Now you see me: finding the right observation space to learn diverse behaviours by reinforcement in games

Raphaël Boige, Nicolas Audebert, Clément Rambour, Guillaume Levieux

Now you see me: finding the right observation space to learn diverse behaviours by reinforcement in games

Raphaël Boige[1], Nicolas Audebert*[1], Clément Rambour[1], and Guillaume Levieux[1]

[1]CEDRIC (EA4629) – Conservatoire national des arts et métiers, HESAM Université, France

Abstract

Training virtual agents to play a game using reinforcement learning (RL) has gained a lot of traction in recent years. Indeed, RL has delivered agents with super-human performances on multiple gameplays. Yet, from a human-machine interaction standpoint, raw performance is not the only dimension of a "good" game AI. Exhibiting diversified behaviours is key to generate novelty, one of the core components of player engagement. In the RL framework, teaching agents to discover multiple strategies to achieve the same task is often framed as *skill discovery*. However, we observe that the current RL literature defines diversity as the exploration of different states, *i.e.* the incentive of the agent to "see" new observations. In this work, we argue that this definition does not make sense from a gameplay point of view. Instead, diversity should be defined as a distance on observations from an *observer*, external to the agent. We illustrate how DIAYN/SMERL, state of the art RL algorithms for skill discovery, fail to discover meaningful behaviours in a simple tag game. We propose an easy fix by introducing the notion of diversity spaces, defined as the observations gathered by a third-party external to the agent.

Keywords: reinforcement learning, video games, diversity.

1 Introduction and related work

1.1 Diversity for games

Novelty and difficulty are often considered as major components of the inherent appeal of video games [Mal81; KW04]. *Novelty* is the pleasure of discovering and trying to understand a new universe, an uncharted territory that is yet to explore, with its own rules and its own fantasy. *Difficulty* comes from the challenge posed by the game [Lev11], that requires motor as well as adaptation and problem-solving skills from the player who needs to develop one or several winning strategies. Designing AI that are able to use a diverse set of strategies while achieving human-like performances in the game tackles these components of fun at the same time: they both provide new strategies for the player to discover, while simultaneously helping to maintain the level of cognitive challenge. However, few works have investigated game AI from this point of view.

These last years, the community has focused on achieving super-human performance in various games, such as Go [Sil+16], Atari games [Bad+20], DotA 2 [Ber+19] or StarCraft 2 [Vin+19]. Reinforcement learning strives to find optimal solutions that reach human-like performances [Mni+15], yet these "perfect" agents are probably not the most useful to serve as actual opponents. For example, Lee Se-dol, the world Go champion, retired from all competitions because "AI can't be defeated" [BBC19]. In addition, optimal opponents that know a single strategy tend to feel predictable, which is neither exciting nor rewarding for the player. Diversity using AI in games have mostly put great effort on Procedural Content Generation (PCG) [Kha+20; Liu+20] to automate content generation of assets such as levels, music, textures and environments, leaving behind the idea of creating a single agent capable of multiple behaviours.

1.2 Diversity in reinforcement learning

One of the main challenges in RL is the trade-off between *exploration* and *exploitation*. Exploration is often offloaded to naive approaches, such as ϵ-greedy policies [SB18] or addition of gaussian noise [Lil+15]. This leads to *curiosity*, an elusive concept in RL, loosely defined as the incentive for exploration. It is generally implemented as intrinsic motivation, a reward inter-

*Corresponding author: `nicolas.audebert@cnam.fr` nal to the agent maximized when learning the policy. At its core, curiosity is a way to encourage the agent to reach diversified states by crafting ad hoc rewards, *e.g.* by using count-based (or pseudo-count based) rewards [KS02; BT02] that encourage the agent to visit states "far" from those previously visited.

