DreamerV3 in Walker2d: Confounding and generalization

Shubham Subhnil [1] **Vinny Cahill** [1] **Ivana Dusparic** [1]

Abstract

DreamerV3 is a SOTA model-based reinforcement learning (MBRL) agent that is sample efficient in both low-dimensional and vision based environments. It also shows superior scalability compared to other MBRL agents on a single GPU for training. Our aim is to gather empirical evidence on DreamerV3's performance in confounded environments and few/zero-shot generalization in DM Control Suite's Walker2d environment. We chose the low-dimensional vector based, rather than image based, observation setting called proprio. Note that DreamerV3 isn't reportedly the SOTA in generalization.

1. Introduction

We used the original implementation of DreamerV3 (Hafner et al., 2023) to train on the Walker2d environment from DM Control Suite (DMC) (Tassa et al., 2018). DreamerV3, among few other methods, were chosen to run baseline experiments in confounded DMC and Robotsuite (Zhu et al., 2022) environments. The report presents the results from our experiments with DreamerV3 in the Walker2d environment of DMC. The objectives of this study are summarized as:

- Test if DreamerV3 is reproducible.
- Test robustness of DreamerV3 in the presence of unobservable confounders.
- Test DreamerV3 performance when more information is available.
- Test DreamerV3 in few-shot and zero-shot generalization scenarios.

Our results are summarized as follows:

- The results from our experiments in the walk and run tasks of the walker2d environment are consistent with the published work on DreamerV3.
- DreamerV3 does learn a solution in the presence of UCs. Although, it is possible that one of the other methods to be tested can find a better policy than DreamerV3.
- DreamerV3 doesn't show any improvement in sample efficiency when we have extra information available.
- DreamerV3 exhibits poor few-shot efficiency and never converges to the expert policy in the new task with and without UCs. However, training in tasks with UCs leads to better few-shot generalization compared to training without UCs.
- DreamerV3 shows forgetfulness when evaluated on the base task after few-shot training on other task.

2. Experiments

The walker2d environment in DMC that DreamerV3 is tested in, is a popular continuous control benchmark for reinforcement learning agents. See appendix A for environment details.

2.1. Run codes

The following codes are used to describe individual runs.

- **W / R :** Walk / Run tasks
- **V / Cs / Cp :** Vanilla / Confounded Swelling force / Confounded Step force
- **U / Of / Oa :** Unobservable / Observable force / Observable acceleration (if part of observation space or not)
- **S / M :** Small / Medium model sizes

2.2. Baselines

In figure 1, two model sizes were tested on *walk* and *run* tasks: *small* and *medium*. No confounding effects were introduced. *Small* model is efficient for mobile GPUs and

*Equal contribution [1]School of Computer Science and Statistics, Trinity College Dublin. Correspondence to: Shubham Subhnil <subhnils@tcd.ie>, Vinny Cahill <vjcahill@tcd.ie>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

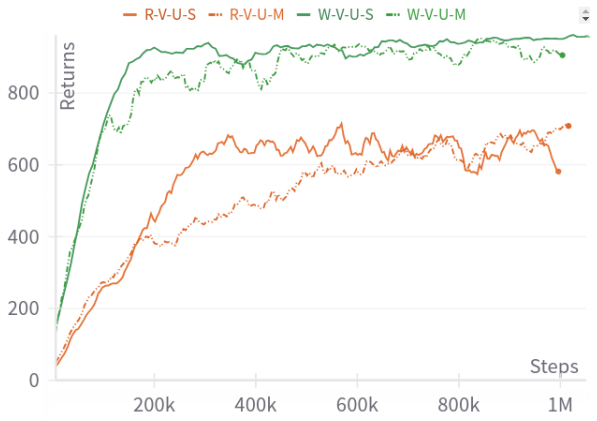

Figure 1. Walk and *Run* tasks on small *(solid)* and medium *(dashed)* model sizes without confounding force. Same color represents same task.

needs less training time than the *medium* model to converge to the optimal policy. However, the medium model may converge to better optimal policy given more training interactions according to the results in the original paper. The *walk* task tracks a higher return per episode than the *run* task. The walker has to run at a speed of at least 8kmph to score the reward for running.

