Appendix

A. Classifier Training

Training objective: The classifier is trained using binary cross-entropy loss:

$$\begin{aligned}\mathcal{J}(\theta) = &\mathop{\mathbb{E}}_{(s,g,l)\sim\mathcal{D}_{l,a}}\left[\log\left(f_\theta(s,g,l)\right)\right] \\ &+ \mathop{\mathbb{E}}_{(s^-,g^-,l^-)\sim\mathcal{D}^-_{l,a}}\left[\log\left(1 - f_\theta(s^-,g^-,l^-)\right)\right].\end{aligned} \tag{2}$$

where $D_{l,a}$ is the language-annotated dataset that consists of trajectory and language task pairs, and N is a function for generating negative examples from the dataset. Given a dataset $D_{l,a}$, N generates negatives from $D_{l,a}$ in the following ways:

1) **Wrong Instruction:** (s, g, l') where l' is sampled from a different transition than s and g.
2) **Wrong Goal Image:** (s, g', l) where g' is sampled from a different transition than s and l.
3) **Reverse Direction**: (g, s, l), where the order of the current image observation and the subgoal image have been switched.

Across all our experiments, we sample 50% of each training batch to be positive examples and 50% of each training batch to be negative examples. Of the negative examples, 40% are "wrong instruction", 40% are "reverse direction", and 20% are "wrong goal image".

Goal sampling: In a given training tuple (s_t, g, l), g is sampled by taking the goal image from the s_{t+k}, where k is a uniformly sampled integer from 16 to 24.

Network architecture and training hyperparameters: The classifier network architecture consists of a ResNet-34 encoder from [11], followed by a two-layer MLP with layers of dimension 256. Separate encoders are used to encode the image observations and the goal images (parameters are not shared between the two). Both of these encoders use FiLM conditioning [92] after each residual block to condition on the language instruction. Classifier networks are trained using a learning rate of 3×10^{-4} and a batch size of 256 for $100,000$ gradient steps. A dropout rate of 0.1 is used.

B. Image Augmentations

During training of low-level policy networks and classifier networks, we apply the following augmentations to the image observations and the goal images, in the following order:

1) Random Resized Crop:
   - scale: $(0.8, 1.0)$
   - ratio:$(0.9, 1.1)$
2) Random Brightness Shift:
   - shift ratio: 0.2
3) Random Contrast:
   - Contrast range: $(0.8, 1.2)$
4) Random Saturation:
   - Saturation range: $(0.8, 1.2)$
5) Random Hue:
   - shift ratio: 0.1

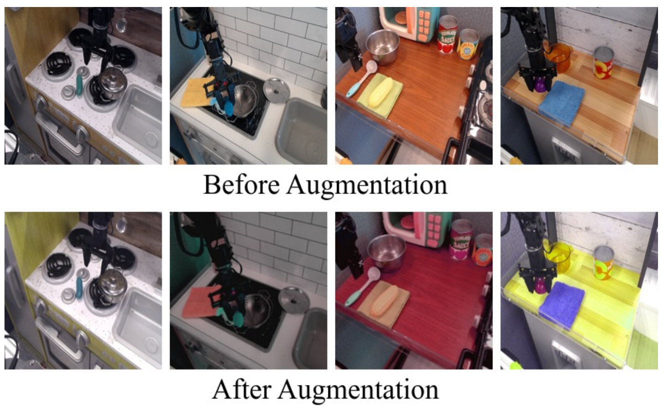


Fig. 4: **Image augmentation examples** Examples of images from the Bridge dataset before and after having the image augmentations applied to them that are used during policy and classifier training.

