LATENT SPACE PERTURBATIONS IMPROVE VISUAL GENERALIZATION IN REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual generalization in reinforcement learning (RL) means keeping the same task performance when the visuals change, for example the background, lighting, or object appearance, while the dynamics and rewards stay the same. Humans handle such changes well, but RL agents often lose a lot of performance under these visual shifts. Inspired by efficient coding theory from neuroscience, we study how visual shifts change the encoder latent space. We find that encoders trained in fixed visual settings learn a mostly low-dimensional latent space: out-of-distribution (OOD) visuals mainly cause mean and variance shifts along the top principal components of the training latent space, instead of creating new directions. We validate this on our diagnostic environment (LSP-Env) with controlled tests for background changes, distractors, and agent appearance changes, and we also validate this on Procgen. Based on these results, we propose **Latent Space Perturbation (LSP)**, a PCA-guided method that adds variance-scaled Gaussian noise along dominant latent directions during training. Across LSP-Env, Procgen, and RL-ViGen, LSP improves OOD performance without hurting in-distribution performance, and it does so without changing the encoder or using pixel-level augmentation.

1 INTRODUCTION

Reinforcement learning (RL) agents often generalize poorly to visual conditions unseen during training, even when task dynamics and rewards remain unchanged (Packer et al., 2018; Kirk et al., 2023; Raileanu et al., 2020; Li et al., 2021; Wang et al., 2025; Klepach et al., 2025). Such failures suggest that learned representations capture task-irrelevant visual factors, causing policies to degrade under changes in background, lighting, or appearance.

Existing approaches mainly rely on input-space augmentation, architectural regularization, or invariant representation learning, but provide limited insight into how OOD visual shifts alter the latent representations mediating perception and control. Understanding these changes in latent statistics and geometry is therefore central to diagnosing visual generalization failures and motivates two questions:

1. How do OOD visual changes deform the geometry and statistics of the encoder latent space?
2. How can training be modified to prevent policy degradation under such distributional shifts?

Inspired by the broader intuition of efficient coding (Attneave, 1954; Barlow et al., 1961), we examine whether visual variability in RL representations is concentrated along a small number of latent directions. Empirically, we find that convolutional encoders trained under fixed visual distributions exhibit a low-rank structure, with the leading principal components capturing most of the train-time latent variability. Under controlled protocols, OOD visual perturbations manifest primarily as mean and variance shifts along existing train-time directions rather than introducing new high-variance components. We further verify that this geometry persists for task-trained CNNs under naturalistic video-background shifts and for a frozen DINOv2 ViT under mild-to-moderate shifts. Although the latent marginals are not strictly Gaussian, most of the train-to-evaluation discrepancy is still explained by changes in location and scale, with higher-order effects remaining secondary.

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.

This suggests a direct mitigation. If OOD behavior arises from structured shifts along dominant latent directions, exposing the policy to such perturbations during training should improve inference-time robustness. We formalize this in **Latent Space Perturbation (LSP)**, a PCA-guided method that injects variance-scaled Gaussian noise into encoder outputs along leading train-time principal components. By scaling noise by latent variance, LSP matches the geometry of OOD shifts using latent statistics alone without modifying the encoder or applying pixel-level augmentation. Importantly, LSP does not require task-relevant and nuisance factors to be disentangled into separate principal components.

To evaluate visual robustness systematically, we build a diagnostic task suite (LSP-Env) with configurable protocols that isolate distinct sources of visual shift: background variation, non-agent distractors, and agent appearance. Across LSP-Env, Procgen (Cobbe et al., 2020), and RL-ViGen (Yuan et al., 2023), LSP improves OOD robustness without sacrificing in-distribution performance. On Procgen, it outperforms the pixel-augmentation baseline RAD on four of five tasks and matches or exceeds IBAC-SNI on four of five tasks at the final checkpoint. RL-ViGen shows that the gains extend to continuous-control robotic manipulation. The goal is not to learn an explicitly invariant encoder, but to regularize the policy against latent shifts induced by visual changes.

We summarize our main contributions as follows:

1. We empirically characterize how visual distribution shift affects encoder latents, showing that OOD effects predominantly manifest as structured drift within the leading train-time latent subspace, and verify this geometry across controlled and naturalistic shifts and beyond CNN encoders.
2. We propose **Latent Space Perturbation (LSP)**, which injects PCA-aligned noise during training to improve visual robustness.

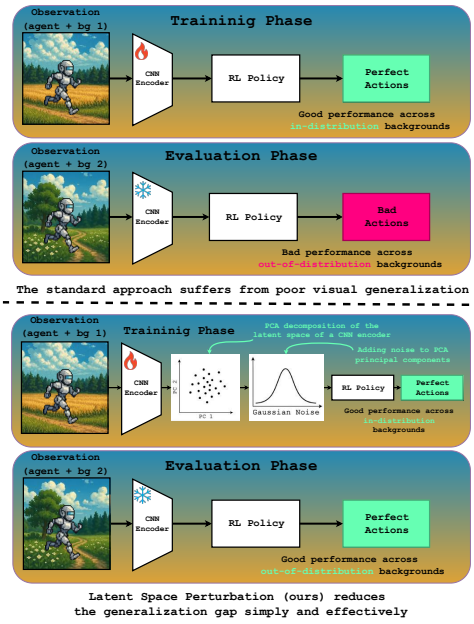


Figure 1: **Latent Space Perturbation (LSP) for visual OOD generalization in RL. Top:** Standard vision-based RL pipeline. A CNN encoder and policy trained under a fixed visual distribution perform well in-distribution but fail under evaluation-time visual shifts, as latent representations move outside the region seen during training. **Bottom:** Latent Space Perturbation (LSP). During training, Gaussian noise is injected along PCA-identified latent principal components, training the policy to tolerate dominant visual variation and yielding robust performance under both in-distribution and OOD conditions.

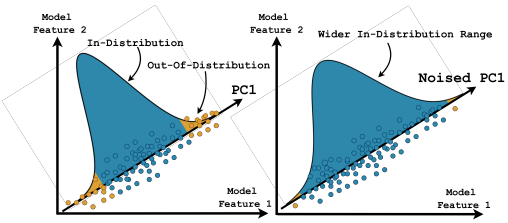


Figure 2: **Geometric intuition for Latent Space Perturbation. Left:** OOD observations shift latents along dominant principal components, moving outside the in-distribution region and causing poor generalization. **Right:** LSP injects noise along these directions during training, expanding in-distribution support and improving robustness.

2 Related Work

Prior work on visual generalization in RL falls mostly into four directions: (i) observation-space augmentation, (ii) masking and object-centric generalization, (iii) representation regularization and auxiliary objectives, and (iv) architectural and optimization-level modifications.

Observation-Space Augmentation. RAD (Laskin et al., 2020) shows that simple image transformations improve robustness, while automated methods learn task-specific augmentation policies (Raileanu et al., 2020). Trajectory-level methods synthesize novel visual inputs via style trans-

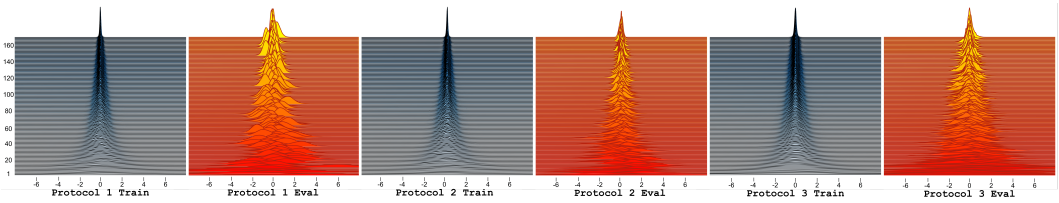


Figure 3: **Plots of latent distributions along the principal components of the train basis.** The blue density functions correspond to the distributions of train latents in the basis formed by the eigenvectors obtained via PCA on the train data itself. The red density functions show the distributions of eval OOD latents projected onto the train basis. Notably, the eval projections remain predominantly single-peaked, while exhibiting shifts in the mean, variance, and higher-order distributional shape.

fer (Thinker (Rahman & Xue, 2022)) or environment interpolation (MixReg (Wang et al., 2020)). SVEA (Hansen et al., 2021) studies instability caused by strong augmentations and improves stability of Q-learning under augmented observations. These approaches assume training-time transformations span evaluation-time variation and can fail under qualitatively different shifts. In contrast, our approach operates in latent space, perturbing statistically dominant directions rather than applying pixel-level transformations.

Masking and Object-Centric Generalization. Several works improve robustness by filtering task-irrelevant visual information before policy learning. VAI (Wang et al., 2021) learns distraction-invariant foreground masks through unsupervised visual attention, while ViSaRL (Liang et al., 2024) incorporates human saliency priors into visual representations. SAM-G (Wang et al., 2023) leverages Segment Anything Model (SAM) for task-relevant object segmentation, and DeGuV (Pham et al., 2025) uses depth masking to suppress irrelevant visual variation. These methods improve robustness through observation filtering and object-centric representations, whereas our work studies how OOD shifts deform latent geometry itself and regularizes the policy directly in latent space.

Representation Regularization and Auxiliary Objectives. IBAC-SNI (Igl et al., 2019) combines an information bottleneck with selective noise injection, while CTRL (Mazoure et al., 2021), ATC (Stooke et al., 2021), and ITER (Igl et al., 2020) use self-supervised or contrastive objectives to decouple representation learning from policy optimization. CLOP (Bertoin & Rachelson, 2022) and architectural scaling choices such as frame stacking and increased channel capacity also reduce generalization gaps (Cobbe et al., 2020; Jesson & Jiang, 2024). Recent work also explored disentangled latent representations for zero-shot generalization (Batra & Sukhatme, 2024). None of these methods analyze which latent directions drive OOD failures; our approach directly characterizes these axes and trains the policy to tolerate perturbations along them.

Architectural and Optimization-Level Decoupling. PPG (Cobbe et al., 2021) decouples actor and critic optimization, while IDAAC (Raileanu & Fergus, 2021) adds auxiliary losses for visual invariance. IPO (Sonar et al., 2021) and REx (Krueger et al., 2021) (building on IRM (Arjovsky et al., 2019)) penalize spurious correlations via multi-domain objectives. These methods promote invariance through optimization constraints without analyzing how visual shifts manifest in latent geometry, which is the central focus of our work.

3 Background

This section formalizes visual generalization in RL and introduces the notation and assumptions used throughout the paper. We model visual distribution shift as changes in the observation process that preserve underlying task dynamics and rewards, allowing us to isolate failures arising from perception rather than control. This formalization provides the foundation for our latent-space analysis and the method proposed in subsequent sections.

3.1 Visual Generalization in RL

We consider a family of partially observable Markov decision processes (POMDPs) parameterized by an observation function: $\mathcal{M}(\Omega) = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P, R, \Omega)$, where $\mathcal{S}$ denotes the *latent environment state space* encoding all task-relevant variables, $\mathcal{A}$ is the action space, $\mathcal{O}$ the observation space, $P(s' \mid s, a)$ the state transition kernel, and $R(s, a)$ the reward function. The observation function $\Omega : \mathcal{S} \to \mathcal{P}(\mathcal{O})$ maps latent states to probability distributions over visual observations, where $\mathcal{P}(\mathcal{O})$ denotes the set of probability measures on the observation space $\mathcal{O}$. Throughout, we assume all environments in the family share the same $(\mathcal{S}, \mathcal{A}, P, R)$ and differ only in the Ω. That is, visual variation affects perception but does not alter underlying task dynamics or reward structure.

Let Ω_{train} denote the observation function used during training, and let Ω_{eval} denote an evaluation-time observation function that differs from Ω_{train} only in task-irrelevant visual factors such as texture, background, lighting, or appearance, while preserving the underlying task semantics encoded by (P, R). We assume the optimal policy $\pi^\star(a \mid s)$ is invariant to observation function Ω.

For $\pi(a \mid o)$ and Ω, we define the expected discounted return as

$$J_\Omega(\pi) \triangleq \mathbb{E}_{\substack{s_0 \sim \rho, \\ s_{t+1} \sim P(a|s_t, a_t), \\ o_t \sim \Omega(o|s_t), \\ a_t \sim \pi(a|o_t)}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \tag{1}$$

where ρ denotes the initial state distribution. The training objective is $\pi^\star \in \arg\max_\pi J_{\Omega_{\text{train}}}(\pi)$.

Definition (Visual Generalization). A policy π is said to *generalize visually* if its expected return remains unchanged when the observation function is changed in task-irrelevant visual aspects. Formally, given a training observation function Ω_{train} and an evaluation-time observation function Ω_{eval} that preserves the same underlying dynamics and rewards (P, R), we define the visual generalization gap as

$$\Delta_{\text{vis}}(\pi) \triangleq \left| J_{\Omega_{\text{train}}}(\pi) - J_{\Omega_{\text{eval}}}(\pi) \right|. \tag{2}$$

The policy π is said to generalize visually if $\Delta_{\text{vis}}(\pi) \leq \varepsilon$ for a small $\varepsilon > 0$.