Intrinsic motivation is frequently applied to *skill discovery.* This approach stems from goal-conditioned algorithms, that learn a policy $\pi(a|s, g)$ conditioned over a goal g. Often, the goal is a state that we want the agent to attain, and the agent is rewarded by the inverse distance between the goal and its current state, *e.g.* in Reinforcement Learning with Imaginary Goals [Nai+18], or by maximizing mutual information between a goal and the trajectory of the agent in Variational Intrinsic Control (VIC) [GRW17]. Maximization of mutual information between a latent variable and the state is a popular skill discovery method. In [Cam+20], it is shown that two views of the mutual information lead to different algorithms such as VIC, Diversity is All You Need (DIAYN) [Eys+18] or Dynamics Aware Unsupervised Discovery of Skill [Sha+20].

2 RL and skill discovery

We aim to train an agent by reinforcement to discover diverse meaningful strategies in a game environment. We leverage existing algorithms on skill discovery and define a strategy (or a "behaviour") as the application of a skill in a given setting. We describe below DIAYN – and its extension SMERL –, the skill discovery algorithm that will serve as the basis for our experiments.

2.1 DIAYN

Diversity is All You Need (DIAYN) [Eys+18] is an unsupervised skill discovery algorithm for RL. If Z is the random variable that represents a latent skill with distribution $p(z)$, DIAYN learns a policy conditioned to the skill, *i.e.* $\pi(a|s, z)$ where $a \in \mathcal{A}$ is an action and $s \in \mathcal{S}$ is a state. The goal is to discover both meaningful and diverse skills, based on three assumptions:

- skills should encode information about state and thus maximize mutual information $\mathcal{I}(Z; S)$;
- skills are independent from actions in a given state and thus minimize $\mathcal{I}(Z; A|S)$;
- skills should cover the state space, *i.e.* they should be different and with high entropy.

Overall, this results in maximizing the objective:

$$\begin{aligned} \mathcal{F} &= \mathcal{I}(Z; S) - \mathcal{I}(Z; S|A) + \mathcal{H}(A|S) \\ &= \mathcal{H}(A|S, Z) - \mathcal{H}(Z|S) + \mathcal{H}(Z) \end{aligned} \quad (1)$$

As the posterior $p(z|s)$ is intractable, DIAYN approximates it with a learned discriminator $q_\phi(z|s)$. In other words, DIAYN trains a neural network parametrized by its weights ϕ to predict the skill that conditions the policy based on the current state.

The DIAYN algorithm builds on top of Soft Actor Critic (SAC) [Haa+18]. SAC's objective already includes the maximization of the entropy of the policy, *i.e.* it already takes care of maximizing $\mathcal{H}(A|S, Z)$. DIAYN therefore only needs to define the intrinsic reward that maximizes mutual information between states and skills. It is defined as the predictability of a skill z by the neural network q_ϕ, knowing the state s:

$$r_t^{\text{diversity}} = \log(q_\phi(z|s_{t+1})) - \log(p(z)) \quad (2)$$

By maximizing this reward, the algorithm encourages the states reached by different skills to be discriminable. Diverse behaviours should emerge since different skills explore different regions of the state space $\mathcal{S}$.

2.2 SMERL

[Kum+20] introduced Structured Maximum Entropy Reinforcement Learning (SMERL). It extends DIAYN to the supervised setting, *i.e.* where a task-oriented reward is available. Consider a task defined by its true reward r^{true}. Let J_{π_θ} denote the cumulative reward of the policy π_θ. The idea of SMERL is to seek diversity when the cumulative reward is close to the optimal reward J_{π^*}, *i.e.* to look for slightly sub-optimal but diverse solutions. Intuitively, SMERL starts by learning the optimal SAC policy and then starts diversifying around the optimal solution , by introducing the following pseudo-reward:

$$r_t = r_t^{\text{true}} + \beta \mathbb{1}_{\{J_{\pi_\theta} \geq J_{\pi^*} - \epsilon\}} r_t^{\text{diversity}} \quad (3)$$

