Scheduled Intrinsic Drive:
A Hierarchical Take on Intrinsically Motivated Exploration

Jingwei Zhang * 1 Niklas Wetzel * 1 Nicolai Dorka * 1 Joschka Boedecker 1 Wolfram Burgard 1

Abstract
Exploration in sparse reward reinforcement learning remains a difficult open challenge. Many state-of-the-art methods use intrinsic motivation to complement the sparse extrinsic reward signal, giving the agent more opportunities to receive feedback during exploration. Most commonly, these signals are added as bonus rewards, which results in the mixture policy faithfully conducting neither exploration nor task fulfillment for an extended amount of time. In this paper, we instead learn separate intrinsic and extrinsic task policies and schedule between these different drives to accelerate exploration and stabilize learning. Moreover, we introduce a new type of intrinsic reward denoted as successor feature control (SFC), which is general and not task-specific. It takes into account statistics over complete trajectories and thus differs from previous methods that only use local information to evaluate intrinsic motivation. We evaluate our proposed scheduled intrinsic drive (SID) agent using three different environments with pure visual inputs: VizDoom, DeepMind Lab and OpenAI Gym classic control from pixels. The results show a greatly improved exploration efficiency with SFC and the hierarchical usage of the intrinsic drives. A video of our experimental results can be found at https://youtu.be/4ZHcBo7006Y.

1. Introduction
Reinforcement learning (RL) agents learn on evaluative feedbacks, requiring only reward signals rather than ground truth labels as needed in instructive feedbacks (Sutton & Barto, 2018). This goes one step further on the automation of developing intelligent problem-solving agents. With deep networks as powerful function approximators bringing traditional RL into high-dimensional domains, deep reinforcement learning (DRL) has shown great potential (Mnih et al., 2015; 2016; Schulman et al., 2017; Horgan et al., 2018). However, the success of DRL often relies on carefully shaped dense extrinsic reward signals. Although shaping extrinsic rewards can greatly support the agent in finding solutions and shortening the interaction time, designing such dense extrinsic signals often requires substantial domain knowledge, and calculating them typically requires ground truth state information, both of which is hard to obtain in the context of robots acting in the real world. When not carefully designed, the reward shape could sometimes serve as bias or even distraction and could potentially hinder the discovery of optimal solutions. More importantly, learning on dense extrinsic rewards goes backwards on the progress of reducing supervision and could prevent the agent from taking full advantage of the RL framework.

In this paper, we consider terminal reward RL settings, where a signal is only given when the final goal is achieved. When learning with only an extrinsic terminal reward indicating the task at hand, intelligent agents are given the opportunity to potentially discover optimal solutions even out of the scope of the well established domain knowledge. However, in many real-world problems defining a task only by a terminal reward means that the learning signal can be extremely sparse. The RL agent would have no clue about what task to accomplish until it receives the terminal reward for the first time by chance. Therefore in those scenarios guided and structured exploration is crucial, which is where intrinsically-motivated exploration (Oudeyer & Kaplan, 2008; Schmidhuber, 2010) has recently gained great success (Pathak et al., 2017; Burda et al., 2018b). Most commonly in current state-of-the-art approaches, an intrinsic reward is added as a reward bonus to the extrinsic reward. Maximizing this combined reward signal, however, results in a mixture policy that neither acts greedily with regard to extrinsic reward maximization nor to exploration. Furthermore, the non-stationary nature of the intrinsic signals could potentially lead to unstable learning on the combined reward. In addition, current state-of-the-art methods have been mostly looking at local information calculated out of 1-step lookahead for the estimation of the intrinsic rewards,
e.g., one step prediction error (Pathak et al., 2017), or network distillation error of the next state (Burda et al., 2018b). Although those intrinsic signals can be propagated back to earlier states with temporal difference (TD) learning, it is not clear that this results in optimal long-term exploration. We seek to address the aforementioned issues as follows:

1. We propose a hierarchical agent \textit{Scheduled Intrinsic Drive (SID)} that focuses on one motivation at a time:
   It learns two separate policies which maximize the extrinsic and intrinsic rewards respectively. A high-level scheduler periodically selects to follow either policy to gather experiences. Disentangling the two policies allows the agent to faithfully conduct either pure exploration or pure extrinsic task fulfillment. Moreover, scheduling (even within an episode) inexplicitly increases the behavior policy space exponentially, which drastically differs from previous methods where the behavior policy could only change slowly due to the incremental nature of TD learning.

2. We introduce \textit{successor feature control (SFC)}, a novel intrinsic reward that is based on the concept of successor features. This state representation characterizes states through the features of all its successor states, instead of looking at local information only. This implicitly makes our method temporarily extended, which enables more structured and far-sighted exploration that is crucial in exploration-challenging environments.

We note that both the proposed intrinsic reward SFC and the hierarchical exploration framework SID are without any task-specific components, and can be incorporated into existing DRL methods with minimal computation overhead.

We present experimental results in three sets of environments, evaluating our proposed agent in the domains of visual navigation and control from pixels, as well as its capabilities of finding optimal solutions under distraction.

2. Related Work

Our work is connected to a range of DRL topics including intrinsic motivation, auxiliary tasks, successor representation and hierarchical RL. Below we discuss the most relevant.

