Skill Transformer: A Monolithic Policy for Mobile Manipulation

Xiaoyu Huang
Georgia Tech

Dhruv Batra
Meta AI (FAIR), Georgia Tech

Akshara Rai
Meta AI (FAIR)

Andrew Szot
Georgia Tech

Abstract

We present Skill Transformer, an approach for solving long-horizon robotic tasks by combining conditional sequence modeling and skill modularity. Conditioned on egocentric and proprioceptive observations of a robot, Skill Transformer is trained end-to-end to predict both a high-level skill (e.g., navigation, picking, placing), and a whole-body low-level action (e.g., base and arm motion), using a transformer architecture and demonstration trajectories that solve the full task. It retains the composability and modularity of the overall task through a skill predictor module while reasoning about low-level actions and avoiding hand-off errors, common in modular approaches. We test Skill Transformer on an embodied rearrangement benchmark and find it performs robust task planning and low-level control in new scenarios, achieving a 2.5x higher success rate than baselines in hard rearrangement problems.

1. Introduction

A long-standing goal of embodied AI is to build agents that can seamlessly operate in a home to accomplish tasks like preparing a meal, tidying up, or loading the dishwasher. To successfully carry out such tasks, an agent must be able to perform a wide range of diverse skills, like navigation and object manipulation, as well as high-level decision-making, all while operating with limited egocentric inputs. Our particular focus is the problem of embodied object rearrangement [41]. An example in Figure 1 shows that the Fetch mobile manipulator must rearrange a bowl from an initial position to a desired goal position (both circled in orange) in an unfamiliar environment without access to any pre-existing maps, 3D models, or precise object locations. Successfully completing this task requires executing both low-level motor commands for accurate arm and base movements, as well as high-level decision-making, such as determining whether a drawer needs to be opened before accessing a specific object.

Long-horizon tasks like object rearrangement require many low-level control steps and naturally break down into distinct phases of skills such as navigation, picking, placing, opening, and closing. Prior works [6, 33, 37] show that it is possible to end-to-end learn a single policy that can perform such diverse tasks and behaviors. Such a monolithic policy directly maps observations to low-level actions, reasoning both about which skill to execute and how to execute it. However, scaling end-to-end learning to long-horizon, multi-phase tasks with thousands of low-level steps and egocentric visual observations remains a challenging research question. Other works decompose the task into individual skills and then separately sequence them to solve the full task [40, 16]. While this decomposition makes the problem tractable, this modularity comes at a cost. Sequencing skills suffers from hand-off errors between skills, where one skill ends in a situation where the next skill is no longer feasible. Moreover, it does not address how to learn the skill sequencing or how to adapt the planned sequence to unknown disturbances.

We present **Skill Transformer**, an end-to-end trainable transformer policy for multi-skill tasks. Skill Transformer predicts both low-level actions and high-level skills, conditioned on a history of high-dimensional egocentric camera observations. By combining conditional sequence modeling with a skill prediction module, Skill Transformer can model long sequences of both high and low-level actions without breaking the problem into disconnected sub-modules. In Figure 1, Skill Transformer learns a policy that can perform diverse motor skills, as well as high-level reasoning over which motor skills to execute. We train Skill Transformer autoregressively with trajectories that solve the full rearrangement task. Skill Transformer is then tested in unseen settings that require a variable number of high-level and low-level actions, overcoming unknown perturbations.

Skill Transformer achieves state-of-the-art performance in rearrangement tasks requiring variable-length high-level planning and delicate low-level actions, such as rearranging an object that can be in an open or closed receptacle, in the Habitat environment [41]. It achieves 1.45x higher success rates in the full task and up to 2.5x higher in especially challenging situations over modular and end-to-end baselines

Figure 1: A Fetch robot is placed in an unseen house and must rearrange an object from a specified start to a desired goal location from egocentric visual observations. The robot infers the drawer is closed and opens the drawer to access the object. Next, it picks the object, navigates to the goal, and places the object. The Skill Transformer is a monolithic "pixels-to-actions" policy for such long-horizon mobile manipulation problems which infers the active skill and the low-level continuous control.

and is more robust to environment disturbances, such as a drawer snapping shut after opening it, despite never seeing such disturbances during training. Furthermore, we demonstrate the importance of the design decisions in Skill Transformer and its ability to scale with larger training dataset sizes. The code is available at https://bit.ly/3qH2QQK.

2. Related Work

Object Rearrangement: Object rearrangement can be formulated as a task and motion planning [15] problem where an agent must move objects to a desired configuration, potentially under some constraints. We refer to [4] for a comprehensive survey. Prior works have also used end-to-end RL training for learning rearrangement [44, 32, 14, 10], but unlike our work, they rely on abstract interactions with the environment, no physics, or simplified action spaces. Other works focus on tabletop rearrangement [28, 21, 12] while our work focuses on longer horizon mobile manipulation. Although mobile manipulation has witnessed significant achievements in robotics learning [29, 43, 26, 30, 19], due to the vastly increased complexity, in rearrangement tasks, a majority of previous works [6, 40] explicitly separate the base and arm movements to simplify the task. A few works [16, 47, 45] have shown the superiority of mobile manipulation over static manipulation. In the Habitat Rearrangement Challenge [40, 41], methods that decompose the rearrangement task into skills are the most successful [16, 45] but they require a fixed high-level plan and cannot dynamically select skills based on observations during execution.

