A Implementation Details

A.1 Distance function

As mentioned in Section 3, we wish to learn an encoding such that our distance metric d is the euclidean distance between the encoded states.

$$d(s, s') = ||h(s) - h(s')||^2 \tag{5}$$

To learn the encoder h, we optimize a contrastive loss on encodings of the current and future states along the same trajectory. At each update step we have a batch of trajectories $\tau_1, ..., \tau_B$ where $\tau_i = s_{i,1}, ..., s_{i,H}$. For $s_{i,1}$ the positive sample is $s_{i,H}$ and the negative samples are $s_{j,H}$ where $j \neq i$. We use the InfoNCE Loss van den Oord et al. (2018),

$$\mathcal{L}_q = \log \frac{\exp(q^T W k)}{\exp\left(\sum_{i=0}^{K} \exp(q^T W k_i)\right)}, \tag{6}$$

with query $q = h(s_t)$ as the encoded starting state, and the keys $k = h(s_{t+H})$ as the encoded future states along the K trajectories in the dataset $\mathcal{D}$. By doing so, we are attracting the states that are reachable after H steps and pushing those not reachable far out. This approach has been used successfully in prior work in Liu et al. (2020); Emmons et al. (2020) and we use this method for a generalizable solution for finding the state to condition the future on. This particular choice of distance function is a means to an end for obtaining the future state to condition on and further research is required for investigating similar choices for a robust performance across a diverse set of tasks. We leave this investigation for future work.

A.2 Training

The training for both skill extraction and fine-tuning were done on a single NVIDIA 2080Ti GPU. Skill extraction takes approximately 3-4 hours, and fine-tuning requires less than 10 minutes. Our codebase builds upon the SPiRL released code and is located at `https://anonymous.4open.science/r/fist-C5DF/README.md`. Hyperparameters used for training are listed in Table 3.

A.3 Fine-tuning

Fine-tuning for both FIST and SPiRL follows the same implementation details. It is done only on $\mathcal{D}^{\text{demo}}$ which includes 10 trajectories of the agent fulfilling the long horizon task. These trajectories are segmented into sub-trajectories of length $H = 10$ first (similar to pre-training), and then we update all the network parameters[2] by minimizing the loss in equation 3, for 50 epochs of the small dataset. The original training was done on 200 epochs of the large and diverse dataset. Everything else, including batch size, learning rate, and the optimizer remain the same between pre-training and fine-tuning. The hyper-parameters for fine-tuning are listed in Table 4.

[2]During our initial experimental analysis, we tried a version of FIST that only finetuned the inverse skill dynamics model and that did not show any meaningful results. We hypothesise that without fine-tuning the VAE components, the skill-set of the agent will not include the out of distribution parts of the task.

Table 3: Training Hyperparameters

Hyperparameter	Value
Contrastive Distance Metric	
Encoder output dim	32
Encoder Hidden Layers	128
Encoder # Hidden Layers	2
Optimizer	Adam($\beta_1 = 0.9$, $\beta_2 = 0.999$, LR=1e-3)
Skill extraction	
Epochs	200
Batch size	128
Optimizer	Adam($\beta_1 = 0.9$, $\beta_2 = 0.999$, LR=1e-3)
H (sub-trajectory length)	10
β	5e-4 (Kitchen), 1e-2 (Maze)
Skill Encoder	
dim-$\mathcal{Z}$ in VAE	128
hidden dim	128
# LSTM Layers	1
Skill Decoder	
hidden dim	128
# hidden layers	5
Inverse Skill Dynamic Model	
hidden dim	128
# hidden layers	5
Fine-tuning	
Epochs	50
Batch size	128
Optimizer	Adam($\beta_1 = 0.9$, $\beta_2 = 0.999$, LR=1e-3)

Table 4: Fine-tuning hyperparameters

Hyperparameter	Value
Epochs	50
Epoch cycle train	10

B Environment and Dataset Details

In this section we explain the tasks and the procedure for data collection for both pre-training the skills and the downstream fine-tuning in each environment separately. The instruction for downloading the dataset as well as its generation is included in our code repository.

PointMaze. In this environment, the task is to navigate a point mass through a maze, from a start to a goal location. The outline of the maze is shown in Figure 3. We train the skills on three different datasets, each

blocking one side of the maze. To test the method's ability to generalize to unseen long-horizon tasks, we use 10 expert demonstrations that start from random places in maze, but end at a goal within the blocked region. This ensures that our demonstrated trajectories are out of distribution compared to training data. We evaluate the performance by measuring the episode length and the success rate in reaching the demonstrated goals.

