# LATENT CRITIC, PIXEL POLICY: GOAL-CONDITIONED CONTROL THAT REUSES A FROZEN WORLD MODEL ONLY IN TRAINING

**Anonymous authors**
Paper under double-blind review

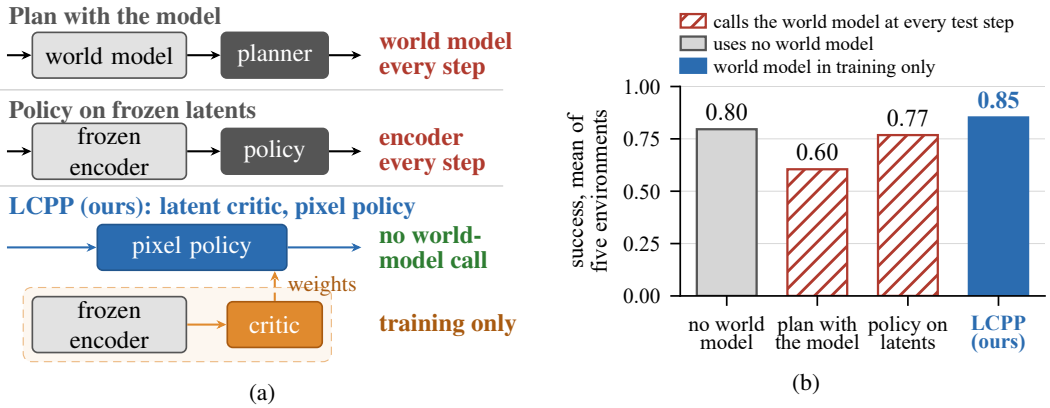



Figure 1: (a) Three places where a given frozen world model can enter an offline goal-conditioned agent, marked with its use at every deployed step. (b) Success of each placement as the mean over the five columns of Table 1 of the best row per group, with planning rows RC-aux, ACID and AdaJEPA and latent-policy rows SMWM and the HIQL head. Reusing the frozen world model at test time or as the policy input stays below the best pixel agents trained from scratch, whereas LCPP, using it only in the critic, exceeds them. Planning rows are tuned on validation goals (Appendix C).

## ABSTRACT

In offline goal-conditioned reinforcement learning, a learned value weights the imitation of dataset actions, so it must tell which of two nearby frames lies closer to a goal. Pretrained latent world models are released as frozen checkpoints, which current control methods query at test time to plan over predicted futures. We propose LCPP, which instead uses the frozen world-model encoder in a critic, a value on its latents from which advantage-weighted regression extracts a pixel policy that makes no world-model call. An analysis bounds the reliability of advantage weights by value errors at a state and its successor, including a floor from a frozen input map. At 100k updates on five environments, we compare LCPP with seven from-scratch methods and five methods that reuse the same frozen world model. LCPP is best or tied best in all five columns. In place of the latent value, a pixel value lowers Cube and Scene significantly across 20 seeds per arm, while a latent actor lowers them in each of five seeds. Code and demo are available at `anonymous.4open.science/r/lcpp`.

## 1 INTRODUCTION

Offline goal-conditioned reinforcement learning turns reward-free trajectories into a goal-reaching policy by relabeling reached states as goals and fitting a value to them (Andrychowicz et al., 2017). The value then acts as the critic, because its advantage weights decide how strongly the policy imitates each dataset action (Kostrikov et al., 2022). To do so, it has to rank a state against its successor, asking which of the two frames lies closer in time to the goal. A value trained on pixels



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.

learns this ranking from a sparse goal-reaching reward, with an encoder that starts from random weights. Pretrained latent world models are released as frozen checkpoints whose latents already support planning toward image goals in navigation and manipulation (Maes et al., 2026; Zhou et al., 2025). Such a checkpoint is a given resource, so the open question is where it enters an offline agent rather than how it was pretrained. We therefore treat judging progress and producing actions as two jobs that need not share an input, since the value serves training alone.

Methods that reuse frozen world models keep them at test time, planning over predicted futures with a latent distance as the cost (Terver et al., 2026). Adaptation methods revise that cost with an inverse-dynamics verifier or update selected model layers online, still acting through the predictor (Seo et al., 2026; Wang et al., 2026). Reachability objectives reshape the latent space through gradients into the encoder, which changes the pretrained model instead of reusing it (Li et al., 2026). On our five environments, reusing the frozen model at test time or as the policy input trails the best pixel agents trained from scratch (Figure 1b). Offline goal-conditioned agents learn value and policy from the same early-fused pixels through encoders trained from scratch (Park et al., 2025). Pretrained visual representations supply rewards for robot control, fixing the value signal instead of fitting a value in their space (Ma et al., 2023b;a). Asymmetric actor-critic training gives the critic inputs the actor lacks, relying on simulator state that offline image datasets do not record (Pinto et al., 2018).

A goal-conditioned value depends on the state and the full goal, so any fixed map in front of it must keep every distinction that changes the value (Schaul et al., 2015). Under a discounted goal-reaching reward, the value gain of one correct step decays geometrically with the remaining distance to the goal (Park et al., 2023). Extraction therefore needs a value whose errors at a state and its successor stay below that gain at the evaluated goal distances. The deployed policy still has to resolve fine visual detail, which offline pixel agents obtain by fusing state and goal images early. A method for this purpose would place a given pretrained encoder in the value alone and keep the policy on pixels.

We propose Latent Critic, Pixel Policy (LCPP), which places the frozen encoder of LeWorldModel (LeWM) in the value alone and learns the policy on pixels (Figure 1a). Its value reads standardized latents of state and goal and is fitted by expectile regression toward bootstrapped targets. Its actor reads the early-fused state and goal images and imitates dataset actions with weights computed from that value. The world model enters training alone, so the deployed policy keeps the input, network and per-step cost of a pixel agent. LCPP is best or tied best in all five columns of the main comparison, with its lead over two baselines holding over 20 training seeds. Swapping in a pixel value, a latent actor or an untrained encoder lowers success in three to five columns.

