Learning by Playing – Solving Sparse Reward Tasks from Scratch

Martin Riedmiller [*1] **Roland Hafner** [*1] **Thomas Lampe** [1] **Michael Neunert** [1] **Jonas Degrave** [1]
Tom Van de Wiele [1] **Volodymyr Mnih** [1] **Nicolas Heess** [1] **Tobias Springenberg** [1]

Abstract

We propose Scheduled Auxiliary Control (SAC-X), a new learning paradigm in the context of Reinforcement Learning (RL). SAC-X enables learning of complex behaviors – from scratch – in the presence of multiple sparse reward signals. To this end, the agent is equipped with a set of general auxiliary tasks, that it attempts to learn simultaneously via off-policy RL. The key idea behind our method is that active (learned) scheduling and execution of auxiliary policies allows the agent to efficiently explore its environment – enabling it to excel at sparse reward RL. Our experiments in several challenging robotic manipulation settings demonstrate the power of our approach. A video of the rich set of learned behaviours can be found at https://youtu.be/mPKyvocNe_M.

1. Introduction

Consider the following scenario: a learning agent has to control a robot arm to open a box and place a block inside. While defining the reward for this task is simple and straightforward (e.g. using a simple mechanism inside the box such as a force sensor to detect a placed block), the underlying learning problem is hard. The agent has to discover a long sequence of "correct" actions in order to find a configuration of the environment that yields the sparse reward – the block contained inside the box. Discovering this sparse reward signal is a hard exploration problem for which success via random exploration is highly unlikely.

Over the last decades, a multitude of methods have been developed to help with the above mentioned exploration problem. These include for example: shaping rewards (Ng et al., 1999; Randløv & Alstrøm, 1998; Gu et al., 2017), curriculum learning (Heess et al., 2017; Ghosh et al., 2018; Forestier et al., 2017), transfer of learned policies from simulation to reality (see Duan et al. (2017); Sadeghi et al. (2017); Tobin et al. (2017); Rusu et al. (2017); Hanna & Stone (2017) for recent examples), learning from demonstrations (Ross et al., 2011; Vecerik et al., 2017; Kober & Peters, 2011; Sermanet et al., 2017; Nair et al., 2017), learning with model guidance, see e.g. Montgomery & Levine (2016), or inverse RL (Ng & Russell, 2000; Ziebart et al., 2008). All of these approaches rely on the availability of prior knowledge that is specific to a task. Moreover, they often bias the control policy in a certain – potentially suboptimal – direction. For example, using a shaped reward designed by the experimenter, inevitably, biases the solutions that the agent can find. In contrast to this, when a sparse task formulation is used, the agent can discover novel and potentially preferable solutions. We would thus, arguably, prefer to develop methods that support the agent during learning but preserve the ability of the agent to learn from sparse rewards. Ideally, our new methods should reduce the specific prior task knowledge that is required to cope with sparse rewards.

In this paper, we introduce a new method dubbed Scheduled Auxiliary Control (SAC-X, with X denoting the scheduler type), as a first step towards such an approach. It is based on four main principles:

1. Every state-action pair is paired with a vector of rewards, consisting of (typically sparse) externally provided rewards and (typically sparse) internal auxiliary rewards.
2. Each reward entry has an assigned policy, called intention in the following, which is trained to maximize its corresponding cumulative reward.
3. There is a high-level scheduler which selects and executes the individual intentions with the goal of improving performance of the agent on the external tasks.
4. Learning is performed off-policy (and asynchronously from policy execution) and the experience between intentions is shared – to use information effectively.

Although the approach proposed in this paper is generally applicable to a wider range of problems, we discuss our method in the light of a typical robotics manipulation application with sparse rewards: stacking various objects and cleaning a table.

Auxiliary rewards in these tasks are defined based on the

[*]Equal contribution [1]Google DeepMind, London, GB. Correspondence to: Martin Riedmiller <riedmiller@google.com>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

mastery of the agent to control its own sensory observations (e.g. images, proprioception, touch sensors). They are designed to be easily implementable in a real robot setup. In particular, we define auxiliary rewards on a raw sensory level – e.g. whether a touch is detected or not. Or, alternatively, define them on a higher level that requires a small amount of pre-computation of entities, e.g. whether any object moved or whether two objects are close to each other in the image plane. Based on these basic auxiliary tasks, the agent must effectively explore its environment until more interesting, external rewards are observed; an approach which is inspired by the playful phase of childhood in humans.

We demonstrate the capabilities of SAC-X in simulation on challenging robot manipulation tasks such as stacking and tidying a table-top using a robot arm. All tasks are defined via sparse, easy to define, rewards and solved using the same set of auxiliary reward functions. In addition, we demonstrate that our method is sample efficient, allowing us to learn from scratch on a real robot.

2. Related Work

The idea of using auxiliary tasks in the context of reinforcement learning has been explored several times in literature. Among the first papers to mention this idea, is the work by (Sutton et al., 2011); where general value functions are learned for a large collection of pseudo-rewards corresponding to different goals extracted from the sensorimotor stream. General value functions have recently been extended to Deep RL in work on Universal Value Function Aproximators (UVFA) (Schaul et al., 2015). These, in turn, are inherently connected to learning to predict the future via "successor" representations (Dayan, 1993; Kulkarni et al., 2016b; Barreto et al., 2017) or forecasts (Schaul & Ring, 2013; Lample & Chaplot, 2017; Dosovitskiy & Koltun, 2017) which are trained to be predictive of features extracted from future states. In contrast to the setting explored in this paper, all of the aforementioned approaches do not utilize the learned sub-policies to drive exploration for an external "common" goal. They also typically assume independence of policies. In a similar vein to the UVFA approach, recent work on Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) proposed to generate many tasks for a reinforcement learning agent by randomly sampling goals along previously experienced trajectories. Our approach can be understood as an extension of HER to semantically grounded, and scheduled, goals.

