Gradient-based Planning with World Models

Jyothir S V[1]* **Siddhartha Jalagam**[1]* **Yann LeCun** [1, 2] **Vlad Sobal**[1, 2]
[1]New York University [2]Meta AI
{jyothir, scj9994, us441}@nyu.edu
yann@cs.nyu.edu

Abstract

The enduring challenge in the field of artificial intelligence has been the control of systems to achieve desired behaviors. While for systems governed by straightforward dynamics equations, methods like Linear Quadratic Regulation (LQR) have historically proven highly effective, most real-world tasks, which require a general problem-solver, demand world models with dynamics that cannot be easily described by simple equations. Consequently, these models must be learned from data using neural networks. Most model predictive control (MPC) algorithms designed for visual world models have traditionally explored gradient-free population-based optimization methods, such as Cross Entropy and Model Predictive Path Integral (MPPI) for planning. However, we present an exploration of a gradient-based alternative that fully leverages the differentiability of the world model. In our study, we conduct a comparative analysis between our proposed method and other MPC-based alternatives, as well as policy-based algorithms. In a sample-efficient setting, our method achieves on par or superior performance compared to the alternative approaches in most tasks. Additionally, we introduce a hybrid model that combines policy networks and gradient-based MPC, which outperforms pure policy based methods thereby holding promise for planning with world models in complex real-world tasks.

1 Introduction

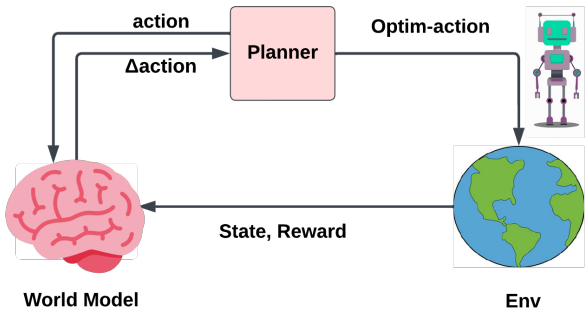
Until recently, model-free reinforcement learning (RL) algorithms [24][28] have been the predominant choice for visual control tasks, particularly in simple environments like Atari games. However, these model-free algorithms are notorious for their sample inefficiency and lack of generality. If the tasks change, the policy needs to be trained again. They are constrained by their inability to transfer knowledge gained from training in one environment to another. Consequently, they must undergo retraining for even minor deviations from the original task. Real-world applications where the agent needs to solve a multitude of different tasks in the environment, such as robotics, demand a more general approach.

To address this limitation, multiple types of methods have been proposed. In this work, we focus on model-based planning methods. These model-based approaches encompass three key components: a learned dynamics model that predicts state transitions, a learned reward or value model analogous to the cost function in Linear Quadratic Regulation (LQR) [6], which encapsulates state desirability information, and a planner that harnesses the world model and reward model to achieve desired states.

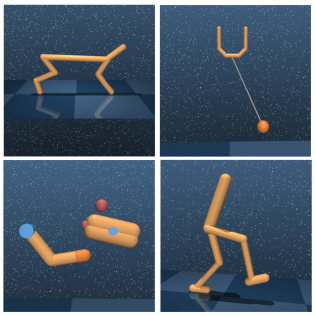
While previous research in planning using Model Predictive Control (MPC) [25] has primarily focused on gradient-free methods like cross-entropy[27, 9], these methods are computationally expensive and do not utilize the differentiability of the learned world model.

*Equal Contribution.

Preprint. Under review.



(a) **Gradient based Planning for world models**



(b) **DM Control**

Additionally Bharadhwaj et al. [5] have explored a combination of cross-entropy with gradient-based planning on a few tasks in the Deep Mind control suite, without fully exploring the potential of pure gradient based planning.

In this research paper, we delve into the potential of pure gradient-based planning, which derives optimal actions by back-propagating through the learned world model and performing gradient descent. Additionally, we propose a hybrid planning algorithm that leverages both policy networks and gradient-based MPC.

