IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Playbook: Scalable Discrete Skill Discovery from Unstructured Datasets for Long-Horizon Decision-Making Problems

Minjae Kang[1], Mineui Hong[2], and Songhwai Oh[1]

***Abstract*—Skill discovery methods enable agents to tackle intricate tasks by acquiring diverse and useful skills from task-agnostic datasets in an unsupervised manner. To apply these methods to more general and everyday tasks, the skill set must be scalable. However, current approaches struggle with this scalability, often facing the challenge of catastrophic forgetting when learning new skills. To address this limitation, we propose a scalable skill discovery algorithm, a *playbook*, which can accommodate unseen tasks by acquiring new skills while maintaining previously learned ones. The scalable structure of the playbook, consisting of finite and independent plays and primitives, enables expansion by adding new elements to accommodate new tasks. The proposed method is evaluated in the complex robotic manipulation benchmarks, and the results show that the playbook outperforms existing state-of-the-art methods. We release code for the playbook and pretrained checkpoints at https://github.com/rllab-snu/Playbook.**

***Index Terms*—Learning from Demonstrations, Continual Learning, Machine Learning for Robot Control**

I. Introduction

RECENT studies on skill discovery have successfully addressed challenging decision-making tasks such as maze navigation [1], [2], locomotion [3], [4], and robotic manipulation [5], [6]. Skill discovery, a hierarchical policy learning method, equips an agent with the skills necessary to solve complex tasks by identifying and acquiring useful and diverse skills through an unsupervised learning manner. These methods learn the skill space or skill set by embedding task-agnostic trajectories, which are collected by actively exploring the environment [7], [8] or pre-collected using a behavior policy [9]–[11]. The discovered skills can be leveraged for solving downstream tasks, e.g., reaching goals or maximizing rewards designed for a specific task.

To apply skill discovery methods to more general and everyday tasks, the learned skill space or skill set must be scalable. For example, a cooking robot should be able to learn additional recipes using new cooking tools or ingredients. However, continually adopting new skills is a challenging problem. In practical scenarios, it is commonly assumed that access to the original dataset is limited or even unavailable due to constraints such as storage limitations and restricted access to the initial data [12], [13]. The restricted access to original datasets often leads to catastrophic forgetting [14], which means losing previously acquired abilities when learning new ones. Therefore, despite its importance, there is a lack of research on expanding an existing skill set to address unseen tasks. Existing studies such as [15], [16] also solve downstream tasks using pre-acquired skills rather than focusing on learning entirely new skills. To address this limitation, we propose a novel algorithm, a *playbook*, that accommodates new skills from new datasets while maintaining previously learned ones with only a small portion of the initial datasets.

The playbook consists of a finite number of plays and primitives. Each primitive of the playbook generates an independent action distribution using only the current state. In contrast, each play of the playbook represents a weight vector used to combine the multiple action distributions generated by primitives, determined by the current state and goal, as shown in Figure 1. Thus, each primitive outputs the same action distribution regardless of goals, but the combined action distribution changes based on the selected play. Therefore, plays represent the skills of the playbook, and the tasks performed by the playbook vary depending on which play is chosen.

The playbook features a scalable structure with multiple plays and primitives. Since the plays and primitives are independent, the pre-trained playbook can be structurally expanded by adding more plays and primitives to cover new datasets. Increasing the number of primitives enhances the expressive power of the playbook. Meanwhile, increasing the number of plays widens the range of actions the playbook can choose from. With a finite number of plays, we can interpret a goal-conditioned decision-making problem as a sequential play classification problem. From this perspective, we extend the playbook using class-incremental learning techniques for image classification [17]. Using the extended playbook, we can solve compounded problems, which are a mixture of old and new tasks. For instance, if we have a pre-trained playbook that can open a drawer and extend it for the new task of picking up a block, an extended playbook can learn to pick up the block in the closed drawer.