## 2 Related Work

**Latent world models and their reuse for control.** PlaNet and Dreamer learn latent dynamics together with a reward head and act by planning or by imagined rollouts inside the learned model (Hafner et al., 2019; 2025). World models built on pretrained features predict DINOv2 patch tokens or self-supervised video embeddings and plan toward a goal embedding with a fixed distance (Zhou et al., 2025; Assran et al., 2025; Oquab et al., 2024). LeWM trains encoder and predictor end to end from pixels and releases checkpoints that plan with the same goal distance (Maes et al., 2026). The JEPA-WM study identifies which architectural and training choices make such planning succeed, keeping a fixed latent distance as the planning cost (Terver et al., 2026). RC-aux adds multi-horizon prediction and budget-conditioned reachability supervision, whose gradients also reach the encoder whose latents the planner searches over (Li et al., 2026). ACID scores predicted transitions with a separately trained inverse-dynamics verifier and changes the planning cost according to their action consistency (Seo et al., 2026). AdaJEPA updates selected encoder and predictor layers on newly observed transitions, adapting the model at test time rather than the policy (Wang et al., 2026). Sensorimotor world models (SMWM) train encoder, predictor and inverse model jointly and probe frozen features for physical-state predictability rather than for control (Ivashkov et al., 2026). Our method uses the frozen LeWM encoder inside the training-only value of an offline goal-conditioned pixel agent, leaving the world model outside the deployed policy (Park et al., 2025).

**Offline goal-conditioned reinforcement learning.** Universal value function approximators and hindsight relabeling established value as a function of state and goal, learned from trajectories relabeled with reached states (Schaul et al., 2015; Andrychowicz et al., 2017). Goal-conditioned

behavior cloning (GCBC) imitates the actions of relabeled trajectories with a supervised loss, which requires no value function at any stage of training (Ghosh et al., 2021). Diffusion policies model multimodal action distributions by iterative denoising, a policy class that our comparison adapts to goal conditioning as the imitation baseline Diffusion Policy-GC (Chi et al., 2023). Implicit Q-learning fits an upper expectile of the value targets, which approximates the best in-support value without querying unseen actions (Kostrikov et al., 2022). Its goal-conditioned variant GCIQL learns an action value by expectile regression on goals relabeled from the offline trajectories (Park et al., 2025). HIQL extracts a high-level policy over subgoal representations and a low-level policy toward them from one goal-conditioned state value (Park et al., 2023). Contrastive (CRL) and quasimetric (QRL) methods learn representations whose inner product or quasimetric distance equals a goal-conditioned value or temporal distance, without a separate value network (Eysenbach et al., 2022; Wang et al., 2023; Myers et al., 2024). Dual goal representations (DGR) describe a goal by its temporal distances from all other states, a goal input that depends on the dynamics alone (Park et al., 2026). Temporal representation alignment (TRA) adds an alignment loss between present and future representations, which our comparison attaches to an expectile state value following a public implementation (Myers et al., 2025). These methods learn value and policy on the same input, whereas our value reads frozen world-model latents and our policy reads pixels (Park et al., 2025).

**Pretrained representations for value and critics with extra inputs.** Time-contrastive networks learn visual representations from unlabeled video by self-supervision, a pretraining signal that needs neither rewards nor goal labels (Sermanet et al., 2018). VIP and LIV pretrain visual representations whose embeddings supply rewards for downstream robot control (Ma et al., 2023b;a). These methods take the reward or value signal from the pretrained embedding, whereas we fit a value network on frozen latents by expectile regression (Ma et al., 2023b; Kostrikov et al., 2022). Asymmetric actor-critic training feeds the critic a richer input than the image-based actor and uses the critic during training alone (Pinto et al., 2018). Its critic reads exact simulator state, whereas our value reads a world-model latent computed from the same images as the actor input (Pinto et al., 2018; Maes et al., 2026). It learns in simulation, where that state is available, whereas our value learns offline from recorded images, actions and relabeled goals (Pinto et al., 2018; Park et al., 2025).

**Goal equivalence and reachability in latent spaces.** Bisimulation metrics measure the behavioral similarity of two states through their reward difference and a distance between their next-state distributions (Ferns et al., 2004). Goal-conditioned bisimulation lifts the metric to state-goal pairs, so that states with equal reward and transition differences under one goal become analogous (Hansen-Estruch et al., 2022). These metrics decide which states a representation may merge, which is the question any fixed map before a value must answer (Ferns et al., 2004; Hansen-Estruch et al., 2022). RC-aux shows that states close in a predictive latent space can be unreachable within a planning budget (Li et al., 2026). A frozen world-model latent raises this merging question for a value on it, because its encoder was trained before any goal or value existed (Maes et al., 2026).

# 3 METHOD

## 3.1 PRELIMINARIES

**Offline goal-conditioned control.** Given an observed image $o$ and a goal image $o_g$, a policy $\pi(a \mid o, o_g)$ has to reach the goal within a fixed budget of steps (Park et al., 2025). Training uses an offline dataset $\mathcal{D}$ of trajectories $(o_0, a_0, o_1, a_1, \dots)$ without rewards, whose later frames are relabeled as goals for earlier ones (Andrychowicz et al., 2017). The reward is $r = 0$ at the goal frame and $r = -1$ otherwise, discounted by $\gamma = 0.99$ per step (Park et al., 2025). We write $s$ and $g$ for a state and a goal in the form a network reads, images or latents, and $s'$ for the next state of the trajectory.