2.1. Intrinsic Motivation and Auxiliary Tasks

Intrinsic motivation can be defined as agents conducting actions purely out of the satisfaction of its internal rewarding system rather than the extrinsic rewards (Oudeyer & Kaplan, 2008; Schmidhuber, 2010). There exist various forms of intrinsic motivation and they have achieved substantial improvement in guiding exploration for DRL, in tasks where extrinsic signals are sparse or missing altogether.

Curiosity, one of the most widely used kinds of intrinsic motivation, is quantified by Pathak et al. (2017) as the one-step prediction error of the features of the next state made by a forward dynamics model. Their ICM module has been shown to work well in visual domains including first-person view navigation. Since ICM is potentially susceptible to stochastic transitions (Burda et al., 2018a, Burda et al. (2018b) propose as a reward bonus the error of predicting the features of the current state output by a randomly initialized fixed embedding network. Another form of curiosity, learning progress or the change in the prediction error, has been connected to count-based exploration via a pseudo-count (Bellemare et al., 2016; Ostrovski et al., 2017) and has also been used as a reward bonus. Savinov et al. (2018) propose to train a reachability network, which gives out a reward based on whether the current state is reachable with a certain amount of steps from any state in the current episode. Similar to our proposed SFC, their intrinsic motivation is related to choosing states that could lead to novel trajectories. However we note that the reachability reward bonus captures the novelty of states with regard to the current episode, while our proposed SFC reward implicitly captures statistics over the full distribution of policies that have been followed, since the successor features are learned using states sampled from all past experiences.

Auxiliary tasks have been proposed for learning more representative and distinguishable features. Mirowski et al. (2016) add depth prediction and loop closure prediction as auxiliary tasks for learning the features. Jaderberg et al. (2016) learn separate policies for maximizing pixel changes (pixel control) and activating units of a specific hidden layer (feature control). However, their proposed UNREAL agent never follows those auxiliary policies as they are only used to learn more suitable features for the main extrinsic task.

2.2. Hierarchical RL

Various HRL approaches have been proposed (Kulkarni et al., 2016a; Bacon et al., 2017; Vezhnevets et al., 2017; Krishnan et al., 2017). In the context of intrinsic motivation, feature control (Jaderberg et al., 2016) has been adopted into a hierarchical setting (Dilokthanakul et al., 2017), in which options are constructed for altering given features. However, they report that a flat policy trained on the intrinsic bonus achieves similar performance to the hierarchical agent.

Our hierarchical design is perhaps inspired mostly by the work of Riedmiller et al. (2018). Unlike other HRL approaches that try to learn a set of options to construct the optimal policy, their proposed SAC agent aims to learn one flat policy that maximizes the extrinsic reward. While SAC schedules between following the extrinsic task and a set of pre-defined auxiliary tasks such as maximizing touch sensor readings or translation velocity, in this paper we in-
vestigate scheduling between the extrinsic task and intrinsic motivation that is general and not task-specific.

2.3. Successor Representation

The successor representation (SR) was first introduced to improve generalization in TD learning (Dayan, 1993). While previous works extended SR to the deep setting for better generalized navigation and control algorithms across similar environments and changing goals (Kulkarni et al., 2016b; Barreto et al., 2017; Zhang et al., 2017), we focus on its temporally extended property to accelerate exploration.

SR has also been investigated under the options framework (Sutton et al., 1999). Machado et al. (2017); Tomar* et al. (2019) evaluate successor features with random policies to discover bottlenecks or landmarks based on the clustering of such features. Options are then learned to navigate to those sub-goals. However, it remained unclear if the option framework would help in sparse exploration setups.

When using SR to measure the intrinsic motivation, the most relevant work to ours is that of Machado et al. (2018). They also design a task-independent intrinsic reward based on SR, however they rely on the concept of count-based exploration and propose a reward bonus, that vastly differs from ours. We will present our proposed method in the next section.

3. Methods

3.1. Preliminaries

We use the RL framework for learning and decision-making under uncertainty. It is formalized by Markov decision processes (MDPs) defined by the tuple \((\mathcal{S}, \mathcal{A}, p, \gamma)\). At time step \(t\) the agent selects an action \(a \in \mathcal{A}\) depending on its current state \(s \in \mathcal{S}\), receives a reward \(r \in \mathbb{R}\) and transits to the next state \(s' \in \mathcal{S}\). The state, action and reward at time \(t\) are random variables denoted by \(S_t, A_t, R_t\). Their probability distributions are defined by the probability kernel \(p(s', r | s, a) = \mathbb{P}(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)\) and the agent’s policy \(\pi(a | s) = \mathbb{P}(A_t = a | S_t = s)\).

The state-value is defined by \(v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]\) and the action-value \(q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]\). When representing the optimal action-value function by a deep network parameterized by \(\theta\) (Mnih et al., 2015), optimal policies can be found by minimizing the loss

\[
\mathcal{L}(\theta) = \frac{1}{2} \sum_{s,a} (y_t - q_\pi(s_t, a_t; \theta))^2,
\]

where \(y_t\) denotes the TD target. Since our algorithm requires an off-policy learning strategy, and in consideration for faster learning and less computation overhead, we use the same \(K\)-step target \((K = 5)\) as in the Ape-X DQN (Horgan et al., 2018) for bootstrapping without off-policy correction \((\theta^-\) denotes the target network)

\[
y_t = \sum_{k=1}^{K} \gamma^{k-1} R_{t+k} + \gamma^K \max_a q(s_{t+K}, a; \theta^-; \theta).
\]

Next, we will first introduce our proposed intrinsic reward successor feature control (SFC) (Sec.3.2.3.3.4), then present our proposed hierarchical framework for accelerating intrinsically-motivated exploration, which we denote as scheduled intrinsic drive (SID) (Sec.3.5.3.6).