Transformers for Control: Prior work uses a transformer policy architecture to approach RL as a sequence modeling problem [8, 49, 20, 7]. Our work also uses a transformer-based policy, but with a modified architecture that predicts high-level skills along with low-level actions. Other works demonstrate that transformer-based policies are competent multi-task learners [23, 48, 38, 6, 33]. Likewise, we examine how a transformer-based policy can learn multiple skills and compose them to solve long-horizon rearrangement tasks. [27, 5] further show that transformers can encode underlying dynamics of the environment that benefits learning for individual skills.

Skill Sequencing: Other works investigate how to sequence together skills to accomplish a long horizon task. By utilizing skills, agents can reason over longer time horizons by operating in the lifted action space of skills rather than over low-level actions [39]. Skills may be either learned from demonstrations [13, 2, 17], from trial and error learning [36, 11], or hand defined [25]. This work demonstrates that a single transformer policy can learn multiple skills. However, when sequencing together skills, there may be "hand-off" issues when skills fail to transition smoothly. Prior works tackle this issue by learning to match the initiation and termination sets of skills [24, 9] or learning new skills to handle the chaining [3, 22]. Decision Diffuser [1] learns a generative model to combine skills, but requires specifying the skill combination order. We overcome these issues by distilling all skills into a single policy and training this monolithic policy end-to-end to solve the overall task while maintaining skill modularity through a skill-prediction module.

3. Task

We focus on the problem of embodied object rearrangement, where a robot is spawned in an unseen house and must rearrange an object from a specified start position to a desired goal position entirely from on-board sensing and without any pre-built maps, object 3D models, or other privileged information. We follow the setup of the 2022 Habitat Rearrangement Challenge [41]. Specifically, a simulated Fetch robot [34] senses the world through a 256×256 head-mounted RGB-D camera, robot joint positions, whether the robot is holding an object or not, and base egomotion giving the relative position of the robot since the start of the episode. The 3D position of the object to rearrange, along with the 3D position of where to rearrange the object to is also provided to the agent.

The agent interacts with the world via a mobile base, 7DoF arm, and suction grip attached to the arm. The episode is successful if the agent calls an episode termination action when the object is within 15cm of its target. The agent fails if it excessively collides with the environment or fails to rearrange the object within the prescribed low-level time steps. See Appendix B for more task details.

We evaluate our method in the **Rearrange-Hard Task**, which presents challenges in both low-level actions and high-level reasoning. Unlike the Habitat Rearrangement tasks [16, 40] that prior works focus on, this task involves objects and goals that may spawn in closed receptacles. Therefore, the agent may need to perform intermediate actions, such as opening a drawer or fridge to access the target object or goal. Furthermore, the correct sequence of skills required to solve the task differs across episodes and is unavailable to the agent. As a result, the agent must infer the correct behaviors from egocentric visual observations and proprioception. This more natural setting better reflects real-world rearrangement scenarios, thus making it the focus of this work.

From the dataset provided in [41], we create a harder version of the full Rearrange track dataset for evaluation. In this harder version, more episodes have objects and goals contained in closed receptacles. Specifically, the updated episode distribution in evaluation for the rearrange-hard task is divided into three splits:
- 25% *easy* scenes with objects initially accessible.
- 50% *hard* scenes with objects in closed receptacles.
- 25% *very hard* scenes with both objects and goals in closed receptacles.

The original rearrange-hard dataset is used for training.

4. Method

We propose *Skill Transformer*, an approach for learning long-horizon, complex tasks that require sequencing multiple distinct skills. In Section 4.1, we describe the components of Skill Transformer, which include a Skill Inference module for reasoning over skill selections and an Action Inference module to predict actions for controlling the robot, both modeled using a Transformer architecture and conditioned on past and current observations. Next, in Section 4.2, we detail how we train Skill Transformer in an autoregressive fashion from demonstrations of the full task.

4.1. Skill Transformer Architecture

Skill Transformer breaks the long-horizon task of object rearrangement into first inferring skill selections using a Skill Inference module and then predicting low-level control using an Action Inference module, given this skill choice. Both modules are implemented as causal transformer neural networks [31], conditioned on past and current observations of the robot.

Input: Both modules in Skill Transformer take as input a sequence of tokens that encode robot observations. We refer to the maximum length of the input sequence as the context length C. Let o_t denote the $256 \times 256 \times 1$ depth visual observations, and p_t denote the proprioceptive inputs of the joint angles and holding state at time t. The visual observation o_t is first encoded with a Resnet-18 encoder [18] to a 256-dimension visual embedding. Likewise, non-visual inputs are encoded with a multi-layer perceptron network into a 256-dimension sensor embedding. The visual and non-visual continuous embeddings are concatenated to form a 512-dimension input token x_t. We also add a per-time step positional embedding to each input token x_t to enable modeling time across the sequence. x_t forms the input token that encodes the current observations, and Skill Transformer takes as input a history of such tokens up to context length C. Note that we do not use RGB input as prior work [40] found it unnecessary for geometric object rearrangement, and training with only depth input is faster.