**Expectile value learning.** A goal-conditioned value $V(s, g)$ estimates the discounted return from $s$ to $g$, which under this reward counts the remaining steps with discounting (Schaul et al., 2015). It is fitted toward bootstrapped targets from a slowly moving copy $\bar{V}$, whose parameters track those of $V$ by Polyak averaging at rate $0.005$,

$$\mathcal{L}_V = \mathbb{E}_{(s,s',g)}\big[\, |\tau - \mathbb{1}(u < 0)|\, u^2 \,\big], \qquad u = r + \gamma \bar{V}(s', g) - V(s, g), \tag{1}$$

where the expectile $\tau \in (0.5, 1)$ gives targets above the prediction more weight than targets below it (Kostrikov et al., 2022). Values of $\tau$ near one approximate the best value that the actions in the

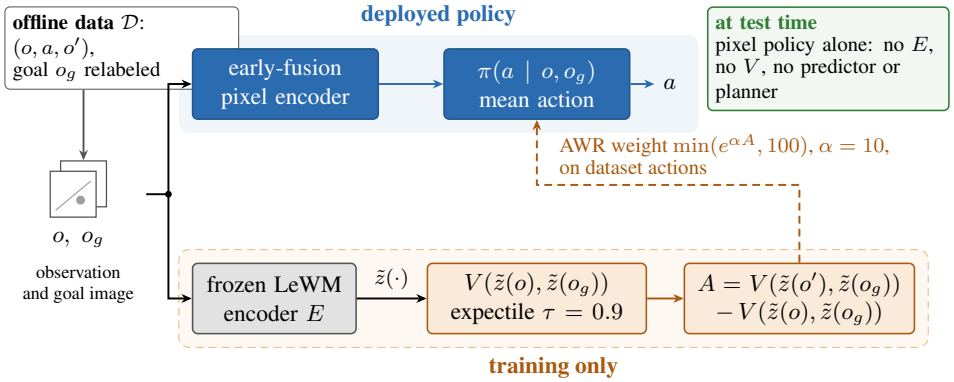



Figure 2: LCPP. The deployed policy reads the early-fused observation and goal images and outputs its mean action (top, blue). In training alone, an expectile value on standardized frozen LeWM latents (grey) is the critic, whose advantage weights the imitation of dataset actions (bottom, orange).

dataset support, without evaluating actions that the dataset does not contain (Kostrikov et al., 2022). Value goals are the current frame with probability 0.2 and otherwise a later frame of the trajectory, with no bootstrap beyond the goal frame. That frame lies a geometric number of steps ahead with success probability $1 - \gamma$, capped at the end of the trajectory (Park et al., 2025). Under this reward, a value maps to an implied number of remaining steps,

$$\hat{d}(s, g) = \log\big(1 + (1 - \gamma)V(s, g)\big) / \log \gamma. \tag{2}$$

**Advantage-weighted extraction.** A policy is extracted by advantage-weighted regression, which imitates dataset targets with weights that grow exponentially in their advantage (Kostrikov et al., 2022),

$$\mathcal{L}_\pi = -\mathbb{E}\big[\min\big(e^{\alpha A}, 100\big) \log \pi(y \mid x)\big], \tag{3}$$

where $x$ is the policy input and $y$ its target, either the action $a$ with $A = V(s', g) - V(s, g)$ or a subgoal $k$ steps ahead (Park et al., 2023). At $\alpha = 0$ the loss reduces to goal-conditioned behavior cloning, whereas a larger $\alpha$ concentrates imitation on the targets that the value ranks as progress (Ghosh et al., 2021). Actor goals are later frames of the same trajectory drawn uniformly, while the trained policy acts without querying the value (Park et al., 2025). We use a Gaussian policy with a learned mean and a fixed unit standard deviation, whose mean is the action at test time.

**Pixel and world-model encoders.** A pixel agent reads $64 \times 64$ RGB images through a small IMPALA network with convolutional stages of 16, 32 and 32 channels and a 512-unit projection (Park et al., 2025). Early fusion stacks the state and goal images along the color channels, whereas late fusion encodes them separately (Park et al., 2025). A latent world model pairs an encoder $E$ that maps an image to a latent with a predictor that maps a latent and an action to the next latent (Maes et al., 2026). LeWM trains both parts end to end from pixels and releases one checkpoint per environment, whose encoder produces 192-dimensional latents (Maes et al., 2026). We keep $E$ frozen and export latents once per dataset frame, so our method uses no predictor or planner.

### 3.2 Proposed Method

LCPP assigns the two jobs of Section 1 to two representations, as Figure 2 shows. The value acts as a training-only critic on frozen world-model latents, whereas the actor is the pixel policy, as the next two paragraphs describe.

**Latent value.** The value answers the question of progress, so it reads the frozen latents rather than pixels (Figure 2, bottom). We standardize each of the 192 latent dimensions with the mean and SD of the training frames, writing $\tilde{z}(o)$ for the result. The value $V(\tilde{z}(o), \tilde{z}(o_g))$ is a three-layer MLP with 512 units on the concatenated latents, without layer norm and without any pixel input. It is fitted by Equation 1 at $\tau = 0.9$ with one target network and the value goals of Section 3.1.

**Pixel actor.** The actor answers the question of control, so it keeps the early-fused images of state and goal (Figure 2, top). It is the Gaussian policy of Section 3.1 and is trained by Equation 3 at $\alpha = 10$ on the latent advantage $A = V(\tilde{z}(o'), \tilde{z}(o_g)) - V(\tilde{z}(o), \tilde{z}(o_g))$. At test time it outputs its mean action, clipped to $[-1, 1]$, so no deployed step uses the frozen encoder or the value.

**Remaining settings.** The encoder architecture and the optimization settings of the actor follow the pixel agents of Park et al. (2025), listed in Appendix B. LCPP uses a single value network without image augmentation, each of its components helping in at least one environment (Section 4.2).

**Formal results.** All statements below, proved in Appendix A, rest on one idealization of the learned value that Remark 1 discusses further.

**Assumption A.** On the support of $\mathcal{D}$, (i) the dynamics are deterministic, (ii) $d(s, g) \in \mathbb{N}$ is the shortest number of steps from $s$ to $g$, and (iii) a reached goal is absorbing with zero reward.

