World Model Agents with Change-based Intrinsic Motivation

Bachelor's Project Thesis

Jeremias Lino Ferrao, s4626451, j.l.ferrao@student.rug.nl,
Supervisor: Rafael Fernandes Cunha

Abstract: Sparse reward environments present significant challenges in reinforcement learning due to the infrequency of feedback, making it difficult for agents to learn effective policies. This thesis explores the performance of the DreamerV3 agent, enhanced with CBET, in comparison to the IMPALA agent within such environments. By leveraging a world model to internalize environment dynamics, DreamerV3 achieves superior sample efficiency and faster convergence. The CBET variant of DreamerV3 further improves performance by incorporating intrinsic motivation to guide exploration, proving especially beneficial in the challenging minigrid and crafter environments. Despite a brief anomaly in the minigrid transfer experiment where IMPALA outperforms DreamerV3, the overall results demonstrate the efficacy of DreamerV3 and its CBET variant in optimizing extrinsic returns and accelerating learning processes in sparse reward settings. Future research should address the limitations observed in policy transfer methods to enhance the robustness and generalizability of these findings. Additionally, exploring the interpretability of reinforcement learning algorithms is crucial to understand the decision-making processes of agents and the impact of intrinsic rewards. Reducing the computational resources required for DreamerV3 without compromising performance will also be a key focus, aiming to make these advanced techniques more accessible and practical for a wider range of applications.

1 Introduction

In Reinforcement Learning (RL), an agent's ability to efficiently discover and optimize reward-generating behaviors is crucial. Yet, conventional RL algorithms often struggle in sparse reward environments (Sutton & Barto, 2018). The lack of frequent feedback in these settings complicates learning, leading to extended exploration phases and slow convergence towards optimal solutions. This challenge has become a significant focus in recent years, driving the development of increasingly sophisticated exploration strategies.

Various methods have been proposed to address this issue, including visitation counts (Bellemare et al., 2016), curiosity empowerment (Pathak et al., 2017), differences in state representation (Raileanu & Rocktäschel, 2020), and view-based exploration (Guo et al., 2022). These approaches center on using intrinsic rewards to encourage exploration alongside the environment's extrinsic rewards. However, a key limitation lies in their primarily 'agent-centric' focus. They prioritize exploration based on the agent's individual history and beliefs, potentially missing out on universally interesting or actionable elements within an environment.

This contrasts with human exploration, where we recognize an inherent 'environment-centric' component – some objects or interactions are simply more likely to elicit curiosity regardless of our individual experiences. The algorithms mentioned above may struggle to transfer this kind of generalized exploration knowledge across similar environments, necessitating additional learning in each new setting. This impracticality highlights a key area for improvement in real-world scenarios.

Transfer learning offers a promising solution to the retraining problem. By reusing skills or knowledge acquired previously, an agent can potentially accelerate the exploration process in a new but related setting. This allows agents to exploit com-

monalities between environments, minimizing the need for exhaustive exploration from scratch (tabula rasa). C-BET (Changed Based Exploration Transfer) exemplifies this paradigm shift in exploration (Parisi et al., 2021), leveraging knowledge gained in prior environments to guide exploration more effectively.

In C-BET, a pre-trained model first explores an environment to identify interesting interactions. This pre-trained model then guides a task-specific model to optimize extrinsic rewards by incentivizing those interesting actions. The authors demonstrated encouraging results across various sparse reward environments, highlighting transfer learning's potential in RL. However, C-BET was evaluated solely with the IMPALA RL algorithm (Espeholt et al., 2018). Since its publication, significant advancements have been made in RL, resulting in more sophisticated algorithms (Hessel et al., 2022; Kapturowski et al., 2022). It is unclear how C-BET would perform with these newer techniques.

DreamerV3 (Hafner et al., 2023), in particular, is of great interest in the field of RL due to its feat of being the first algorithm to craft a diamond in Minecraft, a notoriously challenging sparse reward environment. This approach makes use of an Actor-Critic setup which operates on state representations obtained from a learned world model. The world model is trained to predict future states and rewards, enabling the agent to plan ahead and make informed decisions. We hypothesize that C-BET's exploration mechanisms could be integrated with DreamerV3, leading to superior performance in sparse reward environments.

Therefore, we aim to investigate whether combining C-BET's exploration mechanisms with DreamerV3 yields superior cumulative reward and faster convergence compared to a baseline of C-BET and IMPALA in sparse reward environments. We will evaluate the performance of these algorithms across a suite of sparse reward environments, including Minigrid worlds (Chevalier-Boisvert et al., 2023) and Crafter (Hafner, 2021).

2 Background

This work builds upon several concepts and techniques discussed in the literature. We provide an overview of these topics to establish a foundation for our research.