In practice, the algorithm consists in sampling a skill $z \sim p(z)$, generating a trajectory according to π_θ^z and storing the transitions $(r_t, \mathbf{s}_t, \mathbf{s}_t p, \mathbf{a}_t, z)$ in a buffer. At the end of the episode, we estimate $J_{\pi_\theta} = \sum_t r_t^{\text{true}}$ and compute the diversity pseudo-rewards from batches sampled from the buffer, update the actor and the critic towards this reward and finally update the discriminator to classify the skills based on states. Note that the diversity reward is only added if $\sum_t r_t^{\text{true}} \geq J_{\pi^*} - \epsilon$, where the "optimal" cumulative reward J_{π^*} is estimated as the mean reward of the policy learnt by SAC alone. ϵ is an hyperparameter controlling how much the diversified solutions from SMERL can deviate from the SAC baseline.

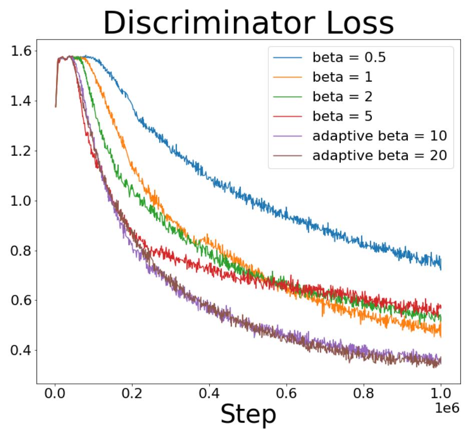
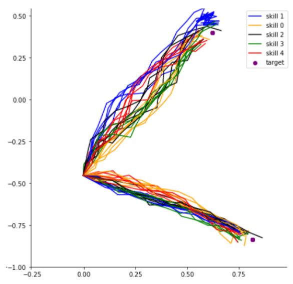
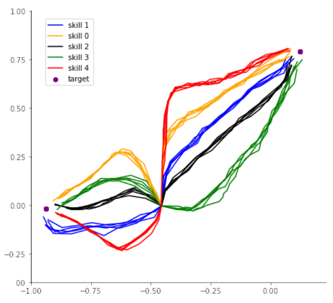


Figure 1: Learning 5 SMERL skills on Nav2D (Static) shows that adaptive β has better learning curve while being less sensitive to hyperparameter variations.



(a) Trajectories of a Nav2D agent trained with SMERL on absolute coordinates $\{(x,y)_{\text{agent}};(x,y)_{\text{target}}\}$.



(b) Trajectories of a Nav2D agent trained with SMERL on relative coordinates $\{x_{\text{agent}} - x_{\text{target}}, y_{\text{agent}} - y_{\text{target}}\}$.

Figure 2: Nav2D: absolute position vs. relative position

Adaptive diversity trade-off As shown in Eq. (3), β is a hyperparameter from SMERL controlling the weight of the diversity reward. When SAC has converged, a high β is desirable to start the diversification of skills. But afterwards, a high β is likely to destabilize the reward. This complicates the practical tuning of β. To alleviate this difficulty, we propose an adaptive beta as a function of the per-skill *diversity* reward. We denote the expected diversity reward of the z-conditioned policy as $\tilde{J}_{\pi^z_\theta}$. The desired behaviour is that β is high when $\tilde{J}_{\pi^z_\theta}$ is low and conversely. We automatically balance β using $\beta^z = \frac{\beta_0}{1+\tilde{J}_{\pi^z_\theta}}$ where β_0 is our replacement hyperparameter. Note that β is defined per-skill z since $\tilde{J}_{\pi^z_\theta}$ depends on z. If $\tilde{J}_{\pi^z_\theta} = 0$, then $\beta^z = \beta_0$ which bootstraps skill-discovery. Conversely, when the diversity reward $\tilde{J}_{\pi^z_\theta}$ increases, β decreases. This heuristic implicitly constrains the expected diversity: while β is high, the diversity reward masks the true reward. The true reward gains more weight when the skills are already diversified. Empirically, we observed significantly better convergence and robustness to the initial value of β_0 using our heuristic, as illustrated in Fig. 1.

