Appendix

A Additional Experiments

A.1 Ablation on prior data

We perform a more fine-grained study on the role of prior data using the CALVIN domain. We compare to variants of our method using 25% or 50% of the available prior data to see whether the quantity of prior data plays a significant role on downstream policy performance. We also compare to a variant of our method that only utilizes prior data collected from different environments than the target task environments. In the context of our CALVIN domain, the prior data spans four environments (A, B, C, D), while the target task data only spans one environment (D). Thus for this ablation we only consider data from unseen environments (A, B, C) for our prior dataset. This ablation examines the robustness of our method under environmental mismatch between the prior data and target task data. We outline all results in Table 3.

Dataset	**No prior data**	**25% prior data**	**50% prior data**	**ABC prior data**	**Ours**
CALVIN-Setting Up	53.3 ± 4.0	71.3 ± 0.5	76.0 ± 2.9	76.7 ± 6.5	$\mathbf{77.3 \pm 3.1}$
CALVIN-Cleaning Up	60.7 ± 3.1	80.3 ± 3.3	82.3 ± 2.1	77.3 ± 10.1	$\mathbf{88.0 \pm 5.1}$

Table 3: **Ablation on prior data.**

First, we see that increasing the size of prior data yields greater downstream policy performance. There is a significant performance gain from using no prior data to using 25% prior data, from which point increasing the amount of prior data leads to smaller gains. In addition, restricting the prior data to unseen environments still results in a meaningful performance increase, which is a promising sign that our method can operate even under controlled environmental distribution shifts between the prior and target task data.

A.2 Ablation on retrieval metric

In this work we use ℓ_2 distance in our latent skill space as the underlying distance measure for our retrieval operation. We also consider performing retrieval based on KL-divergence distances. ie. given two inference distributions q_1 and q_2, we compute their distance as the average forward and reverse KL divergence: $d(q_1, q_2) = \frac{1}{2}(D_{KL}(q_1||q_2) + D_{KL}(q_2||q_1))$. This metric effectively incorporates both the mean and standard deviation of the inference distributions. We compare our standard retrieval procedure with the KL-based retrieval operation in Table 4.

Dataset	**Ours**	**KL-based Retrieval**
CALVIN-Setting Up	77.3 ± 3.1	$\mathbf{79.3 \pm 5.7}$
CALVIN-Cleaning Up	$\mathbf{88.0 \pm 5.1}$	80.7 ± 0.9

Table 4: **Ablation on retrieval method.**

We do not find a significant difference between these two variants, suggesting that our method can work with alternative distance metrics for retrieval.

A.3 Ablation on target dataset

We perform an ablation study on the size of the target dataset. We find for the CALVIN-Setting Up task that increasing the number of target task demonstrations from 30 to 100 yields an increase in success rate from 77.3% to 93.3% (see Table 5). Note that while it is promising that increasing the number of target task demonstrations yields an increase in success rate, this comes at the expense of additional burden for human collecting demonstrations for the target task.

Dataset	30 demos (Ours)	100 demos
CALVIN-Setting Up	77.3 ± 3.1	$\mathbf{93.3 \pm 4.5}$

Table 5: **Ablation on target dataset.**

A.4 Ablation on retrieval dataset

We perform a detailed ablation study on the quantity and quality of the retrieved data for policy learning. Recall that our retrieval procedure: (1) we first randomly sample N sub-trajectories from the prior dataset as possible retrieval candidates, (2) sort them according to their relevance to the target task, and (3) select the top r% of candidates for retrieval. We study the following ablations, which use our standard settings of N=250,000 and r=10% *unless otherwise stated*:

No Retrieval: N=0. Ie. we retrieve no data

All Retrieval: N=sizeof(prior dataset), r=100%. Ie. we retrieve the entire prior dataset. For CALVIN this is 2.3M sub-trajectories

Random Retrieval: instead of sorting the retrieval candidates according to relevance, we randomly select 10% of the candidates. This is a test of data quality, to see whether the relevance of the retrieved sub-trajectories to the target task matters.

2 / 50 / 90 % Retrieval: we retrieve r=2%, 50%, or 90% of the N candidates. This is to test whether our setting of r=10% is a good threshold for retrieval

Large Retrieval: N=sizeof(prior dataset). This ablation uses the same threshold r=10% as our method to perform retrieval but considers all prior sub-trajectories as retrieval candidates and thus retrieves a significantly larger quantity of data.