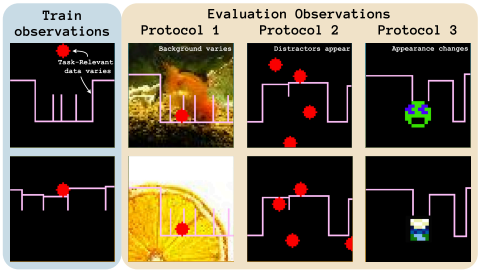


Figure 4: **Train and evaluation observations used for latent space analysis in LSP-Env. Train:** Task-relevant factors (agent position and layout) vary while visual appearance (background, distractors, and agent appearance) is fixed. **Eval:** Task-relevant state is fixed and task-irrelevant visual factors varied across three protocols (background, distractors, and appearance), isolating latent variation induced purely by visual changes.

Task Formalization. The agent implements a policy $\pi(a \mid o)$ that operates on visual observations through an intermediate latent representation produced by a visual encoder $f : \mathcal{O} \to \mathbb{R}^D$. That is, action selection depends on observations only through $z = f(o)$. Since the optimal policy in latent state space, $\pi^\star(a \mid s)$, is invariant across observation functions Ω, the learning problem is perceptual rather than dynamical. Failures of visual generalization arise when visually induced observation shifts cause the latent distribution induced by $z = f(o)$ to move outside the high-density region encountered during training: $o \sim \Omega_{\text{train}}(o \mid s) \longrightarrow o' \sim \Omega_{\text{eval}}(o \mid s)$.

Problem Statement. Given a fixed encoder-policy architecture trained under Ω_{train}, our objective is to design a training procedure that reduces the visual generalization gap $\Delta_{\text{vis}}(\pi)$ defined in Equation 2, without modifying the transition dynamics, reward function, or encoder architecture. In this work, we study how visually induced OOD shifts from Ω_{eval} manifest in the geometry and statistics of the encoder's latent space created by latents $z = f(o')$ and study how visually induced OOD shifts manifest in the geometry and statistics of the encoder's latent space, with the goal of designing training procedures that stabilize the return functional $J_\Omega(\pi)$ under such shifts.

4 Benchmarking Visual Generalization

To examine the hypothesis that visually induced distribution shifts exhibit a structured geometry in encoder latent space, we construct a simple diagnostic environment, **LSP-Env** (Figure 4). LSP-Env is a lightweight 2D platformer in which the agent moves right as quickly as possible while avoiding obstacles, with fixed dynamics and objectives and controllable appearance factors. Observations are 64×64 RGB images and actions are 2D continuous displacements (left/right movement speed and jump force). The reward encourages forward progress and severely penalizes vertical motion, inducing a consistent "move-right" objective. Within this fixed task, we can isolate task-irrelevant visual variation by independently changing the background image, adding non-controllable agent-like distractors, and altering the controlled agent's appearance (texture). Details are in Appendix B.

We then evaluate the resulting method on established visual generalization benchmarks to test whether conclusions from the controlled setting transfer to more complex, less controlled environments. **Procgen** (Cobbe et al., 2020) provides procedurally generated environments with substantial visual diversity, serving as a stress test under uncontrolled appearance variation at scale. **RL-ViGen** (Yuan et al., 2023) comprises popular environments adapted for visual generalization evaluation; we use the Robosuite Door and Lift tasks to assess LSP in a challenging continuous-control robotics setting. These environments also structure the remainder of the work. In section 5,

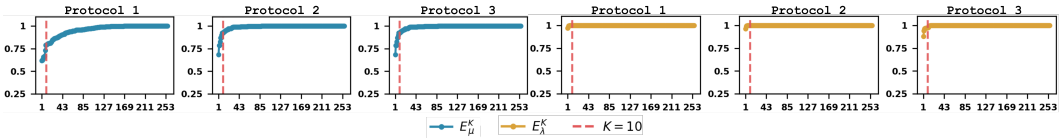


Figure 6: Cumulative fraction of OOD shift energy captured by the top-K train PCs. **Left:** mean-shift E_μ^K; **Right:** variance-shift E_λ^K. Both saturate quickly; $K = 10$ already explains nearly all shift.

we use **LSP-Env** to characterize how appearance-only shifts change the encoder latent distribution and to extract constraints on the kinds of latent perturbations the policy should tolerate. In section 6, we instantiate these constraints as **Latent Space Perturbation (LSP)**, a training-time regularizer at the representation-to-control interface. Finally, subsection 6.3 evaluates whether LSP reduces the visual generalization gap on LSP-Env and whether the gains transfer to Procgen and RL-ViGen.

5 Latent Space Analysis

This section analyzes how appearance-only shifts change the encoder latent distribution when dynamics and rewards are fixed, with the aim of motivating the training procedure in section 6. We measure three effects: shifts in the latent mean, changes in the variance spectrum, and drift of the dominant PCA subspace. These diagnostics distinguish two regimes: OOD variation that remains within the dominant train-time subspace (suggesting extrapolation by the policy head), versus OOD variation that introduces new dominant directions (suggesting a representation-level limitation).

5.1 Controlled evaluation protocols

We use three **LSP-Env** protocols that vary visual appearance while keeping the task state fixed (see Figure 4). They probe scene-context invariance (background variation), disambiguation under visually similar distractors (agent-like distractors), and stability under changes to the controlled entity's visuals (agent appearance changes). This separation is aligned with our diagnostics: whether OOD induces low-dimensional translation/rescaling in latent space, or introduces changes that would require updating the representation rather than the policy head. The agent uses a CNN encoder f_θ and an MLP policy head (details in Appendix C). Each observation maps to $z = f_\theta(o) \in \mathbb{R}^D$, the interface between perception and control, and we analyze how appearance-only perturbations change the distribution of z.

5.2 Data collection with a frozen encoder

We analyze a frozen encoder f_θ trained under Ω_{train} and collect latents under controlled train and eval observation processes with encoder parameters fixed.

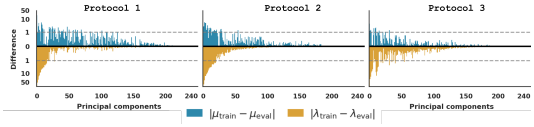


Figure 5: **Per-component mean and variance shifts (train PCA basis) for three protocols.** Top bars: $|\mu_i^{\text{train}} - \mu_i^{\text{eval}}|$; bottom bars: $|\lambda_i^{\text{train}} - \lambda_i^{\text{eval}}|$. Linear scale below dashed threshold, log scale above. Shifts concentrate in leading PCs; tail components contribute negligibly.

Train latents. We collect $Z_{\text{train}} \in \mathbb{R}^{N \times D}$ from N observations $o \sim \Omega_{\text{train}}(o \mid s)$ where task-relevant factors vary (agent position, layout, etc.) and appearance is fixed. In particular, training observations include no background changes, no agent-like distractors, and no changes to the agent appearance (Figure 4).

Eval latents. We collect $Z_{\text{eval}} \in \mathbb{R}^{N \times D}$ from observations $o' \sim \Omega_{\text{eval}}(o \mid s)$ where the latent state is held fixed and only task-irrelevant visual factors vary according to three protocols from subsection 5.1. Thus, differences between Z_{train} and Z_{eval} isolate visual OOD effects.

5.3 Train PCA coordinate system

We next define the coordinate system used for all reported train–eval shift metrics.

Step 1: train PCA basis. We apply PCA to Z_{train} to obtain an orthonormal basis of principal directions and their associated variances: $V_t = (v_1^t, \ldots, v_D^t), \qquad \Lambda_t = (\lambda_1^t, \ldots, \lambda_D^t)$, where v_i^t is the i-th train principal direction and λ_i^t is the variance of train latents along that direction.

Step 2: project eval onto the train basis. Using the same frozen encoder, we collect the eval latents and project them onto the train basis V_t and compute the resulting per-component variances: $\Lambda_e = (\lambda_1^e, \ldots, \lambda_D^e)$, where λ_i^e is the variance of eval latents along the train direction v_i^t.

Analyzing eval latents in the train PCA basis tests whether OOD appearance changes mainly redistribute mean and variance along directions that dominate training. If variability stays in the leading train components, the policy head remains closer to its training support; if substantial variance appears orthogonal to the train subspace, eval latents require extrapolation. Thus, the train PCA basis

separates reweighting of existing structure from new dominant directions. Because this analysis could be misleading if eval induces a rotated dominant subspace, we explicitly verify train–eval subspace alignment in subsection 5.6. This check ensures that train-coordinate metrics reflect latent geometry rather than a coordinate artifact.

This analysis addresses three research questions (RQs):

1. **(RQ1) Training latent structure.** What statistical and geometric properties characterize the encoder latent space on in-distribution data?
2. **(RQ2) Distraction-induced shift.** How do these statistics change when OOD visual distractors are introduced?
3. **(RQ3) Subspace stability.** Does the train PCA subspace remain sufficient to explain eval-time variability, or do OOD observations introduce new latent directions?

To answer these RQs, we introduce metrics that quantify train–eval shifts at different levels.

5.4 Component-level shift metrics in the train basis

Let $Z_{\text{train}}, Z_{\text{eval}} \in \mathbb{R}^{N\times D}$ denote the matrices of train and eval latents collected as described in subsection 5.2. All metrics in this subsection are computed in the train PCA basis V_t.

Mean shift. For each train principal direction v_i^t, we compute the coordinate-wise means μ_i^t and μ_i^e of the projected train and eval latents. We define the mean-shift vector: $\Delta\mu = (\mu_1^e - \mu_1^t, \ldots, \mu_D^e - \mu_D^t)$, and the global mean-shift energy: $E_\mu = \|\Delta\mu\|^2 = \sum_{i=1}^D (\Delta\mu_i)^2$.

Variance shift. Similarly, we define the variance difference vector: $\Delta\lambda = (\lambda_1^e - \lambda_1^t, \ldots, \lambda_D^e - \lambda_D^t)$, and the global variance-shift energy: $E_\lambda = \|\Delta\lambda\|^2 = \sum_{i=1}^D (\Delta\lambda_i)^2$.

These energies provide coarse but informative measures of the overall magnitude of distributional shift induced by OOD visual perturbations. In addition, we compute complementary per-component metrics (mean differences, variance differences, variance fractions, and ratios) in Appendix E and summarize them in Table 1.

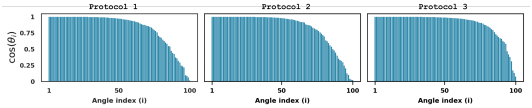


Figure 7: **Cosines of principal angles between train and eval PCA subspaces.** Values near one indicate strong alignment, with deviations only at high-index, low-variance components.

Energy concentration in the dominant train directions. To test whether OOD effects concentrate in the top train PCs, we define a set of the top-K train PCs by training variance: Top-$K(\lambda^t)$. We then define the *top-K mean-shift energy ratio* and *top-K variance-shift energy ratio*:

$$E_\mu^K = \frac{\sum_{i\in \text{Top-}K(\lambda^t)} (\Delta\mu_i)^2}{E_\mu}, E_\lambda^K = \frac{\sum_{i\in \text{Top-}K(\lambda^t)} (\Delta\lambda_i)^2}{E_\lambda}. \tag{3}$$

These ratios answer whether the dominant train subspace explains most of the shift magnitude.

5.5 Eval distribution analysis in the train PCA basis

We now describe how the eval distribution differs from train when expressed in the train PCA basis.

Empirical shape of 1D coordinate distributions. Empirically, the per-component coordinate distributions in the train PCA basis are close to unimodal for most components (see Figure 3). Accordingly, we summarize the shift using coordinate-wise means and variances, a Wasserstein decomposition confirms that these explain most of the train–eval discrepancy, while the remaining residual reflects higher-order shape changes rather than new modes (see Appendix I). Across all metrics and all three protocols, distractor-induced OOD effects appear primarily as: (i) systematic mean shifts and (ii) variance redistribution *within the dominant train subspace* (see Figure 5). Intermediate and tail components can exhibit large ratio-based metrics when λ_i^t is near zero, but their contribution to total shift energy is negligible. Therefore, OOD observations do not substantially change the dominant latent structure learned under Ω_{train}.

