Transfer of Hierarchical Reinforcement Learning Structures for Robotic Manipulation Tasks

Full Research Paper - CSCI-ISAI

Christian Scheiderer *
Technologies and Management of the Digital Transformation
University of Wuppertal
Wuppertal, Germany
scheiderer@uni-wuppertal.de

Malte Mosbach
Information Management in Mechanical Engineering
RWTH Aachen University
Aachen, Germany
malte.mosbach@rwth-aachen.de

Andrés Felipe Posada-Moreno
Information Management in Mechanical Engineering
RWTH Aachen University
Aachen, Germany
andres.posada@ima.rwth-aachen.de

Tobias Meisen
Technologies and Management of the Digital Transformation
University of Wuppertal
Wuppertal, Germany
meisen@uni-wuppertal.de

***Abstract*—While it is apparent that the transfer of knowledge between tasks is beneficial for training efficiency, the application of trained deep reinforcement learning agents to solve new tasks is not trivial. Especially when tasks are differently structured, retraining and fine tuning is not necessarily beneficial. Instead, it is often the most convenient approach to train a new agent from scratch. One potential solution for effectively reusing learned knowledge may be found in hierarchical reinforcement learning. In this paper we investigate the possibility of reusing low-level policies to improve training efficiency when learning manipulation tasks with an industrial robot. We consider four different scenarios and demonstrate for three of them an increased sample efficiency when training a high-level policy on top of pretrained low-level skills. In the fourth scenario we uncover the reason for a failed transfer to be an ambitious higher hierarchy level enforcing a relearning of the low-level skills.**

***Index Terms*—Hierarchical Reinforcement Learning, Transfer Learning, Industrial Robotics**

I. Introduction

Reinforcement learning (RL) is a method to discover strategies for sequential decision-making problems and has greatly expanded its scope of applications through the achievements of deep learning [1], [2]. Despite the recent attention RL has received, its applicability for real-world scenarios, such as robotics, still faces crucial challenges, as classical RL approaches do not scale well with the complexity of a task: On the one hand, an increase in dimensionality of the observation and action space usually results in an exponential increase in required training data. On the other hand, tasks which require long-horizon planning and offer only sparse feedback also drastically increase training effort. The chance of the agent achieving its goal during exploration, a prerequisite for subsequently deducing a solution strategy, diminishes under such circumstances.

One strategy in the pursuit for more sample efficient learning techniques is the concept of hierarchical reinforcement learning (HRL) [3]–[6]. The main idea behind HRL is the explicit separation of the long- and short-term planning of an agent into different policy levels, where each level receives its objectives from the next higher level [7], [8]. The levels are based on different time scales, which leads an agent to use its lower levels for reacting to the current situation, while the higher levels are used for long-term strategy planning.

Besides the ability to solve long-horizon sequential decision-making tasks more effectively through temporal abstraction, the division of an agent into multiple levels introduces straightforward opportunities for transfer learning [9]. Especially when considering the domain of robotics it is plausible that low-level motor skills may be useful for solving a multitude of high level tasks. Thus, it stands to reason that once learned low-level skills should be reused rather than relearned, in order to decrease the amount of training required.

In this work we investigate the transferability of low-level motor skills in HRL. We use Hierarchical Actor-Critic (HAC) [3] to train agents with two hierarchical levels in an industrial robot simulation. Thereafter, the motor skills of the lower level are transferred to new environments containing additional obstacles or intermediate objectives to be achieved and continue the training process. We compare the performance of the transferred agents to agents with the same hierarchical structure, but trained from scratch on the new environments in order to evaluate the benefit of motor-skill transfer. We further exploit the hierarchical structure of the agents to visualize decisions of the higher level in an interpretable way.