Note that Ours, No Retrieval, and All Retrieval are from the original submission and we include these results again for reference. We present results on the CALVIN Setting Up and Cleaning

Dataset	Ours	No Retrieval	All Retrieval	Random Retrieval	2% Retrieval	50% Retrieval	90% Retrieval	Large Retrieval
CALVIN-Setting Up	77.3 ± 3.1	65.0 ± 2.2	64.3 ± 10.8	$\mathbf{82.0 \pm 3.7}$	75.7 ± 5.4	74.3 ± 9.9	80.7 ± 6.5	79.7 ± 0.9
CALVIN-Cleaning Up	$\mathbf{88.0 \pm 5.1}$	70.0 ± 0.8	65.3 ± 7.3	55.3 ± 7.3	80.7 ± 2.9	75.7 ± 5.3	46.3 ± 20.0	83.3 ± 4.6

Table 6: **Ablation on retrieval dataset.**

Up tasks in Table 6. We make the following observations for CALVIN-Cleaning Up:

- Data quality is important. The Random Retrieval retrieves the same quantity but lower quality of data as Ours. The performance significantly degrades as a result. We see the same trend from the 50 / 90% Retrieval experiments. Ie. as we increase the threshold for retrieval from r=10% to 50% and 90% (and thus decrease the quality of data) we see a consistent and significant drop in performance.
- Our standard setting of r=10% is optimal, striking the right balance between diversity and quality of data. Lower and higher thresholds (2%, 50%, 90%) perform worse.
- Retrieving larger amounts of data does not have a major impact on performance. Large Retrieval achieves performance within the margin of error as Ours.

CALVIN-Setting Up however offers a different analysis. For this task data quality does not appear to matter, as the Random retrieval, 2% / 50%, 90% Retrieval ablations all perform similarly to Ours within the margin of error. One possible explanation for this observation is that the Setting Up task involves a more diverse range of behaviors than Cleaning Up – the Setting Up task involves manipulating all components of the environment whereas the Cleaning Up task involves a subset. Another potential hypothesis is that the prior data is more biased towards behaviors seen in the Setting Up task. Because many of the behaviors in the Cleaning Up task are mirror behaviors of the Setting Up task, this may result in an unfavorable bias for the Cleaning Up task, necessitating a retrieval procedure to filter out irrelevant behaviors.

The implication of all of these results is that the importance of retrieval may be task and dataset dependent, with some tasks being especially sensitive to the choice of retrieved data.

B Tasks and Datasets

B.1 Franka Kitchen

The Franka Kitchen domain consists of a simulated 9-DoF Franka robot operating in a kitchen environment comprising a microwave, kettle, light switch, stove knobs, and a sliding and hinge cabinet. In our experiments the agent operates the robot via joint torque control resulting in a 9-dimensional action space. For observations, the agent has access to proprioceptive information consisting of the 9-dimensional joint values of the robot, in addition to RGB images from a third-person view camera and an eye-in-hand camera.

Prior Data. This environment is accompanied by approximately 600 human demonstrations each performing a subset of four out of seven possible subtasks: opening the microwave, turning on the light switch, turning on the top burner, turning on the bottom burner, moving the kettle, opening the hinge cabinet, and opening the sliding cabinet. We consider two prior datasets $\mathcal{D}_{\text{prior}}$: (1) using all demonstrations except the ones corresponding to the target task (`Kitchen-All`); and (2) using all demonstrations except those that involve interacting with the microwave (`Kitchen-No Microwave`). These prior datasets have 584 and 235 demonstrations, respectively.

Target Task. We consider one target task demonstrating a specific permutation of subtasks: opening the microwave, followed by moving the kettle, flipping on the light switch, and opening the sliding cabinet. We define task success as whether the agent has completed all of these subtasks (in no particular order). For the target dataset $\mathcal{D}_{\text{target}}$ we obtain all demonstrations in the original dataset that perform this specific permutation of subtasks, resulting in 18 demonstrations. Note that this dataset is equivalent to the `kitchen-complete-v0` dataset in the d4rl benchmark [46]. These demonstrations have an average length of 194 timesteps.