Our primary contribution is to propose a scalable skill dis-

This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments. This work was partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00480, Development of Training and Inference Methods for Goal-Oriented Artificial Intelligent Agents, 50%, and No. 2019-0-01190, (SW Star Lab) Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning, 50%). *(Corresponding author: Songhwai Oh.)*

[1] Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul, Korea (e-mail: minjae.kang@rllab.snu.ac.kr, songhwai@snu.ac.kr).

[2] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA (e-mail: mineuih@andrew.cmu.edu).

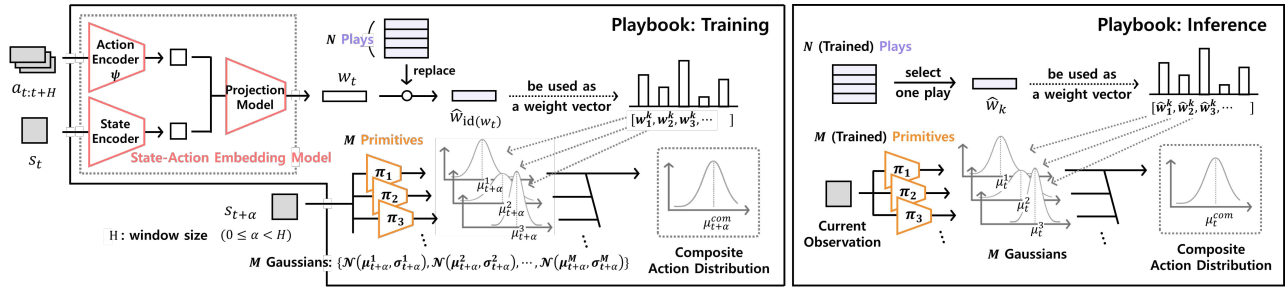


Fig. 1: (Left) Overall structure for the playbook training. The state-action embedding model embeds a state and an action sequence from an offline dataset to select one play vector among the plays. Each primitives outputs an action distribution by using a state as an input. The playbook uses the selected play vector as weights to form a single action distribution with primitives. (Right) Inference process of the trained playbook. The playbook selects a play vector by considering both the current and goal states and uses it to combine distributions generated by primitives.

covery method that can accommodate new tasks by expanding its structure. The playbook has the following strengths: (1) The playbook covers multi-modal behavior distributions with a small number of plays. We have experimentally verified that the playbook with a finite number of plays shows better performance than existing baselines that train the continuous skill space. Specifically, on the CALVIN benchmark [18], the playbook achieves a success rate of 21.4% for challenging robotic problems that require an agent to decide and perform several hidden tasks. (2) The playbook can be extended to adopt new datasets. We have also experimentally verified that when new datasets are provided on CALVIN, the extended playbook shows an average success rate of 77.0% for old and new tasks. (3) The extended playbook can solve compounded problems by mixing plays learned from different datasets. We have verified that the extended playbook records a success rate of 24.4% for challenging compounded problems on CALVIN.

II. Related Work

A. Hierarchical Policy Learning Using an Offline Dataset

Recent research on skill discovery has successfully solved long-horizon tasks by acquiring skills from an offline task-agnostic dataset [10], [19]. These studies generally aim to solve intricate tasks by deploying skills and address downstream tasks by reusing or fine-tuning previously learned skills. Existing methods learn the skill space by encoding actions [1], states [9], and state-action pairs [2], [5] of the dataset and generate raw actions using a skill-conditioned policy. On the other hand, the playbook obtains skills by embedding a state and an action sequence and uses the skill as a weight vector of primitives rather than a direct input of the policy. Skill discovery studies such as [8], [20] train discrete skills to represent multi-modal behaviors of offline datasets, but they do not expand the skill set for new tasks. In contrast, the playbook extends their skill set to cover completely new tasks.