3.2. Successor Representation and Successor Features

In order to encode long-term statistics into the intrinsic reward design for far-sighted exploration, we build on the formulation of successor representation (SR) (Dayan, 1993) which introduces a temporarily extended view of the states.

In SR, the occupancy of state \(s'\) denotes the total number of time steps the state process \((S_t)\) spends in \(s'\). Dayan (1993) introduced the idea of representing a state \(s\) by the expected occupancies of its future states in a process starting in \(s\) following a fixed policy \(\pi\). The SR is defined by

\[
\psi_\pi(s, s') := \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{\{S_t = s'\}} \bigg| S_0 = s \right],
\]

which is guaranteed to be finite for \(\gamma \in (0, 1)\).

The successor features (SF) is then introduced by extending the one-hot encoding in Eq. 3 to an arbitrary feature embedding \(\phi: \mathcal{S} \rightarrow \mathbb{R}^m\) (Kulkarni et al., 2016b; Barreto et al., 2017). It is defined by the \(m\)-dimensional vector

\[
\psi_{\pi,\phi}(s) := \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \phi(S_t) \big| S_0 = s \right].
\]

Analogously the SF represents the expected discounted feature activations, when starting in \(s\) and following a fixed policy \(\pi\). TD-learning can be used to compute the SR (Dayan, 1993), and when transfered to the function approximator case the SF can be learned via the following update rule

\[
\psi_{\pi,\phi}(S_t) \leftarrow \psi_{\pi,\phi}(S_t) + \alpha \left[ \phi(S_{t+1}) + \gamma \psi_{\pi}(S_{t+1}) - \psi_{\pi}(S_t) \right],
\]

where \(\alpha\) is the learning rate and \(S_{t+1} \sim \pi(. | S_t)\).

Previous works for learning the deep SFs have included an auxiliary task of reconstruction on the features \(\phi\) (Kulkarni et al., 2016a; Zhang et al., 2017), while in this work...
we investigate learning $\psi$ without this extra reconstruction stream. Instead of adapting the features $\phi$ while learning the successor features $\psi$, we fix the randomly initialized $\phi$. This design follows the intuition that since SFs ($\psi$) estimates the expectation of features ($\phi$) under the transition dynamics and the policy being followed, more stable learning of the SFs could be achieved if the features are kept fixed.

3.3. Successor Distance Metric

SR and SFs have several interesting properties which make them appealing as a basis for an intrinsic reward signal in the tabular and deep settings respectively:

- They can be learned even in the absence of any extrinsic reward signal and without learning a transition model. They combine advantages of model-based and model-free RL (Stachenfeld et al., 2014).
- They can be learned via computationally efficient TD.
- SR and SF estimates the expected future state visitation counts and feature activations respectively, both with statistics over complete state trajectories. Therefore they contain information even of spatially and temporally distant states which might help for effective far-sighted exploration.

Given the above discussion, we introduce a distance metric, the successor distance (SD), which measures the distance between states by the similarity of their SFs

$$d_{\pi,\phi}(s, s') := ||\psi_{\pi,\phi}(s) - \psi_{\pi,\phi}(s')||_2. \quad (6)$$

This definition bases on a well know approach in distance measures that defines distances by $d_W(x_1, x_2)^2 = (x_1 - x_2)^T W(x_1 - x_2)$. This can be seen by identifying the feature embedding with the $m \times |S|$ dimensional matrix $\Phi(i, j) := \phi(s_j)_i$ and SR with the $|S| \times |S|$ matrix $\Psi_{\pi}(i, j) := \psi_{\pi}(s_i, s_j)$. Then for $W = \Psi_{\pi}^T \Psi_{\pi}$ the distance measures $d_\pi$ and $d_W$ are equal. $W$ is symmetric and positive semi-definite (Eq.6) thus $d_W$ defines a pseudo-metric. This distance metric is illustrated in Fig.1 (Left).

3.4. Successor Feature Control

Using this metric to evaluate the intrinsic motivation, one choice would be to use the SD to a fixed anchor state as the intrinsic reward, which would depend heavily on the position of the anchor state. Even when a sensible choice for the anchor can be found, e.g. the initial state of an episode, the SD of distant states from the anchor assimilate. To circumvent this, we define the intrinsic reward successor feature control (SFC) as the squared SD of a one-step transition

$$R_{t+1}^{\text{SFC}} := d_{\pi,\phi}(S_t, S_{t+1})^2. \quad (7)$$

A high SFC reward indicates a big change in the future state occupancies when $\pi$ is followed. We argue this big change is a strong indicator for bottlenecks states, since in bottlenecks a minor change in the action selection can lead to a vastly different trajectory being taken. This is especially true for highly stochastic policies. Fig.1 (right) shows that those highly rewarding states under SFC and the true bottlenecks agree, which can be very valuable for exploration (Kulkarni et al., 2016b). Even when the environment does not contain distinguishable bottlenecks, SFC can still efficiently guide exploration since it takes into account distant future states.