*Corresponding author

978-1-7281-7624-6/20/$31.00 ©2020 IEEE
DOI 10.1109/CSCI51800.2020.00091

II. Related Work

A. Hierarchical Actor-Critic Reinforcement Learning

RL describes an approach to train an agent on solving a task in an environment in a trial-and-error fashion [1], [10]. In an iterative procedure, an agent observes the state of the environment, chooses an action and observes the outcome. Given enough observations an agent is able to learn which actions are beneficial for solving a given task. Especially in combination with artificial neural networks, many control tasks such as games [11], resource management [12], and robotics [13] have successfully been learned with reinforcement learning.

Hierarchical Reinforcement Learning refers to algorithms decomposing a given task through abstraction. The essential virtue of this approach is the reduction of complexity of the problem [14]. Learning hierarchical agents is a long-standing pursuit in RL [15], [16]. Despite the potential of approaching problems on multiple levels of temporal abstraction, past efforts were mostly limited to discrete domains or resorted to learning multiple levels only successively in a bottom-up fashion [3]. Recent works have introduced stable, sample-efficient algorithms operating on continuous domains [3], [5], [8], thus making the concept of HRL highly relevant for robotics applications.

In this paper we employ the Hierarchical Actor-Critic (HAC) algorithm, which is based on two main concepts: an underlying hierarchical architecture and hindsight transitions [3]. The underlying hierarchical architecture is depicted in Figure 1 and operates as follows. The high-level policy receives the current state and goal as inputs from the environment and outputs a subgoal for the next lower policy. This level also gets the current state as well as the subgoal from the top-level policy and outputs the next subgoal to further break down the task. This extends to the lowest level. Here the policy takes the current state as an input and the subgoal from the policy above as its goal and outputs a primitive action to be executed in the environment. The levels operate on different timescales, whereby from a level perspective one episode spans over one step of its predecessor.

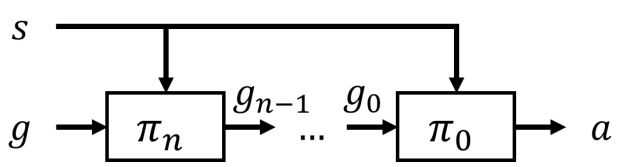


Fig. 1: Structure of a HAC agent. The top level policy π_n observes the current state s and goal g to be achieved. It therefore proposes a subgoal g_{n-1} to be achieved by the subsequent layer. This pattern continues until the lowest level policy π_0, which actually proposes the agents next primitive action a.

An inherent problem of HRL is that the simultaneous training of multiple, nested policies is inherently unstable, as the transition functions for higher levels change due to the adjusting lower levels. In addition, the randomness introduced during exploration of the agent prevents the higher levels on collecting meaningful feedback, as the observed rewards may vary highly with the exploration noise of the lower level policy. To counteract these issues, HAC uses hindsight transitions. The concept of hindsight transitions describes the practice of augmenting the collected experience in order to generate more meaningful training examples for the agent, e.g. by substituting the goal with the achieved state to provide positive feedback in sparse reward environments [3].

Besides being beneficial with respect to the training efficiency, the explicitly defined hierarchical structure offers opportunities for generating interpretable representations of an agents behavior. The idea of analysing learned knowledge by visualizing the value-function of the higher level is based on the findings of *Beyret et al.* in [17].

