

A Network Structure

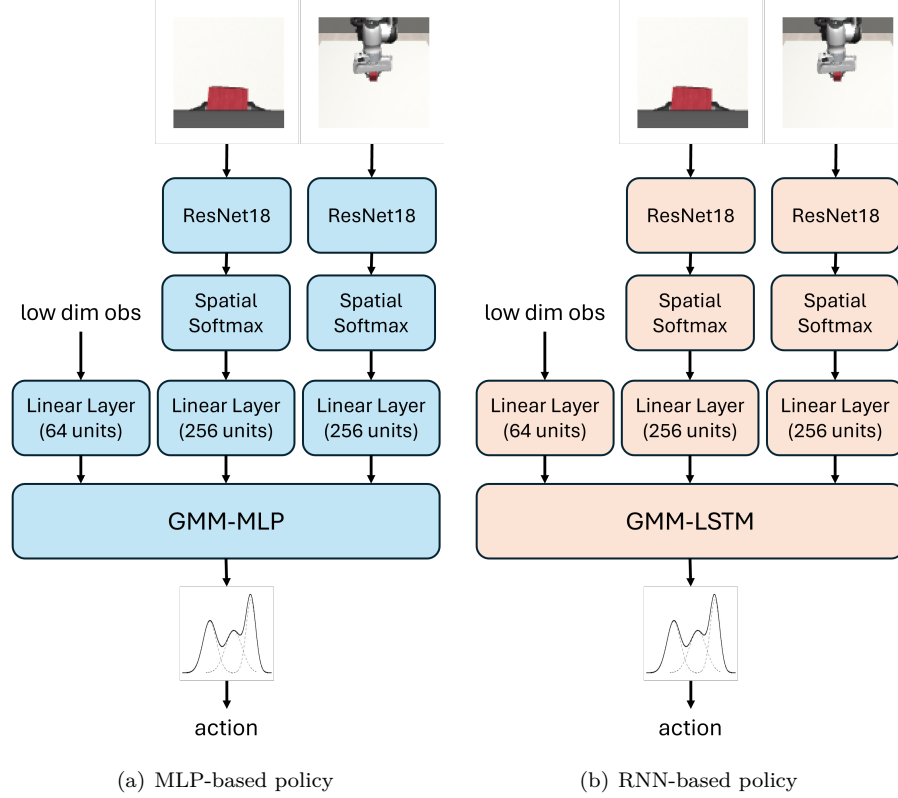


Figure 10: **Policy Network Structure.** The detailed architectures of the visuomotor policy networks. We adopt the RNN-based network for the Pick-And-Place-Can task, and the MLP-based network for other tasks.

| Hyperparameter | Default |
|-----------------------------|--------------------|
| Learning Rate | 1×10^{-4} |
| Action Decoder MLP Dims | [1024, 1024] |
| GMM Num Modes | 5 |
| Image Encoder | ResNet-18 |
| SpatialSoftmax (num-KP) | 64 |
| Image Embedding Layer | 256 units |
| Low Dim Obs Embedding Layer | 64 units |

Table 6: **MLP-based policy Hyperparameters.**

For the baseline methods, we demonstrate the detailed structures of the MLP-base policy network in Fig. 10(a), and the RNN-based policy network in Fig. 10(b). The hyperparameters of the MLP-base network and the RNN-based network are listed in TABLE 6 and TABLE 7 separately.

The policy network of the Perception Stitching method shares all the network structures and hyperparameters as that of the baseline method. The only difference is that it calculates the relative representations of the images after the 256-unit linear layer, as shown in Fig. 2 and equation 3.

B Perception Stitching Algorithm

The algorithm of the Perception Stitching (PeS) is shown in Algorithm 1. The first step of this algorithm is to collect an expert dataset \mathbb{D}_1 in the environment E_1 by teleoperating the robot by a proficient human expert.

