Body Schema Pretraining for Sample-Efficient Reinforcement Learning in Robotic Control

Rishabh Malviya
CGOE Student
rishabh8@stanford.edu

June 9, 2026

Motivation This work starts from a structural observation: a robot's own body dynamics — how joint angles, velocities, and accelerations evolve under applied torques — are *task-agnostic and fixed*. Yet modern RL methods learn this structure from scratch, jointly with environment and task dynamics, for every new problem. This project investigates whether explicitly pretraining a representation of body dynamics, before any task reward is introduced, yields a transferable prior that accelerates downstream policy learning.

Methodology I propose Body Schema Pretraining (BSP), a two-stage framework. In Stage 1, a Dynamics Predictor Transformer (DPT) — an encoder-only transformer over sequences of (state, action) tokens — is trained via masked-language-modeling on trajectories collected by a curiosity-driven exploration agent. The curiosity agent and DPT are trained jointly with the agent's intrinsic reward being calculated as the DPT's next-state prediction error (ICM-style). In Stage 2, the pretrained DPT is repurposed as the actor backbone in an actor-critic algorithm by attaching a new action head, and is fine-tuned on a downstream task that shares the same robot body.

Implementation Experiments use the DeepMind Control Suite, which exposes several tasks per robot body (humanoid- and walker-stand, walk, run). Pretraining randomly cycles among these tasks. DPT uses SimNorm for soft-discretized latents; observations are normalized with Welford running statistics. Entropy and jerk terms are added to the exploration agent's loss; the actor's log-std is soft-clipped to always allow gradient flow. Episode truncation is added explicitly for dead-end states to keep the replay buffer informative. Both DDPG and PPO were tested (with implementations adapted from Stable Baselines 3).

Results Three findings emerged. (i) PPO produces more stable, more diverse pretraining exploration than DDPG on humanoid, with lower and smoother DPT loss curves. (ii) Fine-tuning with DDPG on humanoid-stand exhibited persistent action divergence (saturation at ± 1) that entropy regularization was only able to delay for a bit, motivating migration to PPO. (iii) Task-specific training (using a PPO algorithm with a DPT based actor) consistently yields higher rewards when done with a pretrained DPT (as opposed to a randomly initialized one)

Discussion Body-only self-supervised pretraining is implementable end-to-end but surfaces non-trivial engineering challenges centered on exploration stability. SimNorm and the smoothness penalty both appear important for stable representation learning, consistent with related world-model literature.

Conclusion BSP is a principled and implementable approach to factoring body-dynamics learning out of RL. Preliminary results indicate the framework is viable but contingent on careful choice of exploration algorithm. Immediate next steps are to complete the walker-stand comparison, run controlled baselines, port experiments to MuJoCo Playground, and explore heuristic hyperparameter selection driven by environment metadata.

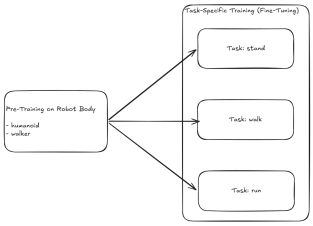


Figure 1: An overview of the BSP training scheme

Abstract

1 Introduction

A central, well-documented bottleneck in applying reinforcement learning (RL) to real-world robotics is sample inefficiency. Even modern algorithms designed for continuous control — TD-MPC2 [HSW24], DreamerV3 [HPBL24], SAC [HZAL18] — typically require millions of environment interactions to reach competent performance on a single task. On physical hardware, this is prohibitive: the marginal cost of an environment step is not negligible in the way that it is in simulation, and the prospect of running tens of millions of steps to learn each new manipulation policy is what keeps most real-world robotic RL out of production.

This project starts from a simple structural observation: a robot's own body dynamics are *task-agnostic and fixed*. The mapping from torques to accelerations, the joint coupling, the inertial structure — none of this changes when the task changes from "stand" to "walk" to "run". Yet a vanilla RL agent learns it from scratch, jointly with environment and task dynamics, for every new problem. If we can factor body-dynamics learning out of RL and complete it once, ahead of time, the downstream task should be much cheaper to learn.