Concentration on top-K train principal components. The energy ratios (E_μ^K, E_λ^K) quantify how much OOD-induced shift is explained by dominant train directions. For background variation, the first ten train PCs account for 87.5% of the mean-shift energy and nearly all variance-shift energy. For the other protocols, the corresponding fractions are $(0.961, 0.991)$ and $(0.852, 0.994)$. Across all protocols, mean and variance shifts concentrate in the high-variance train subspace. While some

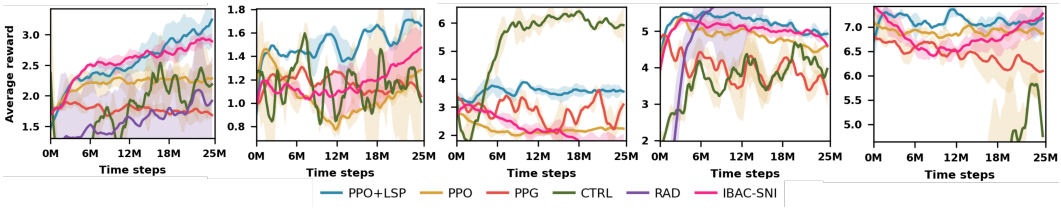


Figure 8: **Training progress on the Procgen environments Bigfish, Dodgeball, Heist, Jumper, and Plunder is shown from left to right.** Latent space noise enables effective handling of visual variations in procedurally generated levels.

tail components exhibit large variance ratios (Appendix E), their contribution is small due to near-zero train variance. Consistently, Figure 6 shows rapid saturation of cumulative shift energy with K, suggesting that OOD effects are captured by a small number of leading components. These conclusions rely on eval latents remaining aligned with the train subspace, which we verify next.

5.6 Subspace similarity analysis

We assess whether train and eval latents share geometry by comparing their PCA subspaces.

Train and eval PCA bases. Applying PCA *separately* to train and eval latents yields orthonormal bases: $V_t = (v_1^t, \ldots, v_D^t)$, $V_e = (v_1^e, \ldots, v_D^e)$, referred to as the *train* and *eval* bases. Since both lie in the same latent feature space $\mathbb{R}^D$, they can be compared directly.

Principal angles. Let $\mathcal{U} = \text{span}(v_1^t, \ldots, v_r^t)$ and $\mathcal{W} = \text{span}(v_1^e, \ldots, v_r^e)$ be the leading train and eval subspaces. Their alignment is measured by principal angles $\{\theta_i\}_{i=1}^r$. Values $\cos(\theta_i) \approx 1$ indicate that dominant eval variability lies within the train subspace, while large drops signal a rotated eval subspace not captured during training.

Per-component projection score. Principal angles capture global alignment but not which eval directions deviate or whether deviations are significant. Let P_t be the projector onto the train subspace $\mathcal{U}$. For a normalized eval direction v_j^e, define its overlap as $\text{overlap}(j) = \|P_t v_j^e\|^2$, and the variance-weighted deviation score $\text{score}(j) = \lambda_j^e\big(1 - \text{overlap}(j)\big)$, where λ_j^e is the eval eigenvalue. Large scores arise only for directions that both lie outside the train subspace and carry substantial variance; small scores indicate variability explained by existing train directions.

Empirical subspace stability. Principal angles and variance-weighted scores assess train–eval subspace alignment and distinguish reweighting from new directions. Across all protocols, eval latents remain almost entirely within the train PCA subspace: leading components satisfy $\cos(\theta_i) \approx 1$, with noticeable deviations only in high-index components that explain negligible variance. Eval directions with large λ_j^e strongly overlap with train directions and have low $\text{score}(j)$, indicating amplification of existing structure rather than the emergence of new axes. Quantitatively, the top-100 subspaces are tightly aligned ($\cos(\theta_0) = 1.0$, $\cos(\theta_{10}) \approx 0.9998$, $\cos(\theta_{30}) \approx 0.995$), with drops only near the tail (Figure 7).

Implication for the train-basis analysis. These results confirm that analyzing eval distributions in the train basis is valid for all three protocols, justifying the component-level interpretations in subsection 5.5 and motivating interventions that regularize policy behavior along dominant train-time latent directions. We replicate the latent space analysis from section 5 on Procgen tasks (Bigfish, Dodgeball, Heist, Jumper, Plunder; Figure 12), observing the same four properties: train and eval subspaces remain closely aligned with no new dominant directions emerging **(1)**; mean and variance discrepancies concentrate in the first few PCs **(2)**; variance allocation ordering is preserved under OOD observations **(3)**; and the top K=10 components retain the majority of both E_μ^K and E_λ^K **(4)**. The same geometry persists for task-trained CNNs under naturalistic DCS video shifts and for a frozen DINOv2 ViT under mild-to-moderate shifts, with degradation appearing only under the strongest ViT shifts (see Appendix J).

Latent Space Analysis Takeaways

1. In-distribution latents are low-rank: visual variation concentrates in leading PCs (RQ1).
2. OOD distractors mainly induce mean shifts and variance reweighting along these leading train directions, not diffuse drift (RQ2).
3. Eval latents stay aligned with the train PCA subspace; the tested perturbations do not introduce new dominant directions (RQ3).

5.7 From latent shift geometry to an intervention

Our analysis reframes "OOD visuals break policies" as a geometric failure with a direct training response: eval-time variability stays aligned with the dominant train PCA subspace (subsection 5.6), meaning OOD shifts move embeddings into low-density regions rather than introducing new directions. Since shift energy concentrates in a leading PCs (Figure 6) and latent variances differ substantially across components, robustness requires variance-scaled noise along those directions, not isotropic perturbation. This motivates **Latent Space Perturbation (LSP)**, introduced in section 6.

6 Latent Space Perturbation

The analysis in section 5 identifies a specific failure mode: appearance-only OOD inputs shift embeddings mainly within the dominant train PCA subspace, but into low-density regions rarely encountered during training. This points to a support mismatch rather than new latent directions and does not require task-relevant and nuisance factors to occupy separate PCs. LSP only requires substantial OOD-shift energy to overlap with the dominant training subspace (see Appendix M). **We propose Latent Space Perturbation (LSP)**, a training-time regularizer at the representation–control interface. LSP expands the effective training support by injecting structured Gaussian perturbations aligned with dominant train PCs, scaled by the corresponding train-time variances, without modifying the encoder or using pixel-level augmentation. LSP is algorithm-agnostic: it modifies only the latent input to the policy networks during training updates, and applies to both on-policy and off-policy methods (see Algorithm 2 and Algorithm 3). LSP remains lightweight: throughput decreases by about 9% and GPU memory increases by 11.5% on LSP-Bench, while PCA recomputation remains on the order of 10–15 ms up to $D = 8192$ (see Appendix H and Appendix L). This design is principled when OOD shifts are naturalistic: adversarial constructions that deliberately maximize subspace mismatch lie outside this regime and are addressed in section 7.

Algorithm 1 Conceptual overview of **Latent Space Perturbation (LSP)**, derived from the latent shift geometry diagnosed in section 5. More details in Appendix G, Algorithm 2, Algorithm 3 and Algorithm 4.

if time to recompute noise directions **then**
  Collect encoder latents $z = f(o)$ from training observations
  Compute PCA on train latents $\Rightarrow$ principal directions $\{v_i\}$ and variances $\{\lambda_i\}$
  Sample distraction power $p_i(t) \sim \mathcal{P}$
  Scale it: $\eta_i(t) = p_i(t)\sqrt{\lambda_i}$
end if
for each model forward during training **do**
  Encode observation: $z \leftarrow f(o)$
  Sample Gaussian noise $\varepsilon \sim \mathcal{N}(0, I)$
  Perturb latent: $\hat{z} = z + \sum_i \eta_i(t)\, \varepsilon_i\, v_i$
  Generate action using policy head $a \sim \mathrm{MLP}(\hat{z})$
end for

6.1 Perturbation parameterization

Let $z \in \mathbb{R}^D$ denote the encoder latent for an observation. Compute PCA on a collection of train-time latents to obtain the top-q principal directions $V_q = [v_1, \ldots, v_q] \in \mathbb{R}^{D\times q}$ and their associated eigenvalues (variances) $\lambda = (\lambda_1, \ldots, \lambda_q)$. Define $\sigma_i \triangleq \sqrt{\lambda_i}$ and $\sigma = (\sigma_1, \ldots, \sigma_q)$. At training step t, LSP samples a dimensionless *distraction power* vector $p(t) = (p_1(t), \ldots, p_q(t))$ from a scheduler $\mathcal{P}$, with $p_i(t) \in [0, p_{\max}]$ where $p_{\max} > 0$ is a hyperparameter. We set the per-component noise amplitudes $\eta(t) = p(t) \circ \sigma \in \mathbb{R}^q$, where $\circ$ denotes element-wise multiplication. Then we sample $\varepsilon \sim \mathcal{N}(0, I_q)$ and perturb the latent by $\Delta z = V_q\big(\varepsilon \circ \eta(t)\big)$, $\hat{z} = z + \Delta z$. Thus, each perturbation magnitude satisfies $\eta_i(t) = p_i(t)\sigma_i$, which is an explicit fraction $p_i(t)$ of one standard deviation along principal direction v_i.

6.2 Perturbation scheduler

The scheduler $\mathcal{P}$ controls the temporal profile and magnitude of perturbations through $p(t)$. We use bounded schedules (linear ramps, sudden on/off, sinusoidal, constant), detailed in Table 3. Because $p_i(t)$ is dimensionless and σ_i carries the latent scale, the magnitude is directly interpretable: for example, $p_i(t) = 0.9$ injects noise with standard deviation $0.9\,\sigma_i$ along v_i. We ablate perturbation strength (see Figure 11). Varying $p_{\max}$ from 0.05 to 3.5 across all three protocols reveals a consistent pattern: performance peaks around $p_{\max} \approx 0.9$, with weaker noise providing insufficient regularization and stronger noise degrading performance, suggesting a moderate task-dependent optimum. At fixed perturbation magnitude, no scheduling strategy consistently dominates, while across Procgen the preferred $p_{\max}$ is more closely associated with policy sensitivity to the perturbed latent directions than with distractor

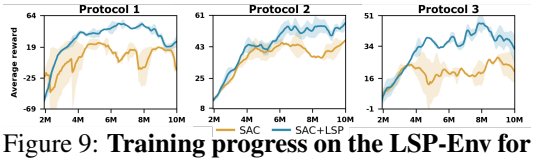


Figure 9: **Training progress on the LSP-Env for protocols 1, 2, and 3 is shown from left to right.** The average per-episode return increased by 20% compared to standard PPO.

count (see Appendix K). To assess whether structured perturbation is necessary at all, we compare LSP against isotropic Gaussian noise $\mathcal{N}(0, I)$ injected into the full latent vector. Naive noise injection consistently *degrades* performance relative to the unperturbed baseline, confirming that the benefit of LSP comes specifically from perturbing along the high-variance train directions, those along which OOD visual shifts concentrate.

6.3 Results

We evaluate LSP as a direct test of the analysis-to-intervention link in section 5 and section 6. The diagnosis predicts that robustness should improve by training the policy head to tolerate structured displacements in the dominant train PCA subspace, without systematic in-distribution degradation since the perturbations follow directions present at train time. Across all benchmarks, we compare each baseline agent to an LSP-enabled variant with identical architecture and optimization, differing only in whether PCA-aligned latent perturbations are applied. We validate the mechanism under controlled appearance-only shifts in LSP-Env, and also in Procgen and RL-ViGen. For reproducibility, all training details (architectures, optimizers, hyperparameters, LSP scheduler settings, evaluation protocols, seeds, and hardware) are reported in Appendix D.

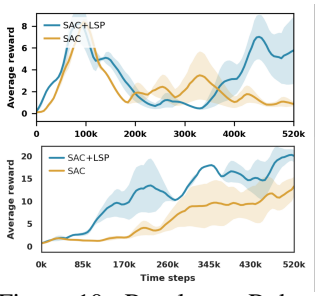


Figure 10: Results on Robosuite tasks Lift (above) and Door (below) in RL-ViGen.

Controlled appearance shifts (LSP-Env). LSP-Env varies appearance while keeping dynamics fixed, so performance differences isolate visual robustness. In Figure 9, LSP improves peak average episodic return by up to 20% over PPO. For Protocol 1 (background), the best configuration uses the sinusoidal schedule (Strategy 5) with $p_{\max} = 0.9$, injecting noise up to $0.9\sigma_i$. For Protocols 2 and 3 (agent-like distractors and agent appearance), the same schedule performs best with $p_{\max} = 0.8$, reflecting the need to limit perturbations when distractors closely resemble the controlled agent.