B. Transfer learning in Hierarchical Reinforcement Learning

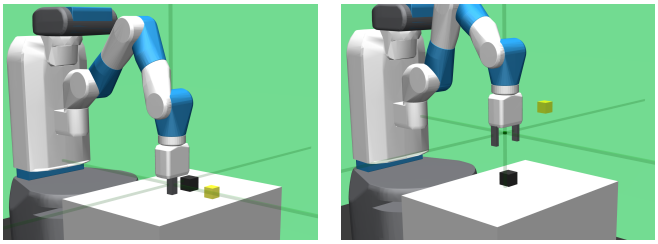
Transfer learning refers to the idea that the behaviour learned in one task is also useful for solving a related, but different task [18]. Generalization appears not only for any individual task, but may happen across related tasks. Then, transferring learned knowledge between tasks should accelerate convergence. Especially in the domain of computer vision, transfer learning has been successfully applied for a multitude of tasks, such as image segmentation [19], object detection [20] or pose estimation [21]. Regarding the use of transfer learning in the domain of HRL, previous works have shown that the reuse of trained higher levels led to a successful transfer of planning capabilities across scenarios [4]. However, it should be noted that the employed HiDE-algorithm [4] was explicitly created in a decomposed structure to foster the composition of new agents from trained modules without retraining. In this work, on the other hand, the transfer of low-level skills through transferring the weights of the bottom level of a general HRL framework will be discussed. Also the transfer of low level skills was previously investigated, whereby a KL-regularization was used to constrain the lower policy to previously learned behavior [22]. By utilizing the KL regularized expected reward objective, policies are optimized to a trade-off between expected reward and closeness to a default policy. Consequently, our approach differs in the method of incorporating prior knowledge into transfer learning agents.

III. Experimental Setup

A. Learning Environments

The conducted experiments are based on tasks introduced by Plappert et al. [23] and on variations of said problems. They all utilize the MuJoCo physics engine [24]. The two environments FetchPush-v1 and FetchPickAndPlace-v1 are selected as baseline tasks, which are depicted in Figure 2. The only modification is an adaptation of the goal visualizations. The state-space is 25-dimensional and contains information on the robot's gripper and the object. The 4-dimensional action-space comprises the desired change of the 3-dimensional Cartesian position of the tool center point, as well as one

value specifying the opening of the robots gripper. Goals are expressed as the desired 3-dimensional Cartesian position of the object. All tasks use sparse rewards, where the agent gets a reward of 1 if the object's position matches the goal up to a threshold of 5 cm, and a reward of 0 otherwise.

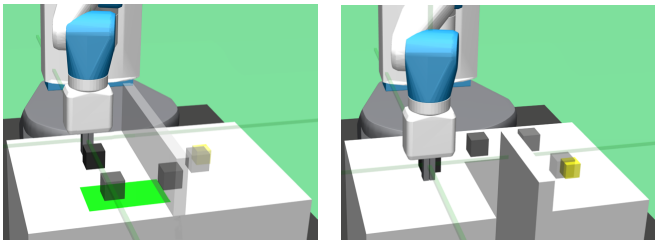


(a) FetchPush: The box must be pushed across the table to the goal position.

(b) FetchPickAndPlace: The box must be picked up and moved to the goal position.

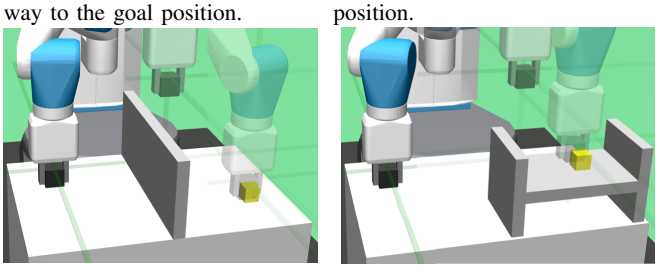
Fig. 2: Baseline environments taken from [23].

In order to investigate the transfer of policies between related, yet different tasks, we introduce two variations for each of the two original environments. As shown in Figure 3, the new scenarios all include obstacles or intermediate objectives that increase the difficulty of the original task.



(a) Push-Gate: The cube must be pushed onto the green area to drive down the barrier and give way to the goal position.

(b) Push-Gap: The cube must be pushed around a gap cut out from the table and then to the goal position.



(c) PickAndPlace-Wall: The cube must be moved up and over the barrier.

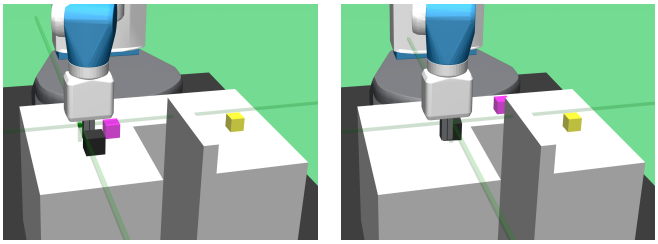
(d) PickAndPlace-Table: The cube must be placed on top of an elevated structure.