B. Class-Incremental Learning for Image Classification

Traditional supervised image classification methods [21], [22] excel in static settings but face challenges in class-incremental learning due to the catastrophic forgetting problem. There exist techniques to address this issue, such as the dynamic architecture method [12], [23], which utilizes a flexible neural network structure to accommodate new classes, and the knowledge distillation method [13], [24], which transfers knowledge from a larger or pre-trained model to a smaller or evolving one. Meanwhile, the gradient boosting method [25], [26] minimizes an empirical error for a new dataset by iteratively adding weak functions to the existing one to mitigate catastrophic forgetting. FOSTER [17], which is a gradient boosting method, defines additional parameterized models to cover a new dataset while fixing the existing models and minimizes image classification loss. To alleviate catastrophic forgetting, we employ FOSTER for training the extended playbook.

III. Learning Playbook from Offline Datasets

In this paper, we propose a novel skill discovery method, a *playbook*, which has a scalable structure. We refer to a skill of the playbook as a *play*. The main components of the playbook include N plays and M primitives, as shown in Figure 1. This section describes the playbook structure and its training process. Note that we train the playbook using unstructured datasets composed solely of states and actions.

In Section IV, we explain how the playbook is extended to learn new plays from new datasets. Since the playbook is composed of independent plays and primitives, it can be structurally expanded by adding more plays and primitives. Based on this idea, we propose the method of playbook extension through class-incremental learning. In Section V, we describe the inference process for generating raw actions using the trained playbook. We introduce methods for play planning, which determine the play indices needed to achieve a given goal.

A. Generating Action Distributions Using Primitives

The playbook has M parameterized primitives $\{\pi_{\phi_1}, \cdots, \pi_{\phi_M}\}$, and each primitive generates an independent probability distribution over the raw action space, $\pi_{(\cdot)}(a|s)$, taking the state s as an input. Then, the playbook combines the M distributions into a single distribution using a weight vector for primitives. Thus, the action distribution inferred by the playbook varies depending on the weight vector used.

There are two reasons for utilizing multiple primitives in the playbook. First, combining multiple distributions provides a wide range of actions rather than using one distribution. It is advantageous in expressing multi-modal behaviors from unstructured datasets. Second, using multiple primitives is suitable for the playbook extension. Since primitives are independent, we can utilize new primitives without affecting existing primitives to improve its expressive capacity.

The playbook can apply diverse methods, which combine multiple distributions using a weight vector. We use the MCP formulation [15] to present diverse and useful action distributions. The composite policy $\hat{\pi}$ is defined as follows:

$$\hat{\pi}(a|s,\hat{w}) = \frac{1}{Z(s,\hat{w})} \prod_{m=1}^{M} \pi_{\phi_m}(a|s)^{\hat{w}[m]}, \quad \hat{w}[m] \geq 0, \tag{1}$$

where $\hat{w}$ is an M-dimensional weight vector, $\hat{w}[m]$ is the m-th element of $\hat{w}$, and $Z(s,\hat{w})$ is a partition function for the normalization of the composite policy. MCP derives the composite policy as a normal Gaussian distribution by modeling each primitive as a normal distribution [15].

B. Play Selection through Embedding Model

To facilitate the extension of the playbook, the playbook does not select weight vectors for primitives from a continuous space but rather chooses from a set of predefined plays. The playbook has N plays, and each play $\hat{w}$ is a learnable positive real vector with M dimensions, i.e., $\hat{w} \in \mathbb{R}^M_{\geq 0}$. The playbook selects plays using a state-action embedding model, which is parameterized by θ. First, a state and an action sequence are encoded separately and then projected into a raw vector $w \in \mathbb{R}^M$. Next, using the vector quantization technique [27], w is replaced by the closest play $\hat{w}$ among N plays $\{\hat{w}_1, \cdots, \hat{w}_N\}$ as follows:

$$\text{quantization}(w) = \hat{w}_{id(w)}, \quad id(w) = \underset{i \in \{1,2,\cdots,N\}}{argmin} \|w - \hat{w}_i\|_2. \tag{2}$$

Since we select a play index, we can consider the play as a discrete variable. In other words, each play is expressed as an integer, which indicates the play index.

