# MuST: Multi-Head Skill Transformer for Long-Horizon Dexterous Manipulation with Skill Progress

Kai Gao[1,2] Fan Wang[1] Erica Aduh[1] Dylan Randle[1] Jane Shi[1]

***Abstract*— Robot picking and packing tasks require dexterous manipulation skills, such as rearranging objects to establish a good grasping pose, or placing and pushing items to achieve tight packing. These tasks are challenging for robots due to the complexity and variability of the required actions. To tackle the difficulty of learning and executing long-horizon tasks, we propose a novel framework called the Multi-Head Skill Transformer (MuST). This model is designed to learn and sequentially chain together multiple motion primitives (skills), enabling robots to perform complex sequences of actions effectively. MuST introduces a "progress value" for each skill, guiding the robot on which skill to execute next and ensuring smooth transitions between skills. Additionally, our model is capable of expanding its skill set and managing various sequences of sub-tasks efficiently. Extensive experiments in both simulated and real-world environments demonstrate that MuST significantly enhances the robot's ability to perform long-horizon dexterous manipulation tasks.**
**The accompanying video is available online.**

## I. Introduction

As robotics technology advances, the deployment of robots in everyday tasks is rapidly transitioning from a conceptual idea to a practical reality. Unlike traditional industrial robots, which are typically designed for repetitive, single-purpose tasks, the next generation of robots is expected to perform complex, dexterous manipulation tasks that require the seamless integration of multiple skills and the execution of diverse sub-tasks.

Recently, advances in policy learning [1]–[3] have shown great promise by effectively learning from human demonstrations to address dexterous manipulation tasks that were notoriously difficult to design and program manually. These methods leverage the richness and variety of human demonstrations, capable of capturing multi-modal data, and many also utilize foundation models to improve generalization and learning efficiency [4]–[8]. However, despite these advances, most systems are designed and tested to handle specific, single tasks and often fall short when faced with long-horizon tasks that require combining and sequencing multiple skills over extended periods [9], [10].

One of our key hypotheses for improving the reliability of policy learning in long-horizon tasks is that these tasks can be decomposed into multiple heterogeneous sub-tasks. Each sub-task requires a distinct skill, and these skills are highly reusable across different tasks. This is particularly evident in warehouse robotics, where tasks like picking and packing involve distinct skills such as flipping, grasping, and pushing. For example, Fig. 1[Top] shows a robot flipping an object from the boundary of a picking tote and compactly packing it in the corner of a packing tote.

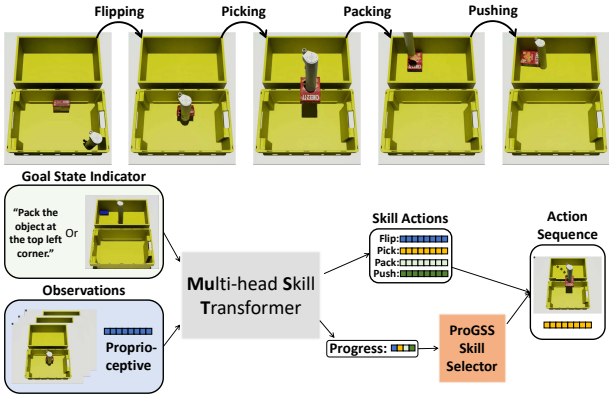


Fig. 1. [Top] An example of long-horizon dexterous manipulation. The robot executes four skills to manipulate an object from the boundary of the picking tote to the corner of the packing tote. [Bottom] Our proposed imitation learning model MuST with N-skill and skill selector ProGSS.

Decades of research have explored the problem of chaining multiple skills to achieve long-horizon tasks. Task and Motion Planning (TAMP) methods integrate high-level symbolic reasoning with low-level motion planning, enabling robots to sequence skills while ensuring physical feasibility [11]–[13]. Similarly, reinforcement learning (RL), particularly hierarchical RL, decomposes tasks into sub-skills, facilitating more efficient exploration in complex environments [14]–[17]. However, these methods still face challenges. For instance, RL methods struggle with exploration and scalability, as the search space for multi-skill tasks grows exponentially with complexity [18].

Instead, we propose a novel approach, MuST (Multi-Head Skill Transformer), designed to enhance the reliability of policy learning. MuST builds on the existing policy structure with minimal added complexity. MuST operates by decomposing long-horizon tasks into a sequence of reusable skills. At each timestamp, the appropriate skill is selected based on the current observation and state, enabled by a robust progress estimator for each skill and a skill selector.

Specifically, MuST extends the policy learning model Octo [3] by introducing multiple heads, each dedicated a specific skill, along with a progress estimator that tracks the progress of skill execution. A skill selector function, ProGSS, evalutes the progress across all skill heads to determine the most appropriate skill to execute at any given state.

[1]Amazon Robotics, MA, USA. Email: {kaigaoar, fanwanf, aduheric, dylanran, janeshi}@amazon.com.
[2]Department of Computer Science, Rutgers University, NJ, USA. Email: {kg627}@cs.rutgers.edu. Work done during the internship at Amazon Robotics.