Algorithm 1 Zero-shot Transfer with Perception Stitching**Collect Dataset 1 with random sampling:**

Task T in the environment E_1 with two visual configurations $o_1^{E_1}$ and $o_2^{E_1}$. Initialize an empty dataset $\mathbb{D}_1 \leftarrow \emptyset$.

for each game i of the task **do**

Random sample the initial state of the task.

Execute the Expert policy to collect the expert trajectory τ_i^1 of this game.

Push τ_i^1 into the dataset \mathbb{D}_1 .

end for

Collect Dataset 2 with trajectories replay:

Task T in the environment E_2 with two visual configurations $o_1^{E_2}$ and $o_2^{E_2}$. Initialize an empty dataset $\mathbb{D}_2 \leftarrow \emptyset$.

for each game i of the task **do**

Replay the trajectory τ_i^1 in E_2 to collect the trajectory τ_i^2 of this game.

Push τ_i^2 into the dataset \mathbb{D}_2 .

end for

Collect Anchor States:

Task T in the environment E_3 with two visual configurations $o_1^{E_1}$ and $o_2^{E_2}$.

K-means center set $\mathbb{C}_1 \leftarrow k - \text{means}(\mathbb{D}_1)$.

Select anchor set $\mathbb{A}_1 = \{a_1^{(j)}\}$ from \mathbb{D}_1 which are images closest to the K-means center set \mathbb{C}_1 .

Select anchor set $\mathbb{A}_2 = \{a_2^{(j)}\}$ from \mathbb{D}_2 which have the same indices as \mathbb{A}_1 in \mathbb{D}_1 .

Anchor set $\mathbb{A}_3 = \{a_3^{(j)}\}$ consists of images of $o_1^{E_1}$ from \mathbb{A}_1 and images of $o_2^{E_2}$ from \mathbb{A}_2 .

Train Modular Policies in Source Environments:

Environment E_1 has two visual configurations $o_1^{E_1}$, $o_2^{E_1}$ and low dimensional observation $o_l^{E_1}$.

Environment E_2 has two visual configurations $o_1^{E_2}$, $o_2^{E_2}$ and low dimensional observation $o_l^{E_2}$.

Initialize policy $f^{E_1}(g_1^{E_1}(o_1^{E_1}), g_2^{E_1}(o_2^{E_1}), o_l^{E_1}, \mathbb{A}_1)$ in E_1 .

Initialize policy $f^{E_2}(g_1^{E_2}(o_1^{E_2}), g_2^{E_2}(o_2^{E_2}), o_l^{E_2}, \mathbb{A}_2)$ in E_2 .

Optimize $f^{E_1}(g_1^{E_1}(o_1^{E_1}), g_2^{E_1}(o_2^{E_1}), o_l^{E_1}, \mathbb{A}_1)$ with dataset \mathbb{D}_1 with the Behavior Cloning algorithm Schaal (1996).

Optimize $f^{E_2}(g_1^{E_2}(o_1^{E_2}), g_2^{E_2}(o_2^{E_2}), o_l^{E_2}, \mathbb{A}_2)$ with dataset \mathbb{D}_2 with the Behavior Cloning algorithm Schaal (1996).

Perception Stitching:

Environment E_3 has visual configurations $o_1^{E_1}$ and $o_2^{E_2}$.

Initialize the visual encoder 1 with parameters from $g_1^{E_1}$.

Initialize the visual encoder 2 with parameters from $g_2^{E_2}$.

Initialize the action decoder with parameters from f^{E_1} .

Construct the stitched policy $f^{E_1}(g_1^{E_1}(o_1^{E_1}), g_2^{E_2}(o_2^{E_2}), o_l^{E_1}, \mathbb{A}_3)$.

Test the Stitched Policy in the Target Environment:

Rollout the stitched policy $f^{E_1}(g_1^{E_1}(o_1^{E_1}), g_2^{E_2}(o_2^{E_2}), o_l^{E_1}, \mathbb{A}_3)$ in E_3 .

