Learning to Act Robustly with View-Invariant Latent Actions

Supplementary Material

A. Implementation Details

We provide the implementation details used for view generation, NVS generation, VILA training, baselines and diffusion policy training.

A.1. View Configurations

For reproducibility, Tables 1 and 2 list the exact world-frame camera poses (positions and MuJoCo quaternions) for all views used in this paper.

A.2. ZeroNVS Configurations

We utilized the official open-source implementation of ZeroNVS[1] and the pretrained checkpoint to generate novel views in our real-world experiments using SO-ARM101. The hyperparameters for this process are summarized in Table 3. We created 26 more combinations of novel views of our dataset and selected 4 views of azimuth, vertical and optical axis translation of $(0.0°, -5\text{cm}, 5\text{cm})$, $(5°, 0\text{cm}, -5\text{cm})$, $(-5°, 5\text{cm}, 0\text{cm})$ and the original data representing $(0.0°, 0.0\text{cm}, 0.0\text{cm})$ to cover all variations of each axis.

A.3. VILA Configurations

We train VILA with a two-stage pipeline: (i) latent action pre-training from multi-view videos, and (ii) latent behavior cloning in the learned latent space. Unless otherwise specified, we use a single configuration across all datasets and tasks, summarized in Table 4.

A.4. Baseline Implementations

CLASS. We adapt CLASS from official open-source implementation[2] to our setting by applying its contrastive objective on top of our Stage-1 encoder, using the same image resolution and latent dimension as VILA.

ReViWo. For ReViWo, we follow the official implementation[3] and adjust its input image size and latent dimensionality to match VILA. Because ReViWo is ViT-based and outputs patch tokens, we choose the token and hidden dimensions so that the resulting representation has the same dimensionality as VILA, ensuring a fair comparison across methods.

VLAs. For the real-world VLA baselines, we fine-tune the models on the ZeroNVS-augmented dataset starting from the pretrained checkpoints `lerobot/pi0.5_base` and `lerobot/smolvla_base`. We use implementations from LeRobot[4] and the main hyperparameters are summarized in Table 7.

[1] https://github.com/kylesargent/ZeroNVS
[2] https://github.com/sean1295/CLASS
[3] https://github.com/Trevor-emt/Reviwo

View	x	y	z	q_w	q_x	q_y	q_z
View 00	0.0981	-0.4091	1.4249	0.9451	0.3051	0.0361	0.1117
View 01	0.2021	-0.4080	1.3951	0.9159	0.3304	0.0773	0.2144
View 02	0.0473	-0.4811	1.3677	0.9293	0.3662	0.0180	0.0456
View 03	0.0204	-0.5587	1.2741	0.8937	0.4483	0.0082	0.0163
View 04	0.2175	-0.5482	1.2234	0.8577	0.4786	0.0915	0.1639
View 05	0.2523	-0.3380	1.4240	0.9029	0.2925	0.0971	0.2998
View 06	0.3973	-0.1771	1.4131	0.7947	0.2686	0.1743	0.5157
View 07	0.3278	-0.3572	1.3663	0.8665	0.3429	0.1335	0.3374
View 08	0.3985	-0.3722	1.2939	0.8271	0.3965	0.1722	0.3592
View 09	0.4504	-0.3576	1.2489	0.7957	0.4215	0.2036	0.3843
View 10	0.3738	0.0263	1.4587	0.6572	0.1817	0.1950	0.7051
View 11	0.4394	-0.0577	1.4061	0.7102	0.2465	0.2162	0.6230
View 12	0.5130	-0.0142	1.3349	0.6579	0.2846	0.2769	0.6400
View 13	0.5352	-0.1022	1.2945	0.6950	0.3327	0.2752	0.5749
View 14	0.5651	0.0328	1.2635	0.6105	0.3135	0.3322	0.6470
View 15	0.2890	0.2414	1.4574	0.4081	0.1136	0.2428	0.8727
View 16	0.4437	0.1645	1.3781	0.5336	0.2036	0.2926	0.7669
View 17	0.3112	0.4094	1.3336	0.2929	0.1271	0.3773	0.8693
View 18	0.4612	0.2553	1.3178	0.4627	0.2092	0.3549	0.7850
View 19	0.4271	0.3932	1.2399	0.3535	0.1908	0.4350	0.8059
View 20	0.2071	0.3366	1.4443	0.2611	0.0775	0.2737	0.9225
View 21	0.2687	0.3871	1.3800	0.2793	0.1059	0.3384	0.8923
View 22	0.2279	0.4535	1.3414	0.2124	0.0903	0.3806	0.8955
View 23	0.2285	0.4949	1.2941	0.1935	0.0927	0.4220	0.8808
View 24	0.1605	0.5482	1.2553	0.1258	0.0657	0.4585	0.8773