The data for PointMaze is collected using the same scripts provided in the D4RL dataset repository Fu et al. (2020). To generate the pre-training dataset we modified the maze outline to have parts of it blocked. Then we run the same oracle way-point controller on the new maze map and collect continuous trajectories of the agent navigating to random goals. For downstream demonstrations we unblock the blocked region, pick a fixed goal within that region, and have the agent navigate from random reset points to the picked goal location. The observation consists of the (x, y) location and velocities.

AntMaze. The task is to control a quadruped ant to run to different parts of the maze. The layout of the maze is similar to PointMaze, and the same sides are blocked off. Similar to PointMaze we measure the episode length and success rate as our evaluation metric.

The data for AntMaze is solely based on the "ant-large-diverse-v0" dataset in D4RL. To construct the pre-training data we filter out sub-trajectories that consist of states within the blocked region. By doing so, we can effectively exclude the region of interest from the pre-training dataset. For the downstream demonstrations we select the trajectories that start from on corner of the environment and end in the blocked region, shown in Figure 3.

Kitchen. The task is to use a 7-DoF robotic arm to manipulate different parts of a kitchen environment in a specific order (e.g. open a microwave door or move the kettle)[3]. During skill extraction we pre-process the offline data to exclude interactions with certain objects in the environment (e.g. we exclude interactions with the kettle). However, for the demonstrations we pick four sub-tasks one of which includes the objects that were excluded from the skill dataset (e.g. if the kettle was excluded, we pick the task to be to open the microwave, move the kettle, turn the top burner, and slide the cabinet door). In evaluation, for completion of each sub-task in the order consistent with the downstream demonstrations, the agent is awarded with a reward of 1.0 for a total max reward of 4.0 per episode.

The pool of demonstrations are downloaded from the repository of Gupta et al. (2019) located at `https://github.com/google-research/relay-policy-learning`. There are a total of 24 multi-task long horizon sets of trajectories that the data is collected from. Each trajectory set is sampled at least 10 times via VR tele-operation procedure and the filenames indicate what the agent is trying to achieve (e.g. microwave-kettle-switch-slide). For creating the pre-training data, we filter the trajectory sets, solely based on the filenames that do not include the keyword for the task of interest. This will essentially remove all multi-task trajectories that include the sub-task and not just the part that we do not want. It is also worth noting that the excluded object (e.g. the kettle) is still part of the environment and all its state vectors are still visible to the agent, despite the exclusion of the "interaction" with that object.

[3]The environment dynamics are still stochastic and therefore, a simple replication of actions would not fulfill the tasks robustly

C EXPERIMENTAL RESULTS

C.1 TABLE OF RESULTS

Table 5: Comparison of our approach to other baselines on the Maze environments. For each experiment we report the average episode length from 10 fixed starting positions with the standard error across 10 evaluation runs (*lower* is better). We also report success rate and its standard deviation. The maximum episode length for PointMaze and AntMaze are 2000 and 1000, respectively.

		FIST (Ours)		**SPiRL**		**BC+FT**	
Blocked Region	Environment	Episode Length	Success Rate	Episode Length	Success Rate	Episode Length	Success Rate
Left	PointMaze	**363.87 ± 18.73**	0.99 ± 0.03	1966.7 ± 32.54	0.02 ± 0.04	1089.76 ± 173.74	0.74 ± 0.11
Right	PointMaze	**571.21 ± 38.82**	0.91 ± 0.07	2000 ± 0	0.0 ± 0.0	1918.99 ± 43.65	0.07 ± 0.06
Bottom	PointMaze	**359.82 ± 3.62**	1.0 ± 0.0	2000 ± 0	0.0 ± 0.0	1127.47 ± 148.24	0.87 ± 0.10
Left	AntMaze	**764.36 ± 8.93**	0.32 ± 0.04	1000 ± 0	0.0 ± 0.0	1000 ± 0	0.0 ± 0.0
Right	AntMaze	**903.98 ± 12.01**	0.22 ± 0.12	1000 ± 0	0.0 ± 0.0	1000 ± 0	0.0 ± 0.0
Bottom	AntMaze	**923.22 ± 6.36**	0.21 ± 0.07	957.85 ± 8.62	0.12 ± 0.07	1000 ± 0	0.0 ± 0.0

Table 6: Comparison of average episode reward for our approach against other baselines on the KitchenRobot environment. The average episode reward (with a max. of 4) along with its standard error is measured across 10 evaluation runs (*higher* is better). Each bolded keyword indicates the task that was excluded during skill data collection.