Calculate the success rate of task T with the policy $f^{E_1}(g_1^{E_1}(o_1^{E_1}), g_2^{E_2}(o_2^{E_2}), o_l^{E_1}, \mathbb{A}_3)$ in E_3 .

| Hyperparameter | Default |
|-----------------------------|--------------------|
| Learning Rate | 1×10^{-4} |
| Action Decoder MLP Dims | [] |
| RNN Hidden Dim | 1000 |
| RNN Seq Len | 10 |
| GMM Num Modes | 5 |
| Image Encoder | ResNet-18 |
| SpatialSoftmax (num-KP) | 64 |
| Image Embedding Layer | 256 units |
| Low Dim Obs Embedding Layer | 64 units |

Table 7: **RNN-based policy Hyperparameters.**

Then we replay the expert trajectories in \mathbb{D}_1 to collect another expert dataset \mathbb{D}_2 in the environment E_2 . We collect the anchor set \mathbb{A}_1 in the dataset \mathbb{D}_1 via the K-means algorithm Hartigan & Wong (1979). Then we collect the anchor set \mathbb{A}_2 which has the same indices in \mathbb{D}_2 as \mathbb{A}_1 in \mathbb{D}_1 . For the environment E_3 with two visual configurations $o_1^{E_1}$ and $o_2^{E_2}$, we assemble an anchor set $\mathbb{A}_3 = \{a_3^{(j)}\}$ consists of images of $o_1^{E_1}$ from \mathbb{A}_1 and images of $o_2^{E_2}$ from \mathbb{A}_2 . In the next step, we train the two polices $f^{E_1}(g_1^{E_1}(o_1^{E_1}), g_2^{E_1}(o_2^{E_1}), o_l^{E_1}, \mathbb{A}_1)$ and $f^{E_2}(g_1^{E_2}(o_1^{E_2}), g_2^{E_2}(o_2^{E_2}), o_l^{E_2}, \mathbb{A}_2)$ in E_1 and E_2 with the Behavior Cloning algorithm Schaal (1996) separately. Then we initialize a stitched policy $f^{E_1}(g_1^{E_1}(o_1^{E_1}), g_2^{E_2}(o_2^{E_2}), o_l^{E_1}, \mathbb{A}_3)$. This stitched policy is tested in the task T in E_3 . Its performance is measured by its success rate.

C Quantitative Analysis of the Module Interface

The cosine and L2 pairwise distance shown in Fig. 8 measures the similarity between two latent representations. For a group of states $\mathbb{S}_{E,T} = \{s_{E,T}^i\}$ in the environment E of task T , they are observed from two cameras and get two groups of observed images $\mathbb{O}_{1,E,T} = \{o_{1,E,T}^i\}$ and $\mathbb{O}_{2,E,T} = \{o_{2,E,T}^i\}$. We obtain the average pairwise distance between the latent representations of two visual encoders by calculating the mean of the pairwise distances across all input states:

$$\bar{d}_{cos} = \frac{1}{|\mathbb{S}_{E,T}|} \sum_{i=1}^{|\mathbb{S}_{E,T}|} (1 - S_C(g_1^E(o_{1,E,T}^i), g_2^E(o_{2,E,T}^i))) / |\mathbb{S}_{E,T}|, \quad (11)$$

$$\bar{d}_{L2} = \frac{1}{|\mathbb{S}_{E,T}|} \sum_{i=1}^{|\mathbb{S}_{E,T}|} d_{L2}(g_1^E(o_{1,E,T}^i), g_2^E(o_{2,E,T}^i)) / |\mathbb{S}_{E,T}|, \quad (12)$$

where g_1^E and g_2^E are the visual encoders for $\mathbb{O}_{1,E,T}$ and $\mathbb{O}_{2,E,T}$ separately, $S_C(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$ is the cosine similarity and $d_{L2}(p, q) = \|p - q\|$ is the L2 distance. Fig. 8 shows the cosine and L2 distances in all the experiments in the Push task. We record the distances data and the mean cosine and L2 distances across all these experiments in Table 8.