The key contributions of this paper can be summarized as follows:

1. Gradient-Based MPC: We employ gradient-based planning to train a world model based on reconstruction techniques and conduct inference using this model. We compare and contrast the performance of traditional population-based planning methods, policy-based methods, and gradient-based MPC in a sample-efficient setting involving 100,000 steps in the DeepMind Control Suite tasks. Our approach demonstrates superior performance on many tasks and remains competitive on others.
2. Policy + Gradient-Based MPC: We integrate gradient-based planning with policy networks, outperforming both pure policy methods and other pure MPC techniques in sparse reward environments.

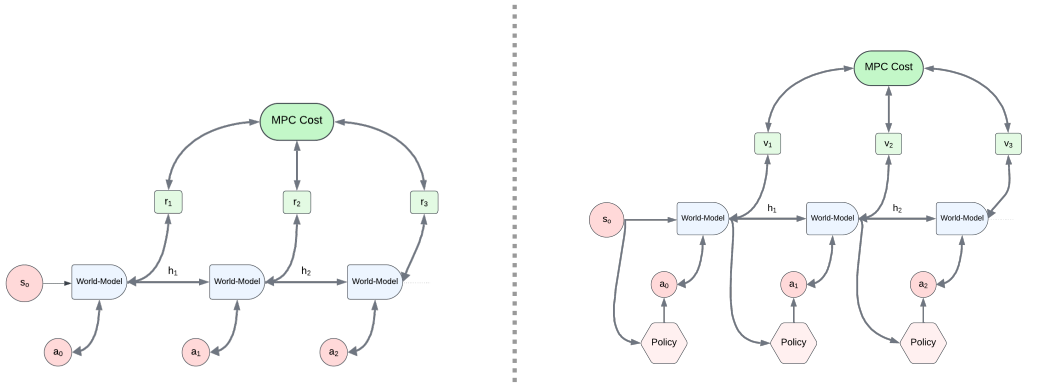
2 Related Work

World modelling (Sutton [33], Ha and Schmidhuber [12]) has emerged as a promising approachfor real world RL. It condenses previous experiences into dense representations [29], allowing for predictions about potential future events. Transformer-based [23, 7, 26] world models have delivered promises of sample efficient representations, which was main issue with Model Free RL methods. A plethora of world modeling methods involving self-supervised loss have emerged BYOL ([11], VICReg[3], [31], MoCo v3 [30]). Reconstruction based methods (DreamerV3 [17]) have proven to work well in diverse set of complex environments[4, 34]. Our current work examines a technique on top of reconstruction based world modelling method, but it is generally applicable on top of any predictive world modelling method. Our proposed Policy+Grad-MPC method is close to the one proposed by [1], although as opposed to our method, MBOP is an offline algorithm and uses gradient free planning .

3 Preliminaries

3.1 Problem Formulation

We consider a partially observable Markov Decision Processes (POMDP) (O, S, A, T, R), where $O \in \mathbb{R}^n$ is observation, $S \in \mathbb{R}^n$ and $A \in \mathbb{R}^m$ are hidden state and continuous action spaces. $T : S \times A \times S \to \mathbb{R}^+$ is the transition (dynamics) model, R is a scalar reward . We use a value V



(a) **Gradient based MPC** (b) **Policy+MPC**

for the hybrid planning algorithm involving both policy network and gradient based MPC, instead of reward R. The goal for gradient based MPC, the hybrid method is to deduce a policy that maximizes $\sum_{i=t}^{t+H-1} R(\tilde{s}_i)$ and $\sum_{i=t}^{t+H-1} V(\tilde{s}_i)$. H is planning horizon.

3.2 Latent World Modelling

$$\begin{aligned} \text{Deterministic state model} &: h_t \leftarrow f(h_{t-1}, s_{t-1}, a_{t-1}) \\ \text{Stochastic state model} &: s_t \leftarrow p(s_t|h_t) \\ \text{Observation model} &: o_t \leftarrow p(o_t|h_t, s_t) \\ \text{Reward model} &: r_t \leftarrow p(r_t|h_t, s_t) \end{aligned}$$

The world model utilized in our study is the Recurrent State Space Model (RSSM), which uses a variational objective Kingma and Welling [19] and GRU Predictor Cho et al. [8] . The RSSM operates by dividing the overall state into two distinct components: the deterministic state and the stochastic state.

The deterministic state model accepts inputs consisting of the current deterministic state, the stochastic state from the previous time step, and an action. It then processes these inputs to produce the current deterministic hidden state.