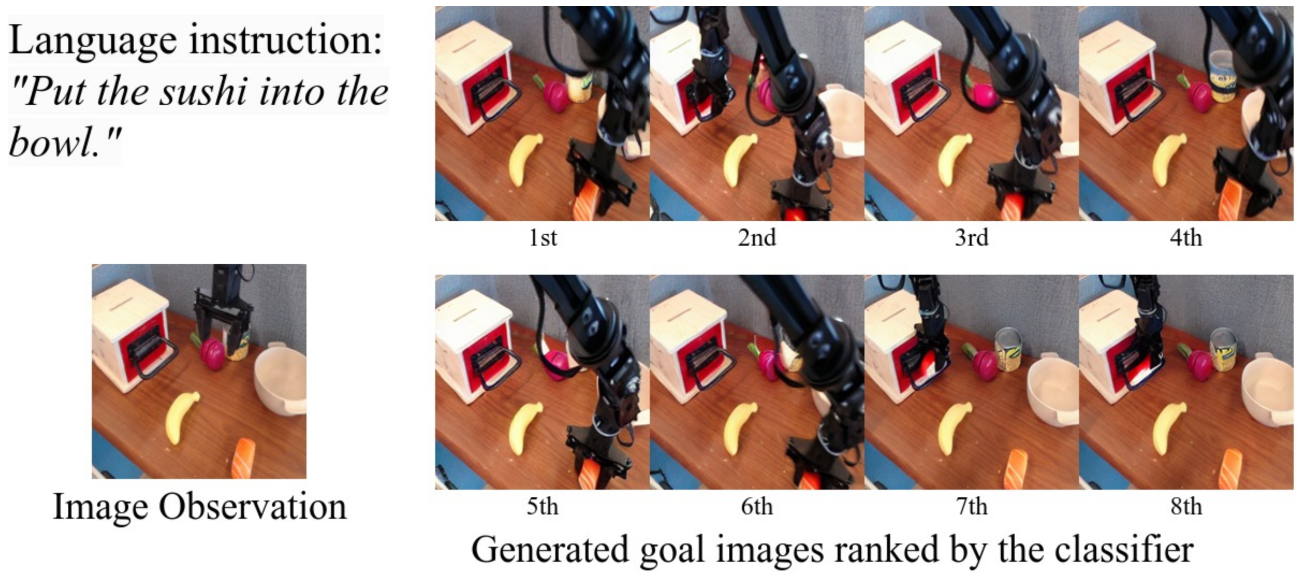
Figure 4 visualizes examples from the Bridge dataset before and after augmentations are applied.

C. Qualitative Analysis

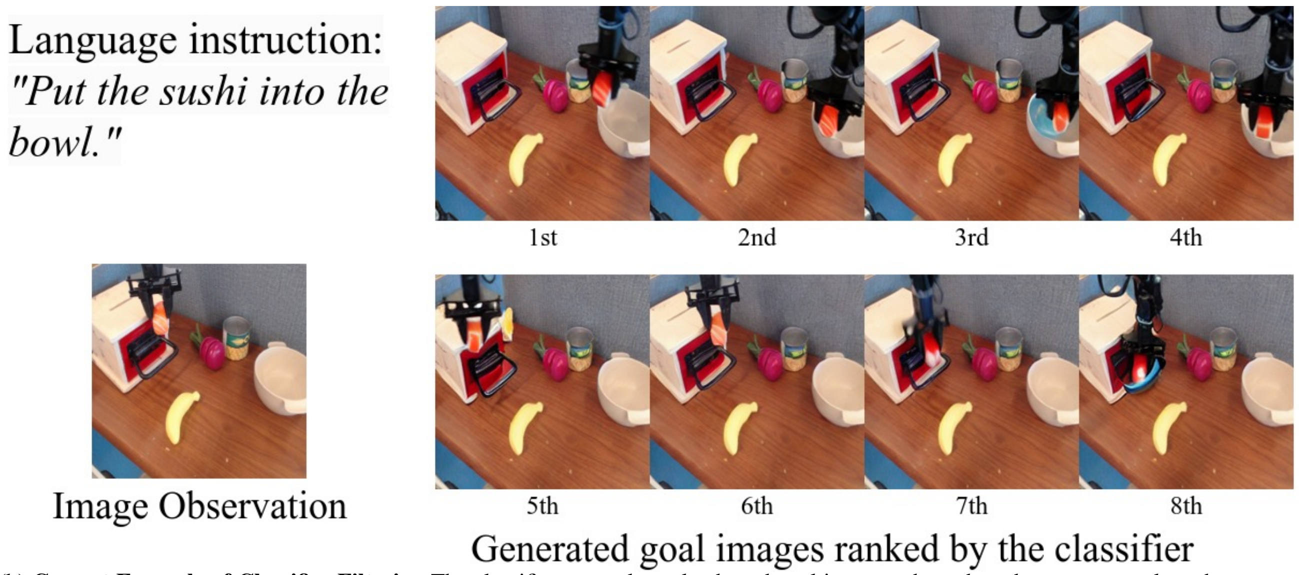
1) Classifier rankings: We show examples of how the classifier network ranks generated goal images on tasks from Scene D of our physical experimental domain. Figures 5a, 5b, 5c show examples of the classifier correctly ranking the generated goal images (highly ranked images correspond to making progress towards correctly completing the language instruction), while Fig. 5d shows an example of the classifier erroneously giving high rankings to goal images that do not make progress towards completing the language instruction. Note that while the classifier scores can be close across various goal images, so long as the relative ranking of the generated goal images is correct, then incorrect subgoal images will be rejected and correct subgoal images will be passed to the low-level policy.

