GCSL FROM PIXELS: GEOMETRIC REGULARIZATION FOR NAVIGABLE VISUAL GOAL SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Goal-Conditioned Supervised Learning (GCSL) learns goal-reaching behavior through hindsight relabeling and self-imitation, without a reward function, value function, or expert demonstrations. We propose that its success in physical state spaces may depend on an assumption it never states, that the state/goal space carries the environment's factors in linearly decodable form. While low-dimensional states provide this for free, we find that a naively trained pixel encoder often fails to obtain linear decodability, and that this gap is fixable. Pretraining the image encoder with a reconstruction loss plus a geometric regularizer combining variance-covariance (VICReg) and an eigenvalue-based collapse penalty (EigenReg) restores linear decodability and reliable goal-reaching across four control environments. GCSL's relabeling objective stays unchanged, though reliable pixel performance also needs frequency-balanced hindsight sampling and horizon weighting in the actor loop. Two controlled comparisons isolate where the improvement comes from. Against end-to-end pixel GCSL, which fails outright with 0% success across all four environments despite working reliably from physical state, our method reaches 94 to 100% success. Against GCBC, a one-shot cloning variant without iterative self-improvement, we isolate the contribution of iteration alone. A further comparison against vanilla-VAE, Beta-VAE, VICReg-only, and EigenReg-only isolates the regularizer's own contribution. Finally, comparing linear against nonlinear decodability of the same representations tests whether linear recoverability is associated with goal-reaching in these environments, or recoverability of any kind is. These results give an empirical characterization of when GCSL's reward-free formulation extends from physical state spaces to pixels.

1 INTRODUCTION

Goal reaching is a core problem in sequential decision-making. Most progress has come through reinforcement learning, using value functions, Bellman backups, and distance-based rewards (Sutton & Barto, 1998; Mnih et al., 2015). Goal-Conditioned Supervised Learning (GCSL) (Ghosh et al., 2019) skips all of this. It relabels trajectories in hindsight, so each one becomes a positive example for whatever goal it reached. The policy retrains on this data, collects new trajectories, and repeats, with no reward function and no value function (Oh et al., 2018).

Visual goal reaching has stayed inside the RL framework. RIG (Nair et al., 2018) and Skew-Fit (Pong et al., 2019) train a variational encoder and use latent distance as a reward for off-policy RL. Recent work replaces the value function with a contrastive classifier (Eysenbach et al., 2022), but the setup stays the same. A value function can work around a bad latent space. A purely supervised, imitation-based method cannot. Prior work has tried GCSL directly on pixels and found it fails without extra machinery (Mendonca et al., 2021). No one has tested whether this failure comes from the representation, or whether regularizing the encoder fixes it.

GCSL's success in physical state spaces may rest on an assumption it never states. The state and goal space may need position, angle, velocity, and similar factors in linearly decodable form, each recoverable by a linear map. GCSL imitates actions along relabeled trajectories and generalizes to new state-goal pairs by interpolating within the conditioning space (Alain & Bengio, 2016; Bengio et al., 2013). Linear decodability may be part of what makes that interpolation meaningful, letting the space behave like a coordinate system where moving toward a goal means progress. Physical

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.

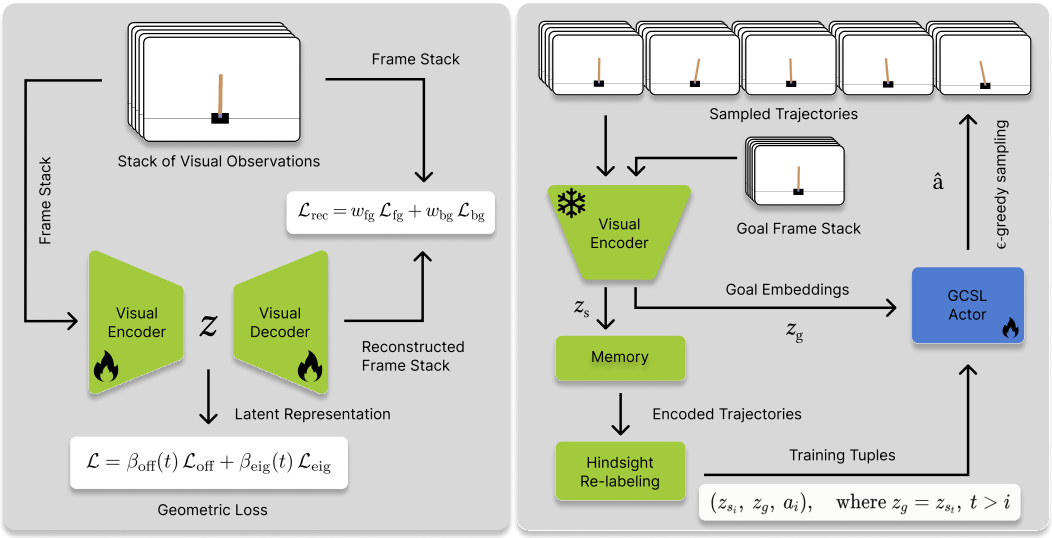

Figure 1: Overview of the two-stage pipeline. In Step 1, the visual encoder trains on stacks of visual observations with a reconstruction loss and a geometric regularizer, producing the latent code z. The reconstruction loss weights motion-masked foreground and background terms $\mathcal{L}_{\text{fg}}$ and $\mathcal{L}_{\text{bg}}$ by w_{fg} and w_{bg}. The geometric loss combines a covariance off-diagonal penalty $\mathcal{L}_{\text{off}}$ and an eigenvalue collapse penalty $\mathcal{L}_{\text{eig}}$, weighted by curriculum terms $\beta_{\text{off}}(t)$ and $\beta_{\text{eig}}(t)$. In Step 2, the frozen encoder embeds sampled trajectories and the goal frame stack into z_s and z_g. Trajectory embeddings are stored in memory and relabeled in hindsight to form training tuples (z_{s_i}, z_g, a_i), where a_i is the action taken at s_i. These tuples train the GCSL actor, which outputs a predicted action $\hat{a}$ for the current state and goal. The actor uses $\hat{a}$ under ϵ-greedy sampling to collect new trajectories.

state spaces give this for free. Pixels do not. Two similar frames can differ a lot in task progress. Two dissimilar frames can look nearly identical in what matters.

We test this directly. We train an encoder end-to-end with the GCSL actor, using only the cross-entropy objective, with no reconstruction loss, no separate pretraining, and no regularization. This encoder fails to reliably reach goals in our tests. The fix is to decouple representation learning from policy learning. We pretrain the image encoder first, then freeze it before actor training, with a reconstruction loss plus a geometric regularizer. The regularizer combines VICReg (Bardes et al., 2021), which discourages collapsed or entangled latent dimensions through an off-diagonal covariance penalty, with an eigenvalue-based collapse penalty we call EigenReg. EigenReg penalizes degenerate latent directions directly, independent of basis (Agrawal et al., 2022; Garrido et al., 2023). We pair this frozen encoder with GCSL's relabeling objective, plus frequency-balanced and horizon-weighted pair sampling (Schaul et al., 2015; Andrychowicz et al., 2017). Together, these restore reliable goal reaching across the four environments we test (Figure 1).

Two controlled comparisons show where the improvement comes from. The first is against end-to-end pixel GCSL, which differs from our pipeline in pretraining, reconstruction, and regularization all at once. The second is against GCBC, a one-shot cloning variant on our regularized encoder that separates out the effect of iteration. We also compare against a vanilla VAE (Kingma & Welling, 2013), a Beta-VAE (Higgins et al., 2017), and VICReg or EigenReg alone, to see what each part of the regularizer contributes. Finally, we compare linear against nonlinear decodability of the same representations, to check whether linear recoverability tracks goal-reaching in these environments.