On the other hand, the stochastic state model is approximated through a neural network that is conditioned on the deterministic hidden state. This model characterizes the stochastic state.

Both the observation model and the reward model are conditioned on both the deterministic hidden state and the stochastic hidden state. The stochastic state component is designed to capture the inherent randomness and variability in the input data, while the deterministic state component is responsible for capturing features that are entirely predictable

we infer approximate state priors from past observations and actions with the aid of an encoder

$$q(s_{1:T}|o_{1:T}, a_{1:T}) = \prod_{t=1}^{t=T} q(s_t|h_t, o_t) \tag{1}$$

Here $q(s_t|h_t, o_t)$ is a Gaussian whose mean and variance are parameterized by conjunction of a convolutional neural network [22] followed by a feed forward neural network.

we consider sequences $(o_t, a_t, r_t)_1^T$, o_t observation, a_t action and r_t reward. The RSSM model is trained with a combination of reconstruction and KL losses,described by the following equation.

DerivationA.3

$$\begin{aligned}\ln p(o_{1:T}|a_{1:T}) &= \ln \int \prod_t p(s_t|s_{t-1}, a_{t-1}) p(o_t|s_t)\, ds_{1:T} \\ &\geq \sum_{t=1}^{T} \big(\mathbb{E}_{q(s_t|o_{\leq t}, a_{<t})} \left[\ln p(o_t|s_t) \right] \\ &\qquad - \mathbb{E}_{q(s_{t-1}|o_{\leq t-1}, a_{<t-1})} \left[\mathrm{KL} \left[q(s_t|o_{\leq t}, a_{<t}) || p(s_t|s_{t-1}, a_{t-1}) \right] \right] \big) \end{aligned} \tag{2}$$

The reward loss is computed similar to the observation loss.

3.3 Planning

Planning can be formalized as finding the best sequence of actions given a predictive model f, reward function r, and value function V. The planning optimization process aims to determine the optimal sequence of actions of length H that maximizes the cumulative reward over the entire trajectory:

$$\pi(s_t) = \arg\max_{a_{t:t+H}} \sum_{i=t}^{t+H-1} \gamma^i R(\tilde{s}_i) + \gamma^H V(\tilde{s}_{t+H}) \qquad \hat{s}_t = s_t,\ \hat{s}_{t+1} = f(\hat{s}_t, a_t) \tag{3}$$

The task of planning can be accomplished through various methodologies. One notable approach, PlaNet, employs the cross-entropy algorithm (see section A.1) to deduce the optimal sequence of actions by leveraging the Recurrent State Space Model (RSSM) world model.

However, it is important to note that the cross-entropy method in addition to being computationally expensive also exhibits scalability challenges, particularly in scenarios involving high-dimensional action spaces. Similar population-based methods are prevalent in the literature, but they share the same limitations.

To address these inherent shortcomings, we turn our attention to the gradient-based paradigm of Model Predictive Control (MPC) as an alternative approach.

4 Gradient based Planning

Online optimization methods can be broadly categorized into two distinct approaches. The first category is Gradient-Free Optimization, which operates without explicit directional information for optimization. Techniques such as Model Predictive Path Integral (MPPI) [36] and Cross-Entropy Optimization fall under this category. The second category is Gradient-Based Optimization, which leverages directional information to guide the optimization process.

Previous research in the domain of planning with world models has predominantly focused on the utilization of gradient-free optimization methods. However, real-world scenarios often involve actions that are high-dimensional, making it computationally infeasible to converge to an optimum using gradient-free optimization procedures. Additionally, these methods require significantly larger amounts of data for training the world model, which may not always be readily available in practical applications.

