Embodied AI - Final Report

Lukas Rüttgers (2023403372), Simon Wilmes (2023403367)
Tsinghua University
ltgslks23@mails.tsinghua.edu.cn

1 Introduction

The ManiSkill2 challenge contains a broad spectrum of objects and tasks to tackle the problem of generalization in reinforcement learning (RL). Generalization here covers both the generalization to unseen tasks aswell as the generalization to unseen objects. Agents must therefore obtain robust abstractions of physical and visual properties of the objects, but also the task structure, to successfully generalize their skills in unseen environments.

After presenting the two winner algorithms from the last years, we also provide a review on the research on task generalization, since we feel that we especially lacked knowledge in this field. We present the research in this field and come up with own research questions and ideas which could improve the agent‘s performance in the ManiSkill setting. Finally, we also provide the videos and learning curve figures of the baselines provided by the ManiSkill authorship.

2 Prior work

2.1 Winning Algorithms of the Last Years

In the original ManiSkill challenges, multiple teams presented ways to achieve the necessary generalization in view of the numerous unseen tasks and objects. In the following, we give a brief summary of the winning algorithms for the Maniskill 2021 challenge and the Maniskill2 challenge, which was conducted one year later.

2.1.1 Maniskill 2021 Winner Algorithm

The winner algorithm of 2021 basically relies on three major changes to the baseline algorithms which were already provided along the challenge by Mu et al. (2021). While the task-specific rewards yield the potential to accurately guide the robot to the desired goal configuration, the authors argue that adopting to task-specific rewards would contradict the spirit of the challenge to generalize to a broad spectrum of unseen tasks. For this reason, they decide to not use the provided dense rewards. To avoid a sparse reward distribution, they adopt the Generative Adversarial Imitation Learning (GAIL) framework by Ho and Ermon (2016), where the learned policy acts as the generator G and the discriminator D tries to tell the expert trajectories and the trajectories from G apart. Most importantly, D provides dense rewards that scale with the success of G to imitate the expert and hence tackle the reward sparsity issue.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

However, not only the rewards, but also the expert trajectories follow a separate distribution for each task. The underlying mathematical models of the policy network naturally produce a uniform distribution of trajectories, G will hence struggle to account for the non-uniform distribution of expert trajectories. For this reason, they impose a handicap on D by throttling its model complexity.

Nonetheless, the embedding architecture of D remains the same over the entire evolution. As observation, D receives the robot state, the action and a point cloud with labeled segmentations of all parts that are involved in the specific task. The point cloud is masked for each part separately and forwarded through a dedicated PointNet. To account for the point that belong to neither of the parts, another two PointNets embed the rest of the point cloud and the entire point cloud respectively. Moreover, a Multilayer-Perceptron (MLP) extracts features out of the robot state and the applied action and together with a bias form the last two components of the embedding, which is finally concatenated.

This embedding is then forwarded through both a simple and a strong model, where the simple model merely selects the maximum as the final feature and the strong model leverages the encoding part of the transformer architecture proposed by Vaswani et al. (2017) to identify relations between the different part embeddings via self-attention. The final output of D is simply a linear interpolation between this two models which tends to the strong model over time.

Since this still does not account for the distribution offset between the non-uniform expert trajectories and the trajectories from G, the authors fill the expert buffer with successful trajectories of G. To ensure that a relatively easy task does not lead to flooding the expert buffer with trajectories from this task, a fixed amount of space is reserved for each task. The resulting buffer structure is coined as *Category-Level Instance Buffer* (CLIB).

2.1.2 ManiSkill2 2022 Winner Algorithm

Gao et al. (2023) sought out pragmatic solutions to these problems. Their core idea is to allow a more stable learning process by splitting the training process into two parts: a general learning phase, followed by a fine-tuning phase. The reasoning behind this split was that the authors noticed that after an initial surge in the success probability after a certain number of training steps, the algorithm would start to overfit and thereby reduce its success performance. To overcome this issue, they stored the best-performing network and used that network as the starting point for the second round of fine-tuning, in which they reduced the batch size and the number of samples in each step. The authors claim that this allows their algorithm to focus on more minor details.

The general setup of their algorithm is first to use the PointNet framework by Qi et al. (2017) to segment the point cloud captured by the robot into features that can directly be passed as input into a traditional network. They employed Behaviour Cloning as their learning strategy for imitation learning, while for Reinforcement Learning, they used Proximal Policy Optimization (PPO) (Schulman et al. (2017)).

Even though their optimization ideas are not fundamentally different from more standard ideas for this task, by making these small adjustments, this setup achieved first place in all three tracks, thereby showing the advantage of their approach.

2.2 Competence progress motivation for intrinsic rewards

Our ability to acquire a thorough understanding of the dynamics inherent in our environment is largely attributable to our intrinsic motivation that drives us to explore beyond our comfort zone.

Among several psychological theories of intrinsic motivation, Hunt (1965) famously coined intrinsic motivation as the strive for *optimal incongruity* between humans' expectation and their actual perception. In face of the sparsity of extrinsic rewards in long-horizon reinforcement learning (RL) tasks, Oudeyer and Kaplan (2007) derived intrinsic reward formulations from these psychological theories and associated the notion of incongruity with *learning progress motivation*, that generates high rewards for decreasing epistemic uncertainty in an agent's world model. Schmidhuber (1991) proposed to measure epistemic uncertainty as the prediction error of a forward dynamics predictor that is learned during exploration. Until today, most of the research on intrinsic rewards follows this paradigm. While Lakshminarayanan et al. (2017), Burda et al. (2018) and Pathak et al. (2019) estimate uncertainty as the disagreement between an ensemble of predictors, Sekar et al. (2020) imagine rollouts inside a latent space to estimate the expected novelty of a future state.

However, there is also another influential psychological theory around intrinsic motivation, which describes it as craving for competence (Deci and Ryan (1985)) or challenge (Csikszentmihalyi (2008, cop. 1990)). It claims that humans deliberately expose themselves to self-determined challenges that exhibit an adequate difficulty to efficiently foster their current aptitudes. In their transfer of this conceptual idea to RL, Oudeyer and Kaplan (2007) merely discussed how to express the difficulty of a challenge and culminated in a coarse distance-based metric, which they coined as *competence progress*. Stout and Barto (2010) concretized this in a limited setting, as they still assumed a finite state and action space and a given fixed set of challenges, where each challenge was defined by a given value function.

2.3 Project Motivation

Therefore, the computational formulation of challenge-based motivation is incomplete: how does the agent discover new tasks in a self-supervised manner? How can he efficiently determine his competence progress in high-dimensional infinite domains? And how does he integrate and keep track of the abundant number of possible skills in one policy? This requires lifting the goal from a mere parameter to part of the policy's input. But to efficiently represent such a policy, the agent must learn analogies between tasks and find patterns by which to compress their representation, similar to how Schmidhuber (2010) described the compression of novel patterns in observations.

Although these questions have received little attention in the last decade of research, we believe that competence-based intrinsic rewards can push agents to discover new skills and learn affinities between them, which in turn is an essential building block towards learning generalizable and robust behaviour.

After analysing and discussing possible computational architectures for intrinsic motivation through competence progress, we want to derive efficient implementations and finally verify their contribution to generalization to unseen tasks on the ManiSkill2 benchmark (Gu et al. (2023)).

3 Computational Architectures for Competence Progress Motivation

Firstly, we will construct an ideal, general architecture that could capture the dynamics of competence progress motivation. Because completely implementing this ideal formulation on our own requires engineering efforts that go beyond the scope of this project, we secondly derive an alternative architecture that builds upon prior work and enables us to realise the core components of our idea more efficiently. Throughout the upcoming sections in the report, we present ideas on an

intuitive level, and refer the interested reader to more extensive explanations and thoughts in the appendix.

3.1 Ideal Two-Party Formulation

Let us consider the dynamics of competence progress motivation as a symbiosis between two parties, the *challenger* C, and the *player* P. The goal of both parties is to maximize the competence of P. To that end, C is supposed to design a challenge-based curriculum for P, while P merely tries to perform as good as possible in these challenges.

3.1.1 Challenge description and criteria

To create an effective learning incentive, C should dynamically adapt the challenges to the current learning state of P. On the one hand, P will not profit from challenges that are already well-known and easy to him. On the other hand, challenges that are too complex will overwhelm P. Both cases will not lead to significant progress in this challenge. As originally modeled by Oudeyer and Kaplan (2007), P could therefore repeatedly attempt the challenge and measure his competence progress as the increase in reward over the last trials. The larger this increase is, the higher P's progress might be. P then reports this progress to C, who in turn can reflect on the adequacy of the current challenges, better understand the strengths and weaknesses of P, and thereupon improve her future challenge suggestions. Such a feedback is also indispensable to track the current competence state of P. The interaction between C and P hence resembles the interaction between a student and a teacher who actively adjusts the student's curriculum to his progress. Besides adequate difficulty, challenges should be meaningful with regard to the actual task that P is trained for.

Before we continue, we should first provide a formal definition of a challenge. The most naïve approach would regard a challenge as a state that ought to be reached. Because the ideal state itself is mostly inaccessible, we can instead represent the goal state by its associated observation from a fixed frame. To abstract from this detail, we refer to both as a state $q \in \mathcal{Q}$ in the following.