Our contributions are as follows.

- GCSL fails when the encoder trains end-to-end with the actor, shaped only by hindsight relabeling, in the environments we test. This happens even though the same algorithm works well from physical state. Pixel observations don't appear to carry the structure GCSL needs on their own.

- The gap can be fixed at the representation level. Combined VICReg and EigenReg regularization recovers linear decodability and restores reliable goal reaching in these environments. GCSL's relabeling objective stays the same. Frequency-balanced sampling and horizon weighting in the actor loop are also needed.
- A comparison against end-to-end pixel GCSL suggests that GCSL's objective alone does not produce a usable pixel representation here. A separate comparison against GCBC separates out the effect of the iterative loop.
- We compare VICReg, EigenReg, and their combination against vanilla VAE and Beta-VAE on the same decodability metric. This shows each term's individual and joint contribution.
- We test linear versus nonlinear decodability of the learned representation. This points to whether a linear structure, rather than a broader gain in factor accessibility, tracks the success we observe.

2 Related Work

Goal-conditioned supervised learning. GCSL (Ghosh et al., 2019) treats goal-reaching as supervised learning. Trajectories are relabeled in hindsight into training examples, and the resulting policy iterates without fitting a value function or reward model. This traces back to hindsight relabeling in Hindsight Experience Replay (HER) (Andrychowicz et al., 2017). But GCSL drops the RL component entirely and treats the problem as pure imitation, with no value function to compensate for a poorly-behaved conditioning space. GCSL and its direct extensions (Yang et al., 2022; Jurgenson & Tamar, 2023) test this only in low-dimensional physical state spaces (Levine et al., 2020). A closely related alternative, goal-conditioned behavioral cloning (Ding et al., 2019), forgoes GCSL's iterative loop for one-shot cloning on goal-labeled data. We use it as a baseline to separate out the contribution of iteration (Section 3.9).

Goal-conditioned reinforcement learning from pixels. Visual goal-reaching has stayed almost entirely inside RL. RIG (Nair et al., 2018) trains a VAE over image observations and uses latent distance as a reward for off-policy RL. Skew-Fit (Pong et al., 2019) adds a goal-sampling distribution that targets under-visited regions of the latent space. Recent work replaces the value function with a contrastive classifier (Eysenbach et al., 2022). Goal-conditioned imitation from pixels has also been studied outside this RL framing (Nair et al., 2022; Majumdar et al., 2023; Ma et al., 2022; Parisi et al., 2022). Play-LMP (Lynch et al., 2020) learns a latent plan space from unsegmented visual play data and conditions on image goals at test time, but its latent space is learned jointly with the policy and evaluated behaviorally. It is never probed for decodability, leaving open whether it carries the linear structure we associate with GCSL's success. In this body of work, representation and policy are optimized jointly (Laskin et al., 2020), and the RL objective can compensate for a poorly-behaved latent space. GCSL cannot do this, so we measure representation quality directly through linear probing rather than inferring it from downstream performance (Stooke et al., 2021). We do not benchmark against RIG, Skew-Fit, or contrastive RL empirically for the same reason. A closed-loop comparison would test the value function's capacity to compensate, not the representation-decodability question this paper studies. Our baselines instead hold the algorithm family fixed, reward-free and hindsight-imitation based, and vary only the representation or the presence of iteration. This is the comparison our empirical characterization needs.

Representation learning and disentanglement. Our regularizer builds on VICReg (Bardes et al., 2021), which discourages collapsed and entangled representations through explicit variance and covariance terms, without negative samples or a contrastive objective. Its covariance term traces back to Barlow Twins (Zbontar et al., 2021), which pushes the cross-correlation matrix toward the identity to reduce redundancy. We pair this with an eigenvalue-based collapse penalty we call EigenReg. Dimensional collapse happens when representations span a lower-dimensional subspace than the full embedding, diagnosable by near-zero covariance eigenvalues (Agrawal et al., 2022; Garrido et al., 2023). This differs from complete collapse in both contrastive and non-contrastive self-supervised learning (Jing et al., 2021), and relates to methods that regularize the covariance spectrum directly (He et al., 2024). VICReg's off-diagonal penalty drives dimensions toward independence. Our spectral penalty instead targets degeneracy directly. It is invariant to basis and agnostic to whether the underlying factors are correlated. It also differs from two other spectral approaches. Whitening-based methods enforce the full covariance toward the identity through explicit eigende-

composition (Ermolov et al., 2021), discarding correlation structure entirely rather than tolerating it in non-degenerate directions. Orthogonality regularization typically constrains weight matrices rather than the representation's covariance directly (He et al., 2024). EigenReg instead applies a one-sided floor only to small eigenvalues, leaving correlated but non-degenerate directions unpenalized. We also compare against Beta-VAE (Higgins et al., 2017), which pursues disentanglement through a weighted KL penalty. Locatello et al. (2019) show that unsupervised disentanglement is not identifiable without inductive biases or weak supervision. So we evaluate against a concrete downstream criterion instead, linear decodability of known factors, rather than disentanglement as an unsupervised property. We further compare linear against nonlinear decodability, a distinction the disentanglement literature does not typically make.

3 METHOD

3.1 GCSL PRELIMINARIES

GCSL learns a goal-conditioned policy $\pi(a \mid s, g)$ through supervised learning on relabeled trajectories (Ghosh et al., 2019). A trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ is collected under the current policy. Each state s_t becomes a hindsight goal for every earlier state s_i with $i < t$. The pair (s_i, s_t, a_i) becomes a training example, since a_i made progress toward s_t by construction. The policy retrains by minimizing cross-entropy against a_i, then collects new trajectories and repeats, with no reward, value function, or Bellman backup. Our contribution centers on what s and g are, though the sampling changes to the actor loop in Section 3.6 are also needed in practice. Physical-state GCSL uses the environment's native state vector. Here, s and g come from a learned image encoder, and the paper's central question is what that encoder needs to satisfy.

3.2 VISUAL ENCODER

We encode a history of L (default 5) grayscale frames as a single-channel spatiotemporal volume of shape $(1, L, H, W)$. Five 3D convolutions process this volume, with channel widths 32, 64, 128, 256, and 512, kernel size 3, and stride $(1, 2, 2)$, preserving the temporal axis while halving spatial resolution at each layer (Tran et al., 2015). Each convolution is followed by GroupNorm and GELU (Wu & He, 2018; Hendrycks & Gimpel, 2016). The final volume is pooled to $(L, 2, 2)$, flattened, and mapped through two linear heads to a mean μ and log-variance $\log \sigma^2$ over a D-dimensional (default 512) latent space, with $\log \sigma^2$ clamped to $[-10, 10]$. During training, we draw a sample $z = \mu + \sigma \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$, using the reparameterization trick (Kingma & Welling, 2013). At evaluation we use μ directly. A symmetric decoder of five 3D transposed convolutions reconstructs the input from z, with a spatial upsampling stage before the first deconvolution and a final sigmoid. This architecture is identical across every encoder variant compared in this paper (Section 4.5), and only the regularization on μ differs.

3.3 REPRESENTATION REGULARIZATION

The encoder is trained with two loss terms.