Task (Unseen)	**Environment**	**FIST (Ours)**	**SPiRL**	**BC+FT**
Microwave, Kettle, **Top Burner**, Light Switch	KitchenRobot	$\mathbf{3.6 \pm 0.16}$	2.1 ± 0.48	0.0 ± 0.0
Microwave, Bottom Burner, Light Switch, Slide Cabinet	KitchenRobot	$\mathbf{2.3 \pm 0.5}$	$\mathbf{2.3 \pm 0.5}$	$\mathbf{2.2 \pm 0.28}$
Microwave, **Kettle**, Slide Cabinet, Hinge Cabinet	KitchenRobot	$\mathbf{3.5 \pm 0.3}$	1.9 ± 0.09	1.3 ± 0.47
Microwave, Kettle, **Slide Cabinet**, Hinge Cabinet	KitchenRobot	$\mathbf{4.0 \pm 0.0}$	3.3 ± 0.38	1.0 ± 0.32

C.2 EXTRA ABLATIONS

Imitation Learning over skills vs. atomic actions. FIST is comprised of two coupled pieces that are both critical for robust performance: the inverse dynamics model over skills and the non-parametric evaluation algorithm. In this experiment we measure the influence of inverse skill dynamics model $q_\psi(z|s_t, s_{t+H-1})$.

An alternative baseline to learning skill dynamics model is to learn an inverse dynamics model on atomic actions $q_\psi(a_t|s_t, s_{t+H-1})$ and perform goal-conditioned behavioral cloning (Goal-BC). This model outputs the first action a_t required for transitioning from s_t to s_{t+H-1} over H steps. We can combine this model with FIST's non-parametric module to determine the s_{t+H-1} to condition on during the evaluation of the policy. As shown in Table 7, temporal abstraction obtained in learning an inverse skill dynamics model is a critical factor in the performance of FIST.

Is our contrastive distance function an optimal approach for picking the future state to condition on? For environments such as PointMaze, where we have access to the same waypoint controller that generates the demonstrations, we can use the ground truth environment dynamics to calculate the oracle state that the waypoint controller would be at, H steps in the future, and use that as the foal. This waypoint controller is not available for the kitchen environment, since the demonstrations were collected using VR tele-operation.

In Table 8, FIST (Oracle) uses the waypoint controller oracle look-up. The results show that with a better future conditioning, an oracle approach can solve the pointmaze task even better than FIST with the contrastive distance. Improving the current semi-parametric approach for obtaining the future conditioning state could be a very interesting direction for future work.

Table 7: We ablate the use of our inverse skill dynamics model by replacing it with an inverse dynamics model on atomic actions. The baseline ablations only succeed on one out of the four tasks. BC learns an inverse dynamics model that takes in state as input and outputs a distribution over atomic actions. Goal-BC uses both state and the goal (sub-task) as input.

Task (Unseen)	FIST (ours)	Goal-BC
Microwave, Kettle, **Top Burner**, Light Switch	$\mathbf{3.6 \pm 0.16}$	0.0 ± 0.0
Microwave, Bottom Burner, Light Switch, Slide Cabinet	$\mathbf{2.3 \pm 0.5}$	1.2 ± 0.3
Microwave, **Kettle**, Slide Cabinet, Hinge Cabinet	$\mathbf{3.5 \pm 0.3}$	1.8 ± 0.44
Microwave, Kettle, **Slide Cabinet**, Hinge Cabinet	$\mathbf{4.0 \pm 0.0}$	0.9 ± 0.1

Table 8: We ablate FIST against an oracle version on pointmaze which uses the ground truth way point planner that has access to the exact state that the agent will end up at H-steps in the future (if it commits to the optimal path).