Skill Inference: The first module of Skill Transformer is the Skill Inference module which predicts the current skill to be executed. The Skill Inference module takes as input a sequence of the input embeddings $\{x_{t-C}, \ldots, x_t\}$ through a causal transformer network [42] with a 512 hidden dimension, to produce a one-hot skill vector c_t corresponding to the predicted skill that should be active at the current time step t. Using the causal mask, label c_i only has access to the previous tokens $\{t : t < i\}$ in the input sequence. Then, we embed c_i into a latent skill embedding $\{z_{t-C}, \ldots, z_t\}$.

Action Inference: Next, the Action Inference module outputs a distribution over robot low-level actions. The Action Inference module takes as input the input embedding x_t, an embedding of the previous time step action a_{t-1}, and the skill embedding z_t. Each of these inputs are represented as distinct tokens in the input sequence of $\{x_{t-C}, a_{t-C-1}, z_{t-C}, \ldots, x_t, a_{t-1}, z_t\}$. This sequence is then fed into a transformer with the same architecture as the
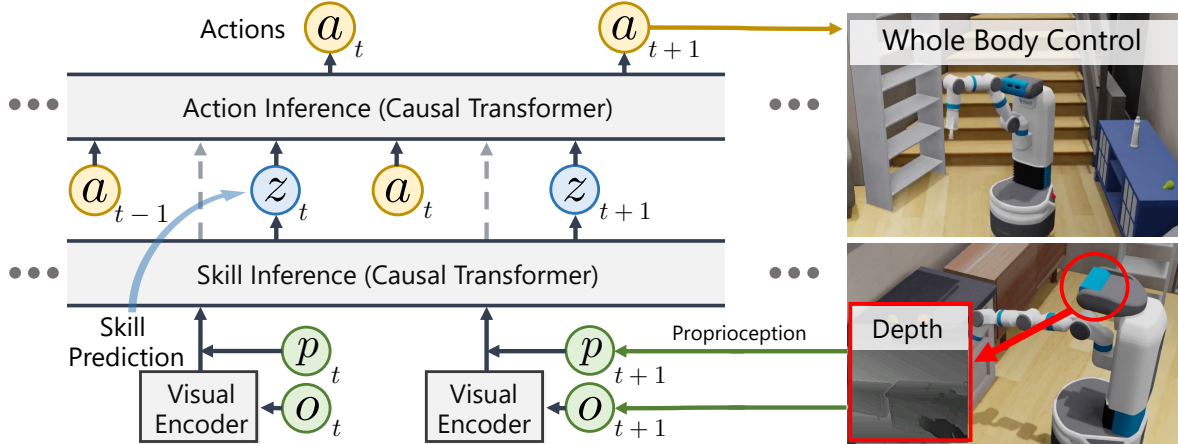
Figure 2: The Skill Transformer architecture. Skill Transformer learns to map egocentric depth images from the Fetch robot's head camera and proprioceptive state to actions controlling the robot, including the base movement, arm movement, and gripper state. First, the raw visual observations are processed by a visual encoder and concatenated with the proprioceptive state. Next, the Skill Inference module, implemented as a causal transformer sequence model, infers the skill ID that is executing for each time step. Finally, the Action Inference module predicts the low-level action to execute on the robot from the inferred skill, observation, and previous action information. The model is trained through end-to-end imitation learning.

Skill Inference module. By taking as input the skill embeddings z_t, the Action Inference module learns consistency in executing individual skills and the transitions between them. As we will show in Section 5.5, this also allows training Skill Transformer with datasets that only successfully complete a portion of the skills.

The Action Inference module then outputs a sequence of predicted next actions $\{a_{t-C}, \ldots a_t\}$ which control the robot. These actions determine the whole body control actions composed of three components: arm joint positions, base velocities, and gripper state. The module first outputs a 12-dimension output vector. The first seven components are used as the mean of a normal distribution to sample the desired delta joint positions, which will then be actuated via a PD controller. The next two dimensions are used as the mean of a normal distribution to sample the desired base velocities, scaled by the maximum linear and angular velocities of the robot base. The remaining three outputs are passed through a softmax layer to generate probabilities for the gripper action, which indicate opening, closing, or maintaining the previous gripper state.

Unless otherwise specified, we use 2 transformer layers and a context length $C = 30$ for all architectures, giving 2M parameters for the transformer network in the Skill Inference module. The Action Inference module is also trained with the same architecture and a context length C of 30 time steps, translating to a sequence length of 90 tokens. In total, between the observation encoders, Skill Inference, and Action Inference modules, the Skill Transformer policy network consists of 16M parameters.

4.2. Skill Transformer Training

In this section, we describe how Skill Transformer is trained end-to-end to learn long-horizon rearrangement tasks from imitation learning. We train the policy using supervised learning on demonstration trajectories produced by a data-generating policy. We assume the data-generating policy demonstrates a composition of skills to solve the task. Such compositional approaches are the state-of-the-art for the Habitat Rearrangement task [40, 16].

Specifically, we assume the data-generating policy demonstrates one of N skills, and each time step t is labeled with the skill ID y_t^{skill} that is currently executing. In our problem setting, these skills are: pick, place, navigate, open drawer, and open fridge. We also record the policy's actions at time step t that consists of arm action y_t^{a}, base velocity y_t^{b}, and gripper state y_t^{g}. In addition, we acquire from the simulator a one-dimensional indicator label y_t^{pred} defined as $y_t^{\text{pred}} = 0$ if the object does not start in a closed receptacle and $y_t^{\text{pred}} = i$ if i^{th} receptacle contains the object, at the time step t.