Uncontrolled visual diversity (Procgen). We compare PPO+LSP against five baselines: PPO (Schulman et al., 2017), PPG (Cobbe et al., 2021), CTRL (Mazoure et al., 2021), RAD (Laskin et al., 2020), and IBAC-SNI (Igl et al., 2019). As shown in Figure 8, LSP achieves the highest or second-highest performance on all five tasks. At the final checkpoint it outperforms RAD and IBAC-SNI on four of five environments, with RAD performing best on Jumper. LSP also yields a smaller final visual generalization gap than PPO on all five tasks (see Figure 18). The best schedule–strength pairs $\{\text{strategy}; p_{\max}\}$ are: Bigfish $\{3; 0.8\}$, Dodgeball $\{5; 1.1\}$, Heist $\{4; 0.8\}$, Jumper $\{4; 1.1\}$, and Plunder $\{4; 1.1\}$. A policy-sensitivity analysis of these task-dependent values is provided in Appendix K.

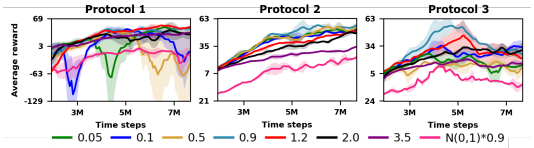


Figure 11: **Ablation over noise strength.** Optimal performance is achieved around 0.9; weaker noise is insufficient, while stronger noise leads to lower returns. Uniform Gaussian noise degrades training.

Continuous-control robotics (RL-ViGen). On the Robosuite Door and Lift tasks, Figure 10 compares SAC with and without LSP. On Door, SAC+LSP reaches an average reward near 20 after about 520k steps, compared with roughly 13 for SAC. On Lift, LSP substantially improves OOD performance while the SAC baseline degrades after its early training peak. Together, these results demonstrate that the benefit of LSP extends beyond discrete-control Procgen environments to continuous-control robotic manipulation.

7 Conclusion and Limitations

We study visual OOD generalization in pixel-based RL with fixed dynamics and rewards. Appearance-only shifts primarily induce structured mean and covariance drift along dominant train-time principal components rather than new latent directions. Motivated by this geometry, we propose Latent Space Perturbation (LSP), which regularizes the policy head with PCA-aligned latent noise without modifying the encoder or using pixel-level augmentation. Across LSP-Env, Procgen, and RL-ViGen, LSP improves visual OOD performance. LSP targets naturalistic shifts in background, distractors, and appearance, where OOD variation largely remains within the training subspace. To probe its limits, we evaluate color inversion ($c \mapsto 255 - c$) and combined background, distractor, and agent-appearance shifts. Both reduce train–eval subspace alignment and introduce variation outside the dominant train basis (Figure 14). LSP remains more robust than vanilla SAC, but its advantage decreases as subspace mismatch grows, marking the natural boundary of the method.

AI USE STATEMENT

In this work, we used generative AI tools for translation. We have not used generative AI tools for implementing methods, generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims or proving them, proposing or refining hypotheses, designing or evaluating research methodology and experiments, cleaning and reformatting datasets, supporting qualitative data analysis, or interpreting results.

Additionally, we used generative AI tools for drafting parts of the paper, improving its readability, suggesting its structure, and proposing the title and keywords. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by the authors, and all LLM-drafted or LLM-translated text was read and edited by the authors to ensure it accurately reflects our results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the Latent Space Perturbation (LSP) procedure and training formulation in the main text. The appendix reports the model architectures, optimization and RL hyperparameters, LSP-specific settings, PCA recomputation frequency, evaluation protocols, random seeds, hardware configuration, and computational overhead measurements used in our experiments. The supplementary material additionally provides the implementation, benchmark-specific configurations, and training and evaluation scripts, including commands required to reproduce the reported experiments.

REFERENCES

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.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Fred Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3):183, 1954.

Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1(01):217–233, 1961.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Sumeet Batra and Gaurav S Sukhatme. Zero-shot generalization of vision-based rl without data augmentation. *arXiv preprint arXiv:2410.07441*, 2024.

David Bertoin and Emmanuel Rachelson. Local feature swapping for generalization in reinforcement learning. *arXiv preprint arXiv:2204.06355*, 2022.

Xiaoyu Chen, Jiachen Hu, Chi Jin, Lihong Li, and Liwei Wang. Understanding domain randomization for sim-to-real transfer. *arXiv preprint arXiv:2110.03239*, 2021.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems*, 36:73383–73394, 2023.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pp. 2020–2027. PMLR, 2021.

Shangding Gu, Laixi Shi, Muning Wen, Ming Jin, Eric Mazumdar, Yuejie Chi, Adam Wierman, and Costas Spanos. Robust gymnasium: A unified modular benchmark for robust reinforcement learning. *arXiv preprint arXiv:2502.19652*, 2025.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. *Advances in neural information processing systems*, 34: 3680–3693, 2021.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.

Andrew Jesson and Yiding Jiang. Improving generalization on the procgen benchmark with simple architectural changes and scale. *arXiv preprint arXiv:2410.10905*, 2024.

Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *International conference on machine learning*, pp. 1809–1818. PMLR, 2017.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

Albina Klepach, Alexander Nikulin, Ilya Zisman, Denis Tarasov, Alexander Derevyagin, Andrei Polubarov, Nikita Lyubaykin, and Vladislav Kurenkov. Object-centric latent action learning. *arXiv preprint arXiv:2502.09680*, 2025.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International conference on machine learning*, pp. 5815–5826. PMLR, 2021.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Bonnie Li, Vincent François-Lavet, Thang Doan, and Joelle Pineau. Domain adversarial reinforcement learning. *arXiv preprint arXiv:2102.07097*, 2021.

Anthony Liang, Jesse Thomason, and Erdem Bıyık. Visarl: Visual reinforcement learning guided by human saliency. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2907–2912. IEEE, 2024.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *arXiv preprint arXiv:2505.19789*, 2025.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

Bogdan Mazoure, Ahmed M Ahmed, Patrick MacAlpine, R Devon Hjelm, and Andrey Kolobov. Cross-trajectory representation learning for zero-shot generalization in rl. *arXiv preprint arXiv:2106.02193*, 2021.

Haoyi Niu, Jianming Hu, Guyue Zhou, and Xianyuan Zhan. A comprehensive survey of cross-domain policy transfer for embodied agents. *arXiv preprint arXiv:2402.04580*, 2024.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.

Tien Pham, Xinyun Chi, Khang Nguyen, Manfred Huber, and Angelo Cangelosi. Deguv: Depth-guided visual reinforcement learning for generalization and interpretability in manipulation. *arXiv preprint arXiv:2509.04970*, 2025.

Md Masudur Rahman and Yexiang Xue. Bootstrap state representation using style transfer for better generalization in deep reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 100–115. Springer, 2022.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 8787–8798. PMLR, 2021.

Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/schaul15.html`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Anoopkumar Sonar, Vincent Pacelli, and Anirudha Majumdar. Invariant policy optimization: Towards stronger generalization in reinforcement learning. In *Learning for Dynamics and Control*, pp. 21–33. PMLR, 2021.

Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite–a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

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.

Zhenxiong Tan, Kaixin Wang, and Xinchao Wang. C-procgen: Empowering procgen with controllable contexts. *arXiv preprint arXiv:2311.07312*, 2023.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. *Advances in Neural Information Processing Systems*, 33: 7968–7978, 2020.

Xudong Wang, Long Lian, and Stella X Yu. Unsupervised visual attention and invariance for reinforcement learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6677–6687, 2021.

Ziyu Wang, Yanjie Ze, Yifei Sun, Zhecheng Yuan, and Huazhe Xu. Generalizable visual reinforcement learning with segment anything model. *arXiv preprint arXiv:2312.17116*, 2023.

Zizhao Wang, Kaixin Wang, Li Zhao, Peter Stone, and Jiang Bian. Dyn-o: Building structured world models with object-centric representations. *arXiv preprint arXiv:2507.03298*, 2025.

Zhecheng Yuan, Sizhe Yang, Pu Hua, Can Chang, Kaizhe Hu, and Huazhe Xu. Rl-vigen: A reinforcement learning benchmark for visual generalization. *Advances in Neural Information Processing Systems*, 36:6720–6747, 2023.

Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *arXiv preprint arXiv:1811.06032*, 2018.

A Forms of Generalization and Visual Benchmarks

Reinforcement learning agents encounter multiple forms of distribution shift between training and deployment. Prior work commonly distinguishes **state and layout**, **dynamics and physics**, **goal and reward**, and **visual and observation** generalization, each targeting a different source of variability.

State and layout generalization concerns robustness to unseen initial states or environment configurations, often addressed through procedural generation as in schema networks, MiniGrid, and Procgen (Kansky et al., 2017; Chevalier-Boisvert et al., 2023; Cobbe et al., 2020). **Dynamics and physics generalization** focuses on changes in physical parameters such as gravity, friction, mass, or morphology, and is typically tackled using robust RL and domain randomization methods (Packer et al., 2018; Rajeswaran et al., 2016; Chen et al., 2021; Sadeghi & Levine, 2016; Peng et al., 2018; Andrychowicz et al., 2020; Niu et al., 2024), as well as frameworks that explicitly vary physical properties in robotics environments, such as Robust Gymnasium (Gu et al., 2025). **Goal and reward generalization** studies adaptation to unseen goals or reward specifications, with universal value function approximators, hindsight experience replay, goal-conditioned policies, and successor features providing key mechanisms (Schaul et al., 2015; Andrychowicz et al., 2017; Liu et al., 2022; Barreto et al., 2017). **Visual and observation generalization** addresses robustness to appearance changes, including textures, colors, backgrounds, and camera pose, using techniques such as domain randomization, natural-video distractors, and visual augmentations or regularization (Cobbe et al., 2020; Zhang et al., 2018; Laskin et al., 2020; Kostrikov et al., 2020; Igl et al., 2019).

Several benchmarks explicitly target visual generalization. Procgen (Cobbe et al., 2020) and its configurable extension C-Procgen (Tan et al., 2023) provide procedurally generated arcade-style environments with held-out levels for evaluating generalization under visual and layout variation. The Distracting Control Suite (DCS) (Stone et al., 2021), built on the DeepMind Control Suite (Tassa et al., 2018), evaluates robustness to changes in background, color, and camera pose. More recently, Liu et al. (2025) introduced a VLA generalization benchmark based on language-conditioned tasks, evaluating out-of-distribution robustness along visual, semantic, and execution axes.

B LSP-Env Additional Details

LSP-Env is a lightweight, side-scrolling 2D platformer designed for fast, controlled studies of visual generalization. Compared to Procgen, which offers broad procedural diversity but limited factor isolation, LSP-Env allows explicit, manual control of individual appearance factors while keeping the underlying dynamics and reward computation fixed. This enables rapid ablations (for example, changing only the background while keeping the same task logic) and mechanistic diagnosis of appearance-induced distribution shift.

Task and episode structure. At each step, the agent aims to move right as efficiently as possible in a procedurally generated terrain containing obstacles (gaps and vertical walls). The environment is side-scrolling: the agent is rendered near the horizontal center of the frame, while the world coordinate `world_position` advances as the agent moves right. Episodes terminate only due to a step-limit truncation (`max_steps`).

Observation space. Observations are RGB images of size 64×64 rendered from the current view:

$$\mathcal{O} = \texttt{Box}(0, 255, (3, 64, 64), \texttt{uint8}). \tag{4}$$

Action space. Actions are continuous and two-dimensional,

$$\mathcal{A} = \texttt{Box}\big((-1, 0), (1, 1), (2,), \texttt{float32}\big), \tag{5}$$

where the first coordinate controls horizontal motion ($a_x \in [-1, 1]$) and the second coordinate is a jump command ($a_j \in [0, 1]$). The jump command is only applied when the agent is on the ground and a minimum jump interval has elapsed; in that case, it sets the initial upward velocity proportional to a_j.

Reward. The per-step reward is shaped to favor fast horizontal progress and discourage unnecessary actions:

$$r_t = \texttt{step_reward} \cdot v_x + \texttt{step_penalty} + \mathbb{I}[\texttt{jumped}] \cdot \texttt{jump_penalty}, \tag{6}$$

where v_x is the executed horizontal velocity for that step. By default, $\texttt{step_reward} = 1.0$, $\texttt{step_penalty} = -0.1$, and $\texttt{jump_penalty} = -0.3$.

Controllable axes of visual variability. We expose three sources of task-irrelevant appearance variation that are used throughout the paper:

- **Background variation.** The background is sampled independently of the task state (for example: solid colors, noise, gradients, procedural textures, or natural images), enabling background-only distribution shifts.
- **Agent-like distractors.** Non-controllable geometric entities are rendered as moving distractors (with configurable number, shape, color, and optional rotation). They affect the pixels but do not affect the physical state or reward computation.
- **Agent appearance.** The controlled agent's rendering is varied across episodes (implemented via configurable shape and color, and optional rotation), without changing dynamics or the reward function.