B.2 CALVIN

The CALVIN domain consists of a simulated 7-DoF Franka robot operating in a playroom environment comprising a drawer, cubbies, two lights, and three blocks. The environment comes in four variants (see Figure 3), each with different textures, block sizes, and fixture locations. In our experiments the agent operates the robot via inverse kinematics control resulting in a 7-dimensional action space. For observations, the agent has access to proprioceptive information consisting of the robot end effector pose and gripper state, in addition to RGB images from a third-person view camera and an eye-in-hand camera.

Prior Data. This environment is accompanied by a large dataset of task-agnostic "play" data across all four environment variants and comprises 2.3M transitions. The play data encompass diverse behaviors, such as opening and closing drawers, turning on and off the lights, and picking, placing, and pushing blocks. We use all play data as $\mathcal{D}_{\text{prior}}$ to solve two target tasks.

Target Tasks. We consider two target tasks:

`CALVIN-Setting Up`: the robot must turn on the lights, retrieve the pink block from the drawer, place it on the table, and retrieve the red and blue blocks from the cubby area and place them on the table. We define task success as whether the agent has completed all of these subtasks (in no particular order). At environment resets the lights are always off, the pink block is randomly initialized inside the (closed) drawer, and the red and blue blocks are randomly initialized inside the cubby area with one block in the left region of the cubby and the other block in the right region of the cubby. For this task we collect 30 demonstrations, amounting to about half an hour of data collection. In these demonstrations, we first turn on the lights, then retrieve the pink block, then retrieve the first unoccluded block from the cubby area, then move the slider to retrieve the other block from the other side of the cubby area. These demonstrations have an average length of 584 timesteps.

`CALVIN-Cleaning Up`: the robot must open the drawer, place all three blocks into the drawer, close the drawer, and turn off the lights. We define task success as whether the agent has completed all of these subtasks (in no particular order). At environment resets the lights are always on, the drawer is closed, and the three blocks are randomly placed in left, center, and right regions of the table. For this task we collect 30 demonstrations, amounting to about half an hour of data collection. In these demonstrations, we first open the drawer, then place the blocks on by one into the drawer from right to left, then close the drawer, and finally turn off the lights. These demonstrations have an average length of 572 timesteps.

B.3 Real World Kitchen

We designed a real world kitchen environment to study the utility of our method on physical hardware. Our kitchen environment comprises a Flexa toy kitchen[2], a set of toy food items[3], a number of serving items (placemat, plate, knife, fork), and a small pot and pan that we purchased from a local store. We use a 7-DoF Franka Emika Panda robot which is operated via Operational Space Control (OSC) [49]. We found OSC to be a fitting choice, as it offers task-space compliant behavior that makes for a more intuitive data collection experience. We restrict the OSC controller to the position and yaw of the end effector[4], which combined with the gripper controller results in a 5-dimensional action space. For observations, the agent has access to proprioceptive information consisting of the robot end effector pose and gripper state, in addition to RGB images from a third-person view camera and an eye-in-hand camera.

Prior Data. We collect a large prior dataset of task-agnostic play behaviors involving the food items and the pot and pan. Overall our prior dataset involves 150 trajectories each with approximately 2,000 timesteps, resulting in approximately 300,000 total timesteps. For each trajectory we first initialize the scene by randomly sampling four out of eight food items (milk, bread, butter, sausage, fish, tomato, banana, cheese) and randomly placing these four items around the serving area. We also randomly initialize the pot and pan on the two front stove burners or occasionally place one on the table next to the serving area. We then randomly pick and place food items either on the table, the serving area, or the pot and pan. We also occasionally pick and place the pot or pan to the table or stove burners.

Target Tasks. We consider three target tasks:

`Real-Breakfast`: the objective of this task is to place the bread, butter, and milk from the table onto the serving area. These food items are initialized randomly in the vicinity of three possible locations on the table: the left, center, and right of the region preceding the serving area. We consider two possible permutations for the placement of object onto these three regions (in left-center-right format): butter-bread-milk, bread-butter-milk, and butter-milk-bread. The pots and pans are initialized on the front stove burners. We define task success as whether the robot has (in no particular order) placed the bread onto the plate, the butter to the left of the plate on the placemat, and the milk to the right of the plate on the placemat. For this task we collect 30 demonstrations, amounting to about half an hour of data collection. In these demonstrations we place the bread, butter, and milk in order onto their corresponding goal locations. These demonstrations have an average length of 546 timesteps.