4.2.1 Training Loss

We separately supervise the Skill Inference and Action Inference modules. We supervise the Skill Inference module over the context length C using loss function:

$$\mathcal{L}_{\text{skill}} = \sum_{t=1}^{C} \mathcal{L}_{\text{FL}}(c_t, y_t^{\text{skill}}) \qquad (1)$$

where $\mathcal{L}_{\text{FL}}$ represents the focal loss. The focal loss counteracts the imbalance of skill execution frequencies. For example, the navigation skill executes more frequently than

the other skills as it is needed multiple times within a single episode, and takes more low-level time steps to execute than manipulation skills.

In order to train the Action Inference module in parallel with the Skill Inference module, we use teacher-forcing, and input the ground truth skill labels during training, instead of the inferred distributions that are used at test time. We supervise the Action Inference module over the context length C with loss function:

$$\mathcal{L}_{\text{act}} = \sum_{t=1}^{C} (\mathcal{L}_{\text{MSE}}(a_t^{\text{b}}, y_t^{\text{b}}) + \mathcal{L}_{\text{MSE}}(a_t^{\text{a}}, y_t^{\text{a}}) + \mathcal{L}_{\text{FL}}(a_t^{\text{g}}, y_t^{\text{g}})) \quad (2)$$

where $\mathcal{L}_{\text{MSE}}$ represents the mean-squared error loss.

In addition, we introduce auxiliary heads and losses for the Skill Inference module. The auxiliary head reconstructs oracle information unavailable to the policy. Specifically, assuming there exist M receptacles that may contain the target object, the auxiliary head outputs an $M + 1$ dimensional distribution v_t. We then supervise v_t with y_t^{pred} over the context length C via the loss function:

$$\mathcal{L}_{\text{aux}} = \sum_{t=1}^{C} \mathcal{L}_{\text{CE}}(v_t, y_t^{\text{pred}}) \quad (3)$$

where $\mathcal{L}_{\text{CE}}$ represents the Cross-Entropy loss.

We incorporate this auxiliary loss because the data-generating policy used to generate the demonstrations has a low success rate in correctly opening the receptacles. This low success rate can hinder learning because Skill Transformer needs to reason if it failed due to not opening the receptacle at all or opening an incorrect receptacle. The auxiliary loss incentivizes Skill Transformer to identify the key features that lead to the decision to open specific receptacles and to avoid wrong ones. This auxiliary objective with privileged information is only used for training and not used at test time.

The final loss is the unweighted sum of individual loss terms: $\mathcal{L} = \mathcal{L}_{\text{skill}} + \mathcal{L}_{\text{action}} + \mathcal{L}_{\text{aux}}$. Note that in training, we calculate the loss over all time steps, whereas at test time, we only output action a_t for the current time step t to the environment. While training, we also train the visual encoder, in contrast to previous works using transformers in embodied AI that only demonstrate performance with pre-trained visual encoders [8, 6]. Since Skill Transformer is an end-to-end trainable policy, it can also be trained with offline-RL using the rewards from the dataset or with online-RL by interacting with the simulator. We leave exploring Skill Transformer with RL training for future work.

4.2.2 Training Scheme

We use a similar scheme as [8], where we input a sequence of 30 timesteps and compute the loss for the skill and action distributions for all timesteps in the sequence using autoregressive masking. Skill Transformer uses a learning warm-up phase where the learning rate is linearly increased at the start of training. After the warm-up phase, Skill Transformer uses a cosine learning rate scheduler which decays to 0 at the end of training. The entire dataset was iterated through approximately 6 times in total during the complete training process. We report the performance of the checkpoint with the highest success rate on seen scenes. We provide further details about Skill Transformer in Appendix A.

5. Experiments

In this section, we describe a series of experiments used to validate our proposed approach. First, we introduce the experiment setup and state-of-the-art baselines for comparison. Then, we quantify the performance of Skill Transformer in the rearrange-hard task and compare it to the baselines. Furthermore, the robustness of the methods to disturbances at evaluation time are evaluated by adversarial perturbations, like closing drawers after the agent has opened them. Finally, we perform an ablation study to confirm the key design choices involved in Skill Transformer.

5.1. Experiment Setup

We conduct all experiments in the Habitat 2.0 [40] simulation environment and the rearrangement tasks as described in Section 3. For training, we use the *train* dataset split from the challenge consisting of 50,000 object configurations across 63 scenes. Each episode consists of starting object positions and a target location for the object to be rearranged. The robot positions are randomly generated in each episode. We first train individual skills (pick, place, navigate, open, close) that are used by a data generating policy to generate 10,000 demonstrations for training Skill Transformer. To train skills, we use the Multi-skill Mobile Manipulation (M3) [16] method with an oracle task planner with privileged information about the environment. This approach trains skills with RL and sequences them with an oracle planner. Further details of the data generating policy and the demonstration collected are detailed in Appendix C.1.

5.2. Baselines

We compare Skill Transformer (ST) to the following baselines in the rearrange-hard task. The parameter count of each baseline policy is adjusted to match the 16M parameters in Skill Transformer.

