then, using a sampled batch of ε, z is obtained by the following: $z = \mu + \sigma\varepsilon$. This error is then averaged over a whole batch, as denoted by $\mathbb{E}$, leading to, for the general case in Fig. 4:

$$L_r = \mathbb{E}[||x - \tilde{x}||] \text{ where } \tilde{x} = D(z) \tag{2}$$

Next, the KL (Kullback-Leiber) divergence loss incites the model to minimize the distance between the current distribution of *code* z, parametrized by the vectors μ and σ, and a normal distribution centered on the origin with a unit standard deviation:

$$L_{\text{KL}} = \text{KL}[\text{N}(\mu, \sigma), \text{N}(0, \mathbb{I})] = \sigma + \mu^2 - 1 - \log \sigma \tag{3}$$

In our case, we rely on an alternative VAE architecture designed for assistive control, introduced in [23] and represented in Fig. 5. In this application, the dataset is composed of sequences of states s and actions a. The architecture task is to reconstruct, from the code z and the current state s, the target action a from the dataset. Thus, the reconstruction objective from Eq. 2 becomes:

$$L_r = \mathbb{E}[||a - \tilde{a}||] \text{ where } \tilde{a} = D(z, s) \tag{4}$$

Conceptually, it is possible to see this setup as a way to embed the expert's strategy/mind within the *code* z. Then, conditioned by the current agent state s, it allows the decoder to adequately reconstruct the action, even though the action dimensionality is higher than the code size, see Fig. 5. This process results in the creation of an intuitive and user-friendly controller for an agent with the same body structure as the expert. On test phase, the *code* z is given by the user, allowing him to control a higher DoF agent with a low-dimensional input, as shown in Fig. 6.

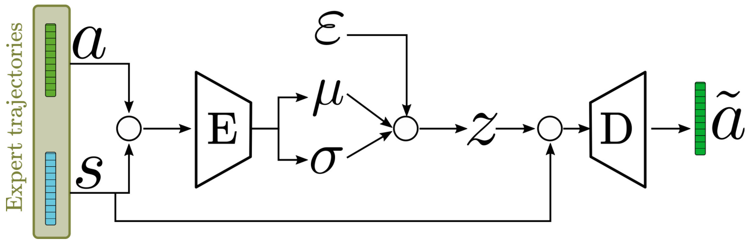

Fig. 5. User-assisted VAE architecture training during training phase.

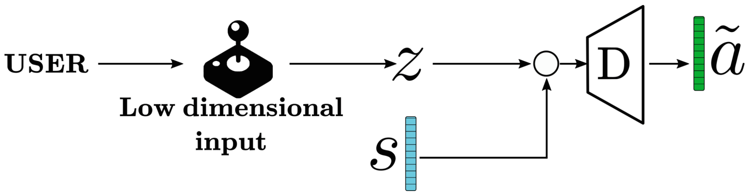

Fig. 6. User-assisted VAE architecture during usage/testing phase.

The Task-Specific Loss (TSL), the main contribution of this paper, extends the framework proposed in [23] by allowing the constructed assistive controller to be transferred to another structurally different robot. Furthermore, our approach is scalable and can be considered for the transfer of skills between autonomous agents as long as a differentiable TSL can be defined, as detailed below.

4.2 Transfer in the task specific loss framework

From the expert dataset, an intuitive agent (the teacher) has been created, it is represented by the decoder D_T shown in Fig. 7. The goal is now to create a decoder D_S for a student agent with a body configuration that is distinct from those of the teacher agent, see Fig. 7. The main motivation here is to leverage the previously trained model and thus avoid recreating a dataset of expert trajectories for the student agent. Indeed, there are numerous examples of systems where generating several instances of datasets could be costly, difficult or even plainly impossible, for instance demonstrations of human experts on a physical system. However, here lies the main obstacle to the transfer. Indeed, as the two agents do not share the same body structure, it is unclear how to force the student to mimic the teacher. The Task-Specific Loss (TSL) offers a conceptual answer to this issue by introducing a differentiable task-relative metric as the loss function. **Its goal is to induce a similar state variation, for the student as for the teacher, given a task-specific metric**.