Figure 2 on *walk* task shows that DreamerV3 can learn an optimal policy as good as the *vanilla* scenario in the presence of *swelling* force. Swelling force is less sparse and gradual than step force which tends to knock the walker down. It is possible that the agent finds the most optimal solution. However, looking at the video of the walker in *step* confounded environment, we can imagine better solutions to counter the *step* force, which the DreamerV3 agent cannot find. Further experiments with few other agents can help us find innovative solutions that Dreamer agent hasn't explored yet.

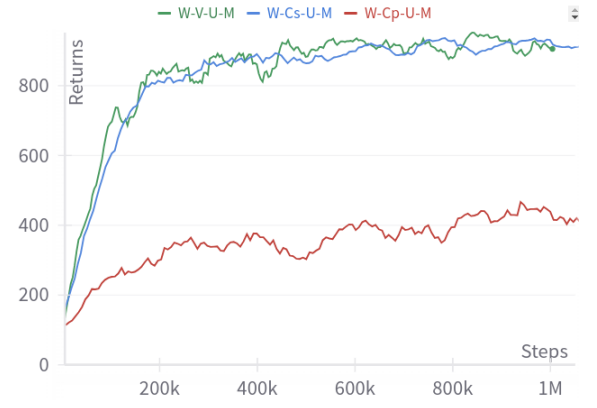

Figure 2. Walk task: Comparison of confounding vs no confounding forces

Canonically, the *run* task returns a lower reward per episode since the walker has to run at $\geq$ 8kmph to receive the reward for running, in addition to other reward parameters **(see appendix)**. From figure 3, adding the confounder (swelling and step force) shows a larger drop in episodic reward than the *walk* task.

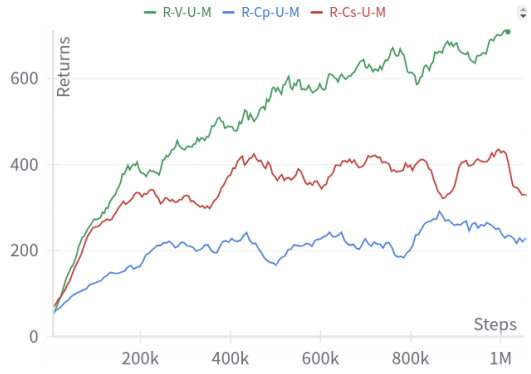

Figure 3. Run task: shows significant performance loss from vanilla *(green)* to swelling force *(red)* confounding.

2.2.1. Confounders are observable

We hypothesised whether a better optimal policy can be learnt in confounded scenarios if the confounding variables are visible i.e., part of the observation space. For fairness towards DreamerV3, we, in separate runs, made either the external forces on or the acceleration of the torso observable to the agent. Our intention was to only study the agent's behavior given the external forces are observable. Figure 4 shows that Dreamer finds a sub-optimal policy *(solid lines)* when the forces on the torso of the walker are made observable (i.e., part of the observation space).

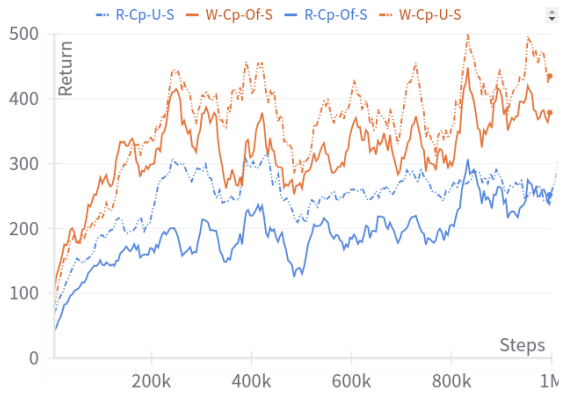

Figure 4. Walk and *run* tasks: Making force observable *(solid lines)* leads to a sub-optimal policy for both walk *(orange)* and run *(blue)* tasks.