A related strand of research has considered learning a shared representation for multiple RL tasks. Closest to the ideas presented in this paper and serving as the main inspiration for our approach, is the recent work on Deep Reinforcement Learning with the UNREAL agent (Jaderberg et al., 2017) and Actor Critic agents for navigation (Mirowski et al., 2016) (discrete action control) as well as the Intentional Unintentional Agent (Cabi et al., 2017) (considering continuous actions). While these approaches mainly consider using auxiliary tasks to provide additional learning signals – and additional exploration by following random sensory goals – we here make active use of the auxiliary tasks by switching between them throughout individual episodes (to achieve exploration for the main task).

Our work is also connected to the broader literature on multitask (reinforcement) learning (see e.g. Caruana (1997) for a general overview and Lazaric et al. (2008); Mehta et al. (2008) for RL applications) and work on reinforcement learning via options (Dietterich, 1998; Bacon et al., 2017; Daniel et al., 2012). In contrast to these approaches we here learn skills that are semantically grounded via auxiliary rewards, instead of automatically discovering a decomposed solution to a single task.

The approach we take for scheduling the learning and execution of different auxiliary tasks can be understood from the perspective of "teaching" a set of increasingly more complicated problems – see e.g. the literature on curriculum learning (Bengio et al., 2009) – where we consider a fixed number of problems and learn a teaching policy online. Research on this topic has a long history, both in the machine learning and psychology literature. Recent examples from the field of RL include the PowerPlay algorithm (Schmidhuber, 2013), that invents and teaches new problems on the fly, as well as research on learning complex tasks via curriculum learning for RL – with either fixed (Heess et al., 2017) or automatically generated curricula (Narvekar et al., 2017) – and hierarchical learning of real robot tasks (Forestier et al., 2017). (Hierarchical) Reinforcement Learning with the help of so called "intrinsic motivation" rewards (Chentanez et al., 2005; Singh et al., 2009) has, furthermore, been studied for controlling real robots by Ngo et al. (2012) and combined with Deep RL techniques by Kulkarni et al. (2016a); Dilokthanakul et al. (2017). In contrast to our work these approaches typically consider internal measures such as learning progress to define rewards, rather than auxiliary tasks that are grounded in physical reality.

3. Preliminaries

We consider the problem of Reinforcement Learning (RL) in a Markov Decision Process (MDP) . We make use of the following basic definitions: Let $\mathbf{s} \in \mathbb{R}^S$ be the state of the agent in the MDP $\mathcal{M}$ – we use the term state and observation of the state (e.g. proprioceptive features, object positions or images) interchangeably to simplify notation. Denote with $\mathbf{a} \in \mathbb{R}^A$ the continuous action vector and $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ the probability density of transitioning to state $\mathbf{s}_{t+1}$ when executing action $\mathbf{a}_t$ in $\mathbf{s}_t$. All actions are assumed to be sampled from a policy distribution $\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})$, with parameters

$\boldsymbol{\theta}$. After executing an action – and transitioning in the environment – the agent receives a scalar reward $r_{\mathcal{M}}(\mathbf{s}_t, \mathbf{a}_t)$.

With these definitions in place, we can define the goal of Reinforcement Learning as maximizing the sum of discounted rewards $\mathbb{E}_\pi[R(\tau_{0:\infty})] = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \mid a_t \sim \pi(\cdot|\mathbf{s}_t), \mathbf{s}_{t+1} \sim p(\cdot|\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_0 \sim p(\mathbf{s})]$, where $p(\mathbf{s})$ denotes the state visitation distribution, and we use the short notation $\tau_{t:\infty} = \{(\mathbf{s}_t, \mathbf{a}_t), \dots\}$ to refer to the trajectory starting in state t. For brevity of notation, we will omit the dependence of the expectation on samples from the transition and state visitation distribution where unambiguous.

4. Scheduled Auxiliary Control

We will now introduce our method for RL in sparse reward problems. For the purpose of this paper, we define a sparse reward problem as finding the optimal policy π^* in a main MDP $\mathcal{M}$ with a reward function that is characterized by an 'ϵ-region' in state space. That is we have

$$r_{\mathcal{M}}(\mathbf{s}, \mathbf{a}) = \begin{cases} \delta_{\mathbf{s}_g}(\mathbf{s}) & \text{if } d(\mathbf{s}, \mathbf{s}_g) \leq \epsilon \\ 0 & \text{else,} \end{cases} \tag{1}$$

where $\mathbf{s}_g$ denotes a goal state, $d(\mathbf{s}, \mathbf{s}_g)$ denotes the distance between the goal state and the current state s – defined on a subset of the variables comprising s and measured according to some metric, i.e. we could have $d(\mathbf{s}, \mathbf{s}_g) = \|\mathbf{s} - \mathbf{s}_g\|_2$. Further, $\delta_{\mathbf{s}_g}(\mathbf{s})$ defines the reward surface within the epsilon region; in this paper we will choose the most extreme case where ϵ is small and we set $\delta_{\mathbf{s}_g}(\mathbf{s}) = 1$ (constant).[1]

4.1. A Hierarchical RL Approach for Learning from Sparse Rewards

To enable learning in the setting described above we derive an algorithm that augments the sparse learning problem with a set of low-level auxiliary tasks.

Formally, let $\mathcal{A} = \{\mathcal{A}_1, \dots, \mathcal{A}_K\}$ denote the set of auxiliary MDPs. In our construction, these MDPs share the state, observation and action space as well as the transition dynamics with the main task $\mathcal{M}$,[2] but have separate auxiliary reward functions $r_{\mathcal{A}_1}(\mathbf{s}, \mathbf{a}), \dots, r_{\mathcal{A}_K}(\mathbf{s}, \mathbf{a})$. We assume full control over the auxiliary rewards; i.e. we assume knowledge of how to compute auxiliary rewards and assume we can evaluate them at any state action pair. Although this assumption might appear restrictive at first glance, we will – as mentioned before – make use of simple auxiliary rewards that can be obtained from the activation of the agents sensors.

Given the set of reward functions we can define intention policies and their return as $\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}, \mathcal{T})$ and

$$\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s},\mathcal{T})}\Big[R_{\mathcal{T}}(\tau_{t:\infty})\Big] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s},\mathcal{T})}\Big[\sum_{t=0}^{\infty} \gamma^t r_{\mathcal{T}}(\mathbf{s}_t, \mathbf{a}_t)\Big], \tag{2}$$

where $\mathcal{T} \in \mathfrak{T} = \mathcal{A} \cup \{\mathcal{M}\}$, respectively.