Section	FIST		FIST (oracle)	
	Episode Length	Success Rate	Episode Length	Success Rate
Left	363.87 ± 18.73	0.99 ± 0.03	236.00 ± 1.02	1.0 ± 0.00
Right	571.21 ± 38.82	0.91 ± 0.07	280.93 ± 5.61	1.0 ± 0.00
Bottom	359.82 ± 3.62	1.0 ± 0.00	269.89 ± 3.75	1.0 ± 0.00

One-shot Imitation Learning: The FIST algorithm can be directly evaluated on one-shot in-distribution downstream tasks without any fine-tuning. In this experiment, we want to see if the agent can pick up the right mode within its skill-set with only one demonstration for fulfilling a long-horizon task in the kitchen environment. The difference between this experiment and our main result is that the down-stream task is within the distribution of its pre-trained skill-set. This is still a challenging task since the agent needs to correctly identify the desired mode of skills.

Our hypothesis is that in SPiRL, the skill prior is only conditioned on the current state and therefore is, by definition, a multi-modal distribution and would require more data to adapt to a specific long-horizon trajectory. For instance, in the kitchen environment, after opening the microwave door, the interaction with any other objects in the environment is a possible choice of skills that can be invoked. However, in FIST, by conditioning the skill prior on the future states, we fit a uni-modal distribution over skills. In principle, there should be no need for fine-tuning for invoking those skills within the distribution of the pre-trained skill set.

We compare our approach to SPiRL (Section 4.2) as a baseline. In addition, we can provide supervision on which skills to invoke to fulfill the long-horizon task by fine-tuning SPiRL (hence *SPiRL-FT*) for a few epochs on the downstream demonstration. As summarized in Table 9, FIST, without any fine-tuning, can fulfill all the long-horizon tasks listed with almost no drift from the expert demonstration. We also see that it is tricky to fine-tune SPiRL in a one-shot setting, as fine-tuning only on one demonstration may cause over-fitting and degradation of performance.

Table 9: With all subtasks seen in the skill dataset, FIST is able to imitate a long-horizon task in the kitchen environment. We compare to a baseline method, SPiRL, which fails to follow the single demo.

Order of tasks (seen in the skill dataset)	FIST (ours)	SPiRL-FT	SPiRL-no-FT
Kettle, Bottom Burner, Slide Cabinet, Hinge Cabinet	$\mathbf{4.0 \pm 0.0}$	0.8 ± 0.19	2.4 ± 0.35
Kettle, Top Burner, Light Switch, Slide Cabinet	$\mathbf{3.8 \pm 0.19}$	0.5 ± 0.16	1.1 ± 0.22
Microwave, Kettle, Slide Cabinet, Hinge Cabinet	$\mathbf{4.0 \pm 0.0}$	1.1 ± 0.22	1.0 ± 0.37
Top Burner, Bottom Burner, Slide Cabinet, Hinge Cabinet	$\mathbf{4.0 \pm 0.0}$	0.1 ± 0.1	0.6 ± 0.25

D BROADER IMPACTS AND LIMITATIONS

Limitations As with all imitation learning methods, the performance of FIST is related to the quality of the provided demonstrations. Concretely, when the skill training demonstrations are poor, we expect the extracted skills to be also sub-optimal, thus, hurting downstream imitation performance. To better understand this limitation, we analyze an extremely noisy versions of the PointMaze dataset and use it for skill extraction. As shown in Table 10, despite achieving a high success rate, the episode length is substantially worse than FIST trained on expert data.

Environment	Episode Length	Success Rate
PointMaze	621.02 ± 69.87	1.0 ± 0.0

Table 10: We evaluate FIST on the maze environment with goal at the bottom when the inverse skill model is trained on an extremely noisy dataset. In this case, FIST achieves sub-optimal performance, or is unable to imitate the test time demonstration.

Learning structured skills from noisy offline data is an exciting direction for future research.

Broader Impacts The ability to extract skills from offline data and adapt them to solve new challenging tasks in few-shot could be impactful in domains where large offline datasets are available but control is challenging and cannot be manually scripted. Examples of such domains include autonomous vehicle navigation, warehouse robotics, digital assistants and perhaps in the future, home robots. However, there are also negative potential consequences. First, since in real-world settings offline data will be collected from users at scale there will likely be privacy concerns, especially for video data collected from users' cars or homes. Additionally, since FIST extracts skills, without labeled data, quality for large datasets becomes increasingly opaque and if there are harmful skills or behavior present in the dataset FIST may extract those and use them during deployment which could have unintended consequences. A promising direction for future work is to include a human in the loop for skill verification.