Motion-weighted reconstruction We split reconstruction error into foreground (moving) and background (static) pixels, using the frame-to-frame absolute difference $\Delta_t = |x_t - x_{t-1}|$ for $t = 1, \ldots, L-1$ in the input volume x, with $\Delta_0 \triangleq \Delta_1$ (frame x_0 has no predecessor). We normalize each sample against its own maximum motion, floor it to avoid amplifying noise, and threshold it at 30% to give a foreground mask.

$$\Delta_{\max} = \max\left(\text{amax}_{c,h,w}(\Delta),\ \epsilon_{\text{floor}}\right), \quad \hat{\Delta} = \frac{\Delta}{\Delta_{\max} + \epsilon}, \quad M = \mathbb{1}\left[\hat{\Delta} > \tau\right] \tag{1}$$

with $\epsilon_{\text{floor}} = 0.02$, $\tau = 0.3$. The reconstruction loss weights foreground and background separately.

$$\mathcal{L}_{\text{rec}} = w_{\text{fg}} \cdot \frac{\sum M \odot |\hat{x} - x|}{\sum M} + w_{\text{bg}} \cdot \frac{\sum (1 - M) \odot |\hat{x} - x|}{\sum (1 - M)}, \qquad w_{\text{fg}} = 4.0,\ w_{\text{bg}} = 1.0 \tag{2}$$

Geometric regularization We regularize the batch statistics of μ with two terms, both computed from the empirical covariance $\Sigma = \bar{\mu}^\top \bar{\mu}/(B-1)$ over a batch of size B, where $\bar{\mu} = \mu - \text{mean}(\mu, \text{dim} = 0)$. The first term is VICReg's covariance penalty (Bardes et al., 2021), which penalizes off-diagonal covariance and drives the encoder toward statistically independent dimensions. This aligns its axes with the environment's factors when those factors are independent, but penalizes genuine correlation when they are coupled. The second term, EigenReg, penalizes small eigenvalues of Σ directly.

$$\mathcal{L}_{\text{off}} = \frac{1}{D^2} \sum_{i \neq j} \Sigma_{ij}^2, \qquad \mathcal{L}_{\text{eig}} = \frac{1}{D} \sum_{k=1}^{D} \max\left(0,\ \epsilon - \lambda_k(\Sigma)\right)^2 \tag{3}$$

Unlike $\mathcal{L}_{\text{off}}$, $\mathcal{L}_{\text{eig}}$ is basis-invariant, discouraging degenerate directions without penalizing correlation among dimensions that both carry genuine variance.

We combine the two terms with a curriculum instead of a fixed weighting. $\mathcal{L}_{\text{off}}$'s axis-alignment pressure dominates early, and $\mathcal{L}_{\text{eig}}$ takes over as the sole collapse guard later, once axes are aligned. For E total epochs, at epoch e with progress $t = e/(E-1) \in [0, 1]$, the weights follow $\beta_{\text{off}}(t) = (1-t)w$ and $\beta_{\text{eig}}(t) = tw$. We implement VICReg's variance term but weight it at zero here, since $\mathcal{L}_{\text{eig}}$ already handles collapse prevention. The full objective combines all three terms.

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \beta_{\text{off}}(t)\, \mathcal{L}_{\text{off}} + \beta_{\text{eig}}(t)\, \mathcal{L}_{\text{eig}} \tag{4}$$

Algorithm 1 (Appendix A) gives the training loop.

3.4 Measuring Linear Decodability

For each ground-truth factor (position, angle, velocity, angular velocity, as applicable), we fit a separate Ridge regression (McDonald, 2009) from μ to that factor, using an 80/20 train/test split, and report R^2 on the held-out split. This separates what the representation captures from what the policy does with it (Alain & Bengio, 2016).

Frame stacks near a trajectory's start are front-padded by repeating the first frame. We exclude these when probing velocity and angular velocity, since they carry no genuine motion, and probe position and angle on the full sample set.

3.5 Goal-Conditioned Actor

The policy conditions on the current latent state z_s and latent goal z_g, each passed through a fixed whitening transform (statistics from the encoder's training distribution) and a linear-LayerNorm-ReLU projection (Ba et al., 2016). We concatenate the two D-dimensional vectors, pass them through four further linear-LayerNorm-ReLU blocks at width $2D$, and map them to per-action logits.

3.6 Iterative Training Loop

Training runs N (default 10) outer iterations. In each iteration, the current policy collects trajectories, acting greedily except with probability ϵ (Sutton & Barto, 1998), annealed linearly from 1.0 to a floor of 0.1 over the first 5 iterations. New trajectories merge into a fixed 500-trajectory buffer through softmin-weighted retention, scored by minimum distance to the target goal embedding z_g (Schaul et al., 2015). Past capacity, we resample trajectories without replacement with probability $\propto \exp(-\text{score}/\tau)$, $\tau = 0.1$, favoring closer trajectories while keeping diversity.

Hindsight (s_i, g_i, a_i) triples pair each state with a goal g_i drawn later in the same trajectory (Andrychowicz et al., 2017), with horizon $h_i = t(g_i) - t(s_i)$, within a maximum lookahead. Rather than sample pairs uniformly, we upweight state and goal observations that fall in sparsely-populated bins.

$$w_{\text{state}}(i) = c_s\left[b(s_i)\right]^{-\alpha_{\text{state}}}, \qquad w_{\text{goal}}(i) = c_g\left[b(g_i)\right]^{-\alpha_{\text{goal}}} \tag{5}$$

where $b(\cdot)$ is the bin index and $c_s[\cdot]$, $c_g[\cdot]$ are per-bin counts, countering GCSL's tendency to overrepresent common, easily-reached configurations. We also reweight by inverse horizon length to favor shorter paths.

$$w_{\text{horizon}}(i) = h_i^{-\alpha_{\text{horizon}}}, \qquad w_i = \frac{w_{\text{state}}(i)\, w_{\text{goal}}(i)\, w_{\text{horizon}}(i)}{\sum_j w_{\text{state}}(j)\, w_{\text{goal}}(j)\, w_{\text{horizon}}(j)} \tag{6}$$

with $\alpha_{\text{state}} = \alpha_{\text{goal}} = 0.5$, $\alpha_{\text{horizon}} = 2.0$ in our main results. Algorithm 2 (Appendix A) gives the complete loop.

3.7 Data Collection

Encoder pretraining uses trajectories from states sampled uniformly at random across the environment's full valid range, not restricted to policy-reachable states, followed by a short window of random actions, rendered and JPEG-compressed (quality 98). Actor training and evaluation instead use on-policy rollouts (Section 3.6) under ϵ-greedy exploration.

3.8 Goal Specification and Encoding

We construct the goal by injecting a known target state and rendering forward from it. A static repeated rendering would misrepresent any goal with nonzero velocity, so where the environment has an available zero-force action, we step under it for $L - 1$ steps after injection to generate consistent motion. This applies to Acrobot and MountainCar, whose target states carry nonzero angular velocity and velocity respectively. CartPole and FrozenLake have no such action, but their targets need none, since FrozenLake's goal tile has no velocity concept and our CartPole target has zero velocity by construction. A static repeated frame is therefore exact, not approximate, for both. This L-frame window is encoded like any other observation, using μ as z_g. The underlying state vector is only ever a construction tool. *It is not itself an input to the policy or encoder anywhere in the pipeline.*

3.9 Baselines

Two families of controlled comparison hold all other pipeline components fixed. The first holds GCSL's actor-training procedure fixed and varies the representation it operates on. End-to-end pixel GCSL trains an encoder of the same architecture jointly with the actor, through the GCSL cross-entropy objective alone, with no reconstruction loss, no separate pretraining stage, and no regularization on the latent space. This baseline differs from our method in three respects at once, end-to-end instead of pretrained, no reconstruction instead of reconstruction, and unregularized instead of regularized, so it shows only that GCSL's objective alone fails in this setting, not that regularization specifically is the fix. GCBC separates that out. It holds our regularized encoder fixed instead, and replaces GCSL's iterative relabeling loop with a single batch containing the same total number of trajectories as full 10-iteration GCSL training, separating out the contribution of iteration.