Table 1. **Exact Camera Poses for the 25 Main Views.** Positions are given in the world frame (MuJoCo's default metric units, i.e., meters), and orientations are MuJoCo quaternions (q_w, q_x, q_y, q_z).

View	x	y	z	q_w	q_x	q_y	q_z
Extra View 00	0.2902	0.0585	1.5040	0.6269	0.1256	0.1568	0.7527
Extra View 01	0.3111	0.0577	1.4936	0.6240	0.1391	0.1672	0.7505
Extra View 02	0.3316	0.0570	1.4825	0.6209	0.1525	0.1776	0.7481
Extra View 03	0.2880	0.0686	1.5040	0.6137	0.1228	0.1590	0.7635
Extra View 04	0.3294	0.0686	1.4825	0.6078	0.1494	0.1803	0.7588
Extra View 05	0.2854	0.0786	1.5040	0.6003	0.1200	0.1611	0.7741
Extra View 06	0.3063	0.0793	1.4936	0.5975	0.1331	0.1720	0.7718
Extra View 07	0.3268	0.0800	1.4825	0.5945	0.1462	0.1829	0.7693

Table 2. **Exact Camera Poses for the Extrapolated Views.** Positions are given in the world frame (MuJoCo's default metric units, i.e., meters), and orientations are MuJoCo quaternions (q_w, q_x, q_y, q_z).

A.5. Diffusion Policy Configurations

For simulation experiments, we used the official open-source implementation[5] of diffusion policy in Robomimic [1]. The main hyperparameters are summarized in Table 5. All methods (ours and baselines) use the same diffusion-policy architecture and training settings.

[4] https://github.com/huggingface/lerobot
[5] https://github.com/ARISE-Initiative/robomimic

Table 3. **ZeroNVS Hyperparameters.** Configuration for ZeroNVS for real-world data augmentation.

Hyperparameter	Value
Noise Scheduler	DDIM
Inference Steps	50
Guidance Scale	7.5
FOV deg	13.0°
Default Elevation	-10.0°
Default Azimuth	0.0°
Default Distance	1.0m
Translation Azimuth	$\{-5.0, 0.0, 5.0\}^\circ$
Translation Vertical	{-5.0, 0.0, 5.0}cm
Translation Optical	{-5.0, 0.0, 5.0}cm

Table 4. **VILA Training Hyperparameters.** We use a two-stage training pipeline: Stage 1 latent action pre-training and Stage 2 latent behavior cloning.

Hyperparameter	Value
Stage 1: Latent Action Pre-training	
Optimizer	AdamW
Learning Rate	1×10^{-4}
Latent Dimension (D_z)	128 (sim.), 512 (real-world)
Prediction Horizon (K)	$1 \sim 10$
Time Indices per Batch (N)	16
Views per Batch (V)	8
Image Resolution	64×64 (sim.), 128×128 (real-world)
Distance Type	L_2
InfoNCE Temperature (τ)	1.0
λ_1 ($\mathcal{L}_{\text{W-NCE}}$)	1.0
λ_2 ($\mathcal{L}_{\text{struct}}$)	1.0
Weighting Temperature (β)	0.001
Epochs	100
Gradient Clipping Norm	1.0
Target EMA Coef.	0.001
Target Update Interval (Epochs)	1
IDM Head Hidden Dim.	1024
FDM Head Hidden Dim.	1024
Stage 2: Latent Behavior Cloning	
Optimizer	AdamW
Learning Rate	5×10^{-5}
BC Batch Size	256
Prediction Horizon (K)	10
Image Resolution	64×64
Epochs	100