- Monolithic RL (**Mono**): Train a monolithic neural network directly mapping sensor inputs to actions with end-to-end RL using DD-PPO [46], a distributed form of PPO [35]. The monolithic RL policy is trained for 100M timesteps, reflecting a wall time of 21h.
- Decision Transformer (**DT**) [8]: Train the monolithic policy from the offline dataset using a decision transformer architecture. The DT policy is trained with the

same learning rate scheme and number of epochs as Skill Transformer. At test time, we condition the DT policy with the highest possible reward in the dataset.

- Decision Transformer with Auxiliary Skill Loss (**DT (Skill)**): Train the Decision Transformer with an auxiliary loss of skill prediction. This baseline incorporates an auxiliary head that predicts the current skill during training and is trained with focal loss like Skill Transformer.

Note that Decision Transformer requires the environment reward at evaluation time to compute the reward-to-go input. Skill Transformer and the other baselines do not require this assumption. We also compare to the following hierarchical baselines which first decompose the task into skills and then sequence these skills to solve the tasks.

- Task-Planning + Skills RL (**TP-SRL**) [40]: Individually train pick, place, navigate, and open skills using RL and then sequence the skills together with a fixed task planner. Each skill predicts when it should end and the next skill should begin by a skill termination action. TP-SRL trains all manipulation skills with a fixed base.

- Multi-skill Mobile Manipulation (**M3**) [16]: M3 is a modular approach similar to TP-SRL, but instead trains skills as *mobile* manipulation skills with RL. All these skills are trained to be robust to diverse starting states. Like TP-SRL, it uses a fixed task planner to sequence the skills.

- Behavior Cloning Modular skills (**BC-Modular**): Individually train transformer-based policies for each of the skills using behavior cloning with the same training dataset as Skill Transformer. Then sequence together with the same fixed task planner as the other modular baselines. This baseline helps measure the gap in performance from training policies with behavior cloning as opposed to the RL-trained policies, M3 and TP-SRL.

However, these modular baselines are designed to only execute a fixed pre-defined sequence of skills (navigate, pick, navigate, place). They cannot adjust their skill sequence based on environment observations, i.e. the task plan is inaccurate when the robot needs to open a receptacle to pick or place an object.

To address this, we additionally include a version of the modular baselines, **M3 (Oracle)** and **BC-Modular (Oracle)**, that extracts the correct task plan directly from the simulator, which we refer to as an *oracle* plan. In contrast, the proposed method, Skill Transformer, infers which skill to execute from egocentric observations without using this privileged oracle plan. Note that neither modular baseline re-executes skills if they fail, while Skill Transformer can re-plan. For further details about baselines and all hyperparameters, see Appendix C.2

5.3. Rearrange-Hard

First, we show the performance of Skill Transformer on the rearrange-hard task described in Section 3. In this task,

the agent must rearrange a single object from the start to the goal, and the object or the goal may be located in a closed receptacle. Therefore, methods need high-level reasoning to decide if the receptacle should first be opened or not.

We evaluate the success rate of all methods on both "Train" (Seen) and "Eval" (Unseen) scenes and object configurations. The success rate is measured as the average number of episodes where the agent placed the target object within 15cm of the goal within 5000 steps and without excessively colliding with the environment. Specifically, the episode will terminate with failure if the agent accumulates more than 100kN of force throughout the episode or 10kN of instantaneous force due to collisions with the scene or objects. We also report the success rate on each of the three dataset splits described in Section 3. Our evaluation dataset splits have 100 episodes in the easy split, 200 in the hard split (100 for objects starting in closed receptacles and 100 for goals in closed receptacles), and 100 in the very hard split.

Table 1 shows that Skill Transformer achieves higher success rates in both the "Train" and "Eval" settings than all of the baselines that do not have oracle planning. The Mono baseline cannot find any success, even after training for 100M steps. This demonstrates the difficulty of learning the rearrange-hard task from a reward signal alone.

The modular TP-SRL, M3, and BC-Modular methods rely on a fixed task plan to execute the separately learned skill policies. As a result, the state-of-the-art M3 method achieves a significantly lower success rate than Skill Transformer in the overall result as well as the *hard* and *very hard* splits because it is unable to adapt the skill sequence to different situations. On the other hand, Skill Transformer does not rely on a fixed ordering of skills and can adapt its high-level task-solving to new situations, achieving a 35% increase in success rate across all episodes.

Notice that even with an oracle plan, the success rate of M3 (Oracle) on the overall result for unseen scenes is only 3% higher than that of Skill Transformer. More importantly, Skill Transformer outperforms M3 (Oracle) in both *hard* and *very hard* settings, even without oracle task planning information. We find that Skill Transformer consistently achieves higher success rates in episodes with hidden objects or goals, while M3 (Oracle) experiences a substantial decrement as the difficulty increases. This difference is likely due to the low success rate in M3 (Oracle)'s opening skill, hindering its ability to complete the skill in one attempt. In contrast, as we will discuss quantitatively later in Section 5.4, Skill Transformer is capable of detecting and retrying failed skills, eventually leading to better performance in retrieving and placing the object.