Finally, we train the playbook using the following loss:

$$\begin{aligned} \mathscr{L}_{Base} = \mathop{\mathbb{E}}_{\tau \sim \mathscr{D}, t, w_t \sim p_\theta(w|\tau)} \Big[& -\log(\hat{\pi}(a_t|s_t, \hat{w}_{id(w_t)})) \\ & + c_1 \|\mathbf{sg}[w_t] - \hat{w}_{id(w_t)}\|_2^2 + c_2 \|w_t - \mathbf{sg}[\hat{w}_{id(w_t)}]\|_2^2 \Big], \end{aligned} \tag{3}$$

where $\mathscr{D}$ is the given dataset, **sg** is the stop-gradient operator, and c_1 and c_2 are constants. The first term maximizes the likelihood of the composite policy, while the other terms encourage raw vectors and plays to get close.

IV. Playbook Extension by Class-Incremental Learning

When new datasets are given, we can accommodate them through structural extension of the playbook. By adding new plays and primitives to the trained playbook, its expressive power is enhanced, and the range of possible actions increases to cover new datasets. We assume that new datasets for the playbook extension are sequentially given. Due to memory limitations, we cannot store all previous data, so the dataset used for training is left with only a small amount.

To cover new datasets efficiently, we reuse existing plays while training new plays. For example, reusing the play that reaches a specific position or object can be helpful because it is frequently used for various manipulation tasks. During the training for new datasets, the extended playbook chooses between existing and new plays, which can be interpreted as a class-incremental learning problem for play selection. However, training the expanded playbook may face catastrophic forgetting, risking the loss of previously learned plays. To mitigate catastrophic forgetting, we apply the gradient boosting technique for class-incremental learning inspired by FOSTER [17]. FOSTER fixes previously trained models and adds new parameterized models. Then, FOSTER trains additional models to cover the new dataset while maintaining the output of the original model for the remaining dataset. After training additional models, FOSTER compresses the extended model to the original model size through knowledge distillation. Since the state-action embedding model of the playbook selects plays, we apply FOSTER for the embedding model.

We extend the playbook in the following order. First, we freeze a pre-trained playbook consisting of a state-action embedding model and multiple plays and primitives. Next, we add a new parameterized embedding model and the fixed number of plays and primitives. Then, we define an extended embedding model that adds the output of the new and existing embedding models. Finally, the extended embedding model and additional plays and primitives are trained to minimize loss (3) for the new dataset. By training newly added plays and primitives, the expressive power of the playbook over the raw action space can be improved. Since we have the fixed original playbook, the extended embedding model can choose previously learned plays or train new plays when learning new tasks. After training the extended model, the embedding model is reduced to its original network size through knowledge distillation of FOSTER. In summary, when the playbook extension is completed, the size of the embedding model is maintained, and the number of plays and primitives increases. We refer to the above process of extending a playbook as a *continual play learning*.

V. Sequential Play Plan Using a Playbook

In this section, we assume a scenario where a goal state is given, and we infer an action sequence to reach the goal using a trained playbook, i.e., plays and primitives. As explained in Section III, the action distribution generated by the playbook changes solely based on the selected play index. Therefore, by planning a sequential order of play indices to achieve the goal, we can generate the corresponding actions.

In general, we can utilize all planning methods applicable in the discrete search space for the play plan. In this paper, we propose two sampling-based methods: beam search and Monte Carlo tree search (MCTS) planners. The beam search planner is suitable for a playbook trained with a single dataset, and the MCTS planner addresses an extended playbook. These two

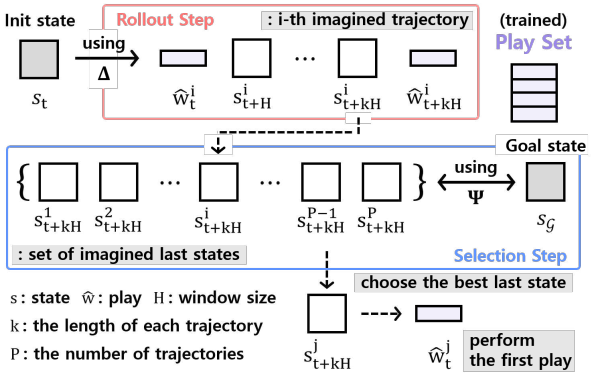


Fig. 2: Play plan using planning sets.

