STYLECRAFT: LATENT AUGMENTATIONS AS A STEP TOWARD GENERALIZABLE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning agents often perform well in their training environments but degrade under visual distribution shifts. Existing approaches commonly address this problem with predefined pixel augmentations, whose effectiveness depends on how well the selected transformations cover unseen visual variation. We propose StyleCraft, a two stage framework that combines task style disentanglement with learned visual augmentation. During the first stage, pixel augmentations together with gradient reversed reconstruction encourage the policy visible task representation to discard appearance specific information while a separate style representation captures visual variation. Once the generative model has learned a sufficiently structured style space, the second stage uses interpolation between style representations to construct new visual observations while keeping the task representation fixed. These generated observations replace fixed pixel augmentations in the invariance objective and provide additional supervision for learning a more visually robust task representation. Experiments across KAGE Bench, DCS based locomotion, RL4VLA, and ManiSkill show that StyleCraft improves robustness to unseen visual conditions across controlled visual shifts, continuous control, and robotic manipulation. We additionally reproduce the ManiSkill scene on a real robot, where StyleCraft retains nonzero success under unseen table colors while the PPO baseline fails.

1 INTRODUCTION

Reinforcement learning agents can achieve strong performance within the environments used for training while remaining sensitive to changes in visual appearance that do not alter the underlying task. Variations in backgrounds, textures, lighting, camera configuration, and object appearance may leave the dynamics and reward function unchanged, yet still cause a substantial degradation in policy performance. This issue becomes particularly important when policies are evaluated outside the training distribution or transferred from simulation to real world deployment, where the visual conditions observed during training cannot fully represent the variability encountered at test time.

A common approach to this problem is to increase the diversity of observations seen during training. A large family of methods uses predefined image transformations to expose the agent to multiple visual realizations of the same underlying state Laskin et al. (2020a); Yarats et al. (2021b); Hansen et al. (2021); Hansen & Wang (2021). These approaches have shown that carefully chosen augmentations can improve robustness to unseen visual conditions and reduce overfitting to the training environment. More generally, they demonstrate that increasing visual diversity during learning is an effective mechanism for improving generalization. However, this strategy also introduces a fundamental dependency on the augmentation family selected before training. The set of transformations is typically designed manually and represents only a limited collection of possible visual changes. Moreover, the transformations that are useful for one environment may not provide the appropriate variation for another. As a result, the generalization obtained through conventional augmentation is constrained by how well the predefined transformation family covers the visual changes that will appear during evaluation.

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

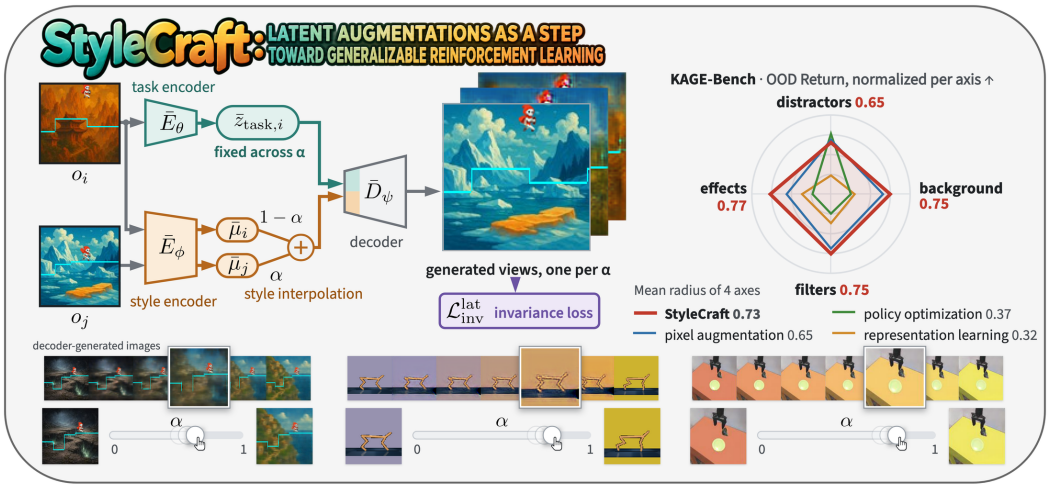


Figure 1: Visual overview of StyleCraft. Task information is kept fixed while style representations are interpolated and decoded into additional training views used by the invariance objective. The right panel shows KAGE-Bench OOD returns normalized per axis to $[0, 1]$ over $[V_{\min} - s,\ V_{\max} + s]$, where s is the sample standard deviation across the four plotted series. Baseline series are group means: pixel augmentation (RAD, SVEA), representation learning (CURL, IBAC-SNI, IDAAC), and policy optimization (PPG); StyleCraft is shown separately. The legend reports the mean radius over the four axes. The bottom row illustrates latent style interpolation.

An alternative direction focuses on the representations learned by the agent rather than on the direct transformation of input observations. Several methods encourage the policy representation to retain information required for control while becoming less sensitive to task irrelevant appearance Laskin et al. (2020b); Raileanu & Fergus (2021); Batra & Sukhatme (2025). Disentangled representation learning makes this separation explicit by decomposing an observation into factors associated with task content and visual style. Such a decomposition provides an opportunity that goes beyond representation invariance. If an observation can be represented as a combination of task and style factors, then visual style does not need to be treated only as nuisance information that should be removed from the policy representation. Instead, as illustrated in Figure 1, it can become a controllable source of visual variation. This leads to a natural question. *Can the learned style space itself be used to generate new task preserving observations for policy training?*

In this work, we introduce StyleCraft, a latent augmentation framework that learns disentangled task and style representations and uses the learned style space to generate additional training observations. As summarized in Figure 1, the task representation is kept fixed while the style representation is modified and decoded back into image space, producing multiple visual realizations of the same task content. Unlike conventional approaches based on predefined pixel transformations, StyleCraft derives augmentation from the learned representation of visual variation. The generated observations are used in a consistency objective that encourages the task representation to remain stable across different visual appearances, complementing the reinforcement learning objective without altering the underlying control problem.