Gradient-Based Model Predictive Control (Grad-MPC) necessitates the establishment of an objective to assess the desirability of a particular state. This can be achieved through various means. In the context of standard Reinforcement Learning (RL), two primary approaches are employed: the use of a reward function and the utilization of a value function. The reward function provides the planner with immediate information regarding the desirability of a state, based on the returns assigned to that state by the environment. However, the reward function can exhibit short-sightedness, as it may not consider the desirability of states encountered along the trajectory from the current state to the end state. Therefore, in certain cases, a value function is employed, which captures the expected cumulative reward of the trajectory starting from a particular state and extending to the end. The definitions of the reward function and the value function for a given state are as follows:

$$r_t = R(s_t) \tag{4}$$

$$V(s_t) = E\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau\right] \tag{5}$$

Gradient-based planning commences with the generation of a set of action trajectories, each with a fixed length, drawn from a Gaussian distribution with zero mean and unit variance. This set of trajectories is sampled in consideration of the current state of the system. The initial state, in conjunction with the sampled actions, is then provided as input to the world model, which simulates future states based on the sequence of actions. Subsequently, the reward model or value model serves as a means to convey the desirability assessment for a given state back to the planner. Armed with this information, the planner employs gradient descent optimization to iteratively refine actions to maximize the expected reward.

This entire process is repeated iteratively over a few cycles to converge towards the optimal set of actions that lead to desirable states. The method is outlined in algorithm 1.

Algorithm 1 Planning with Grad-MPC

Input:
H Planning horizon distance
I Optimization iterations
J Candidates per iteration
$q(s_t|o_{\leq t}, a_{<t})$ Current state belief
$p(s_t|s_{t-1}, a_{t-1})$ Transition model
$p(r_t|s_t)$ Reward model
Initialize:
Actions candidates (J) are sampled $a_{t:t+H} \leftarrow \text{Normal}(0, 1)$.
for optimization iteration $i = 1..I$ **do**
  for candidate action sequence $j = 1..J$ **do**
    $s^{(j)}_{t:t+H+1} \sim q(st|o_{1:t}, a_{1:t-1}) \prod_{\tau=t+1}^{t+H+1} p(s_\tau|s_{\tau-1}, a^{(j)}_{\tau-1})$
    $R^{(j)} = \sum_{\tau=t+1}^{t+H+1} \mathbb{E}[p(r_\tau|s^{(j)}_\tau)]$
    $a^{(j)}_{t:t+H} = a^{(j)}_{t:t+H} - \nabla R^{(j)}$
  end for
end for
$J \leftarrow \text{argsort}(\{\sum_{\tau=1}^{H+1} R^{(\tau)}\}_{j=1}^{J})$
return $a_t^{J[0]}$.

Table 1: **DM-Control 100K Results**

	SAC Pixels	**CURL**	**PlaNet**	**Dreamer**	**Grad-MPC**
Cartpole	419 ± 40	597 ± 170	563 ± 73	326 ± 27	470 ± 55
Reacher Easy	145 ± 130	517 ± 113	82 ± 174	314 ± 155	663 ± 25
Finger Spin	166 ± 128	779 ± 108	560 ± 77	341 ± 70	660 ± 32
Walker Walk	42 ± 12	344 ± 132	221 ± 43	277 ± 12	237 ± 56
Cheetah Run	103 ± 38	307 ± 48	165 ± 123	235 ± 137	184 ± 81

5 Experiments

In our research, we employ PlaNet as the foundational world model for our experimentation. To enhance PlaNet's planning capabilities, we substitute its planning module with our custom gradient-based planner, Grad-MPC. PlaNet utilizes planning both during training and evaluation, we substitute CEM with Grad-MPC for both. In figure 3, we present a comparative analysis of the performance of our Grad-MPC approach against the results obtained from the Cross-Entropy and Policy Network methods on five Deep Mind Control [34] tasks: Cartpole Swingup, Reacher Easy, Finger Spin, Walker Walk, Cheetah Run.