2) Trajectory Visualizations: We show examples of rollouts of GHIL-Glue (SuSIE) on our physical experiment set up. These examples showcase when GHIL-Glue successfully filters out off-task subgoal images (Figure 6a), as well as an instance of when GHIL-Glue nearly causes a failure (Figure 6b).

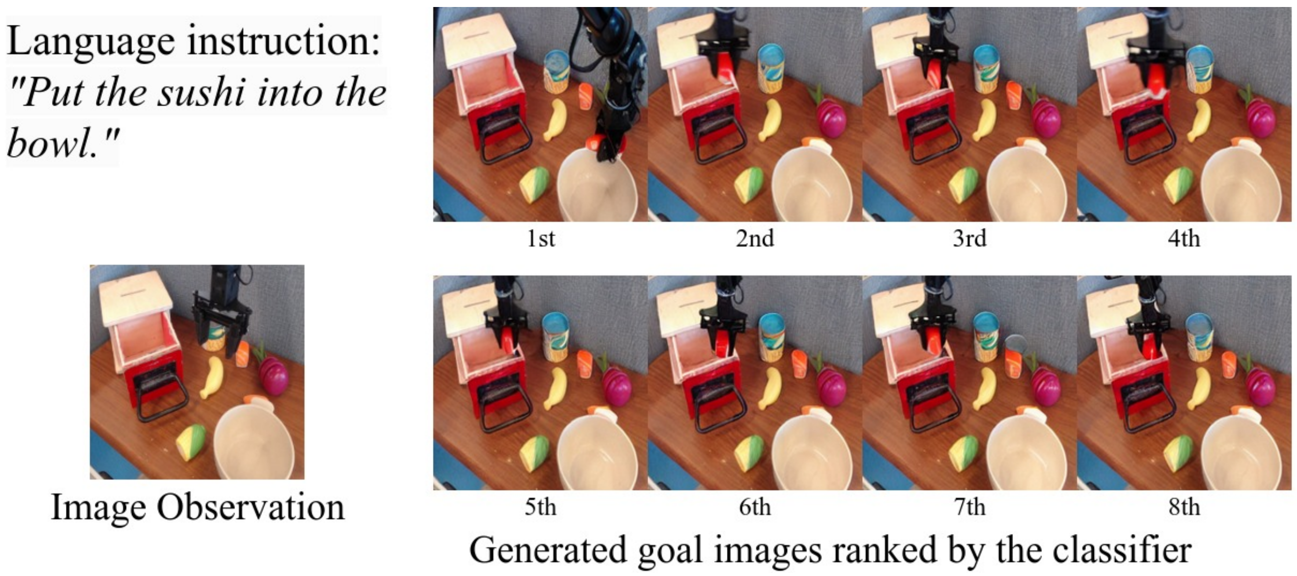
Fig. 5: **Classifier ranking examples** Examples of the classifier network rankings on 8 generated candidate subgoals given an observation from Scene D of the physical experiments and a language instruction. Note that during GHIL-Glue inference, only the first-ranked subgoal is passed to the low-level policy.