The *step* force profile is quite sparse and the force observation vector is always populated by zeros except the time step

where the force is being applied. This sparsity introduces a bias in either the world model or the policy agent's gradient (e.g., A2C) which leads to sub-optimal performance compared to when forces are unobservable.

Intuitively, no physical robot can sense the force but the IMUs (inertial measurement unit) present in every machine sense the accelerations. We chose to make the acceleration visible (see figure 5 and 6) given the non-sparse nature of acceleration vector. Acceleration is present regardless of the confounding force and the acceleration vector itself is confounded when there is external force in the respective time-step. Confounding in this case can be viewed as a random variable additive to the acceleration.

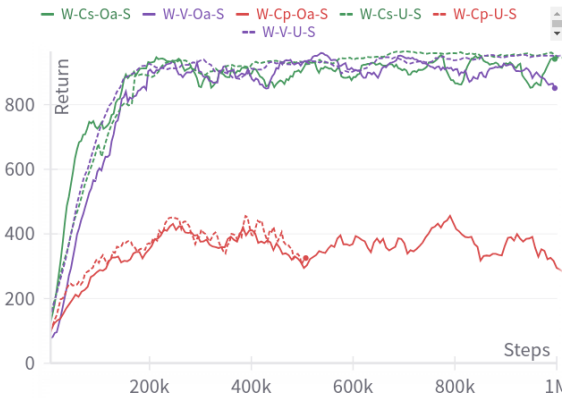

Figure 5. Walk task: Dreamer agents with observable *(solid lines)* or unobservable *(dashed lines)* acceleration vectors track similar policies.

Figure 6. Run task: Dreamer agents with observable *(solid lines)* or unobservable *(dashed lines)* acceleration vectors track similar policies.

One of our earlier expectations was to see performance decline with a larger observation space. But it is the sparsity in the observations (i.e., force vector) that leads to a performance decline (figure 4) while having more consistent extra information (i.e., acceleration vector) doesn't lead to a better optimal policy or sample efficiency. In figure 5 for the

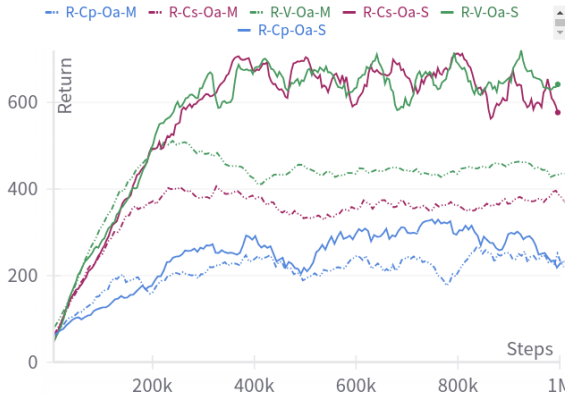

Figure 7. Run task: Given that the acceleration is visible, the small model *solid lines* in all cases, converges to a better optimal policy than the medium model *dashed lines*.

walk task, we see that the vanilla *(purple)* and confounded *(green, red)* runs converge to similar optimal policies for both observable *(solid line)* and unobservable *(dashed line)* acceleration vectors. Figure 6 on the *run* task shows similar behaviour. For the walker2d environment, we can conclude that having extra information doesn't lead to a better policy or sample efficiency.

2.3. Generalization

Few and zero-shot generalization performance in the presence of UCs is a crucial part of the current proposal for the PhD thesis. We put DreamerV3 through various tests where the first task is always *walk* followed by either few-shot training and evaluation or zero-shot evaluation in the *run* task. As mentioned in section 1, we find that DreamerV3 shows poor zero and few-shot generalization especially in the case of vanilla *walk* task to vanilla *run* task possibly because of the lack of exploration outside the distribution of the reward model in the vanilla *walk* task.

