Seeking Visual Discomfort: Curiosity-driven Representations for Reinforcement Learning

Elie Aljalbout¹, Maximilian Ulmer¹ and Rudolph Triebel¹,²

Abstract—Vision-based reinforcement learning (RL) is a promising approach to solve control tasks involving images as the main observation. State-of-the-art RL algorithms still struggle in terms of sample efficiency, especially when using image observations. This has led to increased attention on integrating state representation learning (SRL) techniques into the RL pipeline. Work in this field demonstrates a substantial improvement in sample efficiency among other benefits. However, to take full advantage of this paradigm, the quality of samples used for training plays a crucial role. More importantly, the diversity of these samples could affect the sample efficiency of vision-based RL, but also its generalization capability. In this work, we present an approach to improve sample diversity for state representation learning. Our method enhances the exploration capability of RL algorithms, by taking advantage of the SRL setup. Our experiments show that our proposed approach boosts the visitation of problematic states, improves the learned state representation, and outperforms the baselines for all tested environments. These results are most apparent for environments where the baseline methods struggle. Even in simple environments, our method stabilizes the training, reduces the reward variance, and promotes sample efficiency.

I. INTRODUCTION

To solve complex tasks in unstructured environments, agents should be capable of learning new skills based on their understanding of their surroundings. Vision-based reinforcement learning is a promising technique to enable such an ability. These methods learn mappings from pixels to actions and may require millions of samples to converge, especially for physical control tasks [5]. This sample inefficiency could be attributed to the complexity of the dynamics encountered in such environments, but also to the difficulty of processing raw image information.

A recent paradigm to approach the latter problem is to enforce meaningful mid-level representations, via integrating perception modules in the RL pipeline. These modules are trained either in a supervised [8] or self-supervised/unsupervised fashion [9]. While supervised methods are simpler and easier to train, they require access to labeled datasets, which are usually hard to obtain, especially for real-world robotics scenarios. Thus, unsupervised and self-supervised approaches are the most popular ones in recent work. In these settings, the main goal is to integrate state representation learning objectives in the RL process [4], [41]. In contrast to end-to-end methods, approaches that leverage SRL explicitly encourage the policy to learn a state representation mapping based on observations. The additional objective improves sample efficiency as it provides an extra signal for training. However, during RL, the agent performs several trials to achieve a certain behavior. This trial and error process, together with the exploitative nature of RL algorithms could result in very similar samples being collected in the replay buffer. This lack of diversity can harm the generalization capability of the learned encoders and hinder the improvement in sample efficiency that could be achieved with SRL. Hence, data diversity could be very beneficial for vision-based RL, and exploration strategies tailored for diverse and SRL-problematic observations could boost sample efficiency even further.

In this work, we aim at improving the sample diversity of vision-based RL. We present an approach for exploration that makes the agent specifically curious about the state representation. Our approach takes advantage of the off-policy property of most state-of-the-art RL algorithms and trains a separate curious policy based on the SRL error. A preliminary version of this work can be found in [3]. Our experiments show that the proposed method encourages the visitation of SRL-problematic states. Additionally, it improves the performance of downstream tasks, especially for environments where recent approaches struggle. It also stabilizes the training and reduces the reward variance for all environments. Our contributions can be summarized as follows:

- We present an approach for learning policies that are curious about the state representation.
- Our approach is independent of the choice of SRL methods.
- We demonstrate how the curious policy can be integrated into a vision-based RL pipeline.
- Our method improves the exploration, training stability, and overall performance of vision-based RL.
- Our approach enables learning previously unsolved vision-based RL tasks on the deepmind control suite (DMC).
- Our implementation will be made open-source upon publication.

II. RELATED WORK

Integrating SRL. One popular approach for SRL is the autoencoder (AE) [4]. One of the earliest works to integrate AEs in batch-RL can be found in [21]. Later work explored the use of variational AEs [16] as well as regularized autoencoders (RAE) [41]. AEs could either be trained simultaneously with the policy [9], [41], or in certain cases, separately pretrained before the RL start [2], [16], [23].
Replay Buffer

In environments with rewards that are easy to encounter, but importantly, AE-based approaches lead to a more explainable task-specific objectives such as contact prediction [23]. More able multi-modal and multi-view fusion [1], [23], as well as self-supervised objectives such as jigsaw puzzle [25], advantages. They are simple to implement, allow for integration of AE-based approaches [22], [34], these methods have many advantages. They are simple to implement, allow for integrating self-supervised objectives such as jigsaw puzzle [25], enable multi-modal and multi-view fusion [1], [23], as well as task-specific objectives such as contact prediction [23]. More importantly, AE-based approaches lead to a more explainable state representation, especially when using generative AEs.