We evaluate StyleCraft across four complementary visual generalization settings: controlled visual shifts in KAGE Bench Cherepanov et al. (2026), background shifts in DCS based locomotion tasks from RL ViGen Yuan et al. (2023), robotic manipulation under table color shifts in RL4VLA Liu et al. (2025), and a ManiSkill manipulation setting reproduced on a real robot. On KAGE Bench, StyleCraft reaches a mean normalized OOD score of 0.73 compared with 0.65 for the strongest baseline group, corresponding to an improvement of approximately 12.3%. Across the 18 DCS task and difficulty combinations, StyleCraft achieves an average final return of approximately 413 compared with 261 for the strongest baseline, an improvement of about 58%. In RL4VLA, StyleCraft reaches approximately 0.27 evaluation success while PPO and RAD remain close to zero. Finally, in the ManiSkill setting, StyleCraft achieves 40.8% average OOD success in simulation and retains 26.7% success on the real robot, whereas PPO fails on all evaluated OOD table colors.

Our main contributions are as follows.

- We introduce StyleCraft, a latent augmentation framework that generates task preserving visual observations by recombining disentangled task and style representations.
- StyleCraft improves visual generalization across all three simulated settings, with gains of about 19% on KAGE Bench, 58% on DCS, and 27% OOD success on RL4VLA where the baselines remain near zero.
- We demonstrate that the learned visual robustness transfers to a physical robotic setting. In a ManiSkill scene reproduced on a real robot, StyleCraft retains an average success rate of 26.7% on unseen table colors, whereas PPO obtains zero success on all evaluated OOD colors.

2 Related Work

Pixel Space Data Augmentation. Pixel based methods improve visual generalization by exposing the agent to predefined image transformations. RAD Laskin et al. (2020a) applies standard augmentations directly during policy learning, while DrQ Yarats et al. (2021b) regularizes Q learning using multiple transformed views of the same observations. DrQ v2 Yarats et al. (2021a) extends this approach to continuous control while retaining random shift augmentation. SVEA Hansen et al. (2021) stabilizes off policy learning under strong augmentations by keeping bootstrap targets clean, and SODA Hansen & Wang (2021) separates policy optimization on clean observations from an auxiliary consistency objective on augmented views. These methods remain tied to a transformation family selected before training, whereas StyleCraft uses such transformations only to establish task style separation and subsequently generates additional variation through the learned style space.

Representation Learning for Generalization. Representation based methods instead shape the features used by the policy. CURL Laskin et al. (2020b) learns visual representations through contrastive agreement between augmented views, while IBAC SNI Igl et al. (2019) uses an information bottleneck to retain control relevant information. IDAAC Raileanu & Fergus (2021) separates policy and value representations, CTRL Mazoure et al. (2022) aligns behaviorally similar trajectories through self supervised learning, and SGQN Bertoin et al. (2022) encourages reliance on control relevant image regions using Q function saliency. Unlike approaches that primarily suppress nuisance variation in the policy representation, StyleCraft additionally uses the separated style representation as a source of new training observations.

Policy Optimization and Disentangled Representations. PPG Cobbe et al. (2021) improves policy learning by separating policy and auxiliary value optimization, reducing interference between the two objectives. ALDA Batra & Sukhatme (2025) instead learns a factorized latent representation with a quantized autoencoder and maps unfamiliar latent components toward representations observed during training. StyleCraft also relies on representation factorization, but explicitly separates task and style information and then interpolates in the learned style space to generate new observations while keeping the task representation fixed.

3 Preliminaries

We consider a discounted Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \rho_0, \gamma)$. At time t, the agent receives a visual observation $o_t \in \mathcal{O}$ and selects $a_t \sim \pi_\theta(\cdot \mid o_t)$ to maximize

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{1}$$

We separate the underlying task state from its visual appearance. Let $e \in \mathcal{E}$ denote a visual style and $q_e : \mathcal{S} \to \mathcal{O}$ the corresponding observation mapping, $o_t = q_e(s_t)$. We consider shifts that change q_e while leaving the dynamics, reward, and task objective unchanged. Visual generalization therefore requires a policy trained under $p_{\text{train}}(e)$ to remain effective under a different $p_{\text{eval}}(e)$.

Conventional pixel based augmentation samples $\tilde{o} = \tau(o)$, $\tau \sim \mathcal{F}$, assuming that o and $\tilde{o}$ share the same task content. Its visual diversity is consequently limited by the predefined transformation family $\mathcal{F}$. We instead consider a factorized representation $z_{\text{task}} = E_{\text{task}}(o)$ and $z_{\text{style}} = E_{\text{style}}(o)$,