To derive a learning objective based on these definitions it is useful to first remind ourselves what the aim of such a procedure should be: Our goal for learning is to both, i) train all auxiliary intentions policies and the main task policy to achieve their respective goals, and ii) utilize all intentions for fast exploration in the main sparse-reward MDP $\mathcal{M}$. We accomplish this by defining a hierarchical objective for policy training that decomposes into two parts.

Learning the intentions The first part is given by a joint policy improvement objective for all intentions. We define the action-value function $Q_{\mathcal{T}}(\mathbf{s}_t, \mathbf{a}_t)$ for task $\mathcal{T}$ as

$$Q_{\mathcal{T}}(\mathbf{s}_t, \mathbf{a}_t) = r_{\mathcal{T}}(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\pi_{\mathcal{T}}}\Big[R_{\mathcal{T}}(\tau_{t+1:\infty})\Big], \tag{3}$$

where we have introduced the short-hand notation $\pi_{\mathcal{T}} = \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{x}, \mathcal{T})$. Using this definition we define the (joint) policy improvement objective as finding $\arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ where θ is the collection of all intention parameters and,

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}; \mathcal{M}) + \sum_{k=1}^{|\mathcal{A}|} \mathcal{L}(\boldsymbol{\theta}; \mathcal{A}_k), \tag{4}$$

$$\text{with } \mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = \sum_{\mathcal{B} \in \mathfrak{T}} \mathop{\mathbb{E}}_{p(s|\mathcal{B})}\Big[Q_{\mathcal{T}}(\mathbf{s}, \mathbf{a}) \mid \mathbf{a} \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}, \mathcal{T})\Big]. \tag{5}$$

That is, we optimize each intention to select the optimal action for its task starting from an initial state drawn from the state distribution $p(\mathbf{s}|\mathcal{B})$, *obtained by following any other policy* $\pi(\mathbf{a}|\mathbf{s}, \mathcal{B})$ with $\mathcal{B} \in \mathfrak{T} = \mathcal{A} \cup \{\mathcal{M}\}$ (the task which we aimed to solve before). We note that this change is a subtle, yet important, departure from a multi-task RL formulation. By training each policy on states sampled according to the state visitation distribution of each possible task we obtain policies that are "compatible" – in the sense that they can solve their task irrespective of the state that the previous intention-policy left the system in. This is crucial if we want to safely combine the learned intentions.

Learning the scheduler The second part of our hierarchical objective is concerned with learning a scheduler that sequences intention-policies. We consider the following setup: Let ξ denote the period at which the scheduler can switch between tasks.[3] Further denote by H the to-

[1]Instead, we could also define a small reward gradient within the ϵ-region to enforce precise control by setting, for example, $\delta_{\mathbf{s}_g}(\mathbf{s}) = \exp(-d(\mathbf{s}, \mathbf{s}_g))$.

[2]We note that in the experiments we later also allow for multiple external (main) tasks, but omit this detail here for clarity of the presentation.

[3]We choose $\xi = 180$ in our experiments. In general h should span multiple time-steps to enforce commitment to one task.

tal number of possible task switches (including the initial intention choice) within an episode[4] and denote by $\mathcal{T}_{0:H-1} = \{\mathcal{T}_0, \dots, \mathcal{T}_{H-1}\}$ the H scheduling choices made within an episode. We can define the return of the main task given these scheduling choices as

$$R_{\mathcal{M}}(\mathcal{T}_{0:H-1}) = \sum_{h=0}^{H} \sum_{t=h\xi}^{(h+1)\xi-1} \gamma^t r_{\mathcal{M}}(\mathbf{s}_t, \mathbf{a}_t), \quad \text{where } \mathbf{a}_t \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_t, \mathcal{T}_h). \tag{6}$$

Denoting the scheduling policy with $P_{\mathcal{S}}(\mathcal{T}|\mathcal{T}_{0:h-1})$ we can define the probability of an action $\mathbf{a}_t$, when behaving according to the scheduler, as

$$\pi_{\mathcal{S}}(\mathbf{a}_t|\mathbf{s}_t, \mathcal{T}_{0:h-1}) = \sum_{\mathcal{T}} \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t, \mathcal{T}) P_{\mathcal{S}}(\mathcal{T}|\mathcal{T}_{0:h-1}), \tag{7}$$

from which we can sample in two steps (as in Eq. (6)) by first choosing a sub-task every ξ steps and then sampling an action from the corresponding intention; we note that the chosen sub-task can either be an auxiliary or the main task. Combining these two definitions, the objective $\mathcal{L}(\mathcal{S})$ for learning a scheduler $\mathcal{S}$ – by finding the solution $\arg\max_{\mathcal{S}} \mathcal{L}(\mathcal{S})$ – reads:

$$\mathcal{L}(\mathcal{S}) = \mathbb{E}_{P_{\mathcal{S}}}\Big[R_{\mathcal{M}}(\mathcal{T}_{0:H-1}) \mid \mathcal{T}_h \sim P_{\mathcal{S}}(\mathcal{T}|\mathcal{T}_{0:h-1})\Big]. \tag{8}$$

Note that, for the purpose of optimizing the scheduler, we consider the individual intentions as fixed in Equation (8) – i.e. we do not optimize it w.r.t. $\boldsymbol{\theta}$ – since we would otherwise be unable to guarantee preservation of the individual intentions (which are needed to efficiently explore in the first place). We also note that the scheduling policy, as defined above, ignores the dependency on the state $\mathbf{s}_{h\xi}$ in which a task is scheduled (i.e. $P_{\mathcal{S}}$ uses a partially observed state). In addition to this learned scheduler we also experiment with a version that schedules intentions at random throughout an episode, which we denote with SAC-U. Note that such a strategy is not as naïve as it initially appears: due to the fact that we allow several intentions to be scheduled within an episode they will naturally provide curriculum training data for each other. A successful 'move object' intention will, for example, leave the robot arm in a position close to the object, making it easy for a lift intention to discover reward.