Curiosity in RL. Classical deep RL algorithms work well in environments with rewards that are easy to encounter, but tend to fail once high-reward areas are harder to reach [7]. This clearly motivates the use of exploration techniques as a means to achieve this goal. Popular paradigms for exploration include counts and pseudo-counts [6], [28], learning distributions over value functions or policies [27], and information gain based methods [17], [31], [32]. While most of these approaches aim at improving the diversity in the replay buffer for improving the RL itself, our goal is to emphasize exploring states in which the SRL module struggles. This could help the agent in learning a more representative state space which would subsequently improve the RL. In addition, most previous approaches depend on the prediction error of a dynamics model. In contrast, we leverage the SRL error for intrinsic motivation. This enables seamless integration of exploration in vision-based RL without any need for training additional dynamics models in the process. Furthermore, this creates an interplay between the SRL and the exploration which are both crucial aspects of successful and sample efficient vision-based RL.

Most closely related to our method is the work in [33]. This work attempts to maximize state entropy using random convolutional encoders. The method uses a k-nearest neighbor entropy estimator in the representation space and uses this estimation as an additional intrinsic reward bonus for RL. Similar to our work, their approach doesn’t require any dynamics models for training. However, using a k-nearest neighbor entropy estimator could be either compute-expensive if all observations need to be embedded at each step, or memory-expensive when those embeddings are saved in the replay buffer. Furthermore, a random encoder doesn’t guarantee any notion of meaningful similarity between observations. In fact, in certain degenerate cases, the similarity in the representation space of a random encoder could be a measure of dissimilarity of the states.
III. BACKGROUND

Reinforcement Learning (RL) is a computational approach to automate policy learning by maximizing cumulative reward in an environment [36]. RL tasks are usually formulated as Markov Decision Processes (MDP). A finite-horizon, discounted MDP is characterized by the tuple $\mathcal{M} = (S, A, \mathcal{P}, r, \rho_0, \gamma, T)$, where the state and action spaces are respectively $S$ and $A$, transition dynamics $\mathcal{P} : S \times A \rightarrow S$, reward $r : S \times A \rightarrow \mathbb{R}$, an initial state distribution $\rho_0$, discount factor $\gamma \in [0, 1]$, and horizon $T$. The optimal policy $\pi : S \rightarrow P(A)$, maximizes the expected discounted reward:

$$J(\pi) = E_{\pi} \left[ \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) \right]$$

State Representation Learning (SRL). While representation learning methods focus on learning abstract features from observations, SRL aims at learning low-dimensional features as state representations that are suitable for control. Namely, the goal of SRL is to learn a mapping from observations to state representations $q : \mathcal{O} \rightarrow \mathcal{Z}$, where $\mathcal{O}$ is the observation space and $\mathcal{Z}$ the embedding space. The mapping can also have as input a history of observations [24]. In fully observable environments, SRL methods could attempt to recover the true state (depending on its definition). However, in partially observable settings, SRL aims at finding latent representations of the state.

In this work, we focus on two methods for SRL. The first one is the regularized autoencoder (RAE) [10]. RAE was introduced as a deterministic alternative to variational autoencoders (VAE) [15], [20]. It is trained using the following loss:

$$L_{SRL}(RAE) = \mathbb{E}_{o \sim D} [\log p_\theta(o | z) + \lambda_2 ||z||^2 + \lambda_0 ||\theta||^2]$$

RAEs preserve the regularization properties of VAEs by explicitly penalizing the learned representation $z = g_\phi(o)$ and the decoder weights $\theta$, where $o$ is the image observation, $g_\phi$ is the encoder, and $\lambda_2, \lambda_0$ are hyperparameters which respectively specify the influence of the $L_2$ penalty on $z$ and the weights decay for the decoder parameters. We choose RAEs over other autoencoder methods as they were previously shown to yield better performance when integrated with vision-based RL [41].