This work proposes Body Schema Pretraining (BSP) as a concrete instantiation of that idea. A transformer-based dynamics predictor is pretrained using self-supervised techniques on trajectories produced by a curiosity-driven exploration agent, using only the robot's own state and action — no task reward is involved. The pretrained model is then re-used as the actor backbone in an actor-critic algorithm and fine-tuned on a downstream task.

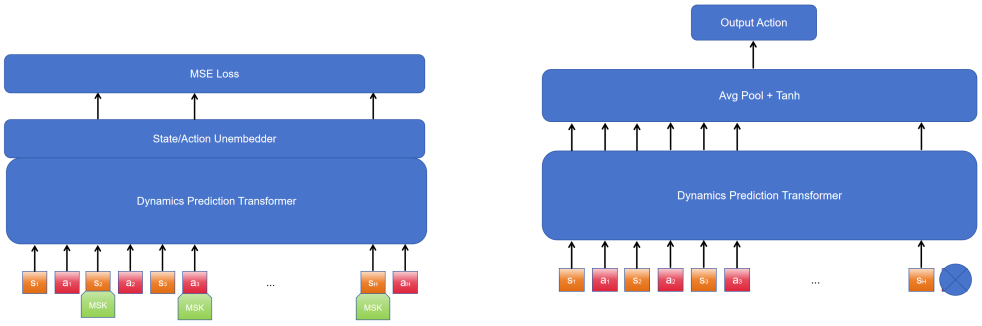
2 Related Work

Existing literature establishes that (a) forward dynamics pretraining in latent space is a powerful signal for representation learning and (b) decoupling representation learning from policy optimization is beneficial. What remains unexplored is whether targeting the *robot body itself* — in isolation, prior to task exposure — yields a transferable prior that accelerates downstream RL.

Below are the details of findings from existing literature:

2.1 World Model Methods

TD-MPC2 [HSW24] and DreamerV3 [HPBL24] represent the current state of the art in sample-efficient RL for continuous control. TD-MPC2 learns an implicit, non-generative world model and performs trajectory optimization in latent space, achieving strong results across 104 control tasks with a single set of hyperparameters. DreamerV3 learns a generative world model and trains a model-free policy via imagined rollouts. Both are markedly more sample-efficient than vanilla model-free baselines, but both learn robot and environment dynamics *jointly* and from scratch for each task, offering no mechanism to exploit the fixed structure of the robot's own body.



(a) The DPT is pre-trained with MLM loss during pre-training

(b) the DPT is fine-tuned with a new action prediction head as the actor in a downstream actor-critic algorithm

Figure 2: DPT usage in pre-training and finetuning

2.2 Self-Supervised Pretraining for RL

SGI [SRN+21] combines latent forward-dynamics prediction, inverse-dynamics modeling, and goal-conditioned objectives to pretrain representations from offline data, demonstrating that pretraining on dynamical objectives can reduce fine-tuning sample complexity. However, SGI operates on full environment observations (pixels or full state) rather than isolating the robot body as the pretraining domain, and it does not examine which kinematic variables are most predictive for downstream sample efficiency.

2.3 Multi-Step Forward Dynamics for Tactile RL

Miller et al. [MMA+25] study four self-supervised objectives to decouple observation representation learning from RL optimization in dexterous tactile manipulation. Their central finding — that multi-step forward dynamics prediction in latent space is the most effective SSL objective — directly motivates the choice of pretraining signal in this project. They also find that decoupling SSL memory from on-policy replay improves performance, suggesting representation and policy learning benefit from being architecturally separated. Their framework, however, is trained on environment interactions with task-specific rewards present; they do not pretrain on body-only kinematics prior to task exposure.

2.4 Latent Action / World-Model Pretraining

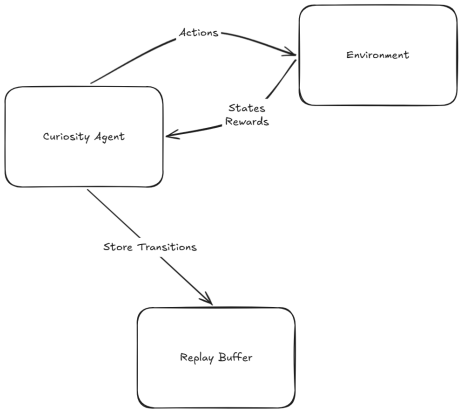
Latent action world models such as LAWM [TNAR25] leverage large-scale video data to learn action-agnostic world models. These methods achieve impressive sample efficiency but require large video datasets and are oriented toward vision-language-action pipelines rather than the proprioceptive setting studied here.