Since our base algorithm is $K$-step Ape-X, we are only accessible to $K$-step experience tuples for learning the SFs. Thus we generalize SFC to $K$-step transitions

$$R_{t+K}^{\text{SFC}} = ||\psi_{\pi,\phi}(S_{t+K}) - \psi_{\pi,\phi}(S_t)||_2^2. \quad (8)$$

3.5. Scheduled Intrinsic Drive

Having proposed the intrinsic reward SFC, now we propose a hierarchical take on intrinsically motivated exploration.

When learning optimal value functions or optimal policies via TD or policy gradient with deep function approximators, optimizing with algorithms such as gradient descent means that the policy would only evolve incrementally: It is necessary that the TD-target values do not change drastically over a short period of time in order for the gradient updates to be meaningful. The common practice of utilizing a target network in off-policy DRL (Mnih et al., 2015) stabilizes the update but in the meanwhile making the policy adapt even more incrementally over each step.
But intrinsically motivated exploration, or exploration in general, might benefit from an opposite treatment of the policy update. This is because the intrinsic reward is non-stationary by nature, as well as the fact that the exploration policy should reflect the optimal strategy corresponding to the current stage of learning, and thus is also non-stationary.

With the commonly adopted way of using intrinsic reward as a bonus to the extrinsic reward and train a mixture policy on top, exploration would be a balancing art between the incrementally updated target values for stable learning and the dynamically adapted intrinsic signals for efficient exploration. Moreover, neither the extrinsic nor the intrinsic signal is followed for an extended amount of time.

Therefore, we propose to address this issue with a hierarchical approach that by design has slowly changing target values while still allowing drastic behavior changes. The idea is to learn not a single, but multiple policies, with each one optimizing on a different reward function. To be more specific, we assume to have $N$ tasks $\mathcal{T} \in \mathcal{T}$ (e.g. $N = 2$ and $\mathcal{T} = \{\mathcal{T}_E, \mathcal{T}_I\}$ where $\mathcal{T}_E$ denotes the extrinsic task and $\mathcal{T}_I$ the intrinsic task) defined by $N$ reward functions (e.g. $R_E$ and $R_I$) that share the state and action space. The optimal policy for each of these $N$ different MDPs can be learned with arbitrary off-policy DRL algorithms. During each episode, a high-level scheduler periodically selects a policy for the agent to follow to gather experience, and each policy is trained with all experience collected following those $N$ different policies. The overall learning objective is to maximize the main extrinsic task reward

$$\mathbb{E}_{(T|S_t)}[\mathbb{E}_{\pi_T(A_t|S_t)}[q_{T_E}(S_t,A_t|\pi_T(S_t))]], \quad (9)$$

where $\omega$ denotes the macro-policy of the scheduler.

By allowing the agent to follow one motivation at a time, it is possible to have a pool of $N$ different behavior policies without creating unstable targets for off-policy learning. By scheduling $M$ times even during an episode, we implicitly increase the behavior policy space by exponential to $N^M$ for a single episode (e.g. for $N = 2$ and $M = 8$ the behavior policy space could go up to 256). We investigated several types of high-level schedulers, however, none of them consistently outperforms a random one. We suspect the reason why a random scheduler already performs very well under the SID framework, is that a highly stochastic scheduling can be beneficial to make full use of the big behavior policy space. We present the different scheduler choices we tested in Supplementary F, and leave more sophisticated scheduler design to future work.

Moreover, disentangling the extrinsic and intrinsic policy strictly separates stationary and non-stationary behaviors, and the different sub-objectives would each be allocated with its own interaction time, such that extrinsic reward maximization and exploration do not distract each other.

We emphasize again that our proposed framework can be applied with any off-policy algorithms, and is directly applicable to settings with multiple extrinsic or intrinsic tasks.

3.6. Algorithm

Now we present in detail an instantiation of our proposed SID framework when using Ape-X DQN as a base off-policy DRL algorithm (Horgan et al., 2018), with SFC as the intrinsic reward. In detail, the algorithm is composed of:

- Q-Net $\{\theta_{\varphi}, \theta_{e}, \theta_{i}\}$: Contains an embedding $\theta_{\varphi}$ and two Q-value output heads $\theta_{e}$ (extrinsic) and $\theta_{i}$ (intrinsic).
- SF-Net $\{\theta_{\phi}, \theta_{v}\}$: Contains an embedding $\theta_{\phi}$ and a successor feature head $\theta_{v}$. $\theta_{\phi}$ is initialized randomly and kept fixed during training. The output of SF-Net is used to calculate the SFC intrinsic reward (Eq. 8).
- A high-level scheduler: Instantiated in each actor, selects which policy to follow (extrinsic or intrinsic) after a fixed number of environment steps (max episode length/$M$). The sheduler randomly picks one of the tasks with equal probability.
- 8 parallel actors: Each actor instantiates its own copy of the environment, periodically copy the latest model from the learner. We learn from $K$-step experiences, so each actor at each environment step stores $(s_{t-K}, a_{t-K}, \sum_{k=1}^{K} \gamma^{k-1} r_{t-K+k}, s_t, \text{terminal}_t)$ into a shared replay buffer. Each actor will act according to either the extrinsic or the intrinsic head based on the current task selected by its scheduler.
- 1 learner: Learns Q-Net and SF-Net parameters from samples from the same shared replay buffer, which contains all experiences collected from following different policies. The SFs are learned via Eq.5, and the value heads $\theta_{e}$ and $\theta_{i}$ are learned with the extrinsic reward $r^e$ and $r^i$ respectively (Eq.1).