The TSL approach shares features with both Reinforcement Learning (RL) and Supervised Learning (SL), but also clearly distinguishes itself from these two

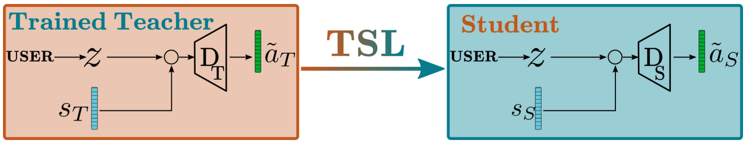

Fig. 7. Distillation process from the teacher to the student agent *via* TSL.

approaches. Specifically, the process takes place in a Markov Decision Process (MDP), similarly to RL, but does not rely on a reward function in the classical way. Indeed, the reward signal from RL is usually delivered from an external process, thus forcing the agent's optimization to act on the action probability distribution, not on the action itself directly. Also, RL relies on exploration to discover suitable policies, while the TSL student agent can directly learn from the teacher. The RL exploration process is prone to lead to very low sample efficiency, whereas the TSL importantly improves this aspect, as demonstrated in Sect. 6.3. The TSL method also differs from SL methods, because although it uses a loss function that recalls classical SL methods, it is not necessary to build a dataset which would defeat the purpose of the transfer from the teacher agent.

The main idea of the TSL approach is to enforce a similar state variation between both agents. This state variation is the result of the action, which is the decoder's output and is consequently differentiable. As a result, if the state modification function can be written in a differentiable fashion, it is then possible to compute directly how accurate was the student's action with respect to the teacher's. This, in turn, allows to compute the error and apply the backpropagation directly to the student's weight matrices. Specifically, given a starting position for the teacher and the student agents, we first compute the value d_0, corresponding to the initial distance based on the current TSL metric, a task-defined state similarity measure.

$$d_0 = ||f_{\text{ISV}}(s_S) - f_{\text{ISV}}(s_T)|| \tag{5}$$

where f_{ISV} is the function defined that transforms the agent state into the ISV, and s_S, s_T are the student and teacher states, respectively.

Next, a *code* vector is sampled from a normalized and centered Gaussian distribution $\mathcal{N} \sim (0, I)$, in order to respect the teacher's training *code* distribution. This code z is used by both agents, along with their current state to produce an action which respectively affects the state of each agents.

$$s'_S = m_s(s_S, D_S(s_S, z)) \tag{6}$$

where s'_S the new student state is the output of the differentiable student model m_s given the current student state s_S and the student action computed by the student decoder $a_S = D_S(s_S, z)$. A similar operation is applied to the teacher agent.

From here, it is possible to compute the next metric value d_1, that is, the differences between the new states for both the student and the teacher:

$$d_1 = ||f_{\text{ISV}}(s'_S) - f_{\text{ISV}}(s'_T)|| \tag{7}$$

which finally leads to the TSL loss:

$$L_{\text{TSL}} = |d_0 - d_1| \tag{8}$$

As both distances were obtained using differentiable pipelines, it is possible to directly determine the gradients based on how close the student action was from the teacher's and optimize the student parameters accordingly.

However, it is paramount that the student mimics the teacher only when their initial states are deemed close enough, task-metric-wise. Hence, before backpropagating the loss, each element is weighted accordingly to the initial distance between the agents. Formally, we use the following weighting strategy:

$$w = \exp(-d_0 \times \alpha_{\text{task}}) \tag{9}$$

That is, the importance of each element is exponentially proportional to its initial distance d_0, where α_{task} is a regularization factor used to take into account the scale of each environment. Finally, the loss is computed as the mean of each example error and we have:

$$\mathcal{L}_{\text{TSL}} = |d_{1,i} - d_{0,i}| \times w_i \tag{10}$$

This process is depicted in Fig. 8.

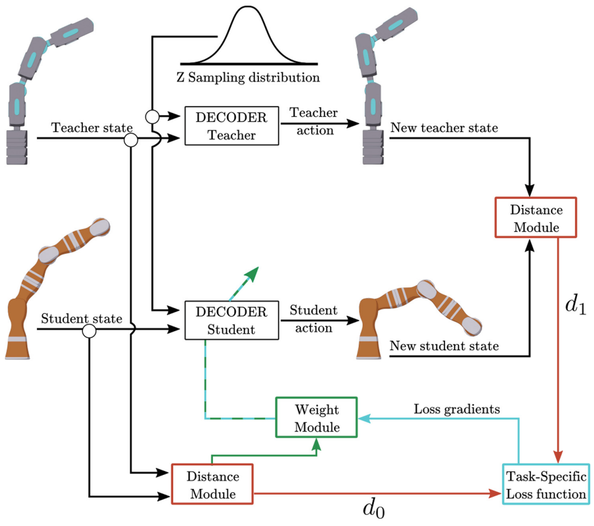

Fig. 8. TSL penalizes the student when its movement does not preserve the estimated metric at the previous timestep.

5 Experimental Setup

In this section, two assistive control tasks used to validate the proposed approach are introduced. Each of them uses two specific serial manipulators controlled by a human user through the *code z*. The setup and goal of each task are now detailed.