`Real-Cook`: the objective of this task is to place the fish, sausage, and tomato from the table into the pan. These food items are initialized randomly in the vicinity of three possible locations on the table: the left, center, and right of the region preceding the serving area. We consider three possible permutations for the placement of object onto these three regions (in left-center-right format): fish-sausage-tomato, sausage-fish-tomato, and fish-tomato-sausage. The pots and pans are initialized on the front stove burners. We define task success as whether the robot has (in no particular order) placed these three items into the pan. For this task we collect 30 demonstrations, amounting to about half an hour of data collection. In these demonstrations we place the food items from left to right (in order) into the pan. These demonstrations have an average length of 548 timesteps.

`Real-Setup-Pan`: the objective of this task is to place the pan from the table onto the stove and subsequently place the fish and sausage into the pan. The pan is initialized randomly in the vicinity of the right region of the table preceding the serving area. The food items are initialized randomly in the vicinity of the left and center regions of the table preceding the serving area. We consider three possible permutations for the placement of the objects onto these three regions (in left-center-right format): fish-tomato-pan and tomato-fish-pan. The pot is initialized on the front stove burners. We define task success as whether the robot has (in no particular order) placed the pan onto the stove

[2]`https://flexa-usa.com/collections/play/products/toys-the-kitchen`

[3]`https://www.amazon.com/Melissa-Doug-Food-Groups-Hand-Painted/dp/B0000BX8MA`

[4]We did not find the roll and pitch actuation to be necessary for our real world tasks and we opted for a simpler action space.

and the two food items into the pan. For this task we collect 30 demonstrations, amounting to about half an hour of data collection.

C Implementation Details

C.1 Model Architecture

We elaborate further on our method architecture outlined in Figure 2. Our implementation is based on top of robomimic[5], a recent open source codebase with extensive benchmarking results across a number of imitation learning algorithms. We adopted the same neural modules (same RNN backbone, VAE, visual perception encoders) for our algorithm, and in fact our BC-RNN baseline uses the exact implementation from robomimic.

Our model specifically consists of five neural network modules: four networks for the skill model comprising an RNN encoder, an RNN decoder, a feedforward VAE prior[6], and a feedforward temporal prediction network; and one RNN network for the policy.

Observation Encoder. Four out of the five modules described above take observation inputs (among other potential inputs), and each of these modules is equipped with an observation encoder to process these observations. The observation encoder specifically consists of ResNet-18 backbones [50] to encode the third-person image and eye-in-hand image, and a multi-layer perceptron (MLP) for all remaining low-dimensional observational inputs. Note that we pre-process the ResNet inputs with random cropping and post-process the outputs with a Spatial Softmax [51] pooling layer. After processing the image and low-dimensional observation inputs we concatenate the resulting outputs to form one unified observation encoding. Note that our RNN encoder, RNN decoder, and RNN policy process a sequence observations individually using the observation encoder and then processes these encoded observations into one unified representation with a recurrent neural network.

Skill Model. The skill model is a Variational Autoencoder that encodes sub-trajectories into a latent skill representation and decodes information back into the actions of sub-trajectories. The skill encoder and decoder are RNNs with a 2-layer LSTM followed by a 2-layer MLP, while the VAE prior and temporal prediction network are 2-layer MLPs.

Policy. The policy is a 2-layer LSTM network that maps a history of F observations into a latent skill z. We also condition the policy on a dataset id $\in \{0, 1\}$ to indicate whether the policy is optimized on the target dataset or the retrieval dataset, to prevent potential interference between the target and retrieved data (see Algorithm 2 for additional details). Note that we can extend our policy to incorporate fine-grained goal information by conditioning on additional context information such as goal images or language goals [44, 52, 23].

C.2 Training

Our algorithm consists of two phases. In the first phase we pre-train our skill model on sub-trajectories in $\mathcal{D}_{\text{prior}}$ (see Algorithm 1 for further details[7]). In the subsequent phase we are given the target dataset $\mathcal{D}_{\text{target}}$ and we proceed to learning the policy and fine-tuning the skill model. Before we perform policy learning, we first retrieve sub-trajectories in $\mathcal{D}_{\text{prior}}$ that have similar embeddings to those in $\mathcal{D}_{\text{target}}$. We aggregate these retrieved embeddings into our retrieval dataset $\mathcal{D}_{\text{ret}}$. We then proceed to train the policy jointly on embeddings from $\mathcal{D}_{\text{target}}$ and $\mathcal{D}_{\text{ret}}$. At the same time we continue to fine-tune the skill model with sub-trajectories sampled from both $\mathcal{D}_{\text{prior}}$ and $\mathcal{D}_{\text{target}}$. We summarize these steps in Algorithm 2.