2.1 Counts and Pseudocounts

One of the fundamental techniques for intrinsic reward-based exploration is the use of visitation counts. A state's visitation count increases each time the agent enters that state, offering a metric of exploration frequency. The intrinsic reward is then designed to be inversely proportional to this count, incentivizing exploration of unvisited or less-frequented states. As outlined by Bellemare et al. (2016), the intrinsic reward can be utilized in a bonus reward:

$$R_n^+(s,a) = \beta(N_n(s) + 0.01)^{-1/2} \tag{2.1}$$

where $R_n^+(s,a)$ is the bonus reward for state-action pair (s,a), $N_n(s)$ is the visitation count for state s at time step n, and β is a scaling factor.

A key limitation of regular counts is their reliance on an exact match of the state representation. In environments with vast state spaces (e.g., millions of pixels), even minor variations in a few pixels would cause the count-based system to treat these as entirely different states. This hinders exploration, as the agent would perpetually perceive the environment as novel even if the changes are insignificant.

To mitigate this issue, Bellemare et al. (2016) introduced pseudocounts, generalizing visitation counts to large state spaces. Their approach employed density models to estimate counts using uncertainty, but this method carries high computational costs and implementation complexities.

Tang et al. (2017) proposed an alternative solution by using a hashing-based method for approximating visitation counts in complex state spaces. This technique transforms high-dimensional data into discrete hash codes for count estimation. Specifically, a multidimensional hash code $\phi(s)$ is generated for each state s:

$$\phi(s) = sgn(Ag(s)) \in \{-1, 1\}^k \tag{2.2}$$

where A is a $k \times D$ matrix of random values drawn from the standard Gaussian distribution, D is the number of dimensions in the pre-processed state, $g(s)$ is a pre-processing function for state s, and k is the hash code's dimensionality. Increasing k

improves the algorithm's ability to differentiate between states that are proximal in the original state space.

Upon obtaining the hash code, a state's visitation count is obtained by the frequency with which its hash code has previously appeared. Unlike simple counts, this method ensures similar visitation counts for closely related states, encouraging broader exploration of the state space.

2.2 C-BET

Parisi et al. (2021) introduce the Change Based Exploration Transfer technique to extend the foundation of visitation counts by incorporating a metric of 'interestingness' to drive the intrinsic reward calculation. Interestingness, in this context, quantifies the number of times a change has been detected in the environment. This change is typically calculated as the difference between the agent's current and previous state representations. The intrinsic reward is then formulated as:

$$r_i(s, a, s') = 1/(N(s') + N(c)) \tag{2.3}$$

where $r_i(s, a, s')$ denotes the intrinsic reward for transitioning to state s' after taking action a in state s, $N(s')$ is the visitation count for state s', and $N(c)$ represents the visitation count for the detected change c.

C-BET further proposes a novel transfer learning evaluation paradigm for RL algorithms. This process involves two phases:

- Pre-training: An 'intrinsic' agent is trained in an exploratory environment, guided solely by intrinsic rewards.
- Transfer: The intrinsic agent's policy is transferred to a slightly modified 'task' environment. A newly initialized 'extrinsic' agent leverages this transferred policy for guidance while learning from extrinsic rewards.

The authors posit that this transfer learning approach aligns with human learning, where knowledge from one domain can accelerate learning in a related domain. In the case of CBET, the agent learns environment centric knowledge during pretraining which it can transfer to the task environment. The transferred intrinsic agent's policy influences the extrinsic agent's policy according to the following equation:

$$\pi_{TASK}(s, a) = \sigma(f_e(s, a) + f_i(s, a)) \tag{2.4}$$

where $\pi_{TASK}(s, a)$ is the policy used in the task environment, σ is the softmax function, $f_e(s, a)$ is the extrinsic agent's policy network, and $f_i(s, a)$ is the intrinsic agent's policy network. During task environment training, the exploratory intrinsic network remains fixed while the extrinsic network is updated. Initially, the exploratory behavior dominates, but as the extrinsic network learns to optimize for rewards, its actions gradually gain influence.

The original C-BET experiments pitted an IMPALA agent guided by C-BET intrinsic rewards against IMPALA agents using alternative intrinsic reward strategies. This comparison was conducted across a diverse set of environments. In several of these environments, C-BET demonstrated significantly higher success rates and extrinsic returns in both transfer learning and the regular tabula rasa approaches. These results strongly indicate that C-BET facilitates effective exploration and policy transfer to new but related environments, making it a valuable tool for RL.

2.3 IMPALA

The IMPALA algorithm (Importance Weighted Actor-Learner Architecture) created by Espeholt et al. (2018) is a distributed RL algorithm which utilizes resources efficiently and easily scales up to thousands of machines. It utilizes an Actor Critic architecture where the process of learning the policy π and value function V^π is decoupled from the generation of experiences. This implies that IMPALA is an off-policy algorithm where the target policy π is updated using data collected by a behavior policy μ.

2.3.1 Actor-Critic Setup

In IMPALA, a set of actors update their behaviour policy μ to the latest target policy π from a learner and interact for n steps in the environment. The trajectories for these n steps are then sent back to the learner along with the policy distributions $\mu(a|s)$ through a queue. The learner then uses these

trajectories to update the policy π and the value function V^{π}.