**Proposition 1** (Signal of flat and hierarchical extraction)**.** *Under Assumption A, let $d = d(s, g) \geq k$. The optimal values of a successor one step closer to $g$ and of a successor at unchanged distance differ by $S_F(d) = \gamma^{d-1}$. The optimal values of a subgoal at distance $d - k$ from $g$ and of a subgoal at distance $d$ differ by*

$$S_H(d) = \gamma^{d-k} \frac{1 - \gamma^k}{1 - \gamma} \;\geq\; k\, S_F(d). \tag{4}$$

*Both signals decay geometrically in $d$, with the hierarchical signal at least $k$ times the flat one.*

**Lemma 1.1** (Closed form of the optimal value)**.** *Under Assumption A, $V^*(s, g) = -(1 - \gamma^{d(s,g)})/(1 - \gamma)$, so $d(s, g) = \log\big(1 + (1 - \gamma)V^*(s, g)\big)/\log\gamma$, which is Equation 2 at $V^*$.*

**Lemma 1.2** (Sign reliability under value noise)**.** *Let $\hat{V} = V^* + b + \xi$, where $|b| \leq \varepsilon$ at every state-goal pair and $\xi$ is jointly Gaussian at the two compared points, with mean zero, variance $\sigma^2 > 0$ and correlation $\rho < 1$. For $\alpha > 0$, if the optimal values of the two compared points differ by $S > 0$, the better point receives the larger unclipped weight in Equation 3 with probability at least $\Phi\big((S - 2\varepsilon)/\sigma_\Delta\big)$, where $\Phi$ is the standard normal CDF and $\sigma_\Delta = \sigma\sqrt{2(1 - \rho)}$.*

*Remark* 1 (Scope of the idealization). Assumption A and the Gaussian noise model idealize a learned value that is trained on stochastic resets and a finite dataset. The bias and noise of a learned value are not observed directly, so our experiments relate these results to measured success rates and to value rankings along data transitions. The learned value follows the data policy at $\tau < 1$, so Equation 2 can overstate the distance that the extracted policy needs, which Appendix L measures.

**Proposition 2** (Floor of a fixed input map)**.** *Let $V(s, g) = f\big(\psi(s, g)\big)$ for a fixed map $\psi$ and an arbitrary function $f$. For any two state-goal pairs with $\psi(s_1, g_1) = \psi(s_2, g_2)$,*

$$\max_{i \in \{1,2\}} \big|V(s_i, g_i) - V^*(s_i, g_i)\big| \;\geq\; \tfrac{1}{2}\big|V^*(s_1, g_1) - V^*(s_2, g_2)\big|. \tag{5}$$

*Any bias bound $\varepsilon$ of Lemma 1.2 for such a value is therefore at least $\varepsilon_\psi = \sup \frac{1}{2}|V^*(s_1, g_1) - V^*(s_2, g_2)|$ over colliding pairs, and $\varepsilon_\psi = 0$ when $\psi$ merges no pairs of different optimal value.*

**Lemma 2.1** (Advantages inherit the floor)**.** *An advantage from such a value is a difference of two of its values, so its bias is at most $2\varepsilon$ with $\varepsilon \geq \varepsilon_\psi$. If $\psi(s', g) = \psi(s, g)$ for a successor $s'$ one step closer to $g$, then $A = 0$ for every $f$, although the optimal values differ by $S_F(d)$.*

The latent value of LCPP is the case $\psi(o, o_g) = (\tilde{z}(o), \tilde{z}(o_g))$, whose collisions are fixed once the encoder $E$ is frozen, while a pixel value learns its own map from the value loss during training.

*Remark* 2 (When flat extraction suffices). Evaluation goals lie 50 data steps ahead, which bounds each start-to-goal distance by 50 and the flat signal of Proposition 1 from below by $\gamma^{49} \approx 0.61$. By Lemma 1.2 with $\varepsilon = 0$, flat extraction then keeps a bound of at least $\Phi(1)$ whenever $\sigma_\Delta \leq 0.61$, which is $\sigma \leq 0.43$ for uncorrelated noise.

*Remark* 3 (Clipping). At $\alpha = 10$ the weight of Equation 3 hits its clip of 100 once $A \geq \ln(100)/10 \approx 0.46$, below the flat bound $\gamma^{49} \approx 0.61$. For $V^*$ every progressing step is clipped, so the weights separate progressing steps from the rest, not from each other (Appendix I).

*Remark* 4 (What a training-only critic can change). The value enters Equation 3 through the advantage alone, so the policy class, its inputs and its deployment cost do not depend on what the value reads. By Lemmas 1.2 and 2.1, its input acts on extraction through the bias and noise at compared points and through the floor $\varepsilon_\psi$ of the fixed map $\psi$ from which it reads state and goal.

Table 1: Main comparison with 100k downstream updates for every trained row, success rate as mean ± SD over training seeds 0–4 in every row. All five columns use hindsight goals from held-out expert trajectories, with the goal 50 steps ahead and a 50-step budget. Frozen-encoder rows use the public LeWM checkpoints except on Scene†, where they and the latent value of LCPP use a LeWM trained by us with the official recipe, to be released. AdaJEPA adapts its last predictor layer at test time every 5-step chunk, following its paper schedule without offline training. ▼ and ▲ mark a row significantly below or above LCPP in a two-sided paired t-test ($p < 0.05$) over matched seeds.