5.1 Tasks and Agents

Circle to Circle Contour Displacement. In this task, the objective is to move the end effector of a 5 rotoid joints (5R) agent between and along two quarter circles of different radii. We first create an expert dataset, that consists of sequences of states (joints angles) and actions (joints speed), using an inverse kinematics based approach to train the teacher, Fig. 5. The size of the latent space used for the *code z* is 2, as inspired by the works in [23]. Next, the task metric is defined as the distance between the between the teacher and the student end-effectors. Specifically, we developed a custom batch forward kinematic module that computes the end-effector position based on joints angles and segment lengths. That is, for each batch element:

$$d_{\text{circle},i} = d_{\text{Euc}}(F(l_t, \text{ang}_{i,t}), F(l_s, \text{ang}_{i,s})) \tag{11}$$

where d_{Euc} is a module that computes the Euclidean distance between two points, F is the differentiable forward kinematics module, l_t is the teacher segments length, $ang_{i,t}$ the teacher joint angles at time i and we use respectively l_s and $ang_{i,s}$ for the student. Then, for a whole batch of examples, we have:

$$\mathcal{L}_{\text{circle}} = \underset{i\in[0\ldots n]}{\mathbb{E}}[w_i \times |d_{\text{circle},i} - d_{\text{circle},i+1}|] \tag{12}$$

Afterwards, this task knowledge is distilled within a 4R agent, which total length equate the 5R, through the TSL $\mathcal{L}_{\text{circle}}$ using this Euclidian distance between end-effectors as a metric, see Fig. 7.

Tennis Task. In this second environment, designed to evaluate our method in a more complex setting, the goal is to bounce a ball against a vertical wall as many times as possible. The task is initially performed by the teacher, a 4R serial manipulator, the last segment serving as a bat to bounce the ball. The expert dataset is generated by a custom RL-agent trained on this task with an additional signal to specify the ideal impact location and a 1D latent space is used for this task. The tennis skill is then transferred to two students: a 5R and a 3R agent. In this task, all agents have the same total length. This time, the TSL relies on three terms: the Euclidian distance between end-effectors positions, the difference in the orientation of the bats and the difference between the magnitude of the actions (joints speeds):

$$\mathcal{L}_{\text{tennis}} = \mathcal{L}_{\text{circle}} + \mathcal{L}_{\text{ori}} + \underset{i\in[0\ldots n]}{\mathbb{E}}[||A_{i,t} - A_{i,s}||] \tag{13}$$

where $\mathcal{L}_{\text{ori}}$ is the equivalent of $\mathcal{L}_{\text{circle}}$ for the bat orientation differences, $A_{i,t}$ and $A_{i,s}$ are respectively the teacher and student action at time i.

5.2 Models

The VAE model for the teacher agent is composed of an encoder and a decoder and can be seen in Fig. 5. The encoder part is composed of one 64 units hidden layer followed by a split leading to the mean and standard deviation heads, having also 64 units each. These two layers allow generating a code of smaller dimension that is then sent to the decoder. This latter model is a feedforward architecture with two hidden layers, with 64 units each. Both the encoder and the decoder use Tanh non-linearities. We initialize both networks using Xavier initialization [60]. For the transfer case, we remove the encoding part and train only the decoder using the TSL as explained in Sect. 4.2 and shown in Fig. 7.

6 Results

6.1 Circle to Circle Contour Displacement

The teacher model has been trained using mini-batches of 4096 examples, an Adam optimizer [18] with a learning rate of 5×10^{-3} and a weight decay value of 10^{-3}. Figure 9 shows the loss evolution of the training set for the teacher agent and the distribution of the latent representation z of a randomly sampled batch of 4096 examples.

The use of a VAE instead of a *vanilla* auto-encoder has been motivated by the idea that the VAE will create a smooth distribution of the latent representation, instead of having sparse clusters. It is here clearly validated, as can be seen in Fig. 9: in both directions, the z distribution appears to be dense, without obvious holes or clusters. Furthermore, VAEs are known for their disentangling capacities [4, 23], meaning that they tend to create statistically independent latent features. As the expert dataset is composed of trajectories between and along two circles of different radii, we are thus able to generate a 2D controller whose main axes respectively translate into radial and axial movements.