B. Additional Experimental Results

B.1. Failure Cases

Figure 1 shows representative failures in simulation and the real world. The dominant failure mode is object mislocalization, with occasional grasp failures in simulation. In the real-world experiments, we only observed mislocalization failures. These examples indicate that errors under viewpoint shifts are largely due to localization.

Table 5. **Diffusion Policy Hyperparameters for Simulations.** Configuration for the diffusion policy used for policy training in simulations.

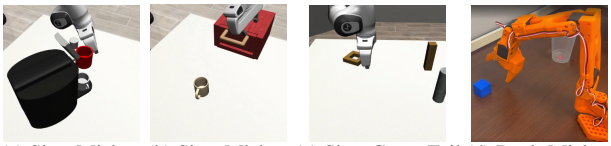
Hyperparameter	Value
Learning Rate	1×10^{-4}
Encoder Learning Rate (In Fine-tune Setting)	1×10^{-5}
Optimizer	AdamW
Optimizer Scheduler	Cosine
Training Epochs (In Fine-tune setting)	50
Training Epochs (In Frozen setting)	100
Steps per Epoch	1,000
Warmup Steps	500
Batch Size (for Lift)	100
Batch Size (for others)	1,000
Image Resolution	64×64
Observation Horizon	1
Prediction Horizon	16
Action Horizon	8
Noise Scheduler	DDIM
Beta Scheduler	squaredcos_cap_v2
Diffusion Training Steps	100
Diffusion Inference Steps	10

Table 6. **Diffusion Policy Hyperparameters for Real-World.** Configuration for the Diffusion Policy for policy finetuning in real-world experiments.

Hyperparameter	Value
Learning Rate	1×10^{-4}
Encoder Learning Rate	1×10^{-5}
Optimizer	AdamW
Optimizer Scheduler	Cosine
Training Steps	200,000
Warmup Steps	500
Batch Size	64
Image Resolution	128×128
Observation Horizon	2
Prediction Horizon	16
Action Horizon	16
Noise Scheduler	DDIM
Beta scheduler	squaredcos_cap_v2
Diffusion Training Steps	100
Diffusion Inference Steps	10

Table 7. **VLA Hyperparameters for Real World.** Configuration for the VLAs for policy finetuning in real-world experiments.

Hyperparameter	$\pi_{0.5}$	SmolVLA
Learning Rate	2.5×10^{-5}	1×10^{-4}
Optimizer	AdamW	AdamW
Optimizer Scheduler	Cosine	Cosine
Training Steps	30,000	200,000
Warmup Steps	1,000	1,000
Batch Size	32	64
Image Resolution	224×224	512×512
Observation Horizon	1	1
Prediction Horizon	50	50
Action Horizon	50	50
FM Inference Steps	10	10



(a) Sim: Misloc. (b) Sim: Misloc. (c) Sim: Grasp Fail (d) Real: Misloc.

Figure 1. **Representative failure cases.** The dominant failure mode is object mislocalization in both simulation and the real world, with occasional grasp failures observed in simulation.

B.2. RGB-D Inputs

We also evaluate RGB-D inputs on the Lift task. Adding depth improves Vanilla from 77.0% to 80.7% unseen-view success, while VILA+RGB-D further improves from 94.7% to 95.3%. This suggests that the proposed objective remains useful and complementary even when depth information is available.