Due to the asynchronous nature of the algorithm, the behaviour policy of an actor can lag behind the target policy. This discrepancy can lead to an instability in the calculation of policy gradients which the authors address using their V-TRACE technique. This algorithm utilizes importance sampling to correct for the difference in the behaviour and target policy.

2.4 DreamerV3

DreamerV3 (Hafner et al., 2023) is a state-of-the-art RL algorithm that implements model-based RL using world models. The algorithm develops a model of the world by learning a state representation from the environment's observations. This state representation is then used to predict future states and rewards, enabling the agent to plan ahead and make informed decisions. Unlike traditional model-based algorithms like Dyna (Sutton, 1991), the agent never observes the true state of the environment, instead relying solely on the world model's predictions.

When compared to its predecessors (Hafner et al., 2020, 2022) DreamerV3 introduces several improvements, including a better transformation to handle the large scale of rewards across environments, discretization of the critic's value function, and an entropy bonus to encourage exploration. Additionally, DreamerV3 uses λ returns which keep track of the reward across a series of steps which helps in stabilizing the learning process. All these changes contribute to the algorithm's ability to learn more efficiently across numerous environments without any environment-specific hyperparameter tuning.

2.4.1 World Model

The dreamer family of algorithms is built around the concept of a world model which is implemented as a Recurrent State-Space Model (RSSM) (Hafner et al., 2019). In contrast to the traditional Markov Decision Process (MDP), the RSSM formulates RL control as a Partially Observable Markov Decision Process (POMDP) and assumes that the environment is governed by a latent state h_t which is not directly observable. In order to maintain the Markov property, the agent must maintain a belief over the latent state h_t given the previous encoded observations z_{t-1}, actions a_{t-1} and the latent state h_{t-1}. This belief is parameterized by a deterministic recurrent neural network m which is used to model the latent state h_t across several time steps:

$$h_t = m(h_{t-1}, z_{t-1}, a_{t-1}) \tag{2.5}$$

To facilitate the planning process, the world model also outputs an expected reward r_t and continuation flag c_t which indicates whether the episode has terminated. These variables along with the latent state h_t permit the agent to solely rely on the world model's predictions to make decisions. Aside from the recurrent neural network used to predict the latent state, the world model uses stochastic neural networks to introduce an element of randomness in its predictions (Rezende et al., 2014; Kingma & Welling, 2022). This is perhaps most pertinent in the encoder q which is a Convolutional Neural Network (CNN) (LeCun et al., 1989) that processes the input observation x_t and model hidden state h_t into an encoded state z_t through reconstruction:

$$z_t \sim q(z_t|h_t, x_t) \tag{2.6}$$

The introduction of stochasticity is crucial for exploration as it allows the agent to sample different trajectories and learn from the resulting experiences.

2.4.2 Actor-Critic Setup

DreamerV3 utilizes an actor-critic architecture. However, unlike traditional setups, the agent receives input exclusively from the world model's latent state representation h_t, and encoded observation z_t. In order to promote continued exploration within environments characterized by sparse reward signals, the actor incorporates entropy regularization (Williams & Peng, 1991). This technique penalizes deterministic policies which could potentially arise in unstable environments.

The critic, in turn, produces an estimate of the value function which is used to evaluate the quality of the actor's policy. The authors make use of two-hot encoding to discretize the value function which allows the critic to capture more complex distributions of rewards across environments. Two-hot

encoding extends the concept of one-hot encoding to continuous values. A fixed-size array is initialized with zeros; the two entries nearest to the continuous value being encoded are set to values that sum to one, with the closer entry receiving a higher weight.

3 Methods

The original C-BET framework introduced a policy transfer mechanism for model-free agents, leveraging intrinsic rewards to guide learning. We propose extending this work to model-based agents, specifically DreamerV3, and assess the transfer learning approach's efficacy in environments with sparse rewards. IMPALA will serve as our baseline model-free agent for a performance comparison.

Our hypothesis is that DreamerV3's world model will enhance the modeling of environmental interestingness compared to IMPALA, leading to a more efficient exploration and policy transfer. This hypothesis builds upon the following key points:

- DreamerV3's world model explicitly represents the latent state of the environment and its evolution. This grants the agent a deeper understanding of environmental dynamics than IMPALA, which learns primarily from observations and rewards.

- DreamerV3's model-based nature allows it to simulate the environment without direct interaction. This translates to significantly higher sample efficiency compared to IMPALA, which necessitates extensive environment interactions for policy learning.

As suggested by Parisi et al. (2021), we will test our agents in two configurations: a tabula rasa setting where the agent learns from scratch in the target environment using both intrinsic and extrinsic rewards, and a transfer learning setting where the agent leverages the intrinsic reward learned in the source environment to accelerate learning in the task environment with extrinsic rewards. These two approaches will demonstrate the efficacy of our transfer learning strategy in sparse reward environments.

3.1 Intrinsic Rewards

We adopt the intrinsic reward mechanism based on interestingness proposed by Parisi et al. (2021) as formulated in Equation 2.3 and utilize pseudo-counts to estimate the novelty of states. Similar to the original C-BET paper, we hash the states as proposed by Tang et al. (2017) to account for similar states in complex environments.