planners have a rollout step and a selection step, as shown in Figure 2. First, in the rollout step, we imagine several future state-play sequences from the current state using the trajectory generation model Δ. Next, in the selection step, we select the best imagined sequence that has reached the closest to the given goal state using the distance metric Ψ. We denote a planning set by $\{\Delta, \Psi\}$.

A. Play Plan Generation through Beam Search

We describe the beam search planner for a playbook trained with a single dataset. In the rollout step of beam search, we use the trajectory transformer (TT) [28] as a trajectory generation model Δ. The original TT learns the conditional probability distribution of state-action-reward sequences from the offline dataset. However, we only model state-play sequences because the task-agnostic dataset we use has no rewards. By utilizing TT trained with the same dataset, we imagine diverse and reasonable future state-play sequences conditioned on the current state.

In the selection step of beam search, we choose the best play sequence among the imagined sequences. To this end, we measure the dynamical distance in the state space between the last state of each sequence and a given goal state. We use the Q-value estimated by the goal-conditioned offline RL algorithm as a distance metric. In goal-conditioned RL, the agent estimates Q-value, $Q(s, s_g, a)$, which indicates the discounted sum of rewards that can be obtained in the future if action a is performed in the current state s given the goal state s_g. We can consider that a state with a higher Q-value reaches the goal in fewer time steps.

We use IQL [29], an offline RL algorithm, for a distance metric Ψ by modifying IQL to fit the goal-conditioned RL setting, i.e., all states are concatenated with the goal state as the input for all parameterized models. To train goal-conditioned IQL, we need not only current states, plays, and the next states that can be sampled from the dataset but also goal states and rewards that are not given. Hence, we use goal states by sampling the time step of the goal state as $t_G = t + \eta H$, which presents a time step after η plays are performed. H is a fixed window size, and $\eta \in \mathbb{N}$ is a random variable sampled along the geometric distribution. We use sparse rewards, which become 1 if the goal is reached within one play from the current state (i.e., $\eta = 1$) and 0 if not.

Finally, through beam search with a planning set, we generate a proper play sequence for a given goal state. We convert the first play of the sequence into raw actions using primitives and perform converted actions. Until the playbook achieves the goal, the above process is repeated.

B. Monte Carlo Tree Search for a Mixed-Play Plan

We explain the MCTS-based play planner for an extended playbook with several datasets. The MCTS planner, like the beam search planner, uses TT as the trajectory generation model and the Q-value of IQL as the distance metric. However, since the extended playbook is trained from multiple datasets, we train and retain separate planning sets, $\{\{\Delta^1, \Psi^1\}, \cdots, \{\Delta^P, \Psi^P\}\}$, from each of P datasets. Note that the extended playbook must maintain all planning sets for each dataset, which is a limitation of the playbook extension.

In the rollout step of the MCTS planner, the planner generates various state-play sequences through iterative tree searches. For each tree search, we repeatedly select one trajectory generation model Δ^i to infer the proper play and next state. As a result, we obtain a future state-play sequence from the current state. Once a tree search concludes, the Q-value of the final imagined state and play and the goal state is measured using Ψ^j, where j is a lastly selected planning set index and stored as the trajectory score.

In the selection step of the MCTS planner, the sequence with the highest trajectory score is selected after all tree searches are completed. Importantly, since the MCTS planner utilizes all planning sets, it can freely mix plays learned from different datasets in the extended playbook. Using a generated mixed-play plan, we can solve compounded problems that are mixtures of old and new tasks.