Further observe that BC-Modular exhibits the lowest performance among modular methods, with an overall success rate of only 5% even when employing oracle task

Method	Train (Seen)				Eval (Unseen)			
	All Episodes	Easy	Hard	Very Hard	All Episodes	Easy	Hard	Very Hard
BC-Modular (Oracle)	5.67± 0.12	20.33± 1.70	0.83± 0.24	0.00± 0.00	5.00± 0.35	15.33± 1.70	1.67± 1.65	0.33± 0.47
M3 (Oracle)	22.92 ± 0.72	59.33 ± 1.70	13.00 ± 1.41	6.33 ± 0.47	22.58 ± 1.18	58.67 ± 1.89	12.67 ± 2.36	6.33 ± 0.47
BC-Modular	5.08± 0.42	20.33± 1.70	0.00± 0.00	0.00± 0.00	3.83± 0.42	15.33± 1.70	0.00± 0.00	0.00± 0.00
TP-SRL	7.25 ± 0.82	29.00± 3.27	0.00 ± 0.00	0.00 ± 0.00	6.67 ± 0.85	26.67 ± 3.40	0.00 ± 0.00	0.00 ± 0.00
M3	14.00 ± 0.35	**55.67** ± 0.94	0.16 ± 0.47	0.00 ± 0.00	13.25 ± 1.08	**53.00** ± 4.32	0.00 ± 0.00	0.00 ± 0.00
Mono	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
DT	7.08 ± 0.63	19.33 ± 0.47	3.83 ± 0.62	2.00 ± 0.94	6.50 ± 0.41	15.33 ± 1.25	3.83 ± 0.62	3.00 ± 1.63
DT (Skill)	8.17 ± 0.67	23.33 ± 2.49	3.00 ± 2.16	3.33± 0.94	7.67± 0.66	18.67± 2.62	4.83± 1.70	2.33± 1.25
ST (Ours)	**21.08**± 1.36	44.00± 0.82	**15.67**± 1.70	**9.00**± 2.16	**19.17**± 1.05	37.00± 0.82	**15.83**± 0.85	**8.00**± 3.27

Table 1: Success rates on the rearrangement task. "Eval" results are from test episodes in unseen scenes. In "Easy", objects and goals start on accessible locations. In "Hard", objects start in closed receptacles. In "Very Hard" objects and goals start in closed receptacles. All results are averages across 100 episodes except for the "Hard" split which is an average across 200 episodes. Grayed methods at the top use the oracle skill order. We report the mean and standard deviation across 3 random training seeds. For hierarchical methods, we retrain all the skills for each random seed.

planning. This indicates that policies trained via Behavior Cloning, which uses only one-tenth the experience of those trained with RL, encounter greater difficulty in executing intricate mobile manipulation skills. Conversely, Skill Transformer, featuring the multi-skill Action Inference module, mitigates this limitation and achieves up to 4x performance improvements across all episodes.

DT and DT (Skill) can also achieve some success in *hard* and *very hard* splits. Nonetheless, the success rates are significantly lower than Skill Transformer, especially in the *very hard* split. We observe that DT-based methods lack the capability to execute skills consistently and often mix up different skill behaviors. Also, notice that DT (Skill) with an auxiliary skill loss does not show a significant improvement in performance compared to DT without the skill loss. These observations highlight the superiority of directly conditioning on skills rather than as auxiliary rewards in learning tasks with clearly distinguished skills, such as rearrangement.

5.4. Robustness to Perturbations

We conduct a perturbation test to evaluate the re-planning ability of Skill Transformer and baselines in the rearrange-hard task. The test perturbs the environment by closing the target receptacle after the agent had opened it for the first time. To succeed, the agent must detect the drawer has been closed and adjust the current plan to re-open the receptacle and retrieve the object. We compare Skill Transformer's skill prediction and M3's task planner against ground truth skill labels derived from the privileged states of the simulation environment (detailed in Appendix C.3). The test is performed on 400 episodes that involved only objects in the closed drawer. This test compares against the highest-performing modular baseline with access to oracle task plans, M3 (Oracle), and the highest-

performing end-to-end baseline, DT (Skill).

Method	Picking Target Object		Predicted Skill Accuracy	
	No Perturb	Perturb	No Perturb	Perturb
M3 (Oracle)	24.00	0.50	44.00	10.00
DT (Skill)	10.00	8.50	39.00	33.99
ST (Ours)	**26.75**	**19.50**	**60.00**	**47.00**

Table 2: Success rates for picking up the target objects and accuracy of Skill Inference module and M3's oracle-plan task planner under both perturbed and unperturbed environment, measured by ground truth skill labels.

The findings in Table 2 demonstrate that Skill Transformer outperforms the M3 (Oracle) baseline both in the success rate of picking up the object and skill selection accuracy. Specifically, when assessing the success rates of object retrieval, M3 (Oracle) performs comparably to Skill Transformer in the absence of perturbations, but its performance significantly deteriorates to nearly 0% with perturbations in the environment. In contrast, Skill Transformer exhibits adaptability to changes in the state of the receptacle and the ability to re-open the receptacle, achieving a success rate for picking the object of 19.5% in the presence of perturbations. Note that Skill Transformer is not trained with any perturbations, and this behavior emerges from the design choices made in Skill Transformer.