Language instruction:
"Put the sushi into the bowl."
Image Observation
1st
2nd
3rd
4th
5th
6th
7th
8th
Generated goal images ranked by the classifier

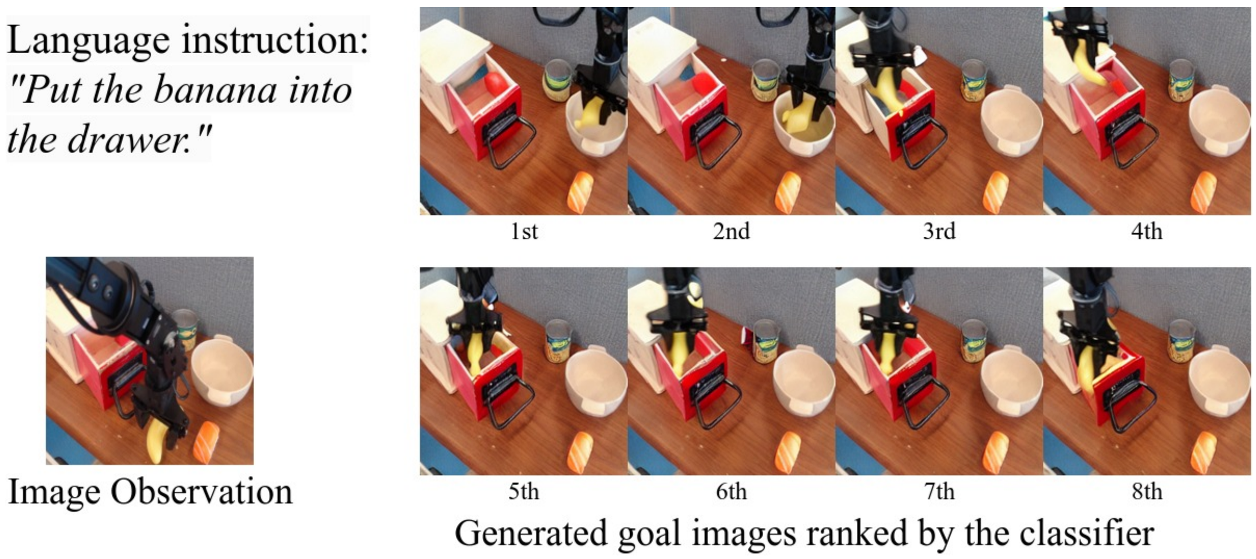
(a) **Correct Example of Classifier Filtering** The classifier correctly ranks the subgoal images where the robot is grasping the sushi higher than the subgoal images where the robot is grasping the drawer handle.


Language instruction:
"Put the sushi into the bowl."
Image Observation
1st
2nd
3rd
4th
5th
6th
7th
8th
Generated goal images ranked by the classifier

(b) **Correct Example of Classifier Filtering** The classifier correctly ranks the subgoal images where the robot moves to place the grasped sushi into the bowl higher than the subgoal images where the robot moves its gripper towards the drawer handle. It ranks the subgoal image with the hallucinated blue bowl-like artifact last.