D Influence of Using Decoder 1 and Decoder 2

This section aims to answer the question: Is there any difference between choosing which action decoder (action decoder 1 v.s. action decoder 2)?

We carry out the zero-shot transfer in the Stack task with six different visual configurations. For each experiment, we try action decoder 1 and action decoder 2, and report their success rates side-by-side in Table 9. We notice that the average success rates of decoder 1 and decoder 2 across the six visual configurations are close to each other for all the methods, and the maximum difference is within 15%. We believe that this difference is generated by the randomness of the testing process, but not the systematic advantage of one

| | Cosine Distance | | L2 Distance | |
|-----------------|-----------------|----------|--------------|----------|
| | ours | baseline | ours | baseline |
| Masked | 0.065 | 0.422 | 4.943 | 6.323 |
| Zoom in | 0.083 | 0.687 | 4.068 | 4.469 |
| Blurred | 0.043 | 0.982 | 4.937 | 5.828 |
| Gaussian Noise | 0.062 | 1.044 | 3.196 | 6.865 |
| Camera Type | 0.045 | 0.759 | 3.588 | 5.764 |
| Camera Position | 0.003 | 1.072 | 1.936 | 5.314 |
| Mean | 0.050 | 0.828 | 3.778 | 5.761 |

Table 8: **Latent Representations Distances Data.** The distances of the latent representations in all the experiments in the Push task are recorded. We also calculate the mean distances across all these experiments.

| | | Mask | Zoom in | Blurred | Noise | Fisheye | Camera Position | Average |
|--------------------------------------|-----------|-------------------|------------------|-------------------|------------------|------------------|------------------|-------------|
| Devin et al. 2017 | Decoder 1 | 0.7±0.94 | 8.0±1.63 | 0.7±0.94 | 24.0±2.83 | 0.0±0.00 | 14.0±3.27 | 7.9 |
| | Decoder 2 | 5.3±2.49 | 0.0±0.00 | 14.0±2.83 | 6.7±2.49 | 3.3±1.89 | 5.3±2.49 | 5.8 |
| Cannistraci et al. 2024 (linear) | Decoder 1 | 47.3±0.94 | 62.0±4.32 | 32.7±3.77 | 30.7±0.94 | 54.0±8.64 | 14.7±6.18 | 40.2 |
| | Decoder 2 | 39.3±4.11 | 56.7±1.89 | 26.7±6.60 | 51.3±4.11 | 46.0±2.83 | 23.3±1.89 | 40.6 |
| Cannistraci et al. 2024 (non-linear) | Decoder 1 | 10.0±1.63 | 12.0±0.00 | 0.0±0.00 | 3.3±0.94 | 0.0±0.00 | 0.7±0.94 | 4.3 |
| | Decoder 2 | 5.3±0.94 | 6.7±0.94 | 8.0±2.83 | 14.7±1.89 | 2.0±1.63 | 0.0±0.00 | 6.1 |
| PeS (-w/o disent. loss) | Decoder 1 | 34.0±11.43 | 10.7±4.11 | 62.0±10.71 | 34.0±7.12 | 22.7±3.77 | 26.0±4.32 | 31.6 |
| | Decoder 2 | 31.3±2.49 | 52.7±6.60 | 28.0±3.27 | 26.7±1.89 | 32.7±3.40 | 19.3±4.11 | 31.8 |
| PeS (w. l1 & l2 loss) | Decoder 1 | 92.7±0.94 | 98.0±0.00 | 62.7±6.60 | 24.0±4.90 | 59.3±7.36 | 58.7±1.88 | 65.9 |
| | Decoder 2 | 72.7±3.77 | 36.0±1.63 | 88.7±4.99 | 58.7±1.88 | 23.3±3.40 | 33.3±0.94 | 52.1 |
| PeS | Decoder 1 | 94.7±0.94 | 96.7±0.94 | 90.0±1.63 | 96.7±1.89 | 97.3±2.49 | 80.0±4.90 | 92.6 |
| | Decoder 2 | 83.3±4.11 | 86.0±4.32 | 94.0±2.83 | 92.7±0.94 | 95.3±0.94 | 68.7±6.18 | 85.0 |

Table 9: **Action Decoder 1 V.S. Action Decoder 2.** All the experiments are carried out with the Stack task. For the PeS method and other baseline and ablation methods, the difference in the average success rates of using action decoder 1 or action decoder 2 is within 15%. The choice of the action decoder does not have distinct influence on the zero-shot transfer success rates.

decoder to another decoder. This result suggests that the choice of the action decoder during the zero-shot transfer process does not make a distinct difference for all the methods on average.