B.3. Entropy Results

Tables 8–11 report the same view and action entropy metrics on the remaining tasks (Square, Stack Three, Coffee, and Mug Cleanup). Across all four datasets, we observe the same qualitative trend as in Lift: VILA consistently attains the highest view entropy on both seen and unseen views, while achieving the lowest action entropy among all methods, both before and after policy fine-tuning.

Table 8. **View and Action Entropy (Square).** Entropy-based analysis of representation quality before and after policy fine-tuning in the Square dataset. "Seen" and "Unseen" denote view entropies (higher is better, ↑) computed over the 25 views, while "Action" denotes action entropy (lower is better, ↓) based on 10 clustered action classes. The upper-bound row corresponds to the entropy of a uniform distribution over 25 views (Seen/Unseen) and 10 action clusters (Action).

Model	Seen (↑)	Unseen (↑)	Action (↓)
Upper Bound	3.219	3.219	2.303
Before Fine-tuning			
VILA (Ours)	**2.894**	**2.705**	**0.631**
CLASS	2.607	1.982	0.861
ReViWo	2.325	1.748	1.316
Fine-Tuned			
VILA (Ours)	**2.917**	**2.712**	**0.598**
Vanilla	2.607	1.982	0.802
CLASS	2.449	1.57	0.912
ReViWo	2.443	1.776	1.384
KYC	1.653	1.143	1.361

Table 9. **View and Action Entropy (Stack Three).** Entropy-based analysis of representation quality before and after policy fine-tuning in the Stack Three dataset. "Seen" and "Unseen" denote view entropies (higher is better, ↑) computed over the 25 views, while "Action" denotes action entropy (lower is better, ↓) based on 10 clustered action classes. The upper-bound row corresponds to the entropy of a uniform distribution over 25 views (Seen/Unseen) and 10 action clusters (Action).

Model	Seen (↑)	Unseen (↑)	Action (↓)
Upper Bound	3.219	3.219	2.303
Before Fine-tuning			
VILA (Ours)	**3.111**	**3.096**	**0.362**
CLASS	2.950	2.826	0.463
ReViWo	2.592	2.303	1.232
Fine-Tuned			
VILA (Ours)	**3.076**	**3.046**	**0.339**
Vanilla	2.986	2.883	0.488
CLASS	2.886	2.723	0.546
ReViWo	2.667	2.341	1.180
KYC	2.118	2.073	1.053

Table 10. **View and Action Entropy (Coffee).** Entropy-based analysis of representation quality before and after policy fine-tuning in the Coffee dataset. "Seen" and "Unseen" denote view entropies (higher is better, ↑) computed over the 25 views, while "Action" denotes action entropy (lower is better, ↓) based on 10 clustered action classes. The upper-bound row corresponds to the entropy of a uniform distribution over 25 views (Seen/Unseen) and 10 action clusters (Action).

Model	Seen (↑)	Unseen (↑)	Action (↓)
Upper Bound	3.219	3.219	2.303
Before Fine-tuning			
VILA (Ours)	**2.554**	**1.924**	**0.487**
CLASS	2.361	1.075	0.700
ReViWo	2.182	1.181	1.053
Fine-Tuned			
VILA (Ours)	**2.426**	**1.632**	**0.488**
Vanilla	2.204	0.809	0.710
CLASS	0.995	0.287	0.812
ReViWo	2.201	1.429	1.222
KYC	1.274	0.457	1.010

