CLOSING THE TRAIN-TEST GAP IN WORLD MODELS FOR GRADIENT-BASED PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

World models paired with model predictive control (MPC) can be trained offline on large-scale datasets of expert trajectories and enable generalization to a wide range of tasks chosen at inference time. Compared to traditional MPC procedures, which rely either on slow search algorithms or on iteratively solving optimization problems exactly, gradient-based planning offers a computationally efficient alternative. However, the performance of gradient-based planning has thus far lagged behind that of other approaches. In this paper, we propose improved methods for training world models that enable efficient gradient-based planning. We begin with the observation that although a world model is trained on a next-state prediction objective, it is used at test-time to instead estimate a sequence of actions. The goal of our work is to close this train-test gap. To that end, we propose train-time data synthesis techniques that enable significantly improved gradient-based planning with existing world models. At test time, our approach outperforms or matches the classical gradient-free cross-entropy method (CEM) across a variety of object manipulation and navigation tasks in 10% of the time budget.

1 INTRODUCTION

In robotic tasks, anticipating how the actions of an agent affect the state of its environment is fundamental for both prediction (Finn et al., 2016) and planning (Mohanan and Salgoankar, 2018; Kavraki et al., 2002). Classical approaches derive models of the environment evolution analytically from first principles, relying on prior knowledge of the environment, the agent, and any uncertainty (Goldstein et al., 1950; Siciliano et al., 2009; Spong et al., 2020). In contrast, learning-based methods infer such models directly from data, enabling them to capture complex dynamics and thus improve generalization and robustness to uncertainty (Sutton et al., 1998; Schrittwieser et al., 2020; LeCun, 2022).

World models (Ha and Schmidhuber, 2018), in particular, have emerged as a powerful paradigm. Given the current state and an action, the world model predicts the resulting next state. These models can be learned either from exact state information (Sutton, 1991) or from high-dimensional sensory inputs such as images (Hafner et al., 2023). The latter setup is especially compelling as it enables perception, prediction, and control directly from raw images by leveraging pre-trained visual representations, and removes the need for measuring the precise environment states, which is difficult in practice (Assran et al., 2023; Bardes et al., 2024). Recently, world models and their predictive capabilities have been leveraged for planning, enabling agents to solve a variety of tasks (Hafner et al., 2019a;b; Schrittwieser et al., 2020; Hafner et al., 2023; Zhou et al., 2025). A model of the dynamics is learned offline, while the planning task is defined at inference as a constrained optimization problem: given the current state, find a sequence of actions that results in a state as close as possible to the target state. This inference-time optimization provides an effective alternative to reinforcement learning approaches (Sutton et al., 1998) that often suffer from poor sample efficiency.

World models are compatible with many model-based planning algorithms. The most commonly used planning methods are search-based, such as the Cross Entropy Method (CEM) (Rubinstein and Kroese, 2004) and Model Predictive Path Integral control (MPPI) (Williams et al., 2017a), because they have proven effective in practice. However, they are computationally intensive as they require iteratively sampling candidate solutions and performing world model rollouts to evaluate each one, a procedure that scales poorly in high-dimensional spaces. Gradient-based methods (SV et al., 2023),

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

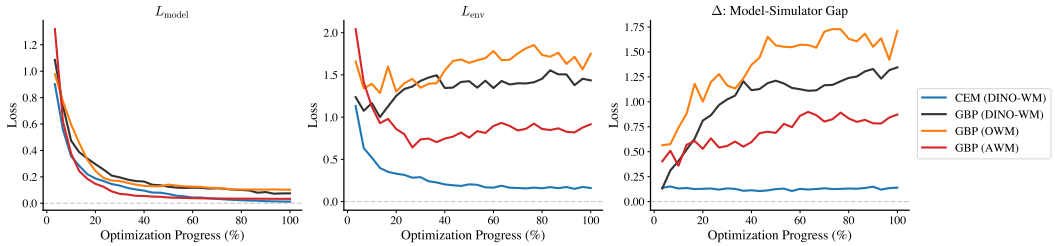


Figure 1: **Model-Simulator Gap.** We analyze the difference between the planning objective and the true simulator distance as a way of diagnosing the failure modes of GBP. In contrast to CEM that inadvertently optimizes simulator distance along with its planning objective, we see GBP's Model-Simulator Gap grow over the course of optimization. We use 50 unseen trajectories on PointMaze to measure these losses.

in contrast, avoid the limitations of sampling by directly exploiting the differentiability of world models to optimize actions end-to-end. These methods eliminate the costly rollouts required by search-based approaches, thus scaling more efficiently in high-dimensional spaces. Despite this promise, gradient-based approaches have thus far seen limited empirical success.

Gradient-based planning (GBP) suffers from a fundamental train-test gap. World models are typically trained using a next-state prediction objective on datasets of expert trajectories. At test time, however, they are used to optimize a planning objective over sequences of actions. We argue that this mismatch underlies the poor empirical performance of GBP, and we offer two hypotheses to explain why. 1) During planning, the intermediate sequence of actions explored by gradient descent drives the world model into states that were not encountered during training. In these out-of-distribution states, model errors compound, making the world model unreliable as a surrogate for optimization. 2) The action-level optimization landscape induced by the world model may be difficult to traverse, containing many poor local minima or flat regions, which hinders effective gradient-based optimization.

In this work, we address both of these challenges with two strategies: Online World Modeling and Adversarial World Modeling. Both expand the region of familiar latent states by continuously adding new trajectories to the dataset and finetuning the world model on them. To manage the distribution shift between offline expert trajectories and predicted trajectories from planning, Online World Modeling uses the environment simulator to correct states along a trajectory produced by performing GBP. Finetuning on these corrected trajectories ensures that the world model performs sufficiently well when GBP enters regimes of latent state space outside of the expert trajectory distribution. To overcome the difficulties of optimizing over a non-smooth loss surface during GBP, Adversarial World Modeling perturbs expert trajectories in the direction that maximizes the world model's loss. Adversarial finetuning smooths the induced action loss landscape, making it easier to optimize via gradient-based planning.

Our contributions are summarized as follows:

1. We identify the train-test gap in world models used for gradient-based planning, using the model-simulator gap as a diagnostic metric to explain why GBP exploits poor world model predictions.
2. We identify that DAgger-style finetuning and adversarial training are strong candidates to address issues specific to GBP, and adapt them using Online World Modeling (OWM) and Adversarial World Modeling (AWM).
3. We demonstrate empirically that AWM enables GBP to match or exceed the performance of the gradient-free Cross-Entropy Method across six simulated robotic/navigation tasks and a real-world high-dimensional egocentric pose planning benchmark, while operating in 10× less wall-clock time and establishing a new Pareto frontier for planning efficiency with latent world models.
4. We analyze aspects of GBP's failure modes to uncover why OWM is unable to improve GBP performance, revealing that data scale alone is not sufficient. Furthermore, this analysis provides evidence for why AWM significantly improves GBP performance on a variety of domains.

2 WHY GRADIENT-BASED PLANNING FAILS

2.1 PRELIMINARIES

World models learn environment dynamics by predicting the state resulting from taking an action in the current state. Then, at test time, the learned world model enables planning by simulating future trajectories and guiding action optimization. Formally, a world model approximates the (potentially unknown) dynamics function $h : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, where $\mathcal{S}$ denotes the state space and $\mathcal{A}$ the action space. The environment evolves according to $s_{t+1} = h(s_t, a_t)$ where $s_t \in \mathcal{S}, a_t \in \mathcal{A}$ denote the state and action at time t, respectively.

Latent world models. In practice, we typically do not have access to the exact state of the environment; instead, we only receive partial observations of it, such as images. In order for a world model to efficiently learn in the high-dimensional observation space $\mathcal{O}$, an embedding function $\Phi_\mu : \mathcal{O} \to \mathcal{Z}$ is employed to map observations to a lower-dimensional latent space $\mathcal{Z}$. Then, given an embedding function Φ_μ, our goal is to learn a latent world model $f_\theta : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$, such that

$$z_t = \Phi_\mu(o_t), \quad z_{t+1} = f_\theta(z_t, a_t), \quad \text{for all } t. \tag{1}$$

The choice of Φ_μ directly affects the expressivity of the latent world model. In this work, we use a fixed encoder pretrained with self-supervised learning that yields rich feature representations out of the box.