| Method | Reacher | TwoRoom | Cube | PushT | Scene† |
|---|---|---|---|---|---|
| LCPP (ours) | $\mathbf{1.000 \pm 0.000}$ | $\mathbf{1.000 \pm 0.000}$ | $\mathbf{0.960 \pm 0.014}$ | $\mathbf{0.396 \pm 0.071}$ | $\mathbf{0.912 \pm 0.036}$ |
| *From-scratch goal-conditioned RL* | | | | | |
| GCBC | $0.768 \pm 0.030^{\blacktriangledown}$ | $0.956 \pm 0.009^{\blacktriangledown}$ | $0.704 \pm 0.030^{\blacktriangledown}$ | $0.240 \pm 0.028^{\blacktriangledown}$ | $0.664 \pm 0.022^{\blacktriangledown}$ |
| Diffusion Policy-GC | $0.348 \pm 0.046^{\blacktriangledown}$ | $0.800 \pm 0.024^{\blacktriangledown}$ | $0.832 \pm 0.046^{\blacktriangledown}$ | $0.324 \pm 0.075$ | $0.732 \pm 0.041^{\blacktriangledown}$ |
| GCIQL | $0.932 \pm 0.018^{\blacktriangledown}$ | $0.980 \pm 0.000^{\blacktriangledown}$ | $0.868 \pm 0.023^{\blacktriangledown}$ | $0.204 \pm 0.046^{\blacktriangledown}$ | $0.648 \pm 0.027^{\blacktriangledown}$ |
| DGR | $0.820 \pm 0.065^{\blacktriangledown}$ | $\mathbf{1.000 \pm 0.000}$ | $0.860 \pm 0.049^{\blacktriangledown}$ | $0.272 \pm 0.067^{\blacktriangledown}$ | $0.844 \pm 0.017^{\blacktriangledown}$ |
| TRA | $0.576 \pm 0.065^{\blacktriangledown}$ | $0.072 \pm 0.027^{\blacktriangledown}$ | $0.880 \pm 0.037^{\blacktriangledown}$ | $0.216 \pm 0.080^{\blacktriangledown}$ | $0.776 \pm 0.026^{\blacktriangledown}$ |
| QRL | $0.760 \pm 0.414$ | $\mathbf{1.000 \pm 0.000}$ | $0.824 \pm 0.033^{\blacktriangledown}$ | $0.104 \pm 0.030^{\blacktriangledown}$ | $0.628 \pm 0.076^{\blacktriangledown}$ |
| CRL | $0.816 \pm 0.064^{\blacktriangledown}$ | $0.960 \pm 0.014^{\blacktriangledown}$ | $0.780 \pm 0.047^{\blacktriangledown}$ | $0.240 \pm 0.080^{\blacktriangledown}$ | $0.668 \pm 0.050^{\blacktriangledown}$ |
| *On the frozen LeWM encoder* | | | | | |
| RC-aux | $0.832 \pm 0.046^{\blacktriangledown}$ | $0.632 \pm 0.041^{\blacktriangledown}$ | $0.596 \pm 0.022^{\blacktriangledown}$ | $0.284 \pm 0.026^{\blacktriangledown}$ | $0.596 \pm 0.050^{\blacktriangledown}$ |
| ACID | $0.776 \pm 0.036^{\blacktriangledown}$ | $0.516 \pm 0.030^{\blacktriangledown}$ | $0.664 \pm 0.036^{\blacktriangledown}$ | $0.300 \pm 0.028^{\blacktriangledown}$ | $0.588 \pm 0.061^{\blacktriangledown}$ |
| AdaJEPA | $0.360 \pm 0.081^{\blacktriangledown}$ | $0.404 \pm 0.043^{\blacktriangledown}$ | $0.568 \pm 0.023^{\blacktriangledown}$ | $0.124 \pm 0.033^{\blacktriangledown}$ | $0.568 \pm 0.046^{\blacktriangledown}$ |
| SMWM | $0.948 \pm 0.011^{\blacktriangledown}$ | $0.968 \pm 0.011^{\blacktriangledown}$ | $0.640 \pm 0.032^{\blacktriangledown}$ | $0.296 \pm 0.078$ | $0.616 \pm 0.033^{\blacktriangledown}$ |
| HIQL | $0.756 \pm 0.097^{\blacktriangledown}$ | $\mathbf{1.000 \pm 0.000}$ | $0.820 \pm 0.058^{\blacktriangledown}$ | $0.372 \pm 0.058$ | $0.700 \pm 0.057^{\blacktriangledown}$ |

**Theorem 1** (Reliability of extraction)**.** *Under Assumption A, $\alpha > 0$ and the noise model of Lemma 1.2, let an extraction unit $u \in \{F, H\}$ compare a choice that progresses toward the goal with one that does not, by one step for $u = F$ and by $k$ steps for $u = H$, at distance $d$. (i) The progressing choice receives the larger unclipped weight with probability at least $\Phi\big((S_u(d) - 2\varepsilon_u)/\sigma_{\Delta,u}\big)$, where $S_u$ is given by Proposition 1, $\varepsilon_u$ and $\sigma_{\Delta,u}$ are the $\varepsilon$ and $\sigma_\Delta$ of Lemma 1.2 at the compared points, and $\varepsilon_u \geq \varepsilon_\psi$ whenever the value reads its inputs through a fixed map $\psi$. (ii) For flat extraction with $\varepsilon_u = 0$ and difference noise $\sigma_\Delta$, this bound is at least $\Phi(1) \approx 0.84$ if and only if $\gamma^{d-1} \geq \sigma_\Delta$.*