However, a challenge description may not only comprise *where* P shall get to. The challenge may also lie in *how* P is commanded to get there. To express more complicated challenges of such kind, we could include multiple landmarks as intermediate sub-goals that P should traverse up to some error on his way to the final goal state. For simplicity, we will however stick with the restricted definition of a challenge that only includes the observation of one final goal state. Besides that, it is crucial to also specify the initial state distribution of a challenge as we elaborate on in Appendix A.1.

3.1.2 Challenge discovery and evaluation

In order to serve as a useful teacher, C should have advantageous knowledge in comparison to P, otherwise it is unlikely that C can come up with challenges that are both meaningful to the task and enriching for P's competence progress. This information asymmetry is the main reason why to divide these roles into two different logical units.

While *Teacher-Student Architectures* like Rapid Motor Adaptation proposed by Kumar et al. (2021) provide the teacher with privileged state information, this privileged state information might not always be available. We therefore undertake a more general formulation where the challenger C itself has to acquire knowledge and talent in the same way as P, but might have an edge in experience and information. That is, while P's view is limited to the challenge-oriented rewards, C has access to the extrinsic reward function of the actual environment and can explore it by herself in the usual way.

Regions where C firstly perceives high extrinsic rewards, or secondly acquires new knowledge about the environment dynamics are both likely to be valuable for P too. To measure gain in knowledge, we can avail to the broad battery of computational formulations for *learning progress motivation* such as the epistemic uncertainty estimators mentioned in Section 2.2.

By this interaction with the true environment, C can assess which regions might be particularly attractive for P's curriculum. She can do so by integrating the aforementioned measurements into a *challenge quality score* $S : \mathcal{Q} \to \mathbb{R}$ that estimates the quality of setting a state $q \in \mathcal{Q}$ as the goal of a challenge. Most importantly, this function is not static but evolves over time, because the learning progress on which the score is partially based is also subject to change.

C may now greedily choose the state $q^* := \arg\max_{q \in \tau} S(q)$ with the highest score as candidate for the next challenge. More sophisticated approaches might also consider the scores of the entire associated trajectory $\tau \ni q^*$ or at least its predecessors and successors as the local context window – we borrow terminology from Natural Language Processing here – to integrate more global information.

3.1.3 Challenge update and multiple challenges

For now, let us assume C can pose only one challenge at each point in time. C stores the current challenge as the observation o_g of a certain goal state g in a shared memory that both C and P have access to. We will refer to this shared memory as the *challenge set* Ω.

After a challenge g was selected by C and undertaken by P, the challenge quality score $S(c)$ should be updated based upon the feedback from P. Specifically, the learning progress upon which C originally based the *estimated* competence progress of P is now step-wisely replaced by the *empirical* competence progress that P reports to C. This could be accounted for by a bilinear superposition $\alpha \cdot \hat{x} + (1 - \alpha) \cdot x$ where $\hat{x}$ is the competence progress estimate and x the empirically reported competence progress, and $\alpha \in [0, 1]$ a parameter that starts at 1 and continues to decay to 0 for each challenge trial P undergoes. The extrinsic rewards are static and do not need to be replaced in the score term.

This sets the stage to discuss under what conditions C updates the current challenges. Intuitively, this should be based upon their respective scores. If the current challenge g still proves more valuable in terms of $S(c)$ than the current challenge candidate q^*, then q^* may be stored in a buffer and taken into consideration in the upcoming update phases. However, as time passed by, the challenge quality score of a candidate $S(q)$ might become deprecated as it does not reflect the *current* competence and knowledge of P any more. For this reason, such candidate states should not be kept too long in such a buffer. Alternatively, their score $S(q^*)$ could be discounted over time by a factor λ.

The overall reward and information flow is again depicted in Figure 1. As it illustrates, Ω might in general comprise multiple challenges that introduces new degrees of freedom such as ordering challenges in Ω by their value. On the other side, it also necessitates a selection strategy on P's side. If this selection strategy is in turn controlled by C, then this is equivalent to the previous single-challenge setting.

3.1.4 Pseudo-rewards for selected challenges

By the nature of RL, C must accompany a challenge g with a proper reward function that lures P to g. With regard to the maxim *"the way [and not the goal itself] is the goal"*, we have to carefully consider *what* we actually want to teach P on the way to the goal when defining this reward function.

Figure 1: Schematic overview of the reward and information flow between C and P.

Besides the *extrinsic* reward from the environment, the proximity to the goal state in some distance metric would provide a dense reward signal. What's more, if there are some particular regions that C wants P to explore more thoroughly on the quest to g, then C can also integrate this bias into the reward function. In Section 4.4.3 on our implementation, we specify how we eventually compose our reward function for challenges. In the following, it suffices to assume a reward function that properly integrates all of the aforementioned aspects.

3.1.5 Considerations for implementation

In general, it might be tedious to infer P's current state of competence only from his competence progress feedback. To simplify the search for challenges of adequate difficulty, we provide C access to the current policy network of P. When taking actions during challenge exploration, C can now decide between taking an action based upon P's current policy or following her own – more proficient – exploration network. The more C will avail to P's policy network, the easier and well-known the traversed regions and challenges will become for P, hence a sound amount of exploration is indispensable for adequate challenge difficulty.

Another difficulty lies in the active curriculum that C has to design to teach P to finally achieve competence at the environment-specific task. Over the course of challenges, our changing artificial rewards might confuse P's policy updates and in the worst case lead to deliberate unlearning of useful knowledge it has achieved in prior challenges. Even without this pitfall, C needs to ensure that through the mastery of her challenges, P's policy eventually converges to a policy that fulfils the environment task. For that reason, we might at some point pose the environment goal state itself as the final challenge or integrate it frequently into Ω throughout the curriculum to ensure the P's policy is sufficiently prepared for transitioning to the final goal of the environment.

3.2 Off-Policy Alternative

While our original formulation could have worked, we also thought about a different architecture. Our main concern with the original idea was the high complexity as it consisted of multiple networks each with their own task, for which we sometimes even did not know if their specific goal was realistic. A more efficient and stable way to realise our core ideas was to build upon an already existing work and modify it to suit our needs. In our case, this was Hindsight Experience Replay (HER) by Andrychowicz et al. (2017) as their idea contains similarities that we thought were worth

exploiting. For one, the idea of training the model not only on trajectories generated with the environment reward but also on trajectories with an artificial reward is an essential component of both training processes. In contrast to HER where the reward usually signifies a single goal state that ought to be reached, our idea rather formulates the reward and challenge goal with the objective of maximizing the learning experience *on the way* to the mastery of the challenge. Although this necessitates to adjust our formulation from an on-policy to an off-policy learning strategy, it actually introduces more freedom in the way the trajectories are generated. In particular, we do not have to account for the initial state distribution problem explained in Appendix A.1.

3.2.1 Structure and challenge discovery behaviour

First of all, let us present the structure of our idea in detail. We stick with our original two-party formulation in the sense that we regard P and C as two agents. On the one hand, C is trained to accurately estimate the epistemic uncertainty of a transition. At the same time, C generates trajectories by alternating between its own exploration policy and P's policy network.

Afterwards, we fill two buffers with these trajectories. The first buffer comprises the complete trajectories with their *extrinsic* reward. We refer to this buffer as the *original* buffer D. The second buffer is dedicated to be filled with our artificial challenges and hence referred to as the *imaginary* buffer I. To that end, C recorded her uncertainty estimates for each transition. She now uses these estimates along with the extrinsic rewards to compute a challenge quality score of each state q. For each trajectory, she truncates the trajectory after the state $q^* := \arg\max_{q \in \tau} S(q)$ with the highest score. Afterwards, she assesses each truncated trajectory based on their scores and other statistics and includes only those trajectories into the buffer I that exhibit sufficient quality. For these sufficiently good trajectories, it replaces the extrinsic reward by a linear combination of an exploration reward, the extrinsic reward, and a distance reward that is supposed to increase for transitions that lead towards the end state of the truncated trajectory q^*. All of these three steps – the computation of the challenge quality score, the assessment of trajectories, and the computation of the new reward – are fundamental elements of our idea and crucial to the success of our algorithm. For this reason, we will thoroughly elaborate on their implementation later in Section 4.

These two buffers will then be used to train P in an off-policy fashion. Regarding the training algorithm of P, we will stick with the original proposal in the HER paper to use Soft Actor-Critic (SAC) (Lakshminarayanan et al. (2017)). The concept of *challenge* has now acquired a more indirect meaning. Instead of directly attempting the challenge, P is challenged to integrate the additional information from the imaginary buffer I into the update of its policy. If C filled I wisely, then the transitions in I will point toward the weak spots of P's policy and provoke him to improve in these aspects.

3.2.2 Measuring uncertainty

So far, we have not specified how C acquires good estimates of the uncertainty of a transition. As announced in Section 3.1.2, we avail to prior work and use the Dreamerv2 World Model (Hafner et al. (2019)) and its Plan2Explore extension (Sekar et al. (2020)). While the idea of long-horizon planning in learned latent representations of the high-dimensional observation space demonstrated impressive results in exploration-heavy environments with complicated tasks, Plan2Explore has been designed to build upon this strong world model to find regions of high novelty, which is exactly what we expect the exploration agent to do. Furthermore, the world model also proves useful for the computation of our artificial rewards, as we explain later in Section 4.4.3. Reusing these prior models enables us to focus on our core contributions, which already exhibit enough freedom to determine

success or total failure of our idea. Algorithm 1 summarizes our training process. More specific implementation details are subject of the following section.

