InfODist: Online distillation with Informative rewards improves generalization in Curriculum Learning

Rahul Siripurapu [†] [*] **Vihang Patil** [§] **Kajetan Schweighofer** [§] **Marius-Constantin Dinu** [§,‡]
Thomas Schmied [§] **Luis Ferro** [†] **Markus Holzleitner** [§] **Hamid Eghbal-Zadeh** [§]
Michael Kopp [†] **Sepp Hochreiter** [§,†]
[†] Institute of Advanced Research in Artificial Intelligence (IARAI)
[§] ELLIS Unit Linz and LIT AI Lab,
Institute for Machine Learning,
Johannes Kepler University Linz, Austria
[‡] Dynatrace Research

Abstract

Curriculum learning (CL) is an essential part of human learning, just as reinforcement learning (RL) is. However, CL agents that are trained using RL with neural networks produce limited generalization to later tasks in the curriculum. We show that online distillation using learned informative rewards tackles this problem. Here, we consider a reward to be informative if it is positive when the agent makes progress towards the goal and negative otherwise. Thus, an informative reward allows an agent to learn immediately to avoid states which are irrelevant to the task. And, the value and policy networks do not utilize their limited capacity to fit targets for these irrelevant states. Consequently, this improves generalization to later tasks. Our contributions: First, we propose InfODist, an online distillation method that makes use of informative rewards to significantly improve generalization in CL. Second, we show that training with informative rewards ameliorates the capacity loss phenomenon that was previously attributed to non-stationarities during the training process. Third, we show that learning from task-irrelevant states explains the capacity loss and subsequent impaired generalization. In conclusion, our work is a crucial step toward scaling curriculum learning to complex real world tasks.

1 Introduction

Deep Reinforcement Learning (DRL) has recently shown good results on various real-world problems (Akkaya et al., 2019; OpenAI, 2018; Silver et al., 2017; Tan et al., 2018; Becker-Ehmck et al., 2020). Many of these methods rely heavily on curricula (e.g., via self-play or automatic domain randomization). Considering the importance of curricula to human learning (Skinner, 1958; Peterson, 2004; Dapena, 2002), it is natural to expect further advances from improved curriculum learning. Manually constructing a curriculum for each task is difficult, and automated curriculum learning methods have thus seen a resurgence in current research (Schmidhuber, 1991; Bengio et al., 2009; Schmidhuber, 2012; Portelas et al., 2020). However, these methods have failed to reach their potential due to a host of reasons, including restricted task search spaces, inadequacies of exploration, and failure of models to continually learn and generalize to more complex tasks in an open-ended fashion (Wang et al., 2019; Jabri et al., 2019; Florensa et al., 2018).

Recent work suggests that the use of DRL in CL would produce limited generalization to downstream tasks, not only due to non-stationarities inherent in DRL methods (Igl et al., 2021) but also due to the

[*]Correspondence to: rahul.siripurapu@iarai.ac.at

distribution shifts induced by the curricula. Igl et al. (2021) identify policy updates, the consequent value updates and bootstrapping as three major sources of non-stationarity inherent in DRL. Kumar et al. (2020) show that bootstrapping, in particular, leads to a capacity loss in the neural networks used to approximate the policy or value. Lyle et al. (2022) ascribe this issue particularly to the non-stationarity in the target values as the policy improves during the learning process. These works use an estimate of the rank of the feature matrix in the penultimate layer of the value network to study this capacity loss phenomenon. They attribute the impaired learning to the reduced discriminative power of the value networks while training. A lower feature rank would imply that the network cannot distinguish between as many states as it has the capacity for, i.e., the network suffers from state aliasing. However, we contend that this is not just because of the changing output values, but also due to the diversity of inputs seen during exploration. This may sound counterintuitive, since exploration is fundamental to learning. We argue that while exploration helps us learn what to avoid, it is often unnecessary to model the failure modes given limited network capacity. In many cases, it is simply sufficient to know an approach is wrong without knowing why. Furthermore, it could be detrimental if exploration induces the agent to alias states that are important for later tasks. Thus, distillation methods (similar to (Igl et al., 2021; Siripurapu et al., 2020)) can be used to relearn the task without spending too much capacity on known failure modes.

Exploration in RL is exponentially hard with respect to the task horizon. Lee et al. (2021) learn a proximity function from expert demonstrations in goal-reaching tasks and use the differences of the function to produce a reward. They show that such a function easily generalizes to unseen states and is particularly useful for imitation learning in data-limited regimes. Building on this work, we find that such learned informative rewards – corresponding to a notion of progress towards the goal – act as an inductive bias for RL agents in learning representations that generalize well to more complex downstream tasks in CL. Ideally, a maximally informative reward helps to illuminate the optimal action for a state immediately, reducing the task horizon to 1. This reduces the task complexity to be quadratic in the horizon, and therefore reduces the number of task-irrelevant states the agent visits while learning. Thus, suggests a simple way to reduce irrelevant information learned by the agent, similar to regularization methods (Hochreiter and Schmidhuber, 1994). We therefore attempt to learn such an informative reward function from demonstrations. We then use this learned reward to train an agent with better state representations. Fig. 1 gives a brief overview of our setting.