If we look at the success rates of each visual configuration, we can see that the success rates difference of the PeS method is within 11%. The choice of decoder does not affect the reassembled policy’s performance with PeS, and both decoders can lead to very satisfying success rates over 85%. However, in some experiments with the baselines and the ablation methods (e.g. PeS (w. l1 & l2 loss)-Fisheye), we can see a huge success rate difference. It indicates that the choice of different decoders has a random influence on the performance of the baselines and ablation methods for different visual configurations, but it doesn’t affect the average success rates drastically.

E Anchors from failure trajectories

| | Mask | Zoom in | Blurred | Noise | Fisheye | Camera Position | Average |
|-------------------|------------------|------------------|------------------|------------------|------------------|------------------|-------------|
| 100% success rate | 94.7±0.94 | 96.7±0.94 | 90.0±1.63 | 96.7±1.89 | 97.3±2.49 | 80.0±4.90 | 92.6 |
| 54% success rate | 98.7±0.94 | 95.3±3.77 | 46.0±1.63 | 85.3±0.94 | 52.7±2.49 | 84.0±3.27 | 77.0 |
| 6% success rate | 49.3±6.80 | 48.7±4.11 | 76.7±8.06 | 98.7±0.94 | 44.7±3.40 | 98.0±1.63 | 69.4 |

Table 10: **Anchors from Failure Trajectories.** All the experiments are carried out with the Stack task. We chose anchor datasets from three different datasets with K-means algorithm: (1) The dataset is collected by an expert agent, and it contains 200 success trajectories. (2) The dataset is collected by an semi-trained policy which has around 50% of success rate. It is used to collect 200 trajectories, among which 54% are successful trajectories and 46% are failure trajectories. (3) The dataset is collected by a poorly behaved policy with a close to zero success rate. It is used to collect 200 trajectories, among which only 6% are successful trajectories and 94% are failure trajectories. The average success rate decreases when the percentage of successful trajectories in the dataset for anchor selection decreases.

This section aims to answer the question: Does the data collection process require some success trajectory? Or even failure/exploratory trajectories are still useful?

We use the Stack task to carry out the experiments. We collect a dataset with 200 successful trajectories and collect an anchor set 1 from this fully successful dataset. Then we train a policy to about 50% of training success rates and execute it to collect 200 trajectories during testing. 54% of these trajectories are successful and the rest 46% are failed. We collect an anchor set 2 from this semi-successful dataset. We also train a policy for only 1 minute so that it has a very low success rate. We use this poorly trained policy to collect a dataset of 200 trajectories with only 6% of them being successful. We collect an anchor set 3 from this dataset in which most trajectories fail. All the anchors are collected with the k-means algorithm as shown in Figure 3.

We use these three different anchor sets for training the policies across the six visual configurations. The data set used for training is still the same dataset collected by an expert agent with 200 successful trajectories, and the only difference is that the anchors are selected from data sets with different success rates. Table 10 reports the success rates. On average, the success rate decreases when the proportion of successful trajectories decreases for the anchor selection. In addition, we notice that the performance of the trained policy becomes unstable when the failure trajectory number during anchor selection increases. When there are no failure trajectories, the policy success rates are stably above 80%. In contrast, with the failure trajectories number increases, although there are still some cases where the trained policy can get over 90% of success rates, there appear more cases where the policy gets around 40% to 50% of success rates. These low-performance experiments make the average success rate decrease.