Mirroring the C-BET approach, we also randomly reset the counts with a probability of $p \leq 1-\gamma_i$ where γ_i is the discount factor of the intrinsic reward. This is done to prevent the intrinsic reward from vanishing as the visitation count increases. Importantly, these resets occur randomly during exploration instead of solely at episode boundaries. This prevents an artificial bias where initial states consistently yield higher intrinsic rewards compared to later states. It's crucial to acknowledge that this formulation results in a non-stationary intrinsic reward, where the reward for a particular state can vary over time. Since the intrinsic reward is used solely for encouraging exploration, this non-stationarity is not a concern.

3.2 Transfer Learning Architecture

We employ a two-stage transfer learning process as outlined in Section 2.2. For our IMPALA agent, we directly replicate the C-BET approach, applying the softmax operator to the logits of the intrinsic and extrinsic agent's actor networks.

However, DreamerV3's reliance on its world model necessitates a distinct transfer approach. Instead of solely transferring the actor network logits, we propose a novel method where both the world model and actor network are transferred for use in the policy of the agent in the task environment:

$$\pi_{TASK(s,a)} = \sigma(f_i(w_i(x), a) + f_e(w_e(x), a)) \quad (3.1)$$

where f_i and f_e are the actor networks of the intrinsic and extrinsic agents, x is the observation of the environment and a are the available actions. We utilize abstract functions w_i and w_e to denote the output of the respective agents, which is then fed into their actor network to determine the policy.

This strategy leverages the world model's learned environmental dynamics to guide the policy transfer process. Crucially, our modification operates at

a high level of abstraction, allowing its application across diverse world model architectures. We theorize that this modification will enhance the transfer learning process, resulting in superior performance within the task environment.

3.3 Environments

To test our agents, we will evaluate their performance on the Minigrid worlds (Chevalier-Boisvert et al., 2023) and Crafter (Hafner, 2021) environments. These environments are known for their sparse reward structures, making them ideal for assessing the efficacy of our tabula rasa and transfer learning approaches.

3.3.1 Minigrid Worlds

The Minigrid suite offers a collection of procedurally generated, grid-based environments designed to evaluate the generalization abilities of RL agents (Chevalier-Boisvert et al., 2023). Agents operating in these environments must navigate rooms, gather keys, and unlock doors in an orthogonal maze to complete their goal. These environments pose a significant challenge for traditional RL algorithms since rewards are only provided upon goal completion. Furthermore, the agent has limited visibility with only the region in front of it visible necessitating a robust exploration strategy.

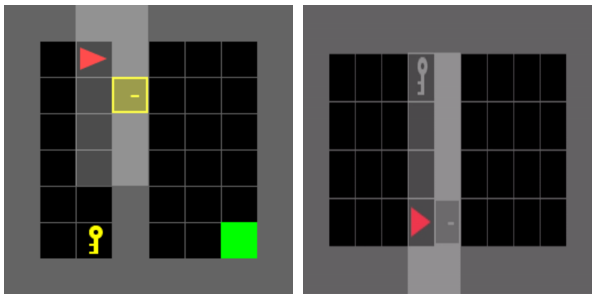

Figure 3.1: Minigrid environments: Doorkey (left) and Unlock (right). The agent's observations (light coloured squares) consist of a 7 × 7 grid infront of it.

Our main task environment will be 'Unlock'. Here, the agent must locate a key within a two-room layout and unlock a door. This environment will be used for evaluating both the tabula rasa agent and the transfer agent after pre-training. The exploration environment will be 'Doorkey'. While retaining the key and door elements, the objective shifts. The agent must additionally reach the green square in the bottom-right of the maze. Moreover, the color of the key and door remain fixed to yellow and the middle wall can shift locations. This environment will be used for pre-training the transfer agent to assess its ability to generalize to novel configurations.

3.3.2 Crafter

Crafter presents a procedurally generated environment where agents must demonstrate complex planning and execution to gather resources, craft items, fend off hostile entities, and construct structures to achieve specific goals (Hafner, 2021). Success in Crafter requires the ability to handle multi-step tasks effectively. The agent's action space is limited to a discrete set including movement, resource gathering, and item crafting. The episode only ends when the agent reaches zero health points. Rewards are sparse, granted only upon achievement completion, with smaller rewards for health gains or losses to promote survival. Crafter offers 22 achievements in total, with a spectrum of difficulty ranging from crafting basic tools to attaining diamonds. The environment provides partial observability, with the agent's view confined to a small grid centered on its position. Notably, this view is direction-independent, reducing the number of unique states to be accounted for.

A significant strength of the Crafter environment is its standardized budget of one million environment frames. This allows for direct performance comparisons across research papers, streamlining the evaluation of different agents. For our experiments, we will pre-train transfer learning agents in a single environment (fixed seed) and subsequently assess their performance in environments with randomized seeds. We believe this approach sufficiently tests the agents' ability to generalize their acquired knowledge to novel configurations, providing a measure of their transfer learning capabilities.