As mentioned in Section 2 the problem formulation described above bears similarities to several other multi-task RL formulations. In particular we want to highlight that it can be interpreted as a generalization of the IUA and UNREAL objectives (Cabi et al., 2017; Jaderberg et al., 2017) to stochastic continuous controls – in combination with active execution of auxiliary tasks and (potentially learned) scheduling within an episode. It can also be understood as a hierarchical extension of Hindsight Experience Replay (Andrychowicz et al., 2017), where the agent behaves according to a fixed set of semantically grounded auxiliary tasks – instead of following random goals – and optimizes over the task selection.

[4]We consider a finite horizon setting in the following to simplify the presentation.

4.2. Policy Improvement

To optimize the objective from Equation (5) we take a gradient based approach. We first note that learning for each intention $\pi(\mathbf{a}|\mathbf{s}, \mathcal{T})$, as defined in Equation (5), necessitates an off-policy treatment – since we want each policy to learn from data generated by all other policies. To establish such a setup we assume access to a parameterized predictor $\hat{Q}^{\pi}_{\mathcal{T}}(\mathbf{s}, \mathbf{a}; \phi)$ (with parameters ϕ) of state-action values; i.e. $\hat{Q}^{\pi}_{\mathcal{T}}(\mathbf{s}, \mathbf{a}; \phi) \approx Q^{\pi}_{\mathcal{T}}(\mathbf{s}, \mathbf{a})$ – as described in Section 4.3. Using this estimator, and a replay buffer B containing trajectories τ gathered from all policies, the policy parameters $\boldsymbol{\theta}$ can be updated by following the gradient

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \approx \sum_{\substack{\mathcal{T}\in\mathfrak{T} \\ \tau\sim B}} \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\substack{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_t,\mathcal{T}) \\ \mathbf{s}_t\in\tau}} \Big[\hat{Q}^{\pi}_{\mathcal{T}}(\mathbf{s}_t, \mathbf{a}; \phi) + \alpha \log \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}_t, \mathcal{T})\Big], \tag{9}$$

where $\mathbb{E}_{\pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_t,\mathcal{T})}[\log \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}_t, \mathcal{T})]$ corresponds to an additional (per time-step) entropy regularization term (with weighting parameter α). This gradient can be computed via the reparametrization trick (Rezende et al., 2014; Kingma & Welling, 2014), for policies whose sampling process is differentiable (such as the Gaussian policies used here), as described in work on stochastic value gradients (Heess et al., 2015). We refer to the supplementary for details.

In contrast to the intention policies, the scheduler has to quickly adapt to changes in the incoming stream of experience data – since the intentions change over time and hence the probability that any intention triggers the main task reward is highly varying during the learning process. To account for this, we choose a simple parametric form for the scheduler: Assuming a discrete set of tasks $\mathfrak{T}$ we can first realize that the solution

$$P_{\mathcal{S}} = \arg\max_{P_{\mathcal{S}}} \mathcal{L}(\mathcal{S}), \tag{10}$$

to Equation (8) can be approximated by the Boltzmann distribution

$$P_{\mathcal{S}}(\mathcal{T}_h|\mathcal{T}_{1:h-1}; \eta) = \frac{\exp(\mathbb{E}_{P_{\mathcal{S}}}[R_{\mathcal{M}}(\mathcal{T}_{h:H})]/\eta)}{\sum_{\bar{\mathcal{T}}_{h:H}} \exp(\mathbb{E}_{P_{\mathcal{S}}}[R_{\mathcal{M}}(\bar{\mathcal{T}}_{h:H})]/\eta)}, \tag{11}$$

where the temperature parameter η dictates the greediness of the schedule; and hence $\lim_{\eta\to 0} P_{\mathcal{S}}(\mathcal{T}|\mathcal{T}_{1:h-1}; \eta)$ corresponds to the optimal policy (the solution from (10)) at any scheduling point. To be precise, the Boltzmann policy corresponds to maximizing $\mathcal{L}(\mathcal{S})$ together with an additional entropy regularizer on the scheduler.

This distribution can be represented via an approximation of the schedule returns $Q(\mathcal{T}_{1:h-1}, \mathcal{T}_h) \approx \mathbb{E}_{P_\mathcal{S}}[R_\mathcal{M}(\mathcal{T}_{h:H})|\mathcal{T}_{1:h-1}]$. For a finite, small, number of tasks – as in this paper – $Q(\mathcal{T}_{1:h-1}, \mathcal{T}_h)$ can be represented in tabular form. Specifically, we form a Monte Carlo estimate using the last $M = 50$ executed trajectories;

$$Q(\mathcal{T}_{0:h-1}, \mathcal{T}_h) = \frac{1}{M} \sum_{i=1}^{M} R^\tau_\mathcal{M}(\mathcal{T}_{h:H}), \tag{12}$$

where $R^\tau_\mathcal{M}(\mathcal{T}_{h:H})$ is the cumulative discounted return along trajectory τ (computed as in Equation (6) but with fixed states and action choices).

Using the improved policy from Equation (9) and the scheduler from Equation (11) we can then collect additional data, following the scheduled action distribution $\pi_\mathcal{S}(\mathbf{a}|\mathbf{s}, \mathcal{T}_{q:h-1})$.

4.3. Policy Evaluation

We use Retrace (Munos et al., 2016) for off-policy evaluation of all intentions. Concretely, we train parametric Q-functions (neural networks) $\hat{Q}^\pi_\mathcal{T}(\mathbf{s}, \mathbf{a}; \phi)$ by minimizing the following loss, defined on data from the replay B:

$$\begin{aligned}
&\min_\phi L(\phi) = \mathbb{E}_{(\tau, b, \mathcal{B}) \sim B}\Big[\big(\hat{Q}^\pi_\mathcal{T}(\mathbf{s}, \mathbf{a}; \phi) - Q^{\text{ret}}\big)^2\Big], \text{with} \\
&Q^{\text{ret}} = \sum_{j=i}^{\infty} \Big(\gamma^{j-i} \prod_{k=i}^{j} c_k\Big)\Big[r_\mathcal{T}(s_j, a_j) + \delta_Q(\mathbf{s}_i, \mathbf{s}_j)\Big], \\
&\delta_Q(\mathbf{s}_i, \mathbf{s}_j) = \gamma \mathbb{E}_{\pi_{\theta'}(\mathbf{a}|\mathbf{s}, \mathcal{T})}[Q^\pi_\mathcal{T}(\mathbf{s}_i, \cdot; \phi')] - Q^\pi_\mathcal{T}(\mathbf{s}_j, \mathbf{a}_j; \phi'), \\
&c_k = \min\Big(1, \frac{\pi_{\theta'}(\mathbf{a}_k|\mathbf{s}_k, \mathcal{T})}{b(\mathbf{a}_k|\mathbf{s}_k, \mathcal{B})}\Big),
\end{aligned} \tag{13}$$