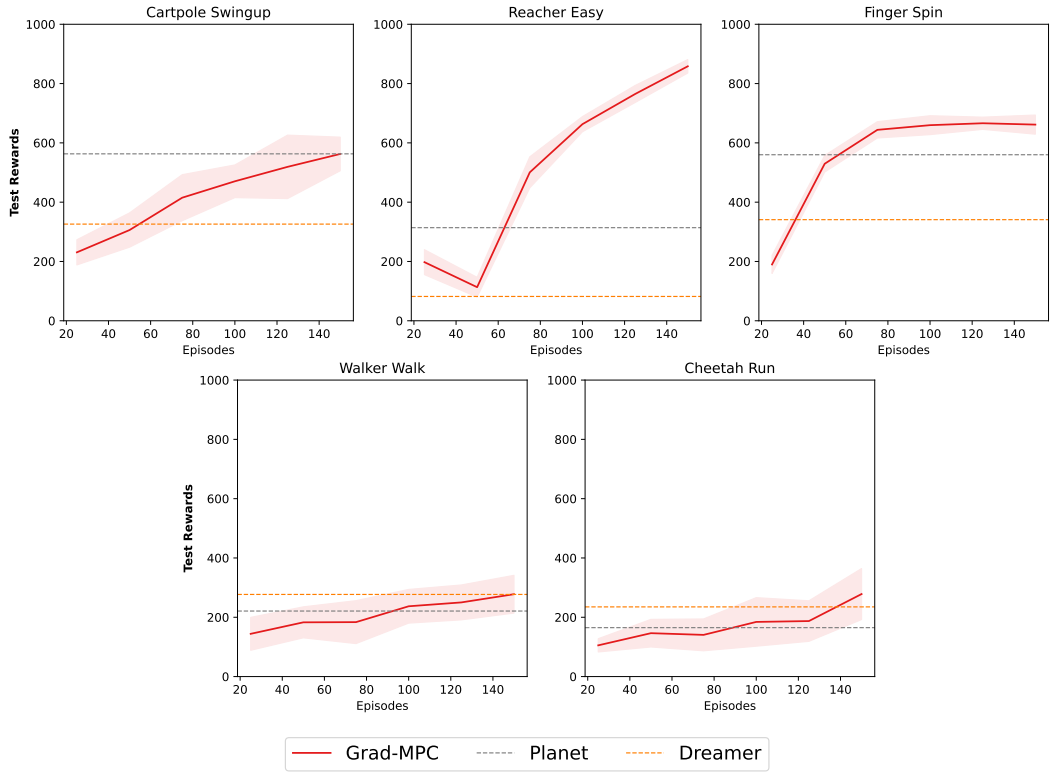


Figure 3: **Test Rewards of Grad-MPC in 150k env steps** These rewards are calculated over 10 test episodes across three random seeds. Dotted lines represent performace of Planet and Dreamer at 100K steps

When subjected to training for 100,000 steps across various tasks in DM Control, Grad-MPC demonstrates equivalent or superior performance in comparison to Cross-Entropy and Policy-based methods. It is vital to acknowledge that when addressing real-world tasks, data availability may be constrained. Hence, it becomes imperative to assess the efficacy of these methods in terms of sample efficiency.

Additionally, in table 1, we compare Grad-MPC's performance at 100,000 steps with four strong baselines consisting of both model-free and model based RL methods:

1. **Soft Actor-Critic [13]**: It is a model free RL method involving policy and action networks. We adopt pytorch code[37] for performance results.
2. **CURL [20]**: It is model based method that uses contrastive representation learning on image augmentations.
3. **PlaNet[15], Dreamer[14]**: Both are image reconstruction based representation learning methods.

Our findings reveal that Grad-MPC excels particularly well in handling simple tasks. We postulate that this effectiveness could stem from its ability to converge to optimal solutions more readily. This characteristic holds significant promise when constructing hierarchical models where complex tasks are decomposed into simpler sub-tasks and subsequently delegated to the planner. In such a scenario, Grad-MPC emerges as the optimal algorithm for low level planning, because for simpler goals the local optimum aligns with the global optimum.

6 Policy + gradient based MPC

Policy networks fall under the offline planning category. During training, policy networks learn with the assistance of a world model and value function and are then locked or frozen for use during testing. These policy networks are considered cutting-edge in model-based Reinforcement Learning

(RL) due to the remarkable memory capabilities of neural networks. However, as the environment becomes more complex, the accuracy of these networks tends to decrease. This is because even minor changes in the state distribution can result in significant errors, since even slight deviation from the training trajectories would result in states which the system has not encountered, thereby rendering policy networks inefficient [10, 32]

This situation becomes especially evident in sparse environments where accumulating errors may cause the system to miss a specific target, which is often the only rewarding state.

To address the errors associated with policy networks, we propose a hybrid planner. This hybrid planner leverages the memory capacity of policy networks and combines it with the precise planning abilities of gradient-based Model Predictive Control (MPC). We call this approach "Policy+grad-MPC."