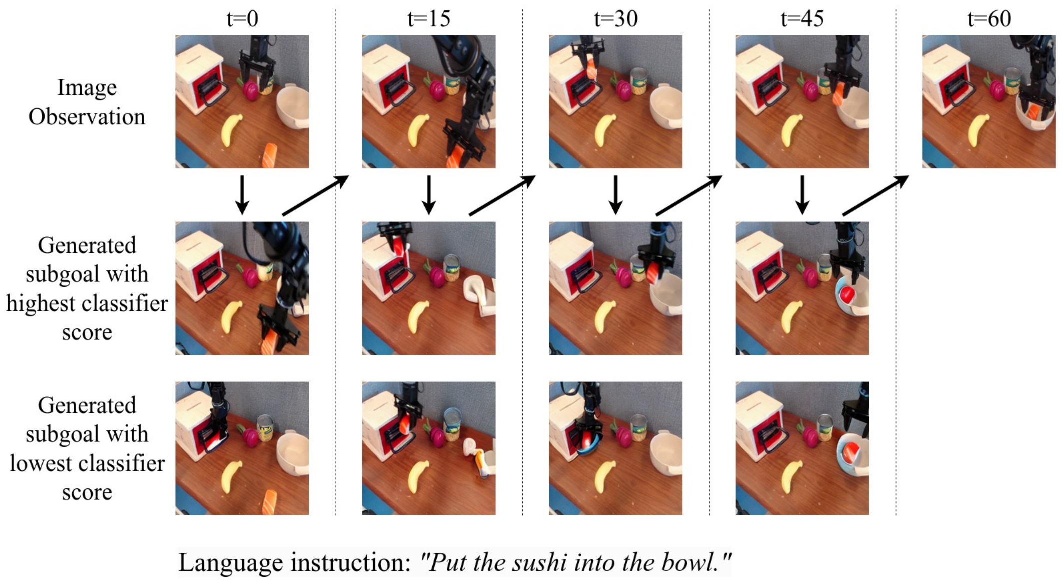

Language instruction:
"Put the sushi into the bowl."
Image Observation
1st
2nd
3rd
4th
5th
6th
7th
8th
Generated goal images ranked by the classifier

(c) **Correct Example of Classifier Filtering** The classifier correctly ranks the subgoal image highest that shows the robot completing the correct task – only a single generated subgoal image shows the robot placing the sushi into the bowl, while all other generated subgoal images show the robot placing the sushi into the drawer.

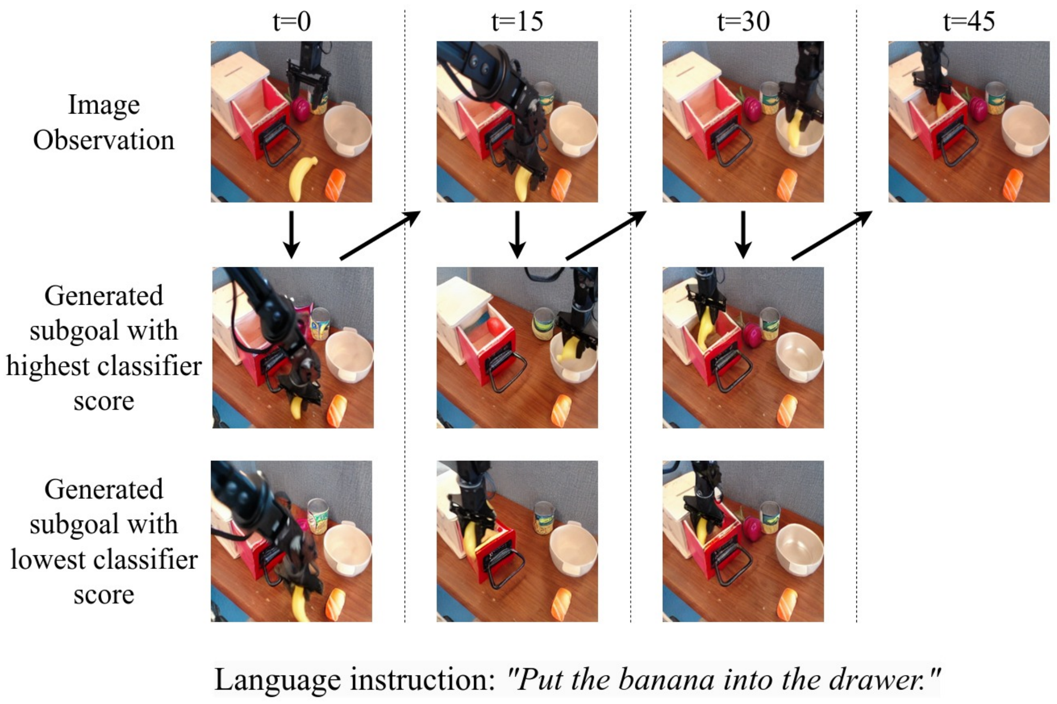

Language instruction:
"Put the banana into the drawer."
Image Observation
1st
2nd
3rd
4th
5th
6th
7th
8th
Generated goal images ranked by the classifier

(d) The classifier incorrectly ranks the subgoal images higher where the robot is placing the banana into the bowl than it ranks the subgoal images where the robot is placing the banana into the drawer. This could be due to there being a strong bias for placing objects in bowls in the Bridge V2 training data.