The second method we consider is based on contrastive learning (CL) [14], [26]. CL approaches learn representations based on similarity constraints pushing similar (positive) samples to be closer in the representation space and dissimilar (negative) ones to be further apart. In this work, we use the InfoNCE loss for CL [26]:

$$L_{SRL}(CL) = \log \frac{\exp q^T W k_r}{\exp q^T W k_+ + \sum_{i=0}^{K-1} q^T W k_i}$$

where $q$ is the anchor, $\{k_i\}_{i=0}^{K}$ are all the targets including one positive $k_r$, and $K-1$ negatives. We follow the work in [22], and use instance discrimination [40] for generating positive and negative keys. This means that the anchor and positive samples are augmentations of the same observation, while the negatives correspond to all other samples in the batch. Similar to [22], we use random crops as the main source of augmentations.

IV. LEARNING CURIOSITY-DRIVEN REPRESENTATIONS

A. General Formulation

Recent work on vision-based RL leverage SRL objectives to improve the sample efficiency of policy search methods [9], [22], [41]. There are two main ways to integrate SRL in reinforcement learning. The first one is to simultaneously update both objectives, and the second is to train the two modules in an alternating fashion [9]. The second option could also mean that SRL is only used to train a feature encoder in a pretraining phase preceding the actual RL. In both cases, the quality of the learned encoder and the resulting representations play a central role in the downstream RL tasks. With a finite amount of data, it is not always possible to collect enough samples to learn a representation that is valid across the state subspace relevant to the task at hand. For instance, in environments with sparse rewards, the SRL training rarely encounters observations corresponding to high-reward regions and their surroundings. The resulting representations for such observations might lack the necessary information for the policy to learn any useful behavior.

In general, this lack of coverage is mostly attributed to the exploitative nature of RL algorithms, which leads to the replay buffer containing a lot of redundant and similar observations. Hence, to improve the quality of the feature extraction and learned representations, it is important to encourage collecting data in states outside of the comfort zone of the SRL model. Formally, that would correspond to maximizing the expected SRL error in the replay buffer $D$:

$$\max E_o [L_{SRL}(o)]$$

where $p_o$ is the distribution of the observations in the replay buffer and $o$ is an observation.

B. CuRe: Curiosity-Driven Representations

Observations in the replay buffer are part of observation-action trajectories of length $T$ with $p(o_0, a_0, ... o_T, a_T) = p(o_0) \prod_{t=0}^{T} \pi(a_t | o_t) p(o_{t+1} | o_t, a_t)$ (we omit the generative process $p(o_t | s_t)$ for simplicity). Hence, only the initial state distribution and policy are relevant for $[4]$, since the system dynamics are dependent on the environment and cannot be altered. In RL settings, the initial state distribution is dependent on the environment resetting mechanism. Although it’s interesting to study the effect of this mechanism on performance, we leave this for future work.

Instead, in this paper, we learn curious policies that maximize $[4]$. This corresponds to training a policy $\pi_{\text{cure}}$ to maximize the objective in $[4]$ with $\gamma = 0$ and the weighted SRL error as an intrinsic reward $r_{\text{cure}} = \beta L_{SRL}$. $\beta$ is a hyperparameter that specifies the degree of curiosity. In our experiments we fix $\beta = 1$. Furthermore, we allow $\gamma$ to have
values different than zero as it didn’t show any negative influence on the training.

C. Integrating CuRe in RL

There are two main ways to integrate CuRe in a vision-based RL algorithm. Namely, the intrinsic reward could either be added to the task reward $r_{task}$ to train the main policy $\pi_{task}$, or used separately for training a separate curious policy $\pi_{cure}$. Previous methods mostly use the earlier approach to integrate intrinsic rewards (based on a dynamics model) [31].

In this work, we choose the option with two separate policies. By doing so, we ensure that the task policy is purely optimizing the task reward, and additionally obtain a representation-curious capable of exploration for similar tasks in the same environment. More importantly, this choice allows our method to be used with both simultaneous and alternating approaches to SRL integration in RL. In addition, our early experiments indicate that a separate curious policy leads to substantially higher reward areas, while the single policy approach could deteriorate the results in comparison to the baselines. Furthermore, adding the rewards together usually introduces extra hyperparameters to weigh the different terms (e.g. in [29], 3 extra hyperparameters are needed). It is important to note that having a separate policy is only possible when using off-policy RL algorithms such as soft-actor-critic (SAC) [12], which is why we use this method in this work.