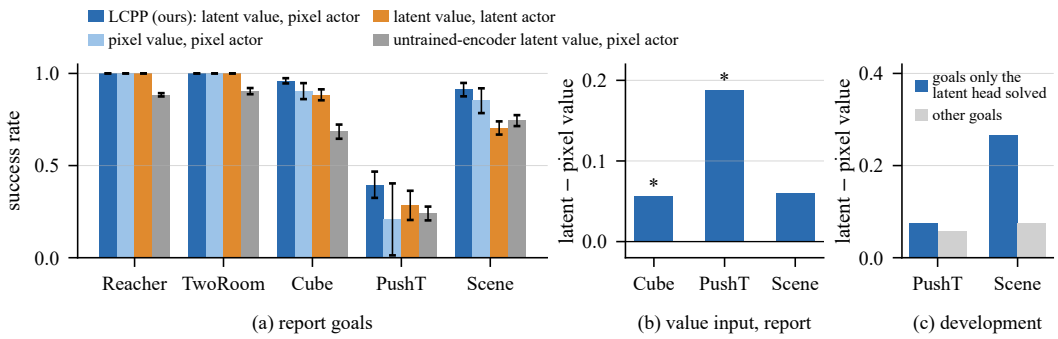
# 4 Experiments

## 4.1 Experimental Settings

**Environments and data.** The main comparison covers Reacher, TwoRoom, Cube and PushT, each with its public LeWM checkpoint and its offline pixel dataset (Maes et al., 2026). Scene has no public LeWM checkpoint, so we train one with the official recipe for ten epochs on the Scene corpus without evaluation episodes (Park et al., 2025). Two checks fixed before training passed: a zero-action success of 0.28 on Scene and a Cube LeWM retrained with this recipe at 0.632 against the public 0.600. The Scene checkpoint also supplies the latents of our value, is marked with a dagger in Table 1 and will be released after the review. Every trained method in an environment uses the same training pool, which excludes every episode that supplies an evaluation start or goal.

**Baselines.** We compare seven from-scratch methods in three branches: imitation (GCBC, Diffusion Policy-GC), expectile value learning (GCIQL, DGR, TRA) and distance or contrastive value learning (QRL, CRL). Five further rows build on the frozen LeWM encoder, namely RC-aux, ACID, AdaJEPA, SMWM and a HIQL head on the latents (Appendix L). These rows adapt each method to the frozen public checkpoint, so RC-aux and SMWM, which originally train their encoders, keep them fixed here (Appendix C). LCPP uses the same 100k updates as every other trained row.

**Protocol.** Every trained part uses 100k updates at batch 256 with the learning rate of its paper. Every column uses 50 report goals taken 50 steps ahead on held-out expert trajectories, with a 50-step budget per episode. Rows that plan on the frozen world model use the CEM configuration selected on validation goals in a separate planning study with the same checkpoints. Every row reports mean ± sample SD over training seeds 0–4, while Table E.1 gives LCPP, SMWM and




(a) report goals (b) value input, report (c) development

Figure 3: Where the value is learned. (a) Report goals, mean ± SD over training seeds 0–4, for LCPP, the same actor recipe with a pixel value, the same recipe with the actor on the latents, and the same recipe with value latents from an untrained encoder. (b) Paired difference between the latent and the pixel value over 250 seed-goal pairs, with * marking rejection in the pair and goal-clustered tests after Holm correction over five columns. Reacher and TwoRoom differ by 0.000. (c) Development goals: the same difference on the goals that a variant acting on frozen latents solved in at least two of three seeds while a pixel agent solved them in at most one, and on the other goals.

GCIQL over 20 seeds. Table 1 fixes the number of downstream updates rather than total compute, because LCPP and the other frozen-encoder rows start from a pretrained world model.

**Development goals and tests.** Design decisions used 50 development goals per environment, disjoint from the report goals, with seeds 100–104 for frozen-encoder heads and 200–202 for pixel-actor variants. Each decision followed a rule whose script hash was recorded before any result. Pre-registered judges flip the signs of the 250 paired seed-goal differences on the report goals 20,000 times, treating those pairs as exchangeable. Since one trained policy serves many goals, each recurring across seeds, we also report a goal-clustered sign flip over the 50 seed-averaged goal differences. A seed-level exact permutation of the five seed means per arm asks whether a difference recurs across training runs, while a seed-goal bootstrap gives intervals (Appendix E). On one NVIDIA RTX 5090, 100k updates of LCPP take 404 to 433 seconds (Appendix M).

### 4.2 Experimental Results

**Main comparison.** Table 1 compares LCPP with seven from-scratch and five frozen-encoder rows at equal updates on the same report goals. LCPP is bold in all five columns, alone in four and tied at 1.000 with DGR, QRL and the HIQL head on TwoRoom. It leads the next row by 0.052 on Reacher over SMWM, by 0.080 on Cube over TRA, by 0.024 on PushT over HIQL and by 0.068 on Scene over DGR. A Welch test on seed means gives $p = 0.0005$ on Reacher, $p = 0.006$ on Cube and $p = 0.010$ on Scene, but $p = 0.58$ on PushT (Appendix E). On Cube the planning rows reach 0.568 to 0.664 against 0.640 for SMWM, all querying the frozen encoder while acting (Figure 1b). LCPP instead reads the same frozen encoder only through its value during offline training.

**Where the value is learned (Proposition 2, Lemma 1.2 and Remark 4).** The pre-registered ablation replaces the latent value by a value on early-fused pixels and keeps the value and actor recipes, the goals and the seeds (Figure 3a). The latent value raised success by 0.056 on Cube and by 0.188 on PushT, both significant after Holm correction in the pair and goal-clustered tests (Figure 3b). On Scene it added 0.060 with two-sided $p = 0.020$ before Holm correction, while Reacher and TwoRoom stayed at 1.000 in both arms. Over 20 seeds per arm the Cube, PushT and Scene gains were 0.055, 0.062 and 0.057, with Cube and Scene seed-level significant after Holm correction (Appendix E). Two of five PushT seeds with the pixel value reached no goal, whereas every PushT seed of LCPP solved 0.34 to 0.50 of its goals. A value warm-up of 50k updates before actor training removed these PushT failures and closed the PushT gap, while the Cube gap persisted (Appendix K). The actors of both arms share their recipe, so by Remark 4 the difference passes through the advantage weights of training alone. With both final values held fixed as teachers, the latent weights still led by 0.060 on Cube over ten seeds and by 0.084 on Scene (Figure 4). In the pair