VI. Auxiliary Objective for Action Embedding

The playbook aims to cover multi-modal action distributions using a finite number of plays, as described in Section III. To achieve this, each play must generate independent and consistent actions. For instance, even in the same state, if different plays are selected, the inferred action sequences should perform different tasks. For effectively mapping diverse action sequences to play vectors, the action encoder of the state-action embedding model should extract the intention contained in each action sequence rather than simply projecting actions into a low-dimensional latent vector. In other words, different action sequences should map to the same play if they lead to similar future states. To this end, we propose the information bottleneck (IB)-based auxiliary objective for extracting intentions from raw actions.

The proposed IB objective is estimated using an action encoder, a latent mapping model and a state encoder-decoder pair, as depicted in Figure 3. First, a state s_{t+H} and an action sequence $a_{t:t+H}$ are embedded into state latent $z^{s_{t+H}}$ and action latent $z^{a_{t:t+H}}$ through different encoders, respectively. In particular, action latent implies an intention of an action sequence of fixed length. Next, we define the latent mapping model parameterized by ξ as $p_\xi(z^{s_{t+H}} | s_t, z^{a_{t:t+H}})$, which predicts the distribution of the future state latent as Gaussian using

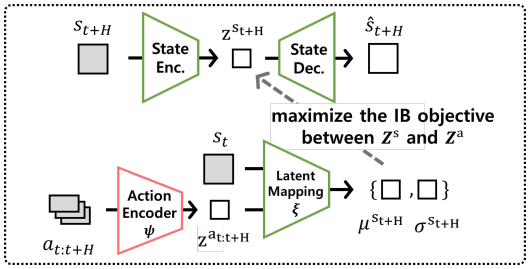


Fig. 3: The process of extracting intention from action sequences using the auxiliary IB objective. Green models are additional models for calculating the IB objective.

the current state and action latent as an input. The action encoder parameterized by ψ and the latent mapping model parameterized by ξ have the following IB objective between state and action latents:

$$\text{maximize } \mathbb{E}_t\left[\mathbb{I}(Z^{s_{t+H}};Z^{a_{t:t+H}}|S_t)-\beta\mathbb{I}(Z^{a_{t:t+H}};A_{t:t+H})\right], \tag{4}$$

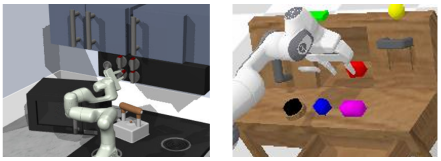
where S_t, $A_{t:t+H}$, Z^{s_t}, and $Z^{a_{t:t+H}}$ are random variables corresponding to s_t, $a_{t:t+H}$, z^{s_t}, and $z^{a_{t:t+H}}$, respectively, and β is a constant. The above IB objective is interpreted as follows. The first term, $\mathbb{I}(Z^{s_{t+H}};Z^{a_{t:t+H}}|S_t)$, means that given the state s_t, an action latent $z^{a_{t:t+H}}$ is informative about a state latent $z^{s_{t+H}}$. Next, the second term, $\mathbb{I}(Z^{a_{t:t+H}};A_{t:t+H})$, means that an action latent $z^{a_{t:t+H}}$ is penalized for preserving information about an action sequence $a_{t:t+H}$. That is, although $z^{a_{t:t+H}}$ is extracted from raw actions, since $z^{a_{t:t+H}}$ only has the minimum information to infer $z^{s_{t+H}}$, the meaning of each action is lost, and the intention of the action sequence remains. We derive the following lower bound of (4), which is used as the additional loss term for the playbook:

$$\begin{aligned}\mathscr{L}_{IB} = \\ -\mathop{\mathbb{E}}_{\tau,t,z^{s_{t+H}},z^{a_{t:t+H}}}\Big[&\log p_\xi(z^{s_{t+H}}|s_t,z^{a_{t:t+H}})-\log\mathbb{E}_{z^a}\left[p_\xi(z^{s_{t+H}}|s_t,z^a)\right]\\ &-\beta D_{KL}\big(p_\psi(Z^{a_{t:t+H}}|a_{t:t+H})\,\|\,q(Z^{a_{t:t+H}})\big)\Big].\end{aligned} \tag{5}$$

On the other hand, we train the state encoder and decoder independently using the β-VAE [30] loss, $\mathscr{L}_{VAE}$, to prevent the state encoder from falling into trivial solutions such as converging state latents to the zero vector. As a result, we train the playbook by minimizing the integrated loss $\mathscr{L} = \mathscr{L}_{Base} + c_3\mathscr{L}_{IB} + c_4\mathscr{L}_{VAE}$, where c_3 and c_4 are constants.