Algorithm 1 CPM Dreamer: Training process

Input: Main Agent $A := (C, P)$ with world model WM, image encoder ENC,
original buffer D, imaginary buffer I, number of trajectories generated per episode T,
challenge quality score S, artificial reward formula R, challenge filter $F : \tau \mapsto \{0, 1\}$.

for all episodes **do**
  for T times **do**
    Generate one trajectory τ by exploiting P and exploring with C ε-greedily.
    For each state $s_t \in \tau$, C estimates its uncertainty as u_t.
    Store τ in D.
    for all transitions $(s_t, a_t, r_t, s_{t+1}) \in \tau$ **do**
      C estimates $u_t \leftarrow$ epistemic uncertainty in s_t according to WM.
      C embeds $e_t \leftarrow \text{ENC}(s_t)$.
      C evaluates the score $S(s_t, r_t, u_t)$.
    end for
    C obtains $g := \arg\max_t S(s_t, r_t, u_t)$ and truncates $\tau' := \tau_{[1:g]}$.
    if $F(\tau') = 0$ **then**
      continue with next trajectory.
    end if
    for all transitions $(s_t, a_t, r_t, s_{t+1}) \in \tau'$ **do**
      C calculates the distance improvement $d_t \leftarrow \|e_g - e_t\| - \|e_g - e_{t+1}\|$.
      C replaces r_t by new reward $r'_t := R(s_t, r_t, u_t, d_t)$.
    end for
    Store T' in imaginary buffer I
  end for
  Train WM, ENC and C off-policy with D.
  Train P off-policy with both D and I.
end for

Figure 2: Pseudocode for training our Competence Progress Motivation (CPM) Dreamer model. In our case, P is an archetypical SAC agent and the challenger C is the Dreamerv2 model with the Plan2Explore extension that already includes a world model and an image encoder.

4 Implementation Details

As already mentioned before, we realised P as a standard SAC model and availed to the SAC implementation in the ManiSkill2-Learn repository as reference.

Regarding the controller, it proved most effective to use the `pd_ee_delta_pose` controller, as we reason more thoroughly in Appendix B.3.

Because it provides the richest learning signal, we decided to receive dense and not sparse rewards.

As reported on GitHub, multi-threaded rendering issues prevented the parallel rollout of the agent's policy in the environment during training. For this reason, one single agent instance was executed in the environment during environment, which slowed down the training process.

4.1 Dreamer and Plan2Explore

On the other hand, we implemented the challenger C as the Dreamerv2 World Model with the Plan2Explore extension. Unfortunately, there were a few conflicts between these models. Firstly,

Dreamer expects RGB inputs while our implementation of SAC relies on a PointNet. For this reason, we adjusted the observation wrappers to receive both RGB and Point Cloud observations from the environment.

For the Point Cloud input, we sample additional 50 points near the goal observation if the goal state is visualized in the environment, such as is in `PickCube-v0`, `PickSingleYCB-v0`, and `PickSingleECAD-v0`. This was recommended by the authors and proved helpful in performance. The RGB images are captured from both the end-effector camera and the external camera, and were scaled to 32×32 pixels.

Secondly, Dreamer and Plan2Explore are both implemented in TensorFlow and hence incompatible with PyTorch's automatic differentiation package `autograd` that the reference implementation of SAC was built upon. For that reason, we decided to reimplement Dreamer and Plan2Explore in PyTorch. Although there was already a reference reimplementation of both modules at https://github.com/jsikyoon/dreamer-torch/tree/main, it turned out to contain several data-processing errors and unfinished code passages – especially in the Plan2Explore extension – that needed to be fixed in the first place.

Furthermore, we also decided to deviate from the reference implementation in a few places and adjusted the numerous hyperparameters to the needs of our tasks. This includes but is not limited to the depth and kernels of the CNN encoder and decoder, the hidden state of the GRU, the size of the latent state, the layer size and depth of the actor, value, and reward head, and their learning rates. We extensively elaborate on the structure of Dreamer, our deviations, and our considerations for fine-tuning in Appendix B.1. For similar insights in our implementation of Plan2Explore, where we most importantly propose a different metric to rate the uncertainty of a state, we refer the interested reader to Appendix B.2.

4.2 Challenge Exploration Decay

When generating trajectories for the replay buffers I and D, we started with a high initial challenge exploration probability ε and applied a decay that steadily enlarges the impact of P's policy network on the generated trajectories. The decay model should firstly ensure that ε remains high enough in the initial phase where P's policy network still exhibits much randomness. Secondly, it should have a gentle decay in the long-term to maintain a small amount of challenge exploration. We present the impacts of several decay models and the empirical advantages of our final choice in Appendix **??**.

4.3 HER and SAC

The SAC agent was trained on both the original and imaginary replay buffer. In each update step, one batch was randomly sampled from the original buffer with a probability of 75% and from the imaginary buffer with a probability of 25%. The agent was then trained with this batch and this random process was repeated six times. In this way, we could ensure a sufficiently accurate perception of the true rewards in the environment while still providing imaginary challenges enough impact.

Moreover, we chose the size of the original buffer to contain 50 full trajectories, while the imaginary buffer could only contain 10 trajectories. This is because the extrinsic rewards stay permanently valid, while the imaginary rewards builds on time-variant uncertainty and knowledge and their consistency decays with time.

4.4 Challenge Choice

In the following, we will explain how we implemented the challenge quality metric that rates each state in the generated trajectory. Secondly, we present how we filter trajectories and states to obtain our final challenge trajectories for the imaginary buffer I. Thirdly, we derive the artificial reward terms that replace the extrinsic rewards in these challenges.

4.4.1 Challenge Quality Metric

Promising regions for challenges should not only be novel to P, but also exhibit some relevance to the task. We hence decided to compute our challenge quality metric as a linear combination of the extrinsic reward r_t and the uncertainty estimate u_t as

$$S(s_t, r_t, u_t) = c \cdot r_t + u_t, \text{ where } c \cdot r_t \approx 0.75 \cdot u_t.$$

Furthermore, u_t is computed as a linear combination of the KL loss between the stochastic prior and posterior of Dreamer's latent state KL-LOSS and the disagreement within the predictor ensemble of the Plan2Explore extension DISAG.

$$u_t = c_1 \cdot \text{KL-LOSS} + \text{DISAG}, \text{ where } c_1 \cdot \text{KLLOSS} \approx 0.1 \cdot \text{DISAG}.$$

Because all values are subject to significant changes throughout the training process, the coefficients c and c_1 must be dynamically adjusted. Since this requirement appears useful for many hyperparameters and coefficients, we implemented a general mechanism to adjust hyperparameters if they deviate from a target ratio too strongly. This mechanism is throughly introduced in Appendix B.7.

4.4.2 Challenge Filtering with Self-Balancing Strictness

Not necessarily all trajectories serve as useful challenges for I. For that reason, a balanced filtering strategy $F : \tau \mapsto \{0, 1\}$ that discards trajectories with too feeble attributes is crucial to maintain a high challenge quality. Intuitively, F validates three criteria. If τ does not fulfil any of them, it is discarded. Firstly, the exploration reward u_t should not be negligibly small compared to r_t. Otherwise, the challenge is not novel enough. Secondly, the reward r_t should not be abysmal low to ensure that a challenge is also related to the actual task. Here, "low" must be regarded relative to the previous reward history of P. Thirdly, there should be some development on the way to the challenge goal g. By definition, g has the highest quality score on the entire trajectory. But if this improvement is relatively small compared to the average scores, then there will unlikely be any advantage of reaching g.

We thoroughly explain how we eventually realise these three criteria in a stable way in Appendix B.5.2.

4.4.3 Progress-Sensitive Artificial Rewards

Similar considerations as for the goal quality scores had to be made when defining how the reward term for artificial challenge trajectories is constituted.

Briefly, our reward term r'_t consists out of four terms. Firstly, the extrinsic reward r_t, secondly, the uncertainty term u_t, thirdly, the improvement in the *embedding distance* to the goal state $d_{emb,t}$, and finally, the improvement in the *step distance* to the goal state $d_{step,t}$.

The embedding distance improvement $d_{emb,t}$ is computed as

$$d_{emb,t} := \|e_g - e_t\|_1 - \|e_g - e_{t+1}\|_1,$$

where $\|\cdot\|_1$ denotes the Minkowski distance with $p = 1$.

Moreover, the step distance $d_{step,t}$ embedding is just a reward that increases proportionally towards the goal state to further lore the SAC agent to explore this state in future trajectories.

To preserve consistency with the extrinsic reward of the same transitions in the original buffer, we scale the other three terms u_t, $d_{emb,t}$, and $d_{step,t}$ to meet certain target ratios to the extrinsic reward r_t in average.

Specifically, we ensure that the extrinsic reward is approximately 1.5 times higher than the exploration reward u_t, while the mean of the distance terms is targeted to be 0.3 times smaller than u_t.

To meet the aforementioned target ratios, we again avail to our dynamic hyperparameter adjustment mechanism that we thoroughly introduce in Appendix B.7.