Fig. 3: Learning environments used for evaluating the transfer of low level policies.

The Push-Gate problem is characterized by a wall that halves the table top. The agent must first move the cube onto the green button to trigger the opening of the gate before they pursue the end-goal. Push-Gap requires the agent to bypass a gap in the tabletop when navigating to the goal. In the PickAndPlace-Wall environment, the agent must learn to lift the cube over a barrier before placing it on the other side, while in the PickAndPlace-Table task the cube must be placed on a table. All variations require solutions which deviate from the shortest-distance path, and thusly deviate from the optimal solution in the base environments.

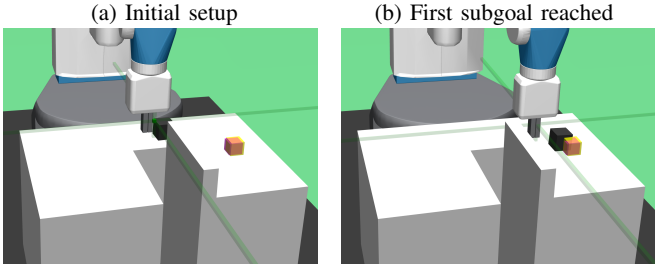
B. Agent Architecture

The architecture of an agent and the interaction of the agent's two levels are based on HAC. The highest level of the algorithm is called *subgoal level* and introduces temporal abstraction by breaking down an initial objective into a sequence of subgoals. In our scenarios, the goal is to move a box to a specific location. The subgoals are intermediate positions which are easier to reach, while at the same time advancing the box towards the desired end goal. We allow for a maximum number of 10 time steps to reach a subgoal, whereas one episode consists of a maximum of 50 time steps for all environments. The *action level* must learn how to achieve subgoals by manipulating the gripper of the robot. This knowledge on how to move the cube to given subgoals (fundamental motor skills of the problem) is transferred from the baseline tasks to the new environments. The desired behavior of an agent is depicted exemplary for the Push-Gap task in Figure 4, highlighting the working principle and decomposable nature of the hierarchical approach.



(a) Initial setup

(b) First subgoal reached



(c) Second subgoal reached

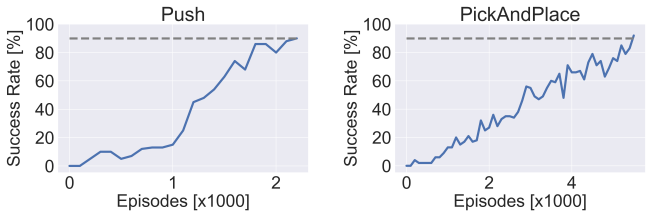
(d) end-goal reached

Fig. 4: A hierarchical agent solves the Push-Gap task. The higher level generates subgoals (purple) for the lower level, which learns suitable robot movements.

C. Pretraining and Transfer

Before the actual comparison of performance on the new environments can be made, agents have to be trained on the baseline environments to create transferable lower level modules. The training performance can be seen in Figure 5. Since previous tests showed that achieving higher success

rates may require much more training, we deliberately set a conservative performance threshold of a 90% success rate as stop criterion for pretraining. This way we guarantee that the pretraining will terminate after a reasonable amount of episodes, which is crucial when including the pretraining in the total amount of training effort required. With this setup we pretrain the lower level for 2,200 episodes for the pushing tasks and 5,500 episodes for the pick and place tasks.