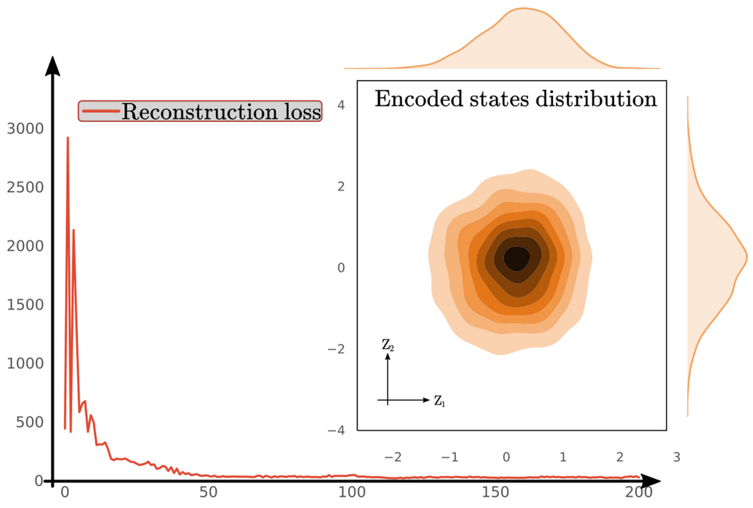

Fig. 9. Reconstruction loss along training epochs and 2D code distribution on the test set for the circle task.

Once the teacher is ready, it becomes possible to create a student agent that mimics the teacher behavior on the task. For this specific task, we use the TSL described in

Eq. 12 that penalizes the student when the variation of its states, TSL metric-wise, differs from the teacher states variation. For similar starting state of the teacher and student agents and given the exact same sequence of user commands $\{z_0, z_1, ..., z_t\}$, the agents ending states should also be similar, TSL metric-wise.

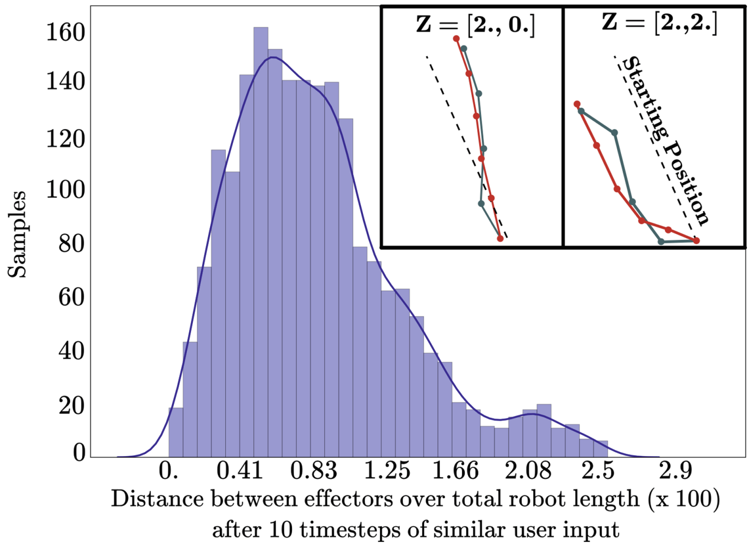

Fig. 10. Distribution of the distance between effectors for 1000 samples after 10 successive actions using the same user input z.

The main plot in Fig. 10 shows the resulting distance between both effectors after 10 successive actions, for a given z and a starting position in which both effectors are located at the same place. It is possible to see most movements result in a distance inferior to 1% of the total body length, an error that is completely acceptable for the tasks considered. In the worst cases, with no user adaptation, the difference is still less than 3% of the total body length, underlining the transfer efficiency.

Additionally, using the experimental conditions described for the distance distribution, we plot in Fig. 10 a small subset of configurations, for a clearer understanding of the movement. In these cases, although the student body configuration may not exactly match the teacher's, their end-effectors end up in a close configuration, as enforced by the TSL, even though there was no closed-loop correction coming from the user. Even though the student has not been taught on a dedicated dataset, it has still been able to generate movements similar to the teacher's, demonstrating the transfer efficiency.

6.2 Tennis Task

The Tennis task has been designed to demonstrate that the TSL is suitable for complex environments, involving physics, external perceptions, and world interactions. The objective is to create an assistive agent to facilitate the control of a serial manipulator playing a game of tennis and allow the user to rely on a single latent variable to get the agent to bounce the ball. This environment is more challenging because it implies a higher reactivity from the agent. Furthermore, writing an analytical model to solve this task proves to be challenging. The first step consists in creating an expert player through RL. To fulfill this step, we used PPO (Proximal Policy Optimization [53]), a battle-tested policy gradient, actor-critic with trust-region reinforcement learning algorithm. Formally, the Tennis MDP components for a 4R robot are:

- The state $s \in \mathbb{R}^9 = [s_{\text{robot}}, s_{\text{ball}}, P_{\text{wall}}]$ where:
  - $s_{\text{robot}} \in \mathbb{R}^4$ holds information about the robot's joints angles
  - $s_{\text{ball}} \in \mathbb{R}^4$ describes the relative ball position and its current speed
  - $P_{\text{wall}} \in \mathbb{R}$ is the vertical target position on the wall used to enhance the distribution of impacts and ultimately have more control over the expert trajectories dataset
- The action $a \in \mathbb{R}^4$: angular speed for each joint
- The reward $r_{\text{tennis}} = \alpha + c \times (\beta + \gamma * \exp(-d))$ where:
  - α is a small constant that incites the agent to keep the ball over a height threshold for as long as possible
  - c is the contact flag. Its value ranges from 0 to 1 when a contact between the ball and the wall is detected, and goes back to 0 at the next step
  - β and γ are constant values used to weight the relative importance of accuracy when touching the wall.
  - d is the vertical offset between the ball and the target on the wall