3.4 Experimental Setup

Our tabula rasa agents will be receive a computational budget of one million steps for training in the

Figure 3.2: Crafter environment. The agent is provided a top down view of the game with statistics at the bottom.

task environment. Conversely, our transfer learning agents will be pre-trained in the exploratory environment for one million steps before being fine-tuned in the task environment for an additional one million steps.

We do not perform any hyperparameter tuning for our agents, opting to use the default settings provided by the original C-BET and DreamerV3 papers. Nonetheless, we still need to tune the intrinsic strength coefficient for tabula rasa agents. This coefficient controls the balance between intrinsic and extrinsic rewards, influencing the agent's exploration behavior. Refer to the Appendix for our tuning methodology and findings.

3.5 Evaluation Metrics

For both Minigrid and Crafter environments, we will evaluate our agents based on their cumulative extrinsic reward also known as the return. This metric allows us to determine the agent's overall performance in the environment, capturing its ability to complete tasks and maximize returns. We utilize several evaluation episodes to assess the agent's performance over time as it progressively trains in the environment.

4 Results

Our results can be observed in Figure 4.1 for both tabula rasa and transfer learning experiments. We observe that the minigrid environments are quite challenging for the agents as all algorithms struggle initially. This phenomenon occurs as a reward is only issued upon goal completion which makes feedback sparse. Conversely, performance in the crafter environments rise steadily as rewards are provided for every achievement completed. Therefore, the agent can obtain several rewards in an episode which provides ample feedback for learning despite the higher complexity of tasks compared to the minigrid environments.

Examination of the tabula rasa experiments reveals that all DreamerV3 agents outperform the IMPALA baselines at every timestep. Additionally, CBET appears to be beneficial to DreamerV3 as its modified CBET variant obtains slightly higher returns compared to the original. In minigrid, this gap is more pronounced towards the beginning of training. However, the difference diminishes as training progresses. In contrast, the crafter environments show a more consistent performance gap between DreamerV3 and and its CBET variant. We can thus conclude that CBET is beneficial for DreamerV3 in both tabula rasa environments.

The transfer learning results paint a similar picture. The DreamerV3 agents outperform IMPALA in both environments at nearly all stages of training. Interestingly, there is a moment towards the beginning of the minigrid experiment where IMPALA outperforms DreamerV3. This phenomenon is not observed in the crafter environment. This result might suggest that either the policy transfer method proposed in Equation 3.1 is ineffective or the CBET intrinsic reward is not aligned with the extrinsic reward in the minigrid environment.

Overall, our results suggest that DreamerV3 obtains superior extrinsic return and faster convergence compared to IMPALA. The CBET variant of DreamerV3 also shows a slight improvement over the original DreamerV3 in the tabula rasa experiments. We can thus conclude that the DreamerV3 agent is more sample efficient and performs better in sparse reward environments compared to IMPALA. The CBET variant of DreamerV3 further enhances its performance in these environments by providing an additional intrinsic reward signal to

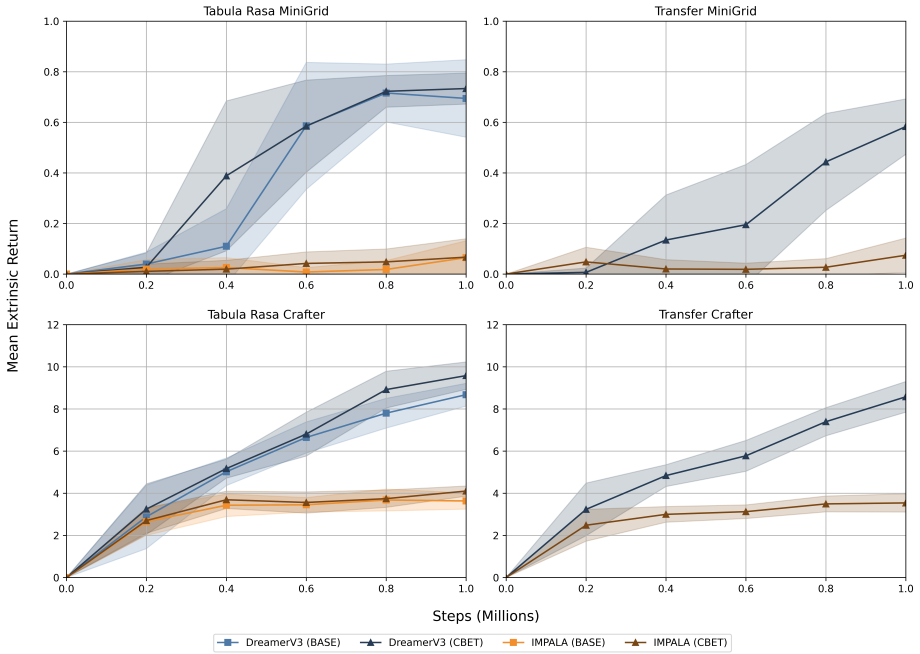

Figure 4.1: Results of the experiments. Mean extrinsic return plotted with standard error. Transfer learning experiments only contain the CBET variants of the algorithms.

guide exploration and search for interesting states.