Training. To train a latent world model, we sample triplets of the form (o_t, a_t, o_{t+1}) from an offline dataset of trajectories $\mathcal{T}$ and minimize the ℓ_2 distance between the true next latent state $z_{t+1} = \Phi_\mu(o_{t+1})$ and the predicted next latent state $\hat{z}_{t+1}$. This procedure is represented by the following teacher-forcing objective:

$$\min_\theta \mathbb{E}_{(o_t, a_t, o_{t+1}) \sim \mathcal{T}} \| f_\theta(\Phi_\mu(o_t), a_t) - \Phi_\mu(o_{t+1}) \|_2^2. \tag{2}$$

Notably, we only minimize this objective with respect to the world model's parameters θ, not those of the potentially large embedding function.

Planning. At test-time, we use a learned world model to optimize candidate action sequences for reaching a goal state. By recursively applying the world model over an action sequence starting from an initial latent state, we obtain a predicted latent goal state and therefore the distance to the true goal state in latent space. This allows us to find the optimal action sequence

$$\{\hat{a}_t^*\}_{t=1}^H = \arg\min_{\{\hat{a}_t\}} \| \hat{z}_{H+1} - z_{\text{goal}} \|_2^2 \tag{3}$$

where $\hat{z}_{H+1}$ is produced by the recursive procedure

$$\hat{z}_2 = f_\theta(z_1, \hat{a}_1), \quad \hat{z}_{t+1} = f_\theta(\hat{z}_t, \hat{a}_t) \quad \text{for} \quad t > 1. \tag{4}$$

We use the function $\text{rollout}_f : \mathcal{Z} \times \mathcal{A}^H \to \mathcal{Z}^H$ to denote this recursive procedure.

Gradient-based planning (GBP) solves the planning objective (3) via gradient descent. Crucially, since the world model is differentiable, $\nabla_{\{\hat{a}_t\}} \hat{z}_{H+1} = \nabla_{\{\hat{a}_t\}} \text{rollout}_f(z_1, \{\hat{a}_t\})_{H+1}$ is well-defined. In contrast, the search-based CEM is gradient-free, but requires evaluating substantially more action sequences. We detail the full algorithms for GBP and CEM in Section C.2.

As errors can propagate over long horizons, Model Predictive Control (MPC) is commonly used to repeatedly re-plan by optimizing an H-step action sequence but executing only the first $K \leq H$ actions before replanning from the updated state.

2.2 GRADIENT-BASED PLANNING'S FAILURE MODES

Gradient-based planning minimizes a planning loss computed through the world model's predictions. Wherever the world model is imperfect, this loss does not reflect what would actually happen in the environment. We find that GBP exploits exactly these inaccuracies, converging to action sequences that achieve low planning loss in the model but fail in the real environment. To measure this, we define

the *model-simulator gap* $\Delta = \mathcal{L}_{\text{model}} - \mathcal{L}_{\text{env}}$, where L_{model} is the planning loss from Algorithm 3 and $\mathcal{L}_{\text{env}} = \|\Phi(s_{\text{final}}) - z_{\text{goal}}\|^2$ is the true distance to the goal obtained by encoding the simulator's final state with the visual encoder Φ. A large positive gap means the world model believes the plan is better than it actually is. In Figure 1, we see that although GBP appears to optimize $\mathcal{L}_{\text{model}}$ well, it completely fails to optimize $\mathcal{L}_{\text{env}}$, resulting in a large model-simulator gap. While GBP drives the action sequence into regions where the world model diverges from reality, CEM's sampling-based optimization appears more robust to model error, resulting in practically no model-simulator gap.

We hypothesize two reasons why this gap emerges specifically in GBP. First, because GBP is a first-order method, it is more likely to smoothly reach latent states outside the training distribution. In these latent states, the model-simulator gap can be large. Second, the action optimization itself could be unstable, resulting in certain local minima where the model-simulator gap could also be large. This instability likely arises from the fact that the world modeling objective does not encourage smooth input gradients, yet the world model is expected to effectively utilize gradients with respect to its input actions. We now present two finetuning methods designed to improve on these fronts.

3 Online and Adversarial World Modeling

3.1 Online World Modeling

During gradient-based planning, the action sequences being optimized are not constrained to lie within the distribution of behavior seen during training. World models are typically trained on fixed datasets of expert trajectories, whereas GBP selects actions solely to improve the planning objective, without regard to whether those actions resemble expert behavior. As a result, the optimization process often proposes action sequences that are out of distribution. Optimizing through learned models under such conditions is known to induce adversarial inputs (Szegedy et al., 2013; Goodfellow et al., 2014). In our setting, these adversarial action sequences drive the world model into regions of the latent state space that were rarely or never observed during training, causing large prediction errors. Even when errors are initially small, they accumulate as the planner rolls the model forward, ultimately degrading long-horizon planning performance.

A natural first approach to closing this gap is to directly expose the world model to the trajectories it encounters during planning. We call this **Online World Modeling**, which iteratively corrects the trajectories produced by GBP and finetunes the world model on the resulting rollouts. Rather than training solely on expert demonstrations, we repeatedly incorporate trajectories induced by the planner itself, thereby expanding the region of latent states that the world model can reliably predict.

First, we conduct GBP using the initial and goal latent states of an expert trajectory τ, yielding a sequence of predicted actions $\{\hat{a}_t\}_{t=1}^{H}$. These actions might send the world model into regions of the latent space that lie outside of the training distribution. To adjust for this, we obtain a *corrected trajectory*: the actual sequence of states that would result by executing the action sequence $\{\hat{a}_t\}_{t=1}^{H}$ in the environment using the true dynamics simulator h. We add the corrected trajectory,

$$\tau' = (z_1, \hat{a}_1, z'_2, \hat{a}_2, \ldots, z'_{H+1}), \tag{5}$$

to the dataset that the world model trains with every time the dataset is updated. Re-training on these corrected trajectories expands the training distribution to cover the regions of latent space induced by gradient-based planning, mitigating compounding prediction errors during planning. We provide more detail in Algorithm 1.

This procedure is a direct adaptation of DAgger (Dataset Aggregation) (Ross et al., 2011) to the world model setting. Just as DAgger iteratively retrains a policy on its own rollouts corrected by an expert, we retrain the world model on trajectories corrected by the ground-truth simulator. While intuitive, Online World Modeling requires access to the environment simulator during training, which is a major drawback that prevents its use in realistic settings.

3.2 Adversarial World Modeling

Since world models are only trained on the next-state prediction objective, there is no particular reason for their input gradients to be well-behaved. Adversarial training has been shown to result in better behaved input gradients (Mejia et al., 2019), consequently smoothing the input loss surface.

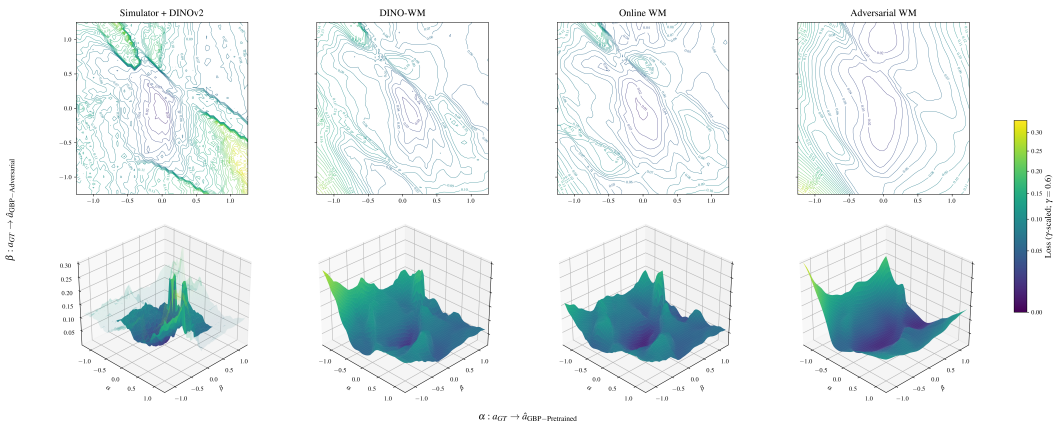


Figure 2: **Comparison of optimization landscapes.** We compare the actions optimization landscape of DINO-WM (Zhou et al., 2025) before and after finetuning with AWM and OWM on the PushT task. AWM yields a smoother landscape with a broader basin around the optimum, while OWM makes a negligible difference in the smoothness of the landscape. Additional details are in Section F.