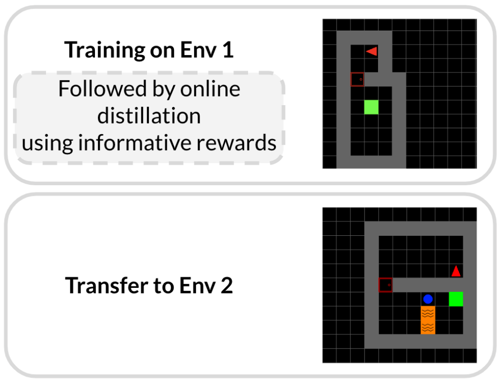
To summarize our **contributions**:

- We propose InfODist, a new online distillation method using informative rewards that improves curriculum learning.
- We find that using informative rewards improves utilization of capacity, as measured by the rank of the value network's features in the penultimate layer.
- We find that avoiding learning from task-irrelevant states explains the improved capacity utilization.

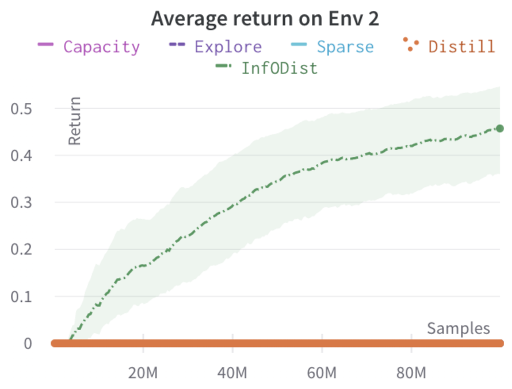
2 Background

In the following section, we describe related work, explain how our method differs from prior work and provide a short summary of the work we build upon.

Using a proximity functions as an informative reward. Lee et al. (2021) improve generalization to unseen states in Imitation Learning (IL) by learning a proximity function. Essentially, they show that learning a temporal distance measure from expert demonstrations, helps the learned reward function to extract more information from the limited demonstrations, enabling it to generalize better to *unseen states*. Furthermore, they compare the IL performance in settings where the demonstrations are limited in diversity and show that their method outperforms other IL methods, including those that learn from observations and also demonstrations in some cases. In our work, we use their proximity function as an informative reward for online policy distillation in CL, to minimize task-irrelevant exploration, and consequently non-stationarity during training. We show that distillation using such informative rewards produces policies that generalize to *unseen environments* (particularly ones containing unseen objects and action consequences) as opposed to rewards that generalize to unseen states. Also, we use more complex procedurally generated grid worlds and adapt their method to work in our setting.



(a)



(b)

Figure 1: **Curriculum stages with InfODist (a) and Effect of using informative rewards (b)**. The stages of the curriculum are depicted in (a). Plain curriculum learning involves training on Env 1 and directly transferring to Env 2. To improve generalization, we use an Inverse RL step (Learns an informative reward) and an online distillation step (Trains an agent from scratch using the learned reward) on Env 1 to help improve learned feature representations. On the right, we show that doing so improves results significantly in comparison with existing baselines that tackle non-stationarity (Distill) and capacity issues (Capacity) in DRL. We find, that only InfODist learns to solve certain hard exploration environments, while all other methods fail to learn at all. (The shaded region depicts the min-max envelope of 3 seeds)

Role of non-stationarity in generalization. Generalization impairment due to non-stationarity has already been well studied in both Supervised Learning (Ash and Adams, 2019) and Reinforcement Learning (Igl et al., 2021; Fedus et al., 2020; Lyle et al., 2022; Steinparz et al., 2022). Igl et al. (2021) propose iterated network distillation as a solution to tackle any non-stationarity that arises in the context of DRL. However, using iterated distillation can erase information about failure modes that are learned by the policy during the early stages of training. This is because later distillation stages rely on replay-buffer data that may no longer contain the initial learning phases. While it is valid to lose the detailed modelling of the failure modes, we still need to know what to avoid. Furthermore, naive distillation could force the network to overfit to the training task (by making strong assumptions given the limited data) and consequently impair generalization to more complex downstream tasks. Our generalization experiments suggest this is indeed the case. Hence, we propose to use an online distillation by learning a task-irrelevant-exploration-minimizing reward function that provides both positive and negative reinforcement, i.e., the agent learns both what to do and what to avoid.

Measuring network capacity using feature rank. Kumar et al. (2020) showed that it is possible to observe the adverse effects of bootstrapping by looking at the rank of the features in the penultimate layer of the value network. Lyle et al. (2022) use similar methods to show that this is due to distribution shifts (concept shifts in particular). First, using the ability to fit random targets, they show in the supervised setting that networks iteratively trained on randomly sampled conditional target label distributions (concept shifts) exhibit increasing loss with further iterations (negative forward transfer). However, when the network is sufficiently large (over-parametrized regime), they show that this effect is reversed, and positive forward transfer is observed. Furthermore, Lyle et al. (2022) shows that we are usually in the under-parametrized regime in DRL, and the networks exhibit the expected *negative* forward transfer to random targets. Second, they use a feature rank measure to show a positive correlation between training performance and feature rank and find that methods that *use a dense reward improve feature rank* and performance.