Both the skill heads and progress estimators are trained simultaneously using the same pre-trained Octo transformer backbone. The multi-head structure of MuST enables the training of multiple skills either synchronously or asynchronously, simplifying the integration of a large skill set and the introduction of new skills as needed.

The primary advantage of MuST is that it provides a clear understanding of both individual skills and the overall task progress through continuous progress estimation. It also offers flexibility in determining when to terminate a skill by setting a termination threshold. Furthermore, MuST improves reliability in the face of disturbance, as the model consistently evaluates when to skip or redo certain skills to ensure the successful completion of the task.

Comprehensive experiments in both simulated and real-world environments demonstrate that MuST effectively addresses challenging long-horizon dexterous manipulation tasks, achieving significant improvements over the Octo baseline models. In tasks involving flipping, picking, packing, and pushing, MuST increased the overall task completion rate from 32.5% with the baseline Octo single policy to 90%.

## II. Related Works

**Policy Learning** Dexterous manipulation is a crucial capability for robots, enabling them to perform complex tasks in the human environment [2], [4], [19]–[21]. Extensive research has focused on learning dexterous skills through policy learning, both in reinforcement learning and imitation learning, with various input modalities, including vision sensory inputs [21], [22], robot states [1]–[3], [19], [23], [24] and language prompts [3], [25]. To achieve more flexible goal-oriented tasks, prior works have also explored skill learning conditioned on goal images [3], [25], [26]. For example, Du et. al. [26] generate key image frames to guide robot execution.

Recent advances have involved fine-tuning foundation models pre-trained on large image datasets [1], [19], [21], [27] or trajectories from various robot embodiments [3], [4], [20]. Specifically, Octo model team [3] pre-trains a transformer-based generalist robot policy with over 800k robot episodes from Open X-Embodiment dataset [4]. In our work, we explore multiple input modalities, including goal state indicators in natural language and goal images. We fine-tune the Octo transformer to learn a diverse set of skills concurrently.

**Long-horizon manipulation planning** Long-horizon manipulation tasks often involve a sequence of sub-tasks, requiring both the selection of the correct motion primitives and the precise execution of each action at every time step. Extensive research has focused on sequencing prehensile actions (e.g., pick and place), addressing the combinatorial challenges [28]–[30] and the need for efficient state sampling [28], [31], [32]. However, when non-prehensile actions (e.g., pushing or poking) are introduced, the increased uncertainty in the resulting states after actions complicates long-horizon planning. Researchers have tackled this challenge using Monte Carlo tree search [31], [33], probabilistic models [34], and deep reinforcement learning models [18], [35]. Additionally, previous works have generated sub-goals in the form of robot poses [24], [36] or future camera observations [26], [37] to guide robot execution. When a manipulation task requires a heterogeneous skill set, a common approach is to leverage a high-level classifier based skill selector to determine which skill to execute. Nasiriany et. al. [18] train a task policy to determine the motion primitive to apply. Belkhale et. al. [36] train a mode head to choose whether moving directly to a distant waypoint or following a dense action sequence. However, the one-hot classification result provides limited information of the progress in execution and is ambiguously annotated during the transition of skills. In this work, we propose ProGSS, a "skill progress" guided skill selector to determine the appropriate skill to execute in long-horizon dexterous manipulation tasks. Instead of training a skill selector from scratch independent to the skill set, our skill selector shares the transformer backbone with the skills to enhance efficiency in training and inference.

## III. Problem Formulation

Let the long-horizon task be represented as a sequence of *skills*, denoted by a finite set $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$, where $N$ is the total number of skills. The task execution is governed by a policy $\pi$, which, at each timestep $t$, takes as input the observation $o_t \in \mathcal{O}$ and the current robot state $x_t \in \mathcal{X}$, and outputs the action $P_t \in \mathcal{A}$, which will be further detailed below for our problem setting.

### A. Single Policy Learning

The observation $o_t$ is a representation of the environment at time $t$, which could include sensor readings or environmental context. The robot state $x_t$ includes the joint configurations, velocities, and other internal variables defining the robot's configuration. The policy $\pi$ learns to map: $\pi : (o_t, x_t) \rightarrow P_t$, where $P_t \in \mathcal{A}$ is the predicted action. This action consists of two components: the robot's 6D pose $p_t = (p_t^x, p_t^y, p_t^z, \theta_t^x, \theta_t^y, \theta_t^z) \in \mathbb{R}^6$ in Cartesian space and orientation, and a discrete value $u_t \in \{-1, 0, 1\}$ representing whether the suction is turned on ($u_t = 1$), turned off ($u_t = -1$), or remains in its current state ($u_t = 0$). Thus, the action $P_t$ predicted by the policy can be expressed as: $P_t = (p_t, u_t)$, where $p_t$ is the 6D pose and $u_t$ is the suction indicator.