VII. Experiment

We conduct experiments to evaluate the playbook in complex and challenging environments. We focus on answering the following questions through experiments: (1) Can a finite number of plays cover a dataset consisting of task-agnostic demonstrations? In other words, can the playbook achieve better performance than existing methods? (2) Can the playbook extension successfully solve new tasks? (3) Can the extended playbook reuse previously learned plays when it solves new tasks? (4) How much do the IB objective and the MCP structure affect the performance of the playbook?



(a) Franka Kitchen (b) CALVIN

Fig. 4: Benchmarks used in experiments.

A. Experiment Setup

1) Benchmark: We evaluate the playbook in simulated environments, which are selected based on the following characteristics: (1) the capability to perform diverse tasks sequentially within a single workspace, (2) the availability of a publicly accessible offline dataset, and (3) the possibility of obtaining a goal observation. Consequently, we utilize the following two benchmarks, as shown in Figure 4.

Franka Kitchen [9] provides a kitchen workspace for manipulating various objects with a Franka robot, aiming to complete four predetermined sub-tasks consecutively. An observation is a 30-dimensional state representation, and an action is a 9-dimensional joint velocity vector for a robot.

CALVIN [18] is a benchmark for robotic manipulation tasks with a Franka robot on a desk. In this paper, we evaluate eight sub-tasks related to a drawer, a slider, an LED, and a light bulb. We utilize an offline dataset in environment D of CALVIN and do not use task labels. An observation is a $3\times64\times64$-dimensional RGB image, and an action is a 7-dimensional robot action.

2) Baselines: **Offline RL algorithms**. We utilize CQL [31], IQL [29], and TT [28] with IQL as offline RL baselines for Franka Kitchen and CALVIN environments. We implement CQL and IQL and report their measured performance. On the other hand, we use the officially published code for TT. Particularly, TT+IQL is a method used for the play plan, allowing us to verify the performance difference between using raw actions and plays.

Hierarchical policy learning methods.

We utilize Play-LMP [10], RIL [9], TACO-RL [11], OPAL [5], SkiMo [2], and XSkill [20] as hierarchical policy learning baselines for the CALVIN environment. For Play-LMP, RIL, and TACO-RL, we measure performance using publicly available checkpoints trained on the CALVIN benchmark. In contrast, for OPAL, SkiMo, and XSkill, we train the skill embedding model using the CALVIN dataset. Since OPAL has no available code, we have implemented the training code while using the officially released code for SkiMo and XSkill. Also, we train IQL, described in Section V-A as the goal-conditioned skill policy for OPAL, SkiMo, and XSkill.

B. Performance Comparison Experiment

1) Franka Kitchen Result: The playbook utilizes 32 plays and 16 primitives to cover the offline dataset of Franka Kitchen. Table I summarizes the performance results in Franka Kitchen. The results are an average success of 50 episodes with three random seeds. Since the Franka Kitchen benchmark provides sparse rewards, the cumulative reward signifies

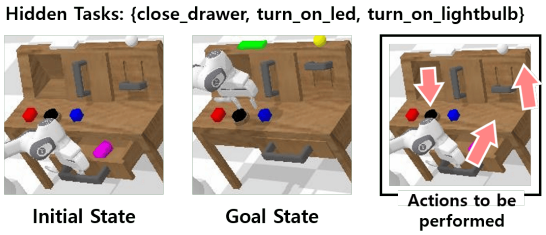


Fig. 5: Example of three sub-task chain problem in CALVIN.