While our results corroborate these findings, we further find that it is not merely the reward's density but rather the reward's informativeness that significantly improves the feature rank. Further, we show that informative rewards reduce the number of irrelevant states visited by the agent, leading to better capacity utilization and feature rank. We also go on to demonstrate its impact in the curriculum learning setting, where we obtain significantly better generalization compared with methods that merely alleviate distribution shifts using distillation. Furthermore, we also use a slightly different version of feature rank that not only considers non-zero singular values but rather the uniformity of the singular values by using an entropy formula (Roy and Vetterli, 2007). Lyle et al. (2022) adapt a method from continual learning, which they call InFeR. InFeR adds multiple linear heads to the penultimate layer and trains these heads to predict the same values that the augmented network does at initialization. This tends to behave as a regularization that prevents the rank from collapsing lower than the number of heads used. While this method does work as intended, we also show that preventing rank collapse is insufficient for improving generalization to downstream tasks.

Minimizing exploration for bootstrapping RL. Jump-Start RL (JSRL) (Uchendu et al., 2022) is a recent method that proposes to accelerate RL by producing a starting state distribution very close to the goal state and then moving away from the goal during training. They mainly address to offline RL, followed by the online fine-tuning setting (i.e., the task does not change or increase in complexity). They show that using a guiding policy during online fine-tuning in the limited data regime performs significantly better than other IL and RL baselines. However, they do not directly address the transfer problem in curriculum learning. We show that JSRL indeed transfers well because of its ability to reduce task-irrelevant exploration, but it does not perform as well as InfODist.

Informative reward functions with return equivalence. RUDDER (Arjona-Medina et al., 2019; Patil et al., 2022; Dinu et al., 2022; Holzleitner et al., 2020; Widrich et al., 2021) learns to assign credit such that the expected sum of future rewards is zero under the current policy. RUDDER uses a lessons buffer to store trajectories from the current policy or an expert to learn a reward redistribution that satisfies *return equivalence*. Return equivalence ensures that the reward redistribution does not modify the task specified by the original MDP. In contrast, we relax this constraint but rely on the generalization ability of neural networks to capture the task definition from the expert trajectories. While we lose guarantees that the learned reward accurately represents the task defined by the original reward function, we argue that for most real-world tasks, it is incredibly hard to define a reward function that accurately represents the desired behavior. Hence, we justify relaxing this constraint, to allows us to redefine the reward and make learning easier (minimize states visited). Furthermore, in our setting, we consider only goal-reaching tasks in the curriculum and do not require return equivalence. Our experiments use the proximity function as our informative reward function.

To summarize, we conclude that the problem we address has not been dealt with in prior work, and our results provide a novel insight into improving generalization in CL.

3 Methodology

In this section, we first formalize the problem setting, then we discuss how we learn the informative reward, and finally, we describe the baselines used to substantiate our claims.

3.1 Preliminaries

Here, we formalize our problem setting. We define the MDP M as the six tuple $(S, A, R, P, \rho_0, \gamma)$, where S and A are the state and action spaces respectively. $R : S \times A \times S \to \mathbb{R}$ is a scalar reward function. $P(s_{t+1} \mid s_t, a_t)$ represents the transition probability distribution. ρ_0 is the initial state distribution and $\gamma \in [0, 1)$ is a discount factor commonly used to ensure that the discounted sum of rewards converges given an infinite horizon. For the experiments, however, we use a fixed horizon H. A policy $\pi_\theta(a_t \mid s_t)$ is the distribution of actions taken by an agent in each state s_t of the MDP, which is commonly approximated using a deep neural network parametrized by θ. The learned informative reward function $R^{\text{inf}}_\phi(s_t, a_t, s_{t+1})$, is parametrized by ϕ. It suffices to say that an informative reward is one that tells the agent what to do at every state immediately, i.e. moving temporally closer to the goal produces a positive reward without delay or otherwise a negative reward. In other words, we expect that the reward captures the notion of *progress* towards the goal (see Fig. 2). Such a reward ensures that visiting task-irrelevant states is immediately penalized.

We assume we have a minimal curriculum of two task MDPs M_1 and M_2. Both of them are assumed to be goal reaching tasks and M_2 is a more complicated version of M_1, in that:

a) It contains objects that are not seen, and or necessitates actions that are not used in M_1.

b) Only agents which solve M_1 are even capable of solving M_2 in a reasonable time frame.

Our assumption ensures that M_2 as defined necessitates a CL method (barring the use of manually engineered rewards). To test for generalization, we train the agent first on M_1, followed by transfer to M_2 (it should also work for M_3, M_4...). Ideally, we want the environment to be complex enough for it to possess diverse task-irrelevant information and to also induce the state aliasing effect. For simpler environments, we find all baselines have very similar performances, making it difficult to analyze the phenomenon that we are interested in.

3.2 Learning informative rewards

Next, we describe how we can learn the informative reward that we need to minimize exploration. Lee et al. (2021) define the proximity function $f_\phi(s_t)$, as an exponential of the temporal distance $(T-t)$ to the goal $f_\phi(s_t) \sim \delta^{(T-t)}, \delta \in (0,1)$, i.e. 1 if close to the goal and 0 if far away. They take differences of such a function as a reward $R^{\text{inf}}_\phi(s_t, a_t, s_{t+1}) = f_\phi(s_{t+1}) - f_\phi(s_t)$. Hence, their reward corresponds to the notion of *progress* towards the goal and helps us to minimize exploration. The proximity function corresponds to a value function when the goal has a reward 1, using a discount $\gamma = \delta$. Hence, this is similar to RUDDER's notion of informativeness, which also proposes taking a difference of the value function.