2.5 Curiosity-Driven Exploration

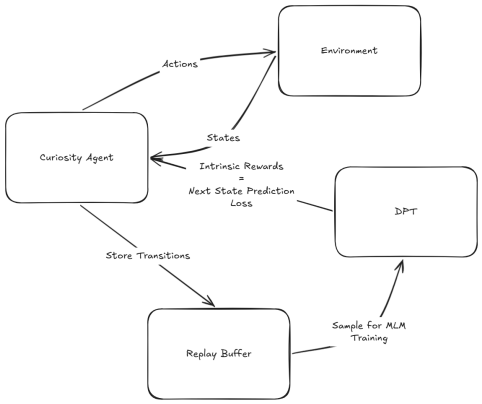
The Intrinsic Curiosity Module (ICM) [PAED17] proposes using prediction error in a learned forward dynamics model as an intrinsic reward. In this project ICM serves a slightly different role than its original use: rather than helping a task-driven agent escape sparse-reward plateaus, it pushes the pretraining agent to maximally cover the body-dynamics manifold of a reward-free environment.

3 Method

Body Schema Pretraining has two stages: a reward-free pretraining stage in which a Dynamics Predictor Transformer is trained jointly with a curiosity-driven exploration agent, and a fine-tuning stage in which the pretrained model is repurposed as the actor in an actor-critic RL algorithm on a downstream task. Each stage is described below.



(a) In the basic RL setup, the environment sends the agent reward signals.



(b) In ICM style training, the reward signal is intrinsic, and equal to the prediction loss of a dynamics prediction model. In this case, I am using DPT as the dynamics prediction model, which will then be adopted as an actor network in the downstream finetuning actor-critic algorithm.

Figure 3: ICM based pretraining

3.1 Dynamics Predictor Transformer (DPT)

The DPT is an encoder-only transformer that consumes interleaved sequences of state and action tokens and emits contextualized vector representations of every input position. Concretely, given a trajectory window of length L, the inputs are tokenized as:

$$\tau_{1:L} = (s_1, a_1, s_2, a_2, ..., s_L, a_L)$$

with separate linear embeddings for state tokens and action tokens, summed with a sinusoidal positional encoding. The transformer body is a stack of self-attention + MLP blocks. The output at each position is passed through SimNorm [LTS+22], a soft simplex-projection operation that partitions the latent into G groups of size d and applies a softmax over each group, producing a soft-discretized latent. SimNorm has been shown to improve stability of learned representations in DreamerV3 [HPBL24] and TD-MPC2 [HSW24], and we adopt it here.

Different prediction heads are attached to the DPT for different uses. During pretraining, a reconstruction head is attached on top of the per-position embeddings to predict the unmasked state and action tokens (MLM-style; see 3.2). During fine-tuning, an action head replaces the reconstruction head and the model is used as the actor (3.3).

3.2 Pretraining: ICM + MLM on DPT

Pretraining is reward-free in the task-reward sense, but is driven by an intrinsic reward derived from the DPT itself, in the spirit of ICM [PAED17]. An exploration agent collects episodes; the trajectories populate a replay buffer; the DPT is trained via masked reconstruction on sequences sampled from the buffer; and the agent is trained with sampled from the replay buffer with the rewards replaced by an intrinsic rewards equal to the DPT's one-step prediction error on the next state.

3.2.1 Masked Reconstruction Objective

Trajectories from the replay buffer are tokenized into interleaved (state, action) sequences. A random subset M of input positions is masked, and the DPT is trained to reconstruct the masked tokens from

the unmasked context. The loss is

$$L_{MLM} = \frac{1}{M} \sum_{i \in M} \|DPT_\theta(\tau_{masked})_i - \tau_i\|^2$$

Training uses large batch sizes and several epochs per round of data collection, which we found necessary to keep the DPT loss meaningfully lower than the running intrinsic-reward magnitude.