We sample fixed-length sub-trajectories uniformly at random to train our model, following recent skill-based imitation learning works [6, 9, 7]. More specifically, for each dataset we concatenate all trajectories into one continuous stream of data and uniformly sample sub-trajectories from this stream. Note that this can result in sampling overlapping sub-trajectories. For training the policy we additionally train on the frame stack of observations preceding the sampled sub-trajectory. There are some edge cases, such as when the sub-trajectory intersects with the next trajectory and when

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

[6]we also utilize a feedforward deterministic inverse dynamics model but we found that it does not lead to a significant change in downstream policy learning results

[7]in our code we also have a slowness term to ensure that two nearby sub-trajectories have similar skill embeddings. We did not find this feature to have a noticeable impact on downstream policy performance and therefore we omit it from the algorithm pseudocode for simplicity.

the frame-stack intersects with the previous trajectory. We deal with these cases by padding all data from the offending consecutive trajectory with the first / last observation of the current trajectory.

Algorithm 1 Skill pretraining

```
Input: prior dataset $\mathcal{D}_{\text{prior}}$
 1: while not done do
 2:     Sample $\tau = \{(o_t, a_t)\}_{t=0}^{H-1} \sim \mathcal{D}_{\text{prior}}$
 3:     Sample temporal offset $d \in [-50, 50]$ and obtain $\tau'$ shifted $d$ timesteps from $\tau$
 4:     $\mu_\tau, \Sigma_\tau \leftarrow q_\phi(\tau)$                    // Encode $\tau$
 5:     $\mu_{\tau'}, \Sigma_{\tau'} \leftarrow q_\phi(\tau')$                  // Encode $\tau'$
 6:     $z \sim \mathcal{N}(\mu_\tau, \Sigma_\tau)$                      // Sample latent z
 7:     $\hat{a}_t \leftarrow p_\psi(o_t, z)$                        // Decode actions through RNN
 8:     $\hat{d} \leftarrow m_\omega(\mu_\tau, \mu_{\tau'})$                   // Predict temporal distance
 9:     $\mathcal{L}_{\text{VAE}} \leftarrow \|\hat{a} - a\|^2 + \beta \cdot D_{KL}(q_\phi(\tau) || p_\theta(o_0, o_H))$  // Compute VAE Loss
10:     $\mathcal{L}_{TC} \leftarrow \alpha \cdot (\hat{d} - d)^2$                   // Compute TC Loss
11:     $\mathcal{L}_{\text{Skill}} \leftarrow \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{TC}}$
12:     update $\phi, \psi, \theta, \omega$ on $\mathcal{L}_{\text{Skill}}$ via gradient descent
13: end while
```

Algorithm 2 Policy learning and skill fine-tuning

```
Input: prior dataset $\mathcal{D}_{\text{prior}}$, target dataset $\mathcal{D}_{\text{target}}$, pre-trained skill encoder $q_\phi$
 1: // Obtain retrieval dataset
 2: $Z_{\text{prior}} \leftarrow \{\mu(q_\phi(\tau_i))\}_{i=1}^{N}, \tau_i \sim \mathcal{D}_{\text{prior}}$        // Sample and encode N sub-trajs from $\mathcal{D}_{\text{prior}}$
 3: $Z_{\text{target}} \leftarrow \{\mu(q_\phi(\tau_j))\}_{j=1}^{M}, \tau_j \sim \mathcal{D}_{\text{target}}$      // Sample and encode M sub-trajs from $\mathcal{D}_{\text{target}}$
 4: $D_{ij} = \|Z^i_{\text{prior}} - Z^j_{\text{prior}}\|_2$               // Compute all pairwise encoding distances
 5: $D\_min_i = \min_j(D_{ij})$  // Find closest target sub-traj for each prior sub-traj
 6: $K \leftarrow$ argsort(D_min)[:n]  // Compute list of indices in $Z_{\text{target}}$ with minimal distance
 7: $\mathcal{D}_{\text{ret}} \leftarrow \{(o^k_{fs}, Z^k_{\text{prior}})\}_{k \in K}$  // Retrieve skill embeddings and their preceding observations
 8:
 9: // Train policy
10: while not done do
11:     Sample $(o_{fs}, z) \sim \mathcal{D}_{\text{target}}$            // target dataset sub-traj encoding and frame stack
12:     Sample $(o'_{fs}, z') \sim \mathcal{D}_{\text{ret}}$              // retrieval dataset sub-traj encoding and frame stack
13:     $\hat{z} \leftarrow \pi(o_{fs}, \text{id} = 0)$                  // predict skill for target dataset sub-traj
14:     $\hat{z'} \leftarrow \pi(o'_{fs}, \text{id} = 1)$                 // predict skill for retrieval dataset sub-traj
15:     $\mathcal{L}_{\text{Policy}} \leftarrow (\hat{z} - z)^2 + \gamma \cdot (\hat{z'} - z')^2$       // Compute Policy Loss
16:     update $\pi$ on $\mathcal{L}_{\text{Policy}}$ via gradient descent
17:     fine-tune $\mathcal{L}_{\text{Skill}}$ on sub-trajectories sampled from $\mathcal{D}_{\text{prior}}$ and $\mathcal{D}_{\text{target}}$  // see Algorithm 1
18: end while
```