To remain in a similar range with the extrinsic rewards in the original buffer, the entire resulting linear combination is then scaled by 0.5.

We thoroughly explain this design decision of the artificial rewards in Appendix B.6.

4.5 Dynamic Hyperparameter Targeting

Throughout the learning progress of the world model and the agent, most values like rewards, uncertainty, or embeddings will fluctuate strongly. Static hyperparameters can not account for this strong deviations in magnitude. By dynamically adjusting the hyperparameters to always meet a certain target ratio, we can ensure their balanced interplay independent of environment-specific value ranges.

For this reason, we implemented a general dynamic hyperparameter adjustment mechanism that we applied to almost all of the aforementioned scales for the artificial reward terms, the challenge quality metric, or the uncertainty term.

In all of these cases, we formulate a target ratio r^* that the hyperparameter h over some comparison statistic g should satisfy. If this ratio deviated too strongly, that is $h < \omega_{low} \cdot r \cdot g$ or $h > \omega_{high} \cdot r \cdot g$ for some static tolerance bounds $\omega_{low}, \omega_{high}$, the parameter h was increased or decreased by a static scale $\lambda > 1$. Here, the tolerances and the adjustment factor were chosen depending how strictly we wanted to satisfy this target ratio and how high we expected its variance to be.

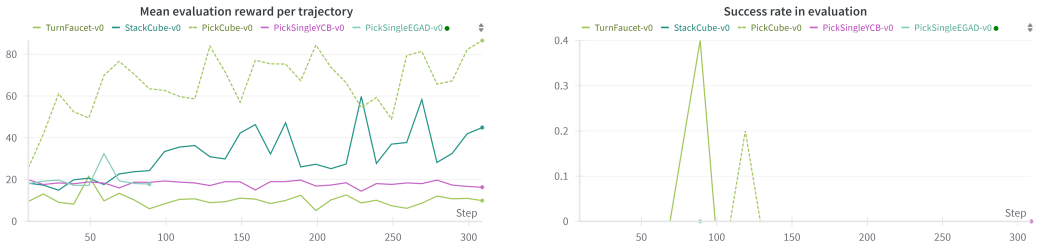
We thoroughly explain our implementation in Appendix B.7.

5 Experiments

5.1 Baselines

As requested, we implemented the baselines as seen in `examples/tutorials/reinforcement-learning/sb3_ppo_liftcube_rgbd.py`. We ran the code for all 5 environments: "PickCube-v0", "PickSingleEGAD-v0", "PickSingleYCB-v0", "StackCube-v0", "TurnFaucet-v0". Because of a rendering error in the cabinet mesh, we did not conduct experiments on "OpenCabinetDoor-v1". Although the baseline was able to achieve sound results in the `LiftCube-v0` environment during our mid term report, it fails in all of the five environments, which are more challenging. The authors reported that these baselines performed

better in an earlier version 0.3.0 of ManiSkill, but after update their rendering to 0.4.0, only the DAPG baseline seemed to preserve good performance according to their GitHub page.



(a) Mean evaluation rewards per trajectory.

(b) Evaluation success rate.

Figure 3: Only in the `PickCube-v0` environment, the PPO SB3 baseline succeeds seldom during evaluation. In all other environments, it never succeeds.

5.2 CPM Dreamer

To showcase the potential of our idea, we compare two handicapped versions of our model with a strong Dreamer model, a PPO that relies on PointCloud observations, and the PPO SB3 baseline that receives RGBD inputs on the `PickCube-v0` task. Despite these handicaps, our models achieve the most frequent success rates already during an early training stage. Although the PPO PointCloud model dominates all other models with regard to rewards, it does not succeed once over a far longer period. Although our handicapped CPM Dreamers already achieve some success, it occurs fairly seldom, which we account to the exaggerated exploration and weakness of the models respectively. The success and evaluation rates are displayed in Figure 4.

(a) Mean reward during evaluation.

(b) Average success during evaluation.

Figure 4: Five models are evaluated on the `PickCube-v0` task. Among them are two handicapped versions or our CPM Dreamer model. While the `CPM Dreamer: High Exploration` model has a far too small exploration decay and therefore does overly explore without letting the player P contribute to the trajectories and hence foster his task-specific knowledge, the `CPM Dreamer: Simple`

As the training procedures for stronger versions of our models unfortunately crashed at least after a few hours because the GPU memory was quite occupied in the last days, we can not demonstrate yet how large the advantage of our final, fine-tuned model over the baselines actually is.

But as these weak versions already exhibit an advantage in evaluation regarding the eventual success, we believe that our idea might contain at least some components such as the dynamic hyperparameter adjustment, the self-balancing challenge filtering and evaluation that prove useful in providing a good learning experience to our main SAC agent.

More experiments in the Appendix C underscore the ability of our model to learn reasonable latent representations of the environment and adjust to new ranges of time-variant hyperparameters.

6 Conclusion

However, there are still possible improvements of the model. More research could be conducted on more sophisticated distance terms in the goal evaluation reward, or the self-balancing policy in goal filtering.

References

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

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

M. Csikszentmihalyi. *Flow: The psychology of optimal experience*. HarperPerennial ModernClssics, New York [etc.], 1st harper perennial modern classics ed. edition, 2008, cop. 1990. ISBN 0060162538.

E. L. Deci and R. M. Ryan. *Intrinsic motivation and self-determination in human behavior*. Springer Science+Business Media, New York, 1985. ISBN 978-1-4899-2273-1. doi: 10.1007/978-1-4899-2271-7.

F. Gao, X. Li, J. Yu, and F. Shaung. A two-stage fine-tuning strategy for generalizable manipulation skill of embodied ai, 2023.

J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, X. Yuan, P. Xie, Z. Huang, R. Chen, and H. Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

J. Ho and S. Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016. URL http://arxiv.org/abs/1606.03476.

J. M. Hunt. Intrinsic motivation and its role in psychological development. *Nebraska Symposium on Motivation*, 13:189–282, 1965.

A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2007. doi: 10.3389/neuro.12.006.2007.

D. Pathak, D. Gandhi, and A. Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pages 5062–5071. PMLR, 2019.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010. ISSN 1943-0604. doi: 10.1109/TAMD.2010.2056368.

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

R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

A. Stout and A. G. Barto. Competence progress intrinsic motivation. In *2010 IEEE 9th International Conference on Development and Learning (ICDL 2010)*, pages 257–262, Piscataway, NJ, 2010. IEEE. ISBN 978-1-4244-6900-0. doi: 10.1109/DEVLRN.2010.5578835.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

A Architecture Details

A.1 Including Start States into the Challenge Description

Usually, a precise formulation of a challenge should also at least implicitly include a starting region or more formally a distribution of initial states. If we would challenge P to climb a hill along a difficult path but allow him to spawn in all regions around the hill, the content of the challenge might eventually strongly shift depending on where P spawns. Over and above, we can only guarantee that our reward function has been designed for the specific path we wanted to guide P over. Perhaps, it will even provide misleading signals if P approaches g from other paths. If P therefore commits to a challenge, we must ensure to adjust the initial state distribution to the current challenge specification.

Unfortunately, not all simulation environments allow this manipulation of the initial state, as was the case for ManiSkill2. This provided us another reason to formulate a restricted alternative for the scope of this project.

B Implementation Details

B.1 Dreamer

To begin with, we explain the Dreamer pipeline based on the parameters in the original implementation. Afterwards, we will discuss and reason our own adjustments.

The original Dreamer implementation received images of input size 64×64. To the end of planning in a latent space of the high-dimensional image observations, the fundamental objective of Dreamer is to obtain compact and rich low-dimensional representations of the observations that facilitate the understanding of the environment dynamics with minimal loss in information.

Firstly, a four-layer CNN processes the raw observation into an embedding vector of a size of $2^5 \cdot$ `depth`, where `depth=32` initially. From this more compact representation, a forward dynamics model obtains a hidden latent state, which comprises both a deterministic part and a stochastic part, where the deterministic part occupies 200 dimensions and the stochastic part 50 dimensions.

This forward dynamics model comprises a one layer GRU cell of hidden state size 200 and is wrapped by two feed-forward fully connected layers. In case of the prior latent state, these networks map the stochastic part of the previous latent state to the hidden state size and vice-versa. For the posterior stochastic latent state that is also conditioned on the embedding of the actual observation, the GRU is directly bypassed and replaced by another fully-connected feed-forward layer that maps the embedding and deterministic part of the prior latent state to the stochastic posterior of the latent state vector.

To ensure that this latent state vector preserves as much information from the environment as possible, the World Model architecture culminates in two heads, which predict the image observation and the perceived reward associated with the observation respectively. While the reward head is implemented as a two-layer MLP of hidden layer dimension 400, the head that predicts the raw image observation is a Decoder CNN that is supposed to complement the original dimension. The deconvolution layers are preceded by a fully-connected layer to again bridge the gap between different dimensionalities in the latent space.

Availing to this world model, one task-specific agent and one exploration agent are trained to optimize their value functions and actions inside the latent space learned by the world model. The exploration agent is an implementation of Plan2Explore, which we will focus on later. The task-specific agent first imagines a trajectory of length $T = 15$ inside the latent space, starting from an initial latent state representation provided by the Dreamer unit. It then uses the reward head from Dreamer to generate rewards for these latent transitions and update its value and policy network with them. The stability of these parameter updates strongly relies on a world model that already captured meaningful representations.