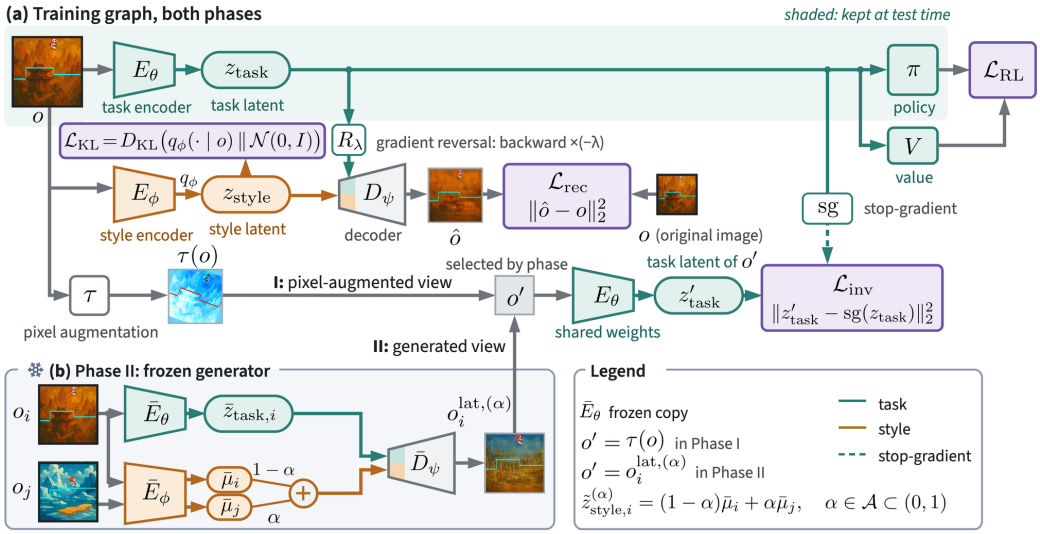


Figure 2: Overview of the StyleCraft training procedure. In Phase I, predefined pixel augmentations provide views for the invariance objective, while gradient reversed reconstruction and KL regularization encourage the separation of task and style information. At the phase transition, the task encoder, style encoder, and decoder are copied into a frozen generator. In Phase II, interpolated style representations are decoded together with a fixed task representation to produce generated views that replace pixel augmentations in the invariance objective. The shaded task encoder and policy branches are retained at test time.

where the former captures control relevant information and the latter visual appearance. A decoder reconstructs the observation as $\hat{o} = D(z_{\text{task}}, z_{\text{style}})$.

4 Method

StyleCraft builds on this factorization through the two phase training procedure shown in Figure 2. Phase I uses pixel augmentation together with reconstruction to induce task style disentanglement. Once the generative model becomes sufficiently accurate, Phase II keeps the task representation fixed and replaces pixel augmented views in the invariance objective with observations generated by interpolation in the learned style space. The resulting progression is *pixel variation* → *task style disentanglement* → *latent augmentation* → *stronger visual invariance*.

Disentangled Task and Style Representation. Given an observation $o \in \mathcal{O}$, we encode it into two latent representations. A task encoder E_θ produces $z_{\text{task}} = E_\theta(o)$, while a separate style encoder E_ϕ parameterizes a diagonal Gaussian distribution $q_\phi(z_{\text{style}} \mid o) = \mathcal{N}\left(\mu_\phi(o), \operatorname{diag} \sigma^2_\phi(o)\right)$.

A style representation is sampled using the reparameterization trick. The policy and value function receive only z_{task}. Consequently, the reinforcement learning objective requires this representation to retain information nec-

Algorithm 1 StyleCraft with a PPO backbone. Highlighted lines are additions to standard PPO.

Input: E_θ, π, V, E_ϕ, D_ψ, pixel augmentations $\mathcal{F}$, coefficients $\mathcal{A}$, switch step N

$t \leftarrow 0$, phase $\leftarrow$ I
for $k = 1, \ldots, K$ **do**
  Collect rollout $\mathcal{B}$ with $\pi(\cdot \mid E_\theta(o))$
  $t \leftarrow t + |\mathcal{B}|$; compute GAE advantages
  if phase $=$ I **and** $t \geq N$ **then**
    Frozen copy $(\bar{E}_\theta, \bar{E}_\phi, \bar{D}_\psi) \leftarrow (E_\theta, E_\phi, D_\psi)$
    phase $\leftarrow$ II
  end if
  for each minibatch $\{o_i\}_{i=1}^M \subset \mathcal{B}$ **do**
    $z_{\text{task},i} \leftarrow E_\theta(o_i)$
    $\mathcal{L}_{\text{RL}} \leftarrow$ PPO loss on $z_{\text{task},i}$
    $z_{\text{style},i} \sim q_\phi(\cdot \mid o_i)$
    $\mathcal{L}_{\text{rec}}, \mathcal{L}_{\text{KL}}$ via Eqs. (2), (3)
    if phase $=$ I **then**
      $o'_i \leftarrow \tau_i(o_i), \quad \tau_i \sim \mathcal{F}$
    else
      Pick $j \neq i$ from the minibatch
      $o'_i \leftarrow \{o_i^{\text{lat},(\alpha)}\}_{\alpha \in \mathcal{A}}$ via Eqs. (6), (7)
    end if
    $\mathcal{L}_{\text{inv}} \leftarrow \left\| E_\theta(o'_i) - \text{sg}(z_{\text{task},i}) \right\|_2^2$, averaged over views
    Minimize $\mathcal{L}_{\text{RL}} + \lambda_{\text{inv}}\mathcal{L}_{\text{inv}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}$
  end for
end for

essary for control. A decoder D_ψ reconstructs the visual observation from the joint task and style representation

$\hat{o} = D_\psi\left(R_\lambda(z_{\text{task}}), z_{\text{style}}\right)$, where R_λ denotes a gradient reversal operator. It behaves as the identity during the forward pass but reverses and scales the gradient propagated to the task encoder during optimization.

The reconstruction objective is

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_o\left[\left\|D_\psi\left(R_\lambda(z_{\text{task}}), z_{\text{style}}\right) - o\right\|_2^2\right]. \tag{2}$$

We additionally regularize the style posterior with KL loss to form a style space

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_o\left[D_{\text{KL}}\left(q_\phi(z_{\text{style}} \mid o) \| \mathcal{N}(0, I)\right)\right]. \tag{3}$$

The decoder and style encoder are optimized to reconstruct the observation. In contrast, the reversed reconstruction gradient discourages the task encoder from retaining appearance information that facilitates image reconstruction. At the same time, the reinforcement learning objective prevents z_{task} from discarding information required for control. This creates complementary optimization pressures that encourage control information and visual appearance to occupy different parts of the representation.