4. Experiments

4.1. Experimental Setup

We evaluate our proposed intrinsic reward SFC and the hierarchical framework of intrinsic motivation SID in three sets of simulated environments: VizDoom (Kempka et al., 2016), DeepMind Lab (Beattie et al., 2016) and OpenAI Gym classic control (Brockman et al., 2016). The extrinsic rewards are designed to be very sparse in all environments such that agents with random exploration strategies are very unlikely to accomplish the task even once. This is done to reveal the exploration capabilities of each agent. Throughout all experiments, agents receive as input only raw pixels with no additional domain knowledge or task specific information.
We mainly compare the following agent configurations: **M**: train with only the extrinsic main task reward; **M+ICM**: add the ICM reward (Pathak et al., 2017) as a bonus to the extrinsic main task reward and train a mixture policy on this combined reward signal; **M+SFC**: same as M+ICM except for using our proposed SFC as the intrinsic reward bonus; **SID (M, ICM)**: schedule between following the extrinsic main task policy and the intrinsic policy trained with the ICM rewards; **SID (M, SFC)**: same as SID(M,ICM) except for using SFC to train the intrinsic policy. We note that except for the agents M and M+ICM, all other agents consist of our proposed algorithms (SFC or SID or both).

For the intrinsic reward normalization and the scaling for the extrinsic and intrinsic rewards we do a parameter sweep for each environment and choose the best setting for each agent. We note that the agents with the SID component are much less sensitive to different scalings. Since our proposed SID setup requires an off-policy algorithm in order to learn from experiences generated by following different policies, we implement all the agents above under the Ape-X DQN framework (Horgan et al., 2018) for a fair comparison. Additional experimental setup and model architecture details are reported in Supplementary B,C,D and E.

After a parameter sweep we set the number of scheduled tasks per episode to $M = 8$ for the SID agents in all experiments, meaning each episode is divided into up to 8 sub-episodes, and for each of which either the extrinsic or the intrinsic policy is sampled as the behavior policy.

### 4.2. VizDoom: FlytrapEscape

We conduct a first set of experiments in the VizDoom research platform (Kempka et al., 2016). The VizDoom game "DoomMyWayHome-v0" was previously used in several intrinsic motivation papers (Pathak et al., 2017; Savinov et al., 2018). It contains 8 rooms which are connected by corridors and each has a distinct unique texture, and the agent needs to navigate to a preset goal location based only on visual inputs in first-person view. We verified our implementation of the baseline algorithms through a set of experiments in this game and report the results in Supplementary A Fig.9. However, we found that the environment is not challenging enough since our base algorithm Ape-X, which already has descent random exploration capabilities, can solve the task reliably without intrinsic motivation.

We then designed a new game in VizDoom which is extremely exploration-challenging. Inspired by how flytraps catch insects, we design the layout of the rooms in a geometrically challenging way that escaping from one room to the next with random actions is extremely unlikely. The maze consists of 4 rooms separated by V-shaped walls pointing inwards the rooms. The small entry to each room locates at the junction of the V-shape, which is extremely difficult to maneuver into without a sequence of precise movements. The same also holds if the agent were trying to reach a next entry (Fig.2b) once stuck in the wing areas (Fig.2c). The layout of FlytrapEscape is shown in Fig.2a. The rooms are roughly 10 times the size of the rooms in "DoomMyWayHome-v0".

In each episode, the agent starts from the red dot shown in Fig.2a with a random orientation. An episode terminates if the final goal is reached in which case the agent will receive a reward of +1, or if a maximum episode steps of 10,000 is reached. We note the maximum episode steps for "DoomMyWayHome-v0" is 2100. The task is to escaping the forth room (marked as a green dot in Fig.2a).

The results of our experiments are shown in Fig.3 and demonstrate that our agent SID(M,SFC) efficiently explores the flytrap map and reliably learns how to navigate to the goal. The M agent without any intrinsic reward did not navigate to the goal even once. The agents with the ICM component perform poorly. Only 1 run of M+ICM learned to nav-
Figure 4. Top-down projection of the SFs of an agent after it learned how to navigate to the goal on FlytrapEscape (Fig.2a). For the purpose of visualization we discretized the map into \(512 \times 330\) grids and position the trained agent SID(M,SFC) at each grid, then computed the successor features \(\psi\) for that location for each of the 4 orientations \((0^\circ, 90^\circ, 180^\circ, 270^\circ)\), which resulted in a \(4 \times 512\) matrix. We then calculated the \(l_2\)-difference of this matrix with a \(4 \times 512\) vector containing the successor features of the starting position with the 4 different orientations. Shown in log-scale.

Figure 5. The AppleDistractions environment layout.