B.2 Plan2Explore

Because the purpose of Dreamer in our setting is to explore the environment and hopefully find unseen regions that yield enriching challenges and experiences to the main agent, the exploration agent – albeit small – is the most important final unit. Plan2Explore avails to the world model learned by dreamer to estimate the expected novelty of states that yet lie ahead multiple steps. Mathematically, it models uncertainty as the disagreement between a predictor ensemble that shall predict a certain target value from the current latent state and action. This target value can either be the next latent state, embedding, or more specific parts of the latent state. While the original design

choice favoured the embedding as target value, we first decided to target the entire next latent state. Another implementation focuses only on predicting the stochastic part of the next latent state, but this seemed incomplete to us since the transition dynamics for the deterministic latent state matter as well in the estimation of uncertainty.

The predictor heads were simple MLPs whose output was interpreted as a distribution. Uncertainty was consequently measured as the average standard deviation between the mode of this distribution across all dimensions of the target vector. This renders aleatoric uncertainty irrelevant to the final uncertainty measure since the predictors should in expectation all converge to the similar mean for states that innately exhibit variance.

For the following argument, it is worth reporting that the final average standard deviation term took values around 0.1 and slowly decreased to 0. In the original implementation, this average standard deviation was projected by the $\log$ function to obtain the final uncertainty scalar. Effectively, this implies that a state where the predictor ensemble achieves a disagreement measure of 0.001 receives a negative term with a far larger magnitude than a state where the disagreement is "only" about 0.01. However, both of these values are extremely low and represent states that the agent already sufficiently explored. Our metric does not need to be sensitive in this value region. More importantly, the metric should filter the most *unknown* states and be more sensitive between higher values close to 1.0. For this reason, we decided to replace the $\log$ term by an $\exp$ term and beforehand scaled the disagreement by a value that dynamically adjusts itself to the overall progress in uncertainty to remain in the optimal value range where $\exp$ provides the desired exponential increase for higher disagreement. Specifically, we consider a distribution of disagreement where the highest quartile has a value around 1.0 as good.

Because it is reasonable to expect large deviations in the disagreement value range throughout the training process, we decided to rescale the disagreement values with a scale factor that dynamically adjusts over time. To that end, we formulate a tolerance interval that the disagreement values should in average not exceed or undercut. Otherwise, the scale factor is multiplied or divided by a small factor of around 1.5. We considered both the mean and the max of the current batch of disagreement values and formulated these tolerance intervals over both statistics. This dynamic, loose adjustment is later introduced more generally, since we followed the same principle to control the coefficients in our reward term for artificial goals. To be fair, one could also formulate such a target value and tolerance intervals for the $\log$ metric and ensure that the best values still remain positive. However, deviations from this target value would incur far more strong oscillations in the exploration terms. In experiments, we still observed situations where almost all exploration terms were negative because the average scaled predictor disagreement slightly decreased below 1.0. This eventually discouraged exploration of the main agent since the uncertainty term was an essential component for the assessment of trajectories as possible imaginary goals.

We believe that the $\exp$ term provides a more stable behaviour for increasingly small disagreements as it maps them all too values slightly greater than 1.0 and is quite insensitive to small deviations in these small values. We could hence safely subtract 1.0 from the final $\exp$ term to lower the uncertainty metrics for these undesirable, sufficiently explored states down to values quite close to 0, while unseen states that achieved higher rewards still obtained values greater than 1.0.

B.3 Controllers

The default `joint_pd_delta_pos` controller tracks only the position of the robot's joints, but not its rotations. On the other hand, the `pd_ee_delta_pose` controller also includes the rotation matrix and

focuses on the end-effector frame. This proves useful for many of the pick-and-place tasks that we deal with in this report. For this reason, we have observed a tremendous performance improvement when switching from `joint_pd_delta_pos` to `pd_ee_delta_pose`. This was also reported by the authors. The `pd_ee_target_delta_pose` controller that applies the ideal transformation matrix that the current action would lead to does not achieve a similarly good performance and was hence not selected for our final model.

Unfortunately, the authors realised all controllers as PD controllers and not the more performant PID controller. Moreover, the stiffness and damping coefficients in the PD formula could not be adjusted. Since the PD controllers exhibited strongly different performance, it is likely that the coefficients in the PD controller error term could profit from more fine-tuning.

B.4 Decay Models for Challenge Exploration

B.5 Challenge Choice

B.5.1 Challenge Quality Metric

After having collected new trajectories through the interplay of our task-related main agent P and the exploration-oriented Dreamer C, we inspect the new trajectories to find state regions where we expect enriching experiences for P. Mainly two factors contribute to this *challenge quality* of a state s_t.

Firstly, the task-specific, extrinsic reward r_t is a good indicator of how useful a certain state might prove for eventually mastering the environment task. Secondly, the epistemic uncertainty measured by Dreamer's Plan2Explore extension signifies novel regions that could aid P to abstract from its limited experience and eventually push forward to unseen regions in the optimization landscape. At the same time, both terms should not dominate the quality score of a state too much.

On the one hand, the agent might lose sight of the actual task if we considered only the epistemic uncertainty. Plan2Explore was originally intended to obtain a task-agnostic, broad distribution of states, which is only of limited benefit to our agent since some of them will yield little benefit for our current task. This trade-off between exploration and goal orientation reminds of the RRT planning algorithm, where we sample the states not purely uniform, but bias towards sampling the goal state with a certain probability to ensure that the final tree is expanding towards the task-specific goal state. On the other hand, too little exploration might render the Dreamer agent superfluous and narrow down the behaviour of the agent to its selective experience, which reflects the usual combination of exploration and exploitation in the earliest reinforcement learning algorithms.

For this reason, we decided to target a ratio of the extrinsic reward over the exploration reward of $c^* = 0.75$ to slightly favour exploration and hence target

$$S(s_t, r_t, u_t) = c \cdot r_t + u_t, \text{ where } c \cdot r_t \approx c^* \cdot u_t, \quad c^* = 0.75.$$

We do not need another coefficient to scale the entire term because we only compare these terms *relatively* to each other. Since the Dreamer agent already computes the KL divergence between the prior and posterior belief states conditioned on the embedding of the current image, we included this value with a ratio of $c_1^* = 0.1$ to the ensemble predictor disagreement into the uncertainty term.

$$u_t = c_1 \cdot \text{KL-Loss} + \text{Disag}, \text{ where } c_1 \cdot \text{KLLoss} \approx c_1^* \cdot \text{Disag}, \quad c_1^* = 0.1.$$

Because the latent state was also split into a deterministic part and stochastic distribution, we felt that it would have the same ability as the predictor disagreement DISAG to mitigate the influence of aleatoric uncertainty by steadily converging to the same mode in the distribution that minimizes the mean squared error. But we still remained conservative and chose the impact of this KL term to be 10 times smaller than the disagreement. Just as mentioned before when discussing the scale of the disagreement value, we ensured that the coefficient of the KL value and the coefficient of the extrinsic reward remained in a tolerance interval around the aforementioned target values. Again, we will discuss our implementation of such a dynamic scale adjustment in more detail in Appendix B.7.

B.5.2 Challenge Filtering with Self-Balancing Strictness

After each state was assigned a goal quality score S, we next had to pick goal states out of the batch of trajectories whose metric seemed promising to consider this state as an artificial goal. To avoid possible error sources, we refrained from overcomplicating the selection strategy at first and defined the state $g := \arg\max_t S(s_t, r_t, u_t)$ with the highest goal quality metric in each trajectory as a *challenge candidate*.

We then ensured that these candidates met five criteria $F_i : \tau \mapsto \{0, 1\}, \quad i = 1, \dots 5$ that related either to the sole candidate state or the entire trajectory from which the candidate stemmed. While the last two criteria were hard criteria that each challenge must meet, the challenge candidate must only satisfy the *majority* of the first three ones.

To validate the criteria for the entire trajectory, we computed its the mean exploration reward $\overline{u}$ and the mean extrinsic reward $\overline{r}$.

Firstly, the exploration benefit of the entire trajectory should be sufficiently high, else the goal state was mainly selected because of its high extrinsic reward. In such a case, we believe that the trajectory will already enrich the agent in the original replay buffer. If $\overline{u}$ is negligibly small compared to $\overline{r}$, we hence rejected the goal candidate. Here, negligible refers to falling short of $\overline{r}$ by a tolerance factor ω_1 that was initialized with 2.0. To reasonably compare the magnitudes of $\overline{r}$ and $\overline{u}$, we avail to the fact that after some iterations, the coefficient c in the goal quality term S of the last section B.5.1 will have been sufficiently adjusted to correctly track the target ratio c^* between r_t and u_t. Expressed formally, it should hold that $c \cdot r_t \approx c^* \cdot u_t$, therefore we have

$$F_1(\tau, g) := \begin{cases} 0, & \omega_1 \cdot c \cdot \overline{r} > c^* \cdot \overline{u} \\ 1, & \omega_1 \cdot c \cdot \overline{r} \leq c^* \cdot \overline{u} \end{cases}.$$

However, this initial value $\omega_1 = 2.0$ is heuristically set as many other hyperparameters are and experimental evaluation proved that this value mostly achieved a good balance, as a static value it could not account for the inevitable fluctuations in $\overline{r}$ and $\overline{u}$. We hence implemented a mechanism that we call *self-balancing strictness*. The intuition behind this mechanism is the observation that humans also adjust their strictness in judgment contextually. We don't blame a small child for not walking stably and crawling over the ground. But as it grows, so grow our expectations on which we base our judgment. Another more related example are relative grading principles which are practice in some schools or universities to relate the performance judgment to the performance of an entire population. In our case, the increasing certainty about the environment and the increasing reward can similarly be regarded as such a developmental progress to which we want to adjust our judgment.