Motivated by this observation, we propose an adversarial training objective that explicitly targets regions of the state-action space where the world model is expected to perform poorly. These adversarial samples may lie outside the expert trajectory distribution, which can expose the model to precisely the regions that matter for action optimization. We find that this procedure, which we call **Adversarial World Modeling**, does in fact smooth the loss surface of the planning objective (see Figure 2), improving the stability of action-sequence optimization.

Adversarial training improves model robustness by optimizing performance under worst-case perturbations (Madry et al., 2018). An adversarial example is generated by applying a perturbation δ to an input that maximally increases the model's loss. To train a world model on adversarial examples, we use the objective

$$\min_{\theta} \mathbb{E}_{(o_t, a_t, o_{t+1}) \sim \mathcal{T}} \left[\max_{\delta_a \in \mathcal{B}_a, \delta_z \in \mathcal{B}_z} \| f_\theta(\Phi_\mu(o_t) + \delta_z, a_t + \delta_a) - \Phi_\mu(o_{t+1}) \|_2^2 \right], \tag{6}$$

where $\mathcal{B}_a = \{\delta_a : \|\delta_a\|_\infty \leq \epsilon_a\}$ and $\mathcal{B}_z = \{\delta_z : \|\delta_z\|_\infty \leq \epsilon_z\}$ constrain the magnitude of perturbations for given ϵ_a, ϵ_z. Training on these adversarially perturbed trajectories provides an alternative method to Online World Modeling for surfacing states that may be encountered during planning, without relying on GBP rollouts. This is a significant advantage in settings where simulation is expensive or infeasible.

We generate adversarial latent states using the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014), which efficiently approximates the worst-case perturbations that maximize prediction error (Wong et al., 2020). Although stronger iterative attacks such as Projected Gradient Descent (PGD) can be used, we find that FGSM delivers comparable improvements in GBP performance while being significantly more computationally efficient (see Section G.1). This enables us to generate adversarial samples over entire large-scale offline imitation learning datasets.

For each state-action pair in a given minibatch, we look for small changes to the latent state or action that most increase the world model's prediction error. Let ϵ_a, ϵ_z denote the radius of the perturbation to the actions $\{a_t\}$ and latent states $\{z_t\}$ respectively. We compute gradients $\nabla_{\delta_a, \delta_z} \| f_\theta(z_t + \delta_z, a_t + \delta_a) - z_{t+1} \|_2^2$ with respect to the perturbations and take a signed gradient ascent step (i.e., in a direction that degrades the prediction) with step sizes $\alpha_a = 1.25\epsilon_a, \alpha_z = 1.25\epsilon_z$. We clip the result so that each entry of the perturbation stays within the radius. This procedure corresponds to a single step of a PGD-style attack, producing perturbations that lie on the edge of the allowed region where they are maximally challenging for the model. See Algorithm 2 for a detailed treatment.

To initialize the perturbation radii ϵ_a, ϵ_z, we use scaling factors λ_a, λ_z and find that Adversarial World Modeling is robust for $0 \leq \lambda_a \leq 1$ and $0 \leq \lambda_z \leq 0.5$. Furthermore, we find that fixing ϵ_a, ϵ_z to the standard deviation of the initial minibatch is stable across all experiments. Updating this estimate for

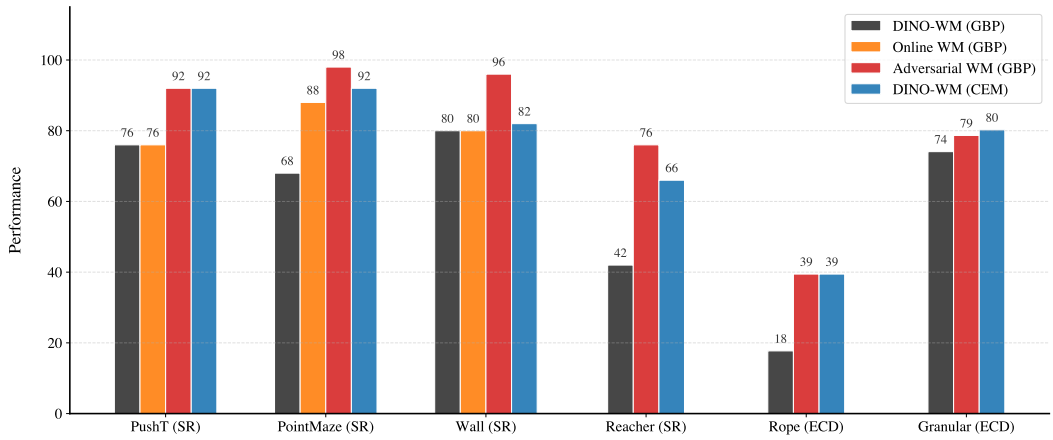


Figure 3: **Planning Performance on Simulation Tasks.** Comparison of world model planning performance across a variety of domains. For PushT, PointMaze, Wall, and Reacher, we report Success Rate (SR) as a percentage of task completed. For Rope and Granular we report Exponential Chamfer Distance (ECD) (Zhang et al., 2014) which is a measure of point-cloud similarity (larger is better). More details can be found in Section C.5.

each batch as in Algorithm 2 yields no consistent improvement in final planning performance. We further analyze design ablations in Section G.

4 Experiments

4.1 Experimental Setup

We evaluate on seven tasks spanning a range of planning domains. The six simulated environments include two navigation tasks (PointMaze, Wall), one rigid-body manipulation task (PushT), one continuous control task (Reacher), and two deformable-object manipulation tasks (Rope, Granular). As a real-world evaluation, we additionally evaluate on egocentric pose planning using the Nymeria dataset (Ma et al., 2024), where the agent plans a sequence of joint angle and translation actions to reach a target body pose from egocentric visual observations. Full task descriptions and dataset statistics are provided in Appendix Section C.1.

We use DINO-WM (Zhou et al., 2025) as our base world model across all tasks so that our experiments emulate planning within the latent space of a large-scale world model. The encoder is a frozen DINOv2 (Oquab et al., 2024) backbone and the transition model f_θ is implemented as a ViT (Dosovitskiy et al., 2021).

All planning results are reported under MPC, where an H-step action sequence is optimized but only the first K actions are kept before replanning from the updated state. For GBP, we use Adam (Kingma and Ba, 2014) throughout. To facilitate progress towards the goal during GBP, we set $\mathcal{L}_{\text{goal}}$ in Algorithm 3 to an alternate loss function: Weighted Goal Loss (WGL). Instead of the standard goal loss function that only minimizes the ℓ_2-distance between the final latent state produced by planning actions and the goal latent state, WGL encourages intermediate latent states to also be close to the goal latent state. WGL is detailed in Section C.4.

We compare the planning performance of DINO-WM with Online World Modeling (OWM) and Adversarial World Modeling (AWM). OWM requires simulator access during training and is therefore evaluated only on PushT, PointMaze, and Wall, where fast simulators are available. Results from these environments help us establish whether data coverage expansion improves GBP. AWM, which requires no simulator, is evaluated on all seven tasks. We report all finetuning, planning, and optimization hyperparameters in Table 2.

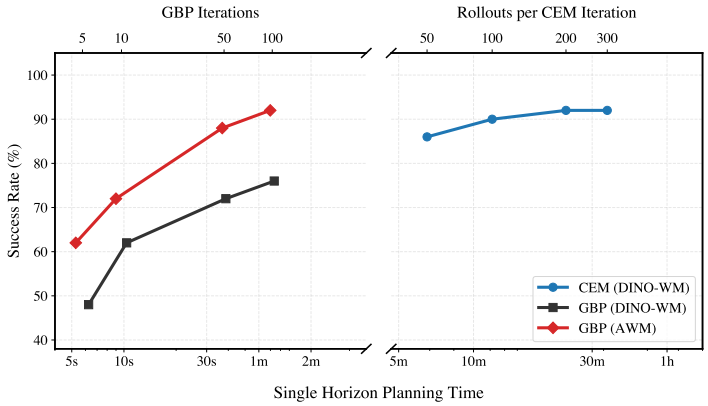


Figure 4: **Planning efficiency on PushT**. *Left:* GBP success rate as a function of wall-clock time controlled by the number of gradient descent iterations taken in a single MPC horizon. *Right:* CEM success rate as a function of wall-clock time controlled by the number of rollouts per CEM iteration. Note that the x-axis is logarithmic, highlighting that GBP operates in over an order of magnitude less wall-clock time than CEM.