4.1 Phase I: Disentanglement from Pixel Augmentations

The reconstruction objective alone does not specify which visual factors should be treated as task irrelevant. During the first training phase, we provide this supervision through a predefined family of task preserving pixel transformations. For an observation o, we sample $o^{\text{pix}} = \tau(o), \quad \tau \sim \mathcal{F}$, where $\mathcal{F}$ denotes the pixel augmentation set. We encourage the task encoder to produce the same representation for the original and transformed observations using

$$\mathcal{L}_{\text{inv}}^{\text{pix}} = \mathbb{E}_{o,\tau}\left[\left\|E_\theta\left(\tau(o)\right) - \text{sg}\left(E_\theta(o)\right)\right\|_2^2\right], \tag{4}$$

where sg denotes the stop gradient operator. This objective identifies visual factors that can change while the underlying task remains unchanged. Combined with the gradient reversed reconstruction path shown in Figure 2, it encourages such information to be removed from z_{task} and represented through z_{style} instead. The first phase therefore serves two purposes. It trains an appearance invariant task representation and simultaneously organizes the latent space into task and style components that can later be recombined. The objective during the first phase is

$$\mathcal{L}^{(1)} = \mathcal{L}_{\text{RL}} + \lambda_{\text{inv}}\mathcal{L}_{\text{inv}}^{\text{pix}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}. \tag{5}$$

The predefined augmentations are therefore used as supervision for learning the initial decomposition. Their role is not only to expose the policy to additional images but also to define visual changes that should be separated from task information.

4.2 Phase II: Latent Augmentation

After N environment steps, the representation learned during the first phase is used to construct a generative source of task preserving visual variation. At this transition, as shown in the lower branch of Figure 2, we copy the task encoder $\bar{E}_\theta$, style encoder $\bar{E}_\phi$, and decoder $\bar{D}_\psi$ into a frozen teacher. This provides a stationary latent space and decoder for generating augmented observations while the student continues to learn.

For each observation o_i in a minibatch, the frozen teacher computes $\bar{z}_{\text{task},i} = \bar{E}_\theta(o_i)$ and the mean of its style posterior $\bar{\mu}_i = \mu_{\bar{\phi}}(o_i)$.

We then sample another observation o_j from the minibatch and interpolate between their style representations

$$\tilde{z}_{\text{style},i}^{(\alpha)} = (1-\alpha)\bar{\mu}_i + \alpha\bar{\mu}_j, \qquad \alpha \in \mathcal{A}, \tag{6}$$

where $\mathcal{A} \subset (0, 1)$ defines the interpolation coefficients.

The task representation of the original observation is kept fixed. A latent augmented observation is generated as

$$o_i^{\mathrm{lat},(\alpha)} = \bar{D}_\psi \left(\bar{z}_{\mathrm{task},i}, \tilde{z}_{\mathrm{style},i}^{(\alpha)} \right). \tag{7}$$

The construction changes the style coordinate while anchoring generation to the task representation of the original observation. Interpolation between style codes allows the decoder to produce visual conditions that are not restricted to the discrete transformations used during the first phase. For each source observation, multiple interpolation coefficients can be used to obtain several visual realizations of the same task content. The resulting images form the augmentation distribution used during the second phase.

4.3 Learning Invariance from Latent Augmentations

The central change between the two training phases is the source of observations used by the invariance objective. Pixel augmented observations are used during the first phase. After the transition, they are replaced by observations generated through latent style interpolation.

The student task encoder processes each generated observation $z_i^{\mathrm{lat},(\alpha)} = E_\theta \left(o_i^{\mathrm{lat},(\alpha)} \right)$.

and minimize

$$\mathcal{L}_{\mathrm{inv}}^{\mathrm{lat}} = \mathbb{E}_i \left[\frac{1}{|\mathcal{A}|} \sum_{\alpha \in \mathcal{A}} \left\| E_\theta \left(o_i^{\mathrm{lat},(\alpha)} \right) - \mathrm{sg} \left(E_\theta(o_i) \right) \right\|_2^2 \right]. \tag{8}$$

The original observation defines the task representation that should be preserved, while the generated observations provide alternative visual realizations on which the student is trained to recover the same representation. The second phase therefore creates a feedback mechanism between disentanglement and generalization. The first phase learns which information can be separated from the control representation. The resulting style space is then manipulated to generate additional observations. These observations in turn provide new supervision that further increases the invariance of the task representation. The training objective during the second phase stays the same except the $\mathcal{L}_{\mathrm{inv}}^{\mathrm{pix}}$ which becomes $\mathcal{L}_{\mathrm{inv}}^{\mathrm{lat}}$.

The reinforcement learning and reconstruction objectives remain active after the phase transition. Only the source of variation used in the invariance objective changes. The frozen teacher is used exclusively to generate latent augmented observations, while gradients from $\mathcal{L}_{\mathrm{inv}}^{\mathrm{lat}}$ update the student task representation.

5 Experimental Setup

We evaluate StyleCraft in four complementary visual generalization settings shown in Figure 3: controlled known axis shifts in KAGE Bench Cherepanov et al. (2026), background variation in the DCS based locomotion environments from RL ViGen Yuan et al. (2023), robotic manipulation in the Simpler based RL4VLA environment Liu et al. (2025), and a ManiSkill manipulation task additionally reproduced on a real robot. In all settings, the control objective remains fixed while the visual observation distribution changes. RL4VLA was originally designed for Vision Language Action policies and provides a textual instruction together with the visual observation. We discard the language input and train the policy from visual observations using only the reward signal.

The choice of baselines follows each setting. On KAGE Bench, we compare against RAD Laskin et al. (2020a), SVEA Hansen et al. (2021), CURL Laskin et al. (2020b), IBAC SNI Igl et al. (2019), IDAAC Raileanu & Fergus (2021), and PPG Cobbe et al. (2021). For the compact comparison in Figure 1, these methods are grouped according to Section 2: RAD and SVEA form *Pixel Space Augmentation*, CURL, IBAC SNI, and IDAAC form *Representation Learning*, and PPG represents *Policy Optimization*. On DCS and RL4VLA, we compare against PPO and RAD, while ManiSkill and real robot experiments use PPO as the baseline.

