Appendix

We include the source code at https://bit.ly/3qH2QQK. We structure the Appendix as follows:

A. Additional details about the Skill Transformer method.

B. Further description of the rearrange-easy and rearrange-hard tasks.

C. Details about the demonstration dataset and baselines.

A. Further Method Details

For each training epoch of Skill Transformer, we iterate over 80 randomly selected demonstrations. We step the learning rate schedule after each of these epochs. The learning rate schedule starts at a value of 1×10^{-8} and then in a warmup phase, linearly increases this to a value of 6×10^{-5} over 200 training epochs. The learning rate schedule then uses a cosine decay function that terminates at zero by the end of the training process. The method is trained until convergence, equating to iterating over the entire dataset approximately 6 times in total.

In training Skill Transformer for the rearrange-hard task, we initialize the policy from the pre-trained policy that is trained only on 10,000 easy split episodes to speed up training times. Then, we include 10,000 demonstrations from the entire rearrange-hard dataset to finetune the policy.

B. Further Task Details

The task settings for our experiments match those in the 2022 Habitat Rearrangement Challenge [41]. Our training episodes consist of 60 scene layouts from the ReplicaCAD dataset [40]. We use a subset of the 50,000 episodes from the Rearrangement Challenge as training episodes. Each episode has different object placements in the scene, and the robot is randomly spawned. We use a subset of these episodes to generate the 10,000 demonstrations for the imitation learning methods.

The agent controls the arm via delta joint position targets. The simulation runs PD torque control at 120Hz while the policy acts at 30Hz. The base is controlled through a desired linear and angular velocity. Like in [41], we disabled the sliding behavior of the robot. This means that the robot will not move if its next base movement results in contact with scene obstacles. This modification significantly increases the difficulty of navigation in scenes with densely placed furniture. An object snaps to the robot's suction gripper if the tip of the suction gripper is in contact with the object and the grip action is active.

The Fetch robot is equipped with an RGBD camera on the head, joint proprioceptive sensing, and base egomotion. The task is specified as the starting position of the object and the goal position relative to the robot's start in the scene. From these sensors, the task provides the following inputs:

- 256x256 90-degree FoV RGBD camera on the Fetch Robot head.
- The starting position of the object to rearrange relative to the robot's end-effector in cartesian coordinates.
- The goal position relative to the robot's end-effector in cartesian coordinates.
- The starting position of the object to rearrange relative to the robot's base in polar coordinates.
- The goal position relative to the robot's base in polar coordinates.
- The joint angles in radians of the seven joints on the Fetch arm.
- A binary indicator if the robot is holding an object (1 when holding an object, 0 otherwise). Note this provides information if the robot is holding any object, not just the target object.

The maximum episode horizon is 5,000 timesteps for the rearrange-hard task. We increased the maximum number of timesteps for the task compared to [41] since we found this increased the success rate of all methods.

We evaluate on 20 unseen scene configurations. We evaluate on a total of 400 episodes split into three difficulties as described in Section 3. Specifically, they consist of 100 (25%) *easy* episodes, 200 (25%) of *hard* episodes, and 100 (25%) of *very hard* episodes.

C. Further Experiment Details

C.1. Data Generating Policy Details

As described in Section 5.1, we use M3 (Oracle) as our data-generating policy. We follow the same setup as [16] for training this policy. Specifically, we first train mobile manipulation policies including navigation, pick, place, open drawer, and open fridge policies. Each policy is trained for 100M steps using PPO. We use the same reward functions as from [16]. M3 (Oracle) then composes these skills together using an oracle planning module. Using the ground truth environment state, the oracle planner detects if the object or goal is in a closed receptacle and then adapts the plan accordingly.

We use M3 (Oracle) to generate a dataset of demonstration trajectories for the rearrange-hard task. We run M3 (Oracle) on the Habitat Rearrangement Challenge train dataset [41]. We record the observations, actions, rewards, and which skill is currently executing. The rewards are used for training DT and DT (Skill). The skill labels are used for training ST and DT (Skill).

C.2. Baseline Details

In this section we provide details on the baselines in Section 5.2:

The monolithic RL (Mono) baseline follows the monolithic RL baseline implementation from [40]. Mono inputs

	DT / DT (Skill)	BC-Modular	ST (easy split)	ST (rearrange-hard finetune)
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning Rate	$6e^{-5}$	$6e^{-5}$	$6e^{-5}$	$3e^{-5}$
Warm-Up Epochs	200	100	200	400
Episodes per Epoch	80	80	80	80
Total Epochs	750	300	750	600

Table 3: Hyperparameters for all imitation learning based methods. All methods are trained with the same dataset.

	TP-SRL	M3	Mono
Optimizer	Adam	Adam	Adam
Learning Rate	$3e^{-4}$	$3e^{-4}$	$2.5e^{-4}$
# Steps Per Policy	100M	100M	100M
Value Loss Coef	0.5	0.5	0.5

Table 4: Hyperparameters for all reinforcement learning based methods. All these methods are trained with PPO. For TP-SRL and M3, these hyperparameters are used for training all the skill policies.

the visual representation into an LSTM network and then inputs the recurrent state into an actor and value function head. Mono is trained with PPO for 100M steps across 4 GPUs with 32 environment workers per-GPU, taking 21 hours to train. We fix the parameter count of Mono and all other end-to-end baselines to be the same as Skill Transformer.

For DT and DT (Skill), we follow the details from [8]. Note that both DT and DT (Skill) are conditioned on the desired *return-to-go*, which is the sum of future rewards. We use the *move-object* reward for the entire rearrange-hard task to train these methods. This rewards the progress of the robot to move the target object from its start position to the goal position. DT (Skill) is also trained with an auxiliary classification objective for which skill is currently active. At evaluation time, we condition on the maximum return from the demonstrations. DT and DT (Skill) both require access to the reward function at inference time, whereas Skill Transformer does not.

We train M3 and M3 (Oracle) using the process described in Appendix C.1. For M3, we execute the fixed task plan of navigating to the object, picking up the object, navigating to the goal, and then placing the object at the goal. Note that M3 (Oracle) only adjusts its initial task plan based on the starting environment state. It does not dynamically adjust its task plan. For example, if the agent fails to open the drawer correctly, it won't retry the open skill but instead continues with the pick skill. Like in [40], TP-SRL uses the same training process as M3 but does not include base movement for the pick and place skills. The navigation skill is also rewarded for moving to the closest navigable point to its goal rather than a nearby region as with M3.

The BC-Modular methods train the individual skills with imitation learning using the same demonstration dataset as Skill Transformer. These methods are deployed in the same fashion as M3 and M3 (Oracle).

Hyperparameters for all RL methods are detailed in Table 4 and all imitation learning methods in Table 3. The set of hyperparameters is the same across both tasks unless stated otherwise in the table.

C.3. Ground Truth Skill Labeling Rules

The rules used to define the ground truth skill labels are the following:

- **Navigate** if the agent is not within a threshold of 1.5m from either the object start or the goal location. We used 1.5m to be consistent with M3 (Oracle)'s training setting.
- **Pick** if the agent is within 1.5m from the object start location and either the object is not hidden in a receptacle or the receptacle containing the object is already opened.
- **Place** if the agent is within 1.5m from the object goal location and either the goal is not hidden in a receptacle or the receptacle containing the goal is already opened.
- **Open-Receptacle** if the agent is within 1.5m from either the object start or the goal location and the receptacle containing the object or the goal is not opened.