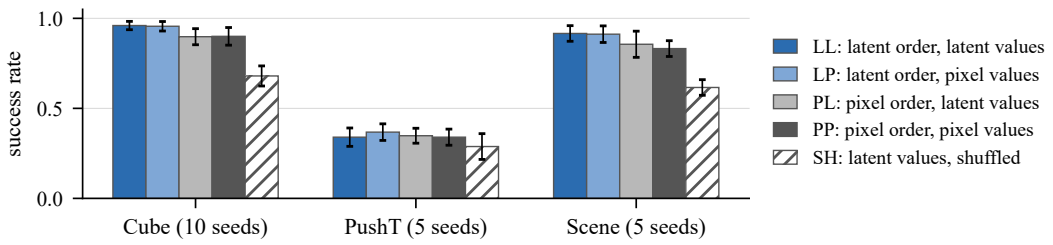
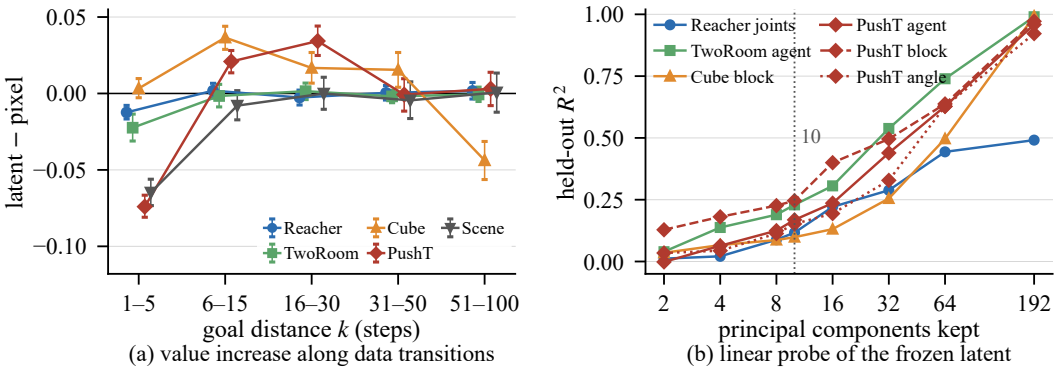



Figure 4: Fixed-teacher weight test. With both final values frozen as teachers, five pixel actors of each seed share their initialization and minibatches and differ in their advantage weights alone, shown as mean ± SD over training seeds. Actors trained with the latent order of weights (LL, LP) lead those trained with the pixel order (PL, PP) on Cube and on Scene (Appendix J).




(a) value increase along data transitions

(b) linear probe of the frozen latent

Figure 5: (a) Share of held-out data transitions with $V(s', g) > V(s, g)$ for the latent value of LCPP minus that of the pixel value, by goal distance, mean over five training seeds with 95% bootstrap intervals over 4,000 pairs per bin. (b) Held-out $R^2$ of a linear probe from the top principal components of the frozen latent to task-state variables, with ten components marked.

test only the order of the weights, not their values, was significant on both, so the gain lies in which transitions receive the larger weights. On Cube this order effect also held at the seed level across the ten training seeds. A second controlled arm changes the actor input alone, feeding the actor the same standardized latents while every other setting of LCPP stays fixed (orange bars). The pixel actor was better by 0.076 on Cube, 0.112 on PushT and 0.208 on Scene in the pair and goal-clustered tests after Holm correction, with both arms at 1.000 elsewhere. On Cube and Scene every seed of LCPP scored above every seed of this arm, a seed-level $p$ of 0.004 that survives the correction. A third arm takes the value latents from a randomly initialized copy of the LeWM encoder, frozen in the same way (grey bars). The pretrained encoder led all five columns by 0.096 to 0.276, significant in every test, with the untrained latents below the pixel value in four columns (Table D.1).

**Which components LCPP needs (Lemma 1.2 and Remark 1).** A pre-registered ablation removed one component of LCPP at a time on the report goals, with its judge fixed before any arm ran (Appendix D). Setting $\alpha = 0$ lowered all five columns by 0.052 to 0.300 in the pair and seed-level tests, leaving no value difference to order actions as Lemma 1.2 requires. An expectile of 0.5, at which the value tracks the data policy more closely (Remark 1), lowered Cube by 0.060 in both tests. Late fusion of state and goal images in the actor collapsed Reacher to 0.020 in every seed and lowered PushT by 0.092, again in both tests. With the three input comparisons of Figure 3, each component of LCPP thus helps in at least one of the five evaluated environments.

**Whether the latent value rises along data transitions (Lemma 1.2 and Remark 4).** We tested the ranking behind Remark 4 on held-out trajectories with the saved checkpoints of LCPP and the pixel-value arm, under a judge fixed in advance. For 20,000 state-successor pairs per environment, with goals 1 to 100 steps ahead in five distance bins, we counted how often $V(s', g) > V(s, g)$, a rate of value rising along recorded transitions rather than of optimal progress. Over all bins the

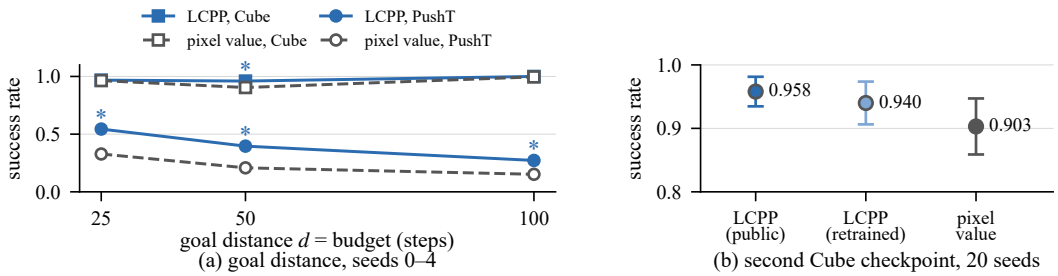



Figure 6: (a) Success of LCPP and the pixel value with goals $d$ steps ahead and a $d$-step budget, with * marking a Holm-significant pair test (Appendix G). (b) Cube success, mean $\pm$ SD over 20 seeds, of LCPP on the public and on a retrained LeWM checkpoint against the pixel value (Appendix H).