Our overall approach is illustrated in Figure 1. CuRe is agnostic to the choice of the SRL algorithm. Besides the encoder and the two policies, our architecture includes an SRL model. This model could refer to different modules depending on the SRL approach used. For instance, when using an AE-based method, it would correspond to a decoder. It could also refer to a dynamics model, an identity transformation, as well as any computational block that is used by representation learning methods to constrain the latent space. Furthermore, the updates of both policies affect the encoder parameters $\phi$. The SRL model parameters $\theta$ are only affected by the SRL update. At every step, we either sample actions from the main policy or the curious one. The choice of which policy to use at every step is based on a hyperparameter $p_c$ which specifies the percentage of times exploration actions should be sampled. Intuitively, the curious policy is trained to reach states which have high SRL error. By occasionally sampling actions from this policy, the replay buffer ends up containing more problematic and diverse samples which helps to learn a better representation and to avoid overfitting. This interaction between the curious policy and the SRL model/loss results in an interplay similar to the one observed in generative adversarial networks [11], as both modules are mutually beneficial to each other, and are trained in an adversarial setting. This interplay is illustrated in figure 1. The overall approach is summarized in algorithm 1.

V. EXPERIMENTS

We design experiments to answer the following questions:

(Q1) Can we train a curious policy to increase the visitation of high SRL error states?

(Q2) How does CuRe affect the performance, sample efficiency, and training stability of vision-based RL methods? Can CuRe be successfully integrated with multiple SRL methods?

(Q3) Does CuRe-driven SRL pretraining improve the performance of vision-based RL on downstream tasks?

A. Setup & Baselines

To answer these questions, we experimentally evaluate our method on six continuous control tasks from the DeepMind Control Suite [37]. The chosen tasks aim to cover a wide range of common RL challenges, such as contact dynamics and sparse rewards. The tasks we use are reacher_easy, cartpole_swingup, ball_in_cup, finger_spin, finger_turn and reacher_hard. As deep learning models could be energy inefficient [35], we use only subsets of these tasks for minor experiments that are only aimed at validating simple aspects of our method.

The main goal of our experiments is to validate the effectiveness of CuRe on improving the performance of already existing SRL-based approaches to vision-based RL. To do so we use two such algorithms as baselines and compare their performance with and without CuRe. To validate, that the method is agnostic to the choice of SRL algorithms, we experiment with two different methods. Namely we use a

**Algorithm 1**

for each timestep $t = 1...T$ do

if $\epsilon < p_c$ then

$a_t \sim \pi_{cure}(\cdot|o_t)$

else

$a_t \sim \pi_{task}(\cdot|o_t)$

$o_{t+1} \sim p(\cdot|o_t, a_t)$

$D \leftarrow D \cup (o_t, a_t, r_{task}(o_t, a_t), o_{t+1})$

$B \leftarrow \text{SampleBatch}(D)$

$r_{cure} \leftarrow \text{UpdateSRL}(B)$

$\text{UpdateTaskAC}(B)$

$\text{UpdateCuriousAC}(B, r_{cure})$

Fig. 2: State representation learning (SRL) error encountered in trajectories sampled with three different policies: random, sac_ae and our curious policy (cure). The bars represent the mean error per step. The error bars represent the minimum and maximum encountered errors. Our method leads to the visitation of high SRL error states, around two orders of magnitudes more than the random and task policies (sac_ae).
Fig. 3: Training curves on six continuous control tasks from the DeepMind Control Suite [37]. The plots show the mean episode rewards of two algorithms. The first one is a baseline (sac ae). The second method combines the same baseline with CuRe (sac ae+cure). In all environments, our method exceeds the performance of the baseline. For easier tasks, the curious exploration either stabilizes the training or improves the maximum achieved reward. For the more difficult tasks, such as finger_spin, finger_turn and reacher_hard, the additional curiosity objective allows to improve the average reward, where the baseline fails to reach high-reward areas.

B. Results

Visiting High SRL Error Regions. Figure 2 shows the SRL error encountered when sampling actions from three different policies. The first policy generates random actions within the action space of the environment. The second one is trained with sac ae, and the last one is a CuRe-based curious policy that maximizes the SRL error without a task reward. While random and sac ae have similar mean errors per step, our method leads to the visitation of states which have on average an SRL error that is around two orders of magnitude higher. This confirms that CuRe fulfills its goal of increasing the probability of visiting high SRL error states. The exact values are shown in Table III in the appendix.