2.3 State space or diversity space?

DIAYN and SMERL implicitly define diversity as a distance function over state space. This has two drawbacks. First, since the number of observations in the state is large, the task of the discriminator $q_\phi : s \to z$ in DIAYN and SMERL becomes easy. More observations means more directions alongside which to partition the state space. As a workaround, Eysenbach et al. suggest feeding the discriminator some observations, but not all, *i.e.* a "diversity" subset $\mathcal{D} \subset \mathcal{S}$ of the state.

However, this is not enough since observations are ego-centered, *i.e.* relevant to the internal state of the agent. This is the second drawback.To illustrate this point, let us consider a simple 2D navigation environment, shown in Fig. 2 (Nav2D). The agent (in blue) can freely move in the (x, y) plane at a fixed speed. It receives a small negative reward for each time step and a large positive reward when it reaches the target (in purple). The target is randomly placed somewhere in the environment at the beginning of an episode, and then stays immobile during the episode. We use SMERL to train the agent to learn five different skills. We consider two cases. In Fig. 2a, the agent has an observation space $\mathcal{S} = \{x_{\text{agent}}, y_{\text{agent}}, x_{\text{target}}, y_{\text{target}}\}$. The coordinates are expressed in the absolute referential of the 2D plane. Since the target does not move, x_{target} or y_{target} are constant and cannot be used to discriminate skills. Therefore the discriminator can only map the position $x_{\text{agent}}, y_{\text{agent}}$ to the skills. However the target position is reset between episodes. SMERL therefore learns a meaningless partitioning of the 2D space: the diversity reward can only be increased through skills that result in very slight variations in trajectory.

But consider Fig. 2b, where the observations are the agent's *relative* position w.r.t. the target, *i.e.* $x_{\text{agent}} - x_{\text{target}}$ and $y_{\text{agent}} - y_{\text{target}}$. The diversification of skills now partition the *relative* position of the agent. The five skills now describe more meaningful behaviours: changing the skill changes the angle of approach of the argent w.r.t. the target. This naive example shows that the origin of the observations must be taken into account to induce distinguishability relevant to third-party *observers*.[1]

In addition, we argue that finding diverse high-level strategies require high-level variables. Raw observations such as position in the 2D plane is mostly irrelevant from a gameplay standpoint. If we want human-relevant skills, we should give the discriminator human-relevant observations. We argue that the best discrimination space $\mathcal{D}$ to learn diverse behaviours is generally not the same as the state space $\mathcal{S}$. We want to make the job of the discriminator hard, otherwise it will use meaningless features to fit a low-margin decision function. We believe that this is better achieved by using a low-dimensional discrimination space, such as a collection of discrete or even binary variables. We also want these variables to be relevant to an observer and encode high-level information.

3 Learning diverse strategies

Let us devise an experiment to illustrate how the choice of the "diversity" space impacts the agent's behaviours.

3.1 Experimental setup

We design a "tag" game where the agent must touch the moving target (red cube). The blue agent (blue cube) can move freely around the environment defined by a square outer wall and one or more inner walls.

Reward The agent receives a reward of +100 when it reaches the target and −0.1 for each timestep.

Observation space The agent knows its absolute 2D position (x, y) in the world and the target's position (x', y'). It has a Lidar-like vision of its surroundings, *i.e.* the agent sends 12 rays that are one hot-encoded into a vector $\mathbf{v}$ where element $\mathbf{v}_i$ is set to 1 if the ray intersects an obstacle (*e.g.* a wall) and 0 if the ray does not intersect anything.

Diversity space The target also has a Lidar-like cone of vision. Contrary to the agent, this vision covers only a specific solid angle and not its complete surroundings. This vision cone controls the "hiddenness" of the agent: if any ray intersects with the agent, then the "observed" variable h is set 1, otherwise it is set to 0. The *hidden/seen* variable is clearly relevant from a perceptual point of view, because for a player, seeing or not the opponent is a crucial gameplay element. As such, it defines an interesting strategy decomposition that be extended to more complex settings.