After 10^5 episodes of training, amounting to 4 h, the 4R agent is effectively able to play and can center its shots around the expected target. Figure 11 shows the evolution of the mean reward through training on the left, and the impact distribution of 100 shots per target for 6 different targets on the right. As we can see, the agent manages to direct the ball around the target and most of the shots land in the vicinity of the expected impact point.

Once the expert is available, the process is similar to the **Circle** task. To train the teacher agent, a balanced (with respect to the impact points) expert trajectories dataset is generated. The teacher VAE model optimizes its parameters to be able to recover the action, while encoding the inputs in a normally-distributed latent representation. In this specific case, the KL loss weight is halved to ensure a better reconstruction. Figure 12 shows the evolution of the reconstruction loss for the training along the epochs, as well as the distribution of the latent encodings for a batch of 2048 examples. Eventually, using the TSL defined by Eq. 13, the task knowledge is distilled within two students: a 5R and a 3R robot.

In this configuration, we assess the controllability and efficiency of the trained student by relying on 4 human players. In our setting, each human player plays five games with each robot in the **Tennis** environment. For each game, we record the number of bounces against the wall. The objective is to maximize the number of bounces. The results of this evaluation are displayed in Fig. 13. As a baseline, we add a plot for the expert, as well as a randomly initialized student agent, for both the 5R and the 3R, controlled by the best human player P0. Although a certain variability can be observed between the players, relative to their personal ability, each of them is able to reach much higher scores than the random baseline, with both students and the teacher. After

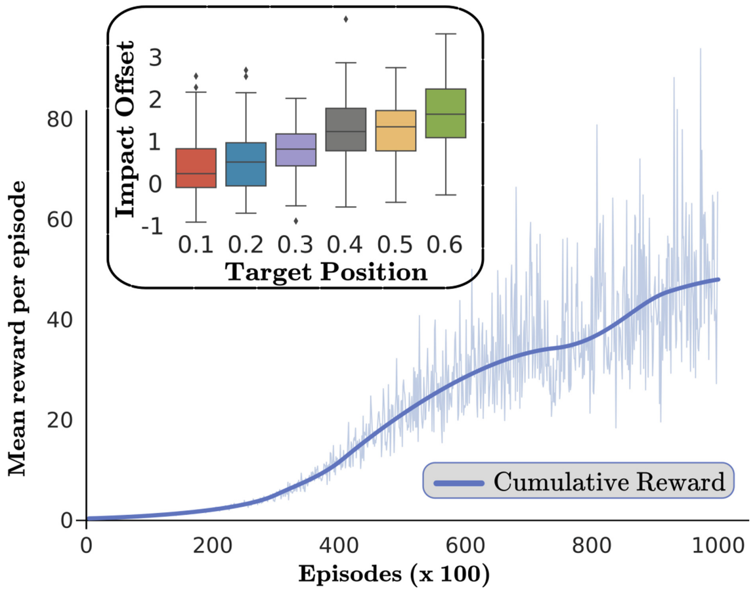

Fig. 11. Mean Cumulative Reward Evolution and ball impact distribution for 100 shots per target point [28].

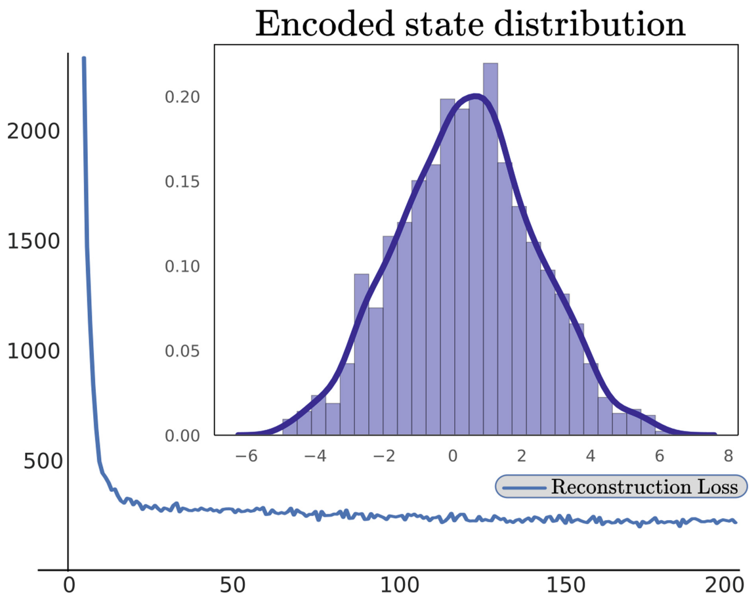

Fig. 12. Tennis Teacher training metrics and code distribution of a randomly sampled batch of 2048 examples.