Evaluation metrics follow the underlying task. For KAGE Bench, we report average episode reward. RL4VLA is evaluated by the success rate of grasping the target object and placing it on the plate. DCS uses the task return, reflecting locomotion progress or the corresponding task specific control objective. For ManiSkill and the real robot, success is defined as grasping the cube and lifting it upward. KAGE Bench experiments use 10 seeds and 640 parallel environments, DCS uses 3 seeds and 6 parallel environments, RL4VLA uses 1 seed and 64 parallel environments, and ManiSkill uses 1 seed and 1024 parallel environments. Each parallel environment is initialized with its own internal seed, so even a run with a fixed global seed contains distinct environment instances rather than identical replicas. Reported values are averaged over the corresponding evaluations, with variability

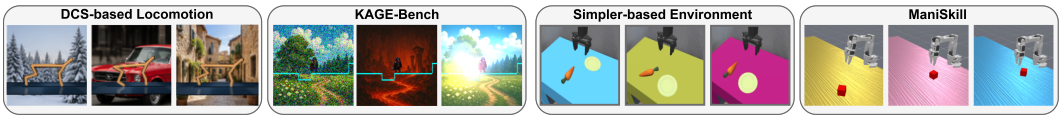


Figure 3: We evaluate our StyleCraft on four environments to test visual generalization: 1) DCS-based locomotion tasks (leftmost photo) with swapped backgrounds, testing skills for robust operation under background shifts, 2) KAGE-Bench (second from left), a general-purpose benchmark for evaluating generalization to backgrounds, filters, visual effects, and distractors, 3) Simpler-based manipulation tasks (second from right), which test robotic manipulation skills under varying table color, and 4) a manipulation task based on the ManiSkill simulator with swapped table color, to evaluate the transfer of knowledge from simulation to a real robot.

shown as the standard error of the mean (s.e.m.). All experiments were conducted on a server with an NVIDIA A100 GPU and 128 GB of RAM. Evaluation is performed exclusively on visual conditions not observed during training. Model parameters, architectural choices and training hyperparameters are available in Appendix Table 2 and Table 3.

6 Experiments

Our evaluation is designed to test StyleCraft under progressively less controlled visual shifts. KAGE Bench isolates individual sources of visual variation, DCS tests robustness across multiple continuous control tasks and increasing background shift severity, and RL4VLA evaluates whether the same mechanism remains effective in robotic manipulation. Finally, ManiSkill and the corresponding real robot setup test whether this robustness persists when moving from simulation to physical deployment.

6.1 Robotic Manipulation under Table Color Shift

We evaluate StyleCraft on the RL4VLA robotic manipulation task Liu et al. (2025), where the agent must grasp an object and place it on a plate. Training and evaluation differ only in the table color, while the task and scene geometry remain unchanged.

Training uses eight table colors: red, green, blue, yellow, cyan, magenta, white, and black. Evaluation uses twelve disjoint held out colors: orange, purple, teal, pink, brown, gray, azure, spring green, burgundy, mustard, olive, and forest green. This setting tests whether visual robustness extends from generic visual changes to a manipulation task in which the shifted appearance belongs directly to the robot workspace.

6.2 Locomotion under Background Shift

We evaluate StyleCraft on the DCS based locomotion setting from RL ViGen Yuan et al. (2023), following the visual distraction protocol of the Distracting Control Suite Stone et al. (2021). Training and evaluation use disjoint background sets while preserving the dynamics, reward, and control objective. We consider six continuous control tasks: `cheetah_run`, `hopper_hop`, `hopper_stand`, `walker_run`, `walker_stand`, and `walker_walk`. Each task is evaluated under *easy*, *medium*, and *hard* background shifts, testing whether the benefit of StyleCraft persists across different control problems and as the visual shift becomes more severe.

Table 1: KAGE-Bench: best evaluation return over training per config (mean ± s.e.m. over 10 seeds).

Config	RAD	SVEA	CURL	IBAC-SNI	IDAAC	PPG	StyleCraft (ours)
Background 1	-614 ± 146	-662 ± 103	-1889 ± 145	-1981 ± 51	-1966 ± 55	-1869 ± 83	$\mathbf{-85 \pm 15}$
Background 2	-304 ± 134	-98 ± 37	-463 ± 113	-1703 ± 82	-1640 ± 143	-1664 ± 129	$\mathbf{-46 \pm 15}$
Background 3	-501 ± 127	-579 ± 91	-716 ± 96	-1469 ± 96	-1420 ± 134	-1399 ± 128	$\mathbf{-347 \pm 225}$
Background 5	-566 ± 115	-479 ± 65	-714 ± 59	-1648 ± 54	-1595 ± 88	-1263 ± 93	$\mathbf{-188 \pm 44}$
Background 8	-132 ± 40	-91 ± 34	-723 ± 113	-1691 ± 196	-1381 ± 108	-1471 ± 138	$\mathbf{-88 \pm 28}$
Distractors 5	-248 ± 98	-208 ± 26	-387 ± 48	-529 ± 108	-719 ± 89	$\mathbf{-174 \pm 23}$	-252 ± 149
Effects 3	-1495 ± 39	-1433 ± 89	-1523 ± 108	-1798 ± 45	-1596 ± 77	-1758 ± 51	$\mathbf{-1279 \pm 148}$
Filters 1	$\mathbf{-603 \pm 58}$	-863 ± 57	-1173 ± 109	-1528 ± 77	-1190 ± 90	-1527 ± 103	-775 ± 190
Filters 2	-1405 ± 77	-1239 ± 105	-1352 ± 141	-1378 ± 60	-1312 ± 111	-1556 ± 84	$\mathbf{-852 \pm 162}$
Filters 3	$\mathbf{-682 \pm 117}$	-1215 ± 108	-1413 ± 118	-1873 ± 110	-1704 ± 110	-1716 ± 127	-1107 ± 188
Average	-655 ± 64	-687 ± 46	-1035 ± 39	-1560 ± 45	-1452 ± 55	-1440 ± 42	$\mathbf{-502 \pm 66}$