(a) Pretraining on FetchPush (b) Pretraining on FetchPickAndPlace

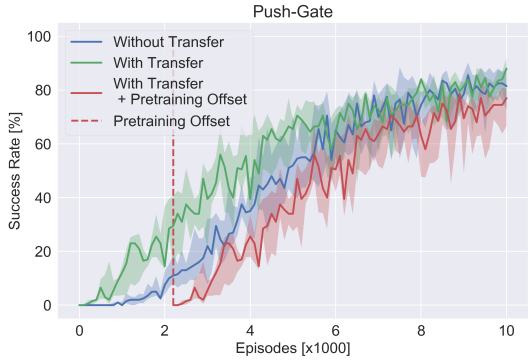
Fig. 5: Training on the baseline environments to create the transferable lower level modules.

After training on the baseline tasks, the weights of the lower level is transferred to the respective task variations, where training is resumed. After every 100 learning episodes, the agents performance is evaluated by running 100 test runs with the current policy and determining the average success rate. In order to be able to evaluate the effects of the knowledge transfer comparatively, agents were additionally trained from scratch for all environments. For each scenario 16 runs were conducted in parallel to account for deviations in performance due to the random nature of RL.

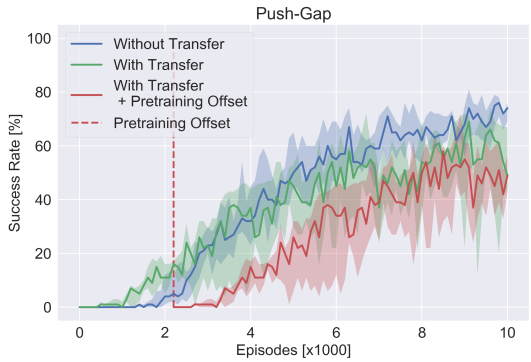
IV. Results and Discussion

The results of our experiments are depicted in Figure 6. The success rate when learning from scratch is plotted in blue and the performance of the transfer learning agents are shown in green. The latter is also shifted by the amount of training required for pretraining as reported in chapter III-C, resulting in the red curves. It must be noted that this depiction must be interpreted as a lower bound for the benefit of the transfer, as it solely considers a one-to-one transfer. However, as one pretrained policy was reused in multiple scenarios, the fixed costs for pretraining lose significance with each additional transfer.

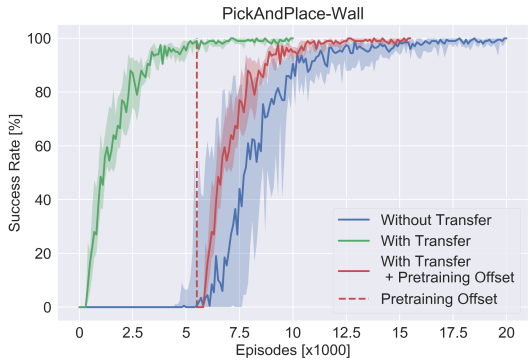
In the Push-Gate task shown in Figure 6a, transfer learning agents have a steeper learning curve than agents that learn from scratch. They exceed a success rate of 50% after 3,300 episodes of training, while regular agents require 4,900 epochs to reach this level. However, TL agents only have a definite advantage until about episode 6,000, after which both variants perform at a similar level. The Push-Gap environment (Figure 6b) does not show a clear transfer learning gain. Although TL agents progress faster in the first 2,000 episodes, there is no long-term benefit, with agents trained from scratch even outperforming TL agents after episode 5,000.