where τ denotes a trajectory (together with action choices and rewards) sampled from the replay buffer, b denotes a behaviour policy under which the data was generated, and $\mathcal{B}$ is the task the behaviour policy tried to accomplish. We again highlight that b was not necessarily aiming to achieve task $\mathcal{T}$ for which $\hat{Q}_\mathcal{T}$ should predict action-values. The importance weights c_k then weight the actions selected under the behavior policy with their probability under π. Here ϕ' and θ' denote the parameters of target policy and Q-networks (Mnih et al., 2015), which are periodically exchanged with the current parameters ϕ, θ. This is common practice in Deep-RL algorithms to improve learning stability.

5. Experiments

To benchmark our method we perform experiments based on a Kinova Jaco robot arm in simulation and on hardware.

5.1. Experimental Setup

In all experiments the auxiliary tasks are chosen to provide the agent with information about how well it is exploring its own sensory space. They are easy to compute and are general – in the sense that they transfer across tasks. They are defined over all available sensor modalities. For proprioception, for example, we choose to maximize / minimize joint angles, for the touch sensors we define tasks for activating / deactivating finger touch or force-torque sensors. In image space, we define auxiliary tasks on the object level (i.e. 'move red object' or 'place red object close to green object in camera plane'). All these predicates can be easily computed and mapped to a sparse reward signal (as in Equation (1)). A full list of rewards is given in the supplementary.

We present learning results for SAC-X – where X denotes the scheduler type – with the two schedulers described in Section 4.1: a sequentially uniform scheduler (SAC-U) and SAC-X with a learned scheduler (SAC-Q). In ablation studies, we also investigate a non-scheduling version of our setup, where we strictly followed the policy that optimizes the external reward. Since this procedure is similar to the one used by the IU agent (Cabi et al., 2017) – but enhanced with retrace and stochastic policies to ensure an even comparison –, we denote this variant with 'IUA' in the following. As a strong off-policy learning baseline we also include a comparison to DDPG (Lillicrap et al., 2016).

All simulation experiments use raw joint velocities (9 DOF) as control signals at 50 ms time steps. Episodes lasted for 360 steps in total with scheduler choices every $\xi = 180$ steps, *resulting in two choices per episode*. We note that allowing for only two intention switches within an episode is sufficient in our setup since the **intentions form a hierarchy** – hence a lift intention can accomplish in one scheduling cycle what touch and move could do in two (once sufficient data for learning lift has been acquired). Observations consist of proprioceptive information (joint angles, joint velocities) of the arm and sensor information coming from a virtual force-torque sensor in the wrist, virtual finger touch sensors and simulated camera images. We provide results for both learning from raw pixels and from extracted features (i.e. pose and velocities of the objects contained in the scene) we refer to the supplementary material for details on the policy architecture. All experiments are repeated with 5 different random seeds; learning curves report the median performance among the runs (with shaded areas marking the 5% and 95% quantiles respectively).

To speed up experimentation, all simulation results are obtained in an off-policy learning setup where data is gathered by **multiple agents (36 actors)** which send collected experience to a pool of 36 learners. This setup is explained in more detail in the supplementary material. While this is a compromise on data-efficiency – trading it off with wall-clock time – our real world experiments, in which a single robot is the only data source, reveal that SAC-X can be very data-efficient.

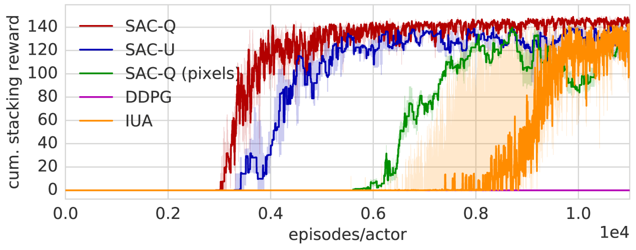

Figure 1. Cumulative reward for the extrinsic task of stacking block one on block two. Both SAC-U and SAC-Q learn the task reliably. The reference experiment using DDPG fails completely (flat line). The IUA approach learns slower and less reliably. Note that all results were obtained via 36 actors and learners.

5.2. Stacking Two Blocks

For our initial set of algorithm comparisons we consider the task of stacking a block on top of another, slightly larger, object. This constitutes a challenging robotics task as it requires the agent to acquire several core abilities: grasping the first block placed arbitrarily in the workspace, lifting it up to a certain height, precisely placing it on top of the second block. In addition, the agent has to find a stable configuration of the two blocks. The expected behavior is shown in the bottom image sequence in Figure 4. We use a sparse reward for a successful stack: the stack reward is one if the smaller object is only in contact with other objects in the scene, but not with the robot or the ground. Otherwise the reward is zero. In addition to this main task reward the agent has access to the standard set of auxiliary rewards, as defined in the supplementary material.

Figure 1 shows a comparison between SAC-X and several baselines in terms of average stacking reward. As shown in the plot, both SAC-U (uniform scheduling) and SAC-Q reliably learn the task for all seeds. SAC-U reaches a good performance after around 5000 episodes per actor, while SAC-Q is faster and achieves a slightly better final performance – thanks to its learned scheduler. To demonstrate that our method is powerful enough to learn policies and action-value functions from raw images, we performed the same stacking experiment with information of the block positions replaced by two camera images of the scene – that are processed by a CNN and then concatenated to the proprioceptive sensor information (see supplementary material for details). The results of this experiment reveal that while learning from pixels (SAC-Q (pixels)) is slower than from features, the same overall behaviour can be learned.