Even in the static case where the assessed population does not experience any change in performance, such a dynamic judgment adjustment also proves useful for another problem. Namely, we want to avoid that we reject all goals, which will effectively leave the replay buffer of imaginary goals empty. On the other side, accepting each goal candidate will not filter good trajectories from worse trajectories, of which there certainly will be many.

We hence employ a separate tolerance ω_i for each filter F_i. Every time a goal candidate g was rejected because $F_i(\tau, g) = 0$, its tolerance value ω_i is adjusted by a slight factor $\gamma_F \approx 1$ such that F_i becomes a slightly more loose criterion in the future. The opposite adjustment is taken when a goal was accepted by this criteria. Because this causes an exponential growth or decay for extreme initial biases towards rejection or acceptance, the value will quickly stabilise itself and moreover dynamically adjust to the quality of the candidate states.

Symmetrically, one could check whether $\overline{r}$ is negligibly small compared to $\overline{u}$. However, we refrain from rejecting candidate states based on this criterion. The reason is that the buffer for the imaginary goals is mainly dedicated to fostering exploration and should not punish states because of their novelty. However, we already stressed our concerns for diverging too exhaustively into task-irrelevant areas and hence formulated another criteria related to the above.

Namely, we ensure that the extrinsic reward of a goal trajectory is not too low in comparison to the total average reward previously achieved by the agent in a certain time window.

This first requires measuring this total average reward. We decided to extend our agent by a scalar variable that we call the *mean reward memory* r_{mem}. Every time the agent was called with new trajectories, the agent computed the mean reward over all these trajectories as r_{new}. Then we updated the reward memory in a bilinear fashion as

$$r_{mem} = \gamma_r \cdot r_{mem} + (1 - \gamma) \cdot r_{new}.$$

Here, γ_r is a reward memory decay, which we set to 0.99. This means that rewards that were perceived 100 steps ago become roughly three times less important than the current rewards. We were worried that smaller values might entail a too short-sighted memory that is sensitive to short-term oscillations in the perceived reward and oriented ourselves on the discount factor γ that was also set to 0.99.

With this mean reward memory in hand, we formulated the second criteria in an analogous fashion to the first criteria, but this time compared the average reward of the entire trajectory with the mean reward in the memory.

$$F_2(\tau, g) := \begin{cases} 0, & \omega_2 \cdot \overline{r} < r_{mem} \\ 1, & \omega_2 \cdot \overline{r} \geq r_{mem} \end{cases}.$$

We ensured that F_2 tolerates short-term deviations in the overall reward of an agent which we expected to occur by nature of the commonly unstable optimization procedures for hard optimization landscapes. After experimental evaluations, we culminated in a bound with a comparably larger value $\omega_2 = 8.0$ than ω_1.

From now on, we focus on the challenge quality score of the candidate goal state g. Our third criterion ensures that there should be some development on the way to the challenge goal g. By definition, g has the highest quality score on the entire trajectory. But if this improvement is relatively small compared to the average scores on τ, then there will unlikely be any advantage of reaching g and lifting g to a "goal" state is indeed questionable. We hence verified whether the goal quality score of the candidate state was at least $\omega_3 := 2.0$ times higher than the average goal quality score.

Of course, this value was dynamically adjusted in the same fashion as the prior values. In the experiments, we observed that this criteria turned out to be one of the strictest, since many trajectories did not include any "highlight". We later experimentally evaluate the influence of these filters in Appendix C.2.2 However, the majority of proposed candidates finally passed all these three criteria.

Finally, the two last criteria seal some edge cases we encountered during debugging. The fourth criterion F_4 rejects candidates with negative scores, which happened in early tests where the $\log$ implementation of the predictor ensemble disagreement was not yet replaced by our $\exp$ metric. With the $\exp$ metric at hand, negative rewards should be impossible, but we left it as an assertion.

The fifth criterion F_5 rejected candidates that were the very first states in a trajectory. Because we decided to truncate trajectories after their goal state, this resulted in erratic behaviour, which we avoided this way.

We further have to note that we removed the first five steps of each trajectory, because the initial prior latent state of each trajectory consisted only of 0s. Because the initial state in the environment is mostly random, it is impossible to correctly predict the next latent state even if the same action is given. Therefore, the predictor ensembles always had their highest disagreement in the first state of each trajectory, which is a correct behaviour, but undesired for our case. After three steps, the predictions were fully grounded on the actual observations and hence pertinent for our challenge discovery.

If a candidate (τ, g) passed all five checks, the trajectory τ was truncated after this state. To integrate them as imaginary goal experiences into the buffer, we yet have to define the artificial reward term, which we discuss in the following section.

B.6 Progress-sensitive Artificial Rewards

Similar considerations as for the challenge quality metric in Appendix B.5.1 had to be made when defining how the reward term for artificial goals should be constituted.

If we neglect the dense extrinsic reward term r_t too strongly, then the reward experience in the imaginary trajectories will directly disagree with the reward signal in the original trajectories and hence unnecessarily disturb the agent. On the contrary, we want to push the agent to explore unseen regions, which again requires a significant weight of the exploration term u_t already used above for the goal quality score.

Moreover, both of these terms do not actively raise the reward towards the specific imaginary goal in the trajectory. To enlarge the emphasis on the goal state g to eventually curl the agent to the goal region, we included two more terms that reflect a *distance metric* to g.

We refer to the first term as the *embedding distance* d_{emb}, which is literally the *improvement* in the embedding distance to the goal embedding, where we make use of Dreamer's image encoder. Because the trajectory is a result of exploration, we cannot expect it to lead straight to the goal. In the lecture on offline RL, Professor Xu introduced a similar problem, where the agent has to learn to stitch good local trajectories together to efficiently approach the goal state without unnecessary detours.

By this assumption, we would punish detours in the trajectory and encourage the agent to directly approach the goal. However, it is foolish to directly take the embedding, because the initial states that are far away but directly lead to the goal still have a large distance while actions that detour from the goal state will still receive good scores if the state they originate from is close to the goal

state. What matters more is the *direction* in which the agent proceeds. For this reason, the *change* in the embedding distance seems a more useful metric to assess how an action roughly contributes to reaching the goal. In this way, actions that moved away from the goal – or at least from its embedding –, received a slightly negative reward. Therefore, the embedding distance d_{emb} is computed as

$$d_{emb,t} := \|e_g - e_t\|_1 - \|e_g - e_{t+1}\|_1,$$

where $\|\cdot\|_1$ denotes the Minkowski distance with $p = 1$.

Finally, we also included the simple step distance d_{step} to the goal as last heuristic for our artificial reward term. When s_t is the goal state in a trajectory, then reaching s_i will yield a reward proportional to i.

$$d_{step,i} := \frac{i}{t}, \quad \text{where } t \text{ is the index of the goal state.}$$

Our intention behind this reward was to complement potential inaccuracies in the embedding distance d_{emb}, which could provide brittle signals especially at the beginning where the encoder network of Dreamer still has not acquired a good embedding.

To mitigate the dissonance between the imaginary rewards and the actual rewards, we targeted the extrinsic reward term to be slightly more important than the exploration reward term, namely by a factor of 1.5. This is converse to their relation in the goal quality score where the exploration term was slightly more important, where it is not necessary to care for accordance with the extrinsic rewards.

In this way, we could still lay an emphasis on trajectories that promise a high exploration reward in the goal evaluation stage without evoking a severe dissonance in the experienced reward that the main agent receives to learn. However, the reoccurrence of this trajectory in the imaginary goal buffer will hone the agent's knowledge of this possibly unfamiliar area.

On the other hand, the two distance terms were both targeted to be approximately 0.3 times less important than the exploration term. This way, the sum of purely fictitious rewards and the extrinsic reward coarsely balance each other out and still leave a lot of freedom for the artificial reward component to lore the agent into regions that the extrinsic reward might not highlight.

Finally, the scaling so far was relative to the exploration reward and does not ensure that the magnitude of rewards are approximately equal to the true original rewards on this trajectory. Practically, we reformulated the scales to be relative to the *extrinsic* reward term. Because of this change, the artifical reward term is approximately two times higher than the true extrinsic reward term if all scales meet their aforementioned target ratios. To slightly emphasize trajectories in the imaginary buffer, we only scaled the entire reward term by 0.5, leaving a slight possible advantage that fully depends on the exploration and distance terms.