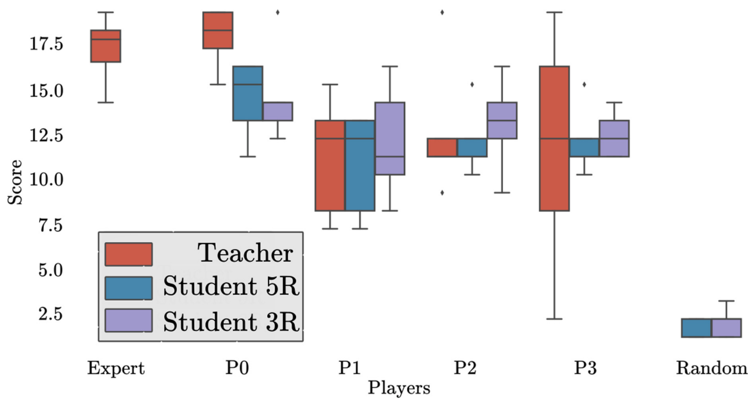

Fig. 13. Comparison of assistive performances of several players using both teacher and student agent with baseline.

switching assistive agents, from teacher to students, most players are able to conserve their average score, except P0 which average score decreases of approximately 15%. However, even these lesser scores are still considerably beyond the random baseline.

This illustrates clearly that transfer through the TSL is effective. Additionally, the three players even decrease the variance of their score when using the student agent, underlining the robustness of the process. However, as it is difficult to assess these performances from sheer numerical values, we provide a video of our results at https://bit.ly/2w4P4e2.

6.3 Training Time and Assistive Aspects

One of the principal appeal in transfer learning is to decrease the time needed to produce a functional model. The TSL approach clearly fits within this framework, as the effective transfer process is quite short on a regular i7 CPU with 64 Gb of RAM. In the **Circle** case, as the expert dataset relies on an inverse kinematics approach, the time needed for generating expert trajectories is negligible. As a result, a user could consider the fact to create an expert with the student structure as well. However, as the teacher training time using the expert data is close to 15 min, the TSL approach nevertheless yields a 66% time reduction in this environment as the transfer only requires 5 min. While it may appear of limited interest given the overall duration of the process, this simple environment allowed us to illustrate the TSL capabilities. The transfer significance is more strongly emphasized within the **Tennis** environment. In this configuration, the RL agent interacts during 4 h with the environment before yielding a suitable expert policy, to which an additional 15 min are necessary to train the teacher agent. In this case, the transfer using the TSL approach can be done in 5 min, that is only 7% of the time initially needed for the teacher, and yet result in performant assistive controllers as demonstrated by our results. This clearly establishes the transfer value and shows that the more the expert dataset generation is costly, the more interesting the transfer of control policies is.

From a qualitative standpoint, this approach yields an important ease-of-use enhancement, particularly in the case of the Tennis task. Indeed, it would be highly challenging and non-intuitive to design and use a controller to manually pilot each rotoid joint of the tennis agent. The assistive method, empowered by the TSL method, allows to easily reach a considerable number of bounces by letting the user focus on a global plan, instead of managing the low-level inputs.

7 Conclusion

This work proposes a new step by step methodology allowing to systematically transfer knowledge of task achievement between two different structural agents. The introduced concept of Task-Specific Loss (TSL) was evaluated over various tasks, within the background of assistive robotics. As can be seen in the results, this method offers considerable time saving compared to state-of-the-art RL methods and can distil the task knowledge of a teacher agent within a structurally different student agent within minutes, as opposed to several hours of training for several other proposed techniques in the literature. Furthermore, our experiments, especially the Tennis task, demonstrate that the transferred knowledge is robust and that various users are able to generate stable performances on a task involving physics, planning and consistency. The TSL application scenarios are numerous: in assistive robotics, as it can be considered to allow disabled people to seamlessly fit their new robotic platform to behave similarly to their previous one. In other robotic domains as well, since learning-based transfer techniques for control tasks are not particularly developed. In future works, we expect to investigate TSL architectures for autonomous control in various complex systems such as dual-manipulation settings or biped walking configurations.

Acknowledgment. This work has been sponsored by the French government research program Investissements d'Avenir through the RobotEx Equipment of Excellence (ANR-10-EQPX-44) and the IMobS3 Laboratory of Excellence (ANR-10-LABX-16-01), by the European Union through the program of Regional competitiveness and employment 2007–2013 (ERDF - Auvergne Region), by the Auvergne region and the French Institute for Advanced Mechanics.