Similarly, the accuracy of the predicted skills demonstrates that Skill Transformer outperforms M3 (Oracle) in high-level reasoning, both in the presence and absence of perturbations. Consistent with object retrieval, Skill Transformer demonstrates a notably higher accuracy, indicating a higher level of adaptability to perturbations, compared to M3 (Oracle), which suffers from the non-reactive task planner that does not correctly respond to perturbations.

We further compare Skill Transformer to DT (Skill), another end-to-end method that has the potential to adapt to
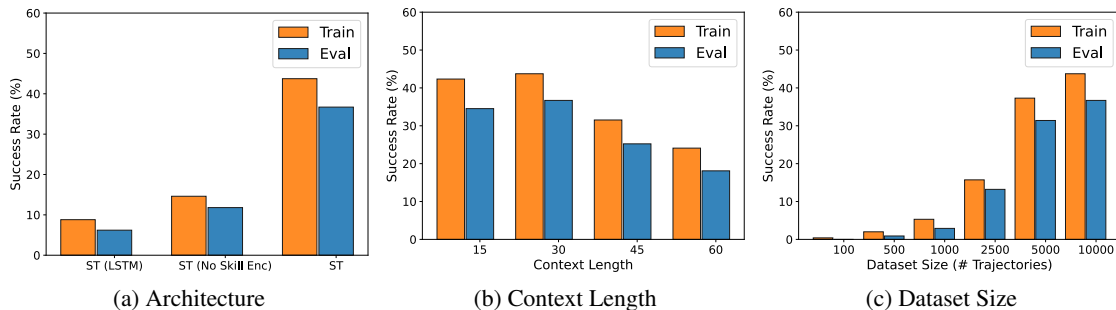
Figure 3: Success rates for ablation and analysis experiments across different policy architectures (Figure 3a), context lengths (Figure 3b), and training dataset sizes (Figure 3c) including performance on "Eval" and "Train" on the *easy* split setting.

perturbations. DT (Skill) has a substantially lower success rate in picking the target object compared to Skill Transformer, with less than half the success rate in both perturbed and unperturbed environments. In contrast to object retrieval, DT (Skill) displays only slightly lower accuracy in predicting correct skill labels than Skill Transformer. Consistent with the qualitative findings in Section 5.3, these observations suggest that although DT (Skill) demonstrates relatively good high-level reasoning ability, its primary challenge remains in learning and effectively leveraging distinct skills. On the other hand, we find that Skill Transformer exhibits the combined advantages of adaptability to environment disturbances and proficiency in completing individual skills.

5.5. Ablations and Analysis

We run ablations to address two key questions. Firstly, how necessary are the design choices of Skill Transformer, such as the transformer backbone, the Skill Inference module, and the choice of context length? Secondly, how do the dataset characteristics, such as its size and completeness, impact the performance? We perform all of the ablation experiments on the "Easy" split of the task.

5.5.1 Policy Architecture Ablation

In Figure 3a, we evaluate the effectiveness of the transformer and the Skill Inference module for mobile manipulation tasks. By replacing the Transformer backbone with an LSTM backbone, we observe a significant decline in performance. We find that the policy struggles to effectively execute individual skills, and we posit that the LSTM architecture is not as data-efficient for learning multiple skills. Our results also indicate a similarly low success rate when the Skill Inference module is removed. Qualitatively, we discovered that without the Skill Inference module, the policy is not able to differentiate between individual skills, and frequently executes inconsistent actions.

Context length refers to the maximum amount of historical information Skill Transformer takes as input. We vary

the context length from 15 to 60 timesteps and evaluate its impact on performance. Shown in Figure 3b, a shorter context length of 15 timesteps performed slightly worse, while performance decreased as the context length increased from 30 to 60 timesteps. We posit that a shorter context length may not be able to fully capture the partially observable nature of the task, while a longer context length may lead to overfitting to the training dataset. Given the random robot start location during test time, the actual trajectory at test time will differ from that in the training dataset. The longer sequence exacerbates the discrepancy between the training dataset and the test time trajectory, resulting in an increased out-of-distribution error. A moderate context length may aid capturing short-term partially observed information and mitigating overfitting to the demonstration trajectories. The trend is similar to findings from previous works [49, 8].

5.5.2 Dataset Size Analysis

As previously demonstrated [6], the size of the demonstration dataset is a vital aspect that impacts the efficacy of data-driven approaches such as Skill Transformer. As shown in Figure 3c, we trained Skill Transformer on subsets of the full demonstration dataset, ranging from 100 to 10k episodes. Our results showed that performance improved significantly when moving from 100 to 5k episodes, but the improvement is less pronounced when going from 5k to 10k demonstrations.

5.5.3 Incomplete Demonstrations

In long-horizon tasks, obtaining sufficient demonstrations of the full trajectory can be challenging. We find that Skill Transformer is also able to work effectively when trained with datasets containing only subsets of the full task. To demonstrate this, we train Skill Transformer on only two subtasks of the task: navigate-pick and navigate-place, and it achieves a success rate of 36.5% in "Train" and 27.8% in "Eval" split, compared to 44% and 37% using complete trajectories. This ablation shows that, in contrast to DT, which uses reward-conditioning across an entire episode,

Skill Transformer can also work with partial and incomplete demonstrations.