Fig. 6: **GHIL-Glue (SuSIE) Trajectory Visualization** Visualization of a rollout of GHIL-Glue (SuSIE) on Scene D in the physical experiments set up. The top row shows the current image observation at every timestep at which the video prediction model is queried. The second and third rows show the highest and lowest ranked generated subgoal images out of the 8 generated subgoal images, as ranked by the classifier. Note that during GHIL-Glue inference, only the first-ranked subgoal is passed to the low-level policy.


t=0
t=15
t=30
t=45
t=60
Image Observation
Generated subgoal with highest classifier score
Generated subgoal with lowest classifier score
Language instruction: "Put the sushi into the bowl."

(a) **"Put the sushi into the bowl."** This rollout shows two examples of the classifier filtering preventing the policy from going off-task: at $t = 0$, the lowest ranked generated subgoal shows the gripper grasping the drawer handle instead of moving to grasp the sushi; at $t = 30$, the lowest ranked generated subgoal shows the gripper moving towards the drawer handle instead of towards placing the sushi into the bowl. Note the hallucinated objects and artifacts visible in the goal images at $t = 15, 30, 45$. Augmentation de-synchronization helps to make the low-level policy and classifier robust to hallucinated artifacts such as these.


t=0
t=15
t=30
t=45
Image Observation
Generated subgoal with highest classifier score
Generated subgoal with lowest classifier score
Language instruction: "Put the banana into the drawer."

(b) **"Put the banana into the drawer."** In this rollout, classifier filtering fails and causes a near-miss. At $t = 15$, the classifier ranks a subgoal image highest that shows the robot placing the banana into the bowl instead of the drawer. However, at $t = 30$, when the robot reaches the state specified by this subgoal image, the subsequent generated subgoals all show the robot correctly placing the banana into the drawer. Although, as in this example, the classifier network can occasionally rank incorrect subgoal images higher than correct subgoal images, such errors occur infrequently as GHIL-Glue (SuSIE/UniPI) outperforms base-SuSIE/UniPi across all of our physical and simulated experiments.

3) *Qualitative Analysis of Augmentation De-synchronization:* We see that when applying augmentation de-synchronization, the number of failures due to low-level policy errors (missed grasps, dropping held objects, etc.) decreases, indicating that augmentation de-synchronization is important for the low-level policy to be able to correctly interpret and follow the subgoal images generated by the video prediction model. This is particularly important in domains where there is a large visual generalization gap between the training data and the evaluation tasks. For example, in the CALVIN benchmark, the colors and shapes of objects differ between the training and evaluation scenes. This difference causes the subgoals generated by the video prediction model to often contain objects with incorrect shapes and colors (Figure 7). Augmentation de-synchronization seems to be critical to allowing the low-level policy to be robust to these hallucinations and artifacts.

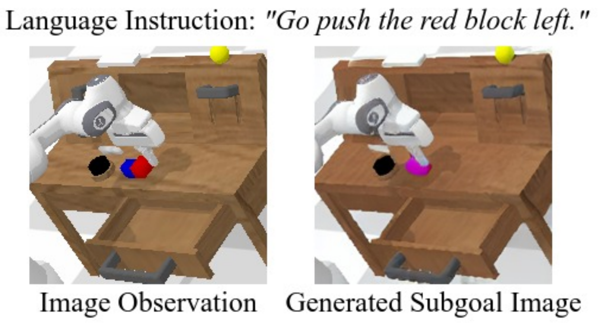


Fig. 7: **Generated Subgoal Image on CALVIN** A subgoal image generated by the SuSIE video model on the unseen environment D of the CALVIN benchmark. The colors and shapes of objects are different in each of the four CALVIN environments, and since the model was not trained on data from environment D, it often generates images with incorrect shapes and colors. Augmentation de-synchronization is important for the low-level policy and classifier to be able to handle these mismatches between image observations and corresponding generated subgoal images.