C Architecture Details

CNN Encoder. We use the standard Nature-CNN backbone to map an RGB observation $o \in \{0, \dots, 255\}^{C \times H \times W}$ to a latent embedding $z \in \mathbb{R}^D$. We first normalize $x = o/255$ and compute

$$h_{\text{conv}} = \text{Flatten}(\phi(\text{Conv}_{64,3\times3,s=1}(\phi(\text{Conv}_{64,4\times4,s=2}(\phi(\text{Conv}_{32,8\times8,s=4}(x)))))))\,, \tag{7}$$

where $\phi(\cdot) = \text{ReLU}(\cdot)$. Let $d_{\text{conv}} = \dim(h_{\text{conv}})$. We then form a normalized hidden representation

$$h = \text{LN}(\phi(W h_{\text{conv}} + b)) \in \mathbb{R}^D, \qquad D = 256, \tag{8}$$

and output the final latent embedding via a linear projection

$$z = W_{\text{out}} h + b_{\text{out}} \in \mathbb{R}^D. \tag{9}$$

Table 1: Metrics used to characterize train-eval latent distribution shift, all computed in the train PCA basis.

Metric	Definition	Interpretation
mean_diff$_i$	$\lvert\mu_i^e - \mu_i^t\rvert$	Absolute mean shift along the i-th train PC
var_diff$_i$	$\lvert\lambda_i^e - \lambda_i^t\rvert$	Absolute change in variance along the i-th train PC
var_ratio$_i$	$\lambda_i^e / \lambda_i^t$	Relative variance change along the i-th train PC
var_frac$_i^{t/e}$	$\lambda_i^{t/e} / \sum_{j=1}^{D} \lambda_j^{t/e}$	Fraction of total variance explained by the i-th train PC
frac_ratio$_i$	var_frac$_i^e$/var_frac$_i^t$	Change in explained variance between eval and train data of the i-th PC
E_μ^K	$\frac{1}{E_\mu} \sum_{i \in \text{Top-}K(\lambda^t)} (\Delta\mu_i)^2$	Mean-shift energy ratio in top-K train PCs
E_λ^K	$\frac{1}{E_\lambda} \sum_{i \in \text{Top-}K(\lambda^t)} (\Delta\lambda_i)^2$	Variance-shift energy ratio in top-K train PCs

D Experimental Setup

This appendix reports the full configuration used to produce the results in subsection 6.3, covering all three benchmarks (LSP-Env, Procgen, RL-ViGen). We list the model architecture, optimizer, RL-algorithm hyperparameters, LSP-specific scheduler settings, evaluation protocol, and compute hardware. Hyperparameter values not specified explicitly follow the reference implementations of SAC and PPO.

D.1 Hardware and Software

All experiments were run on a single workstation: NVIDIA RTX 3060 (12 GB VRAM), Intel Core i7-10700K, 32 GB RAM. Training used PyTorch with mixed-precision disabled, Adam optimizers, and TensorBoard logging. PCA recomputation for LSP was performed on GPU using torch.

D.2 Common Components

All three benchmarks share the same conceptual pipeline: a CNN encoder produces a latent $z \in \mathbb{R}^D$ from RGB observations, an MLP policy head maps z to actions, and LSP injects PCA-aligned Gaussian noise into z during training. The PCA basis is recomputed every f steps from a buffer of recent encoder latents and used until the next refresh. The noise scheduler is identical across benchmarks: a strategy index in $\{1, \ldots, 6\}$ (Table 3) parameterized by $p_{\max}$ (target noise), t_{switch}, and the number of cycles.

D.3 Per-Benchmark Configuration

Table 2 consolidates the full configuration for each benchmark. A few items are worth highlighting before reading the table:

- LSP-Env and RL-ViGen use SAC; Procgen uses PPO. Scheduler hyperparameters reported below correspond to the best configurations identified in subsection 6.3.
- In LSP-Env the encoder follows the Nature-CNN backbone. RL-ViGen uses a deeper 3×3 stride-2 stack to handle 84×84 inputs. Procgen uses the same Nature-CNN output dimension as in Cobbe et al. (2020).
- The PCA recompute period f (`mainvec_frequency`) is given in environment steps for SAC and in policy iterations for PPO.

D.4 Seeds, Aggregation, and Error Bars

Each configuration is trained with independent random seeds (seeds 1–5), differing in environment sampling, network initialization, and replay buffer order. Learning curves report the mean across seeds; shaded regions correspond to standard deviation across seeds after applying a moving average over a window of 10 consecutive evaluation points to reduce point-wise noise.

Table 2: Full experimental configuration across the three benchmarks. Values not listed follow reference SAC/PPO implementations.

	LSP-Env	**Procgen**	**RL-ViGen (Door)**
Algorithm and observations			
RL algorithm	SAC	PPO	SAC
Observation size	$64 \times 64 \times 3$	$64 \times 64 \times 3$	$84 \times 84 \times 3$
Action space	continuous, 2D	discrete (15)	continuous, 4D
Frame stack	1	1	1
Total env steps	1.0×10^7	2.5×10^7	5.0×10^5
Encoder (CNN $\rightarrow$ latent $z \in \mathbb{R}^D$)			
Conv stack	Nature-CNN (32−64−64)	IMPALA / Nature-CNN	32−32−32, 3×3 s2 $\times 3$
Latent dim. D	256	256	1024
Post-conv MLP	FC→LN→FC	FC→LN→FC	FC→LN→FC
Policy / value heads			
Policy MLP	$[256, 256]$	$[256, 64]$	$[512, 512]$
Q / value MLP	$[256, 256]$	shared trunk	$[512, 512]$
Optimization			
Optimizer	Adam	Adam (ϵ=10^{-5})	Adam
Policy LR	1×10^{-4}	2.5×10^{-4}	1×10^{-4}
Critic / value LR	3×10^{-4}	shared	1×10^{-4}
LR schedule	constant	linear decay	constant
Grad-clip norm	10.0	0.5	10.0
Discount γ	0.95	0.99	0.99
GAE λ	–	0.95	–
Entropy coef.	autotuned (α)	0.01	autotuned (α)
Value coef.	–	0.5	–
PPO clip ϵ	–	0.2	–
SAC / PPO specific			
Parallel envs	128	8	1
Rollout length / batch	b=512	n_{steps}=128	b=512
PPO epochs / minibatches	–	3 / 8	–
Replay buffer size	5×10^5	–	4×10^5
Learning starts	500	–	10 000
Target update τ	0.005	–	0.005
Encoder target τ_{enc}	0.001	–	0.001
Policy update freq.	every 2 steps	–	every 2 steps
LSP-specific			
Perturbed directions q	4	1	1
PCA recompute period f	250 env steps	every 1 PPO iter.	5 000 env steps
Noise strategy	5 (sinusoidal)	3 (sudden on)	3 (sudden on)
Target noise $p_{\max}$	0.9 (P1) / 0.8 (P2,P3)	task-dependent (subsection 6.3)	0.8
t_{switch}	5×10^6	10^7	5×10^6
Cycles	1	4	1
Evaluation			
Eval frequency	every 500 env steps	every PPO iter. on held-out levels	every 5 000 env steps
Eval episodes / envs	100 parallel	128 (start_level=0, num_levels=0)	10
Train / eval split	fixed visual config	200 train levels / unbounded eval	original `train` / `eval` split

E Latent Space Analysis Metrics

Mean shift (`mean_diff`). The metric mean_diff$_k = |\mu_k^e - \mu_k^t|$ measures the absolute displacement of the eval mean along the k-th train principal component. As shown in Figure 5, the largest mean shifts occur along leading components ($k = 0, 1, 2, 3, 7, \dots$) with mean_diff $\in [2.9, 20]$, and these components also have high train variance λ_k^t. Thus, background-induced OOD perturbations primarily translate the latent distribution along the dominant (high-variance) train directions rather than along the low-variance tail. Since all task-relevant factors are held fixed in the eval construction, these shifts can be attributed to background variation.

Train variance concentration (`var_frac`^{t}). The metric $\text{var_frac}_k^t = \lambda_k^t / \sum_j \lambda_j^t$ captures the fraction of total train variance explained by component k. The train latent space is highly concentrated: v_0 explains approximately 60% of the variance and the top ten components account for more than 0.88. This indicates that the dominant structure of the train distribution is low-dimensional.

Eval variance in the train basis (`var_frac`^{e}). Projecting eval latents onto the train basis, $\text{var_frac}_k^e = \lambda_k^e / \sum_j \lambda_j^e$ shows that eval variance is likewise concentrated in the leading train directions (e.g., $k = 1, 0, 7, 3, 5, 2, \dots$). Moreover, analyzing the variance-change energy $\sum_k (\Delta\lambda_k)^2$ reveals that variance changes are almost entirely confined to directions that are already dominant during training: the top five PCs by λ_{train} account for approximately 0.998 of $\sum_k (\Delta\lambda_k)^2$ (Figure 5). This supports the hypothesis that OOD variability largely reweights variance within the principal train subspace.

Relative importance shift (`frac_ratio`). The metric $\text{frac_ratio}_k = \text{var_frac}_k^e / \text{var_frac}_k^t$ measures how the *relative* contribution of each component to total variance changes from train to eval. For intermediate components with large var_diff (e.g., $k = 27, 34, 42, 47, \dots$), frac_ratio_k can be substantial (approximately 6.9–11), indicating that these directions become relatively more prominent under eval. However, their absolute explained variance remains small, since both var_frac^t and var_frac^e are on the order of 10^{-4}–10^{-2}. Conversely, components with the largest var_frac^e (e.g., $k = 0, 1, 2, 3, \dots$) typically exhibit frac_ratio_k in the range 0.3–2.2, indicating modest redistribution among the leading directions. Extremely large frac_ratio_k values in the far tail (where $\text{var_frac}^t \sim 10^{-6}$ and $\text{var_frac}^e \sim 10^{-4}$) are not indicative of meaningful structure, as their absolute contributions remain negligible.

Relative variance change (`var_ratio`). The metric $\text{var_ratio}_k = \lambda_k^e / \lambda_k^t$ characterizes multiplicative variance change. Extremely large ratios (on the order of 10^3 and above) occur only when λ_k^t is near zero (e.g., $\lambda_k^t \sim 10^{-13}$, $\text{var_frac}_k^t \sim 10^{-16}$), and these components also have negligible eval variance ($\text{var_frac}_k^e \sim 10^{-7}$). Such values are therefore dominated by denominator effects rather than meaningful variance growth. For components with non-negligible train variance, var_ratio_k remains moderate (e.g., 1.4–2.1), reflecting genuine increases in variability.

Absolute variance change (`var_diff`). The metric $\text{var_diff}_k = |\lambda_k^e - \lambda_k^t|$ identifies directions with the largest absolute increase in spread and is not distorted by small denominators. The largest values ($\sim$ 0.3–0.8) occur at intermediate components ($k = 27, 34, 42, 47, \dots$), where $\lambda_k^t \approx 0.5$–1.2, $\text{var_frac}_k^t \sim 10^{-3}$–$10^{-4}$, and $\text{var_frac}_k^e \sim 10^{-2}$–$10^{-3}$. Thus, although these directions exhibit noticeable absolute variance increases, they account for a small fraction of total variance both before and after the shift. Finally, because the eval construction isolates background variability while freezing other factors, eval variance is likely underestimated relative to settings where all sources of variation are active. Nevertheless, even under this constrained setup, certain leading components (e.g., PC1, PC5, PC7) exhibit increased variance driven by background variation alone.

F SCHEDULING

LSP controls perturbation magnitude over training with a *distraction power* scheduler that modulates the per-step noise scale. This is useful for two reasons: (i) it avoids imposing the same perturbation strength throughout optimization, which can be overly disruptive early on or unnecessarily mild later; and (ii) it lets us probe how robustness depends on the timing and persistence of latent perturbations.

Concretely, at step t the scheduler outputs a bounded scalar (or vector) $p(t) \in [0, p_{\max}]$ that determines the perturbation amplitude applied in the train PCA basis (see subsection 6.1). Because $p(t)$ is dimensionless, $p_{\max}$ is directly interpretable as the maximum fraction of a train-time standard deviation used for perturbation along selected components. We consider schedules that vary in *onset* (gradual vs. abrupt), *duration* (transient vs. persistent), and *periodicity* (constant vs. oscillatory), enabling controlled ablations of when and how strongly the policy head is regularized. Table 3 lists the schedules used in our experiments.

Table 3: Distraction power scheduling strategies used in LSP.