6. Conclusion

This paper presents Skill Transformer, a novel end-to-end method for embodied object rearrangement tasks. Skill Transformer breaks the problem into predicting which skills to use and the low-level control conditioned on that skill, utilizing a causal transformer-based policy architecture. The proposed architecture outperforms baselines in challenging rearrangement problems requiring diverse skills and high-level reasoning over which skills to select.

In future work, we will incorporate a more complex task specification, such as images of the desired state or natural language instructions. With the current geometric task specification, it is challenging to identify the target object after its container receptacle is manipulated. Specifically, because the object shifts inside a drawer when it is opened, its starting location no longer provides an indication of which objects is the target object. This limitation affects the success rate of retrieving objects in the challenging rearrange-hard task.

7. Acknowledgments

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[2] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

[3] Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2019.

[4] Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, et al. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020.

[5] Rogerio Bonatti, Sai Vemprala, Shuang Ma, Felipe Frujeri, Shuhang Chen, and Ashish Kapoor. Pact: Perception-action causal transformer for autoregressive robotics pre-training. *arXiv preprint arXiv:2209.11133*, 2022.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[7] Micah Carroll, Jessy Lin, Orr Paradise, Raluca Georgescu, Mingfei Sun, David Bignell, Stephanie Milani, Katja Hofmann, Matthew Hausknecht, Anca Dragan, et al. Towards flexible inference in sequential decision problems via bidirectional transformers. *arXiv preprint arXiv:2204.13326*, 2022.

[8] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[9] Alexander Clegg, Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Learning to dress: Synthesizing human dressing motion via deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(6):1–10, 2018.

[10] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Manipulathor: A framework for visual object manipulation. In *CVPR*, 2021.

[11] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[12] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[13] Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017.

[14] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, et al. Threed-world: A platform for interactive multi-modal physical simulation. *NeurIPS Datasets and Benchmarks Track*, 2021.

[15] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

[16] Jiayuan Gu, Devendra Singh Chaplot, Hao Su, and Jitendra Malik. Multi-skill mobile manipulation for object rearrangement. *arXiv preprint arXiv:2209.02778*, 2022.

[17] Kourosh Hakhamaneshi, Ruihan Zhao, Albert Zhan, Pieter Abbeel, and Michael Laskin. Hierarchical few-shot imitation with skill transition models. *arXiv preprint arXiv:2107.08981*, 2021.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[19] Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. Learning kinematic feasibility for mobile manipulation

through deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(4):6289–6296, 2021.

[20] Michael Janner, Qiyang Li, and Sergey Levine. Reinforcement learning as one big sequence modeling problem. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*, 2021.

[21] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673. PMLR, 2018.

[22] George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22, 2009.

[23] Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Winnie Xu, Sergio Guadarrama, Ian Fischer, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*, 2022.

[24] Youngwoon Lee, Joseph J Lim, Anima Anandkumar, and Yuke Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. *arXiv preprint arXiv:2111.07999*, 2021.

[25] Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In *International Conference on Learning Representations*, 2018.

[26] Chengshu Li, Fei Xia, Roberto Martin-Martin, and Silvio Savarese. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning*, pages 603–616. PMLR, 2020.

[27] Anthony Liang, Ishika Singh, Karl Pertsch, and Jesse Thomason. Transformer adapters for robot learning. In *CoRL 2022 Workshop on Pre-training Robot Learning*, 2022.

[28] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

[29] Maria Vittoria Minniti, Farbod Farshidian, Ruben Grandia, and Marco Hutter. Whole-body mpc for a dynamically stable mobile manipulator. *IEEE Robotics and Automation Letters*, 4(4):3687–3694, 2019.

[30] Mayank Mittal, David Hoeller, Farbod Farshidian, Marco Hutter, and Animesh Garg. Articulated object interaction in unknown scenes with whole-body mobile manipulation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1647–1654. IEEE, 2022.

[31] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[32] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *Proceed-ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5173–5183, 2022.

[33] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[34] Fetch robotics. Fetch. `http://fetchrobotics.com/`, 2020.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[36] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

[37] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[38] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *arXiv preprint arXiv:2209.05451*, 2022.

[39] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[40] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34, 2021.

[41] Andrew Szot, Karmesh Yadav, Alex Clegg, Vincent-Pierre Berges, Aaron Gokaslan, Angel Chang, Manolis Savva, Zsolt Kira, and Dhruv Batra. Habitat rearrangement challenge 2022. `https://aihabitat.org/challenge/2022_rearrange`, 2022.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[43] Cong Wang, Qifeng Zhang, Qiyan Tian, Shuo Li, Xiaohui Wang, David Lane, Yvan Petillot, and Sen Wang. Learning mobile manipulation through deep reinforcement learning. *Sensors*, 20(3):939, 2020.

[44] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual room rearrangement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5922–5931, 2021.

[45] Erik Wijmans, Irfan Essa, and Dhruv Batra. Ver: Scaling on-policy rl leads to the emergence of navigation in embodied rearrangement. In *Advances in Neural Information Processing Systems*, 2022.

[46] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

[47] Josiah Wong, Albert Tung, Andrey Kurenkov, Ajay Mandlekar, Li Fei-Fei, Silvio Savarese, and Roberto Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. In *Conference on Robot Learning*, pages 1367–1378. PMLR, 2022.

[48] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *International Conference on Machine Learning*, pages 24631–24645. PMLR, 2022.

[49] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. *arXiv preprint arXiv:2202.05607*, 2022.