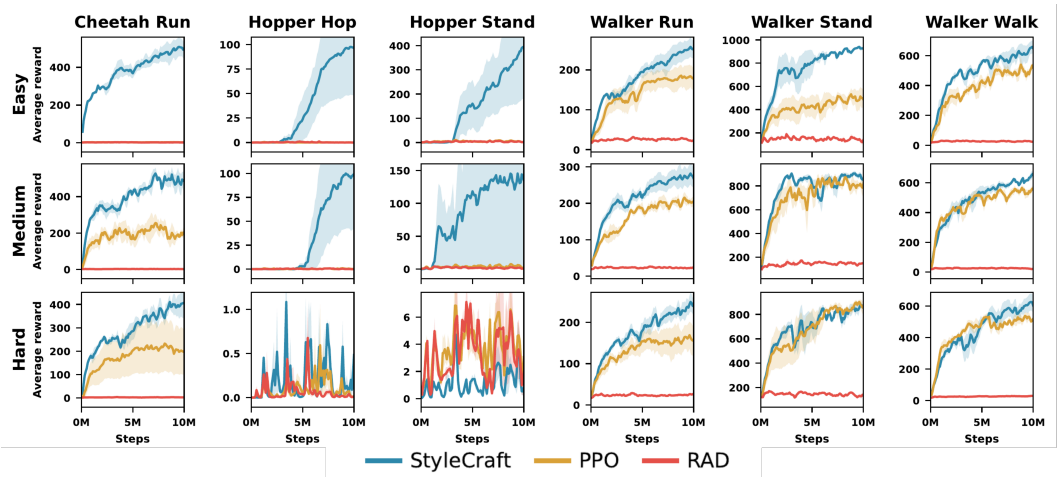


Figure 4: Learning curves for six DCS tasks at three background-distraction levels: Easy, Medium, Hard. Each curve shows the evaluation return in the distracted environment, averaged over 3 seeds and 6 parallel environments, each associated with own seed. Shaded regions denote the standard error of the mean. StyleCraft outperforms PPO and RAD in most settings, while none of the methods learns Hopper Hop and Hopper Stand at the Hard level.

6.3 Known Axis Generalization with KAGE Bench

We evaluate StyleCraft on KAGE Bench Cherepanov et al. (2026), which isolates visual distribution shifts along independently controlled axes while preserving the underlying task. We use four challenging axes: *background*, which changes the scene behind the agent; *distractors*, which adds task irrelevant objects with varying appearance and motion; *effects*, which modifies scene lighting; and *filters*, which applies global image transformations such as color, noise, pixelation, and vignette changes. For each axis, training and evaluation use disjoint visual configurations from the corresponding KAGE Bench protocol. This setting tests whether StyleCraft can generalize across distinct and independently controlled sources of visual variation.

6.4 ManiSkill and Real Robot under Table Color Shift

We additionally evaluate StyleCraft on a robotic manipulation task in ManiSkill and reproduce the corresponding scene on a real robot. The simulation experiment uses three table colors during training and three held out colors during evaluation while preserving the manipulation objective.

To test whether the learned visual robustness transfers beyond simulation, we recreate the tabletop scene on the physical system and evaluate the policy under the same in distribution and out of distribution color split. This setting introduces both unseen table appearance and the additional visual differences associated with simulation to real transfer.

We compare StyleCraft with PPO and report success rate separately for the training colors and held out colors in simulation and on the real robot. This final setting tests whether the visual robustness learned in simulation survives both unseen table appearances and the additional appearance gap introduced by physical deployment.

7 Results

KAGE Bench. StyleCraft achieves the highest average evaluation return across the selected KAGE Bench configurations, reaching -502 ± 66 compared with -655 ± 64 for the strongest baseline, RAD (Table 1). StyleCraft obtains the highest mean return in 7 of the 10 configurations and leads on all five background shifts, as well as on Effects 3 and Filters 2. The remaining cases are more heterogeneous, with PPG performing best on Distractors 5 and RAD on Filters 1 and Filters 3. Overall, the results show that latent augmentation provides broad gains across different sources of visual variation, with the most consistent improvement observed under background shifts.

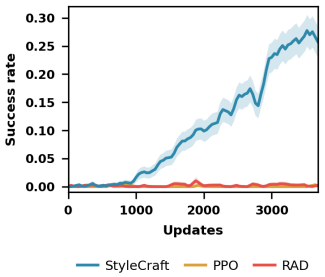


Figure 5: Eval. SR on RL4VLA under unseen table colors for StyleCraft, PPO, RAD.

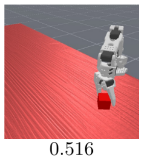
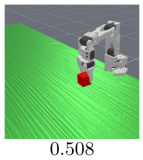
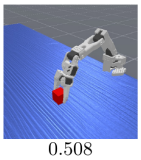
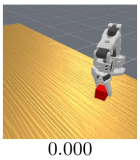
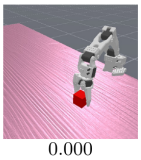
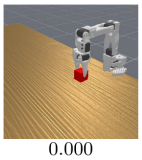
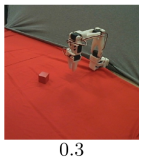
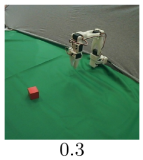
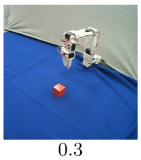
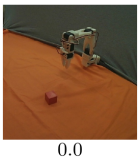
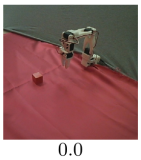
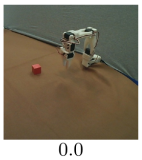
	ID (train colors)			OOD (eval colors)		
Simulation						
PPO	0.516	0.508	0.508	0.000	0.000	0.000
StyleCraft (ours)	**0.609**	**0.562**	**0.562**	**0.442**	**0.391**	**0.390**
Real robot						
PPO	0.3	0.3	0.3	0.0	0.0	0.0
StyleCraft (ours)	**0.5**	**0.5**	**0.5**	**0.3**	**0.2**	**0.3**

Figure 6: Success rate in simulation and on the real robot for in-distribution and out-of-distribution table colors. Photos show the real setup. Best per column in bold. PPO fails on all OOD colors, while StyleCraft transfers to both unseen colors and the real robot.