5 Discussion

DreamerV3's performance, particularly its superiority over IMPALA, underscores its impressive sample efficiency. The enhanced sample efficiency is largely attributable to DreamerV3's ability to learn and simulate the environment dynamics internally through its world model, which significantly reduces the need for extensive interaction with the environment for policy learning. This is in contrast to IMPALA, which relies heavily on direct environment interactions, thus necessitating a greater number of samples and computational resources.

The CBET variant of DreamerV3 demonstrates an additional performance boost over the standard variant. This improvement suggests that the intrinsic rewards provided by the CBET mechanism introduce valuable knowledge to the model, knowledge that is otherwise challenging for the world model to acquire through extrinsic rewards alone. The intrinsic rewards effectively guide the agent to explore novel states, thereby enriching the learning process with diverse experiences that the world model may not capture without such motivation.

A notable observation in the minigrid transfer experiment is the brief period where IMPALA outperforms DreamerV3. This anomaly hints at potential issues with the proposed policy transfer mechanism. It is plausible that the current approach does not adequately facilitate the transfer of learned policies. Future investigations could explore fractional transfer methods, selectively resetting and retraining components of the world model while keeping others fixed (Sasso et al., 2021). This strategy might enhance the effectiveness of policy transfer by maintaining stability in parts of the model that generalize well across tasks.

Despite DreamerV3's overall superior performance, both the standard and CBET variants fall short of the extrinsic rewards reported in the original DreamerV3 paper (11.7 ± 1.9). This discrepancy can be attributed to the constrained planning ratio used in our experiments, compared to the original paper's. The higher planning ratio allows the model to spend more time simulating the en-

vironment per step, leading to better performance. Hardware limitations precluded us from employing such high planning ratios, thereby impacting the agents' learning efficacy.

Comparing IMPALA and DreamerV3, while the latter achieves better performance, it does so at the cost of significantly increased training time and parameter count. DreamerV3's sophisticated world model demands more computational resources and longer training periods. In contrast, IMPALA's simpler architecture allows for faster training completion given equivalent computational resources. This trade-off between performance and computational efficiency is a critical consideration in choosing between these models for practical applications.

The use of intrinsic rewards and the CBET mechanism, despite their benefits, come with limitations. Tuning the intrinsic strength coefficient is challenging; if set too high, the agent may focus excessively on exploration at the expense of task completion, while if set too low, the intrinsic rewards fail to sufficiently drive exploration. Furthermore, CBET might not be suitable for all environments. For instance, in tasks requiring stable behavior, such as autonomous driving, CBET-induced exploration could lead to unsafe actions.

Future research directions include developing an intrinsic strength coefficient scheduler akin to learning rate schedulers. This scheduler would start with a high intrinsic motivation to encourage exploration and gradually reduce it to promote exploitation. This approach, potentially coupled with the ADAM optimizer (Kingma & Ba, 2014), could mitigate the need for extensive tuning. Additionally, examining the interaction of intrinsic reward techniques with various transfer learning methods, such as fractional transfer, could provide deeper insights. Testing these approaches in more complex environments, such as Minecraft or robotics, and improving the interpretability of reinforcement learning algorithms to better understand the agents' decision-making processes are also promising avenues for further investigation.

6 Conclusion

Ultimately, while DreamerV3 and its CBET variant demonstrate significant advantages in sample efficiency and performance in sparse reward environments, they also present challenges in computational demands and applicability across different tasks. Our experiments show that DreamerV3 outperforms IMPALA in terms of learning efficiency and overall performance, particularly when enhanced with intrinsic rewards through the CBET mechanism. However, this improved performance comes at the cost of significantly increased training time and computational resources due to the complex world model architecture. Furthermore, the brief superior performance of IMPALA in the minigrid transfer experiment suggests potential issues with the current policy transfer approach, highlighting the need for more effective transfer learning techniques.

Addressing these issues through advanced transfer learning methods, such as fractional transfer, could enhance the robustness of policy transfer. Additionally, developing a scheduler for intrinsic reward strength can optimize the exploration-exploitation balance over time, reducing the need for extensive parameter tuning. Future research into the interpretability of reinforcement learning algorithms is crucial for understanding the decision-making processes and the impact of intrinsic rewards on agent behavior. Moreover, efforts to reduce the computational resources required for DreamerV3 without compromising its performance will make these advanced techniques more accessible and practical for a wider range of applications. Through these improvements, we can pave the way for more robust and versatile reinforcement learning models that can effectively tackle complex, real-world problems.

References

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. *CoRR*, *abs/1606.01868*. Retrieved from `http://arxiv.org/abs/1606.01868`

Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., ... Terry, J. (2023). Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, *abs/2306.13831*.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., ... Kavukcuoglu, K. (2018). *Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures.*

Guo, Y., Fu, Y., Peng, R., & Lee, H. (2022). Learning exploration policies with view-based intrinsic rewards. In *Deep reinforcement learning workshop neurips 2022.*

Hafner, D. (2021). Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*.