Additionally, Lee et al. (2021) use an ensemble of K proximity functions $f^k_\phi(s_t)$ and add an uncertainty penalty $U_\phi(s_{t+1}) := \sqrt{\text{Var}[f^k_\phi(s_{t+1})]}$ to prevent the agent from visiting states where the proximity function variance is high.

$$R^{\text{inf}}_\phi(s_t, a_t, s_{t+1}) = f_\phi(s_{t+1}) - f_\phi(s_t) - \lambda U_\phi(s_{t+1})$$

Furthermore, they continue training the proximity function to estimate 0 for the student's states, since the limited expert trajectories are not sufficient to learn such a function fully on states not visited by the expert. Hence, they propose the following loss:

$$\mathcal{L}_\phi = \mathbb{E}_{\tau^e_i \sim \mathcal{D}^e, s_t \sim \tau^e_i}\left[f_\phi\left(s_t\right) - \delta^{(T_i - t)}\right]^2 + \mathbb{E}_{\tau \sim \pi_\theta, s_t \sim \tau}\left[f_\phi\left(s_t\right)\right]^2,$$

where $\mathcal{D}^e$ is the expert dataset and τ^e_i is a trajectory sampled from it. Unlike IL, we have access to the goal termination conditions for both tasks, to make use of during the online distillation. Hence, we modify this loss function to include successful samples from the student to expedite learning on the procedurally generated environments that we use.

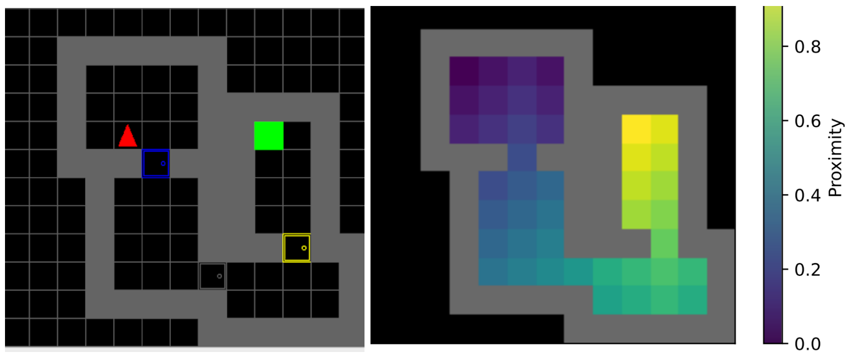
$$\mathcal{L}_\phi = \mathbb{E}_{\tau^e_i \sim \{\mathcal{D}^e, \mathcal{D}^{\pi_\theta}_s\}, s_t \sim \tau^e_i}\left[f_\phi\left(s_t\right) - \delta^{(T_i - t)}\right]^2 + \mathbb{E}_{\tau \sim \pi_\theta, s_t \sim \tau}\left[f_\phi\left(s_t\right)\right]^2,$$

where $\mathcal{D}^{\pi_\theta}_s$ is the set of successful trajectories sampled by the student π_θ. To conclude, we use the proximity function as our informative reward function, since it captures this notion of *progress* towards the goal (see example in Figure 2). Any path that leads to failure modes or irrelevant states would produce an immediate negative reward and consequently allow our agent to avoid visiting such states.

3.3 Baselines and Implementation

In this section we describe the baselines, how we adapt them to CL and finally why we chose them. To evaluate the advantages of using an informative reward for improving transfer in CL, we consider three baselines apart from *Sparse*, which is vanilla CL (because it uses a sparse reward to train on M_1). It involves simply training an RL agent on M_1 and transferring it to M_2.

First, we have Iterated Relearning (ITER) (Igl et al., 2021), which we refer to as **Distill** to better distinguish methods. ITER is explicitly designed to handle non-stationarity in DRL via iterated distillation. Their distillation process combines both offline loss terms with behavioral cloning (BC) terms, where the BC terms are annealed to zero over the distillation process. We can see this as offline distillation in contrast to our method. The student is then hot-swapped with the teacher during training on M_1 and thus at the end of training, we have an agent that has experienced less non-stationarity overall. We then transfer this agent to the more complex tasks corresponding to M_2 to evaluate how well it transfers. We specifically chose this baseline because it addresses non-stationarity directly rather than exploration. This should help us rule out whether it is purely a non-stationarity problem.



(a) (b)

Figure 2: **What is an informative reward?** While dense rewards may have spurious optima, an informative reward is positive when the agent makes progress towards the goal and negative otherwise. Taking the differences of the scaled Manhattan distance from each tile to the next tile (right) produces an informative reward for the agent.

Second, we have Initial Feature Regularization (InFeR) (Lyle et al., 2022), which we refer to as **Capacity**. In InFeR, the authors adapt a method based on Benjamin et al. (2018) designed to tackle catastrophic forgetting. They apply an l_2 penalty on outputs from a set of auxiliary network heads such that the outputs do not change from their values at the start of training. This protects the network from the rank collapse phenomenon, (Kumar et al., 2020) (see Fig. 5) as the penultimate features are forced to have projections along each independently initialized linear head layer. We performed a hyperparameter search to identify the combination of the number of heads, an amplification term β and a loss weight α that achieves comparable performance on M_1. Then we transfer this agent to M_2 where we use plain PPO to evaluate the learned features. We chose this baseline because it addresses network capacity directly, without dealing with either non-stationarity or exploration. This helps us rule out whether it is purely a capacity problem.