RL4VLA. Figure 5 shows a clear separation between StyleCraft and the baselines under unseen table colors. StyleCraft steadily increases its evaluation success rate and reaches approximately 0.27 by the end of training, whereas PPO and RAD remain close to zero. Since the manipulation objective is unchanged and the train evaluation difference is introduced through table appearance, this result indicates that the learned style variation transfers directly to the robotic manipulation policy.

DCS Locomotion. StyleCraft obtains the highest or tied mean return in 16 of the 18 task and difficulty combinations shown in Figure 4. It achieves the highest mean on all six tasks at both Easy and Medium levels. Under the more severe Hard shift, StyleCraft remains strongest on Cheetah Run, Walker Run, and Walker Walk, while all methods fail on Hopper Hop and PPO performs better on Hopper Stand and Walker Stand. The gains are particularly large on several tasks: on Cheetah Run at Easy difficulty, StyleCraft reaches 506 ± 47, compared with 4 ± 0 for PPO and 3 ± 1 for RAD. Overall, StyleCraft remains robust as the background shift becomes more severe, although the improvement is not uniform across all tasks.

ManiSkill and Real Robot. The ManiSkill experiment provides the strongest test of whether the learned invariance survives beyond the training renderer. In simulation, StyleCraft improves performance on the training colors and, more importantly, retains success rates between 0.390 and 0.442 on the three held out table colors, while PPO obtains zero success on all three (Figure 6). The same pattern transfers to the physical setup. StyleCraft reaches a success rate of 0.5 on all three in distribution colors compared with 0.3 for PPO, and retains 0.2 to 0.3 success on the unseen colors, whereas PPO again fails in every OOD case. This suggests that the visual invariance induced by latent augmentation transfers not only across table colors but also across the additional appearance gap between simulation and the real scene.

8 Conclusion and Future Work

We introduced StyleCraft, a two phase framework for visual generalization in reinforcement learning that combines task style disentanglement with latent augmentation. Across KAGE Bench, DCS locomotion, RL4VLA manipulation, and ManiSkill with real robot transfer, StyleCraft improves robustness to unseen visual changes and consistently remains effective beyond the visual conditions observed during training. Our results suggest that augmentation in a learned latent space is a promising alternative to relying exclusively on predefined pixel transformations. Rather than specifying all visual variation in advance, latent augmentation can use the structure learned by the model itself to construct additional training views while preserving the task representation. An important direction for future work is to study this idea with larger and more expressive generative models capable of representing substantially more complex visual scenes. Such models may enable latent augmentation beyond relatively simple appearance changes and make the approach applicable to richer real world observations and more challenging robotic settings.

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 complete StyleCraft training procedure and objectives in the main text, together with the evaluation protocols, metrics, number of seeds, parallel environments, and computational setup used in our experiments. Additional implementation details, hyperparameters, exact training and evaluation visual configurations, and environment specific settings are provided in the appendix and supplementary material in code.

REFERENCES

Sumeet Batra and Gaurav S. Sukhatme. Zero shot generalization of vision-based RL without data augmentation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 3251–3273. PMLR, 2025.

David Bertoin, Adil Zouitine, Mehdi Zouitine, and Emmanuel Rachelson. Look where you look! saliency-guided q-networks for generalization in visual reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30693–30706, 2022.

Egor Cherepanov, Daniil Zelezetsky, Aleksandr Panov, and Alexey Kovalev. KAGE-bench: Fast known-axis visual generalization evaluation for reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=fTwFUnHvKr.

Karl W. Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2020–2027. PMLR, 2021.

Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13611–13617. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9561103.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 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. In *Advances in Neural Information Processing Systems*, volume 32, pp. 13956–13968, 2019.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19884–19895, 2020a.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine*

Learning, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 2020b.

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. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Bogdan Mazoure, Ahmed M. Ahmed, Patrick MacAlpine, R. Devon Hjelm, and Andrey Kolobov. Cross-trajectory representation learning for zero-shot generalization in rl. In *International Conference on Learning Representations*, 2022.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8787–8798. 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.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021a.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021b.

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

A Model Architecture and Hyperparameters

Table 3: Training hyperparameters per environment.

	KAGE-Bench	**DMControl (cheetah-run)**	**ManiSkill3 (SO101 grasp)**	**SimplerEnv (put-on-plate)**
RL algorithm	PPO	PPO	PPO	PPO
Parallel environments	640	6	1024	64
Rollout length	16	512	16	20–80
Transitions per update	10,240	3,072	16,384	$\leq$5,120
Training budget	40M env steps	16.2M frames (action repeat 2)	60M env steps	4,000 updates
Phase-2 start	25M steps	3.1M frames	– (single phase)	separate run, 4,000 updates
Discount γ	0.999	0.99	0.9	0.99
GAE λ	0.95	0.95	0.9	0.95
Optimizer	Adam (ϵ=10^{-5})	Adam	Adam (ϵ=10^{-5})	Adam
Learning rate	$5 \cdot 10^{-4}$, constant	$3 \cdot 10^{-4}$, linear decay to 0	$3 \cdot 10^{-4}$, constant	$3 \cdot 10^{-4}$ (encoder $3 \cdot 10^{-6}$), constant
Epochs / minibatches	3 / 8	4 / 8	4 / 32	4 / $\leq$8
Clip ratio	0.2 (value clipped)	0.2	0.2	0.2
Value coefficient	0.5	0.5	0.5	0.5
Entropy coefficient	0.01	0	0	0.01 (pose), 0 (gripper)
Max grad norm	0.5	0.5	0.5	0.5
Target KL	–	0.05	0.2	–
Advantage normalisation	per minibatch	per minibatch	per minibatch	per rollout
z_{task} / z_{style} size	128 / 64	$1 \times 21 \times 21$ / 4	$4 \times 8 \times 8$ / 4	$1 \times 16 \times 16$ / 4
Reconstruction loss	MSE, weight 1.0, through GRL (λ=1)	L1 + LPIPS, weight 1.0 (64 frames)	L1 + LPIPS, weight 1.0 (32 frames)	L1 + LPIPS, weight 1.0 (128 rollout + 64 colour-diverse frames)
KL on z_{style}	$2.3 \cdot 10^{-5}$	–	–	–
Invariance, phase 1	MSE on z_{task} under pixel augmentation (12 types, intensity 0.5), weight 1.0	MSE on z_{task} under pixel augmentation (intensity 0.5), weight 1.0	NLL of clean action under decoder-restyled view (permuted z_{style}), weight 0.1; value MSE 0.5	–
Invariance, phase 2	MSE on z_{task} for frozen-decoder views with z_{style} interpolated at $\alpha \in \{0.25, 0.5, 0.75\}$, weight 1.0	same as KAGE-Bench, weight 1.0	–	same as KAGE-Bench, weight 1.0 (linear warm-up over 200 updates)
Other	flow-matching style prior, weight 1.0	AMP; return normalisation	–	reverse curriculum (16 stages); SIL (256 episodes)
Evaluation	128 episodes, greedy, held-out configuration	20 episodes per axis, deterministic	128 episodes, 12 held-out table colours	256 episodes, 12 held-out table colours