### B. MuST: Decomposition of Skills and Progress

Unlike learning the long-horizon task in one policy, our method, MuST, decomposes the task into skill-specific predictions. For each skill $s_i \in \mathcal{S}$, we predict not only the robot's action $P_t^{(i)} \in \mathcal{A}$, but also a progress value $\rho_t^{(i)} \in [0, 1]$ that indicates how much of the skill $s_i$ has been completed at time $t$. The progress value evolves over time and reaches $\rho_t^{(i)} = \theta_i$, where $\theta_i$ is the termination threshold when a skill is considered fully executed. Thus, for each

skill $s_i$, MuST outputs a tuple: $(P_t^{(i)}, \rho_t^{(i)})$ at each timestep $t$, where $P_t^{(i)} = (p_t^{(i)}, u_t^{(i)})$ is the predicted action for skill $s_i$, and $\rho_t^{(i)}$ is the progress indicator.

At each timestep $t$, a skill selector function $\sigma$ determines the next skill to execute based on the progress values $\rho_t^{(i)}$ for all skills: $\sigma : \{\rho_t^{(1)}, \rho_t^{(2)}, \ldots, \rho_t^{(N)}\} \rightarrow s_j$, where $s_j \in \mathcal{S}$ is the selected skill to be executed at time $t$. The robot then executes the action $P_t^{(j)}$ predicted by MuST for the selected skill $s_j$: $a_t = P_t^{(j)}$, where $a_t \in \mathcal{A}$ represents the robot's pose and suction status to execute at timestep $t$.

### C. Goal-Conditioned Policy Learning

In both the single policy learning and MuST approaches, an alternative model variant can be used where the input also encodes a goal condition. This goal condition can be represented as either a goal image $I_g \in \mathcal{I}$ or a goal language instruction $l_g \in \mathcal{L}$, where $\mathcal{I}$ is the set of possible goal images and $\mathcal{L}$ is the set of possible language instructions. The policy can then be learned as a mapping that includes the goal condition: $\pi : (o_t, x_t, I_g \text{ or } l_g) \rightarrow P_t$, where the goal condition, whether in the form of a goal image $I_g$ or a language instruction $l_g$, provides additional information to the policy for determining the action $P_t$. In this variant, the policy outputs $P_t = (p_t, u_t)$, as described earlier, but the decision process is now conditioned on the provided goal.

## IV. Methodology

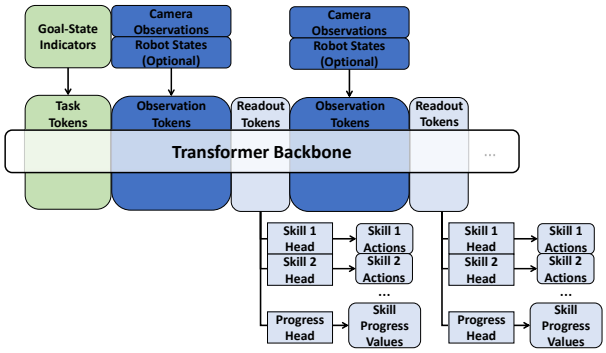


Fig. 2. Overview of MuST (Multi-Head Skill Transformer). The model consists of a pre-trained Octo transformer backbone [3] and $N+1$ heads for an $N-$skill set. Each of the skill head computes an action sequence of its skill.The progress head ProGSS, the skill selector, estimates the progress of the entire skill set.

### A. Multi-Head Skill Transformer (MuST)

MuST reduces action complexity by decomposing long-horizon tasks into a set of skills. It learns each skill individually and then combines them sequentially to complete the overall task. MuST consists of $N+1$ action heads: one for each skill in $\mathcal{S}$ and one *progress head* to estimate the progress of skill execution. The $N+1$ heads all share the same Octo transformer backbone [3], which is pre-trained on 800k robot manipulation episodes.

Fig. 2 is an overview of the MuST architecture, which includes three key components: input tokenizers, a transformer backbone, and output decoding heads for N skills, along with a progress head. The input tokenizers and the transformer backbone are the same as in Octo, while the readout token serves as a fused representation of the multi-modal observations. The skill heads and progress heads take the readout token as input. Each head is an L1 action head consisting of a multi-head attention pooling block followed by a dense layer. Each skill head computes an action sequence for the corresponding skill $P_t^{(i)}$, for all $i = 1, 2, \ldots, N$, and the progress head learns a numerical vector $\{\nu_t^{(1)}, \nu_t^{(2)}, \ldots, \nu_t^{(N)}\}$ to estimate the current progress of each skill.

### B. ProGSS: Progress Guided Skill Selector

Given the progress vector from the progress head, we design a progress-guided skill selector, ProGSS, to determine the optimal skill to execute at the current state. Progress values range from 0 to 1, with lower values indicate pending or ongoing execution, while higher values suggest completion. These values are crucial for identifying the current phase of execution in long-horizon manipulation tasks.

*1) Object-Centric Progress Annotation:* We define each skill's trajectory segment as comprising of three parts: the pre-skill transit trajectory, the skill execution trajectory, and the post-skill transit trajectory.