latent value rose along more pairs on Cube and fewer on TwoRoom and Scene after Holm correction, with no significant change on Reacher and PushT. The difference depended on goal distance, since for goals within five steps the latent value rose less often in four environments, by 0.012 to 0.074 (Figure 5a). For goals 6 to 30 steps ahead the latent value rose more often on Cube and PushT, by up to 0.037. To test whether the gain depends on goal distance, we re-evaluated seeds 0–4 of both arms at goals 25 or 100 steps ahead (Figure 6a, with a second Cube checkpoint in b). The latent value led on PushT by 0.216 at $d = 25$ and 0.120 at $d = 100$, with no significant Cube difference. On a Cube LeWM retrained by us, the latent value led the pixel value by 0.037 over 20 training seeds per arm (Appendix H). This gain was significant in the pair test ($p = 0.0011$) and at the seed level ($p = 0.0056$), so it does not rest on the public checkpoint alone. On Cube a 50k-update warm-up of the pixel value changed neither arm significantly, leaving the difference at 0.044 without and 0.048 with it (Appendix K). The Cube gain of LCPP thus survives the pixel-value warm-up.

**How much of the task state the frozen latent keeps (Proposition 2 and Lemma 2.1).** Proposition 2 puts a floor under the latent value only where the frozen encoder merges states or goals of different value. A held-out linear probe recovers the task state from all 192 dimensions with $R^2$ of 0.92 to 0.99 on TwoRoom, Cube and PushT, against 0.49 for the Reacher arm joints (Figure 5b).

**How the latent advantage weights the data (Remark 3).** We recomputed the advantage weights of both values on the held-out transitions of Figure 5a (Appendix I). Beyond five steps, the latent value gave an advantage whose five-seed mean exceeded its seed SD on 0.04 to 0.16 more pairs on Cube, PushT and Scene. Between 39 and 56 percent of all weights and 50 to 83 percent of those with a positive advantage sat at the clip, the saturation that Remark 3 describes.

## 5 Conclusion

We asked where a frozen world model should enter goal-conditioned control learned offline from images. Our answer separates judging progress, which the latent value does as a critic in training, from acting, which the deployed policy does on pixels. Under equal downstream updates LCPP leads or ties all five columns, while swapping either of its two choices lowers success in three columns. Our analysis bounds the reliability of its weights by value errors and a frozen-map floor. With both values held fixed as teachers, the latent weights kept their lead on Cube and Scene through the order in which they rank transitions.

For goals within five steps the latent value rose along data transitions less often than the pixel value in four environments. Encoders retrained with a time-free or reconstruction objective lost to LeWM on Scene, the only environment where we trained them (Appendix F). With 20 seeds per arm the value-input gain is seed-level significant on Cube and Scene alone, while the pixel value led on Reacher in the pair test. On PushT a 50k-update warm-up of the pixel value removed its gap to LCPP, which points to an early-training source of that gain. Apart from a second Cube checkpoint, where LCPP fell by 0.018 against the public one, all other results use one LeWM checkpoint per environment and test goal distances other than 50 steps on Cube and PushT alone.

### AI USE STATEMENT

In this work, we used generative AI tools to aid and polish the writing, to retrieve and discover related work, to draft sections of the paper and to help prove its mathematical claims. We also used them in research execution, where they wrote experiment code, launched training and evaluation jobs on our compute and analysed the resulting logs. Experiment code was checked with smoke runs and synthetic tests before use, while each pre-registered comparison was decided by a judge script whose hash was fixed before its results existed. Every reported number traces to the result files through the scripts that generate the tables and figures. We checked cited claims against the cited papers and each proof step by step. We reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

The anonymous repository linked below the abstract contains the training and evaluation code of LCPP, the report goals and data pools of the five environments, and a script for its row of Table 1. Section 4.1 describes the environments, goals, seeds and statistical tests, while Appendix B lists the hyper-parameters of LCPP. Appendix C describes how each frozen-encoder baseline was built, with the uncertainty of the input comparisons in Appendix E and compute costs in Appendix M. Appendix A contains the proofs of Propositions 1 and 2, their lemmas and Theorem 1, while the Scene LeWM checkpoint of our value will be released after the review.

## REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

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. *arXiv preprint arXiv:2303.04137*, 2023.

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pp. 162–169, 2004.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/1912.06088.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Philippe Hansen-Estruch, Amy Zhang, Ashvin Nair, Patrick Yin, and Sergey Levine. Bisimulation makes analogies in goal-conditioned reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8407–8426, 2022.

Petr Ivashkov, Randall Balestriero, and Bernhard Schölkopf. Sensorimotor world models: Perception for action via inverse dynamics. *arXiv preprint arXiv:2606.20104*, 2026. URL https://arxiv.org/abs/2606.20104.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.06169.

Wenyuan Li, Guang Li, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama. Predictive but not plannable: RC-aux for latent world models. *arXiv preprint arXiv:2605.07278*, 2026.

Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: Language-image representations and rewards for robotic control. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23301–23320, 2023a.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations*, 2023b.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 37076–37096, 2024.

Vivek Myers, Bill Chunyuan Zheng, Anca Dragan, Kuan Fang, and Sergey Levine. Temporal representation alignment: Successor features enable emergent compositionality in robot instruction following. *arXiv preprint arXiv:2502.05454*, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2307.11949.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.20092.

Seohong Park, Deepinder Mann, and Sergey Levine. Dual goal representations. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.06714.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems*, 2018. URL https://arxiv.org/abs/1710.06542.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, 2015.

Gawon Seo, Dongwon Kim, and Suha Kwak. ACID: Action consistency via inverse dynamics for planning with world models. *arXiv preprint arXiv:2607.02403*, 2026. URL https://arxiv.org/abs/2607.02403.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *IEEE International Conference on Robotics and Automation*, pp. 1134–1141, 2018.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Transactions on Machine Learning Research*, 2026. URL https://arxiv.org/abs/2512.24497. arXiv:2512.24497.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 36411–36430, 2023.

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. AdaJEPA: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 2026. URL https://arxiv.org/abs/2606.32026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135, 2025.