Table 2: Model architecture per environment. Parameter counts (in brackets) were obtained by instantiating the modules with the configuration of the main run. GN = GroupNorm($\min(32, C)$), ResBlock($c_{\text{in}} \to c_{\text{out}}$) = [GN, SiLU, Conv3×3]×2 with a residual (1×1 conv if $c_{\text{in}} \neq c_{\text{out}}$). Modules marked † are used only during training.

	KAGE-Bench	**DMControl (cheetah-run)**	**ManiSkill3 (SO101 grasp)**	**SimplerEnv (WidowX put-on-plate)**
Observation	RGB $3 \times 128 \times 128$	3 stacked RGB frames $9 \times 84 \times 84$ (encoded per frame)	RGB $3 \times 128 \times 128$ + 17-d proprioceptive state	RGB $3 \times 128 \times 128$ (third-person camera)
Action space	8 discrete	6-d continuous, $\tanh$-Gaussian	6-d continuous (joint deltas), Gaussian	7 categorical: 6×21 bins + 2 (gripper)
Shared encoder	Nature CNN: Conv(32,8,4), Conv(64,4,2), Conv(64,3,1), ReLU; Linear 9216→128, LayerNorm [1.26M]	Conv3×3(32); Conv s2; ResBlock(32→64); Conv s2; ResBlock(64); GN, SiLU, Conv3×3→4 ⇒ $4 \times 21 \times 21$ [181K]	Conv3×3 s2(16); 3 stages of ResBlocks (16/32/64) + Conv s2; 2×ResBlock(64); GN, SiLU, Conv→4 ⇒ $4 \times 8 \times 8$ [374K]	Conv3×3(32); 3 stages of 2×ResBlock (32/64/128) + Conv s2; 2×ResBlock(128); GN, SiLU, Conv→4 ⇒ $4 \times 16 \times 16$ [1.49M]
Task branch z_{task}	output of the encoder, 128-d	Conv(32), 2×ResBlock(32), GN, SiLU, Conv→1 ⇒ $1 \times 21 \times 21$ [39K]	Conv(32), 2×ResBlock(32), GN, SiLU, Conv→4 ⇒ $4 \times 8 \times 8$ [40K]	Conv(32), 2×ResBlock(32), GN, SiLU, Conv→1 ⇒ $1 \times 16 \times 16$ [39K]
Style branch $z^{\dagger}_{\text{style}}$	separate Nature CNN, Linear 9216→256, ReLU; μ, $\log \sigma^2$: Linear 256→64 [2.47M]	Conv(32), 2×ResBlock(32), GN, SiLU, global avg. pool, Linear 32→4 [39K]	Conv(32), 2×ResBlock(32), GN, SiLU, global avg. pool, Linear 32→4 [39K]	Conv(32), 2×ResBlock(32), GN, SiLU, global avg. pool, Linear 32→4 [39K]
Memory	GRUCell 128→128 [99K]	none (frame stack)	none	ConvGRU, hidden $32 \times 16 \times 16$, 3×3 kernels [29K]
Policy head	MLP 128→256→256→8, ReLU [101K]	Linear 1323→50, LayerNorm, $\tanh$ (shared with critic) [66K]; MLP 50→1024→1024→6, ReLU [1.11M]	Conv(16)×2, SiLU, AvgPool 4×4, Linear 256→256; state Linear 17→256; MLP 512→512→6 [339K]	AvgPool 4×4, Linear 512→128 logits [66K]
Value head†	MLP 128→256→256→1 [99K]	MLP 50→1024→1024→1 [1.10M]	MLP 512→512→1 [263K]	Linear 512→1 [0.5K]
Decoder†	input $[\text{GRL}(z_{\text{task}}), \mu]$; Linear 192→8192; 4×ConvT(4,2) 128/64/32/3; sigmoid [2.01M]	input $[z_{\text{task}}, z_{\text{style}}]$; Conv(64), ResBlocks, 2 nearest-upsampling stages (64/32); GN, SiLU, Conv→3 [217K]	input $[z_{\text{task}}, z_{\text{style}}]$; Conv(64), ResBlocks, 4 nearest-upsampling stages (64/32/16/16); GN, SiLU, Conv→3 [413K]	input $[z_{\text{task}}, z_{\text{style}}]$; Conv(128), ResBlocks, 3 nearest-upsampling stages (128/64/32); GN, SiLU, Conv→3 [1.63M]
Other†	style prior: flow-matching MLP 65→128→128→128→64, SiLU [50K]	LPIPS-AlexNet, frozen	LPIPS-AlexNet, frozen	LPIPS-AlexNet, frozen
Params, acting	**1.46M**	**1.39M**	**0.75M**	**1.62M**
Params, trainable	**6.08M**	**2.78M**	**1.47M**	**3.28M**
Frozen (training only)	phase-2 teacher 5.73M	LPIPS 2.47M; phase-2 generator 0.50M	LPIPS 2.47M	LPIPS 2.47M; phase-2 generator 3.28M