2.3.1. Few-shot Efficiency

In figure 8, we investigated the few-shot efficiency of DreamerV3 given that it has first been trained on either the vanilla *(red)* or the confounded scenario *(blue, orange)* of the first task (*walk*) followed by training on the *vanilla* scenario of the second task (*run*). The switch between tasks isn't automatic and the model is initialized each time the task is switched. However, the previously trained model is copied onto the current model with fresh seed for other processes. From figure 8 we find that, the model trained on *step* force confounded *walk* task *(orange)* converges to a better optimal policy than the model trained on vanilla *walk* task. This is likely because of the stochasticity in exploration due to UC. For walker2d, this suggests that training in the presence of UCs may lead to better few-shot generalization between

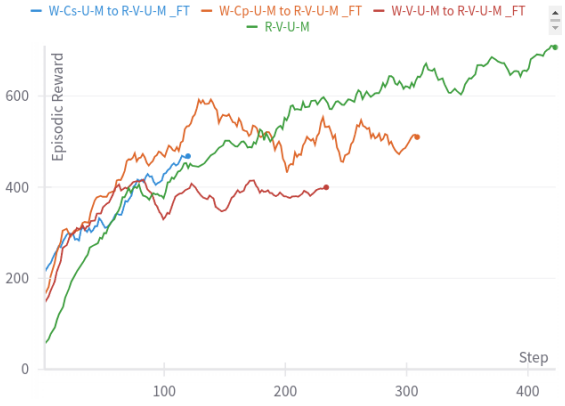

Figure 8. **Few-shot efficiency** from *Walk* to *run* task. _FT stands for few-shot training. All few-shot training performed on vanilla scenario of the *run* task. Agent trained on *vanilla* scenario of the *walk (red)* task has the worst few-shot efficiency.

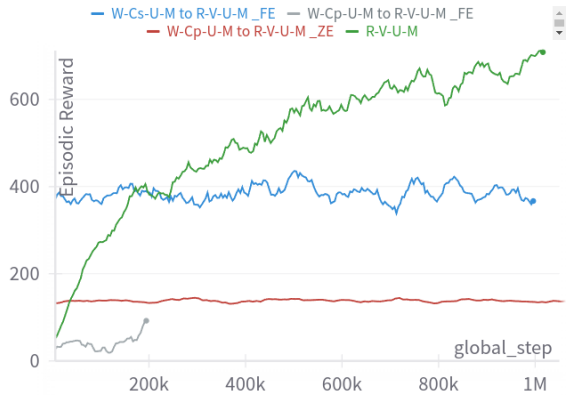

Figure 9. **Few/zero-shot evaluations** in confounded *walk* task to vanilla *run* task. _FE: Few-shot evaluation; _ZE: Zero-shot evaluation

tasks with different reward functions. It is possible that *swelling* force confounded *(blue)* model converges to the expert policy *(green)* while still being better than the *vanilla* scenario policy *(red)*.

None of the models however converge to the expert policy *(green)* which indicates that the reward model parameters cannot adapt to the new distribution of the reward function. This is in conjunction with the results in Plan2Explore (Sekar et al., 2020).

2.3.2. Few and zero-shot generalization

We look deeper into the case of few and zero-shot generalization confounded *walk* task to the vanilla *run* task. Figure 9 shows that the model trained first on the *swelling* force confounded *walk* task *(blue)* is a better policy than the zero-shot evaluation *(red)*. A discrepancy here is that the model trained on the *step* force confounded *walk (grey)* task wasn't given enough few-shot training steps with the *run* task. However, from section 2.3.1, it can be suggested that few-shot interaction with the new task given that the first task was confounded improves generalization.