Hence, the final artificial reward term writes as

$$\begin{aligned} r'_t :&= 0.5 \cdot (r_t + c_{expl} \cdot u_t + c_{emb} \cdot d_{emb,t} + c_{step} \cdot d_{step,t}), \quad \text{such that} \\ \overline{r} &\approx c^*_{expl} \cdot c_{expl} \cdot \overline{u}, \quad c^*_{expl} = 1.5, \\ c^*_{emb} \cdot \overline{u} &\approx c_{emb} \cdot \overline{d}_{emb}, \quad c^*_{emb} = 0.3, \\ c^*_{step} \cdot \overline{u} &\approx c_{step} \cdot \overline{d}_{step}, \quad c^*_{step} = 0.3, \end{aligned}$$

where $\overline{x}$ again represents the mean of x over the entire trajectory.

B.7 Dynamic Hyperparameter Targeting

In the last paragraphs, we have often mentioned that we dynamically adjust hyperparameters to meet a certain ratio between the scaled value and another comparison value. In view of the highly dynamic learning progress, we are concerned that static hyperparameters will cause scales to drift off if new achievements or novel challenges affect the agents learning experience.

To that end, we implemented a mechanism to dynamically adjust the scale coefficient c of a hyperparameter h to meet a certain target ratio r^* with regard to another comparison value g up to a fixed tolerance ω. Although the target ratio r^* of hyperparameters is also another static hyperparameter, it resides on a more abstract, coarse layer. Fixing these strategic hyperparameters to a fixed value is less problematic than fixing more concrete, operational hyperparameters to a specific value. For each hyperparameter h that was subject to this dynamic adjustment, we defined an adjustment factor λ, which was usually chosen to be between 1.1 and 1.5 depending on how large we expected or desired h or its ratio to g to fluctuate.

In each update step of the model, we also check if any dynamic hyperparameter h violated their respective tolerances and adjusted them by λ if necessary. To estimate how well the current h satisfies the ratio r^* in a certain time window, we had to obtain a statistic that provided contextually relevant information on the distribution of this value. In most cases, the average value over trajectories or batches of trajectories was taken as $\overline{h}$. In the optimal case, $\overline{h} = r^* \cdot \overline{g}$, that is h should in average meet the static target ratio to the other value g. We again stress that g is non-static but also time-variant. If this target was missed too strongly, h was updated by λ. This allowed the scale to quickly respond to severe violations of the target ratio with exponential growth or decay. In this way, we could ensure that some values are kept in a certain range and neither explode nor diminish. In the experiments in Appendix **??**, we demonstrate that this mechanism was indispensable to tackle strong natural fluctuations of external values such as reward, uncertainty, or embedding distance. The following Algorithm 2 specifies this dynamic hyperparameter adjustment.

Algorithm 2 CPM Dreamer: Training process

Input: Hyperparameter statistic $\overline{h}$, adjustment $\lambda > 1.0$, comparison value statistic $\overline{g}$, target ratio r^*, lower tolerance ω_{low}, upper tolerance ω_{high}.

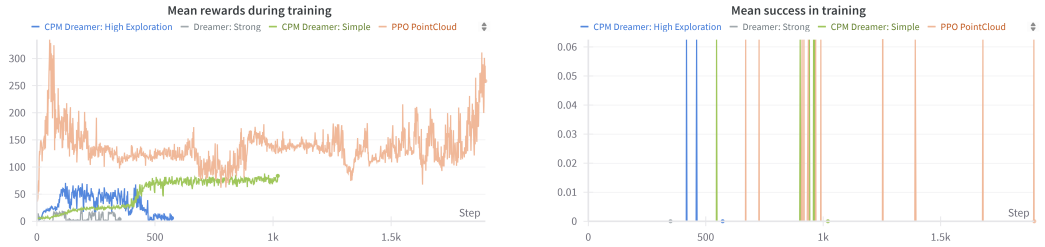
1: **if** $\overline{h} < \omega_{low} \cdot r^* \cdot \overline{g}$ **then**
2: $\quad \overline{h} = \overline{h} * \lambda$
3: **end if**
4: **if** $\overline{h} > \omega_{high} \cdot r^* \cdot \overline{g}$ **then**
5: $\quad \overline{h} = \overline{h} / \lambda$
6: **end if**

C Experiments

The following experiments were done on the `PickCube-v0` environment. They either compare handicapped versions of our CPM (Competence Progress Motivation) Dreamer with a strong, self-standing Dreamer model and PPO models, or experimentally verify the capabilities of our CPM Dreamer.

C.1 Challenge discovery

Figure 5 compares the experienced reward and success during training. We stress that in this early stage, the exploration rate of our CPM Dreamer models are still high and are plotted in Figure 8a. For this reason, the generated trajectories are mostly oriented at exploring unseen states with the Plan2Explore extension and not oriented towards succeeding in the task. Despite that, our models still achieves seldom success. While the PPO PointCloud model achieves far higher rewards and also slightly more frequent successes, it does not achieve any success during evaluation, while our model also sometimes achieves success there (Fig. 4b).



(a) Mean reward during challenge discovery

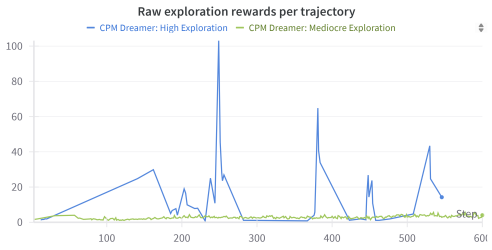
(b) Average success during challenge discovery.

Figure 5: During training, the agent $A = (C, P)$ uses both the Plan2Explore extension of Dreamer C and the SAC agent P to act ε-greedily, where ε is initialized with a high value and decays over time. The resulting behaviour is therefore mainly oriented at exploring unseen states and *not* at trying to master the task. Despite of that, A sometimes even succeeds in the task. During training, the PPO PointCloud model achieves higher success rates and rewards than our CPM Dreamer models, the PPO PointCloud does not achieve any success during evaluation (Fig. 4).

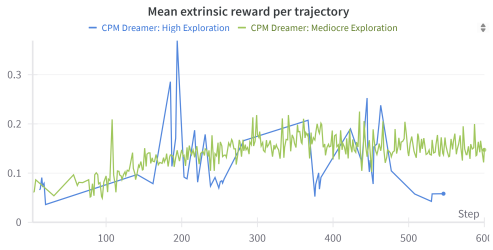
C.2 Challenge choice

C.2.1 Challenge quality metric

The extrinsic reward and the uncertainty that both constitute the challenge quality metric are subject to high fluctuation (Figures 6a, 6b). Accordingly, the coefficient for the extrinsic reward quickly adapts to these deviations and in this way manages to balance the influence of both terms to the final challenge quality metric S (Fig. 6c).



(a) Exploration reward u per trajectory.



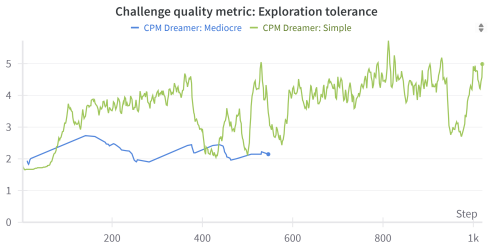
(b) Mean extrinsic rewards per trajectory.

(c) Extrinsic reward coefficient in challenge quality metric.

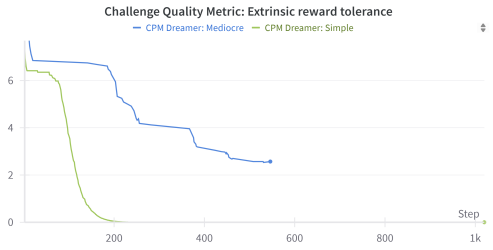
Figure 6: Overview of the exploration and extrinsic rewards and the coefficient that dynamically adjusts to their fluctuating magnitude to satisfy the target ratio.

C.2.2 Challenge filters

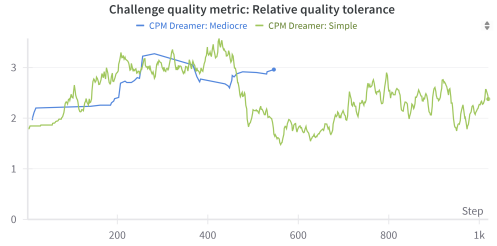
As Figure 7 illustrates for all three tolerances that they dynamically adjust their strictness to the amount of accepted challenge trajectory.



(a) Exploration tolerance.



(b) Extrinsic tolerance.

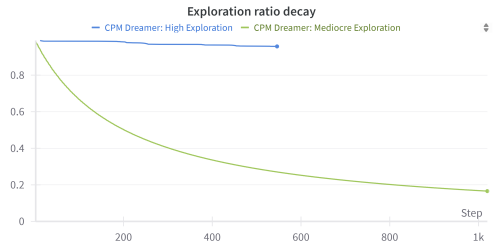


(c) Relative quality tolerance.

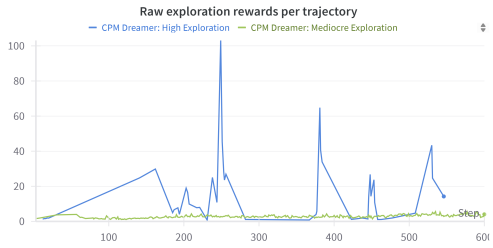
Figure 7: Our self-balancing strictness in challenge filtering dynamically adjusts to the prior acceptance and rejection behaviour.