The second family holds the encoder architecture and the motion-weighted reconstruction loss (Eq. 2) fixed, and varies only the regularization term, isolating its contribution directly through the decodability metric rather than through closed-loop performance. Vanilla VAE uses reconstruction alone (Kingma & Welling, 2013).

$$\mathcal{L}_{\text{vanilla}} = \mathcal{L}_{\text{rec}} \tag{7}$$

Beta-VAE adds a weighted KL term against a standard normal prior (Higgins et al., 2017), giving a combined loss with weight $\beta = 4$.

$$\mathcal{L}_{\text{KL}} = -\frac{1}{2}\sum_{d=1}^{D}\left(1 + \log\sigma_d^2 - \mu_d^2 - \sigma_d^2\right), \qquad \mathcal{L}_{\text{beta-VAE}} = \mathcal{L}_{\text{rec}} + \beta \cdot \mathcal{L}_{\text{KL}} \tag{8}$$

The remaining three vary $\mathcal{L}_{\text{off}}$ and $\mathcal{L}_{\text{eig}}$ (Section 3.3) individually and combined. VICReg-only and EigenReg-only each use a fixed weight ν (Bardes et al., 2021).

$$\mathcal{L}_{\text{VICReg}} = \mathcal{L}_{\text{rec}} + \nu \cdot \mathcal{L}_{\text{off}}, \qquad \mathcal{L}_{\text{EigenReg}} = \mathcal{L}_{\text{rec}} + \nu \cdot \mathcal{L}_{\text{eig}} \tag{9}$$

Our full method uses the curriculum $\beta_{\text{off}}(t)$, $\beta_{\text{eig}}(t)$.

$$\mathcal{L}_{\text{ours}} = \mathcal{L}_{\text{rec}} + \beta_{\text{off}}(t) \cdot \mathcal{L}_{\text{off}} + \beta_{\text{eig}}(t) \cdot \mathcal{L}_{\text{eig}} \tag{10}$$

We evaluate all five on the linear-decodability metric of Section 3.4.

4 EXPERIMENTS

4.1 ENVIRONMENTS

We evaluate on four control environments of increasing dynamical complexity, CartPole, FrozenLake, Acrobot, and MountainCar (Appendix C), simulated with Gymnasium. The encoder and policy observe only rendered RGB frames during both training and evaluation, and never see the underlying state vector. The state vector is available at training time only to compute the frequency-balanced sampling weights of Section 3.6 (Eq. 5), and is never an input to the encoder or policy. FrozenLake and CartPole have low-dimensional, largely disentangled factors. Acrobot has coupled two-joint dynamics. MountainCar poses a hard exploration problem, since a random or greedy policy cannot reach the goal without first moving away from it to build momentum.

4.2 EVALUATION PROTOCOL

We evaluate at two levels. First, we probe the encoder directly, fitting a Ridge regression from μ to each ground-truth factor (position, angle, velocity, angular velocity, as applicable) for each environment. We report R^2 on a held-out split and use a static-control condition to rule out spurious correlation. Second, we evaluate the closed-loop pipeline, rolling out the trained actor for 100 episodes per environment against a fixed goal, and report success rate (reaching the goal at any point) and average episodic reward.

4.3 MAIN RESULTS ON GOAL-REACHING FROM PIXELS

Table 1 reports closed-loop performance for our full pipeline across all four environments.

Success rate and reward improve across all 10 iterations, for every environment and every seed (Appendix G). CartPole and Acrobot improve gradually from the first iteration. FrozenLake and MountainCar stay near zero through iterations 5 and 6, then improve sharply, fitting with ϵ still being near its initial value during these early iterations. After that, annealing and accumulating relabeled data compound. MountainCar reaches its goal at evaluation despite no training rollout ever visiting that exact point. Appendix H digs deeper into this.

Table 1: Closed-loop goal-reaching performance using our VICReg+EigenReg encoder paired with GCSL, mean $\pm$ SE over 3 seeds.

Environment	Success rate (% $\pm$ SE)	Avg. reward (mean $\pm$ SE)
CartPole	94.4 ± 5.7	487.1 ± 8.7
Acrobot	98.7 ± 1.3	-386.7 ± 15.7
FrozenLake	100.0 ± 0.0	1.0 ± 0.0
MountainCar	100.0 ± 0.0	-548.7 ± 19.2

4.4 ISOLATING THE SOURCE OF THE IMPROVEMENT

Table 2 evaluates the two baselines from Section 3.9. End-to-end pixel GCSL tests whether GCSL's objective alone can produce a usable pixel representation, and GCBC separates out the contribution of iteration. Section 4.5 separates out the regularizer. Both comparisons run over 3 seeds.

Table 2: Closed-loop goal-reaching for end-to-end pixel GCSL (encoder trained jointly with the actor, no reconstruction or regularization) and for GCBC using our regularized encoder, mean $\pm$ SE over 3 seeds.

	End-to-end pixel GCSL		GCBC	
Environment	Success rate	Avg. reward	Success rate	Avg. reward
CartPole	0.0 ± 0.0	137.4 ± 27.8	0.0 ± 0.0	81.7 ± 6.5
Acrobot	0.0 ± 0.0	-500.0 ± 0.0	78.4 ± 10.7	-400.8 ± 11.9
FrozenLake	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
MountainCar	0.0 ± 0.0	-1000.0 ± 0.0	12.2 ± 7.4	-973.4 ± 24.8

4.5 COMPARING REPRESENTATION-LEARNING METHODS

Table 3 reports linear decodability at epoch 40 across representation-learning methods (full training curves in Appendix E). Our method converges within 2 to 3 epochs and stays flat from there. Beta-VAE is the only method that gets worse with more training. Section 5.1 compares linear against

nonlinear decodability for the same representations. The table's third column adds each encoder's closed-loop GCSL success rate, checking whether the decodability gap tracks goal-reaching.

Table 3: Linear decodability (avg. test R^2, Ridge probe) of static and dynamic factors, and closed-loop GCSL success rate (mean $\pm$ SE, averaged across all four environments and 3 seeds), across representation-learning methods at epoch 40. Static-factor R^2 is averaged across all four environments. Dynamic-factor R^2 is averaged across the three environments with a dynamic factor (CartPole, Acrobot, MountainCar), since FrozenLake's factors are grid coordinates with no velocity analogue. Encoder architecture and training data are held fixed across the five pretrained variants. The end-to-end encoder (top row) instead trains jointly with the GCSL actor on-policy, with no separate pretraining stage. We include it here to probe its representation directly, rather than only inferring it from closed-loop performance.