C.3 Evaluation

To perform a policy rollout, we first obtain a skill $z = \pi(o'_{fs}, \text{id} = 0)$ from the policy. We execute this skill with our closed-loop skill decoder $p_\psi(o_t, z)$ for H timesteps and we subsequently repeat the process by obtaining a new skill from the policy. Note that we do not preempt skill execution; we execute all H timesetps until completion[8]. We terminate the episode either when the agent has successfully solved the task or if the agent has exceeded the time budget for the rollout. We assess each episode based on whether the agent successfully solved the task in the allotted time budget. While other metric also exist (time to complete task), we chose binary success for its popularity and relative simplicity. We elaborate further on our evaluation protocol:

[8]We believe this is reasonable choice, as (1) the closed-loop skill decoder can react to current environment conditions during skill execution, and (2) the policy is still operating at high frequency and can react accordingly

Simulation Experiments: We evaluate the success rate across 3 seeds (unless otherwise noted) and report the average and standard deviation across all seeds. To evaluate a seed, we perform 100 policy rollouts every n checkpoints and record the success rate for each checkpoint. We then record the success rate for the seed as the highest success rate across all checkpoints evaluated for that seed[9].

Real World Experiments: Due to the challenges of real-world evaluation we only evaluate 1 seed for each baseline. To evaluate an experiment, we perform an initial evaluation of different policy checkpoints, evaluating each checkpoint for only a few trials. Upon choosing the most promising checkpoint we perform 30 rollouts and report the success rate over these rollouts.

C.4 Baselines

All baselines are implemented in the robomimic codebase for fair comparison. We briefly elaborate on these implementations as follows:

BC-RNN: We use the default implementation of BC-RNN in robomimic and we use identical hyperparameters as those reported in the robomimic study paper [1].

BC-RNN (FT): We use an identical architecture and identical hyperparameters as BC-RNN. We first train the baseline on $\mathcal{D}_{\text{prior}}$ and subsequently fine-tune on $\mathcal{D}_{\text{target}}$ via a second stage of training. We also experimented with jointly training a task-conditioned BC-RNN policy in $\mathcal{D}_{\text{prior}}$ and $\mathcal{D}_{\text{target}}$ but we found that it yielded very poor performance due to the multi-modality of actions in the prior data.

BC-RNN (R3M): We use an identical architecture and identical hyperparameters as BC-RNN but with a pretrained R3M visual representation. We specifically replace the weights of our ResNet-18 networks with the pretrained ResNet-18 weights from R3M[10]. We follow the same practice from the R3M paper and we freeze the pretrained ResNet weights during downstream imitation learning.

IQL: We base our implementation off of the publicly available PyTorch implementation of IQL[11].

IQL (UDS): We make small modification to our IQL implementation. For each batch that we sample from $\mathcal{D}_{\text{target}}$, we also sample an equivalent-size batch from $\mathcal{D}_{\text{prior}}$ with the rewards set to 0. We then perform gradient updates on the aggregated data from both of these batches.

FIST: We use the same underlying skill model as our method but a semi-parametric policy in place of our parametric neural network policy. We use an identical scheme for the semi-parametric policy as the FIST paper [7].