Since we need to use an expert dataset for training the policies, and it is a small data set with only 200 trajectories that are easy to collect, we encourage the users of the PeS method to directly collect the anchor set from the training data set with K-means algorithm. Our experiment result shows that this anchor selection method can lead to more stable zero-shot transfer performance and a higher average success rate.

F Latent Representations Visualization

We visualize the latent representations of the Lift, Can, Stack and Door tasks. Among all the visual configuration changing experiments, we choose the camera position variation experiments. We first reduced the 256D representations to 3D with PCA (Hotelling, 1933) for visualization. Since the side view encoder of the policy 2 is stitched to the policy 1 at the position of its original front view encoder, we compare the latent representations of these two encoders.

In these visualizations, the red data points are the first 5 steps of images in each of the 200 games in the dataset of a certain task. The blue data points are the last 5 steps of images, and the green data points are the 5 steps of images in the middle of each trajectory. Therefore, we have 1000 data points for each color which represent the starting, middle, and ending stages of a task.

The visualization results show that PeS can better align the latent space and force an approximate invariance of the latent representations. In contrast, the Devin et al. (2017) baseline without adopting the relative representation and the disentanglement regularization leads to different latent representations between the two encoders, and they have an approximately isometric transformation relationship. These visualizations support our conclusions in the paper.

G Impact of Anchor Number

To study the impact of number of anchors on transfer performance, we picked a challenging task, Stack, and reported the success rates with standard errors over 3 random seeds on all the different visual configurations in Table 11. We found that when the anchor number is too small, the latent space of the visual encoder does not have enough capacity to capture effective visual representations while maintaining an approximate invariance at the same time. Therefore, the zero-shot transfer performance drops drastically. On the other hand, when the number of anchors becomes too large, there are more redundant anchors which are similar to each other. This will make disentangling the features at the latent space with the disentanglement loss