Skill execution period is defined as the time between the robot's first and last contact with the object. We increase progress value only during this phase, making the progress value object-centric and less sensitive to the robot's pose.

Fig. 3 shows the progress annotation of skills within a skill-related trajectory segmentation. For pre- and post-skill transit trajectories, we set the progress value to be $\alpha$ and 1, respectively. Here $\alpha := 1 - t/M$, where $t$ is the duration of skill execution in the current episode, and $M$ is the maximum duration of skill execution across all demonstration episodes for that skill. The progress value increases linearly with each time step in the demonstration episodes.

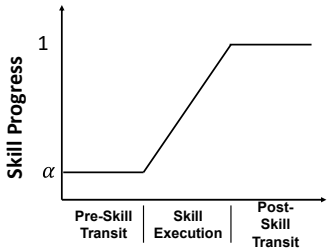


Fig. 3. Annotation of skill progress in a skill-related episode segment.

*2) Skill Selection with Single Skill Sequence:* When a single skill sequence $\tau = \{s_i\}_{i=1}^{|\tau|}$ of the manipulation task is demonstrated, ProGSS selects the first skill $s_i$ in the sequence with the current value $\nu_t^{(i)}$ below its termination threshold $\theta_i$. Fig. 4 is an example where there are 4 skills in the skill set and 3 of them ($|\tau| = 3$) form a sequence $s_1 \rightarrow s_2 \rightarrow s_3$. When ProGSS infers a progress vector of $(1, 0.2, 0, 1)$, $s_2$ is selected for execution next, given that $\theta_i = 0.9, \forall 1 \leq i \leq 4$. In practice, the termination threshold is a hyperparameter that controls the tradeoff between execution precision and speed. Generally, the model's performance is not sensitive to moderate changes in this threshold value. Note that $s_4$ is not included in this skill sequence so its progress value remains at 1 throughout execution.

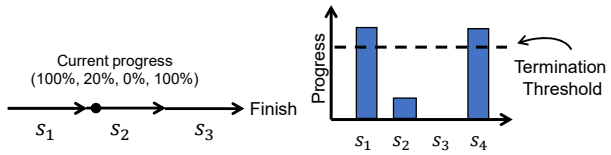


Fig. 4. An example of ProGSS with a single skill sequence.

In some demonstration trajectories, a skill $s_i$ may be divided into $k$ disjointed segments ($k > 1$). Instead of increasing continuously from $\alpha$ to 1, the progress moves between stages, with each stage representing a segment of the skill. The progress value $\nu_t^{(i)}$ increments in steps as the robot completes each segment. For each segment $j$, progress $\nu_t^{(i)}$ increases from $\alpha_j^i$ (start of the segment) to $\alpha_{j+1}^i$ (end of the segment). The upper bound progress for segment $j$ is calculated as: $\alpha_j^i = \alpha_{j-1}^i + (1-\alpha) \cdot T_j^i / \sum_{j=1}^k T_j^i$.

In this formula, $\alpha_0^i = \alpha$ is the initial value. $T_j^i$ is the duration of segment $j$ of skill $s_i$, and $\sum_{j=1}^k T_j^i$ is the total duration of all segments for skill $s_i$. $(1-\alpha)$ represents the available progress range (from $\alpha$ to 1). Thus, the progress value increases based on the relative duration of each segment.

At time $t$, ProGSS selects the first segment $j$ of skill $s_i$ where the progress $\nu_t^{(i)}$ is below both the termination threshold $\theta_i$ and the upper bound $\alpha_j^i$, i.e. $\nu_t^{(i)} < \min(\theta_i, \alpha_j^i)$.

Here, $\theta_i$ is the termination threshold for skill $s_i$, and $\alpha_j^i$ is the progress upper bound for segment $j$. This ensures that the robot progresses through the skill in an incremental manner, selecting the next segment that is not yet completed.

*3) Skill Selection with Multiple Skill Sequences:* When multiple skill sequences $\{\nu_i\}$ are demonstrated for any combination of skills, Alg. 1 is used to select next skill. We compute the progress trajectories of demonstrated skill sequences, resulting in a map, $M$, in the skill progress space (line 1). ProGSS searches for the progress trajectory in $M$ with the least Euclidean distance to $\nu_t$ and outputs the corresponding skill sequence $\tau$ (Line 2). We choose the first skill $s_i \in \tau$ with progress value below its threshold, i.e., $\nu_t^{(i)} < \theta_i$ (Line 3-4).

**Algorithm 1:** Arbitrary-Sequence-Skill-Selector

**Input :** $T$: Demonstrated skill orderings;
$\nu_t$: Estimated progress values;
$\Theta$: Progress termination thresholds.
**Output:** $s$: next skill to execute.