4.2 Planning Results

As shown in Figure 3, AWM substantially improves over the DINO-WM GBP baseline across all six simulated tasks, with gains ranging from $+16\%$ on PushT and Wall to $+34\%$ on Reacher. Notably, AWM exceeds CEM on PointMaze (98% vs. 92%), Wall (96% vs. 82%), and Reacher (76% vs. 66%), and matches CEM on PushT (92%) and Rope (39%). On Granular, AWM comes within one point of CEM (79% vs. 80%). These results demonstrate that AWM enables GBP to consistently match or exceed CEM across a diverse range of planning domains, making it a viable method for world model planning.

We surprisingly find that OWM fails to support GBP. On PushT and Wall, OWM produces no improvement over the DINO-WM GBP baseline (76% and 80% respectively, identical to the baseline). On PointMaze, OWM improves over the baseline (88% vs. 68%) but falls short of both AWM (98%) and CEM (92%). In no environment does OWM match CEM. We analyze the reasons for OWM's poor performance in Section 4.4.

We also compare AWM to the latent space regularization method LeWM (Maes et al., 2026). We find that LeWM is unable to perform GBP as effectively as DINO-WM + AWM. Furthermore, finetuning LeWM with AWM improves its GBP performance, demonstrating the versatility of our method. Additional details can be found in Section E.

4.3 Planning Computational Efficiency

Deploying world model planning on real robotic systems requires planning that is both reliable and fast. While CEM has historically been the dominant planning algorithm because it fulfills the reliability aspect, GBP scales faster by design. Improving performance on GBP requires only more gradient descent iterations over a single world model rollout, whereas CEM must sample additional rollouts. This distinction severely compounds. Figure 4 illustrates this directly on PushT. On a single A100 GPU, GBP optimizes a single horizon of actions on the order of seconds to a couple of minutes, while CEM is on the order of a few minutes to over half an hour. AWM with GBP therefore establishes a new Pareto frontier for world model planning: it achieves performance that matches or exceeds CEM at over an order of magnitude lower wall-clock cost.

4.4 Overcoming GBP's failure modes

We study the effectiveness of OWM to determine if data coverage over latent states that are encountered during planning but might not have been seen during training can lead to gains in closing the

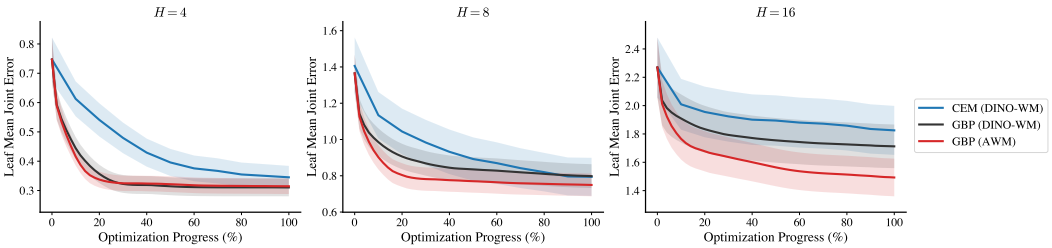


Figure 5: **Egocentric Body Pose Planning Performance.** We plot the Leaf Mean Joint Error (measure of distance between body poses) over the course of optimization for CEM and GBP with both DINO-WM and AWM. Additional details can be found in Section D.

model-simulator gap. As we see in Figure 3, OWM is inconsistent in improving GBP performance. Figure 1 demonstrates that OWM fails to decrease the model-simulator gap, which suggests that data coverage is not necessarily the underlying cause of the gap. The significance of this discovery is that compute and data scaling alone will not make GBP viable, and so we must turn towards the optimization process itself. Figure 2 showing OWM does not appear to change the smoothness of the actions optimization landscape only further motivates a need for a method that does. As expected, we see that AWM successfully smooths the loss landscape, getting rid of many local minima. We also see that AWM improves the model-simulator gap, suggesting that a smooth landscape is crucial for GBP to find faithful solutions.

4.5 Realistic Egocentric Pose Planning

One of the key disadvantages of CEM is the exponentially large search space that arises from high-dimensional actions. We further highlight the benefits of AWM improving GBP performance by evaluating on the task of future body pose prediction using real motion capture and egocentric video data. Specifically, we use the Nymeria Ma et al. (2024) dataset that features a 48 dimensional action space for control over an entire body pose. The details of the dataset can be found in Appendix Section D.

Figure 5 shows that GBP with AWM significantly improves Leaf Mean Joint Error (MJE) on longer horizons over GBP with DINO-WM. Notably, CEM with DINO-WM underperforms both GBP methods on the longest-horizon, indicating that GBP is more suitable for humanoid-style tasks that require long-horizon planning. Figure 7 provides visualizations justifying AWM's superior planning ability that allows it to achieve the most physically plausible goal poses.

5 Limitations

We conclude that OWM is an ineffective solution for improving GBP, however, it is important to reiterate that it requires a simulator to improve data coverage, a tool that is not available for realistic tasks. In Section G.2, we see that setting perturbation radii to be too large can degrade performance, and so AWM introduces new hyper-parameters that might require tuning aside from GBP's hyper-parameters. Although AWM achieves strong performance in long-horizon, realistic environments, it is not yet clear if AWM+GBP could run reliably on real hardware at a reasonable frequency.

6 Conclusion

In this work, we introduce *Online World Modeling* and *Adversarial World Modeling* as techniques for addressing the train-test gap that arises when world models trained on next-state prediction are used for iterative gradient-based planning. Across our experiments, these methods substantially improve the reliability of GBP and, in some settings, allow it to match or outperform sampling-based planners such as CEM. By narrowing this gap, our results suggest that gradient-based planning can be a practical alternative for planning with world models, particularly in settings where computational efficiency is critical. An important direction for future work is to implement these methods on

real-world hardware and take advantage of GBP's planning efficiency. Our work can make significant progress in the development of humanoid robots. Automating physical labor can have massive positive impact in many industries.

AI USE STATEMENT

We used generative AI tools to help with implementing our method and running experiments and baselines. We *have not* used generative AI tools for proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, or interpreting results, and helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and assisting with translation are not applicable to this work. We have reviewed all AI-assisted work. We read and verified the correctness of our AI-assisted implementations of our method, experiments, and baseline scripts. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

Adil Kaan Akan and F. Güney. Stretchbev: Stretching future instance prediction spatially and temporally. In *European Conference on Computer Vision*, 2022. URL https://api.semanticscholar.org/CorpusId:247749011.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics, 2025. URL https://arxiv.org/abs/2511.08544.

Adrien Bardes, Q. Garrido, Jean Ponce, Xinlei Chen, Michael G. Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *ArXiv*, abs/2404.08471, 2024. URL https://api.semanticscholar.org/CorpusId:269137489.

Homanga Bharadhwaj, Kevin Xie, and F. Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. *ArXiv*, abs/2004.08763, 2020. URL https://api.semanticscholar.org/CorpusId:215827996.

Oumayma Bounou, Jean Ponce, and Justin Carpentier. Online learning and control of complex dynamical systems from sensory input. *Advances in Neural Information Processing Systems*, 34: 27852–27864, 2021.

Siwei Chen, Yiqing Xu, Cunjun Yu, Linfeng Li, Xiao Ma, Zhongwen Xu, and David Hsu. Benchmarking deformable object manipulation with differentiable physics. *ArXiv*, abs/2210.13066, 2022. URL https://api.semanticscholar.org/CorpusId:257482484.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Katrina Drozdov, Ravid Shwartz-Ziv, and Yann LeCun. Video representation learning with joint-embedding predictive architectures. *ArXiv*, abs/2412.10925, 2024. URL https://api.semanticscholar.org/CorpusId:274777110.