C.5 Environment Implementation

C.5.1 Gripper Logic

We elaborate on the gripper logic in our environments. The gripper state is either the position of the gripper fingers (Franka Kitchen, Real World Kitchen) or the opening width of the gripper (CALVIN). The gripper action is a continuous 1-D variable, and we interpret this as either opening (if < 0) or closing (if > 0). The gripper is controlled via position control. When the agent specifies a closing action the position target of the controller is set to close the gripper fingers all the way (and for opening the target is set to open the gripper fingers all the way). There are limits on the force and velocity of the fingers in order to ensure gripper stability.

[9]this is the same evaluation protocol used in [1]
[10]`https://github.com/facebookresearch/r3m`
[11]`https://github.com/rail-berkeley/rlkit/tree/master/examples/iql`

D Hyperparameters

We adopted a similar set of hyperparameters as the BC-RNN baseline from robomimic—we used the same LSTM settings, batch size, and RNN policy history length of $F = 10$. We did experiment with different choices of sub-trajectory lengths for the skill model ($H = \{1, 5, 10, 25\}$) and found that $H = 10$ performs optimally. Longer horizons may be helpful in some settings, however we hypothesize that RNN-based architectures lack the capacity to accurately predict actions over significantly longer horizons. It would be interesting to investigate if the optimal H changes under a Transformer-based [53] architecture.

Hyper-parameter	**Value**
Skill encoder: # LSTM hidden units	1000
Skill encoder: MLP hidden sizes	$1024, 1024$
Skill decoder: # LSTM hidden units	1000
Skill decoder: MLP hidden sizes	$1024, 1024$
Skill prior: hidden sizes	$1024, 1024$
TC: hidden sizes	$128, 128$
Policy: # LSTM hidden units	1000
Sub-trajectory length H	10
Skill latent dimension	64
Skill KL weight β	$1e{-5}$
TC weight α	$1e{-6}$
Retrieval weight γ	0.15 for Real tasks, else 1.0
# observation history frames F	10
Batch size	16
Optimizer	Adam [54]
Learning rate: skill VAE	$5e{-4}$
Learning rate: TP	$1e{-4}$
Learning rate: Policy	$1e{-3}$
Retrieval: max # $\mathcal{D}_{\text{prior}}$ samples N	300,000 for Real tasks, else 250,000
Retrieval: max # $\mathcal{D}_{\text{target}}$ samples M	2,500
% of samples chosen for retrieval	5% for Real tasks, else 10%

Table 7: Hyperparameters for our method.

Method	**# Epochs**	**Evaluation checkpoints freq n**
BC-RNN	800	40
BC-RNN (FT): phase 1	Franka Kitchen: 300 CALVIN: 600 Real tasks: 300	–
BC-RNN (FT): phase 2	Franka Kitchen: 400 CALVIN: 400 Real tasks: 600	20 20 50
BC-RNN (R3M)	800	40
IQL	800	40
IQL (UDS)	800	40
FIST: phase 1	Franka Kitchen: 300 CALVIN: 600	–
FIST: phase 2	200	10
Ours: phase 1	Franka Kitchen: 300 CALVIN: 600 Real tasks: 200	–
Ours: phase 2	200	10

Table 8: Training and Evaluation Hyperparameters.

Task	Image Size	Rollout Length
Franka Kitchen	84×84	280
CALVIN: Setting Up	84×84	1000
CALVIN: Cleaning Up	84×84	1000
Real Breakfast	128×128	1500
Real Cook	128×128	1500

Table 9: Task Hyperparameters.

References

[1] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, 2021.

[2] P. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, 2021.

[3] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, 2020.

[4] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin - a benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[5] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In *Robotics: Science and Systems (RSS)*, 2022.

[6] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

[7] K. Hakhamaneshi, R. Zhao, A. Zhan, P. Abbeel, and M. Laskin. Hierarchical few-shot imitation with skill transition models. In *International Conference on Learning Representations*, 2021.

[8] K. Pertsch, Y. Lee, and J. J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning*, 2020.

[9] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim. Demonstration-guided reinforcement learning with learned skills. In *Conference on Robot Learning*, 2021.

[10] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

[11] J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. In *Conference on Robot Learning*, 2021.

[12] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[13] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems (RSS)*, 2018.

[14] A. Mandlekar, F. Ramos, B. Boots, S. Savarese, L. Fei-Fei, A. Garg, and D. Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[15] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, 2019.