In the no scheduling case, i.e. when the agent follows its behaviour policy induced by the external reward ('IUA'), the IUA agent achieves occasional successes in the first half of the experiment, followed by late learning of the task. Presumably learning is still possible since the shared layers in the policy network bias behaviour towards touching/lifting the brick (and Retrace propagates rewards along trajectories

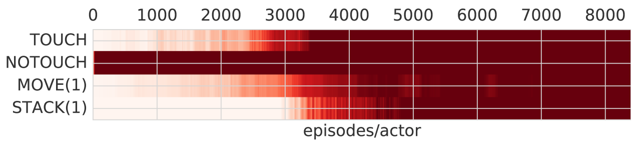

Figure 2. Learning times for a subset of the 13 auxiliary intentions we used in the SAC-Q approach and for the external stacking task. Red color codes for reward. First the agent learns to interact with the objects by touching them and moving them around, then more complex intentions can be learned until, finally, stacking is learned.

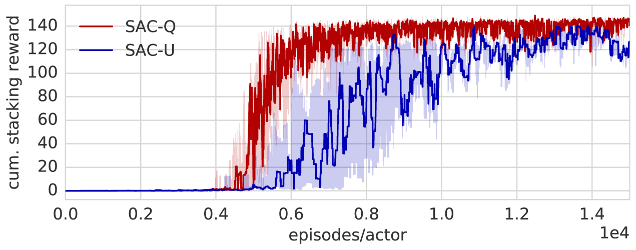

Figure 3. Comparison, in terms of cumulative reward, between SAC-Q and SAC-U for the 'banana' stacking experiment.

quickly, once observed). But the variability in the learning process is much higher and learning is significantly slower. Finally, DDPG fails on this task; the reason being that a stacking reward is extremely unlikely to be observed by pure random exploration and therefore DDPG can not gather the data required for learning. Both results support the core conjecture: scheduling and execution of auxiliary intentions enables reliable and successful learning in sparse reward settings. Figure 2 gives some insight into the learning behaviour, plotting a subset of the learned intentions (see the supplementary for all results). The agent first learns to touch (TOUCH) or stay away from the block (NOTOUCH) then it learns to move the block and finally stack it.

5.3. Stacking a 'Banana' on Top of a Block

Using less uniform objects than simple blocks poses additional challenges, both for grasping and for stacking: some object shapes only allow for specific grasps or are harder to stack in a stable configuration. We thus perform a second experiment in which a banana shaped object must be placed on top of a block. For an approach relying on shaping rewards, this would require careful re-tuning of the shaping. With SAC-X, we can use the same set of auxiliary tasks.

Figure 3 depicts the results of this experiment. Both SAC-U and SAC-Q can solve the task. In this case however, the advantages of a learning scheduler that focuses on solving the external task become more apparent. One explanation for this is that stacking the banana does require a careful fine-tuning of the stacking policy – on which the learned scheduler naturally focuses.

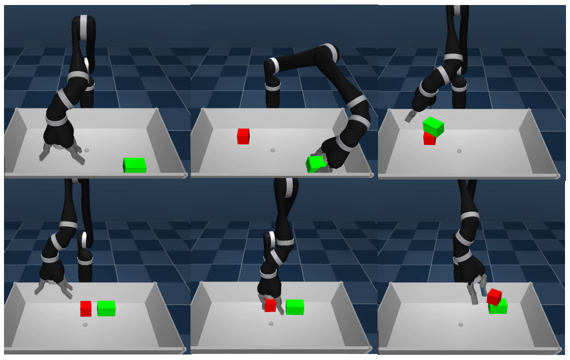

Figure 4. Depiction of the agent stacking two blocks in either configuration, red above green or vice-versa.

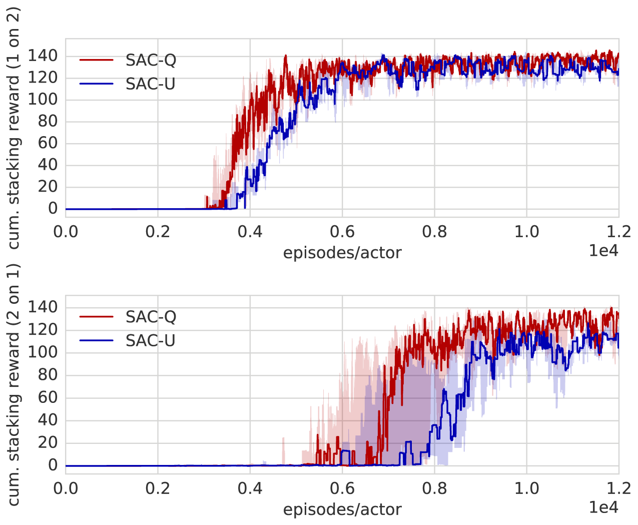

Figure 5. Comparison, in terms of learning speed, between SAC-Q and SAC-U for the two block stacking task.

5.4. Stacking Blocks Both Ways

Next we extend the stacking task by requiring the agent to learn both: stacking the small red block on the large green block (1 on 2 in the Figure 4) as well as vice-versa (2 on 1 in the figure); which is a harder stacking task. This is an example of an agent learning multiple external tasks at once. To cope with multiple external tasks, we learn multiple schedulers (one per task) and pick between them at random (assuming external tasks have equal importance).

Both SAC-U and SAC-Q are able to accomplish the external tasks from pure rewards (see Figure 5). As is also apparent from the figure, the SAC-X agent makes efficient use of its replay buffer: Compared to the initial stacking experiment (Section 5.2), which required 5000 episodes per actor, SAC-X only requires 2500 additional episodes per actor to learn the additional task. In addition to this quantitative evaluation, we note that the observed behaviour of the learned agent also exhibits intuitive strategies to deal with complicated situations. For example, if the agent is started in a situation where block one is already stacked on block two, it has learned to first put block one back on the table, and then stack block two on top of the first block - all in one single policy (please also see the supplementary video at 0:50 mins for a demonstration).