Ashley D. Edwards, Himanshu Sahni, Yannick Schroecker, and C. Isbell. Imitating latent policies from observation. *ArXiv*, abs/1805.07914, 2018. URL https://api.semanticscholar.org/CorpusId:29156793.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793, 2016. URL https://api.semanticscholar.org/CorpusID:2780699.

Chelsea Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *ArXiv*, abs/1605.07157, 2016. URL https://arxiv.org/pdf/1605.07157.pdf.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pages 158–168. PMLR, 2022.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021. URL https://arxiv.org/abs/2004.07219.

Herbert Goldstein, Charles P Poole, and John Safko. *Classical mechanics*, volume 2. Addison-wesley Reading, MA, 1950.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2014.

Yanchen Guan, Haicheng Liao, Zhenning Li, Guohui Zhang, and Chengzhong Xu. World models for autonomous driving: An initial survey. *ArXiv*, abs/2403.02622, 2024. URL https://api.semanticscholar.org/CorpusId:268249117.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019b.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Mikael Henaff, William F Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2017.

Anthony Hu, Gianluca Corrado, Nicolas Griffiths, Zak Murez, Corina Gurau, Hudson Yeo, Alex Kendall, R. Cipolla, and J. Shotton. Model-based imitation learning for urban driving. *ArXiv*, abs/2210.07729, 2022. URL https://api.semanticscholar.org/CorpusId:252907712.

Matthew Jackson, Michael Matthews, Cong Lu, Benjamin Ellis, Shimon Whiteson, and J. Foerster. Policy-guided diffusion. *ArXiv*, abs/2404.06356, 2024. URL https://api.semanticscholar.org/CorpusId:269009692.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, K. Czechowski, D. Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, G. Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari. *ArXiv*, abs/1903.00374, 2019. URL https://api.semanticscholar.org/CorpusID:67856232.

Maximilian Karl, Maximilian Sölch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *ArXiv*, abs/1605.06432, 2016. URL https://api.semanticscholar.org/CorpusID:14992224.

Efstathios Karypidis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. Dino-foresight: Looking into the future with dino. *arXiv preprint arXiv:2412.11673*, 2024.

Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 2002.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Ian Lenz, Ross A Knepper, and Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, volume 10, page 25. Rome, Italy, 2015.

Lingni Ma, Yuting Ye, Fangzhou Hong, Vladimir Guzov, Yifeng Jiang, Rowan Postyeni, Luis Pesqueira, Alexander Gamino, Vijay Baiyya, Hyo Jin Kim, Kevin Bailey, David Soriano Fosas, C. Karen Liu, Ziwei Liu, Jakob Engel, Renzo De Nardi, and Richard Newcombe. Nymeria: A massive collection of multimodal egocentric daily motion in the wild. In *the 18th European Conference on Computer Vision (ECCV)*, 2024. URL https://arxiv.org/abs/2406.09905.

Alessandro Macchelli, Claudio Melchiorri, and Stefano Stramigioli. Port-based modeling and simulation of mechanical systems with rigid and flexible links. *IEEE transactions on robotics*, 25 (5):1016–1029, 2009.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2018.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels, 2026. URL https://arxiv.org/abs/2603.19312.

Felipe A Mejia, Paul Gamble, Zigfried Hampel-Arias, Michael Lomnitz, Nina Lopatina, Lucas Tindall, and Maria Alejandra Barrios. Robust or private? adversarial training makes models more vulnerable to privacy attacks. *arXiv preprint arXiv:1906.06449*, 2019.

MG Mohanan and Ambuja Salgoankar. A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, 100:171–185, 2018.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7559–7566. IEEE, 2018.

Anusha Nagabandi, K. Konolige, S. Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, 2019. URL https://arxiv.org/pdf/1909.11652.pdf.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2004.

Robin Schiewer, Anand Subramoney, and Laurenz Wiskott. Exploring the limits of hierarchical world models in reinforcement learning. *Scientific Reports*, 14, 2024. URL https://api.semanticscholar.org/CorpusId:270214472.

Michael D. Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324:81 – 85, 2009. URL https://doi.org/10.1126/science.1165893.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Archit Sharma, Shixiang Shane Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *ArXiv*, abs/1907.01657, 2019. URL https://api.semanticscholar.org/CorpusID:195791369.

Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks. *ArXiv*, abs/2205.02909, 2022. URL https://api.semanticscholar.org/CorpusID:248562698.

Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer, 2009.

Mark W Spong, Seth Hutchinson, and M Vidyasagar. Robot modeling and control. *John Wiley &amp*, 2020.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Jyothir SV, Siddhartha Jalagam, Yann LeCun, and Vlad Sobal. Gradient-based planning with world models. *arXiv preprint arXiv:2312.17227*, 2023.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018. URL https://arxiv.org/abs/1801.00690.

Tsun-Hsuan Wang, Pingchuan Ma, A. Spielberg, Zhou Xian, Hao Zhang, J. Tenenbaum, D. Rus, and Chuang Gan. Softzoo: A soft robot co-design benchmark for locomotion in diverse environments. *ArXiv*, abs/2303.09555, 2023. URL https://api.semanticscholar.org/CorpusId:257557557.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2): 344–357, 2017a.

Grady Williams, Nolan Wagener, Brian Goldfain, P. Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic mpc for model-based reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017b. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7989202.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

Jie Xu, Viktor Makoviychuk, Yashraj S. Narang, Fabio Ramos, W. Matusik, Animesh Garg, and M. Macklin. Accelerated policy learning with parallel differentiable simulation. *ArXiv*, abs/2204.07137, 2022. URL https://api.semanticscholar.org/CorpusId:248178090.

Xianyuan Zhan, Xiangyu Zhu, and Haoran Xu. Model-based offline planning with trajectory pruning. In *International Joint Conference on Artificial Intelligence*, 2021. URL https://api.semanticscholar.org/CorpusId:234742314.

Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning. In *International conference on machine learning*, pages 7444–7453. PMLR, 2019.

Weichen Zhang, Lifeng Shang, and Antoni B. Chan. A robust likelihood function for 3d human pose tracking. *IEEE Transactions on Image Processing*, 23(12):5374–5389, Dec 2014. doi: 10.1109/TIP.2014.2364113. PMID: 25347879.

Wenbo Zhang, Karl Schmeckpeper, P. Chaudhari, and Kostas Daniilidis. Deformable linear object prediction using locally linear latent dynamics. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13503–13509, 2021. URL https://api.semanticscholar.org/CorpusId:232380092.

Zongyuan Zhang, Tian dong Duan, Zheng Lin, Dong Huang, Zihan Fang, Zekai Sun, Ling Xiong, Hongbin Liang, Heming Cui, and Yong Cui. State-aware perturbation optimization for robust deep reinforcement learning. *ArXiv*, abs/2503.20613, 2025. URL https://api.semanticscholar.org/CorpusId:277322758.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2025.

A Related Work

Learning world models from sensory data. Learning-based dynamics models have become central to control and decision making, offering a data-driven alternative to classical approaches that rely on first principles modeling (Goldstein et al., 1950; Schmidt and Lipson, 2009; Macchelli et al., 2009). Early work focused on modeling dynamics in low-dimensional state-space (Deisenroth and Rasmussen, 2011; Lenz et al., 2015; Henaff et al., 2017; Sharma et al., 2019), while more recent methods learn directly from high-dimensional sensory inputs such as images. Pixel-space prediction methods (Finn et al., 2016; Kaiser et al., 2019) have shown success in applications such as human motion prediction (Finn et al., 2016), robotic manipulation (Finn and Levine, 2016; Agrawal et al., 2016; Zhang et al., 2019), and solving Atari games (Kaiser et al., 2019), but they remain computationally expensive due to the cost of image reconstructions. To address this, alternative approaches learn a compact latent representation where dynamics are modeled (Karl et al., 2016; Hafner et al., 2019b; Shi et al., 2022; Karypidis et al., 2024). These models are typically supervised either by decoding latent predictions to match ground truth observations (Edwards et al., 2018; Zhang et al., 2021; Bounou et al., 2021; Hu et al., 2022; Akan and Güney, 2022; Hafner et al., 2019b), or by using prediction objectives that operate directly in latent space, such as those in joint-embedding prediction architectures (JEPAs) (LeCun, 2022; Bardes et al., 2024; Drozdov et al., 2024; Guan et al., 2024; Zhou et al., 2025). Our work builds upon this latter category of world models and specifically leverages the DINOv2-based latent world models introduced in Zhou et al. (2025). However, unlike prior work that primarily targets improving general representation quality or prediction accuracy, we focus on enhancing the trainability of world models to improve the convergence and reliability of gradient-based planning.