References

1. Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., Mordatch, I.: Emergent tool use from multi-agent autocurricula. In: International Conference on Learning Representations (ICLR) (2020)
2. Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., Mordatch, I.: Emergent complexity via multi-agent competition. International Conference on Learning Representations (ICLR) (2018)
3. Bellemare, M.G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., Munos, R.: Unifying count-based exploration and intrinsic motivation. In: Advances in Neural Information Processing Systems (2016)
4. Burgess, C.P., et al.: Understanding disentangling in β-VAE. In: International Conference on Representation Learning (ICLR) (2018)
5. Cobbe, K., Klimov, O., Hesse, C., Kim, T., Schulman, J.: Quantifying generalization in reinforcement learning. In: Proceedings of Machine Learning Research (MLR) (2018)

6. Dai, Z., Yang, Z., Yang, Y., Carbonell, J.G., Le, Q.V., Salakhutdinov, R.: Transformer-XL: attentive language models beyond a fixed-length context. CoRR abs/1901.02860 (2019). http://arxiv.org/abs/1901.02860
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis (2019). https://doi.org/10.18653/v1/N19-1423. https://www.aclweb.org/anthology/N19-1423
8. Duan, Y., Schulman, J., Chen, X., Bartlett, P.L., Sutskever, I., Abbeel, P.: RL2: fast reinforcement learning via slow reinforcement learning. In: International Conference on Representation Learning (ICRL) (2017)
9. Eysenbach, B., Gupta, A., Ibarz, J., Levine, S.: Diversity is all you need: learning skills without a reward function. In: International Conference on Representation Learning (ICRL) (2019)
10. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning (ICML) (2017)
11. Girshick, R.: Fast R-CNN. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448. IEEE Computer Society, USA (2015). https://doi.org/10.1109/ICCV.2015.169
12. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Computer Vision and Pattern Recognition (CVPR) (2014)
13. Goodfellow, I.J., et al.: Generative adversarial nets. In: International Conference on Neural Information Processing Systems (NeurIPS) (2014)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Computer Vision and Pattern Recognition (CVPR) (2015)
15. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: Association for Computational Linguistics (ACL) (2018)
16. James, S., Johns, E.: 3D simulation for robot arm control with deep Q-learning. arXiv (2016)
17. Kaiser, L., et al.: One model to learn them all. arXiv:170605137v1 (2017)
18. Kingma, D.P., Lei Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980v9 (2017)
19. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations, ICLR Banff, AB, Canada, 14–16 April, Conference Track Proceedings (2014). http://arxiv.org/abs/1312.6114
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS'12, vol. 1, pp. 1097–1105. Curran Associates Inc., Red Hook (2012)
21. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: a lite BERT for self-supervised learning of language representations. In: International Conference on Learning Representations (2020)
22. Lopes, M., Lang, T., Toussaint, M., Yves Oudeyer, P.: Exploration in model-based reinforcement learning by empirically estimating learning progress. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 25, pp. 206–214. Curran Associates, Inc. (2012)
23. Losey, D.P., Srinivasan, K., Mandlekar, A., Garg, A., Sadigh, D.: Controlling assistive robots with learned latent actions. arXiv e-prints arXiv:1909.09674 (2019)
24. Martin, L., et al.: CamemBERT: a tasty French language model (2019)