1 $M \leftarrow$ Progress-Trajectory-Computation($T$)
2 $\tau \leftarrow$ Nearest-Demonstrations($\nu_t, T, M$).
3 **for** $s_i \in \tau$ **do**
4 **if** $\nu_t^{(i)} < \theta_i$ **then return** $s_i$;

### C. Skill Set Training and Expansion

MuST multi-head architecture enables concurrent training for a skill set, fine-tuning of a single skill with additional data, and expansion of the existing skill set. First, during batch training, we alternate training of each skill $s_i$ in the set. Given the dataset for $s_i$, only the parameters of the transformer backbone, its corresponding skill head, and the progress head are updated. Second, when fine-tuning a skill $s_i$ with additional data, we freeze the parameters of the transformer backbone and only update the skill head. Third, to introduce a new skill into the existing model, we add a new action head, extend the dimension of progress head, and then train the new skill head and progress head. The performance of other skills remains unaffected during fine-tuning or expansion, as the backbone is frozen.

## V. Evaluation

In this section, we evaluate the performance of MuST in both simulated and real-robot environments, with a total of six experiments, under the various conditions:

- The performance gain of MuST compared to the single-head Octo model
- MuST's performance conditioned on task goals specified by images or language instructions
- MuST's performance across a diverse set of objects
- MuST's ability to react to unexpected environment disturbance based on the skill progress values

The performance is evaluated using two metrics:

1) **Skill completion and task completion:** we report the completion of each individual skill, and the task is considered complete only if all skills in the task are successfully executed. Any failed skill will result in failure of all subsequent skills.
2) **Execution time:** The execution time is defined as total time spent on manipulation until the object consistently remains at the goal pose $g$ for 100 time steps.

We compare MuST with the single-head Octo model, which learns all skills without progress estimation. Both models finetune Octo-Base (93M params) checkpoint and use L1 action heads as decoding heads for skills and progress. For closed-loop control scenarios, both models carry out inference every 50 time steps and compute the action sequence for the next 50 time steps. We train the models with an Nvidia V100 16GB GPU.

### A. Simulation Experimental Results and Comparison with Octo Baseline

Our goal-state conditioned Pick-n-Pack manipulation task Fig. 1[Top] consists of four skills: First, the robot flips down the object from the tote boundary to enable picking. Next, it picks the object from the picking tote. Based on the goal-state indicator $I_g$ or $l_g$, the robot packs the object near the desired corner of the packing tote. Finally, the robot pushes the object, both rotating and translating it, to fit it tightly the desired corner.

The goal is given by either a language prompt or a goal image Fig. 5 . For the goal images, instead of specific images of objects, we use a blue patch to suggest the goal state which makes it independent of object appearance, enhancing the model's generalization capability across different object types.

Tab.I reports the performance metrics for both MuST and the Single Head Octo. The task is language conditioned Pick-n-Pack with the "long box"(Fig. 6), and each task is repeated

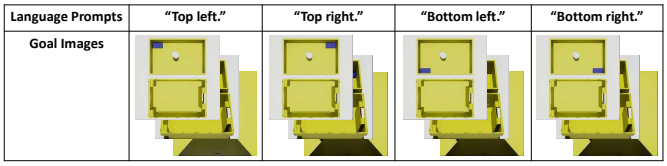


Fig. 5. We use either language prompts or images as goal state indicators to customize packing poses.

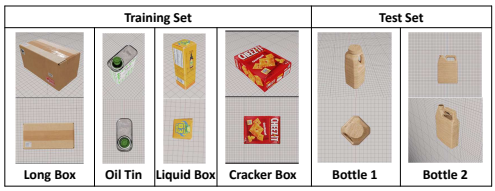


Fig. 6. Training object set and test object set in simulation. The 3D model used are open-source models sampled from the YCB Object and Model Set [38], NVIDIA SimReady assets [39], open-source models from SketchFab [40], and the Google Scanned Objects dataset [41].

10 times. The training dataset include 15 demonstrations for the long box at each of the four goal poses. We collected demonstration trajectories by manipulating the target end-effector pose using mouse and keyboard inputs within the Isaac Sim environment. While Octo model succeeded in $32.5\%$ tasks, MuST maintains $80\% - 90\%$ success rate. In addition, MuST is $23.7\% - 38.4\%$ faster than Octo in execution time for the finished tasks. The results suggest that MuST is more robust than Octo baseline in long horizon manipulation tasks.

TABLE I

LANGUAGE CONDITIONED PICK-N-PACK (LONG BOX)

| | Task Completion | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Flip→Pack | | Push (Orientation) | | Push (Position) | | Execution Time | |
| End State | MuST | Octo | MuST | Octo | MuST | Octo | MuST | Octo |
| Top Left | 10/10 | 7/10 | 10/10 | 4/10 | 10/10 | 4/10 | 1324 | 1734 |
| Top Right | 9/10 | 9/10 | 8/10 | 7/10 | 8/10 | 4/10 | 1530 | 2639 |
| Bottom Left | 10/10 | 7/10 | 9/10 | 5/10 | 9/10 | 3/10 | 1538 | 2337 |
| Bottom Right | 9/10 | 6/10 | 9/10 | 4/10 | 9/10 | 2/10 | 1571 | 2550 |

### B. Additional Evaluation of MuST and ProGSS in Simulation

In this section, we further evaluate the performance of MuST, conditioned with goal images, on a diverse object set, and on progress estimation.