B.4. UMAP Visualization

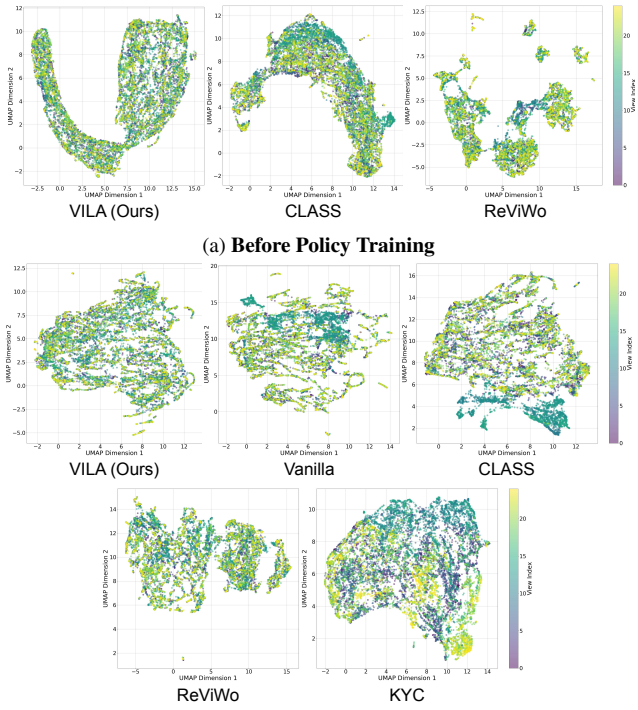
UMAP plots. In Figures 2–6, we visualize encoder representations across the 25 camera views with UMAP [2] on Lift, Square, Stack-Three, Coffee, and Mug-Cleanup. For baselines, unseen views (especially Views 10–14) form separate clusters, whereas VILA intermingles them with seen views. This pattern matches our entropy analysis that VILA

Table 11. **View and Action Entropy (Mug Cleanup).** Entropy-based analysis of representation quality before and after policy fine-tuning in the Mug Cleanup dataset. "Seen" and "Unseen" denote view entropies (higher is better, ↑) computed over the 25 views, while "Action" denotes action entropy (lower is better, ↓) based on 10 clustered action classes. The upper-bound row corresponds to the entropy of a uniform distribution over 25 views (Seen/Unseen) and 10 action clusters (Action).

Model	**Seen (↑)**	**Unseen (↑)**	**Action (↓)**
Upper Bound	3.219	3.219	2.303
Before Fine-tuning			
VILA (Ours)	**2.957**	**2.891**	**0.452**
CLASS	2.564	2.071	0.654
ReViWo	2.383	1.941	1.161
Fine-Tuned			
VILA (Ours)	**2.920**	**2.793**	**0.404**
Vanilla	2.721	2.272	0.614
CLASS	2.153	1.569	0.824
ReViWo	2.437	1.904	1.302
KYC	1.765	1.176	1.142

has the highest view entropy.

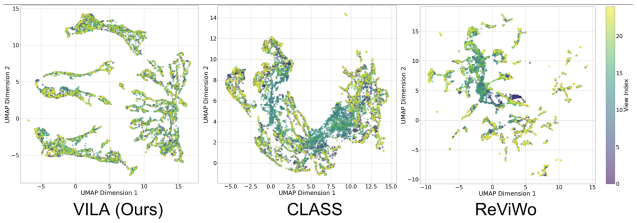
Action-cluster UMAPs. In Figures 7–11, we reuse the same encoder representations as in the view-based UMAPs, but color them by $K=10$ action clusters obtained by applying k-means to 10-step GT action sequences $\{a_t, \ldots, a_{t+9}\}$.



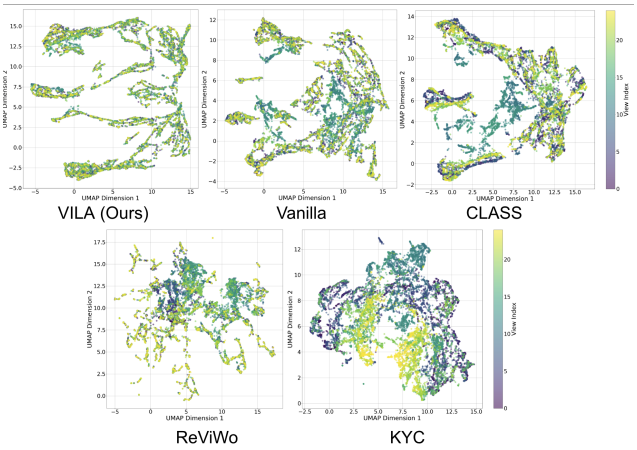
(a) **Before Policy Training**

(b) **After Policy Training**

Figure 2. **UMAP of encoder representations across 25 views on Lift.** On Lift, baselines show distinct clusters for unseen views (especially for views 10–14), whereas VILA representations are more uniformly mixed across views, indicating stronger view invariance both before and after policy training.