Method	Static factors ($R^2 \pm SE$)	Dynamic factors ($R^2 \pm SE$)	Success rate ($\% \pm SE$)
End-to-End Encoder	56.55 ± 8.3	3.5 ± 0.8	0.0 ± 0.0
Vanilla VAE (reconstruction only)	74.4 ± 18.2	63.9 ± 16.4	67.7 ± 13.8
Beta-VAE	48.2 ± 16.4	41.7 ± 15.7	0.0 ± 0.0
VICReg (covariance only)	86.7 ± 7.3	78.9 ± 8.3	73.5 ± 8.8
EigenReg (eigenvalue only)	75.7 ± 13.7	61.8 ± 14.6	65.1 ± 15.6
VICReg + EigenReg (ours)	93.3 ± 5.7	89.3 ± 8.4	98.3 ± 1.3

The end-to-end encoder's own row closes a gap that was previously only inferential. Its 0% closed-loop success (Table 2) had been the only evidence that it fails to capture the environment's factors, and direct decodability measurement here is consistent with that failure being representational, rather than only a policy-learning failure. Probed directly, its dynamic-factor decodability collapses to 3.5. Even its static-factor decodability (56.55) trails every pretrained variant except Beta-VAE. The single largest jump in the table comes from adding a reconstruction loss at all, taking dynamic-factor decodability from 3.5 to 63.9 with no regularizer. Reconstruction alone gets partway there, with 67.7% success, in line with its partial decodability (74.4). Beta-VAE drops to 0%, the same as end-to-end pixel GCSL, despite using the same pretrain-then-freeze setup, so decoupling the encoder from the policy is not enough on its own. What the encoder is trained to preserve appears to matter. Only VICReg-only (73.5%) beats vanilla VAE (67.7%) in closed-loop success. EigenReg-only (65.1%) does not, even though both improve decodability over reconstruction alone. Neither term is reliable on its own in these experiments. Combining them closes the gap to 98.3%, matching the near-ceiling numbers in Table 1.

5 Ablations

This section reports the ablation that bears most directly on the paper's central empirical claim, run on all four environments. Appendix F reports further ablations, including a second representation-level check and data-level and training-loop robustness checks run on Acrobot as a representative environment.

5.1 Representation-Level Ablation

A1. Linear vs. nonlinear decodability. A high linear-decodability score has two possible explanations. Our regularization could recover *linearly* accessible structure specifically, or it could just improve representation quality in a way that happens to correlate with linear decodability. To tell these apart, we build a controlled representation with low linear but high nonlinear decodability by design (Appendix D), and run the full GCSL loop on it in place of our encoder, across all four environments. Table 4 reflects this controlled representation, not our main method.

Nonlinear R^2 stays high throughout (87 to 94), yet CartPole and MountainCar collapse to 0% success. In these environments, high linear decodability tracks reliable GCSL in ways high nonlinear decodability alone does not, at least for larger, continuous state spaces. FrozenLake and Acrobot keep their success under the same change, but for two different reasons. FrozenLake's grid is small and fully discrete, an 8×8 space, and ϵ-random exploration alone covers most of it during training. So the actor's relabeled data spans nearly the full space, regardless of representation geometry, a

Table 4: Linear (Ridge) vs. nonlinear (MLP) decodability of the controlled sin/cos representation, and closed-loop GCSL performance when this representation replaces our encoder, mean ± SE over 3 seeds. The final column repeats the success rate of our VICReg+EigenReg encoder from Table 1 for direct comparison.

	Decodability		Goal Reaching		Ours
Environment	Linear R^2	Nonlinear R^2	Success	Avg. reward	Success
CartPole	43.8 ± 8.7	87.1 ± 4.6	0.0 ± 0.0	93.2 ± 16.7	94.4 ± 5.7
Acrobot	54.3 ± 5.3	88.6 ± 3.7	48.0 ± 17.3	-385.0 ± 20.1	98.7 ± 1.3
FrozenLake	64.7 ± 5.7	94.2 ± 2.6	100.0 ± 0.0	1.0 ± 0	100.0 ± 0.0
MountainCar	68.3 ± 8.2	93.1 ± 2.1	0.0 ± 0.0	-1000.0 ± 0	100.0 ± 0.0

property of the environment's exploration difficulty rather than the representation. Acrobot's partial recovery (48%) coincides with a property of the representation itself. Its factors are angles, and the sin/cos construction's periodic basis represents angles fairly naturally, unlike CartPole's Cartesian position or MountainCar's velocity. These are two separate explanations, and neither undermines the pattern we observe linking linear decodability to success, where nonlinear decodability alone does not appear to compensate.

6 Limitations

Our evaluation covers four simulated, low-dimensional, single-object control environments, with a static camera and no occlusion. Real robotic settings and cluttered or multi-object scenes remain untested, bounding what the current results support. Our evidence for an association between linear decodability and GCSL's success also rests on one controlled representation (Appendix D) and one regularizer family, VICReg with EigenReg. Ablation A1 separates linear from nonlinear decodability by construction, and closed-loop success tracks it in these environments. Further experiments on other architectures or regularizers would strengthen this characterization.

Encoder pretraining and the actor loop use pixels as input, but the sampling in Section 3.6 bins on the ground-truth state, not the latent code, information available in simulation but not always in the real world. We also evaluate against a single fixed goal per environment, not a goal distribution, leaving open how well this pattern extends to broader goal generalization. The hyperparameter α_{horizon} also differs by environment in our main results (Table 6), pointing to some sensitivity our method does not resolve on its own. Finally, we do not benchmark empirically against RIG, Skew-Fit, or contrastive RL, for the reason given in Section 2. Their value function could compensate for a poor latent space, so we cannot claim our method beats the strongest existing pixel-based methods.

7 Conclusion and Future Work

We asked whether GCSL's reward-free formulation is associated with linear decodability of the environment's factors in the state and goal space. Across the four environments we test, we find that it is. A naively trained pixel encoder does not give this property on its own. A VICReg+EigenReg regularizer restores it, paired with GCSL's unchanged relabeling objective and frequency-balanced, horizon-weighted pair sampling, giving reliable goal-reaching across all four. Controlled comparisons point to this improvement coming from the encoder's training objective specifically, not just from decoupling it from the actor, and to the combination of VICReg and EigenReg rather than either term alone. A linear-versus-nonlinear comparison is consistent with the recovered structure being specifically linear in these cases.

Future work includes testing this characterization in richer domains with occlusion, multi-object scenes, or real robotic settings, across a broader range of encoder architectures and regularizers, and aligning the latent space with language, as in GRIF (Myers et al., 2023), to specify goals through instructions rather than target images. Other directions include improving trajectory scoring and sampling for shorter paths, and characterizing how factor orthogonality interacts with covariance regularization's effect on decodability.

REPRODUCIBILITY STATEMENT

Full training and evaluation code is included in the supplementary material. This covers encoder pretraining, actor training, and probing scripts for all four environments. Appendix B lists complete hyperparameters per environment. All reported closed-loop results (Tables 1, 2, 4) use $N = 10$ outer GCSL iterations and a memory buffer of $K = 500$ trajectories, uniformly across environments. All reported encoder results (Tables 3, 5, 6) use a $D = 512$-dimensional latent space and $L = 5$-frame history stacks, uniformly across environments. One sampling hyperparameter, α_{horizon}, differs by environment in the reported configuration (Table 6). We note this explicitly, rather than presenting it as uniform.

AI USE STATEMENT

We used generative AI tools to implement parts of the codebase supporting the methods in this paper. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, propose or refine hypotheses, design or give feedback on research methodology or experiments, assist with translation, clean or reformat datasets, support qualitative or thematic data analysis, or interpret results. The authors performed all of these themselves.