The Policy+grad-MPC method operates in a manner similar to the grad-MPC method explained in previous sections. However, in this approach, trajectories are initialized from the output of the policy network.

In our experiments, we utilize the Dreamer model (see section A.2) as our foundation and replace the policy network with our custom hybrid planner. Dreamer uses the policy network $q_\phi(a_t|s_t)$ and value model $v_\psi(s_t)$ to infer the optimal actions instead of the reward model unlike PlaNet.

$$a_t^i = a_t^{i-1} - \alpha.\nabla V(s_t^{i-1}), i = 1..iters \tag{6}$$

The policy network and value model are learnt using the objectivesA.4.

Dreamer evaluates value estimate as mentioned in eq(2). It is essentially mix between immediate reward, value in imagined trajectory and value function.
We test our method in two sparse environments across 3 seeds utilizing the Dreamer Model pre-trained on 500,000 environment steps. Demonstrating superior performance compared to the pure policy-based approach of Dreamer.

Table 2: Performance in Sparse Environments

Env	Pure Policy(Mean rewards,σ)	Policy+MPC(Mean rewards,σ)
Ball in cup catch	608.5 ± 336.7	725.6 ± 237.3
Cartpole swingup sparse	639.5 ± 64.2	701.2 ± 40.3

7 Discussion and Future Work

Sub-Optimal Local Minima : Despite the successes of grad-MPC in sampling efficiency and scaling to high dimensional action spaces. Pure gradient based planning suffers from the problem of local minima. Hence if trained with enough data, policy networks eventually beat grad-MPC. Policy networks themselves might also fail to generalize for complex real world tasks,therefore they are not the complete solution either. We hypothesize that a hierarchical [21] method might hold the key. A hierarchical system in the style of director [16] wherein a complex goal is broken down into subgoals using a policy network and the resulting simpler goal could be solved by using grad-MPC.

Gradient based methods can further be enhanced with regularisation, consistency and robust world modelling techniques. Many other techniques can be performed on top or in conjuction with gradient based methods. Our paper demonstrates potential of this method.

References

[1] A. Argenson and G. Dulac-Arnold. Model-based offline planning. (arXiv:2008.05556), Mar. 2021. URL `http://arxiv.org/abs/2008.05556`. arXiv:2008.05556 [cs, eess, stat].

[2] K. Arulkumaran. Planet pytorch. `https://github.com/Kaixhin/PlaNet/`, 2021.

[3] A. Bardes, J. Ponce, and Y. LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[4] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013. doi: 10.1613/jair.3912. URL https://doi.org/10.1613%2Fjair.3912.

[5] H. Bharadhwaj, K. Xie, and F. Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. In *Learning for Dynamics and Control*, pages 277–286. PMLR, 2020.

[6] S. J. Bradtke, B. E. Ydstie, and A. G. Barto. Adaptive linear quadratic control using policy iteration. In *Proceedings of 1994 American Control Conference-ACC'94*, volume 3, pages 3475–3479. IEEE, 1994.

[7] C. Chen, Y.-F. Wu, J. Yoon, and S. Ahn. Transdreamer: Reinforcement learning with transformer world models, 2022.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[9] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[10] J. Farebrother, M. C. Machado, and M. Bowling. Generalization and regularization in dqn, 2020.

[11] Z. Guo, S. Thakoor, M. Pîslar, B. Avila Pires, F. Altché, C. Tallec, A. Saade, D. Calandriello, J.-B. Grill, Y. Tang, et al. Byol-explore: Exploration by bootstrapped prediction. *Advances in neural information processing systems*, 35:31855–31870, 2022.

[12] D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[13] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

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

[15] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[16] D. Hafner, K.-H. Lee, I. Fischer, and P. Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35:26091–26104, 2022.

[17] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[21] Y. LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62, 2022.

[22] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[23] V. Micheli, E. Alonso, and F. Fleuret. Transformers are sample efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[25] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers & chemical engineering*, 23(4-5):667–682, 1999.

[26] J. Robine, M. Höftmann, T. Uelwer, and S. Harmeling. Transformer-based world models are happy with 100k interactions. *arXiv preprint arXiv:2303.07109*, 2023.