Planning with world models. Planning with world models is challenging due to the non-linearity and non-convexity of the objective. Search-based methods such as CEM (Rubinstein and Kroese, 2004) and MPPI (Williams et al., 2017a) are widely used in this context (Williams et al., 2017b; Nagabandi et al., 2019; Hafner et al., 2019b; Zhan et al., 2021; Zhou et al., 2025). These methods explore the action space effectively, helping to escape from local minima, but typically scale poorly in high-dimensional settings due to their sampling-based nature. In contrast, gradient-based methods offer a more scalable alternative by exploiting the differentiability of the world model to optimize actions directly via backpropagation. Despite their efficiency, these methods suffer from local minima in highly non-smooth loss landscapes (Bharadhwaj et al., 2020; Xu et al., 2022; Chen et al., 2022; Wang et al., 2023), and gradient optimization can induce adversarial action sequences that exploit model inaccuracies (Schiewer et al., 2024; Jackson et al., 2024). Zhou et al. (2025) have observed that GBP is particularly brittle when used with world models built on pre-trained visual embeddings, such as DINOv2 (Oquab et al., 2024), often underperforming compared to CEM. To address these challenges, several stabilizing techniques have been proposed. For instance, random-sampling shooting helps mitigate adversarial trajectories by injecting noise in the action sequence and exploring a broader set of actions during trajectory optimization (Nagabandi et al., 2018), and Zhang et al. (2025) introduce adversarial attacks on learned policies to make them robust to environmental perturbations by selectively perturbing state inputs at inference time. In contrast, we apply perturbation directly to latent states and latent actions during world model training. Florence et al. (2022) add gradient penalties when training an implicit policy function to improve its smoothness and stabilize optimization, but their method does not involve training or using a world model. Other approaches aim to use a hybrid method that combines search and gradient steps to balance global exploration and local refinement (Bharadhwaj et al., 2020). In our work, we modify the world-model training procedure itself to improve GBP stability. In particular, through our Adversarial World Modeling approach, we enhance the robustness of the world model to perturbed states and actions, producing more stable and informative gradients that prevent adversarial action sequences at test time.

B OWM AND AWM ALGORITHMS

Algorithm 1 Online World Modeling (OWM)

Input: Pretrained world model f_θ, simulator dynamics function h, encoder Φ_μ, dataset of trajectories $\mathcal{T}$, online iterations N, horizon H, planning optimization iterations M
Initialize new trajectory dataset $\mathcal{T}'$
for $i = 1, \ldots, N$ **do**
  $\tau_i = (z_1, a_1, z_2, a_2, \ldots, a_H, z_{H+1}) \sim \mathcal{T}$
  $\{\hat{a}_t\}_{t=1}^{H} \leftarrow \text{GBP}(z_1, z_{H+1}, f_\theta, H, M)$
  $\{s'_t\}_{t=2}^{H+1} \leftarrow \text{rollout}_h(s_1, \{\hat{a}_t\})$
  $\{z'_t\}_{t=2}^{H+1} \leftarrow \{\Phi_\mu(s'_t)\}_{t=2}^{H+1}$
  $\tau'_i \leftarrow (z_1, \hat{a}_1, z'_2, \hat{a}_2, \ldots, \hat{a}_H, z'_{H+1})$
  $\mathcal{T}' \leftarrow \mathcal{T}' \cup \tau'_i$
  Train f_θ on next-state prediction using $\mathcal{T}'$
end for

Algorithm 2 Adversarial World Modeling (AWM)

Input: Pretrained world model f_θ, dataset of trajectories $\mathcal{T}$, action perturbation scaling λ_a, state perturbation scaling λ_z, horizon H, training iterations N, minibatch size B
Initialize new trajectory dataset $\mathcal{T}'$
for $i = 1, \ldots, N$ **do**
  Sample minibatch $\tau \leftarrow \{(z_t^j, a_t^j, z_{t+1}^j)_{t=1}^{H}\}_{j=1}^{B} \sim \mathcal{T}$
  $\epsilon_a \leftarrow \lambda_a \,\text{mean}_j\left[\text{std}(\{a_1^j, \ldots, a_H^j\})\right]$
  $\epsilon_z \leftarrow \lambda_z \,\text{mean}_j\left[\text{std}(\{z_1^j, \ldots, z_{H+1}^j\})\right]$
  $(\alpha_a, \alpha_z) \leftarrow (1.25\,\epsilon_a, 1.25\,\epsilon_z)$
  $\delta_a \sim \text{Uniform}(-\epsilon_a, \epsilon_a)$
  $\delta_z \sim \text{Uniform}(-\epsilon_z, \epsilon_z)$
  for $t = 1, \ldots, H$ **do**
    $\nabla_{\delta_a}, \nabla_{\delta_z} \leftarrow \nabla_{\delta_a, \delta_z} \left\| f_\theta(z_t + \delta_z, a_t + \delta_a) - z_{t+1} \right\|_2^2$
    $\delta_a \leftarrow \text{clip}(\delta_a + \alpha_a \,\text{sign}(\nabla_{\delta_a}), -\epsilon_a, \epsilon_a)$
    $\delta_z \leftarrow \text{clip}(\delta_z + \alpha_z \,\text{sign}(\nabla_{\delta_z}), -\epsilon_z, \epsilon_z)$
    $a'_t \leftarrow a_t + \delta_a$
    $z'_t \leftarrow z_t + \delta_z$
  end for
  $\tau' \leftarrow \{(z'^j_t, a'^j_t, z^j_{t+1})_{t=1}^{H}\}_{j=1}^{B}$
  Train f_θ on next-state prediction using τ'
end for

C EXPERIMENTAL DETAILS

C.1 SIMULATION TASK DETAILS

PushT: This task introduced by Chi et al. (2024) uses an agent interacting with a T-shaped block to guide both the agent and block from a randomly initialized state to a feasible goal state within 25 steps. We use the dataset of 18500 trajectories given in Zhou et al. (2025), in which the green anchor serves purely as a visual reference. We draw a goal state from one of the noisy expert trajectories at 25 steps from the starting state.

PointMaze: In this task introduced by Fu et al. (2021), a force-actuated ball which can move in the x, y Cartesian directions has to reach a target goal within a maze. We use the dataset of 2000 random trajectories present in Zhou et al. (2025), with a goal state chosen 25 steps from the starting state.

Wall: This task introduced by DINO-WM (Zhou et al., 2025) features a 2D navigation environment with two rooms separated by a wall with a door. The agent's task is to navigate from a randomized starting location in one room to a random goal state in the other room, passing through the door. We use the dataset of 1920 trajectories as provided in DINO-WM, with a goal state chosen 25 steps from the starting state.

Reacher: This is a continuous control task from the DeepMind Control Suite (Tassa et al., 2018), where a 2-joint robotic arm aims to reach a target position in a fixed 2D plane. We use a dataset of 10000 trajectories of 200 steps each from (Maes et al., 2026), that were generated using an expert Soft Actor-Critic policy.

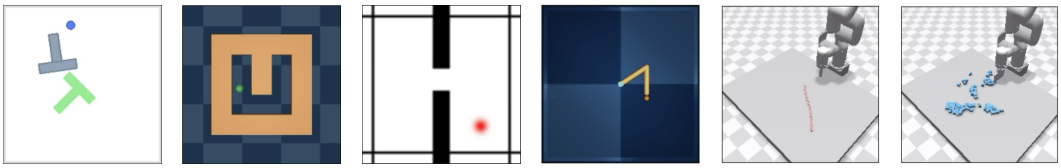
Rope: In this task introduced by Zhang et al. (2024) a simulated Xarm must push a piece of rope into the goal orientation. We use the dataset of 1000 trajectories of 20 steps each provided in DINO-WM.

Granular: In this task introduced by Zhang et al. (2024) a simulated Xarm must push around one hundred small particles into the goal configuration. We use the dataset of 1000 trajectories of 20 steps each provided in DINO-WM.

Each of the tasks' environment is illustrated in Figure 6. We reproduce the dataset statistics used to train the base world model for each environment from Zhou et al. (2025) as detailed in Table 1.