Tab. II summarizes the performance of MuST on image-conditioned Pick-n-Pack task on the same Long Box object. MuST maintains $80\% - 90\%$ success rate for 40 evaluation trials.

Furthermore, we evaluate MuST on a diverse object set. In this experiment, we train MuST on four different boxes(Fig. 6). We test MuST on all objects including four training objects and two novel objects, and results are reported in Tab. III for 20 trials on each object. For the first three skills, MuST maintains over $90\%$ completion rate on training object set and that drops to $70\% - 75\%$ on objects in the new category. For the last push skill, MuST solves $80\%$ test cases on the cracker box, $65\% - 70\%$ on other training objects. MuST only finishes around $40\%$ pushes on novel objects. In the training dataset with cuboid objects, the push demonstrations use box corners for rotation. Our hypothesis is that the same push behavior does not generalize well to the tested novel objects with irregular shapes.

TABLE II

IMAGE CONDITIONED PICK-N-PACK (LONG BOX)

| | Task Completion | | | | | |
|---|---|---|---|---|---|---|
| Packing Corner | Flip | Pick | Pack | Push (Orientation) | Push (Position) | Execution Time |
| Top Left | 10/10 | 10/10 | 10/10 | 9/10 | 8/10 | 1570 |
| Top Right | 9/10 | 9/10 | 9/10 | 8/10 | 8/10 | 1765 |
| Bottom Left | 10/10 | 10/10 | 10/10 | 9/10 | 9/10 | 1225 |
| Bottom Right | 10/10 | 10/10 | 10/10 | 8/10 | 9/10 | 1158 |

TABLE III

LANGUAGE-CONDITIONED PICK-N-PACK WITH DIVERSE OBJECT SET

| | Task Completion | | | | | |
|---|---|---|---|---|---|---|
| Test Object | Flip | Pick | Pack | Push (Orientation) | Push (Position) | Execution Time |
| Cracker Box | 20/20 | 20/20 | 19/20 | 16/20 | 16/20 | 2073 |
| Liquid Box | 20/20 | 20/20 | 19/20 | 14/20 | 13/20 | 1769 |
| Long Box | 20/20 | 20/20 | 18/20 | 14/20 | 13/20 | 1330 |
| Oil Tin | 20/20 | 20/20 | 19/20 | 14/20 | 13/20 | 1875 |
| Bottle1 (OOD) | 20/20 | 20/20 | 15/20 | 12/20 | 9/20 | 1608 |
| Bottle2 (OOD) | 20/20 | 20/20 | 14/20 | 9/20 | 7/20 | 2009 |

Additionally, we demonstrate that MuST can react to unexpected environment disturbance based on the skill progress values. For example, when the user resets an object on the tote edge, MuST will roll back to select the Flip skill. Similarly, MuST will skip the first skill, and start directly with the second Pick skill when the object is in the pick state. We include these demonstrations with the skill progress value graphs in the accompanying video.

### C. Handling Multiple Sequences in Simulation

We designed the task of multi-sequence pick-n-pack to evaluate the performance of MuST when multiple skill sequences are demonstrated. As shown in Fig. 7, the robot is tasked to flip down a box and place it at the packing tote. Specifically, when the object is located in the central area of the tote, the robot can choose to flip the object before or after pick-n-place. However, when the object is located at the boundary of the picking tote, the robot cannot execute pick-n-place before a successful flip. For each of the three cases of skill sequences in Fig. 7, we made 50 demonstrations.

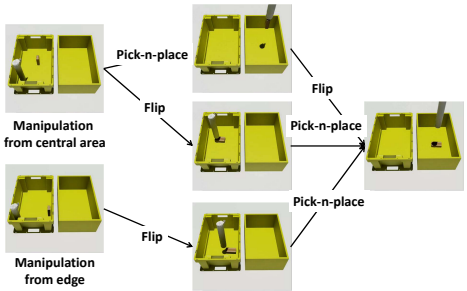


Fig. 7. Multi-sequence pick-n-pack. When the object is sampled at the central area of the tote, there are two possible skill orderings; when the object is sampled at the edge, the robot cannot directly pick it up before flipping.

Tab. IV shows sequence selection distribution and task completion rate. When the object is sampled at the central area of the picking tote, out of the 80 trials, MuST chooses "pick first" and "flip first" sequences in 37.5% and 62.5% trials respectively with around 80% success rate on both skill sequences. When the object is sampled at the edge of the picking tote, picking directly is impossible. MuST chooses the "flip first" sequence in 96.2% test cases. The results suggest that MuST effectively handles multiple skill sequences and avoids "modality collapses" in long horizon manipulation tasks.