[27] R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

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

[29] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. C. Courville, and P. Bachman. Data-efficient reinforcement learning with momentum predictive representations. *CoRR*, abs/2007.05929, 2020. URL `https://arxiv.org/abs/2007.05929`.

[30] Y. Seo, D. Hafner, H. Liu, F. Liu, S. James, K. Lee, and P. Abbeel. Masked world models for visual control. In *Conference on Robot Learning*, pages 1332–1344. PMLR, 2023.

[31] V. Sobal, J. SV, S. Jalagam, N. Carion, K. Cho, and Y. LeCun. Joint embedding predictive architectures focus on slow features. *arXiv preprint arXiv:2211.10831*, 2022.

[32] X. Song, Y. Jiang, S. Tu, Y. Du, and B. Neyshabur. Observational overfitting in reinforcement learning, 2019.

[33] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2:160–163, 1990. URL `https://api.semanticscholar.org/CorpusID:207162288`.

[34] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller. Deepmind control suite, 2018.

[35] Y. Urakami. Dreamer pytorch. `https://github.com/yusukeurakami/dreamer-pytorch`, 2022.

[36] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[37] D. Yarats. Soft actor-critic (sac) implementation in pytorch. `https://github.com/denisyarats/pytorch_sac`, 2019.

A Appendix

A.1 Cross - Entropy

The cross-entropy method, a population-based optimization technique, initiates by randomly sampling a set of actions from a Gaussian $\mathcal{N}(\mu, \Sigma)$, during each iteration n action trajectories are sampled, and the top k sequences with the highest reward (refer) are used to update the parameters of the gaussian, same procedure is repeated for m iterations.For i=1,2,...m ,The update equations are as follows.

$$\mu^i = \mu^{i-1} + mean[(a^{i-1}_{t:t+T-1})^k_{j=1}] \tag{7}$$

$$\Sigma^i = \Sigma^{i-1} + variance[(a^{i-1}_{t:t+T-1})^k_{j=1}]. \tag{8}$$

A.2 Model components of dreamer

Components of the dreamer model are as follows

$$Representation \rightarrow p_\theta(s_t|s_{t-1}, a_{t-1}, o_t)$$

$$Transition \rightarrow q_\theta(s_t|s_{t-1}, a_{t-1})$$

$$Reward \rightarrow q_\theta(r_t|s_t)$$

$$Valuemodel \rightarrow v_\psi(s_t)$$

$$Actionmodel \rightarrow q_\phi(a_t|s_t)$$

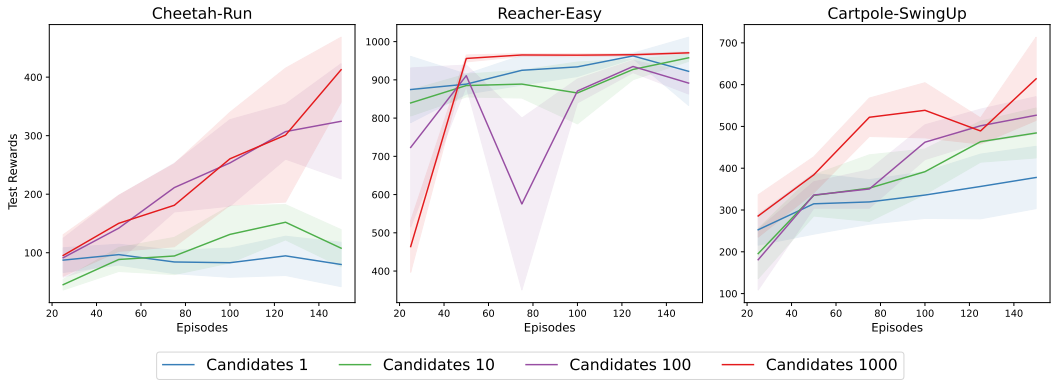


Figure 4: **Effect of number of Grad-MPC candidates(number of sampled trajectories) on performance for each environment(150 episodes=150k environment steps) across single seed**

A.3 Derivation

Assuming $p1 = p(s_{1:T}|a_{1:T})$ and $q1 = q(s_{1:T}|o_{1:t}, a_{1:T})$ and using jensens inequality.