[16] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. In *Conference on Robot Learning*, 2019.

[17] S. Cabi, S. Gómez Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2020.

[18] R. Goyal, S. E. Kahou, V. Michalski, J. Materzyńska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic. The ”something something” video database for learning and evaluating visual common sense. *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[19] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[20] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, M. Martin, T. Nagarajan, I. Radosavovic, S. K. Ramakrishnan, F. Ryan, J. Sharma, M. Wray, M. Xu, E. Z. Xu, C. Zhao, S. Bansal, D. Batra, V. Cartillier, S. Crane, T. Do, M. Doulaty, A. Erapalli, C. Feichtenhofer, A. Fragomeni, Q. Fu, C. Fuegen, A. Gebreselasie, C. Gonzalez, J. Hillis, X. Huang, Y. Huang, W. Jia, W. Khoo, J. Kolar, S. Kottur, A. Kumar, F. Landini, C. Li, Y. Li, Z. Li, K. Mangalam, R. Modhugu, J. Munro, T. Murrell, T. Nishiyasu, W. Price, P. R. Puentes, M. Ramazanova, L. Sari, K. Somasundaram, A. Southerland, Y. Sugano, R. Tao, M. Vo, Y. Wang, X. Wu, T. Yagi, Y. Zhu, P. Arbelaez, D. Crandall, D. Damen, G. M. Farinella, B. Ghanem, V. K. Ithapu, C. V. Jawahar, H. Joo, K. Kitani, H. Li, R. Newcombe, A. Oliva, H. S. Park, J. M. Rehg, Y. Sato, J. Shi, M. Z. Shou, A. Torralba, L. Torresani, M. Yan, and J. Malik. Ego4d: Around the World in 3,000 Hours of Egocentric Video. In *IEEE/CVF Computer Vision and Pattern Recognition (CVPR)*, 2022.

[21] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation, 2022. URL `https://arxiv.org/abs/2203.12601`.

[22] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik. Masked visual pre-training for motor control, 2022. URL `https://arxiv.org/abs/2203.06173`.

[23] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, 2021.

[24] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, 2021.

[25] C. Lynch and P. Sermanet. Language conditioned imitation learning over unstructured data. In *Robotics: Science and Systems (RSS)*, 2021.

[26] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *International Journal of Robotics Research*, 2012.

[27] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[28] S. Krishnan, R. Fox, I. Stoica, and K. Goldberg. DDCO: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on Robot Learning*, 2017.

[29] T. Shankar and A. Gupta. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, 2020.

[30] T. Shankar, S. Tulsiani, L. Pinto, and A. Gupta. Discovering motor programs by recomposing demonstrations. In *International Conference on Learning Representations*, 2020.

[31] T. Kipf, Y. Li, H. Dai, V. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and P. Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, 2019.

[32] D. Tanneberg, K. Ploeger, E. Rueckert, and J. Peters. Skid raw: Skill discovery from raw trajectories. *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[33] K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, 2018.

[34] Z. Su, O. Kroemer, G. E. Loeb, G. S. Sukhatme, and S. Schaal. Learning manipulation graphs from demonstrations using multimodal sensory signals. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[35] Y. Zhu, P. Stone, and Y. Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 2022.

[36] S. Belkhale and D. Sadigh. Plato: Predicting latent affordances through object-centric play, 2022.

[37] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

[38] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[39] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[40] A. Allshire, R. Martín-Martín, C. Lin, S. Manuel, S. Savarese, and A. Garg. Laser: Learning a latent action space for efficient reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[41] M. Yang, S. Levine, and O. Nachum. Trail: Near-optimal imitation learning with suboptimal data. In *International Conference on Learning Representations*, 2022.

[42] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine. Time-contrastive networks: Self-supervised learning from video. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[43] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, 2020.

[44] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning*, 2021.

[45] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

[46] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020. URL `https://arxiv.org/abs/2004.07219`.

[47] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. 2020.

[48] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, C. Finn, and S. Levine. How to leverage unlabeled data in offline reinforcement learning. In *International Conference on Machine Learning*, 2022.

[49] O. Khatib. Inertial properties in robotic manipulation: An object-level framework. *International Journal of Robotics Research*, 1995.

[50] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, 2016.

[51] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[52] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. In *Robotics: Science and Systems (RSS)*, 2020.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[54] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.