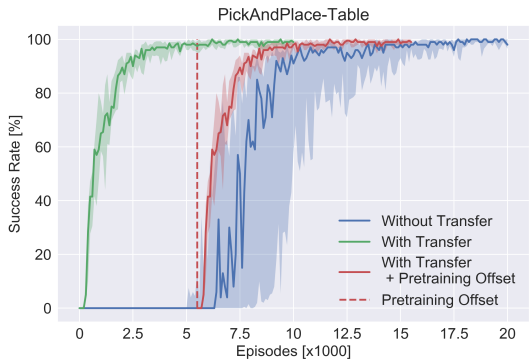
(a) Push-Gate



(b) Push-Gap



(c) PickAndPlace-Wall



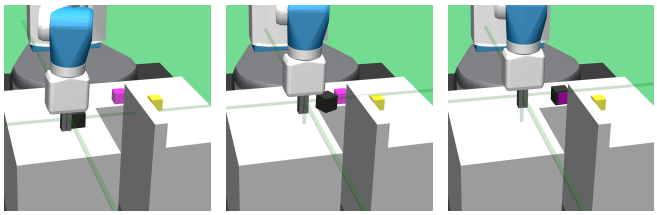
(d) PickAndPlace-Table

Fig. 6: Results of the comparative evaluation of transfer learning and regular agents on the four environments

The variations of the FetchPickAndPlace-v1 task (Figures 6c and 6d) show a significant transfer benefit in both cases. When learning from scratch, most test runs do not make any progress inside the first 5,000 episodes. The agents with transferred lower level modules however show immediate improvement on both tasks. This benefit remains even when accounting for the cost of pretraining. In addition to a faster training, the transfer fostered a more stable and consistent training behavior.

Due to varying results regarding transferability for our scenarios, we take a closer look at the Push-Gap scenario, which was the least successful. We hereby exploit the explicit hierarchical structure of the agent to gain insights into the decision-making process. By determining the expected state-action values of the higher level through sampling the critic network and visualizing them for individual states, we are able to deduce the agents behavior.

The Push-Gap task appears to share significant structures of the original pushing environment, which makes the poor transferability surprising. We find that transfer learning agents as well as agents learning the task from scratch converge to similar behaviors. As illustrated in Figure 7, the agents all learned to skip the cube across the gap.



(a) The robot hits the cube at high velocity. (b) The cube skips over the gap. (c) The cube reaches the subgoal.

Fig. 7: The agents learn to skip the cube over the gap instead of moving around.

Figure 8 depicts the value-function on the tabletop of a trained agent for the initial position and clearly shows that the agents choose a goal which is behind the gap. While this behavior risks the cube falling into the gap, this solution allows the higher level policy of the agent to complete the task faster compared to going safely around the gap.

In order to successfully skip the cube across the gap, the lower level must hit the cube at a high velocity. We find that this requirement is not initially met by the pretrained lower level. This is not surprising considering the baseline task used for pretraining does not include any obstacles and can be solved with a more moderate strategy.

Contrary to the expectation of the higher level taking the shortcomings of the pretrained lower level into account and proposing more conservative subgoals around the gap, our observations imply that the higher level pressured the lower level to learn the high risk-high reward strategy. While the lower level is able to eventually adapt, the learning curves show clearly that this behavior adjustment is inferior to learning the desired strategy from scratch.

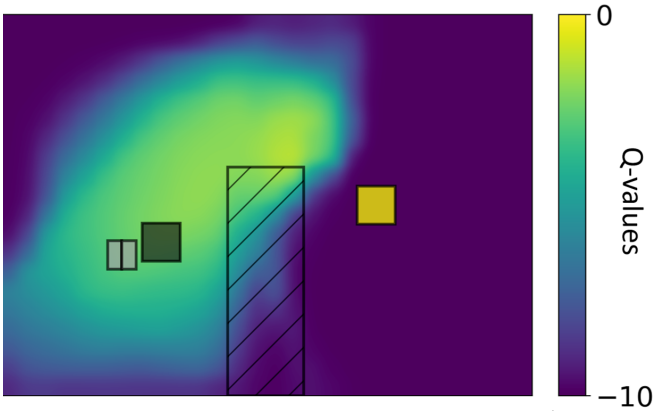


Fig. 8: Evaluation of the Q-value function of the higher level of a transfer learning agent trained on Push-Gap.

With these findings we can also explain why the transfer to the Push-Gate task was more successful. As the environment does not allow for any shortcuts, at least none that neither we nor the agents have found, the low-level skill of pushing the cube in straight lines without accounting for cube velocity is usable without significant adjustments by the higher level.