CuRe-based Exploration During RL. To answer (Q2), we study the effect of integrating CuRe into two different baselines, namely sac ae and curl. The integration is based on algorithm 1. Figure 3 shows the task reward for sac ae with and without CuRe. In all environments, our method exceeds the performance of the baseline. Specifically, for tasks where the baseline doesn’t show any signs of improvement, such as reacher_hard and finger_turn, CuRe leads to exploring high-reward areas, as can be seen when looking at the maximum rewards achieved in those environments. For simpler tasks such as reacher_easy and finger_spin, our method approaches the maximum environment rewards, while sac ae converges to 80%. In addition, CuRe stabilizes the training and reduces the reward variance significantly.

We chose those two SRL methods since their integration in RL is fairly recent while also being well-established in robotics applications. We refrain from comparing our approach to classical exploration methods, since the two have different goals: classical exploration in RL is concerned with improving the sample diversity for RL while our method is aimed at encouraging the visitation of SRL-problematic states (discomfort zones). Hence comparing methods from these two categories could be misleading. Both baselines and our method are implemented using PyTorch [30]. For simplicity, we use the same hyperparameters for all experiments except for the action repeat value which changes per task, according to [13]. The actor and critic networks for the RL agent and the curious agent are trained using the Adam optimizer [19], using default parameters. We store trajectories of experiences in a standard replay buffer. For implementing SAC, we follow the training procedure detailed in [41]. For the sake of reproducibility, we provide more information about the training procedure, and an overview of the hyperparameters in Appendix A. Our experiments required a training period of over four months on 6 GPUs (NVIDIA RTX2080 and RTX3090).
TABLE I: Comparison of the performance (in terms of episode reward) of different versions of sac\_{ae}: vanilla is the original algorithm [41], random-pretraining and CuRe-pretraining refer to the cases where the vanilla procedure is preceded by an RAE pretraining phase using data collected with a random policy and a CuRe-based policy respectively.

| methods          | cartpole\_swingup | ball\_in\_cup | finger\_spin | reacher\_easy | reacher\_hard | finger\_turn |
|------------------|-------------------|--------------|-------------|--------------|--------------|--------------|
| vanilla          | 833\pm 27         | 953 \pm 4    | 820 \pm 144 | 714 \pm 113  | 169 \pm 179  | 229 \pm 135  |
| random-pretraining| 784 \pm 12        | 955 \pm 10   | 975 \pm 3   | 615 \pm 129  | 84 \pm 33    | 256 \pm 40   |
| CuRe-pretraining | 846 \pm 25        | 504 \pm 187  | 981 \pm 7   | 804 \pm 52   | 431 \pm 40   | 402 \pm 58   |

This last feature is not given enough attention in RL research. However, in real-world scenarios, when deploying RL agents, there could be cases where only one training run is possible. An algorithm with lower reward variance could guarantee a sufficiently good policy, while it's hard to say the same when this condition fails. This effect can also be seen for cartpole\_swingup and ball\_in\_cup. We observe that CuRe has a minor effect on the maximum reached reward for these last two environments. This could be attributed to the already good performance of the baseline on these tasks. In fact, in these environments, sac\_ae already approaches the performance achieved by SAC trained with the true states [41]. Nonetheless, the additional curious exploration objective accelerates the convergence of all evaluation tasks, thus improving the sample efficiency, which is one key limitation of state-of-the-art model-free algorithms. In general, our experiments show that CuRe becomes more effective when the task complexity increases.

To study the effect of CuRe on curl [22], we run experiments on the four environments where CuRe had the most influence on sac\_ae. Figure 4 shows the reward plots for curl with and without CuRe. Similar to our previous results, CuRe has a positive impact on the overall performance, sample efficiency, reward variance and stability of training. This improvement is not as big as the one observed in our sac\_ae experiments. However, this difference is understandable, since curl is a more recent algorithm and has previously shown better results on similar deepmind control suite tasks [22]. Despite that, when looking at results on finger\_turn (Figures 3 and 4), CuRe applied to sac\_ae reaches a higher final episode reward than vanilla curl. Additionally, we observe that sac\_ae+cure has a better sample efficiency than curl in the finger\_spin environment.