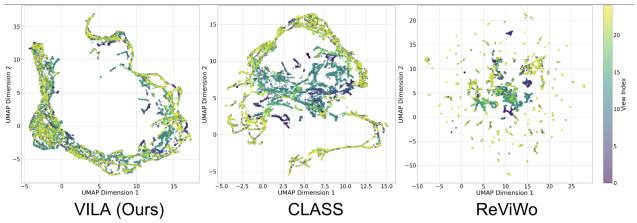


(a) **Before Policy Training**

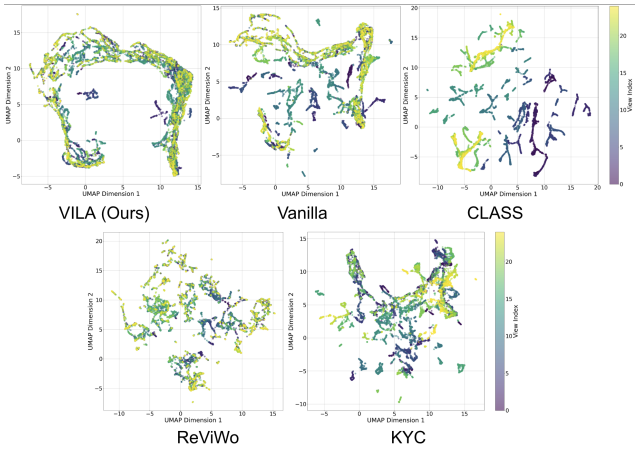


(b) **After Policy Training**

Figure 3. **UMAP of encoder representations across 25 views on Square.**

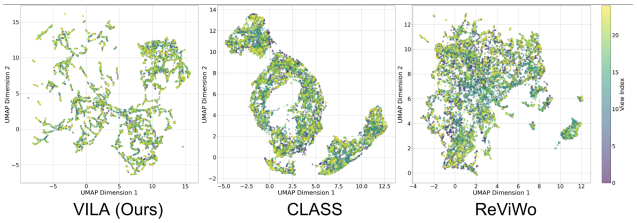


(a) **Before Policy Training**

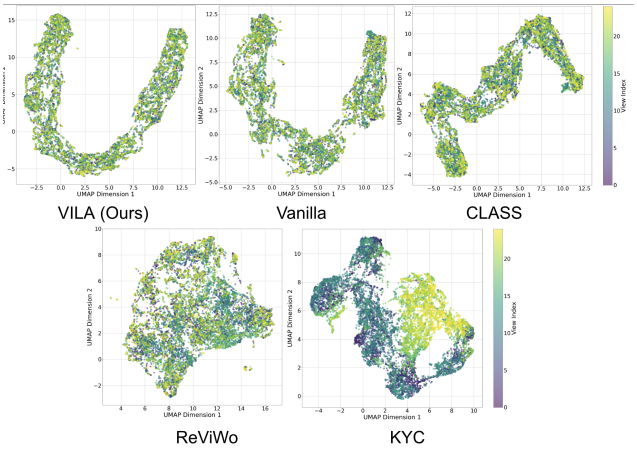


(b) **After Policy Training**

Figure 5. **UMAP of encoder representations across 25 views on Coffee.**

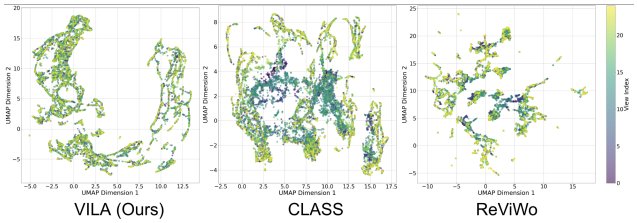


(a) **Before Policy Training**

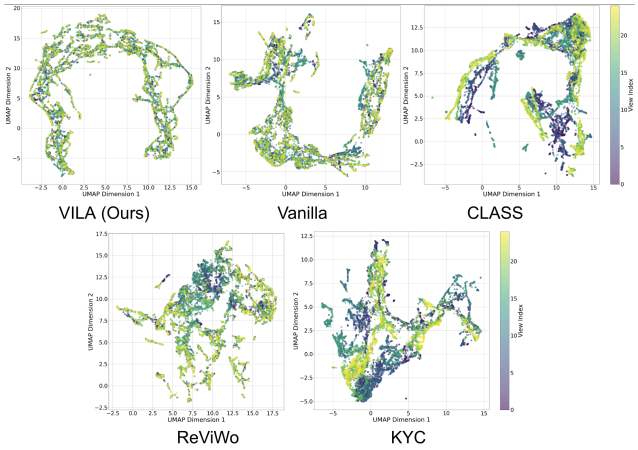


(b) **After Policy Training**