As an additional evaluation, in Fig.4 we visualize the SFs of the consecutive states the agent visits. As the agent has learned to navigate to the goal, the SFs of the intermediate states are highly and consistently influenced by the later states along the trajectory to the goal. Furthermore, we observe big intensity changes around the bottlenecks (the room entries) in the heatmap, which also supports the hypothesis that SFC leads the agent to bottleneck states. We believe this is the first time that SFs are shown to behave in a first-person view environment as one would expect from its definition. The evolution of the SFs over time is shown in the attached video [https://youtu.be/4ZHcBo7006Y](https://youtu.be/4ZHcBo7006Y).

### 4.3. DeepMind Lab: AppleDistractions

In the second experiment, we set out to evaluate the capabilities of an agent to reliably collect the faraway big reward in the presence of small nearby distractive rewards. For this experiment we use the 3D visual navigation simulator of DeepMind Lab (Beattie et al., 2016), which provides such a level "Stairway to Melon". However, we found it is easily solved without intrinsic rewards. Out of the same reasoning as for VizDoom, we constructed a much more challenging level "AppleDistractions" (Fig.5) with a maximum episode length of 900. In this level, the agent starts in the middle of the map and can follow either two corridors. Each corridor has multiple sections and each section consists of two dead-ends and an entry leading to next section. Each section has different randomly generated floor and wall textures. One of the corridors gives a small reward for collecting an apple in each of the section, while the other one contains a single big reward at the end of its last section.

As in the level "Stairway to Melon", the optimal policy of an agent in "AppleDistractions" would be to go for the single big reward at the end. But since the small apple rewards are much closer to the spawning location of the agent, the challenge here is to still explore other areas sufficiently often so that the optimal solution could be recovered.

All agents were evaluated on three different environment seeds, for each of which the texture of the map is different. We evaluated each agent with two random seeds for each of the three environment seeds, which makes a total of 6 runs.
Scheduled Intrinsic Drive

Figure 6. Average performance of all agents under comparison in AppleDistractions. Each agent is evaluated on the same 3 sets of random floor and wall textures, and for each set of textures 2 individual runs are conducted with different seeds. Each plot shows the average performance with ± standard derivation over the 6 runs for each agent. The SID agents perform better than their reward bonus agents, also the SFC agents received higher rewards than their ICM counterparts.

Neither the random nor the environment seed is tuned.

The results are visualized in Fig.6. Our SID(M, SFC) agent received on average the highest rewards. Furthermore, we see that scheduling helped both SFC and ICM to find the big reward and not settle for the small rewards. Their respective reward bonus counterparts were more prone to the small nearby rewards. This behavior is expected. With a separate policy for intrinsic motivation the agent can for some time interval completely “forget” about the extrinsic reward. The extrinsic policy can simultaneously learn from the new experiences and might learn about the large reward that it did not discover before. Surprisingly, only following the extrinsic reward resulted in a behavior that gained more reward than when the ICM bonus is added. On manual inspection we noticed that the agent with this mixture policy very often gets stuck without even collecting all the small apple rewards. This might be due to that the very different textures in each room makes predicting future states difficult, and hence the high intrinsic signals generated by ICM distract the agent from collecting the extrinsic rewards.

4.4. Classic Control From Pixels: PendulumPixels

To test if our proposed methods could be used in domains other than first-person visual navigation, we chose to conduct a third set of experiments in the classical control problem of swinging up an idealized pendulum. We based this experiment on the "Pendulum-v0" environment of OpenAI Gym (Brockman et al., 2016). The default setting provides dense reward signals with low-dimensional state inputs (angle and angle velocity). In order to learn this task in a sparse setup with pixel inputs, we did the following modifications: we added a custom render method to efficiently generate the front-view as inputs at each time step; we discretized the action space to 7 discrete actions with a max torque of 0.5 (with this low max torque, several back and forth wings are required for a full swing-up); a single reward signal of +1 will only be provided upon a successful swing-up; an episode will be terminated after a success or after a maximum episode length of 400 is reached (Supplementary H).

We show the performance of each agent in Fig.7. We can observe that, SFC is not domain specific to visual navigation, as both SFC agents work very well in this 2-D front-view environment which is quite different from the first-person view 3-D environments. Either used as a reward bonus (M+SFC) or as a separate policy (SID(M,SFC)), SFC outperforms the baseline M+ICM by a wide margin, although there are no clear bottlenecks in this environment.

5. Conclusion

In this paper, we investigate an alternative way of applying intrinsic motivation for exploration in DRL. We propose a hierarchical agent SID that schedules between following extrinsic and intrinsic drives. Moreover, we propose a new type of intrinsic reward SFC that is general and evaluates the intrinsic motivation based on longer time horizons. We conduct experiments in three sets of environments (VizDoom, DeepMind Lab and OpenAI Gym) and show that both our contributions: the hierarchical framework SID of using intrinsic drives, and the intrinsic reward SFC help greatly in improving exploration efficiency.

We consider many possible future research directions that could stem from this work, including designing more efficient scheduling strategies, incorporating several intrinsic drives (that are possibly orthogonal and complementary) instead of only one into the hierarchical framework, test our framework in other control domains such as manipulation, and extend our evaluation onto real robotics systems.
Scheduled Intrinsic Drive

References

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In AAAI, pp. 1726–1734, 2017.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. Successor features for transfer in reinforcement learning. In Advances in neural information processing systems, pp. 4055–4065, 2017.