Effect of Pretraining. In addition to our main results, to assess the quality of the learned representation with CuRe, and to answer (Q3), we study the effect of two different pretraining procedures on sac\_ae. Namely, we look at pretraining the RAE using samples collected either using a random policy (random-pretraining) or using a policy trained with CuRe only, without any task reward (CuRe-pretraining). For both options, we perform the pretraining for 500 thousand steps. We also compare the performance of those two variants to the case where no pretraining is performed at all (vanilla). The results are shown in Table I. For all six environments, the best results are obtained when using one of the two pretraining mechanisms. In most cases, CuRe-based pretraining leads to better performance than random-pretraining. This become especially apparent for tasks where the vanilla method struggles, such as reacher\_hard and finger\_turn. However, for the ball\_in\_cup environment, CuRe-pretraining seems to deteriorate the performance when compared to both vanilla and random-pretraining. This could be attributed to the simplicity of the task, which reduces the need for SRL and SRL-tailored exploration. In general, although CuRe is beneficial for both SRL pretraining (Table I) and RL (figure 5), we observe that it is more
effective during task learning than in the pretraining phase.

VI. CONCLUSION

We introduce CuRe, a curiosity-based exploration technique that can be easily used together with state representation learning methods used in RL. This method exploits the SRL error to incentivize visiting more diverse and problematic states. We extensively evaluate our method on complex continuous control tasks in simulation. Our results show that our curious exploration method improves the performance of vision-based RL based on two different SRL methods. When comparing the baselines to the curiosity-driven extensions, we show that the added curiosity improves the performance in terms of speed of convergence, stability, and the total achieved reward. In future work, we plan to experiment with the transfer learning capability of our architecture, scale it up to multi-modal tasks and perform real-world experiments.

REFERENCES

[1] Iretiayo Akinola, Jacob Varley, and Dmitry Kalashnikov. Learning precise 3d manipulation from multiple uncalibrated cameras. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 4616–4622. IEEE, 2020.

[2] Elie Aljadjbout, Ji Chen, Konstantin Ritt, Maximilian Ulmer, and Sam Haddadin. Learning vision-based reactive policies for obstacle avoidance. arXiv preprint arXiv:2010.16298, 2020.

[3] Elie Aljadjbout, Maximilian Ulmer, and Rudolph Triebel. Making curiosity explicit in vision-based rl. arXiv preprint arXiv:2109.13588, 2021.

[4] Dana H Ballard. Modular learning in neural networks. In AAI, pages 279–284, 1987.

[5] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruba TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributional policy gradients. In International Conference on Learning Representations, 2018.

[6] Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. arXiv preprint arXiv:1606.01868, 2016.

[7] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In International Conference on Learning Representations, 2019.

[8] Bryan Chen, Alexander Sax, Gene Lewis, Iro Armeni, Silvio Savarese, Amir Zamir, Jitendra Malik, and Lerrel Pinto. Robust policies via mid-level visual representations: An experimental study in manipulation and navigation. arXiv preprint arXiv:2011.06698, 2020.

[9] Tim de Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. Integrating state representation learning into deep reinforcement learning. IEEE Robotics and Automation Letters, 5(3):1394–1401, 2018.

[10] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael Black, and Bernhard Scholkopf. From variational to deterministic autoencoders. In International Conference on Learning Representations, 2020.

[11] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2, pages 2672–2680, 2014.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International Conference on Machine Learning, pages 1861–1870. PMLR, 2018.

[13] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In International Conference on Machine Learning, pages 2553–2565. PMLR, 2019.

[14] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In Proceedings of the 37th International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR, 2020.

[15] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In International Conference on Learning Representations (ICLR), 2017.

[16] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In International Conference on Machine Learning, pages 1480–1490. PMLR, 2017.

[17] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. arXiv preprint arXiv:1605.09674, 2016.

[18] Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. Autonomos Robots, 59(3):407–428, 2015.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[20] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In International Conference on Learning Representations (ICLR), 2014.

[21] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In The 2010 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2010.

[22] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curie: Count-based unsupervised representations for reinforcement learning. In International Conference on Machine Learning, pages 5639–5650. PMLR, 2020.

[23] Michelle A Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In 2019 International Conference on Robotics and Automation (ICRA), pages 8943–8950. IEEE, 2019.

[24] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-François Goudou, and David Filliat. State representation learning for control: An overview. Neural Networks, 108:379–392, 2018.

[25] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In European conference on computer vision, pages 69–84. Springer, 2016.

[26] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018.

[27] Ian Oshband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. arXiv preprint arXiv:1602.04621, 2016.

[28] Georg Ostrovski, Marc G Bellemare, Aarón Oord, and Rémi Munos. Count-based exploration. In International Conference on Machine Learning, pages 2721–2730. PMLR, 2017.