Figure 4. **UMAP of encoder representations across 25 views on Stack Three.**

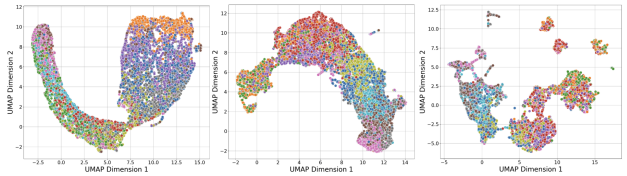


(a) **Before Policy Training**

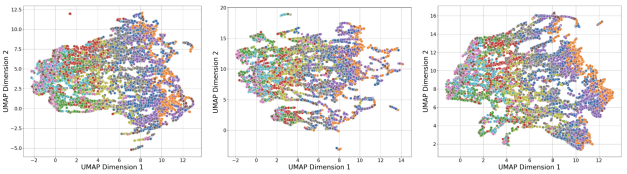


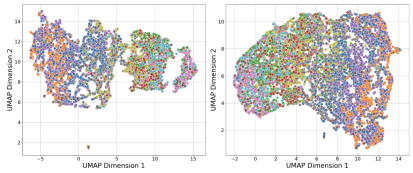
(b) **After Policy Training**

Figure 6. **UMAP of encoder representations across 25 views on Mug Cleanup.**



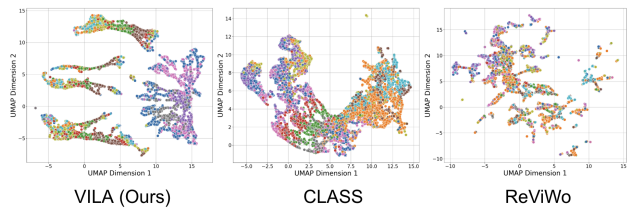
(a) **Before Policy Training**



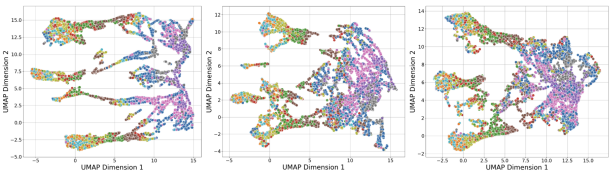


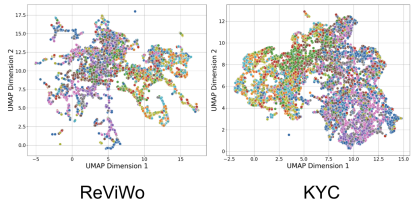
(b) **After Policy Training**

Figure 7. **UMAP of encoder representations colored by action clusters on Lift.**