Beattie, C., Leibo, J. Z., Tepleyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., et al. Deepmind lab. arXiv preprint arXiv:1612.03801, 2016.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In Advances in Neural Information Processing Systems, pp. 1471–1479, 2016.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. arXiv preprint arXiv:1808.04355, 2018a.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. arXiv preprint arXiv:1810.12894, 2018b.

Dayan, P. Improving generalization for temporal difference learning: The successor representation. Neural Computation, 5(4):613–624, 1993.

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

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. Distributed prioritized experience replay. arXiv preprint arXiv:1803.00933, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397, 2016.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In Computational Intelligence and Games (CIG), 2016 IEEE Conference on, pp. 1–8. IEEE, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

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, pp. 3675–3683, 2016a.

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

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesaruo, G., and Campbell, M. Eigenoption discovery through the deep successor representation. arXiv preprint arXiv:1710.11089, 2017.

Machado, M. C., Bellemare, M. G., and Bowling, M. Count-based exploration with the successor representation. arXiv preprint arXiv:1807.11622, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529, 2015.

Oudeyer, P.-Y. and Kaplan, F. How can we define intrinsic motivation? In Proceedings of the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Lund University Cognitive Studies, Lund: LUCS, Brighton. Lund University Cognitive Studies, Lund: LUCS, Brighton, 2008.

Paszke, A., Gross, S., Chintala, S., Chan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.
Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, 2017.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degraeve, J., Van de Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., and Gelly, S. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*, 2018.

Schmidhuber, J. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Stachenfeld, K. L., Botvinick, M., and Gershman, S. J. Design principles of the hippocampal cognitive map. In *Advances in neural information processing systems*, pp. 2528–2536, 2014.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Tomar*, M., Ramesh*, R., and Ravindran, B. Successor options : An option discovery algorithm for reinforcement learning, 2019. URL https://openreview.net/forum?id=Byxr73R5FQ.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017.

Zhang, J., Springenberg, J. T., Boedecker, J., and Burgard, W. Deep reinforcement learning with successor features for navigation across similar environments. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 2371–2378. IEEE, 2017.
A. Supplementary: Comparison to Random Network Distillation

To further verify our method, we compared it with random network distillation (RND) (Burda et al., 2018b). We tested on the doom map "DoomMyWayHome-v0" and on FlytrapEscape. The results can be seen in Fig.8 and Fig.9. While RND reaches a reasonable performance on the relatively easy map "DoomMyWayHome-v0", it fails to find any reward on FlytrapEscape.

![Figure 8. Extrinsic rewards per episode for different agents in FlytrapEscape. Each plot shows the average performance with ±1 standard derivation over 3 random non-tuned seeds.](image)

![Figure 9. Extrinsic rewards per episode for different agents on "DoomMyWayHome-v0". Each plot shows the average performance with ±1 standard derivation over 3 random non-tuned seeds.](image)

From the FlytrapEscape experiment we present in Sec.??, one might suspect that the big performance margin of SFC over ICM could be due to the fixed random features of the SF-Net. But from the results of the comparison with RND shown in Fig.8, we can verify that the improvement from our method is not due to the fixed random features, but due to the temporarily extended nature of the SFs.

B. Supplementary: Implementation Details

For computational efficiency, we implement our own version of the prioritized experience replay. We split the replay buffer into two, with size of 40,000 and 10,000. Every transition is pushed to the first one, while in the second one only transitions are pushed on which a very large TD-error is computed. We store a running estimate of the mean and the standard deviation of the TD-errors and if for a transition the error is larger than the mean plus two times the standard deviation, the transition is pushed. In the learner a batch of size 128 consists of 96 transitions drawn from the normal replay buffer and 32 are drawn from the one that stores transition with high TD-error, which as a result have relatively seen a higher chance of being picked.

To adapt the $\epsilon$ settings from the 360 actors in the Ape-X DQN (Horgan et al., 2018) to our setting of 8 actors, we set a fixed $\epsilon_i$ for each actor $i \in \{1, \ldots, 8\}$ as

$$\epsilon_i = \epsilon \left( \frac{a - 1}{360 - 1} \right) \alpha,$$

where $\alpha = 7$ and $\epsilon = 0.4$ are set as in the original work.

C. Supplementary: Reward Normalization

Most network parameters are shared for estimating the expected discounted return of the intrinsic and extrinsic rewards. The scale of the rewards has a big influence on the scale of the gradients for the network parameters. Hence, it is important that the rewards are roughly on the same scale, otherwise effectivley different learning rates are applied. The loss of the network comes from the regression on the Q-values, which approximate the expected return. So our normalization method aims to bring the discounted return of both tasks into the same range. To do so we first normalize the intrinsic rewards by dividing them by a running estimate of their standard deviation. We also keep a running estimate of the mean of this normalized reward and denote it $r'_I$. Since every time step an intrinsic reward is received we estimate the discounted return via the geometric series. We scale the extrinsic task reward that is always in $\{0, 1\}$ with $\eta r'_I$, where $\gamma_I$ is the discount rate for the intrinsic reward. Furthermore, $\eta$ is a hyperparameter which takes into account that for Q-values from states more distant to the goal the reward is discounted with the discount rate for the extrinsic reward depending on how far away that state is. In our experiments we set $\eta = 3$ (see Appendix E for how we searched for this hyperparameter).