[29] Eric Pairet, Paola Ardón, Frank Broz, Michael Mistry, and Yvan Petillot. Learning and generalisation of primitives skills towards robust dual-arm manipulation. arXiv preprint arXiv:1904.01568, 2019.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fan, Jingjie Bie, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems. 2019.

[31] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In International Conference on Machine Learning, pages 2778–2787. PMLR, 2017.

[32] Juergen Schmidhuber. Curious model-building control systems. In Proc. international joint conference on neural networks, pages 1458–1463, 1991.

[33] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for coupled representation learning from reinforcement learning. In Proc. international joint conference on neural networks, pages 69–84. Springer, 2016.

[34] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. arXiv preprint arXiv:2009.08319, 2020.

[35] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:1906.02243, 2019.

[36] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.
Appendices

APPENDIX A IMPLEMENTATION DETAILS & HYPERPARAMETERS

For the encoder and decoder, we employ the architecture from [41]. Both consist of four convolutional layers with $3 \times 3$ kernels and 32 channels and use ReLU activations, except for the final deconvolution layer. Both networks use a stride of 1 for each layer except for the first of the encoder and the last of the decoder, which use stride 2.

For all shown experiments, we train with multiple seeds for each task. At the beginning of each seed, we pretrain the models with 1000 samples which we collect by rolling out random actions. Afterwards, we evaluate the model every 10 thousand environment steps over 10 episodes and report the average reward. The total number of episodes depends on the complexity of the task. All hyperparameters used in our experiments are summarized in Table II.

APPENDIX B ADDITIONAL RESULTS

Here we show the exact values obtained for the SRL error visitation experiment in section V-B. These values were roughly illustrated in figure 2 and exactly shown in Table III.

| Parameter                      | Setting |
|-------------------------------|---------|
| Batch size                    | 128     |
| Replay buffer capacity        | 80000   |
| Discount $\gamma$             | 0.99    |
| Hidden dimension              | 1024    |
| Curious exploration probability $p_c$ | 0.2    |
| Observation size              | $84 \times 84 \times 3$ |
| Frames stacked                | 3       |
| Critic learning rate          | $10^{-3}$ |
| Critic target update frequency| 2       |
| Critic soft target update rate $\tau$ | 0.01   |
| Actor learning rate           | $10^{-3}$ |
| Actor update frequency        | 2       |
| Actor log std bounds          | [-10, 2] |
| Autoencoder learning rate     | $10^{-3}$ |
| Decoder update frequency      | 1       |
| Temperature learning rate     | $10^{-4}$ |
| Init temperature              | 0.1     |

TABLE II: The hyperparameters used in our experiments.

| method/env          | Vals | random | sac_ac | cure |
|---------------------|------|--------|--------|------|
| reacher_easy        | Min  | 0.0001 | 0.0001 | 0.0390 |
|                     | Mean | 0.0002 | 0.0004 | 0.0399 |
|                     | Max  | 0.0003 | 0.0007 | 0.0424 |
| ball_in_cup         | Min  | 0.0005 | 0.0003 | 0.0762 |
|                     | Mean | 0.0006 | 0.0005 | 0.0774 |
|                     | Max  | 0.0008 | 0.0007 | 0.0783 |
| cartpole_swingup    | Min  | 0.0002 | 0.0002 | 0.0531 |
|                     | Mean | 0.0002 | 0.0002 | 0.0540 |
|                     | Max  | 0.0003 | 0.0003 | 0.0548 |
| finger_spin         | Min  | 0.0002 | 0.0004 | 0.0755 |
|                     | Mean | 0.0003 | 0.0012 | 0.0766 |
|                     | Max  | 0.0004 | 0.0015 | 0.0774 |
| finger_turn_easy    | Min  | 0.0003 | 0.0003 | 0.0749 |
|                     | Mean | 0.0004 | 0.0005 | 0.0759 |
|                     | Max  | 0.0006 | 0.0008 | 0.0769 |
| reacher_hard        | Min  | 0.0002 | 0.0002 | 0.0393 |
|                     | Mean | 0.0003 | 0.0004 | 0.0397 |
|                     | Max  | 0.0008 | 0.0006 | 0.0404 |

TABLE III: Mean, minimum and maximum SRL error encountered per step when using three different agents on six deepmind control suite tasks.