2.3.3. Forgetfulness

Longer few-shot exploration time in the second task leads to worse performance in the first task. In figure 10, we few-shot trained the expert *walk* task agents in the *vanilla run* task. Followed by re-evaluation of the few-shot agents in the same version of the *walk* task in which they were experts. No further exploration steps were allowed when re-evaluating the few-shot models in the first task they were trained on.

Figure 10 shows significant performance loss in the base task once the expert has few-shot explored the new task.

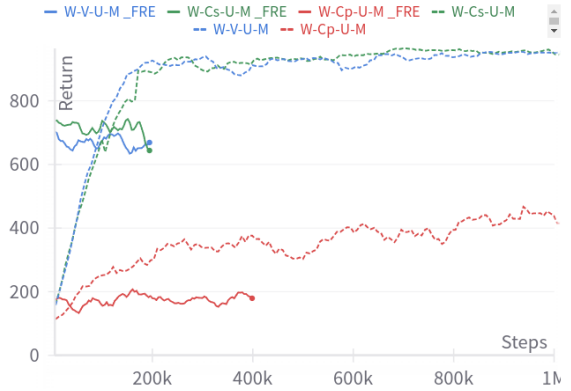

Figure 10. **Few-shot re-evaluations** on the base task. Same colors represent same force profile of the *walk* task. *Dashed* lines represent the expert policies trained on first *walk* task. The expert policies were then few-shot trained on *run* task. The *run* task is always vanilla. The few-shot policies are re-evaluated in the first *walk* task. **_FRE:** Few-shot re-evaluation on the same *walk* task scenarios.

To rephrase it, longer few-shot exploration in the new task leads to worse performance in the base task.

3. Conclusion

DreamerV3 is the SOTA in sample-efficiency in high-dimensional observation spaces. In the presence of UCs, however, the DreamerV3 agents fall short of finding the optimal policy (assuming other methods aren't as sample efficient as DreamerV3). Having extra observable information doesn't necessarily improve upon the shortcomings. It also has poor few and zero-shot generalization while exhibiting severe forgetfulness. Our PhD research aims at solving the issues that DreamerV3 cannot solve by either augmenting DreamerV3 with better new capabilities (like few publications) or proposing a similar approach but new modules to

handle generalization, forgetfulness and sample-efficiency in the presence of unobserved confounders.

3.1. Citations and References

References

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering Diverse Domains through World Models, January 2023. URL http://arxiv.org/abs/2301.04104. arXiv:2301.04104 [cs, stat].

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to Explore via Self-Supervised World Models, June 2020. URL http://arxiv.org/abs/2005.05960. arXiv:2005.05960 [cs, stat].

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. DeepMind Control Suite, January 2018. URL http://arxiv.org/abs/1801.00690. arXiv:1801.00690 [cs].

Zhu, Y., Wong, J., Mandlekar, A., Martín-Martín, R., Joshi, A., Nasiriany, S., and Zhu, Y. robosuite: A Modular Simulation Framework and Benchmark for Robot Learning, November 2022. URL http://arxiv.org/abs/2009.12293. arXiv:2009.12293 [cs].

A. Walker2d environment details

A.1. Dimensions

The dimensions of state, actions and observation spaces are as follows:

- **State space:** 18
- **Action space:** 6
- **Observation space:** 24

The observation space consists of the following dimensions by default:

- **Orientations:** 14
- **Height:** 1
- **Velocity:** 9

In the scenarios where we make the forces on the torso or the acceleration of the torso observable, the dimension of observation space increases by 3.

A.2. Reward functions

A.2.1. Walk task:

The walk task consists of 3 reward parameters:

- Upright: Reward for maintaining an upright angle of the torso
- Stand: Reward for maintaining a certain torso height above the floor.
- Move: Reward for moving at at least a certain speed in forward direction. For *walk* task, the speed is 1kmph.

A.2.2. Run task:

In *run* task, all the reward parameters are similar as *walk* task with the exception of the move speed. The move speed here is at least 8kmph.