We also used generative AI tools for brainstorming research directions, searching for and summarizing related work, editing code, and editing the manuscript text for clarity and readability. The authors reviewed and tested all AI-assisted code before use, and reviewed and edited all AI-assisted text for accuracy. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Kumar K Agrawal, Arnab Kumar Mondal, Arna Ghosh, and Blake Richards. α-ReQ: Assessing Representation Quality in Self-Supervised Learning by Measuring Eigenspectrum Decay. *Advances in Neural Information Processing Systems*, 35:17626–17638, 2022.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *International conference on machine learning*, pp. 3015–3024. PMLR, 2021.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank. In *International conference on machine learning*, pp. 10929–10974. PMLR, 2023.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

Junlin He, Jinxiao Du, and Wei Ma. Preventing dimensional collapse in self-supervised learning via orthogonality regularization. *Advances in Neural Information Processing Systems*, 37:95579–95606, 2024.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*, 2021.

Tom Jurgenson and Aviv Tamar. Goal-conditioned supervised learning with sub-goal prediction. *arXiv preprint arXiv:2305.10171*, 2023.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pp. 5639–5650. PMLR, 2020.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pp. 4114–4124. PMLR, 2019.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pp. 1113–1132. Pmlr, 2020.

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. *arXiv preprint arXiv:2210.00030*, 2022.

Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Tingfan Wu, Jay Vakil, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *Advances in Neural Information Processing Systems*, 36:655–677, 2023.

Gary C McDonald. Ridge regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1 (1):93–100, 2009.

Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in neural information processing systems*, 34:24379–24391, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Vivek Myers, Andre Wang He, Kuan Fang, Homer Rich Walke, Philippe Hansen-Estruch, Ching-An Cheng, Mihai Jalobeanu, Andrey Kolobov, Anca Dragan, and Sergey Levine. Goal representations for instruction following: A semi-supervised language interface to control. In *Conference on Robot Learning*, pp. 3894–3908. PMLR, 2023.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pp. 3878–3887. PMLR, 2018.

Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *international conference on machine learning*, pp. 17359–17371. PMLR, 2022.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International conference on machine learning*, pp. 9870–9879. PMLR, 2021.

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

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl. *arXiv preprint arXiv:2202.04478*, 2022.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pp. 12310–12320. PMLR, 2021.

A Training Algorithms

Algorithm 1 details the encoder pretraining loop described in Section 3.3. Algorithm 2 details the GCSL actor training loop on the frozen encoder, described in Section 3.6.

B Hyperparameters

Table 5 lists the encoder pretraining hyperparameters (Algorithm 1). Table 6 lists the GCSL actor training hyperparameters (Algorithm 2). Both are uniform across environments except where noted. One setting is environment-specific by design, the hindsight lookahead window `MAX_DEPTH`. Table 7 gives this value per environment. With $B = 64$, $D = 512$, Σ has rank at most 63. So 449 eigenvalues are zero by construction, and $\mathcal{L}_{\text{eig}}$ (Eq. 3) carries an unavoidable floor of $\approx 0.877\,\epsilon^2$.

Algorithm 1 Encoder Training with Geometric Regularization

Input: dataset $\mathcal{D}$ of randomly-collected frame stacks, epochs E, base regularization weight w
for $e = 1$ to E **do**
  $t \leftarrow e/(E-1)$
  $\beta_{\text{off}} \leftarrow (1-t)\, w, \quad \beta_{\text{eig}} \leftarrow t\, w$
  for each batch $x \sim \mathcal{D}$ **do**
    $\mu, \log \sigma^2 \leftarrow \text{Encoder}(x)$
    $z \leftarrow \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$
    $\hat{x} \leftarrow \text{Decoder}(z)$
    compute $\mathcal{L}_{\text{rec}}$ (motion-weighted L1, Eq. 2)
    $\Sigma \leftarrow$ batch covariance of μ (Section 3.3)
    compute $\mathcal{L}_{\text{off}}$, $\mathcal{L}_{\text{eig}}$ from Σ (Eq. 3)
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{rec}} + \beta_{\text{off}}\, \mathcal{L}_{\text{off}} + \beta_{\text{eig}}\, \mathcal{L}_{\text{eig}}$
    update Encoder, Decoder via $\nabla \mathcal{L}$
  end for
end for

Algorithm 2 GCSL Actor Training on Frozen Encoder

Input: frozen Encoder, target state s^*, outer iterations N, memory capacity K
$z_g \leftarrow \text{Encoder}(\text{rollout-from}(s^*))$
initialize actor π_θ, memory buffer $\mathcal{M} \leftarrow \emptyset$
for $n = 1$ to N **do**
  $\epsilon_n \leftarrow \text{anneal}(n)$ ▷ $1.0 \rightarrow 0.1$ over first 5 iterations
  collect trajectories $\{\tau\}$ under π_θ with ϵ_n-greedy exploration
  merge $\{\tau\}$ into $\mathcal{M}$; if $|\mathcal{M}| > K$, resample by softmin retention
  construct hindsight triples (s_i, g_i, a_i) from $\mathcal{M}$, with $g_i = s_t, t > i$
  compute pair weights $w_i \propto w_{\text{state}}(i)\, w_{\text{goal}}(i)\, w_{\text{horizon}}(i)$
  for 10 epochs **do**
    sample batch $\{(s_i, g_i, a_i)\}$ according to w_i
    $z_{s_i}, z_{g_i} \leftarrow \text{Encoder}(s_i), \text{Encoder}(g_i)$
    $\hat{a}_i \leftarrow \pi_\theta(z_{s_i}, z_{g_i})$
    update π_θ via cross-entropy$(\hat{a}_i, a_i)$
  end for
end for

This is harmless. Our environments have at most four ground-truth factors, far fewer than the 63 directions Σ can still resolve.

We bin the ground-truth state rather than the latent, to avoid a curse-of-dimensionality problem. Uniform binning is intractable over 512 dimensions, but works over the 2 to 6 true state dimensions. This shapes sampling weights only, never the encoder or policy input. A latent- or image-derived binning scheme is left to future work.

C Environment Details

Figure 2 shows one rendered frame from each of the four control environments evaluated in this paper, along with each environment's goal, underlying factors, and factor ranges.

D Controlled Sin/Cos Representation

The controlled representation used in Ablation A1 (Section 5.1) encodes each state's normalized coordinate $x \in [0, 1]$ as high-frequency sinusoids.

$$\phi(x) = \left[\sin(2^k \pi x),\ \cos(2^k \pi x)\right]_{k=0}^{K-1}, \tag{11}$$

A fixed, untrained random projection W mixes these.

$$z = W\, \phi(x) + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma^2 I). \tag{12}$$

Table 5: Encoder pretraining hyperparameters, uniform across all four environments unless noted.

Frame history length (L)	5
Input resolution	256×256 px
JPEG quality	98 (85 for FrozenLake)
Latent dimensionality (D)	512
Encoder pretraining batch size (B)	64
Encoder pretraining dataset size	10,000 random-state trajectories
Foreground/background weights ($w_{\text{fg}}, w_{\text{bg}}$)	4.0, 1.0
Motion threshold (τ)	0.3
Motion-normalization floor (ϵ_{floor})	0.02
Eigenvalue floor (ϵ_{eig}), Eq. 3	1.0
Curriculum base weight (w), ours only	0.1
Fixed regularization weight (ν), VICReg-only & EigenReg-only baselines	1.0
Training epochs	40
Learning rate	1e-5

Table 6: GCSL actor training hyperparameters. Most are uniform across environments; those noted are environment-specific by design.