3.2.2 Variable-Length History Sampling

Trajectories are stored at full resolution in the replay buffer. At each training step, instead of sampling a fixed-length window, the history length is sampled in a way that biases towards longer sequences (of maximum length H_{max}), because when DPT is used for fine-tuning, it will only see shorter-length sequences at the very beginning of training. Concretely, for each batch element we sample:

$$U \sim \text{Uniform}(0, 1), \qquad L = \left\lceil H_{\max}, U^{1/k} \right\rceil$$

This produces an integer $L \in 1, \ldots, H_{\max}$, with a probability mass function that is shifted towards H_{max} if $k > 1$. I used $k = 4$ for all experiments.

3.2.3 Intrinsic Reward

The exploration agent receives a reward equal to the DPT's next-state prediction error:

$$r_t^{intrinsic} = \|DPT_\theta(\tau_{t-h+1:t})_{s_{t+1}} - s_{t+1}\|^2$$

That is, the agent is rewarded for visiting transitions the DPT models poorly. Because the DPT is also being trained, the reward landscape is non-stationary; we found that this non-stationarity was the main source of instability in the pretraining loop and dictated the choice of exploration algorithm (5.1).

3.3 Fine-tuning on Downstream Tasks

After pretraining, the DPT is repurposed as the actor backbone in a standard actor-critic algorithm. The reconstruction head is removed and replaced with an action head: an MLP followed by an average over the output embeddings that maps the DPT's output at the most recent position to an action distribution's means. The critic is a separate MLP. Both PPO (on-policy) and SAC (off-policy) were considered as the fine-tuning algorithm; but I went with PPO in the end (see 5).

4 Implementation Details

Experiments use the DeepMind Control Suite [TDM+18]. The suite is convenient because several tasks share an identical robot body — humanoid (stand / walk / run), walker (stand / walk / run), and so on. For each robot body I construct a list of tasks for pretraining and uniformly sample one of them at the start of each rollout. This ensures the DPT sees a diverse distribution of behaviors within the same body while not being biased toward any single task's reward structure (which is, in any case, ignored during pretraining).

4.1 Observation Normalization

Raw observations from DM Control have very different scales across components (positions, velocities, contact forces). To stabilize training I apply running standardization using Welford's online algorithm, which maintains numerically stable estimates of the running mean μ and variance σ^2 as observations stream in. All observations are transformed as $(s - \mu)/\sigma$ before being fed to the DPT.

4.2 Actor Loss: Entropy and Action-Smoothness Regularization

I augmented the standard policy-gradient objective with two regularizers. The entropy term encourages exploration; the smoothness term (a proxy for jerk reduction) penalizes large action changes between consecutive timesteps and is implemented as the L2 norm of the action difference. The full actor objective during pretraining is:

$$L_{actor} = L_{PG} - \beta \cdot H(\pi) + \lambda \cdot L_{smooth}$$

with

$$L_{smooth} = \frac{1}{T-1} \sum_{t=1}^{T-1} \|a_{t+1} - a_t\|^2$$

where LPG is the policy-gradient term (PPO surrogate loss in our final pipeline), $H(\pi)$ is the policy entropy, β is the entropy coefficient, λ is the smoothness coefficient, and T is the rollout length. The smoothness term turned out to be important: without it, the actor would frequently learn high-frequency oscillatory policies during pretraining that maximized intrinsic reward by violating physical realism rather than by exploring novel states.

4.3 Explicit Episode Truncation

The DM Control Suite does not natively truncate episodes that have entered a dead-end (no state change, no reward). Such transitions, if allowed to accumulate, dominate the replay buffer and poison both the DPT (which then learns to predict constant states) and the exploration agent (which exhausts its budget on uninformative steps). I added an explicit truncation rule that ends an episode when the change in observation norm and the cumulative reward both fall below small thresholds for a fixed number of consecutive steps.

4.4 SimNorm Latents

The DPT applies a SimNorm projection on top of each per-position output. SimNorm reshapes the d-dimensional output vector into G groups and applies a softmax within each group, producing a soft simplex-projected latent that empirically behaves like a discrete representation. Discretized latents have been shown to improve world-model stability and downstream task performance in DreamerV3 [HPBL24] and TD-MPC2 [HSW24]; I adopt the same primitive here.