Schedule	**Power function** $p(t)$	**Description**
Linear ramp-up	$p_{\max}\,\dfrac{t}{T}$	Linearly increases distraction from 0 to $p_{\max}$
Linear ramp-down	$p_{\max}\left(1-\dfrac{t}{T}\right)$	Linearly decreases distraction from $p_{\max}$ to 0
Sudden activation	$\begin{cases}0, & t < t_{\text{switch}},\\ p_{\max}, & t \geq t_{\text{switch}}\end{cases}$	Abruptly turns distraction on at t_{switch}
Sudden deactivation	$\begin{cases}p_{\max}, & t < t_{\text{switch}},\\ 0, & t \geq t_{\text{switch}}\end{cases}$	Abruptly turns distraction off at t_{switch}
Cyclical sinusoidal	$\dfrac{p_{\max}}{2}\left[1+\sin\left(2\pi c\,\dfrac{t}{T}-\dfrac{\pi}{2}\right)\right]$	Periodic modulation with frequency c
Constant	$p_{\max}$	Fixed distraction strength throughout training

G Algorithm and Training Details

This section summarizes the training configurations used in our experiments. We report the key hyperparameters and model architectures for both PPO, used in Procgen, and SAC, used in LSP-Env. Unless stated otherwise, all other settings follow standard implementations, and the reported values were selected based on preliminary sweeps and prior work to ensure stable training and fair comparison across methods.

Table 4: **PPO hyperparameters and model size (Procgen).**

Parameter	**Value**
Number of environments (n_{envs})	8
Epochs	3
Mini-batch size	256
Discount factor γ	0.99
GAE parameter λ	0.95
Learning rate	2.5×10^{-4}
Gradient clipping norm	0.5
Entropy coefficient	0.01
Value loss coefficient	0.5
Model size	
CNN output dimension	256
MLP hidden dimensions	[256, 64]

Table 5: **SAC hyperparameters and model size (LSP-Env).**

Parameter	**Value**
Total timesteps	10,000,000
Learning starts	500
Replay buffer size	500,000
Batch size	512
Discount factor γ	0.95
Target update rate τ	0.005
Policy update frequency	2
Q-network learning rate	3×10^{-4}
Policy learning rate	1×10^{-4}
Entropy coefficient α	0.2
Model size	
CNN output dimension	256
Policy MLP layers	[256, 256, act_dim]
Q-network MLP layers	[256, 256, 1]

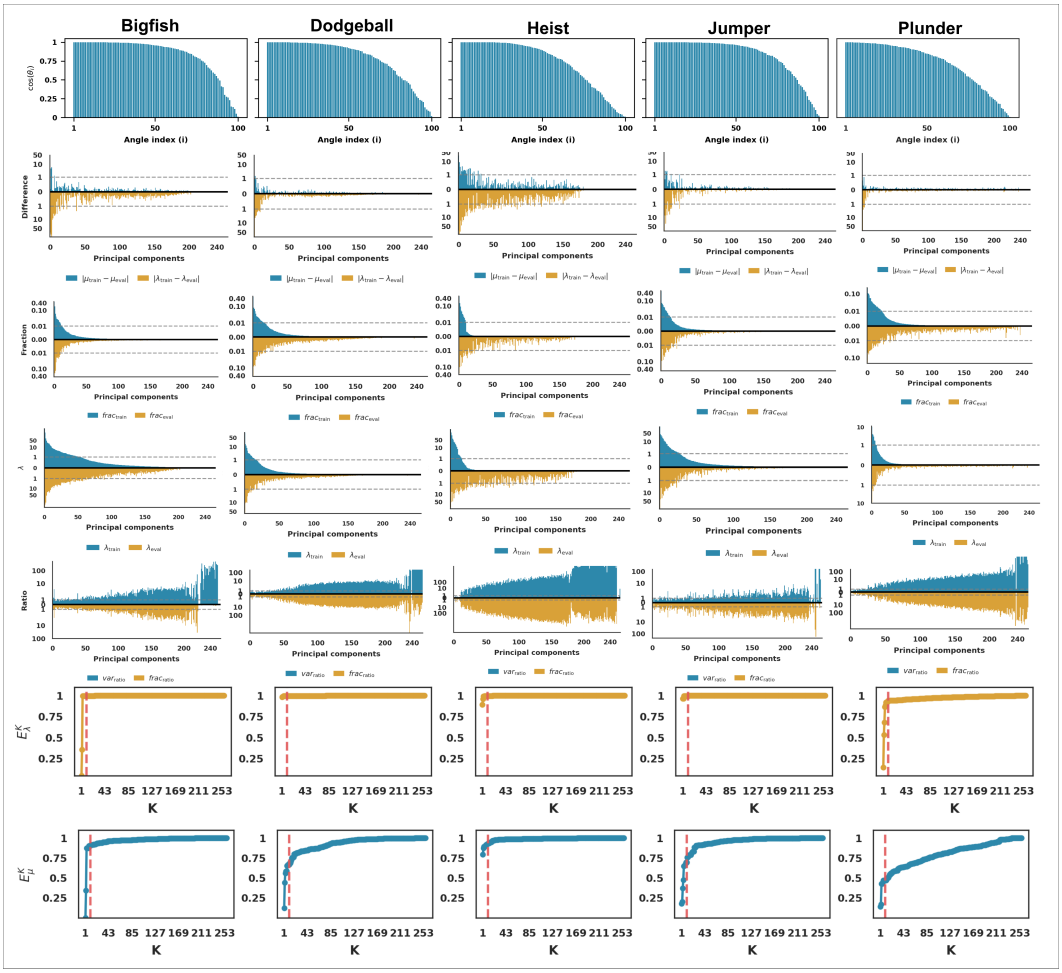
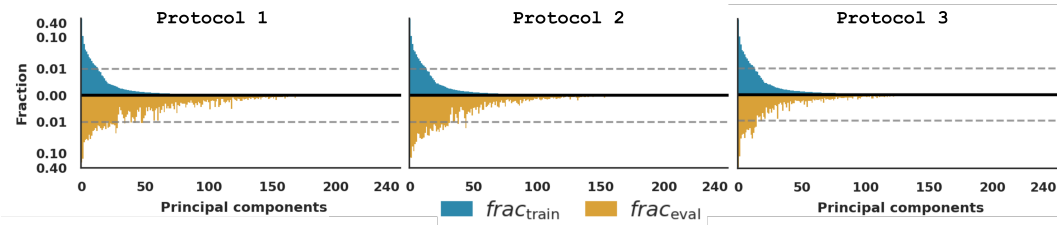
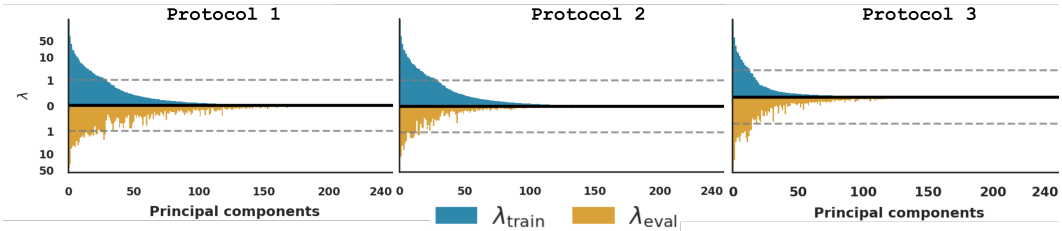


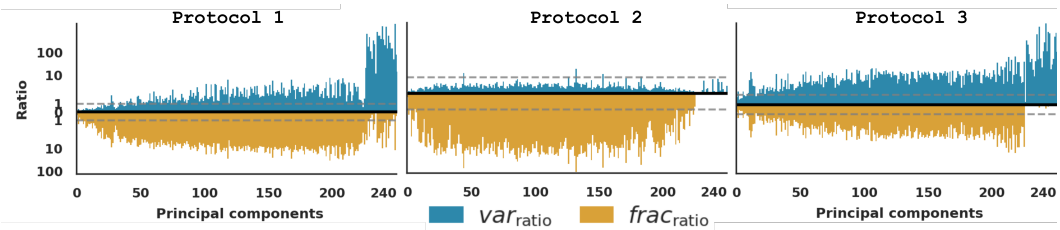
Figure 12: **Latent space analysis on Procgen environments.** Results demonstrate the same properties as on LSP-Env: **1)** Train and Eval spaces are similar to each other and do not introduce new variance directions (line 1) **2)**The biggest differences between μ and λ occurs along the first principal components (line 2), **3)** During evaluation OOD observations processing, variation allocation follows the same pattern as in tran ID data (line 4) **4)** The first K (K=10) components still preserve the most of the mean-shift energy E_μ^K and mean-variance energy E_λ^K (lines 6,7).



(a) Fraction of training and eval explained variance within each principal component in train basis.



(b) Training and eval variances within each principal component in train basis.



(c) Variance ratios for train and eval data in the train basis.

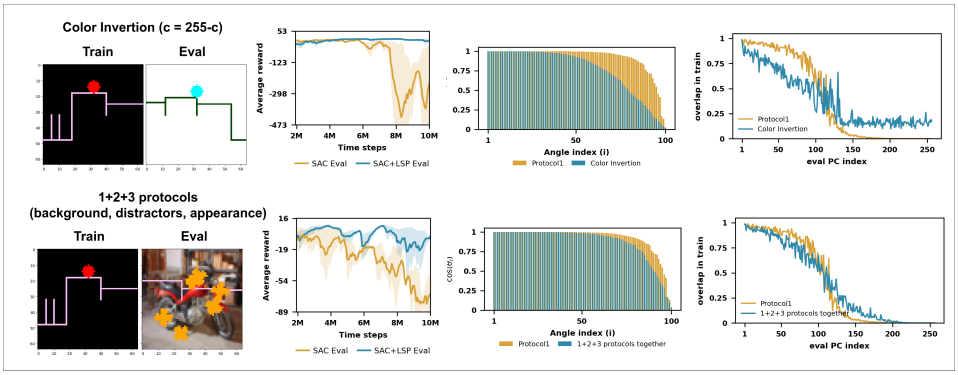


Figure 14: In the figures, we present sample observations, performance comparisons between SAC and SAC+LSP, and the behavior of **subspace-angle cosines** and the **overlap** metric relative to the background-only setting. This comparison is included because background changes, while visually salient (change $\sim 90\%$ of the observation), do not introduce new directions of variability according to our analysis. The **overlap** metric is defined as the squared norm of the projection of a normalized eval direction onto the train subspace, i.e., $\text{overlap}(j) = |P_t v_j^e|^2$, and thus measures how well a given eval direction is explained by the dominant train subspace. We emphasize that subspace-angle cosines capture changes in variability directions, whereas the overlap metric indicates whether dominant variability remains concentrated in leading coordinates of the train basis.

Algorithm 2 SAC with Latent Space Perturbation. Highlighted components indicate modifications introduced by LSP relative to standard SAC.

Input: policy π_θ, critics $Q_{\theta_1}, Q_{\theta_2}$, perturbation scheduler $\mathcal{P}$, PCA period f, number of components q
replay buffer $\mathcal{B} \leftarrow \emptyset$
Initialize target networks $\theta' \leftarrow \theta$, $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$
for $t \leftarrow 1, \dots, T$ **do**
  Sample $a \sim \pi_\theta(o \mid s)$
  Execute a, observe r, s', and done flag d
  Store transition (s, a, r, s', d) in $\mathcal{B}$
  if $t \bmod f = 0$ **then**
    Recompute PCA basis from samples in $\mathcal{B}$:
    $\mathbf{V} \leftarrow [v_1, \dots, v_q]$, $\mathbf{e} \leftarrow (e_1, \dots, e_q)$ {top-q principal directions and their variances}
  end if
  Sample batch (s, a, r, s', d) from $\mathcal{B}$
  Sample $\tilde{a} \sim \pi_\theta(o \mid s')$ and set $\mathcal{H} \leftarrow -\log \pi_\theta(\tilde{a} \mid s')$
  $y \leftarrow r + \gamma(1-d)\big(\min_i\{Q_{\theta'_i}(s', \tilde{a})\} + \alpha\mathcal{H}\big)$
  Update critics: $\theta_i \leftarrow \arg\min_{\theta_i} N^{-1} \sum \big(y - Q_{\theta_i}(s, a)\big)^2$
  Sample perturbation $\mathbf{p} = (p_1, \dots, p_q) \sim \mathcal{P}(t)$
  Set $\mathbf{z} \leftarrow \mathbf{p} \circ \sqrt{\mathbf{e}} = (p_1\sqrt{e_1}, \dots, p_q\sqrt{e_q})$
  Sample $\hat{a}_\theta \sim \pi_\theta(o \mid s, \mathbf{z}, \mathbf{V})$ via Algorithm 4
  Set $\mathcal{H} \leftarrow -\log \pi_\theta(\hat{a}_\theta \mid s)$
  Update actor: $\nabla_\theta N^{-1} \sum \big(\min_i\{Q_{\theta_i}(s, \hat{a}_\theta)\} + \alpha\mathcal{H}\big)$
  Update targets: $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$, $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
end for

Algorithm 3 PPO with Latent Space Perturbation (LSP). Highlighted components indicate modifications introduced by LSP relative to standard PPO.