Outer iterations (N)	10
Trajectories collected per outer iteration	500
Memory buffer capacity (K)	500
ϵ schedule	$1.0 \rightarrow 0.1$, annealed over first 5 iterations
Softmin retention temperature (τ)	0.1
Pair-sampling α_{state}, α_{goal}	0.5, 0.5 (uniform)
Pair-sampling α_{horizon}	2.0 (1.0 for MountainCar)
Max. hindsight pairs per goal	50 (CartPole, MountainCar); 20 (FrozenLake, Acrobot)
State/goal binning resolution (n_{bins})	10, for continuous-state environments (CartPole, Acrobot, MountainCar); exact per-tile counts for FrozenLake's 64 discrete states
State/goal binning domain	Ground-truth environment state, never the encoder's latent or an image; used only to set sampling weights, never as a model input
Hindsight lookahead window (`MAX_DEPTH`)	Environment-specific, see Table 7
Actor optimizer	AdamW
Actor learning rate	1×10^{-4}
Actor batch size	128
Epochs per outer iteration	10

At high frequency, ϕ is strongly non-monotonic. This defeats a linear probe. It stays a fixed, smooth, deterministic function that a nonlinear probe can learn directly. Noise η prevents trivial interpolation of a small number of distinct states. This gives the linear/nonlinear dissociation by design, independent of any property specific to our encoder.

E Decodability Training Curves

Figure 3 shows static-factor decodability across all 40 training epochs, not just the epoch-40 endpoint reported in Table 3. Our method converges within 2 to 3 epochs and stays flat after that. VICReg-only climbs more slowly on FrozenLake and CartPole. It eventually catches up to our plateau by epoch 40. On Acrobot, all four methods end up in a similar range by epoch 15. Vanilla VAE tracks us closely on MountainCar, but plateaus well below us on FrozenLake and CartPole. Beta-VAE is the only method that gets worse over training instead of converging. This shows most clearly on FrozenLake and MountainCar.

Table 7: Hindsight lookahead window, set per environment to match its typical episode length.

Environment	MAX_DEPTH
CartPole	32
Acrobot	500
FrozenLake	64
MountainCar	1000

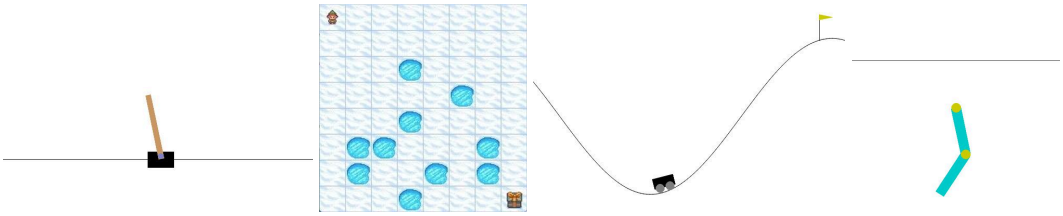

Figure 2: The four control environments evaluated in this paper, observed only as rendered pixels during training and evaluation. (a) **CartPole**. Balance the pole upright by keeping cart position $x \in [-2.4, 2.4]$ m and pole angle $\theta \in [-12°, 12°]$ within bounds. Velocities $\dot{x}$ and $\dot{\theta}$ are unbounded in the observation space. (b) **FrozenLake**. Navigate an 8×8 discrete grid from a fixed start to a goal cell, avoiding holes. The underlying factors are the agent's grid coordinates $(x, y) \in \{0, \ldots, 7\}^2$. (c) **MountainCar**. Drive an underpowered car up a valley to a target position. This requires the agent to first move away from the goal to build momentum. Underlying factors are car position $x \in [-1.2, 0.6]$ and velocity $\dot{x} \in [-0.07, 0.07]$. (d) **Acrobot**. Swing the free end of a two-link arm above a target height by actuating only the joint between the links. Underlying factors are the two joint angles $\theta_1, \theta_2 \in [-\pi, \pi]$ and angular velocities $\dot{\theta}_1 \in [-4\pi, 4\pi]$, $\dot{\theta}_2 \in [-9\pi, 9\pi]$ rad/s.

F Additional Ablations

This appendix reports one further representation-level ablation (A2), plus data-level and training-loop ablations (B1 to C3). These supplement Section 5.1's central ablation (A1). A2 runs on all four environments, like A1. Unless noted otherwise, B1 to C3 run on Acrobot as a representative environment.

F.1 Representation-Level Ablation

A2. Motion-weighted reconstruction. We replace the motion-weighted L1 loss with plain, unweighted L1. We hold the curriculum weight fixed at its main-result value and report decodability separately for static and dynamic factors. This isolates whether motion weighting specifically recovers dynamic factors, as claimed, rather than improving reconstruction quality overall.

Table 8: Motion-weighted reconstruction substantially improves decodability of dynamic factors, with a smaller gain on static factors, for our VICReg+EigenReg encoder.

Method	Avg. R^2 (static)	Avg. R^2 (dynamic)
VICReg + EigenReg (no motion weighting)	81.8	68.4
VICReg + EigenReg (Motion-weighted)	93.3	89.3

F.2 Data-Level Ablations

B1. Input resolution. We compare encoder training at reduced input resolution and JPEG quality against our main-result setting (256 px, quality 98). We report decodability at each. This isolates whether resolution, rather than latent capacity, is the main lever for recovering dynamic factors. Table 9 shows the reduced setting costs 14.6 points of static decodability, but 28.1 points of dynamic decodability. Resolution matters most for the dynamic factors.

B2. Dataset size. We train the encoder on subsets of the random-collection dataset (10%, 30%, 100% of trajectories) and report decodability at each size. This shows how much state-space cov-

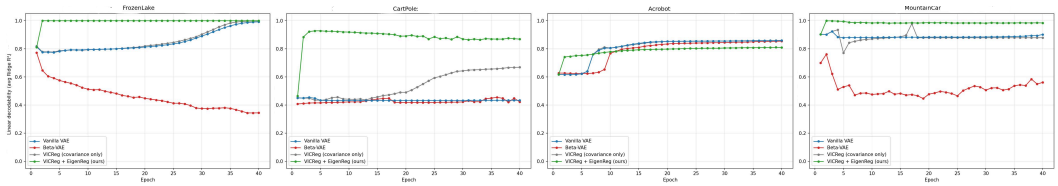

Figure 3: Linear decodability (avg. Ridge R^2 on static factors) over 40 training epochs, across all four environments, for our VICReg+EigenReg encoder against three alternatives.

Table 9: Reduced input resolution and JPEG quality substantially degrade decodability, particularly for dynamic factors.

Setting	Avg. R^2 (static)	Avg. R^2 (dynamic)
Low resolution and quality (84px, Q85)	78.7	61.2
Main-result setting (256px, Q98)	93.3	89.3

erage the regularization needs to work well. Table 10 shows decodability rising with dataset size. Most of the gain comes by 30% of the data.

Table 10: Decodability improves with encoder training set size, with diminishing returns beyond 30% of the full dataset.

Fraction of full dataset	10%	30%	100%
Avg. R^2	84.4	88.1	91.3

B3. Non-uniform state coverage. Our main-result encoder trains on data collected by sampling states uniformly at random across the environment's full valid range. Here we instead train on data from a narrower, imbalanced region of state space, and report the resulting decodability. This tests whether uniform coverage during encoder pretraining is necessary for the regularization to work, rather than an arbitrary design choice. Table 11 shows imbalanced coverage fits its training region better. But it generalizes far worse to held-out states (68.2 vs. 91.3 test R^2).

F.3 Training-Loop Ablations

C1. Memory buffer size. We sweep the retention buffer capacity (100, 200, 500, 1000 trajectories) and report closed-loop success rate and reward. This shows how sensitive the softmin-retention scheme is to buffer size. Table 12 shows success jumping between 200 and 500 trajectories, then plateauing. This is why we use $K = 500$.