Environment We consider five variants of the "tag" environment, illustrated in Fig. 3:

- `static`: the target does not move and stays at an arbitrary position and angle;
- `alternate`: the target does not move but looks to the left or to the right randomly at the beginning;
- `scan`: the target rotates on itsef to "scan" the area, therefore changing the cone of vision at a slow pace;
- `patrol`: the target patrols in a straight line without moving its cone of vision;
- `shortcut`: same as `static` but the wall now has an "L"-shape, making one path significantly shorter.

`static` is the simplest declination. Since the wall layout is fixed and symetrical, there are two equivalent optimal solutions to reach the target: take the right path (the one where the target looks at), or take the left path (the one the target does not watch). In `shortcut`, the longer path is the unguarded one. But since it is longer, it is also sub-optimal. As we will see, even in these cases, SMERL learns skills that do not significantly differ in a meaningful way.

Discrimination spaces Our experiment consists in comparing the skills learnt by SMERL depending on the state space that is used to compute the diversity reward, *i.e.* the intput space of the discriminator.

Our first baseline is the vanilla SMERL that takes all the state as input for the discriminator. The second baseline is SMERL but with relevant selected discrimination variables, including the *hidden/seen* variable. The considered discrimination spaces are:

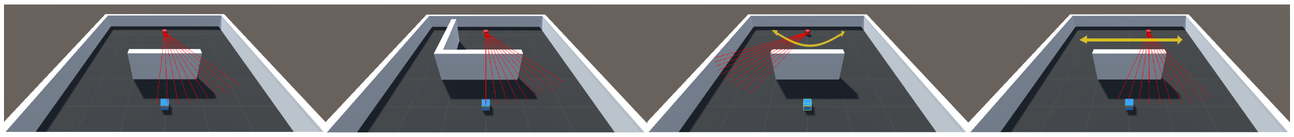
- Full state: all 16 state variables are passed to the discriminator;
- Selected state: only 5 observation variables ($\{x_{\text{agent}}, y_{\text{agent}}, x_{\text{target}}, y_{\text{target}}, \text{dist}\}$) are considered and the *hide/seen* variable from the target;
- Observer state: only the *hide/seen* variable is passed to the discriminator.

Training The agent is trained using SMERL on top of SAC. The discriminator is a fully connected neural network fed observations depending on the setting (full state, selected state or observer state). It is trained with a batch size 512 and a replay buffer of 1×10^6 steps. We use our adaptive beta with $\beta_0 = 20$.

3.2 Experimental results

We want to show that enforcing discrimination on the binary variable *hidden/seen* leads to the discovery of the relevant corresponding *hide/show* skills.

[1]Amusingly, [Eys+18] had identified the notion of *perceived diversity* from observers but only for actions, and not for states: "*we want to use states, not actions, to distinguish skills, because actions that do not affect the environment are not visible to an outside observer*".



(a) Static/Alternate (b) Shortcut (c) Scan (d) Patrol

Figure 3: Variants of our “tag” game environment.

	Static			Alternate			Shortcut			Scan			Patrol		
	Show	Hide	Mean	Show	Hide	Mean	Show	Hide	Mean	Show	Hide	Mean	Show	Hide	Mean
Full state	3.18	26.72	14.94	14.88	18.95	16.91	5.72	39.20	22.46	16.89	21.01	18.95	15.26	17.84	16.55
Selected state	3.39	26.94	15.16	17.85	14.88	16.37	3.87	39.47	21.67	20.78	16.68	18.73	15.54	17.94	16.74
Observer state	**2.84**	**5.65**	**4.25**	**4.14**	**4.34**	**4.24**	**3.71**	**6.58**	**5.14**	**3.32**	**4.54**	**3.93**	**3.31**	**5.82**	**4.56**

Table 1: DTW error between trajectories of hide/show explicit policies and skill-learned policies generated automatically. The results are averaged over 10 random seeds.