Hafner, D., Lillicrap, T., Ba, J., & Norouzi, M. (2020). *Dream to control: Learning behaviors by latent imagination.*

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2019). *Learning latent dynamics for planning from pixels.*

Hafner, D., Lillicrap, T., Norouzi, M., & Ba, J. (2022). *Mastering atari with discrete world models.*

Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). *Mastering diverse domains through world models.*

Hessel, M., Danihelka, I., Viola, F., Guez, A., Schmitt, S., Sifre, L., ... van Hasselt, H. (2022). *Muesli: Combining improvements in policy optimization.*

Kapturowski, S., Campos, V., Jiang, R., Rakićević, N., van Hasselt, H., Blundell, C., & Badia, A. P. (2022). *Human-level atari 200x faster.*

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR, abs/1412.6980*.

Kingma, D. P., & Welling, M. (2022). *Auto-encoding variational bayes.*

Küttler, H., Nardelli, N., Lavril, T., Selvatici, M., Sivakumar, V., Rocktäschel, T., & Grefenstette, E. (2019). *Torchbeast: A pytorch platform for distributed rl.* Retrieved from `https://arxiv.org/abs/1910.03552`

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541-551. doi: 10.1162/neco.1989.1.4.541

Parisi, S., Dean, V., Pathak, D., & Gupta, A. (2021). Interesting object, curious agent: Learning task-agnostic exploration. In A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems.*

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). *Curiosity-driven exploration by self-supervised prediction.*

Raileanu, R., & Rocktäschel, T. (2020). *Ride: Rewarding impact-driven exploration for procedurally-generated environments.*

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). *Stochastic backpropagation and approximate inference in deep generative models.*

Sasso, R., Sabatelli, M., & Wiering, M. A. (2021). *Fractional transfer learning for deep model-based reinforcement learning.* Retrieved from `https://arxiv.org/abs/2108.06526`

Sutton, R. S. (1991, jul). Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, *2*(4), 160–163. Retrieved from `https://doi.org/10.1145/122344.122377` doi: 10.1145/122344.122377

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.* Cambridge, MA, USA: A Bradford Book.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., ... Abbeel, P. (2017). *#exploration: A study of count-based exploration for deep reinforcement learning.*

Williams, R. J., & Peng, J. (1991). Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, *3*(3), 241–268. Retrieved from `https://doi.org/10.1080/09540099108946587` doi: 10.1080/09540099108946587

A Computational Resources

We thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Hábrók high performance computing cluster. All experiments mentioned in the main report were conducted for 1 million steps. Our DreamerV3 models were run on a single node with an Intel Xeon Gold 6150 CPU and Nvidia V100 GPU for approximately 12 hours. Our IMPALA models used a single node with 8 Intel Xeon Gold 6150 CPUs and an V100 GPU for approximately an hour and a half.

B Model and Experiment Hyper-parameters

For DreamerV3, we used the implementation provided by the authors. Conversely, we used the TorchBeast implementation for IMPALA (Küttler et al., 2019). We mainly utilize the same hyper-parameters as described in the original DreamerV3 and IMPALA papers (Hafner et al., 2023; Espeholt et al., 2018). The number of actors which interact with the environment in IMPALA were reduced to 8 due to the limited resources available to us. Likewise, we had to carefully select the planning ratio in DreamerV3 which determines the amount of time the model spends simulating the environment. We used a planning ratio of 32 in Minigrid and a higher value of 64 in the more complex Crafter environment. Both algorithms share the same evaluation strategy. 8 Evaluation episodes were scheduled every 10000 steps and the mean extrinsic return was recorded to obtain the results presented in the main report.

C Equivalent Training Time Comparison

We were curious how IMPALA would compare to DreamerV3 if we were to train them for an equivalent amount of time. We conduced an experiment in Crafter where both DreamerV3 and IMPALA were augmented with CBET and given a time budget of 15 hours. As seen in Figure C.1, DreamerV3 still manages to outperform IMPALA in terms of extrinsic reward. These findings attest to the high sample efficiency of DreamerV3. Nonetheless, it is important to note that IMPALA can be efficiently scaled to use more resources and potentially outperform DreamerV3 in the same time frame. The best choice of algorithm will depend on the available computational resources and the desired training time.

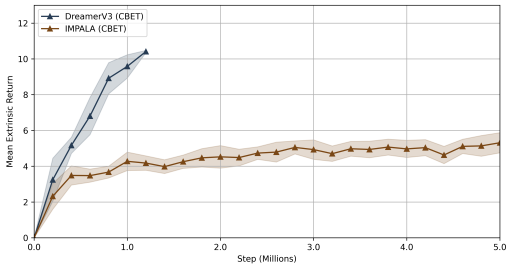

Figure C.1: Equivalent Training Time Comparison between IMPALA and DreamerV3.