TABLE IV

MULTI-SEQUENCE PICK-N-PACK

| Test Cases | Pick First | Flip First |
|---|---|---|
| Manipulate from central area | 24/30 | 39/50 |
| Manipulate from edge | 0/3 | 73/77 |

### D. Physical Experimental Results

Our robotic test bed (Fig. 8[Right]) comprises a collaborative manipulator equipped with a customized suction gripper(Fig. 8[Left]), which is capable of vacuum suction and dexterous contact with its soft tip. Two 5 MP 3D cameras are positioned with one above each tote.

Similarly, the experimental task (Fig. 9) consists of four skills: flips down an object from the edge of the picking tote, grasps it with the suction cup, packs it at the proper pose, based on a goal image using a generic brown box(Fig. 10(c), and pushes it to the corners of the tote. The first two skills have clear success or failure criteria, while packing succeeds if the object is in the correct quarter of the tote, and pushing succeeds if the object is within 2 cm of the correct corner.

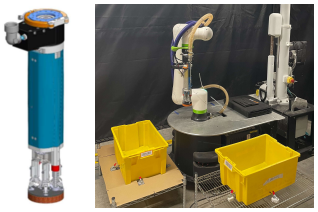

Fig. 8. [Left] A customized suction gripper capable of vacuum suction and dexterous contact. [Right] Physical robotic system.

We evaluate MuST in a open-loop control framework, where MuST takes a single state observation, two images of two totes plus an image of the in-hand object (if any), and outputs the trajectory for the selected skill. The robot then executes the entire trajectory in an open loop and moves out of the observable environment. To collect offline data, we developed a software that enables annotation of sparse waypoints during robot execution, which are then interpolated to generate dense trajectories. We use a set of five objects (Fig. 10 for the physical experimental task. For each object in our training set (Fig. 10(a)), we collect 15, 15, 24, and 24 human demonstrations for the four skills respectively.

We first evaluate both MuST and Octo on the couscous box with different goal-state images (Tab. V). Both models succeed in the first three skills in all the test cases but Octo only successfully pushes the object to corners in 35% test cases. In Tab. VI, we show the experiment results on the


Fig. 9. Task sequence of real robot goal-state conditioned pick-n-pack.


(a) (b) (c)

Fig. 10. Training (a) and test (b) object set for physical experiments. (c) Image of the packing tote with a universal object as the goal-state indicator.

diverse object set. Octo has 0% success rate on the small object ("Instax" box) and the novel object (Bag). It also fails in pushing other objects to corners in most test cases. In contrast, MuST finishes 88% test cases among the objects. In addition, most of the failures of Octo in pushing are due to collisions with the environment, while two failures of MuST on pushing are inaccurate location with the open loop limitation: the final object pose is slightly outside the goal region (2.1 cm and 2.6 cm away from the corner). In summary, MuST has much higher success rate in finishing the whole task than Octo, especially on small objects and novel objects.

TABLE V

IMAGE CONDITIONED PICK-N-PACK I (COUSCOUS BOX)

| | Flip | | Pick | | Pack | | Push | |
|---|---|---|---|---|---|---|---|---|
| **Goal State** | **MuST** | **Octo** | **MuST** | **Octo** | **MuST** | **Octo** | **MuST** | **Octo** |
| Top Left | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 4/5 | 3/5 |
| Top Right | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 2/5 | 0/5 |
| Bottom Left | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 2/5 |
| Bottom Right | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 2/5 |

TABLE VI

IMAGE-CONDITIONED PICK-N-PACK II (BOTTOM LEFT CORNER)

| | Flip | | Pick | | Pack | | Push | |
|---|---|---|---|---|---|---|---|---|
| **Object** | **MuST** | **Octo** | **MuST** | **Octo** | **MuST** | **Octo** | **MuST** | **Octo** |
| "Instax" Box | 4/5 | 0/5 | 4/5 | 0/5 | 4/5 | 0/5 | 2/5 | 0/5 |
| "Mina" Box | 5/5 | 4/5 | 5/5 | 4/5 | 5/5 | 4/5 | 5/5 | 3/5 |
| Couscous Box | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 2/5 |
| Rice Box | 5/5 | 5/5 | 5/5 | 5/5 | 5/5 | 4/5 | 5/5 | 1/5 |
| Bag (OOD) | 5/5 | 0/5 | 5/5 | 0/5 | 5/5 | 0/5 | 5/5 | 0/5 |

## VI. CONCLUSION

In this work, we introduce MuST, a skill chaining model designed for long horizon dexterous manipulation. Given a task decomposed into $N$ skills, MuST trains $N + 1$ heads: $N$ heads to learn individual skills and an additional head for skill progress estimation. Based on the estimated progress values, a skill progress guided skill selector ProGSS chooses the proper skill to execute at each time step. Qualitative results demonstrate that ProGSS effectively adapts to unexpected disturbance. Comprehensive experiments, both in simulation and real world settings. highlight MuST's performance advantages over the single-head Octo baseline, as well as its ability to manage various skill sequences and diverse object sets.