Reference behaviours As a reference, we train two agents using SAC that explicitly capture *hide/show* skill by hand-crafting the reward. The *hide* agent receives an additionnal reward of -2 when it is not hidden (*i.e.* it interesects the target's vision cone). The *show* agent receives the opposite reward. We then compare our skills implicitly learnt using SMERL with those two explicit SAC reference policies.

Metrics We aim to measure the distance between the enforced *hide/show* policies and the skill-learned policies. To do so, a convenient way is to sample trajectories and compute distance over the sampled trajectories. The chosen metric to compare the explicit *hide/show* policies and the skills-learned policies is the Dynamic Time Warping (DTW) measure. DTW allows to easily compute the similarity of two time series, even if they have not the same length. This distance will be computed between the $(x, y)_t$ trajectories of each skill of our agents, against the reference agents trained with SAC with an explicit *hide* or *show* reward.The lower the DTW between a trajectory and the reference *hide* skill trajectory, the more similar the policies are.

Results Our results from Table 1 show that our method learns the two skills (*i.e.* two conditioned policies) that are the closest to the explicit *hide/show* policies for all five variants of the “tag” environment, as illustrated in Fig. 4. This demonstrates the ability of the *perceptual discrimination* method to generate strategies based on high-level observer-based variables. Conversely, the other SMERL methods are able to learn the *show* skill on the `static` and `shortcut` environment for instance, since this skill and the optimal trajectory (without diversity, *i.e.* shortest path to the red target) are nearly identical. For the other environments, the skills are uncorrelated with hiddenness since obtaining the diversity reward is easy: changing the trajectory a little bit in (x, y) is enough to separate states.

4 Discussion and perspectives

Skill discovery algorithms from the RL literature try to learn diverse strategies by encouraging the agent to visit “different” states. However, state of the art approaches such as DIAYN and SMERL assume that the internal state of the agent, *i.e.* ego-centered observations, are enough to learn meaningful behaviours. In this work, we show that this is far from true. The state space $\mathcal{S}$ is generally of high dimensionality and filled with low-level attributes that do not meaningfully describe the behaviour of an agent from a human observer point of view. Instead, we believe that high-level observer-based variables are key to enable meaningful strategy discovery: they relate to high-level perception of human players while not being too easy for the discriminator that defines the implicit distances of states. Our experiments on the tag game shows that a single well-crafted variable is enough to make meaningful behaviours emerge.

There are two directions to go further. First, DIAYN and its successors discriminate on a single state $\mathbf{s}_t$. This makes the discrimination imperfect in cases where skills are bound to overlap (*e.g.* at the beginning of

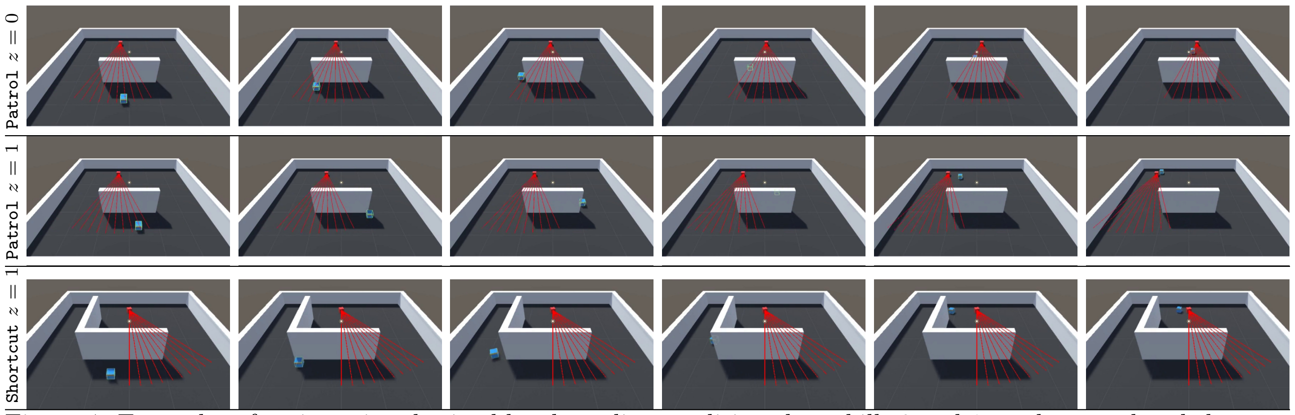


Figure 4: Examples of trajectories obtained by the policy conditioned on skills 0 and 1 on the patrol and shortcut environments using our approach (discriminator trained on the observer state).