$$\begin{aligned}
\ln p(o_{1:T}|a_{1:T}) &\geq E_{p1}\left[\ln \prod_{t=1}^{T} p(o_t|s_t)\right] \\
&= E_{q1}\left[\ln \prod_{t=1}^{T} \frac{p(o_t|s_t)p(s_t|s_{t-1}, a_{t-1})}{q(s_t|o_{\leq t}, a_{<t})}\right] \\
&= \sum_{t=1}^{T} \Big(E_{q(s_t|o_{\leq t}, a_{<t})}\left[\ln p(o_t|s_t)\right] \\
&\quad - E_{q(s_{t-1}|o_{\leq t-1}, a_{<t-1})}\left[KL\left[q(s_t|o_{\leq t}, a_t)\|p(s_t|s_{t-1}, a_{t-1})\right]\right]\Big) \qquad (9)
\end{aligned}$$

A.4 Dreamer Model

Training loss for the action model and the value function are defined as follows:

$$\text{PolicyLoss} \rightarrow \max_{\phi} \mathbb{E}_{q_\theta, q_\phi}\left[\sum_{\tau=t}^{t+H} V_\lambda(s_\tau)\right] \qquad (10)$$

$$\text{ValueLoss} \rightarrow \min_{\psi} \mathbb{E}_{q_\theta, q_\phi}\left[\sum_{\tau=t}^{t+H} \frac{1}{2}(v_\psi(s_\tau) - V_\lambda(s_\tau))\right]^2 \qquad (11)$$

$$V_k^N(s_\tau) = \mathbb{E}_{q_\theta, q_\phi}\left[\sum_{n=\tau}^{h-1} \gamma^{n-\tau} r_n + \gamma^{h-\tau} v_\psi(s_h)\right], \qquad (12)$$

$$\text{here } h = \min(\tau + k, t + H),$$

$$V_\lambda(s_\tau) = (1-\lambda)\left(\sum_{n=1}^{H-1} \lambda^{n-1} V_n^N(s_\tau)\right) + \lambda^{H-1} V_H^N(s_\tau). \qquad (13)$$

A.5 Rewards vs Candidates

We run experiments on test performance by varying number of candidiates across three different environments. We observe that more sampled trajectories lead to better test reward performance4.

A.6 Implementation Details

We use Pytorch implementation of PlaNet [2], it is distributed under MIT license. We also use Pytorch implementation of Dreamer [35], it is distributed under MIT license.

A.7 Hyperparameters

Table 3: Hyper-parameters and their default values for the PlaNet experiments.

Parameter	Value
Optimizer	Adam [18]
max-episode-length	1000
experience-size	1000000
activation-function	relu
embedding-size	1024
hidden-size	200
belief-size	200
state-size	30
exploration-noise	0.3
seed-episodes	5
collect-interval	100
batch-size	50
overshooting-distance	50
overshooting-kl-beta	0
overshooting-reward-scale	0
global-kl-beta	0
free-nats	3
bit-depth	5
learning-rate	1e-3
adam-epsilon	1e-4
grad-clip-norm	1000
planning-horizon	12
optimisation-iters	40
candidates	1000
action-learning-rate	0.1-0.01-0.005-0.0001

Table 4: Action Repeat values across environments.

Env	Action Repeat
cartpole swingup	8
reacher easy	4
finger spin	2
cheetah run	4
cup catch	6
walker walk	2

Table 5: Hyper-parameters and their default values for the Dreamer experiments.

Parameter	Value
Optimizer	Adam [18]
embedding-size	1024
hidden-size	400
belief-size	200
state-size	30
exploration-noise	0.3
overshooting-distance	50
overshooting-kl-beta	0
overshooting-reward-scale	0
global-kl-beta	0
free-nats	3
bit-depth	5
learning-rate	1e-3
adam-epsilon	1e-4
grad-clip-norm	1000
planning-horizon	1
candidates	1

A.8 DM Control Suite

Table 6: Difficulty and Action Dimension for Various Tasks

Task	Sparsity	Difficulty	Dim(A)
Cartpole Swingup	dense	Easy	1
Cup Catch	sparse	Easy	2
Finger Spin	dense	Easy	2
Walker Walk	dense	Easy	6
Cheetah Run	dense	Medium	6
Reacher Easy	dense	Medium	2
Cartpole Swingup Sparse	sparse	Medium	1