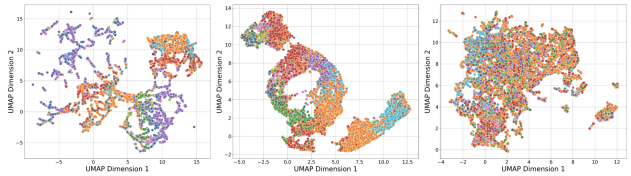
(a) **Before Policy Training**



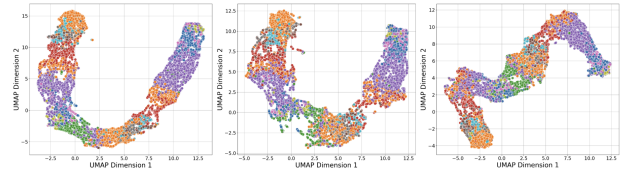


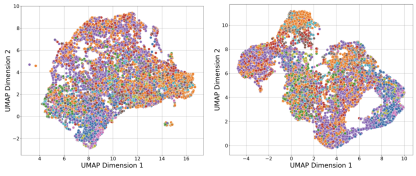
(b) **After Policy Training**

Figure 8. **UMAP of encoder representations colored by action clusters on Square.**



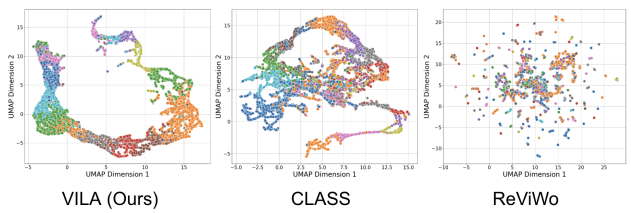
(a) **Before Policy Training**



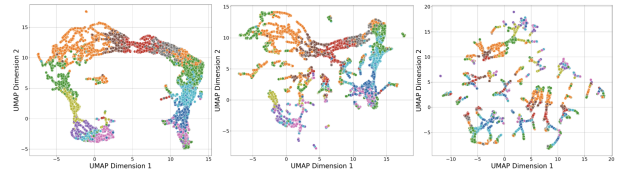


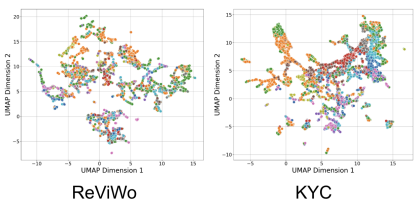
(b) **After Policy Training**

Figure 9. **UMAP of encoder representations colored by action clusters on Stack Three.**



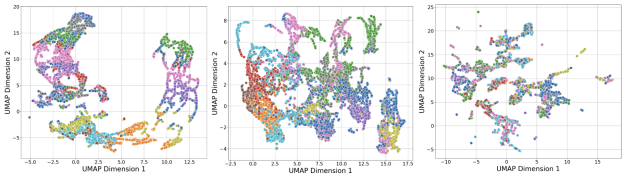
(a) **Before Policy Training**



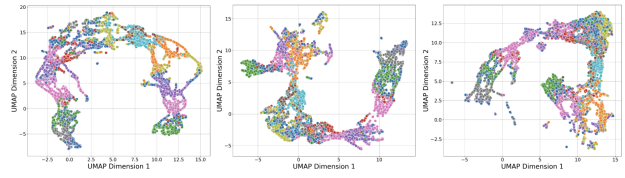


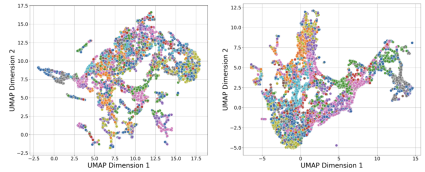
(b) **After Policy Training**

Figure 10. **UMAP of encoder representations colored by action clusters on Coffee.**



(a) **Before Policy Training**





(b) **After Policy Training**

Figure 11. **UMAP of encoder representations colored by action clusters on Mug Cleanup.**

References

[1] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021. 1

[2] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. 3