C2. Frequency-balanced sampling. We disable the inverse-frequency state and goal reweighting in the pair sampler, using uniform sampling over all valid hindsight pairs instead. We report closed-loop success rate to test whether this component contributes to goal-reaching performance, as claimed. Table 13 shows that removing it collapses success from 98.4% to 17.5%.

C3. Horizon weighting. We sweep $\alpha_{\text{horizon}} \in \{0, 0.5, 1.0, 2.0\}$ and report closed-loop success rate at each value. This directly measures the effect of this parameter, rather than relying on an unverified claim. Table 14 shows success rising monotonically with the exponent, from 18.0% with no horizon weighting to 98.0% at $\alpha_{\text{horizon}} = 2.0$.

G Per-Iteration Learning Curves

Section 4.3 reports closed-loop goal-reaching performance at the final (10th) outer training iteration. Figure 4 shows the full trajectory across all 10 iterations, for both success rate and average episodic reward, on each of the four environments.

Table 11: Imbalanced encoder training data yields higher train decodability but substantially worse held-out generalization than uniform coverage.

	Avg. R^2 (train)	Avg. R^2 (test)
Imbalanced coverage	96.8	68.2
Uniform coverage (ours)	92.4	91.3

Table 12: Goal-reaching success improves with memory buffer size, plateauing beyond 500 trajectories.

Buffer size	100	200	500	1000
Success rate	83.4%	83.7%	98.0%	98.4%

Table 13: Frequency-balanced pair sampling appears necessary for reliable goal-reaching in this setting; uniform sampling collapses success rate.

	Success rate
Uniform pair sampling	17.5%
Frequency-balanced sampling (ours)	98.4%

Table 14: Goal-reaching success rate increases sharply with horizon-weighting exponent, consistent with its role in the sampling scheme.

α_{horizon}	0	0.5	1.0	2.0
Success rate	18.0%	57.5%	61.7%	98.0%

H Directed Exploration in MountainCar and FrozenLake

MountainCar succeeds differently from the other three environments. It reaches the goal in every evaluation episode, yet never finishes in under 200 steps. Reward variance is far higher (min -968, max -316). A balanced action distribution rules out a degenerate shortcut.

Its fixed evaluation target is a single point in a continuous two-dimensional state space. No training rollout ever visits that exact point. So no hindsight-relabeled training pair is ever literally conditioned on it. The policy's reliable goal-reaching at evaluation time is therefore best explained by generalization. It generalizes from nearby, non-identical states encountered during training, through proximity in the learned latent space, rather than direct exposure to the literal goal.

FrozenLake's discrete grid lets us build a controlled, directly visualizable version of this same phenomenon. We collect FrozenLake trajectories as in our main results, but explicitly exclude the goal cell from every hindsight-relabeled training pair. Unlike MountainCar's continuous-space near-miss, this exclusion is not a structural inevitability. It is a deliberate removal. It ensures the actor never receives a supervised training signal for the goal state at any point in training, even though the environment itself is unrestricted and the goal tile remains physically enterable during rollouts. Figure 5 shows the resulting state-visitation pattern across all 10 training iterations. Despite the complete absence of goal-reaching supervision, visitation develops transient, non-monotonic structure rather than staying uniform. Iterations 5 to 8 show concentration along the top row of the grid. Iteration 9 shows a shift toward the lower-left region, tracing a route-like path that skirts the hole cluster near the start without extending to the goal itself, which stays at zero visitation throughout. We read this as evidence that GCSL's relabeling and pair-weighting mechanics alone can produce structured, if unstable, exploration behavior purely from latent-space proximity, independent of ever observing task success. It also suggests that MountainCar's reliable but inefficient goal-reaching reflects the same underlying generalization mechanism, operating in a continuous setting.

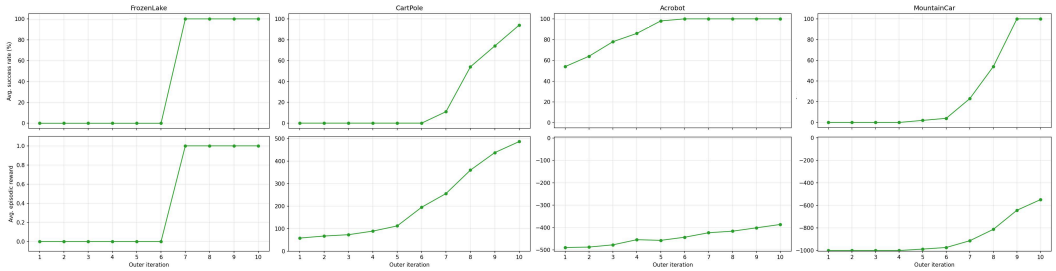

Figure 4: Success rate (top row) and average episodic reward (bottom row) across all 10 outer GCSL training iterations, for our VICReg+EigenReg encoder, on each of the four environments. Reward scales differ by environment. FrozenLake ranges from 0 to 1, CartPole from 0 to 500, and Acrobot and MountainCar use negative rewards, where less negative is better. So reward panels are not directly comparable across columns. CartPole and Acrobot improve gradually from the first iteration. FrozenLake and MountainCar remain near zero through iterations 5 to 6, then improve sharply. This fits with ϵ-greedy exploration still being near its initial value of 1.0 during these early iterations. After that, the annealing schedule and accumulating relabeled data begin to compound.

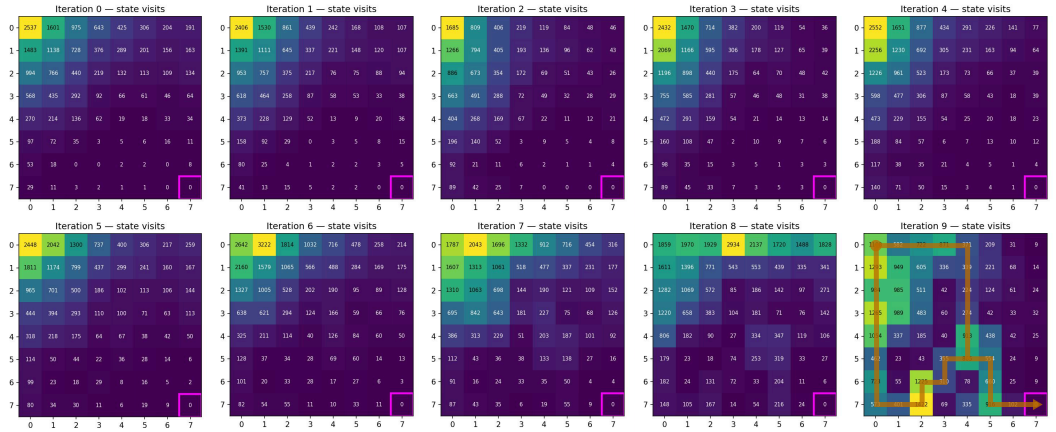

Figure 5: State-visitation counts across all 10 GCSL training iterations on FrozenLake, with goal-reaching deliberately excluded from every hindsight-relabeled training pair. The overlaid line on Iteration 9 marks an example trajectory taken by the policy during this iteration's rollouts. It shows route-like structure through the lower-left region of the grid, skirting the nearby hole cluster. This shows that GCSL's relabeling and pair-weighting mechanics can produce coherent, route-like exploration structure in the latent space, even when the algorithm has never observed a successful goal-reaching trajectory during training.