Third, we consider Jump-Start Reinforcement Learning (JSRL) (Uchendu et al., 2022), which we refer to as **Explore**. We describe JSRL in more detail in Section 2. We modify their approach slightly. They consider two settings, one where the roll-in is randomly sampled from $[0, H)$ and another where a roll-in schedule is fixed a priori to decrease during the training. We start with a fixed roll-in set to a constant $R = 10$, and then reduce the roll-in for the next episode by 1 if the agent solves the previous episode or increase by 1 if it fails. Also, we restrict the roll-in value to lie in range $(0, R]$. Using this, we have a natural curriculum for the roll-in, which is decided by the performance of the agent directly. We thus train an agent in such a fashion on M_1 and then transfer the learned weights to a plain PPO agent on M_2. We chose this baseline because it indirectly attempts to minimize task-irrelevant exploration, without dealing with capacity. But it does add additional non-stationarity to the task in the form of a moving initial state distribution. This should also help us understand to what extent the problem is linked to exploration.

Finally, for all baselines and experiments, we train using an actor-critic method, PPO (Schulman et al., 2017), which uses function approximators for both the value and the policy.

4 Experiments and results

In this section, we describe the three major experiments we perform and discuss the results obtained.

First, we construct challenging curricula using the `MiniGrid` environment (Chevalier-Boisvert et al., 2022). In our case, for the first task M_1, we consider the `MiniGrid-MultiRoom-N2-v0` environment (2-*room*), which consists of two procedurally generated rooms in a 13×13 grid with the agent in the first room and a green goal square in the second room. To reach the goal, the agent needs to learn to navigate to the door, open it, and then move to the goal square within a horizon of 40 steps. For M_2, we consider a set of 6 different, more difficult versions of the 2-room environment, as shown in Fig. 3. To make the tasks moderately challenging, for each of the environments, we set the horizon

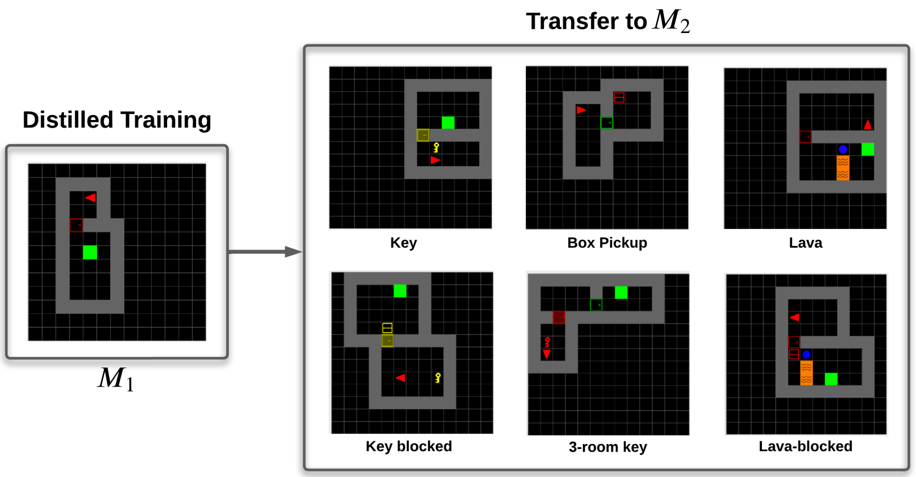


Figure 3: **Multiroom curriculum.** We first train the agent using various methods on 2-room (M_1) and evaluate the generalization capability of the learned policy by transferring them to 6 different, more complex settings (M_2) that involve picking up/dropping keys, boxes, balls and avoiding lava. While merely adding a room can be seen as an extrapolation of the original 2-room setting, all the tasks involve objects *(lava, key, box, ball)* or actions *(pick, avoid, drop)* that the agent has never seen or taken before and hence measure generalization capability. The environments are procedurally generated every episode, making this a challenging test.

based on the number of rooms, with 20 steps per room. Hence, all the 2-room environments have a horizon for 40 and the 3-room environments have a horizon of 60.

The first experiment involves standard curriculum learning. We use a particular method to train on M_1 and then evaluate generalization to M_2 by initializing the agent with the weights trained on M_1. We always train three seeds and transfer the best performing seed. Thus, our first experiment (see Fig. 4) tests how well each method above generalizes to the downstream task. In Fig. 4, we show that online distillation with an informative reward produces the best downstream generalization in comparison with all other methods. We observe that methods which indirectly reduce task-irrelevant exploration like JSRL (Explore) come next. Methods which tackle non-stationarity like ITER (Distill) come next. This suggests that reducing exploration directly is more useful in our setting. Finally, the vanilla CL baseline, sparse performs somewhat better than the InFeR (Capacity) baseline which regularizes a capacity measure (feature rank) showing that directly regularizing feature rank can have a detrimental effect on transfer as it would be easy for a network to memorize irrelevant state-information.

Second, we count the number of times that the agents visit a randomly sampled square (In the first room) in the 2-room environment while training using the respective methods (see Fig. 5(a)). As expected, methods which aim to minimize task-irrelevant exploration reduce the diversity of states visited by the agent during the course of learning. Other methods do not have a noticeable effect on exploration.