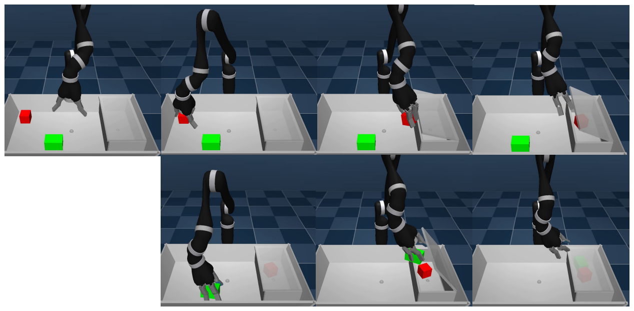

Figure 6. The 'clean-up' task. The images depict a trajectory (left-to-right, top-to-bottom) of the final behaviour for the 'put all in box' intention.

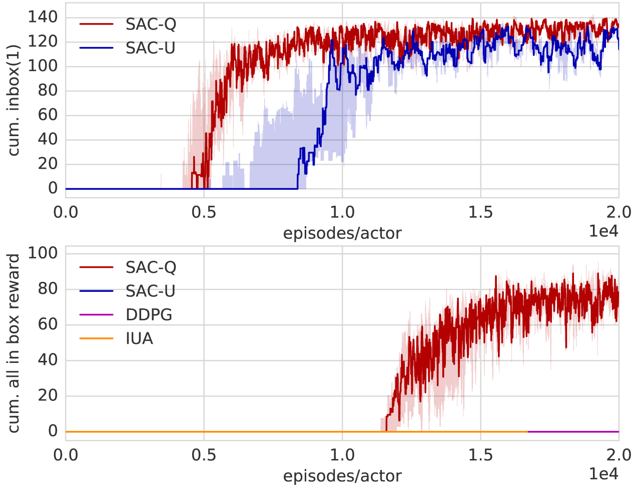

Figure 7. Learning for the cleanup task, shown is the most difficult external task where two blocks are required to be in the box to get a reward signal. SAC-Q is the only successful approach (bottom). SAC-U here 'only' learns to put a single block into the box (top).

5.5. The 'clean-up' Task

The clean-up task (see Figure 6) is an example where a sequence of specific movements have to be executed in order to solve the task. In addition to the two different sized blocks from the last experiments, we add a new object to the scene: a static box with a lid that can be opened.

We rely on the same auxiliary tasks as in the stack blocks experiment, adding one additional sparse auxiliary intention for each object in relation to the box: 'bring object above and close to the box'. In contrast to previous experiments,

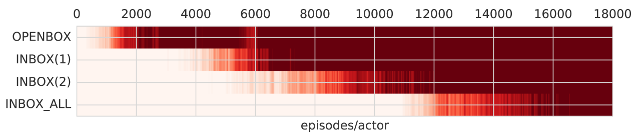

Figure 8. Expected reward in the 'clean-up' experiment, SAC-Q learns alll four extrinsic tasks reliably. In addition it reliably learns to also solve the 15 auxiliary tasks (not depicted here).

we now have 4 sparse extrinsic tasks and corresponding intention policies: i) open the box (OPENBOX in the Figure), ii) put object 1 in box (INBOX(1)), iii) put object 2 in box (INBOX(2)), and iv) put all objects in the box (INBOX_ALL). With a total of 15 auxiliary and 4 extrinsic tasks, this is the most complex scenario presented in this paper. Figure 7 shows a comparison between SAC-X and baselines for this task. Remarkably, even though the reward for placing the objects into the box can *only be observed once they are correctly placed*, SAC-Q learns all extrinsic tasks (see also Figure 8 and the supplementary for a detailed comparison to SAC-U), and the auxiliary tasks, reliably and can interpolate between intention policies (see supplementary video). All baselines fail in this setting, indicating that SAC-X is a significant step forward for sparse reward RL.

5.6. Learning from Scratch on the Real Robot

For learning on the real robot, we consider two tasks: lifting a block and a bring task. We first checked the feasibility of both tasks in simulation by learning using a single actor run in real-time. Using SAC-X, both tasks can be successfully learned from pure rewards with full 9 DOF raw joint velocity control. The learning time on the real robot however would have been the equivalent of several days of non-stop experimentation on the real robot. For practical feasibility we therefore made the following adaptations: we used a cartesian controller for velocity based control of the hand plus one control action for actuation of two fingers, resulting in a 4 dimensional continuous control vector. Note that the proprioceptive information provided to the controller still consist of the joint positions and velocities.

In the lift experiment three auxiliary rewards were defined (rewarding the robot for closing fingers, opening fingers and proximity to the brick). The learning curves, depicted in Figure 9 (top), reveal that using a single robot arm SAC-Q successfully learns to lift after about 1200 episodes, requiring about 10 hours of learning time on the real robot. When tested on about 50 trials on the real robot, the agent is 100% successful in achieving the lifting task.

In an even more challenging setup, we trained SAC-Q to also place the block at a given set of locations in its workspace; adding additional tasks that reward the agent for reaching said location. Again, learning was successful (see Figure 9, bottom), and the agent showed robust, non-trivial control

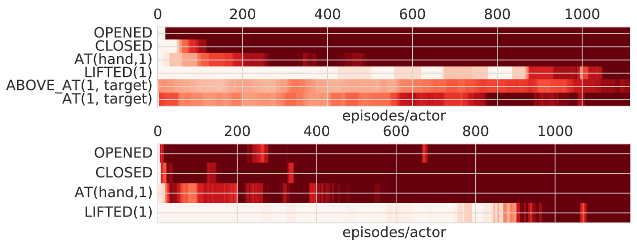

Figure 9. Learning statistics for a real robot experiment for the bring (top) and lift (bottom) task. As before, red indicates reward within an episode (averaged over the last 10), and we plot successes for all used auxiliary tasks (see the supplementary for a detailed listing).

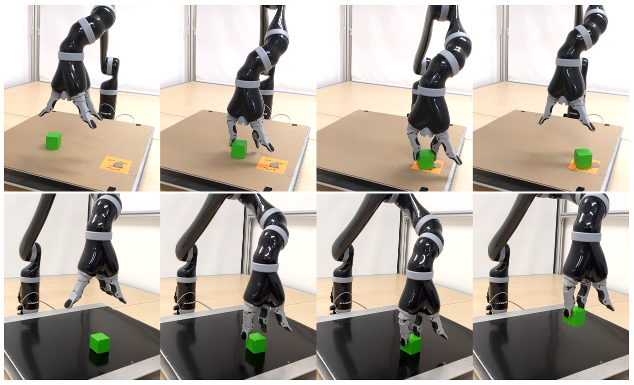

Figure 10. Image sequence depicting a trained SAC-Q agent on the real robot solving the bring (top) and lift (bottom) task.

behavior: The resulting policy developed various techniques for achieving the task including dragging and pushing the block with one finger as well as lifting and carrying the block to the goal location. Furthermore, the agent learned to correct the block position of imprecisely placed objects and learned to move the gripper away once the task is completed. This reactive and rich control behaviour is due to the closed-loop formulation of our approach.