Input: policy π_θ, value V_ψ, perturbation scheduler $\mathcal{P}$, PCA period f, number of components q, clip ϵ, epochs E
Initialize policy/value parameters θ, ψ
for $k \leftarrow 1, \ldots, K$ {policy iterations} **do**
  Collect on-policy rollout $\mathcal{D}$ of length T using π_{θ_k}
  Compute returns $\hat{R}_t$ and advantages $\hat{A}_t$ from $\mathcal{D}$
  if $k \bmod f = 0$ **then**
    Recompute PCA on encoder latents in $\mathcal{D}$
    $\mathbf{V} \leftarrow [v_1, \ldots, v_q]$, $\mathbf{e} \leftarrow (e_1, \ldots, e_q)$ {top-q PCs}
  end if
  for epoch $\leftarrow 1, \ldots, E$ **do**
    Sample minibatch $\mathcal{M} \subset \mathcal{D}$
    Sample perturbation $\mathbf{p} = (p_1, \ldots, p_q) \sim \mathcal{P}(k)$
    Set $\mathbf{z} \leftarrow \mathbf{p} \circ \sqrt{\mathbf{e}} = (p_1\sqrt{e_1}, \ldots, p_q\sqrt{e_q})$
    For samples in $\mathcal{M}$, compute $\log \pi_\theta(a_t \mid s_t, \mathbf{z}, \mathbf{V})$ via Algorithm 4
    $r_t(\theta) \leftarrow \exp\Big(\log \pi_\theta(a_t|s_t, \mathbf{z}, \mathbf{V}) - \log \pi_{\theta_k}(a_t|s_t)\Big)$
    $L^{\text{CLIP}}(\theta) \leftarrow \frac{1}{|\mathcal{M}|} \sum \min\Big(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)$
    Update actor: $\theta \leftarrow \arg\max_\theta\ L^{\text{CLIP}}(\theta)$
    Update value: $\psi \leftarrow \arg\min_\psi\ \frac{1}{|\mathcal{M}|} \sum \big(V_\psi(s_t) - \hat{R}_t\big)^2$
  end for
end for

Algorithm 4 Policy forward with latent perturbation

Input: encoder CNN, policy head MLP, image $o \in \mathbb{R}^{H\times W\times C}$, perturbation scale $\mathbf{z} \in \mathbb{R}^q$, PCA directions $\mathbf{V} \in \mathbb{R}^{D\times q}$
Encode observation: $i \leftarrow \text{CNN}(o) \in \mathbb{R}^D$
Sample base noise: $\varepsilon \sim \mathcal{N}(0, I_q)$
Scale along PCA axes: $\Delta \leftarrow \varepsilon \circ \mathbf{z} \in \mathbb{R}^q$
Project to latent space: $\Delta^{\text{lat}} \leftarrow \mathbf{V}\Delta \in \mathbb{R}^D$
Perturb latent: $\hat{i} \leftarrow i + \Delta^{\text{lat}}$
Compute action distribution parameters: $(\mu, \Sigma) \leftarrow \text{MLP}(\hat{i})$
Evaluate log-likelihood of action a: $\log \pi_\theta(a \mid o, \mathbf{z}, \mathbf{V}) \leftarrow \log \mathcal{N}(a; \mu, \Sigma)$
Output: (μ, Σ) and $\log \pi_\theta(a \mid o, \mathbf{z}, \mathbf{V})$

Table 6: End-to-end overhead of LSP on LSP-Bench (SAC).

Metric	SAC	SAC+LSP	Δ
SPS ↑	3107.67 ± 26.42	2824.33 ± 22.37	-9.11%
GPU RAM (GB) ↓	13.05 ± 0.036	14.55 ± 0.235	$+11.49\%$

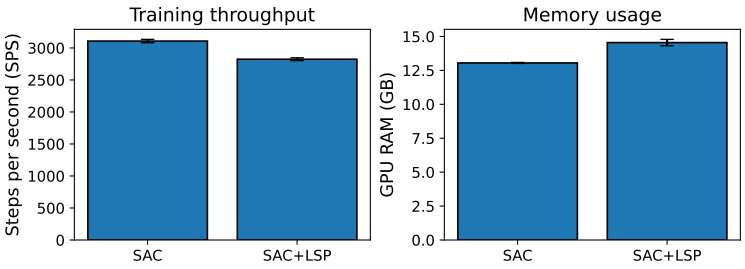


Figure 15: **End-to-end training throughput and memory (SAC vs. SAC+LSP) on LSP-Bench.** LSP adds modest overhead: SPS drops by $\approx 9\%$ and mean GPU RAM rises by $\approx 11.5\%$, indicating it is lightweight relative to standard SAC updates.

H Compute and Memory Overhead of LSP

We quantify the additional systems cost introduced by Latent Space Perturbation (LSP) relative to the underlying RL baseline. We report end-to-end training throughput and GPU memory usage, and then isolate the two LSP-specific contributors: (i) the actor forward pass under latent perturbation and (ii) the periodic PCA recomputation that defines the perturbation directions.

H.1 End-to-end overhead (SAC vs. SAC+LSP)

Table 6 compares standard SAC to SAC augmented with LSP on LSP-Bench. We report steps per second (SPS) and average GPU RAM usage during training; Figure 15 visualizes the same comparison. Overall, LSP reduces end-to-end throughput by $\approx 9\%$ and increases average GPU RAM usage by $\approx 11.5\%$ on this benchmark.

H.2 Actor forward-pass overhead vs. number of perturbed directions q

To isolate the incremental cost in the policy forward pass, we measure actor throughput (actions/s) as a function of the number of perturbed PCA directions q. Table 7 reports throughput and slowdown relative to the unperturbed actor ($q = 0$), with Figure 16 providing the corresponding plot.

The slowdown increases with q as expected, since larger q adds latent-space sampling and a low-rank projection. The small non-monotonicity across adjacent q values is typical for GPU workloads at these matrix sizes and does not change the end-to-end conclusion in Table 6.

H.3 Cost of PCA recomputation

LSP periodically recomputes a PCA from encoder latents. We benchmark PCA on latent matrices of shape $(N, 256)$ while extracting the top $q = 10$ components. Table 8 reports runtime and throughput, and Figure 17 visualizes scaling with N.

At the latent dimensionalities used in this work, PCA is inexpensive in absolute terms and is amortized by performing recomputation only periodically. In deployments with tighter compute budgets, overhead can be reduced by increasing the PCA period f (less frequent recomputation) and/or using smaller q (cheaper actor-side perturbation; Table 7 and Figure 16).

H.4 Discussion

On LSP-Bench with SAC, LSP incurs a $\approx 9\%$ reduction in end-to-end throughput and a $\approx 11.5\%$ increase in average GPU RAM usage (Table 6, Figure 15). In the main paper, this overhead corresponds to substantially improved robustness under appearance-only shifts. When OOD visual

Table 7: Actor throughput under latent perturbation vs. the number of perturbed PCA directions q (slowdown computed w.r.t. $q = 0$).

q	actions/s ↑	slowdown ↓
0	99,190	0.00%
1	87,345	13.58%
2	75,037	32.19%
3	84,941	16.78%
4	74,880	32.47%
5	56,443	75.73%
6	54,507	81.98%
7	82,001	20.96%
8	75,492	31.39%
9	79,455	24.84%
10	76,715	29.30%

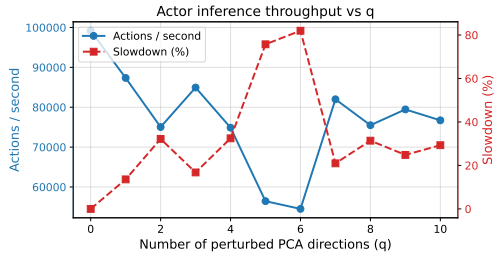


Figure 16: Actor throughput as a function of the number of perturbed directions q.

generalization is a target requirement, this trade-off is favorable; when the visual distribution is stationary, the baseline may be preferred for slightly higher throughput.

H.5 Generalization Gap

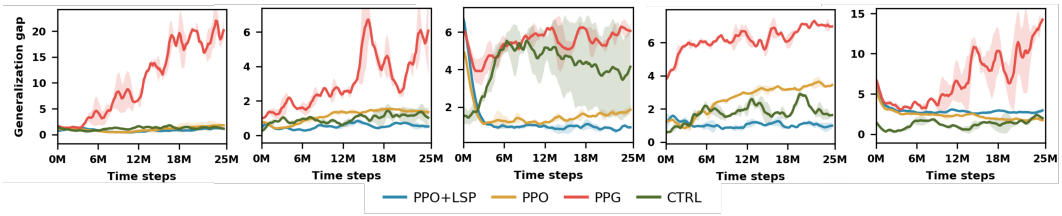


Figure 18: Generalization Gap of PPO+LSP, PPO, PPG and CTRL baselines on the Procgen benchmark.

I Higher-Order Structure of the Train-to-Evaluation Latent Shift

Our analysis does not assume Gaussian latent marginals. Instead, LSP relies on the weaker observation that train-to-evaluation shifts along dominant training directions are largely characterized by changes in location and scale. We quantify this using the Wasserstein-1 distance, normalized by the training standard deviation, before normalization, after removing the mean shift, and after additionally matching the variance.

Aggregated over the top-10 principal components and weighted by their explained training variance, the total discrepancy is approximately 0.60. Mean shifts account for about 67%, variance shifts for an additional 7%, and approximately 25% remains as a higher-order residual. The split-half distance

Table 8: PCA benchmark on latent batches of size N (latent dim. 256, extracting $q = 10$ components).

N	ms / PCA ↓	PCA iters/s ↑	drop
32	7.5145	133.08	0.00%
128	15.3417	65.18	51.03%
256	14.0577	71.14	46.54%
512	12.2255	81.80	38.53%
1024	14.5511	68.72	48.36%
2048	12.4709	80.19	39.74%

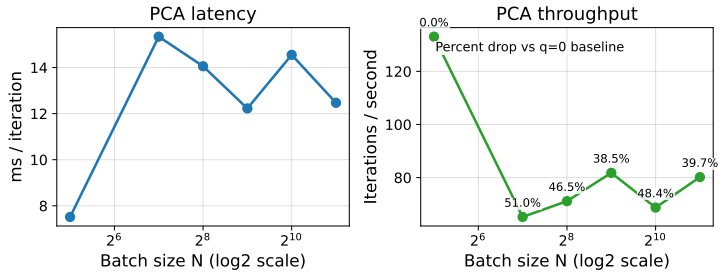


Figure 17: PCA recomputation cost for latent matrices of shape $(N, 256)$ (top-$q = 10$ components).

between independent subsets of the training distribution is only approximately 0.007, confirming that this residual is well above the sampling floor.

The decomposition is strongly component-dependent. PC0 is dominated by a mean shift, whereas PC5 mainly changes in scale, with evaluation variance increasing by approximately $1.4\times$. PC9 exhibits comparable contributions from both. Thus, no single moment explains all dominant directions.

The remaining residual does not indicate the emergence of new dominant modes. The marginals remain single-peaked, while the residual mainly reflects higher-order changes in shape. For example, PC0 has excess kurtosis of approximately -1.2, while PC1 has skewness of approximately -0.9.

These results clarify the assumption behind LSP. The method neither requires Gaussian latent distributions nor attempts to reproduce the full evaluation-time distribution. It assumes only that OOD variation remains concentrated along dominant training directions and that a substantial part of the shift is captured by changes in location and scale. LSP therefore expands support along these directions using variance-scaled perturbations.

The higher-order residual also marks a limitation: LSP does not explicitly model changes in skewness, kurtosis, or other higher-order statistics. Nevertheless, because most of the observed shift remains aligned with the training subspace and is explained by low-order changes, perturbing these directions can improve robustness without explicitly modeling the complete OOD distribution.

J Cross-Architecture Validation of Latent Geometry under Naturalistic Visual Shifts

To test whether the observed latent-space geometry is specific to CNN representations, we repeat the analysis with a frozen DINOv2 Vision Transformer. We evaluate whether OOD visual changes remain concentrated within the dominant training directions on both RL-ViGen and the Distracting Control Suite (DCS), where evaluation uses naturalistic real-world image and video backgrounds.

RL-ViGen. We analyze RL-ViGen Robosuite Door observations under the training distribution and evaluation settings with video-background shifts. Despite the different architecture and pre-

Table 9: Variance-weighted decomposition of the train-to-evaluation distribution shift over the top-10 training principal components. The residual denotes the discrepancy remaining after separately removing mean and variance differences.