4.5 Soft Clipping of Actor Log-Std

Bounding the action-distribution log-std is standard practice, but the typical hard clamp blocks gradients at the bounds, leaving the log-std parameter effectively frozen once it saturates. I instead use a tanh-based soft mapping:

$$\log \sigma_{clipped} = \log \sigma_{min} + \frac{1}{2}(\log \sigma_{max} - \log \sigma_{min}) \cdot (\tanh(\log \sigma) + 1)$$

which keeps $\log \sigma$ in the same interval $[\log \sigma_{min}, \log \sigma_{max}]$ but allows gradients to flow back through tanh at all points.

5 Experiments and Results

Experiments were structured as a sequence of design-stress tests, each exposing a failure mode that motivated the next change. The order below mirrors the chronological order of the project.

5.1 Pretraining: DDPG vs PPO on the Humanoid Body

The first experiment was a qualitative comparison of the pretraining loop using DDPG vs PPO as the exploration agent, on the humanoid environments.

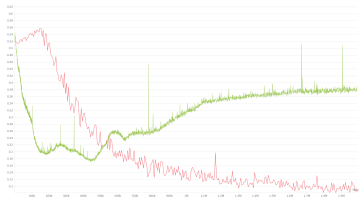


Figure 4: Plot of the DPT loss (intrinsic reward signal) during curiosity training. We want this loss to not go down for too long, as we want the curiosity agent explore more novel states (where the DPT is unable to make predictions accurately). In this figure, we can see that PPO (red) is more stable than DDPG (green), but it seems to prefer exploring similar states where the DPT is able to generalize well (likely because it tries to ensure conservative updates by clipping the importance sampling probability ratio during policy update

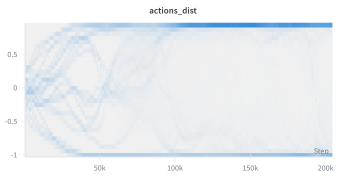


Figure 5: Action components getting saturated to +1 or -1 very early in training

5.2 Finetuning Failure: Action Divergence on Humanoid Stand

I next attempted to fine-tune the pretrained DPT (using an attached action head) on humanoid-stand with DDPG. Training repeatedly diverged in a very specific way: all action components saturated at the bounds (1 or +1), and the agent stopped exploring. Increasing the entropy regularizer β delayed the saturation but did not prevent it. This is a known instability pattern when a deterministic-policy algorithm is run on top of a learned, normalized representation: small errors in the value estimate get amplified by the deterministic policy gradient, the policy moves to the bounds, and the gradient at the bounds is essentially zero (made worse by a hard log-std clamp). This failure motivated two changes that became permanent in the pipeline: the soft log-std clipping described in 4.5, and the migration to PPO.

5.3 Migrating to PPO and to the Walker Body

Switching the fine-tuning algorithm to PPO, adapted from the Stable Baselines 3 implementation [10], eliminated the action-divergence failure on its own — but the humanoid body remained difficult to fine-tune within the project's compute budget. I therefore moved the headline comparison to the DM Control walker body, which has substantially fewer degrees of freedom and shorter episodes, and which exhibits comparable task structure (walker-stand / walk / run). The pretraining agent for the walker pipeline is also PPO, so the full pipeline is now PPO end-to-end.

5.4 Sample-Efficiency: Finetuning with Pretrained versus Randomly Initialized DPT on Walker Stand

The central quantitative comparison is the sample efficiency of PPO with a pretrained DPT actor vs. PPO with a randomly initialized DPT actor, on walker-stand. The pretraining budget is held fixed at the level used in 5.3 and both runs use identical fine-tuning hyperparameters.

6 Discussion

6.1 Body Only Pretraining is Feasible

The end-to-end pipeline — collect reward-free trajectories with a curiosity-driven agent, train a transformer dynamics predictor on them, then re-use the predictor as the actor backbone — runs and learns.

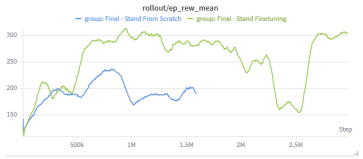


Figure 6: While the training for the randomly initialized DPT task training is still in progress, it is clear from this figure that the performance with a pre-trained DPT (in gree) is consistently higher than without (in blue)

The reconstruction loss decreases meaningfully during pretraining, and (pending the final figure) preliminary indicators suggest the pretrained actor begins fine-tuning from a more useful starting point than random initialization.