Third, we measure the feature rank using an entropy based measure Roy and Vetterli (2007) of the features in the penultimate layer using states sampled from a vanilla RL expert trained on 2-room. Here, we find that the feature rank regularization methods indeed help to avoid the rank collapse phenomenon (see Fig. 5(b)). Furthermore, we find that an informative reward (InfODist) prevents rank collapse to an extent and also leads to much higher rank after training. This is likely because it reduces state aliasing, as the reward helps to focus learning only on task-relevant states. Furthermore, JSRL (Explore), which also indirectly minimizes task-irrelevant exploration, has a very low feature rank because it initially sees states from a localised distribution near the goal. However, despite having additional non-stationarity in the form of a moving initial state distribution ρ_0, it still achieves a high feature rank at the end of training, and we consequently see that methods which achieve higher feature rank naturally tend to perform better on downstream tasks than ones that don't.

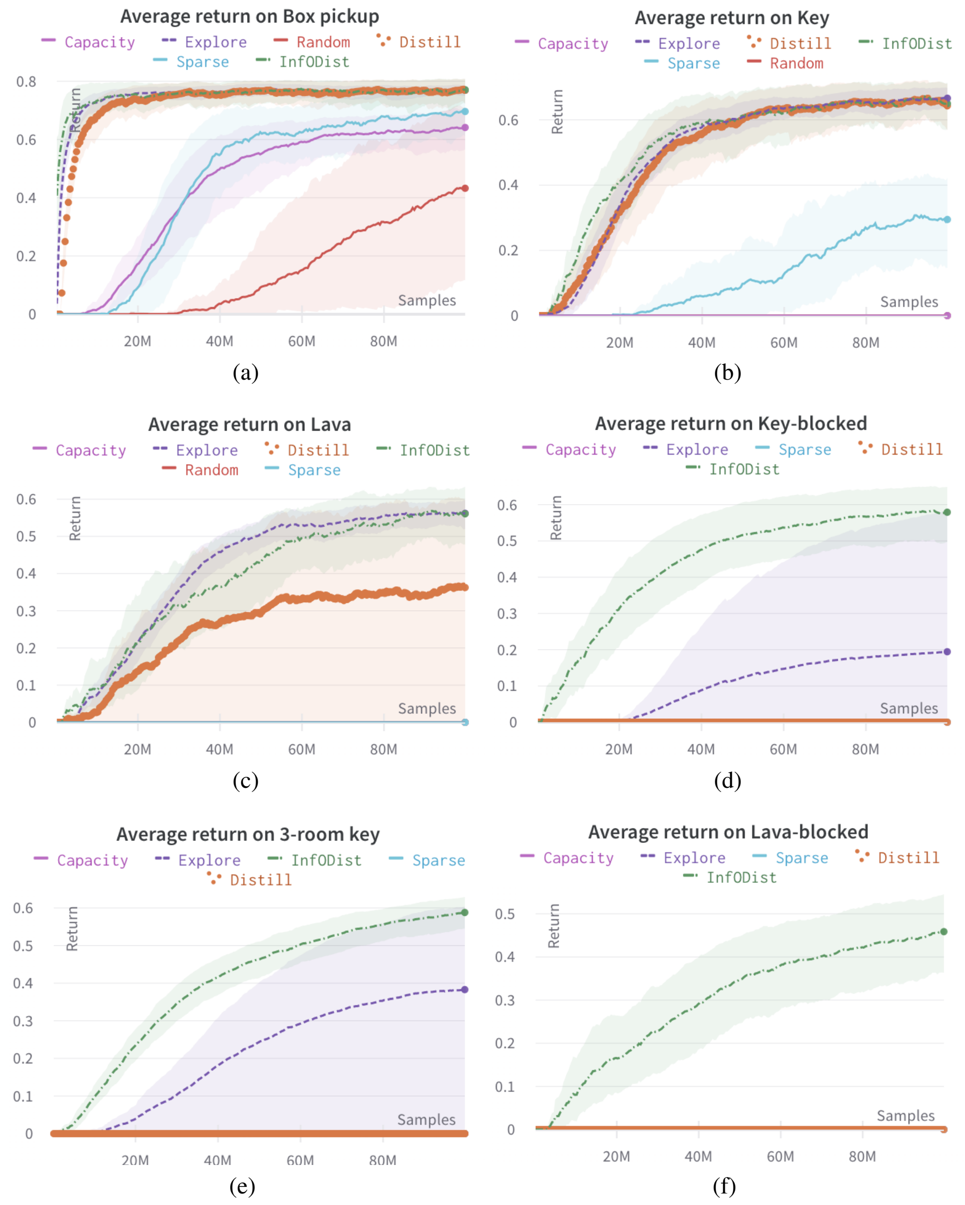
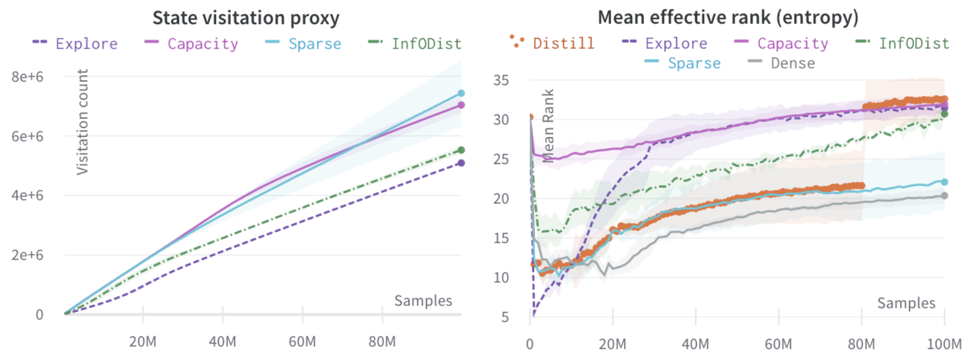