C.3 Artificial rewards

Similar to Figure 6, this experiment again underscores the ability of our dynamic hyperparameter adjustment mechanism to account for strong fluctuations in time-variant values. In this case, the exploration reward experiences severe peaks (Fig. 8b). The coefficients for the extrinsic reward (Fig. 8c) and the step distance (Fig. 8d) both mimic this increase to similarly amplify the influence of their terms, but also quickly decrease after the exploration rewards go down again. This demonstrates that our reward term remains robust against different value scales.

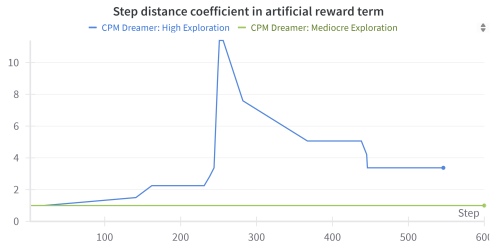


(a) Exploration decay of our CPM dreamer models.



(b) Uncertainty rewards per trajectory.

(c) Extrinsic reward coefficient.



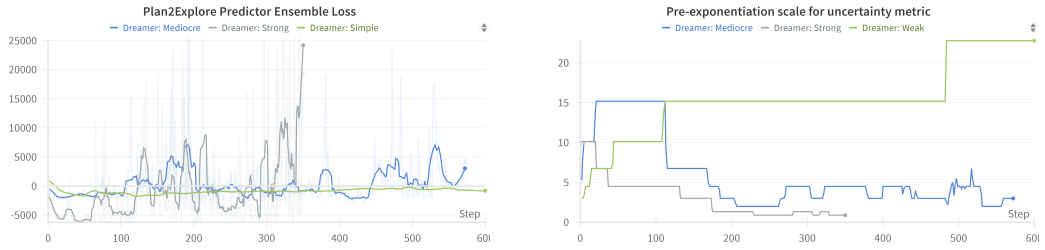
(d) Distance embedding coefficient.

Figure 8: Overview of our dynamic coefficients development in view of a highly fluctuating uncertainty during challenge discovery. Owing to the dynamic adjustment, the other terms can adjust to the fluctuating level of the exploration reward term (uncertainty). We showcase these adjustments for two models with highly different exploration decays.

C.4 Dreamer

C.4.1 Plan2Explore

The following Figure 9 is supposed to visualize how the pre-exponentiation scale (Fig. 9b) adjusts itself to the current range of the predictor ensemble loss (Fig. 9a). This scale hence adjusts the range to ensure that the most unfamiliar states are significantly highlighted in comparison to more known ones. As expected, the strong Dreamer has the largest prediction capabilities and hence the lowest possible loss. The pre-exponentiation scales and hence lowered to sensitize the filter. Because we applied a loose upper threshold, this scale remains stably low even when the predictor ensemble face a new unknown region which severely aggravates their loss.



(a) Predictor Ensemble Loss

(b) Pre exponentiation scale

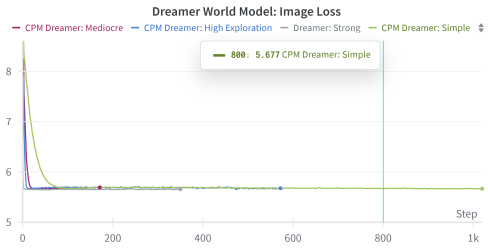
Figure 9: Juxtaposition of Plan2Explore's predictor ensemble loss with the associated adjustment of the dynamic scale coefficient before the uncertainty terms u_t are fed into $\exp$.

C.4.2 Model Loss

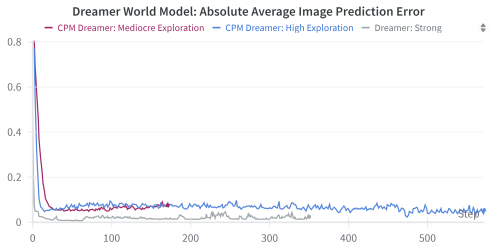
The Dreamer model is indeed capable of learning meaningful latent representations of the observations, as the following experiment results summarized in Figure 10 demonstrate.

On the one hand, the image head loss converges at roughly 5.7 and achieves a low absolute average pixel prediction error of less than 0.1. It is important to note that the pixel values have been normalized to $[-0.5, 0.5]$. Interestingly, the loss for the simple Dreamer and the strong Dreamer is similar, although the strong Dreamer achieves far better reconstruction results (Fig. 10b). We can not explain these results, as we would have expected the loss to be more lower for the strong model.

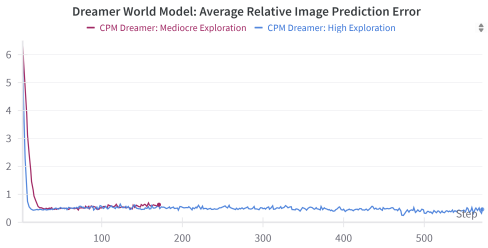
Similarly, the reward model reduces its absolute and relative prediction error (Fig. 10f) proportional to the reward loss (Fig. 10e). To much exploration seems to prohibit the model from truly grasping the nature of the rewards, as it explores regions that are too unrelated to the task.



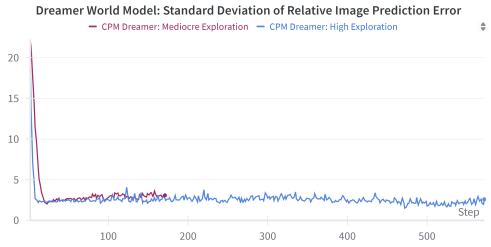
(a) Image head loss



(b) Average absolute image prediction error

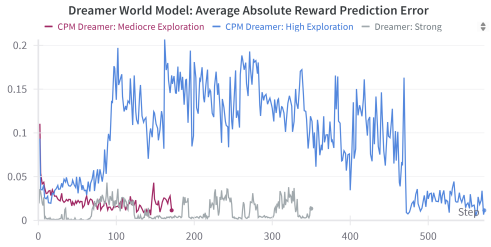


(c) Average relative image prediction error



(d) Standard deviation of relative image prediction error

(e) Reward head loss

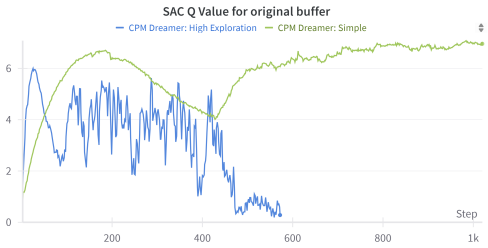


(f) Average absolute reward prediction error

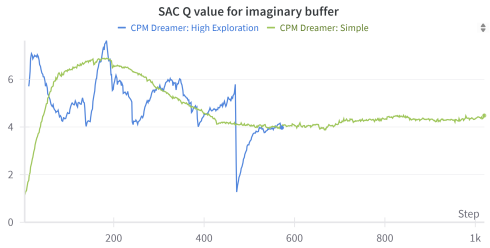
Figure 10: Dreamer's model loss reasonably coincides with accurate prediction of both the image observation and the reward. The image decoder and reward head are trained to predict the image observation and the perceived reward from the hidden latent state of the world model.

C.5 SAC Agent

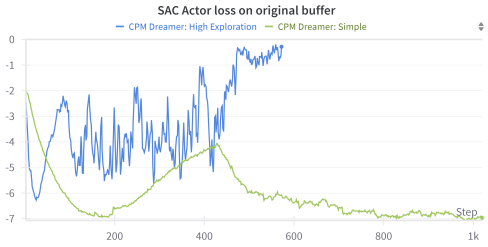
Finally, we juxtapose the loss and Q values of the SAC agent for both the original and the imaginary reward in Figure 11. Owing to the similar scale of the new reward term, the Q values and loss values for both buffers are roughly in the same range for both buffers and do not exhibit large dissonance.



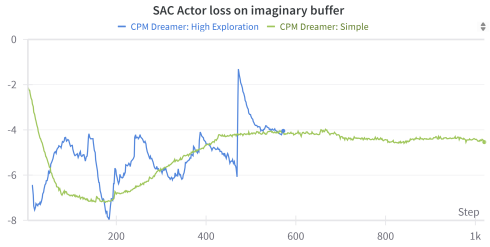
(a) Q Value for D



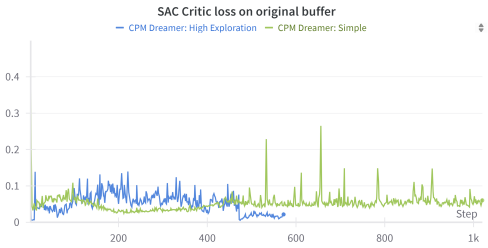
(b) Q Value for I



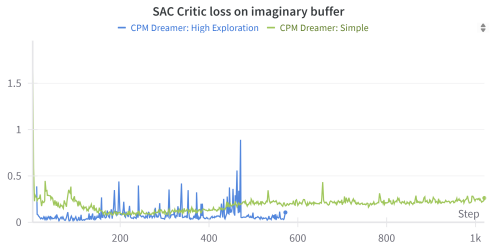
(c) Actor Loss for D



(d) Actor Loss for I



(e) Actor Loss for D



(f) Actor Loss for I

Figure 11: Juxtaposition of the SAC agent's Q values, actor, and critic loss for both the original buffer D and the imaginary buffer I.