Table 1: Trajectory datasets used to pretrain the base DINO-WM.

Environment	History	Frameskip	Dataset Size	Trajectory Length
Push-T	3	5	18500	100-300
PointMaze	3	5	2000	100
Wall	1	5	1920	50
Reacher	3	5	10000	200
Rope	1	1	1000	5
Granular	1	1	1000	5



PushT PointMaze Wall Reacher Rope Granular

Figure 6: Illustrations of the six simulation tasks used in our experiments. Images from Zhou et al. (2025).

C.2 GBP AND CEM PLANNING ALGORITHMS

We detail the gradient-based and cross-entropy methods used in our planning experiments in Algorithm 3 and Algorithm 4.

Algorithm 3 Gradient-Based Planning via Gradient Descent

Input: Start state z_1, goal state z_{goal}, world model f_θ, horizon H, optimization iterations N
Initialize action prediction $\{\hat{a}_t\}_{t=1}^{H} \sim \mathcal{N}(0, I_H)$
for $i = 1, \ldots, N$ **do**
  $\hat{z}_{H+1} \leftarrow \text{rollout}_f(z_1, \{\hat{a}_t\})_{H+1}$
  $\mathcal{L}_{\text{goal}} \leftarrow \|\hat{z}_{H+1} - z_{\text{goal}}\|_2^2$
  $\{\hat{a}_t\} \leftarrow \{\hat{a}_t\} - \eta \cdot \nabla_{\{\hat{a}_t\}} \mathcal{L}_{\text{goal}}$
end for

Algorithm 4 Cross-Entropy Method (CEM) Planning

Input: Current observation o_0, goal observation o_g, encoder Φ_μ, world model f_θ, horizon H, population size N, top-K selection K, iterations I
$\hat{z}_1 \leftarrow \Phi_\mu(o_1)$
$z_g \leftarrow \Phi_\mu(o_g)$
Initialize Gaussian distribution parameters: mean μ_0, covariance Σ_0
for $i = 1, \ldots, I$ **do**
  Sample N action sequences $\{a^{(j)}_{1:H}\}_{j=1}^{N} \sim \mathcal{N}(\mu_{i-1}, \Sigma_{i-1})$
  for $j = 1, \ldots, N$ **do**
    $\hat{z}^{(j)}_1 \leftarrow \hat{z}_1$
    for $t = 2, \ldots, H+1$ **do**
      $\hat{z}^{(j)}_t \leftarrow f_\theta(\hat{z}^{(j)}_{t-1}, a^{(j)}_{t-1})$
    end for
    $C^{(j)} \leftarrow \|\hat{z}^{(j)}_{H+1} - z_g\|^2$
  end for
  Select K sequences with lowest cost: $\mathcal{E} = \{a^{(j)}\}_{\text{top-}K}$
  $\mu_i \leftarrow \frac{1}{K}\sum_{a\in\mathcal{E}} a$
  $\Sigma_i \leftarrow \frac{1}{K}\sum_{a\in\mathcal{E}}(a-\mu_i)(a-\mu_i)^\top$
end for
return μ_I as the final action sequence estimate$\{\hat{a}_t\}_{t=1}^{H}$

C.3 Finetuning and Planning Hyperparameters

In Table 2, we list all shared hyperparameters used in training and planning. For all planning experiments we use 10 MPC iterations, taking the first $K = 1$ actions at each iteration before replanning. For all simulator tasks, we use a horizon of $H = 5$ actions. We provide data quantity and synthetic data parameters for our Online and Adversarial World Modeling training setups in Tables 3 and 4 respectively. In addition to maintaining perturbation radii for the visual latent and action embeddings, we use a distinct radius for the proprioceptive embeddings. We empirically find that the scales of the visual and proprioceptive embeddings are incompatible and semantically distinct, thereby necessitating independent perturbation. Throughout all of our experiments, we set the perturbation radii of the action embedding and proprioceptive embedding identically for simplicity.

Table 2: We report **(a)** shared hyperparameters for OWM/AWM finetuning across all environments, **(b)** GBP parameters, and **(c)** CEM parameters.

(a) Finetuning Parameters

Name	Value
Image size	224
Optimizer	AdamW
Predictor LR	1e-5

(b) GBP

Name	Value
Optimizer steps	100
LR	0.2
Optimizer	Adam

(c) CEM

Name	Value
Iterations	30
Number of Elites	30
Rollouts per Iteration	300

Table 3: Finetuning parameters for Online World Modeling.

Environment	**# Rollouts**	**Batch Size**	**GPU**	**Epochs**
PushT	6000	32	4x B200	1
PointMaze	500	32	4x B200	1
Wall	1920 (all)	80	4x B200	1

Table 4: Training parameters for Adversarial World Modeling.

Environment	# Rollouts	Batch Size	GPU	Epochs	ϵ_{visual}	$\epsilon_{\text{proprio}}$	ϵ_{action}
PushT	20000 (all)	16	8x B200	1	0.05	0.02	0.02
PointMaze	2000 (all)	16	1x B200	1	0.20	0.08	0.08
Wall	1920 (all)	48	1x B200	2	0.20	0.08	0.08
Reacher	10000 (all)	32	1xA100	2	0.05	0.02	0.02
Rope	1000 (all)	16	1xRTX3080	1	0.05	0.02	0.02
Granular	1000 (all)	16	1xRTX3080	1	0.05	0.02	0.02

C.4 Weighted Goal Loss

We use an exponentially weighted goal loss to improve progress in navigation-based tasks. Weighted Goal Loss (WGL) minimizes the distance between all latent states in the predicted rollout and the goal latent state, but weighted by an exponential weighting. Formally,

$$\mathcal{L}_{\text{WGL}} = \frac{1}{H} \sum_{i=2}^{H+1} w_i \|\hat{z}_i - z_{\text{goal}}\|_2^2. \tag{7}$$

where the sequence of normalized weights $\{w_i\}_2^{H+1}$ is a hyperparameter choice. Empirically, we find that using this objective for Gradient-Based Planning either maintains or improves planning performance. For PointMaze and Wall, we found that exponentially upweighting later states in the planning horizon improved planning performance, so we set $w_i = 2^i$. For PushT, we found that exponentially upweighting earlier states improved planning performance, so we set $w_i = (1/2)^i$. We leave the optimal selection of this sequence of weights as future work.

C.5 Simulation Tasks Planning Results

In Table 5, we report mean planning performance across a total of 50 trajectories for each method and task. We calculate standard deviations using the averages of 3 batches of size 16. Exponential Chamfer Distance Zhang et al. (2014) is a bounded version (from 0 - 100) of the traditional chamfer distance, allowing us to measure the performance of Rope and Granular on the same scale as the other simulation tasks.

Table 5: Planning performance across simulation tasks and methods.

Environment	DINO-WM (GBP)	AWM (GBP)	OWM (GBP)	DINO-WM (CEM)
PushT (SR)	76 ± 9.31	92 ± 5.55	76 ± 8.63	92 ± 2.69
PointMaze (SR)	68 ± 10.52	98 ± 2.77	88 ± 4.65	92 ± 5.73
Wall (SR)	80 ± 12.02	96 ± 2.86	80 ± 11.79	82 ± 9.62
Reacher (SR)	42 ± 8.87	76 ± 4.13	-	66 ± 7.47
Granular (ECD)	74.08 ± 2.96	78.66 ± 2.36	-	80.25 ± 2.41
Rope (ECD)	17.73 ± 3.19	39.46 ± 4.34	-	39.46 ± 6.71

D Egocentric Body Pose Planning and Nymeria Dataset Details

Nymeria Ma et al. (2024) is a large scale dataset of first-person videos and associated motion capture. The motion capture at a particular timestep is represented by a pose p_t, which consists of the 3D position of the pelvis and Euler angles for each joint. An action is defined as the change between two subsequent poses, namely the change in pelvis position and the Euler angles. With 15 joints, the action becomes 48 dimensional. To measure the quality of predicted actions we use Leaf Mean Joint Error (MJE). MJE is defined as

$$\text{MJE} = \frac{1}{|\text{joints}|} \sum_{j \in \text{joints}} |\hat{x}_j - x_j|.$$

"Leaf" indicates that $\hat{x}$ only comes from the final predicted pose.