Figure 4: **Generalization to downstream tasks** Generalization results after training an agent on the plain 2-room environment M_1 using 5 different methods and transferring that agent to 6 complex downstream tasks with differing difficulties representing M_2 (see Fig. 3). We find that the InfODist, where we use a proximity function as an informative reward, generalizes well to even the most complex environments involving unseen objects such as lava, keys, balls and boxes. Explore, which also minimizes task-irrelevant exploration by rolling in with a guide policy, performs slightly worse. Distill, which uses offline distillation (Combining BC and Offline RL terms) is next, showing that minimizing non-stationarity also works. The vanilla CL baseline, sparse performs better than Capacity, which explicitly regularizes network capacity (feature rank). This shows that regularizing feature rank is not as powerful as minimizing task-irrelevant exploration or non-stationarity. This makes intuitive sense, since a network may find it easy to buff up its rank by memorizing irrelevant state information. Finally random is a vanilla PPO baseline which fails to learn all but the first task showing that our choices of M_2, indeed, require training on M_1 and are challenging choices to evaluate generalization in CL. (Note: the shaded area corresponds to min-max envelope of 3 seeds per baseline)



(a) (b)

Figure 5: **State visitation counts (a) and Mean feature rank (b)** In (a), we show the number of times the agent visits a randomly sampled square in the first room during a particular episode. We see that using an informative reward significantly reduces the number of times the agent visits the chosen square, which is unlikely to lie on the shortest path to the goal. Furthermore, we see that while Explore minimizes this measure, it only succeeds in the initial stages due to the roll-in taking it towards the goal, which leads to episode termination lowering its chances of visiting our chosen square. Other methods do not reduce task-irrelevant exploration. In (b), we show a measure of the feature rank in the penultimate layer of the value network for data collected from a vanilla RL expert in 2-room. We see that using an informative reward reduces the initial drop in the feature rank due to zero rewards in the sparse setting. Furthermore, we observe that the feature rank is improved not because the reward is dense (Dense, a Gaussian noise reward is worse than the sparse reward) but because the reward is informative of the task and reduces exploration. We also note the correlation between the higher variance in state visitation and the feature rank, especially for the sparse reward, and the trend holds within the seeds as well. The feature rank drops very low immediately for Explore because the starting state distribution is very limited. The feature rank catches up to its real value once the roll in vanishes. And finally, we also see that the capacity regularization in *Capacity* directly prevents the feature rank from dropping too low.

5 Conclusion

Informative reward functions simplify the exploration problem, reducing non-stationarity and improving generalization to downstream tasks. While they also improve capacity utilization as measured by penultimate layer feature rank measures, we find that such measures, when directly optimized for, stop being good measures. To summarize our main experimental findings: First, we find that online distillation via informative rewards is effective in improving generalization in curriculum learning and, in particular, more effective than other methods that directly target non-stationarity or capacity measures. Second, we found that informative rewards lead to improved capacity utilization by visualizing the penultimate layer feature ranks. Further, we find that feature ranks are good capacity measures only when they are not optimized for. Finally, we validated that the improvements in generalization relate to the reduction in learning from task-irrelevant states by measuring state visitation counts.

Limitations and future outlook. First, while our method improves upon other curriculum learning baselines, the curriculum still needs to be provided manually, and we do not address the problem of curriculum generation. Furthermore, while capacity is efficiently utilized, we do not have a way of increasing capacity on the fly, apart from initializing the retraining process with a larger network. On the flip side, the online distillation using a learned reward function makes it agnostic to the learning algorithm used. This makes our method complementary to improvements in learning algorithms. Although we use complex visual environments with procedural generation, our analysis only covers goal reaching tasks in discrete environments. Furthermore, the proximity function is not an ideal way to minimize task-irrelevant exploration for online distillation as it does not always produce students which perform better than the teachers, but for the purpose of our experiments (since we take the best performing seed out of 3), we found it sufficient. Identifying the ideal way to learn such informative rewards is left for future work.

Acknowledgements

The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AI-MOTION (LIT-2018-6-YOU-212), AI-SNN (LIT-2018-6-YOU-214), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), AIRI FG 9-N (FWF-36284, FWF-36235), ELISE (H2020-ICT-2019-3 ID: 951847). We thank Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, University SAL Labs initiative of Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, Software Competence Center Hagenberg GmbH, TÜV Austria, Frauscher Sensonic, AI Austria Reinforcement Learning Community and the NVIDIA Corporation. We would also like to acknowledge all the collaborators at IARAI and special thanks to Louis Kirsch and Marco Gallieri for discussions on an earlier version of this project.

References

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. (2019). Solving Rubik's Cube with a Robot Hand. *arXiv:1910.07113 [cs, stat]*.

Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J., and Hochreiter, S. (2019). Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32.

Ash, J. T. and Adams, R. P. (2019). On the Difficulty of Warm-Starting Neural Network Training. *arXiv:1910.08475 [cs, stat]*.