## REFERENCES


[1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668.

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[3] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 2024.

[4] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "Rt-1: Robotics transformer for real-world control at scale," 2023. [Online]. Available: https://arxiv.org/abs/2212.06817

[6] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *7th Annual Conference on Robot Learning*.

[7] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," in *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*, 2024.

[8] J. Yang, Z. Cao, C. Deng, R. Antonova, S. Song, and J. Bohg, "Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning," in *8th Annual Conference on Robot Learning*.

[9] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," 2021. [Online]. Available: https://arxiv.org/abs/2003.06085

[10] Y. Duan, M. Andrychowicz, B. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1087–1098.

[11] N. Dantam, Z. Kingston, S. Chaudhuri, and L. Kavraki, "Incremental task and motion planning: A constraint-based approach," 06 2016.

[12] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 639–646.

[13] J. Wolfe, B. Marthi, and S. Russell, "Combined task and motion planning for mobile manipulation," in *Proceedings of the Twentieth International Conference on International Conference on Automated Planning and Scheduling*, ser. ICAPS'10. AAAI Press, 2010, p. 254–257.

[14] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *Advances in neural information processing systems*, vol. 29, 2016.

[15] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *International Conference on Learning Representations*.

[16] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 3540–3549.

[17] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[18] S. Nasiriany, H. Liu, and Y. Zhu, "Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7477–7484.

[19] R. M. Shah and V. Kumar, "Rrl: Resnet as representation for reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9465–9476.

[20] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, A. Irpan, N. J. Joshi, R. Julian, S. Kirmani, *et al.*, "Autort: Embodied foundation models for large scale orchestration of robotic agents," in *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*.

[21] T. Yang, Y. Jing, H. Wu, J. Xu, K. Sima, G. Chen, Q. Sima, and T. Kong, "Moma-force: Visual-force imitation for real-world mobile manipulation," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 6847–6852.

[22] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[23] 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*. PMLR, 2022, pp. 1678–1690.

[24] F. Liu, K. Fang, P. Abbeel, and S. Levine, "Moka: Open-vocabulary robotic manipulation through mark-based visual prompting," in *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*.

[25] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "Vima: General robot manipulation with multimodal prompts," in *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

[26] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel, "Learning universal policies via text-guided video generation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[27] K. Xu, S. Zhao, Z. Zhou, Z. Li, H. Pi, Y. Zhu, Y. Wang, and R. Xiong, "A joint modeling of vision-language-action for target-oriented grasping in clutter," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 597–11 604.

[28] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proceedings of the international conference on automated planning and scheduling*, vol. 30, 2020, pp. 440–448.

[29] K. Gao, S. W. Feng, B. Huang, and J. Yu, "Minimizing running buffers for tabletop object rearrangement: Complexity, fast algorithms, and applications," *The International Journal of Robotics Research*, vol. 42, no. 10, pp. 755–776, 2023.

[30] R. Wang, K. Gao, D. Nakhimovich, J. Yu, and K. E. Bekris, "Uniform object rearrangement: From complete monotone primitives to efficient non-monotone informed search," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6621–6627.

[31] H. Chang, K. Gao, K. Boyalakuntla, A. Lee, B. Huang, J. Yu, and A. Boularias, "Lgmcts: Language-guided monte-carlo tree search for executable semantic object rearrangement," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 13 607–13 612.

[32] K. Gao, D. Lau, B. Huang, K. E. Bekris, and J. Yu, "Fast high-quality tabletop rearrangement in bounded workspace," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1961–1967.

[33] E. R. Vieira, K. Gao, D. Nakhimovich, K. E. Bekris, and J. Yu, "Effective and robust non-prehensile manipulation via persistent homology guided monte-carlo tree search," in *International Symposium on Experimental Robotics*. Springer, 2023, pp. 192–202.

[34] U. A. Mishra, S. Xue, Y. Chen, and D. Xu, "Generative skill chaining: Long-horizon skill planning with diffusion models," in *Conference on Robot Learning*. PMLR, 2023, pp. 2905–2925.

[35] C. Agia, T. Migimatsu, J. Wu, and J. Bohg, "Stap: Sequencing task-agnostic policies," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7951–7958.

[36] S. Belkhale, Y. Cui, and D. Sadigh, "Hydra: Hybrid robot actions for imitation learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 2113–2133.

[37] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," *arXiv preprint arXiv:2003.06085*, 2020.

[38] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 International Conference on Advanced Robotics (ICAR)*, 2015, pp. 510–517.

[39] NVIDIA Corporation, "Nvidia simready assets," https://developer.nvidia.com/omniverse/simready-assets, 2024, https://developer.nvidia.com/omniverse/simready-assets.

[40] Sketchfab, Inc., "Sketchfab open-source 3d models," 2024, accessed: 2024-10-31. [Online]. Available: https://sketchfab.com/tags/open-source

[41] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke, "Google scanned objects: A high-quality dataset of 3d scanned household items," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2553–2560.