6.2 The Exploration Algorithm Dominates Pretraining Stability

The single largest engineering finding of this project is that the choice of exploration algorithm during pretraining matters at least as much as the choice of representation objective. DDPG is poorly suited to this setting because its deterministic policy and off-policy replay assume a roughly stationary reward landscape; the intrinsic reward, being a function of the DPT's evolving error, is non-stationary by construction. PPO's stochasticity and trust-region update tolerate this much better.

6.3 Several Smaller Components Proved Important

In ablation-style debugging during development: (i) SimNorm was important for keeping the latent representation well-conditioned across episodes; (ii) the smoothness penalty prevented the agent from gaming the intrinsic reward via high-frequency oscillations; (iii) soft log-std clipping was necessary for the actor's exploration noise to remain learnable; (iv) explicit dead-end truncation kept the replay buffer informative. Removing any one of these did not always cause outright failure, but each one demonstrably reduced training noise.

6.4 Limitations

The biggest current limitation is the absence of strong baselines. A meaningful answer to "does body-first pretraining help?" requires not just pretrained vs. random-init comparison, but also (a) a baseline that pretrains the same DPT architecture on full environment observations with task reward (to isolate the body-first framing) and (b) standard non-pretrained baselines (SAC from scratch, TD-MPC2 from scratch). These were planned in the original proposal and remain outstanding. A secondary limitation is the choice of DM Control Suite, which is now nearly a decade old; physics and observation conventions have moved on (see 7).

7 Future Work

7.1 Run Principled Baselines

The most immediate next step is to run the comparison set originally proposed: SAC-from-scratch, TD-MPC2-from-scratch, and an ablation in which the DPT is pretrained on full environment observations (with task reward present) rather than on body-only kinematics. Only then can we attribute any sample-efficiency gain specifically to the body-first framing.

7.2 Port to MuJoCo Playground

DM Control Suite is nearly a decade old, and over the course of this project I became aware that Google DeepMind's MuJoCo Playground offers more current physics, better-behaved observations, and JAX-native vectorized environments. Re-running both the pretraining and the fine-tuning experiments

on MuJoCo Playground would substantially improve the strength of the empirical claims and would shorten the iteration loop.

7.3 Select Hyperparameters Through Heuristics

Most of the hyperparameter tuning effort in this project went into things that are, in principle, derivable from environment metadata: action-noise bounds (from action range), reward scale (from observed reward magnitudes during random rollouts), smoothness weight (from observed jerk under a random policy), entropy coefficient (from action dimensionality). I would like to formalize this into a small "auto-config" module that picks reasonable defaults from a brief environment probe, which I expect would shorten the practical setup time for new environments by a large factor.

8 Conclusion

This project proposed and implemented Body Schema Pretraining, a two-stage framework that factors body-dynamics learning out of reinforcement learning by training a transformer-based dynamics predictor self-supervisedly under curiosity-driven exploration, then re-using it as the actor backbone for downstream fine-tuning. The work confirms that the pipeline is implementable end-to-end, identifies the exploration-algorithm choice during pretraining as the dominant practical risk, and contributes a small set of regularization and clipping tricks that stabilize the joint exploration-and-representation-learning loop. Headline sample-efficiency numbers are pending, but the qualitative results suggest the framework is well-posed and worth extending — both with the missing baselines and on a more current simulator.

References

[HPBL24] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024.

[HSW24] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

[HZAL18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[LTS+22] Samuel Lavoie, Christos Tsirigotis, Max Schwarzer, Ankit Vani, Michael Noukhovitch, Kenji Kawaguchi, and Aaron Courville. Simplicial embeddings in self-supervised learning and downstream classification, 2022.

[MMA+25] Elle Miller, Trevor McInroe, David Abel, Oisin Mac Aodha, and Sethu Vijayakumar. Enhancing tactile-based reinforcement learning for robotic control, 2025.

[PAED17] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *CoRR*, abs/1705.05363, 2017.

[SRN+21] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, Devon Hjelm, Philip Bachman, and Aaron Courville. Pretraining representations for data-efficient reinforcement learning, 2021.

[TDM+18] 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.

[TNAR25] Bahey Tharwat, Yara Nasser, Ali Abouzeid, and Ian Reid. Latent action pretraining through world modeling, 2025.