Becker-Ehmck, P., Karl, M., Peters, J., and van der Smagt, P. (2020). Learning to Fly via Deep Model-Based Reinforcement Learning. *arXiv:2003.08876 [cs, stat]*.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, Montreal, Quebec, Canada. ACM Press.

Benjamin, A. S., Rolnick, D., and Kording, K. (2018). Measuring and regularizing networks in function space. *arXiv preprint arXiv:1805.08289*.

Chevalier-Boisvert, M., Willems, L., and Pal, S. (2022). Minimalistic Gridworld Environment (MiniGrid). Technical report, Farama Foundation.

Dapena, J. (2002). The evolution of high jumping technique: biomechanical analysis. In *ISBS-Conference Proceedings Archive*.

Dinu, M.-C., Hofmarcher, M., Patil, V. P., Dorfer, M., Blies, P. M., Brandstetter, J., Arjona-Medina, J. A., and Hochreiter, S. (2022). *XAI and Strategy Extraction via Reward Redistribution*, pages 177–205. Springer International Publishing, Cham.

Fedus, W., Ghosh, D., Martin, J. D., Bellemare, M. G., Bengio, Y., and Larochelle, H. (2020). On Catastrophic Interference in Atari 2600 Games.

Florensa, C., Held, D., Geng, X., and Abbeel, P. (2018). Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR.

Hochreiter, S. and Schmidhuber, J. (1994). Simplifying neural nets by discovering flat minima. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press.

Holzleitner, M., Gruber, L., Arjona-Medina, J., Brandstetter, J., and Hochreiter, S. (2020). Convergence Proof for Actor-Critic Methods Applied to PPO and RUDDER. *arXiv:2012.01399 [cs, math]*.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. (2021). Transient Non-Stationarity and Generalisation in Deep Reinforcement Learning. *arXiv:2006.05826 [cs, stat]*.

Jabri, A., Hsu, K., Gupta, A., Eysenbach, B., Levine, S., and Finn, C. (2019). Unsupervised curricula for visual meta-reinforcement learning. *Advances in Neural Information Processing Systems*, 32.

Kumar, A., Agarwal, R., Ghosh, D., and Levine, S. (2020). Implicit Under-Parameterization Inhibits Data-Efficient Deep Reinforcement Learning. *arXiv:2010.14498 [cs, stat]*.

Lee, Y., Szot, A., Sun, S.-H., and Lim, J. J. (2021). Generalizable Imitation Learning from Observation via Inferring Goal Proximity. In *Advances in Neural Information Processing Systems*.

Lyle, C., Rowland, M., and Dabney, W. (2022). Understanding and Preventing Capacity Loss in Reinforcement Learning. *arXiv:2204.09560 [cs]*.

OpenAI (2018). OpenAI Five. https://openai.com/blog/openai-five/.

Patil, V., Hofmarcher, M., Dinu, M.-C., Dorfer, M., Blies, P. M., Brandstetter, J., Arjona-Medina, J., and Hochreiter, S. (2022). Align-RUDDER: Learning from few demonstrations by reward redistribution. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17531–17572. PMLR.

Peterson, G. B. (2004). A day of great illumination: Bf skinner's discovery of shaping. *Journal of the experimental analysis of behavior*, 82(3):317–328.

Portelas, R., Colas, C., Weng, L., Hofmann, K., and Oudeyer, P.-Y. (2020). Automatic Curriculum Learning For Deep RL: A Short Survey. *arXiv:2003.04664 [cs, stat]*.

Roy, O. and Vetterli, M. (2007). The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE.

Schmidhuber, J. (1991). Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463 vol.2.

Schmidhuber, J. (2012). POWERPLAY: Training an Increasingly General Problem Solver by Continually Searching for the Simplest Still Unsolvable Problem. *arXiv:1112.5309 [cs]*.

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

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv:1712.01815 [cs]*.

Siripurapu, R., Schmidhuber, J., and Kirsch, L. (2020). Curriculum learning through distilled discriminators.

Skinner, B. F. (1958). Reinforcement today.

Steinparz, C., Schmied, T., Paischer, F., Dinu, M.-C., Patil, V., Bitto-Nemling, A., Eghbal-zadeh, H., and Hochreiter, S. (2022). Reactive exploration to cope with non-stationarity in lifelong reinforcement learning.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. (2018). Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. *arXiv:1804.10332 [cs]*.

Uchendu, I., Xiao, T., Lu, Y., Zhu, B., Yan, M., Simon, J., Bennice, M., Fu, C., Ma, C., Jiao, J., Levine, S., and Hausman, K. (2022). Jump-Start Reinforcement Learning. *arXiv:2204.02372 [cs]*.

Wang, R., Lehman, J., Clune, J., and Stanley, K. O. (2019). Poet: open-ended coevolution of environments and their optimized solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 142–151.

Widrich, M., Hofmarcher, M., Patil, V. P., Bitto-Nemling, A., and Hochreiter, S. (2021). Modern Hopfield Networks for Return Decomposition for Delayed Rewards. In *Deep RL Workshop NeurIPS 2021*.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In conclusion.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We do not anticipate any direct negative impacts
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[TODO]** We will include upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Major ones reported which are different from baselines. However, we would like to submit full data after acceptance.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **[TODO]** We will mention after acceptance.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? **[TODO]**Upon acceptance
   (b) Did you mention the license of the assets? **[TODO]**Upon acceptance
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]