V. Conclusion and Future Work

As demonstrated in this paper, the inherent skill-division of HRL agents introduces opportunities for knowledge transfer across related environments. By training on a simpler version of a task we were able to pretrain reusable low-level skills. We found that this approach can significantly improve sample efficiency, even when the cost of pretraining is taken into account. However, we also found that allowing the continued training of the pretrained low-level skills may result in the higher level pushing towards strategies which require significantly different low-level behavior. In such cases it is more beneficial to learn the desired low-level skills from scratch, instead of adapting the pretrained policy.

Building upon our findings, we expect that the identified shortcomings may be dealt with by constraining the adaption of the lower level. This should effectively force the higher level to find a suitable conservative, even if not optimal, solution first. This approach might be extended by eventually relaxing the constraints on the lower level, as to allow the agent to strive for more optimal solutions.

Further, as the transferability significantly depends on the task used for pretraining, we believe that a decomposition into task- and robot-specific levels is crucial for the robust transfer of hierarchical structures. The positive effect of this distinct modularization on transfer was already shown by Devin et al. [25] and Sharma et al. [26] proposed an unsupervised approach to discovering robotic movement skills . We believe that further research should investigate the creation and integration of task-agnostic skills as method to further decouple the hierarchical levels and towards universal transferability.

REFERENCES

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[2] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. *arXiv preprint arXiv:2003.13350*, 2020.

[3] Andrew Levy, Robert Platt Jr., and Kate Saenko. Hierarchical actor-critic. *CoRR*, abs/1712.00948, 2017.

[4] Sammy Christen, Lukas Jendele, Emre Aksan, and Otmar Hilliges. Learning functionally decomposed hierarchies for continuous control tasks with path planning, 2020.

[5] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *CoRR*, abs/1805.08296, 2018.

[6] Josh Merel, Matthew Botvinick, and Greg Wayne. Hierarchical motor control in mammals and machines. *Nature communications*, 10(1):1–12, 2019.

[7] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278, 1993.

[8] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *CoRR*, abs/1703.01161, 2017.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[10] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[11] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[12] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56, 2016.

[13] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[14] Long-Ji Lin. Hierarchical learning of robot skills by reinforcement. In *IEEE International Conference on Neural Networks*, pages 181–186 vol.1, 1993.

[15] Richard Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112, 05 1999.

[16] Jürgen Schmidhuber. Learning to generate subgoals for action sequences. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pages 453 vol.2–, 1991.

[17] Benjamin Beyret, Ali Shafti, and A. Aldo Faisal. Dot-to-dot: Achieving structured robotic manipulation through hierarchical reinforcement learning. *CoRR*, abs/1904.06703, 2019.

[18] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(56):1633–1685, 2009.

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[20] Joseph Lim, Russ Salakhutdinov, and Antonio Torralba. Transfer learning by borrowing examples for multiclass object detection. In *Advances in neural information processing systems*, pages 118–126, 2011.

[21] Carl Doersch and Andrew Zisserman. Sim2real transfer learning for 3d human pose estimation: motion to the rescue. In *Advances in Neural Information Processing Systems*, pages 12949–12961, 2019.

[22] Dhruva Tirumala, Hyeonwoo Noh, Alexandre Galashov, Leonard Hasenclever, Arun Ahuja, Greg Wayne, Razvan Pascanu, Yee Whye Teh, and Nicolas Heess. Exploiting hierarchy for learning and transfer in kl-regularized RL. *CoRR*, abs/1903.07438, 2019.

[23] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR*, abs/1802.09464, 2018.

[24] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[25] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2169–2176, 2017.

[26] Archit Sharma, Michael Ahn, Sergey Levine, Vikash Kumar, Karol Hausman, and Shixiang Gu. Emergent real-world robotic skills via unsupervised off-policy reinforcement learning. *arXiv preprint arXiv:2004.12974*.