25. Mnih, V., et al.: Playing atari with deep reinforcement learning. CoRR abs/1312.5602 (2013). http://arxiv.org/abs/1312.5602
26. Mounsif, M., Lengagne, S., Thuilot, B., Adouane, L.: Universal notice network: transferable knowledge among agents. In: 6th 2019 International Conference on Control, Decision and Information Technologies (IEEE-CoDIT) (2019)
27. Mounsif, M., Lengagne, S., Thuilot, B., Adouane, L.: BAM! base abstracted modeling with universal notice network: fast skill transfer between mobile manipulators. In: 7th 2020 International Conference on Control, Decision and Information Technologies (IEEE-CoDIT) (2020)
28. Mounsif, M., Lengagne, S., Thuilot, B., Adouane, L.: CoachGAN: fast adversarial transfer learning between differently shaped entities. In: 17th International Conference on Informatics in Control, Automation and Robotics (ICINCO) (2020)
29. Nachum, O., Norouzi, M., Xu, K., Schuurmans, D.: Bridging the gap between value and policy based reinforcement learning. In: International Conference on Neural Information Processing Systems (NeurIPS) (2017)
30. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. In: International Conference on Representation Learning (ICRL) (2019)
31. OpenAI, Akkaya, I., et al.: Solving Rubik's cube with a robot hand. arXiv e-prints arXiv:1910.07113 (2019)
32. OpenAI, Andrychowicz, M., et al.: Learning dexterous in-hand manipulation. CoRR abs/1808.00177 (2018). http://arxiv.org/abs/1808.00177
33. Peng, X.B., Abbeel, P., Levine, S., van de Panne, M.: DeepMimic: example-guided deep reinforcement learning of physics-based character skills. ArXiv e-prints (2018)
34. Peng, X.B., Andrychowicz, M., Zaremba, W., Abbeel, P.: Sim-to-real transfer of robotic control with dynamics randomization. In: International Conference on Robotic and Automation (ICRA) (2018)
35. Peng, X.B., Berseth, G., Yin, K., van de Panne, M.: DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning. ACM Trans. Graph. (Proc. SIGGRAPH 2017) **36**(4), 1–13 (2017)
36. Peng, X.B., Chang, M., Zhang, G., Abbeel, P., Levine, S.: MCP: learning composable hierarchical control with multiplicative compositional policies. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32, pp. 3681–3692. Curran Associates, Inc. (2019)
37. Peng, X.B., Kanazawa, A., Malik, J., Abbeel, P., Levine, S.: SFV: reinforcement learning of physical skills from videos. ACM Trans. Graph. **37**(6), 1–14 (2018). https://doi.org/10.1145/3272127.3275014
38. Peng, X.B., Kumar, A., Zhang, G., Levine, S.: Advantage-weighted regression: simple and scalable off-policy reinforcement learning. CoRR abs/1910.00177 (2019). https://arxiv.org/abs/1910.00177
39. Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., Abbeel, P.: Asymmetric actor critic for image-based robot learning. In: Kress-Gazit, H., Srinivasa, S.S., Howard, T., Atanasov, N. (eds.) Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 26–30 June 2018 (2018)
40. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Report (2019)
41. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection (2016)
42. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: Computer Vision and Pattern Recognition (CVPR) (2017)
43. Redmon, J., Farhadi, A.: Yolov3: an incremental improvement. CoRR abs/1804.02767 (2018), http://arxiv.org/abs/1804.02767

44. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 28, pp. 91–99. Curran Associates, Inc. (2015)
45. Riedmiller, M.A., et al.: Learning by playing - solving sparse reward tasks from scratch. In: International Conference on Learning Representation (ICLR) (2018)
46. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28. http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a. arXiv:1505.04597 [cs.CV]
47. Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning. ArXiv e-prints (2017)
48. Schmidhuber, J.: Formal theory of creativity, fun, and intrinsic motivation (1990–2010). IEEE Trans. Auton. Ment. Dev. **2**(3), 230–247 (2010)
49. Schmidhuber, J.: A possibility for implementing curiosity and boredom in model-building neural controllers. In: Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats (1991)
50. Schmidhuber, J.: Evolutionary principles in self-referential learning. Ph.D. thesis, Technische Universitat München (1987)
51. Schulman, J., Abbeel, P., Chen, X.: Equivalence between policy gradients and soft Q-learning. CoRR abs/1704.06440 (2017). http://arxiv.org/abs/1704.06440
52. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: Trust region policy optimization. In: International Conference on Machine Learning (ICML) (2015)
53. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR abs/1707.06347 (2017). http://arxiv.org/abs/1707.06347
54. Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(4), 640–651 (2017)
55. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. J. Nat. (2015)
56. Szegedy, C., et al.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9 (2015)
57. Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain randomization for transferring deep neural networks from simulation to the real world. ArXiv e-prints (2017)
58. Wang, J.X., et al.: Learning to reinforcement learn. CoRR abs/1611.05763 (2016). http://arxiv.org/abs/1611.05763
59. Wang, R., Lehman, J., Clune, J., Stanley, K.O.: Paired open-ended trailblazer (POET): endlessly generating increasingly complex and diverse learning environments and their solutions. CoRR abs/1901.01753 (2019). http://arxiv.org/abs/1901.01753
60. Xavier, G., Bengio, Y.: Understanding the difficulty of training deep feedforward neural network. In: International Conference on Artificial Intelligence and Statistics (AISTATS) (2010)
61. Xie, S., Girshick, R.B., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Computer Vision and Pattern Recognition (CVPR) (2017)
62. Yu, T., et al.: Meta-world: a benchmark and evaluation for multi-task and meta-reinforcement learning (2019). https://github.com/rlworkgroup/metaworld
63. Ziebart, B.D., Maas, A., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: Association for Advancement of Artificial Intelligence (AAAI) (2008)
64. Zintgraf, L.M., Shiarlis, K., Kurin, V., Hofmann, K., Whiteson, S.: CAML: fast context adaptation via meta-learning. ArXiv e-prints (2018)