6. Conclusion

This paper introduces SAC-X, a method that simultaneously learns intention policies on a set of auxiliary tasks, and actively schedules and executes these to explore its observation space - in search for sparse rewards of externally defined target tasks. Utilizing simple auxiliary tasks enables SAC-X to learn complicated target tasks from rewards defined in a 'pure', sparse, manner: only the end goal is specified, but not the solution path. We demonstrated the power of SAC-X on several challenging robotics tasks in simulation, using a common set of simple and sparse auxiliary tasks and on a real robot. The learned intentions are highly reactive, reliable, and exhibit a rich and robust behaviour. We consider this as an important step towards the goal of applying RL to real world domains.

Acknowledgements

The authors would like to thank Yuval Tassa, Tom Erez, Jonas Buchli, Dan Belov and many others of the DeepMind team for their help and numerous useful discussions and feedback throughout the preparation of this manuscript.

References

Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5055–5065, 2017.

Bacon, P., Harb, J., and Precup, D. The option-critic architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., Silver, D., and van Hasselt, H. P. Successor Features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems 30 (NIPS)*. 2017.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *International Conference on Machine Learning, ICML*, 2009.

Cabi, S., Colmenarejo, S. G., Hoffman, M. W., Denil, M., Wang, Z., and de Freitas, N. The Intentional Unintentional Agent: Learning to solve many continuous control tasks simultaneously. In *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, Proceedings*, pp. 207–216, 2017.

Caruana, R. Multitask learning. *Machine Learning*, 1997.

Chentanez, N., Barto, A. G., and Singh, S. P. Intrinsically Motivated reinforcement learning. In *Advances in Neural Information Processing Systems 17 (NIPS)*. 2005.

Daniel, C., Neumann, G., and Peters, J. Hierarchical relative entropy policy search. In *Fifteenth International Conference on Artificial Intelligence and Statistics*, JMLR Proceedings, 2012.

Dayan, P. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 1993.

Dietterich, T. G. The maxq method for hierarchical reinforcement learning. In *In Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, 1998.

Dilokthanakul, N., Kaplanis, C., Pawlowski, N., and Shanahan, M. Feature control as intrinsic motivation for hierarchical reinforcement learning. *CoRR*, arXiv:abs/1705.06769, 2017.

Dosovitskiy, A. and Koltun, V. Learning to act by predicting the future. In *International Conference on Learning Representations (ICLR)*, 2017.

Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1087–1098, 2017.

Forestier, S., Mollard, Y., and Oudeyer, P. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

Ghosh, D., Singh, A., Rajeswaran, A., Kumar, V., and Levine, S. Divide-and-conquer reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2018.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Hanna, J. and Stone, P. Grounded action transformation for robot learning in simulation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S. M. A., Riedmiller, M. A., and Silver, D. Emergence of locomotion behaviours in rich environments. *arXiv:abs/1707.02286*, 2017.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Unreal: Reinforcement learning with unsupervised auxiliary tasks. In *ICLR 2017*, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)*, 2014.

Kober, J. and Peters, J. Policy search for motor primitives in robotics. *Machine Learning*, 2011.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical Deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems (NIPS)*, 2016a.

Kulkarni, T. D., Saeedi, A., Gautam, S., and Gershman, S. J. Deep Successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016b.

Lample, G. and Chaplot, D. S. Playing FPS games with deep reinforcement learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Lazaric, A., Restelli, M., and Bonarini, A. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Repepresentations (ICLR)*, 2016.

Mehta, N., Natarajan, S., Tadepalli, P., and Fern, A. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 2008.

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A. J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., and Hadsell, R. Learning to Navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518, 2015.

Montgomery, W. and Levine, S. Guided policy search as approximate mirror descent. 2016.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. G. Safe and efficient off-policy reinforcement learning. In *Neural Information Processing Systems (NIPS)*, 2016.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Overcoming exploration in reinforcement learning with demonstrations. *CoRR*, arXiv:abs/1709.10089, 2017.

Narvekar, S., Sinapov, J., and Stone, P. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

Ng, A. Y. and Russell, S. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, 2000.

Ng, A. Y., Harada, D., and Russell, S. J. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, 1999.

Ngo, H. Q., Luciw, M. D., Förster, A., and Schmidhuber, J. Learning skills from play: Artificial curiosity on a katana robot arm. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012.

Randløv, J. and Alstrøm, P. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, 1998.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014.

Ross, S., Gordon, G. J., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 15 of *JMLR Proceedings*, pp. 627–635. JMLR.org, 2011.

Rusu, A. A., Vecerik, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R. Sim-to-real robot learning from pixels with progressive nets. In *1st Annual Conference on Robot Learning (CoRL)*, 2017.

Sadeghi, F., Toshev, A., Jang, E., and Levine, S. Sim2real view invariant visual servoing by recurrent control. *arXiv preprint arXiv:1712.07642*, 2017.

Schaul, T. and Ring, M. B. Better Generalization with Forecasts. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Beijing, China, 2013.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal Value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Schmidhuber, J. PowerPlay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. In *Front. Psychol.*, 2013.

Sermanet, P., Xu, K., and Levine, S. Unsupervised perceptual rewards for imitation learning. In *Robotics: Science and Systems XIII (RSS)*, 2017.

Singh, S., Lewis, R. L., and Barto, A. G. Where do rewards come from? In Taatgen, N. and van Rijn, H. (eds.), *Proceedings of the 31st Annual Conference of the Cognitive Science Society*. Austin, TX, 2009.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '11, pp. 761–768, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0-9826571-6-1, 978-0-9826571-6-4.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. A. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv:abs/1707.08817*, 2017.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.