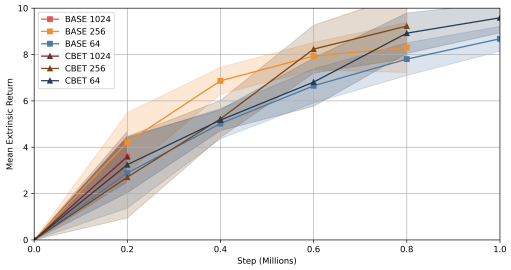

Figure D.1: Impact of Planning Ratio on DreamerV3 with CBET in Crafter.

D Impact of Planning Ratio

We wished to determine the effect of the interaction between the planning ratio and the use of CBET on the performance of DreamerV3. We hypothesized that as the planning ratio increases, the gap between the performance of DreamerV3 with and without CBET would decrease. We arrived at this conclusion because a higher planning ratio allows the model to spend more time simulating and understanding the environment per step, potentially making the intrinsic rewards redundant.

To test our hypothesis, we performed a grid search in the Crafter environment using Dream-

erV3 with and without CBET. As the training time drastically increases with higher planning ratio, we applied a time limit of 24 hours for each configuration along with a limit of 1 million steps. The results of this experiment are shown in Figure D.1. We observe that the gap between returns appears to decrease as the planning ratio increases. With a planning ratio of 1024, there is no substantial difference between the performance of DreamerV3 with and without CBET. However, we refrain from drawing definitive conclusions from this experiment due to our lack of data points and computational resources. We severely underestimated the real world time required during the planning phase of the model. As a result, the experiments were forced to terminate before reaching the desired number of steps.

E Intrinsic Reward Scaling

To prevent the agent from focusing excessively on exploration at the expense of task completion, we needed to scale the intrinsic reward before adding it to the extrinsic reward. If this constant factor c is set too high, the agent will neglect its task. Conversely, if c is set too low, the intrinsic rewards may not sufficiently drive exploration. We performed a grid search to find the optimal scaling factor among our candidates across both environments and algorithms. Figure E.1 shows the results of this process. We selected the scaling factor that yielded the best performance in each case at the end of the training period. The best scaling factors were:

1. IMPALA in Minigrid: 0.0025
2. DreamerV3 in Minigrid: 0.0025
3. IMPALA in Crafter: 0.005
4. DreamerV3 in Crafter: 0.001

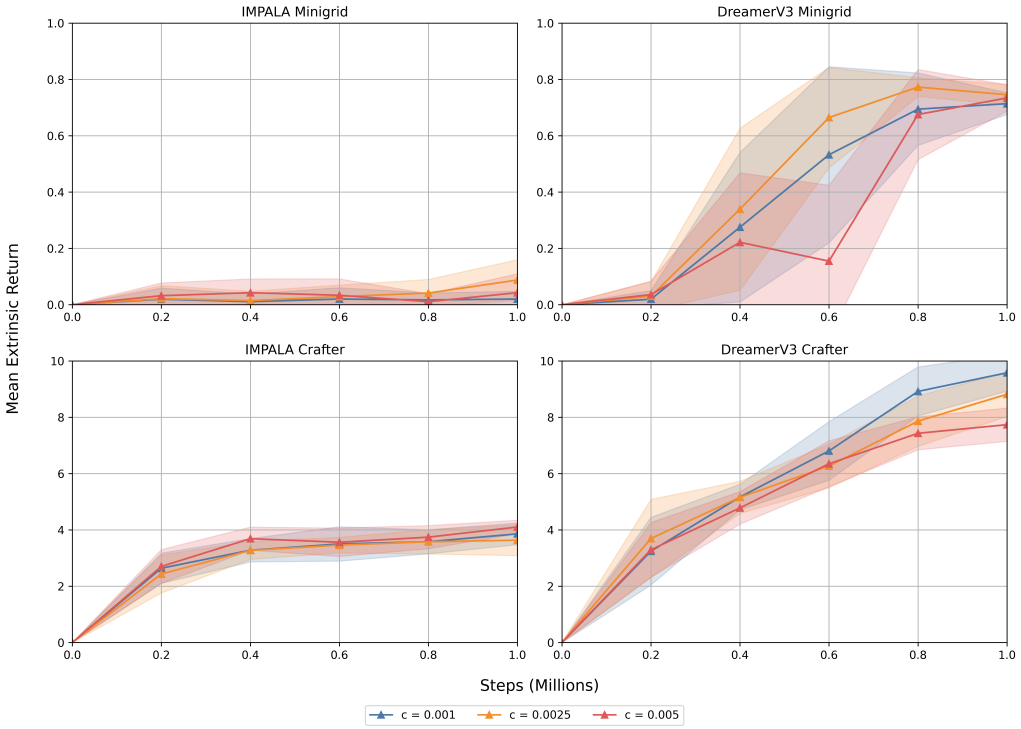

Figure E.1: Results of the grid search for the optimal intrinsic reward scaling factor c. Extrinsic returns plotted with standard errors.