Shift component	Fraction of total discrepancy
Mean shift	67%
Variance shift	7%
Higher-order residual	25%

training procedure, the DINOv2 representation remains strongly low-rank: the first ten training principal components explain approximately 82% of the total latent variance.

The train and evaluation principal subspaces remain closely aligned, with

$$\cos\theta_0 \approx 0.995, \qquad \cos\theta_{30} \approx 0.95.$$

The distribution shift is also concentrated in the leading training directions. The fraction of variance-shift energy captured by these components is approximately

$$E_\lambda^K \approx 0.99,$$

while the mean shift is similarly concentrated within the dominant training subspace. These properties remain qualitatively stable as shift severity increases, indicating that the visual shift primarily changes statistics along existing latent directions rather than introducing new dominant ones.

Distracting Control Suite. We further analyze Cartpole Balance and Cheetah Walk under Easy, Medium, and Hard DCS shifts using both the task-trained CNN and frozen DINOv2 ViT.

For the CNN, the first ten training principal components explain approximately 90% of the variance on Cartpole Balance and 94% on the Cheetah task. Across all evaluation severities, the shift remains concentrated within these directions. The top-ten mean-shift energy lies approximately in the range

$$E_\mu^{10} \approx 0.94\text{–}0.98,$$

while the corresponding variance-shift concentration remains close to

$$E_\lambda^{10} \approx 1.00.$$

The principal subspaces also remain highly aligned:

$$\cos\theta_0 \geq 0.999,$$

with deeper alignment remaining high,

$$\cos\theta_{30} \approx 0.86\text{–}0.98.$$

Thus, even under real-world video backgrounds, the task-trained CNN primarily expresses the new variability along directions already dominant during training.

The DINOv2 representation shows the same qualitative behavior, although it weakens with shift severity. Its first ten training components explain approximately 79% of the variance on Cartpole Balance and 77% on the Cheetah task. Under the Easy setting, the shift remains strongly concentrated in the dominant training directions, with

$$E_\lambda^{10} \approx 0.94\text{–}0.98, \qquad \cos\theta_{30} \approx 0.98.$$

Table 10: Ablation over the six LSP perturbation scheduling strategies with the perturbation magnitude fixed to 1.0. Values report mean episodic return ± standard deviation. Fixing the magnitude isolates the effect of the temporal scheduling profile from the effect of perturbation strength.

Strategy	0M	1M	2M	3M	4M	5M	6M	7M	8M	9M
Strategy 1	-86.1 ± 2.6	-37.4 ± 23.6	1.7 ± 6.8	37.5 ± 3.5	43.9 ± 10.5	52.0 ± 13.3	46.0 ± 16.4	47.4 ± 10.7	47.8 ± 6.3	$\mathbf{55.8 \pm 9.4}$
Strategy 2	-76.1 ± 0.8	15.0 ± 3.4	$\mathbf{26.0 \pm 4.3}$	34.4 ± 3.6	36.4 ± 6.2	29.2 ± 3.2	39.5 ± 4.9	32.3 ± 0.9	25.5 ± 12.4	25.0 ± 10.8
Strategy 3	-87.2 ± 0.5	$\mathbf{16.1 \pm 1.6}$	22.1 ± 8.9	38.0 ± 7.4	35.2 ± 12.7	46.3 ± 8.9	41.6 ± 9.2	26.6 ± 17.9	39.2 ± 6.6	47.4 ± 6.5
Strategy 4	-85.4 ± 15.4	11.8 ± 1.1	12.9 ± 13.1	38.1 ± 6.8	37.3 ± 16.4	43.5 ± 20.7	46.8 ± 17.8	44.9 ± 16.9	6.6 ± 42.1	19.5 ± 31.9
Strategy 5	-80.8 ± 3.7	8.7 ± 3.4	25.1 ± 7.0	$\mathbf{41.4 \pm 5.1}$	$\mathbf{51.7 \pm 1.7}$	$\mathbf{53.3 \pm 4.3}$	$\mathbf{55.4 \pm 0.1}$	$\mathbf{49.4 \pm 2.7}$	$\mathbf{50.3 \pm 0.8}$	43.7 ± 4.1
Strategy 6	-98.6 ± 6.6	14.9 ± 3.2	13.1 ± 13.3	37.5 ± 6.4	33.9 ± 14.1	40.8 ± 18.9	44.4 ± 16.2	37.5 ± 12.0	1.5 ± 38.7	14.1 ± 28.3

The leading directions are also almost perfectly aligned, with $\cos\theta_0$ approaching 0.999.

Under stronger video shifts, however, the DINOv2 representation increasingly expresses OOD variation outside the first ten training components. This effect is weaker for the task-trained CNN, suggesting that task-specific representation learning helps preserve the dominant latent subspace under severe visual changes.

Implications for the LSP assumption. These experiments show that the latent geometry motivating LSP is not specific to CNNs. Both the task-trained CNN and frozen DINOv2 ViT exhibit low-dimensional training representations, and under mild-to-moderate shifts, OOD variability remains concentrated within dominant training directions.

At the same time, DCS reveals an important limitation: sufficiently strong naturalistic shifts can cause a frozen general-purpose encoder to occupy directions outside the dominant training subspace. We therefore do not assume that arbitrary OOD shifts remain confined to a fixed low-dimensional subspace. Rather, LSP targets the regime where most OOD variation remains aligned with dominant training directions, which is observed most consistently for task-trained representations.

K Robustness to Perturbation Strategy and Interpretation of $p_{\max}$

LSP introduces two related design choices: the perturbation scheduling strategy and the maximum perturbation magnitude $p_{\max}$. We perform additional experiments to separate their effects and clarify what determines an appropriate perturbation strength.

Robustness to the perturbation scheduling strategy. To isolate the effect of the scheduling strategy, we fix the perturbation magnitude to 1.0 and evaluate all six strategies considered in the paper.

Table 10 shows that the strategies generally form a relatively tight performance cluster. Strategy 5 performs best at several intermediate checkpoints, while other strategies become competitive at different stages; for example, Strategy 1 achieves the highest mean return at 9M steps. Thus, no single scheduling strategy consistently dominates throughout training.

These results suggest that the perturbation magnitude matters more than the exact temporal schedule. Once magnitude is controlled, the strategies do not exhibit a single sharply isolated optimum, indicating limited sensitivity to the scheduler choice.

What determines the appropriate $p_{\max}$? A natural hypothesis is that $p_{\max}$ should be smaller in environments with more visual distractors. Our results instead suggest that it is more closely related to how sensitive the policy is to perturbations along dominant latent directions.

For each Procgen environment, we perturb the leading training principal component with amplitude 0.9 training standard deviations and measure the KL divergence between the original and perturbed policies, denoted $\mathrm{KL}@0.9$. Larger values indicate that the perturbed direction carries more task-relevant information.

The resulting measurements are reported in Table 11.

Across the five environments, the tuned $p_{\max}$ is negatively correlated with policy sensitivity:

Table 11: Relationship between the tuned perturbation magnitude $p_{\max}$, policy sensitivity to perturbation of the leading training principal component, and the number of visual distractors in the five Procgen environments. Policy sensitivity is measured as the KL divergence between the original and perturbed policy outputs for a perturbation magnitude of 0.9 training standard deviations.

Environment	$p_{\max}$	KL@0.9	Distractors
Bigfish	0.80	0.014	4
Dodgeball	1.10	0.005	5
Heist	0.80	0.062	1
Jumper	1.10	0.017	1
Plunder	1.10	0.006	3

$$r = -0.68.$$

By contrast, its correlation with the number of visual distractors is weak:

$$r \approx +0.15,$$

suggesting that distractor count alone does not determine the preferred perturbation strength.

Heist illustrates this most clearly. Its policy sensitivity is substantially larger than in the other environments,

$$\mathrm{KL@0.9} = 0.062,$$

compared with 0.005–0.017 elsewhere, and it uses the lower value $p_{\max} = 0.8$. This suggests that $p_{\max}$ is governed primarily by how much task-relevant information is carried by the perturbed latent directions.

Interpretation. A dominant principal component may encode both nuisance variation and task-relevant information. Stronger perturbations therefore improve robustness only until they begin to disrupt information used by the policy. The policy KL response provides a direct measure of this trade-off: directions that produce larger policy changes should be perturbed more conservatively.

Because this analysis contains only five Procgen environments, the correlation should be interpreted as a mechanistic trend rather than a statistically established relationship. Nevertheless, it provides a practical interpretation of $p_{\max}$ and suggests policy sensitivity to PCA-aligned perturbations as a useful diagnostic for selecting perturbation strength.

L Scalability to High-Dimensional Latent Spaces

LSP periodically estimates the dominant principal directions of the training latent distribution. To test whether this becomes expensive for high-dimensional representations, we benchmark PCA recomputation as the latent dimensionality D increases.

LSP requires only the leading q principal directions rather than a full eigendecomposition. We therefore use truncated randomized PCA, whose complexity for fixed sample count N and number of retained components q is approximately

$$\mathcal{O}(NDq),$$

and thus scales linearly with D.

We benchmark PCA recomputation from $D = 256$ to $D = 8192$, keeping N and q fixed to their training values. Results are shown in Table 12.

Table 12: PCA recomputation time as a function of the latent dimensionality D. The number of samples N and retained principal components q are held fixed. The final column reports the runtime relative to $D = 256$.

Latent dimension D	PCA time (ms)	Relative time
256	11.73	1.00×
512	12.09	1.03×
1024	10.41	0.89×
2048	12.95	1.10×
4096	12.57	1.07×
8192	14.71	1.25×

Increasing D by a factor of 32, from 256 to 8192, increases recomputation time only from 11.73 ms to 14.71 ms, corresponding to approximately 1.25×. Across this range, PCA recomputation therefore remains on the order of 10–15 ms on our hardware.

The PCA basis is recomputed only once every f training steps and reused between updates, further amortizing this cost. The $D = 1024$ regime is already used in our RL-ViGen experiments, demonstrating that LSP is applicable to substantially larger latent representations than those used in LSP-Env and Procgen. Separately, our end-to-end measurements on LSP-Bench show approximately a 9% throughput reduction and an 11.5% increase in GPU memory usage relative to the baseline (see Appendix H).

Overall, PCA recomputation does not become a practical bottleneck over the tested dimensionalities: increasing D from 256 to 8192 produces only a small increase in absolute recomputation time, which is further amortized by periodic updates.

Finally, LSP applies PCA to the single latent representation consumed by the policy head. Applying perturbations to per-token Key or Value representations inside a Transformer would constitute a different method and is outside the scope of this work.

M Does LSP Require Disentangled Representations?

LSP does not assume that task-relevant information and nuisance visual factors are disentangled into separate latent directions. It neither *induces* a disentangled representation nor *requires* the encoder to have already separated these factors.

LSP does not explicitly induce disentanglement. LSP operates between the visual representation and the policy head, perturbing dominant directions of the training latent distribution during policy optimization. It introduces no disentanglement objective, assigns no semantic meaning to individual principal components, and does not require PCs to correspond to independent generative factors.

Its goal is therefore not to learn a factorized representation, but to make the policy robust to structured variation along dominant training directions.

Individual principal components need not be purely task-related or purely nuisance-related. LSP only requires that a substantial fraction of the OOD-induced shift lies within the dominant training subspace. In our experiments, the concentration metric E^K typically lies in the range 0.85–0.99. This does not imply that individual PCs contain exclusively task or nuisance information.

A leading PC may contain both. Perturbing such a direction can therefore also disturb task-relevant signal, and $p_{\max}$ controls this trade-off. This is consistent with the policy-sensitivity analysis in Appendix K: directions that produce larger policy changes require smaller perturbation magnitudes.

Mixed task and nuisance information in LSP-Env. LSP-Env Protocol 3 provides a mixed-feature example. Here, the nuisance factor is the agent's appearance, which is inseparable in pixel space from task-relevant information such as its position. Thus, nuisance information need not occupy separate "style-only" PCs.

Nevertheless, LSP improves OOD performance in this protocol. Its preferred perturbation magnitude is lower than in Protocol 1, with $p_{\max} = 0.8$ versus $p_{\max} = 0.9$, consistent with more conservative perturbation when dominant directions contain more task-relevant information.

Required assumption. The actual assumption of LSP is therefore weaker than disentanglement: OOD-induced variation should substantially overlap with the dominant training-time latent subspace.

PCA identifies directions of large statistical variation rather than semantic factors. LSP therefore does not require individual PCs to correspond to background, appearance, or task state. It exploits only the geometric observation that visual OOD shifts often remain concentrated along directions already present during training.

When this condition holds, perturbing those directions exposes the policy to structured variation resembling the OOD shift. When substantial energy appears in previously unseen directions, the assumption weakens and the benefit of LSP is expected to decrease, consistent with the severe-shift analyses elsewhere in the paper.