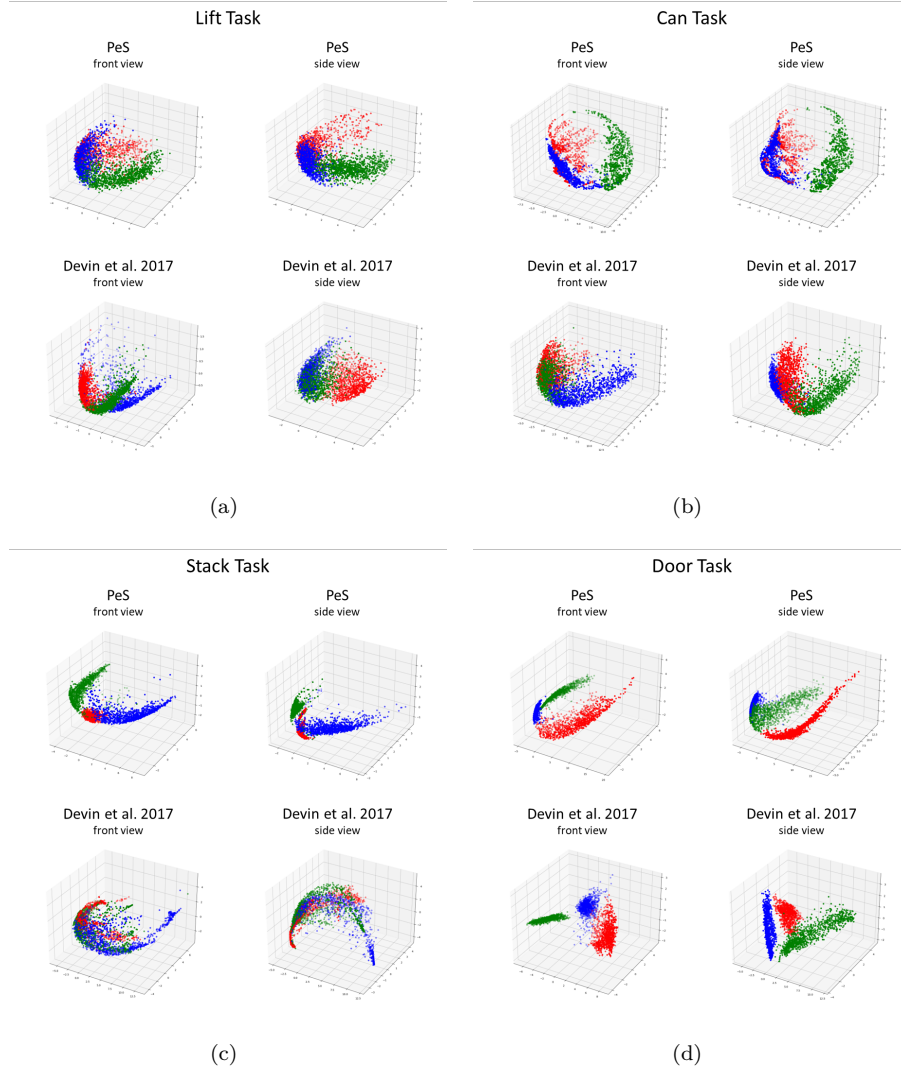


Figure 11: **Latent Representations Visualization.** We visualize the latent representations of the front-view encoder of policy 1 and the side-view encoder of policy 2 of the Lift, Can, Stack and Door tasks with the Camera Position changes in the visual configuration.

harder, which can also lead to some drop of the performance. We empirically found that an anchor number between 256 to 512 can achieve optimal performance in our tasks.

| Anchor Number | Masked | Zoom in | Blurred | Noise | Fisheye | Position | Average |
|---------------|------------------|------------------|------------------|------------------|------------------|------------------|-------------|
| 1024 | 72.7±0.94 | 56.0±4.32 | 88.7±3.40 | 86.7±3.77 | 65.3±2.49 | 69.3±2.49 | 73.1 |
| 512 | 90.7±3.40 | 94.7±0.94 | 91.3±0.94 | 92.7±5.73 | 95.3±3.40 | 88.7±9.84 | 92.2 |
| 256 | 94.7±0.94 | 96.7±0.94 | 90.0±1.63 | 96.7±1.89 | 97.3±2.49 | 80.0±4.90 | 92.6 |
| 128 | 72.7±10.87 | 43.3±5.73 | 42.0±1.63 | 24.0±2.83 | 73.3±1.89 | 30.0±7.12 | 47.6 |
| 64 | 1.3±0.00 | 6.7±0.94 | 6.7±4.11 | 0.0±0.00 | 5.3±2.49 | 0.0±0.00 | 3.3 |

Table 11: Success rates of the Stack task with different anchor numbers

We also investigated the impact of the anchor number on computational efficiency. We picked the Stack task and trained the policies on a NVIDIA A6000 GPU with the batch size of 32. As shown in Table 12, a larger anchor number will lead to a larger model size, longer anchor searching time, and longer training time for each epoch.

| Anchor Number | Model Size | Anchor Searching Time (s) | Training Time Per Epoch (s) |
|---------------|------------|---------------------------|-----------------------------|
| 1024 | 26,835,803 | 1455 | 182 |
| 512 | 25,720,667 | 922 | 107 |
| 256 | 25,163,099 | 570 | 66 |
| 128 | 24,884,315 | 288 | 33 |
| 64 | 24,744,923 | 194 | 23 |

Table 12: network model sizes, anchors searching time, and training time per epoch of the Stack task with different anchor numbers

To sum up, either a too large or too small anchor number can hurt the transfer performance, and larger anchor numbers can reduce the computational efficiency. It is important to select an appropriate anchor number for each task.