D. Supplementary: Model Architecture

We use the same model architecture as depicted in Fig. 10 across all 3 sets of experiments:

ReLU activation is added after every layer except for the
Scheduled Intrinsic Drive

Figure 10. Model architecture for the SID (M, SFC) agent. Components with color yellow are randomly initialized and not trained during learning.

last layers of each dashed blocks in the above figure. For the experiments with the ICM (Pathak et al., 2017), we added BatchNorm (Ioffe & Szegedy, 2015) before activation for the embedding of the ICM module following the original code released by the authors. Code is implemented in pytorch (Paszke et al., 2017).

E. Supplementary: Training Details

We use a batch size of 128 for all experiments. For optimizer, we use Adam (Kingma & Ba, 2014) with a learning rate of $1e^{-4}$.

We did the same search for hyperparameters and normalization technique for all four algorithms that include an intrinsic reward and found the setting described in Section C to work the best for all of them. The algorithms were evaluated on the flytrap. For $\eta$ we tried the values in $\{0.3, 1, 3, 10\}$. We also tried to not normalize the rewards and just scale the intrinsic reward. To scale the intrinsic reward we tried the values $\{0.001, 0.01, 0.1, 1\}$. However, we found that as the scale of the intrinsic rewards is not the same over the whole training process this approach does not work well. We also tried to normalize the intrinsic rewards by dividing it by a running estimate of its standard deviation and then scale this quantity with a value in $\{0.01, 0.1, 1\}$.

F. Supplementary: Scheduler Designs

We investigated three types of high-level schedulers:

- Random scheduler: Sample a task from uniform distribution every task steps.
- Switching scheduler: Sequentially switches between extrinsic and intrinsic task.
- Macro-Q Scheduler: Learn a scheduler that learns with macro actions and from sub-sampled experience tuples.

- Threshold-Q Scheduler Scheduler: Selects task according to the Q-value output of the extrinsic task head.

Macro-Q Scheduler: In each actor, we keep an additional local buffer that stores $N + 1$ subsampled experiences: $\{s_{t-N+1}, s_{t-N+2}, \ldots, s_{t-N+1}, s_{t}\}$. Then at each environment step, besides the $K$-step experience tuple mentioned above, we also store an additional macro-transition $\{s_{t-N+1}, s_{t}\}$ along with its sum of discounted rewards to the shared replay buffer. This macro-transition is paired with the current task as its macro-action. The Macro-Q Scheduler is then learned with an additional output head attached to $\theta_\phi$ (we also tried $\theta_\psi$).

Threshold-Q Scheduler: For this scheduler we do not need additional learning. It just selects a task based on the current Q-value of the extrinsic head $\theta_e$. We tried the following selection strategies:

- Running mean: select intrinsic when the current Q-value of the extrinsic head is below its running mean, extrinsic otherwise
- Heuristic median: observing that the running mean of the Q-values might not be a good statistics for selecting tasks due to the very unevenly distributed Q-values across the map, we choose a fixed value that is around the median of the Q-values ($0.007$), and choose intrinsic when below, extrinsic otherwise

As we report in the paper, none of the above scheduler choices consistently performs better across all environments than a random scheduler. We leave this part to future work.

G. Supplementary: AppleDistractions

Figure 11. The AppleDistractions environment. 11a,11b,11c show exemplary first-person views captured in AppleDistractions.

H. Supplementary: PendulumPixels

The pendulum dynamics are given through the ordinary differential equation

$$\ddot{\theta} = -3 \sin(\theta + \pi) + 3a$$
with initial conditions \( \theta_0, \dot{\theta}_0 = 0 \), where \( \theta \) is the angle of the rod. The action \( a \) corresponds to a torque that acts on the pendulum and is chosen from the discretized action space \( \mathcal{A} = \{-0.5, -0.25, -0.125, 0, 0.125, 0.25, 0.5\} \). \( \theta_0 \) is uniformly sampled from the interval \( [(1 - 3/180)\pi, (1 + 3/180)\pi] \), where the angle \( \theta = \pi \) corresponds to the downwards position. The ODE is solved by semi-implicit euler integration with fixed time interval \( \Delta t = 0.05 \).

In contrast to the OpenAI Gym we use images as observations. A front-view image of the pendulum, as seen in Fig. 12. Where some standard preprocessing steps are taken. The color scheme is changed to greyscale and the rod is given the brightest color (1) and the background is black (0). Taken actions are not visualized explicitly and can be just inferred by the dynamics.

I. Supplementary: Further Analysis of Pendulum experiments

The pendulum experiment also illustrates, why a single mixture policy of intrinsic and extrinsic reward can be hurtful. Interestingly, the SID component helped the SID(M+ICM) agent outperform the baseline M+ICM a lot, while for the 2 SFC agents no big differences can be observed. An explanation for the behaviors of the SFC agents could be that in this environment there are no clear bottleneck states as those in the FlytrapEscape environment. Regarding the ICM we hypothesize that as it calculates rewards depending on how well the model can predict the next state, the policy does not stabilize and starts choosing different actions in a given state as soon as the model has learned for old transitions, which might happen very quickly as the environment is visually very simple. Thus when using ICM as a reward bonus this problem can be critical, as the mixture policy is not given too much time to converge to the optimal solution once the ICM has learned the transition in a given state since the corresponding intrinsic reward would be very low. This is no serious issue when scheduling between two separate policies, as the extrinsic one is given its own time slots to concentrate on fulfilling the task while the intrinsic