We train DINO-WM on 2 epochs of a split containing half the Nymeria dataset, and perform AWM finetuning for 1 epoch on the same split. For CEM planning, we use 128 rollouts per iteration, 16 elites, and 10 iterations. For GBP planning, we use Adam, LR of 0.2, and 50 optimizer steps.



Figure 7: **Visualization of Predicted Body Pose Rollouts.** *Left:* The initial observation. *Middle:* Sequence of 8 future body poses from 16 predicted action with each planning method. *Right:* The goal observation. For visualization, we label the pelvis ●, head ●, left-hand ●, and right-hand ●. In the top example, AWM not only reaches the window, but raises its right-hand correctly which CEM fails to do. In the bottom example, we see that AWM is the only method to get close to the gray sofa.

E Comparison to LeWorldModel

LeWorldModel (LeWM) Maes et al. (2026) is a latent world model that uses a SIGReg Balestriero and LeCun (2025) regularization term to prevent trivial collapse by enforcing Gaussian-distributed latent states. In Table 6 we show how LeWM alone underperforms DINO-WM with AWM, and that LeWM benefits from AWM as well. For each environment, we finetune LeWM with a single epoch of AWM.

Table 6: GBP performance on PushT and Wall of LeWM and DINO-WM with and without AWM. The same planning setup is used for each architecture.

Environment	DINO-WM	LeWM	LeWM + AWM	DINO-WM + AWM
PushT (SR)	76 ± 9.31	72 ± 6.33	82 ± 4.68	92 ± 5.55
Wall (SR)	80 ± 12.02	74 ± 4.53	80 ± 2.37	96 ± 2.86

F Visualizing the Optimization Landscape

We visualize the loss landscape of both the DINO World Model before and after applying the AWM and OWM objective. We perform a grid search over the subspace spanned by

1. $\hat{a}_{\text{GBP-Pretrained}}$: Gradient-Based Planning on original Dino World Model with 300 optimization steps of Adam with LR = 1e-3. We set a fixed initialization a_{init}.
2. $\hat{a}_{\text{GBP-Adversarial}}$: Gradient-Based Planning on our Adversarial World Model (or Online World Model) with 300 optimization steps of Adam with LR = 1e-3. We use the same a_{init} as our initialization.
3. a_{GT}: the ground-truth actions from the expert demonstrator.

We define the axes as $\alpha = \hat{a}_{\text{GBP-Pretrained}} - a_{\text{GT}}$ and $\beta = \hat{a}_{\text{GBP-Adversarial}} - a_{\text{GT}}$, and compute the loss surface over a 50×50 grid spanning $\alpha, \beta \in [-1.25, 1.25]$.

The "Simulator" landscape is obtained by executing all interpolated action sequences in the ground-truth environment and computing the embedding-space loss; it validates that the basin near the origin is indeed low-loss and that the overall landscape shape is consistent across models. Since the interpolated actions become physically implausible far from the origin, the simulator landscape exhibits edge artifacts at extreme (α, β) values. We reduce opacity beyond $|\alpha|, |\beta| > 0.75$ in the 3D surface plot for clarity.

G Adversarial World Modeling: Design Decisions

G.1 Fast Gradient Sign Method (FGSM) vs. Projected Gradient Descent (PGD)

Projected Gradient Descent (PGD) has been used as an iterative method for generating adversarial perturbations (Madry et al., 2018). At each step, PGD takes a gradient ascent step and projects the result onto the space of allowed perturbations (some ball with radius ϵ around the input). Projection (Π) is typically via clipping or scaling. Formally,

$$\delta^{(k+1)} = \Pi_{\|\delta\|_\infty \leq \epsilon} \left(\delta^{(k)} + \alpha \cdot \nabla_x \mathcal{L}(f_\theta(x + \delta^{(k)}), y) \right) \tag{8}$$

However, this is computationally expensive to use for adversarial training as it requires an additional backward pass for each iteration. If one uses a single-step, replaces the gradient by its sign, and uses step size $\alpha = \epsilon$, this recovers the well-known Fast Gradient Sign Method (FGSM) update (Goodfellow et al., 2014).

$$\delta = \epsilon \text{sign} \left(\nabla_x \mathcal{L}(f_\theta(x), y) \right). \tag{9}$$

In Wong et al. 2020, the authors demonstrate that initializing δ in the ℓ_∞-ball with radius ϵ and performing FGSM adversarial training on these perturbations substantially improves robustness to PGD attacks and matches performance of PGD-based training. We leverage this observation to perform cheap adversarial training that only requires $2\times$ the backward passes of traditional supervised learning. In comparison, K-step PGD requires K more backward passes ($3\times$ more for $K = 2$ and $4\times$ for $K = 3$). In Table 7, we show that 2/3-Step PGD does not consistently outperform FGSM, despite requiring a much larger training budget.

Table 7: Comparison between 2/3-Step PGD and FGSM demonstrating that FGSM has improved or only slightly subpar performance for significantly less training budget.

		PointMaze		**Wall**	
	Backward Passes	**Min/Epoch**	**GBP SR(%)**	**Min/Epoch**	**GBP SR(%)**
FGSM	**2**	**120**	94	**14**	**94**
2-Step PGD	3	165	**96**	20	90
3-Step PGD	4	201	94	24	94

G.2 Scaling Factor (λ) & Perturbation Radii (ϵ) Ablations

To assess the robustness of Adversarial World Modeling to the scaling factor and perturbation radius hyperparameters, we conduct an ablation study varying these two factors, shown in Figure 8. We evaluate $\lambda_a, \lambda_z \in [0.0, 0.02, 0.05, 0.20, 0.50, 1.0]^2$ and either fix $\epsilon_a, \epsilon_p, \epsilon_z$ to the standard deviation of the first minibatch ("Fixed") or recompute it for every minibatch ("Adaptive"). We observe no consistent improvement or degradation across any value of λ_a, for $0 \leq \lambda_z \leq 0.5$, or between the "Fixed" or "Adaptive" perturbation radii. We note that setting the visual scaling factor λ_z too high (e.g., $0.5, 1.0$) can significantly degrade performance. We hypothesize that excessively large perturbations distort the semantic content of the visual latent state, pushing it outside the range of semantically equivalent representations.

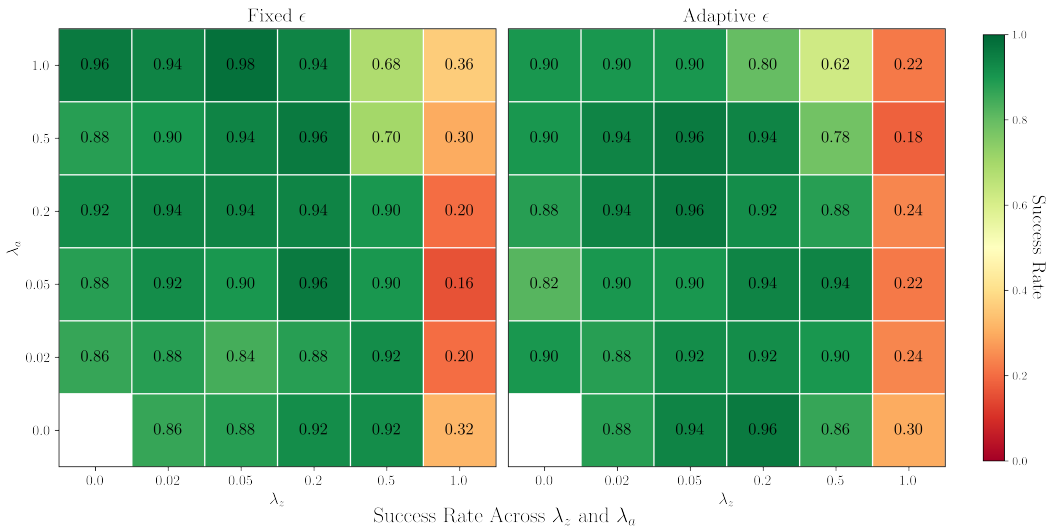


Figure 8: Success rate of GBP planning on an Adversarial World Model trained with scaling factors λ_a, λ_z and perturbation radii ϵ_a, ϵ_z on the Wall environment. We find that $0 \leq \lambda_z, \lambda_a \leq 0.2$ are stable for either "Fixed" or "Adaptive" perturbation radii.