H Impact of Disentanglement Loss Weight

| Weights | Masked | Zoom in | Blurred | Noise | Fisheye | Camera Position | Average |
|---------|------------------|------------------|------------------|------------------|------------------|------------------|-------------|
| 0.0 | 34.0±11.43 | 10.7±4.11 | 62.0±10.71 | 34.0±7.12 | 22.7±3.77 | 26.0±4.32 | 31.6 |
| 0.0002 | 34.7±1.89 | 8.0±1.63 | 88.0±2.83 | 30.0±1.63 | 16.0±2.83 | 24.7±2.49 | 33.6 |
| 0.002 | 94.7±0.94 | 96.7±0.94 | 90.0±1.63 | 96.7±1.89 | 97.3±2.49 | 80.0±4.90 | 92.6 |
| 0.02 | 90.7±2.49 | 80.7±10.87 | 92.0±4.90 | 88.0±3.27 | 81.3±5.25 | 70.0±4.32 | 83.8 |
| 0.2 | 35.3±3.40 | 56.0±1.63 | 88.7±2.49 | 31.3±0.05 | 48.0±1.63 | 16.0±4.32 | 45.9 |

Table 13: Success rates of the Stack task with different disentanglement loss weight

We have also studied the effect of different weights of disentanglement loss. We picked the Stack task and tested different weights on all the various visual configurations. When the weight is close to zero, the transfer performance gets close to that of the PeS (w/o disent. loss) ablation method and is significantly lower than the PeS full method. When the weight becomes too large, the optimization process leans too much to disentangling the latent features than imitating the expert behaviors. Therefore, the weaker imitation of the expert agent will also cause the performance drop. We empirically found that the optimal weight of the disentanglement loss in our tasks should be around the range of 0.002 to 0.02. The success rates with standard errors over 3 random seeds are shown in Table 13.

I Impact of Replay Trajectory Deviations

One limitation of PeS is that it requires a trajectory replay to collect the anchor images. However, in real-world applications, it is usually hard to accurately replay the exact trajectories. In order to understand how the trajectory deviation errors in replay could influence the performance of PeS, we conduct additional experiments to introduce trajectory deviations with different amplitudes.

We use the Stack task in the simulation to test the effect of trajectory deviations. During the trajectory replaying, we add a random horizontal vector to every position point on the trajectory except for the gripper close or open action point. The length of the random vector to cause deviation is sampled from a uniform distribution within a certain range, and we test out the range options of 0 to 1 cm, 0 to 3 cm, and 0 to 5 cm. The direction of the deviation vector is randomly sampled within the x-y plane.

| Amplitude | Masked | Zoom in | Blurred | Noise | Fisheye | Position | Lighting | Average |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---------|
| 0 cm | 94.7±0.94 | 96.7±0.94 | 90.0±1.63 | 96.7±1.89 | 97.3±2.49 | 80.0±4.90 | 82.0±1.63 | 91.1 |
| 1 cm | 72.0±5.89 | 86.7±4.71 | 77.3±6.18 | 88.0±1.63 | 93.3±0.94 | 74.7±7.36 | 78.0±4.32 | 81.4 |
| 3 cm | 34.7±3.77 | 35.3±1.89 | 44.7±8.22 | 26.7±1.89 | 37.3±2.49 | 19.3±7.36 | 23.0±2.83 | 31.6 |
| 5 cm | 12.0±2.83 | 14.7±4.99 | 16.7±0.94 | 23.3±3.40 | 16.7±3.40 | 8.3±1.89 | 3.3±0.94 | 13.6 |

Table 14: Success rates of the Stack task with different replay trajectory deviation amplitudes

The experiment results are shown in the table below. When the trajectory deviation is within the range of 1 cm, The average performance of PeS drops by about 10%, but it can still achieve 81.4% of average success rate, which is much higher than all the baselines that don't require trajectory replay and all the ablation methods that require trajectory replay without deviation. When the trajectory deviation goes up to the range of 3cm, PeS can achieve an average success rate of 31.6%, which is on par with the best performing baselines RT1 (29.5%) and Cannistraci et al. 2024 (linear) (34.6%). When the trajectory deviation goes up to a very large range of 5cm, the average success rate of PeS drops to 13.6%.

In summary, although PeS currently requires trajectory replays for anchor selection, it doesn't require very precise replay and can perform well within 1 cm of replay error range.