a level). Instead, one could investigate discriminating *trajectories*, *i.e.* defining a skill as the latent variable conditioning a sequence of states. Discrimination would then be done on the series $(s_0, \dots, \mathbf{s}_t)$, *e.g.* with recurrent neural network by defining a new diversity reward $r_t^z = \log(q_\phi(z|\mathbf{s}_{t+1}, h_t)) - \log(p(z))$, with h_t the hidden activation of the RNN. Another direction is the perceptual route. If what matters is the diversity perceived by a player, we could feed to the discriminator images of the agent as captured by a virtual camera inside the environment. The discriminator could then be replaced by a CNN, taking these "photos" as an input. Only variations in behaviours that result in visible changes would increase the diversity. However, this should be done carefully, since the high dimensionality of images could incent "near-adversarial" changes that are visible for the CNN, but not meaningful for humans.

References

[Bad+20] A. P. Badia et al. "Agent57: Outperforming the Atari human benchmark". In: ICML. 2020.

[BBC19] BBCNews. *Go master quits because AI 'cannot be defeated'*. 2019.

[Ber+19] C. Berner et al. "Dota 2 with Large Scale Deep RL". In: *CoRR* abs/1912.06680 (2019).

[BT02] R. Brafman et al. "R-max-A General Polynomial Time Algorithm for Near-Optimal RL". In: *Journal of Machine Learning Research* 3 (2002).

[Cam+20] V. Campos et al. "Explore, discover and learn: Unsupervised discovery of state-covering skills". In: *ICML*. 2020.

[Eys+18] B. Eysenbach et al. "DIAYN: Learning Skills without a Reward Function". In: *ICLR*. Sept. 2018.

[GRW17] Karol Gregor et al. "Variational Intrinsic Control". In: *ICLR Workshop Track*. 2017.

[Haa+18] T. Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *ICML*. 2018.

[Kha+20] A. Khalifa et al. "PCGRL: Procedural Content Generation via Reinforcement Learning". In: *AAAI Conference on AIIDE* (2020).

[KS02] M. Kearns et al. "Near-optimal RL in polynomial time". In: *Machine Learning* (2002).

[Kum+20] S. Kumar et al. "One Solution is Not All You Need: Few-Shot Extrapolation via SMERL". In: *Neural Information Processing Systems* (2020).

[KW04] R. Koster et al. *A Theory of Fun for Game Design*. Paraglyph Press, 2004.

[Lev11] G. Levieux. "Mesure de la difficulté des jeux vidéo". PhD thesis. 2011.

[Lil+15] T. P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: ICLR. 2015.

[Liu+20] J. Liu et al. "Deep Learning for Procedural Content Generation". In: *Neural Computing and Applications* (2020).

[Mal81] T. Malone. "What Makes Things Fun to Learn? A Study of Intrinsically Motivating Computer Games". In: *Pipeline* 6 (1981).

[Mni+15] V. Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* (2015).

[Nai+18] A. Nair et al. "Visual RL with imagined goals". In: *Neural Information Processing Systems* (2018).

[SB18] R. Sutton et al. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[Sha+20] A. Sharma et al. "Dynamics-Aware Unsupervised Discovery of Skills". In: ICLR. 2020.

[Sil+16] D. Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* (2016).

[Vin+19] O. Vinyals et al. "Grandmaster level in StarCraft II using multi-agent RL". In: *Nature* (2019).