D. Additional Ablation Experiments

1) *Effect of Augmentation Desynchronization:* We ablate the different components of GHIL-Glue when applied to SuSIE in the CALVIN benchmark (Table III). Removing the augmentation desynchronization from only the low-level policy results in similar performance to base-SuSIE and GHIL-Glue (SuSIE) with the augmentation desynchronization removed from both the low-level policy and the classifier. This suggests that the low-level policy performance of SuSIE without augmentation desynchronization is a significant bottleneck for SuSIE–even when selecting better subgoals via the use of filtering, performance cannot increase if the low-level policy cannot reliably reach those goals. Conversely, removing the augmentation desynchronization from only the classifier results in similar performance to GHIL-Glue (SuSIE) without subgoal filtering. This suggests that, like the low-level policy, augmentation desynchronization is important for the classifier network to correctly perform its function in GHIL-Glue (SuSIE).

Method	Tasks completed in a row					
	1	2	3	4	5	Avg. Len.
SuSIE	89.8%	75.0%	57.5%	41.8%	29.8%	2.94
GHIL-Glue (SuSIE) - Aug De-sync Only	95.2%	84.0%	69.5%	56.0%	46.2%	3.51
GHIL-Glue (SuSIE) - Subgoal Filtering Only	88.5%	75.5%	56.2%	43.0%	32.5%	2.96
GHIL-Glue (SuSIE) - w/o Aug De-sync on policy	91.5%	74.2%	56.0%	41.2%	29.8%	2.93
GHIL-Glue (SuSIE) - w/o Aug De-sync on classifier	95.0%	86.2%	70.0%	57.8%	47.0%	3.56
GHIL-Glue (SuSIE)	**95.2%**	**88.5%**	**73.2%**	**62.5%**	**49.8%**	**3.69**

TABLE III: **Effect of Augmentation Desynchronization in GHIL-Glue (SuSIE)** Success rates on the validation tasks from the D environment of the CALVIN Challenge averaged across 4 random seeds. Results are shown comparing the performance of SuSIE, GHIL-Glue (SuSIE), and ablations of GHIL-Glue (SuSIE). *GHIL-Glue (SuSIE) - Aug De-sync Only* is GHIL-Glue without applying subgoal filtering, *GHIL-Glue (SuSIE) - Subgoal Filtering Only* is GHIL-Glue without applying augmentation de-synchronization to either the low-level policy or the subgoal classifier, *GHIL-Glue (SuSIE) - w/o Aug De-sync on policy* is GHIL-Glue without applying augmentation de-synchronization on the low-level policy, and *GHIL-Glue (SuSIE) - w/o Aug De-sync on classifier* is GHIL-Glue without applying augmentation de-synchronization on the subgoal classifier.

2) *Number of Candidate Subgoals:* We conduct an ablation over the number of candidate subgoals used for subgoal filtering in GHIL-Glue (SuSIE) in the CALVIN benchmark (Table IV). We find that GHIL-Glue (SuSIE) achieves similar performance whether 4, 8, or 16 candidate subgoals are used. In our main results (Section V-C), we report the performance of GHIL-Glue (SuSIE) on the CALVIN benchmark when using 8 candidate subgoals for filtering. For GHIL-Glue (UniPi) on the CALVIN benchmark, we use 4 candidate subgoals for filtering, due to the increased computation burden of generating video subgoals with the UniPi video model vs. generating image subgoals with the SuSIE image model. In our physical experiments, we run GHIL-Glue (SuSIE) using 4 candidate subgoals for filtering.

Method	Tasks completed in a row					
	1	2	3	4	5	Avg. Len.
GHIL-Glue (SuSIE) - 4 samples	95.2%	86.0%	71.2%	60.5%	50.0%	3.63
GHIL-Glue (SuSIE) - 8 samples	**95.2%**	**88.5%**	**73.2%**	**62.5%**	**49.8%**	**3.69**
GHIL-Glue (SuSIE) - 16 samples	95.0%	86.5%	72.8%	60.8%	48.0%	3.63

TABLE IV: **Effect of Number of Candidate Goal Images Sampled in GHIL-Glue (SuSIE)** Success rates on the validation tasks from environment D of the CALVIN Challenge when using GHIL-Glue (SuSIE) when using 4, 8, or 16 candidate goal images with classifier filtering. Results are averaged across 4 random seeds. Results are similar across all numbers of samples, with 8 samples performing the best by a slight margin.