Algorithm	Kitchen-Partial	Kitchen-Mixed
CQL	1.81 ± 0.18 (1.99)	1.64 ± 0.26 (2.04)
IQL	1.90 ± 0.27 (1.85)	1.82 ± 0.28 (2.04)
TT+IQL	1.84 ± 0.37	1.92 ± 0.16
Playbook	**2.32 ± 0.42**	**2.50 ± 0.20**

TABLE I: Performance results in Franka Kitchen. Each mean and standard deviation of the cumulative reward are calculated over 50 episodes with three random seeds. Numbers in parentheses are the results reported in the cited papers.

the number of completed sub-tasks. Note that the goal of Franka Kitchen is to perform four predetermined sub-tasks. On average, all offline RL algorithms only succeed in less than two sub-tasks, but the playbook completes more than two sub-tasks. In summary, the playbook outperforms offline RL baselines for all dataset types.

2) CALVIN Result: We conduct experiments in CALVIN requiring two or three sub-tasks to be performed sequentially to reach a given goal state, as depicted in Figure 5. It is challenging because we provide the agent with only one goal image, and the goal can only be achieved when all sub-tasks are completed. Therefore, the agent should consider both which tasks to perform and the order of tasks.

The playbook utilizes 64 plays and 32 primitives for the CALVIN benchmark. Table II shows the performance results in CALVIN, and we use the same trained model for solving two and three sub-task chain problems for each algorithm. The results are an average success rate of 1000 scenarios with three random seeds. First, offline RL methods show low performance, which means that using raw actions is disadvantageous for addressing long-horizon problems. On the other hand, hierarchical policy-based methods perform better, but on average, even one sub-task cannot be completed. In contrast, the playbook performs the best and succeeds in more than half of all sub-tasks, representing that using plays effectively solves long-horizon problems.

For both benchmarks, we infer play sequences using a beam search planner described in Section V-A. The window size is set to 10, meaning each play generates 10 raw actions. We generate 64 future state-play trajectories, each containing eight play vectors, i.e., the planner predicts states after 80 timesteps. On average, the beam search planner takes 0.11 seconds to infer a single raw action using an RTX 4080 GPU.

C. Playbook Extension

In this experiment, we extend the trained playbook and evaluate it using complex image scenarios in the CALVIN benchmark. To accomplish this, we extract demonstrations of four predetermined sub-tasks (*close drawer*, *move slider left*, *turn on LED*, and *turn on lightbulb*) from the offline dataset of environment D of CALVIN. Consequently, we have five independent datasets: four task datasets, each involving the trajectories of a single task, and one base dataset containing data from other tasks. First, we train the initial playbook using the base dataset. And then, we progressively extend the playbook using four task datasets sequentially. We remove the previously used data for the playbook training, retaining only a ratio of 1%. The initial playbook has 64 plays and 32 primitives, and we add four plays and two primitives to enrich the expressive capability when extending the playbook.

We incrementally extend a playbook by incorporating sub-tasks in the order of *close drawer*, *move slider left*, *turn on LED*, and *turn on lightbulb*. Therefore, four continual play learning steps are required in total, so we obtain five trained models, including the initial playbook. For those five models, the success rates of 25 scenarios for eight sub-tasks are shown in Table III. We find the best play plan via MCTS, as explained in Section V-B. The results show that the extended playbook maintains the success rates of previously learned tasks and solves new tasks through repeated continual play learning. Finally, the final model of the playbook successfully performs all eight sub-tasks.

Furthermore, we evaluate the extended playbook for compounded tasks, which are a mixture of old and new sub-tasks in CALVIN. We generate compounded tasks that combine two sub-tasks among eight sub-tasks in Table IV, i.e., four sub-tasks on the left side and four sub-tasks on the right side of Table IV become old and new tasks, respectively. Also, we use the MCTS planner to find the proper mixed-play plan. We experiment with the final model, which has completed all continual play learning steps. Table IV shows the success rates of sequential two sub-tasks over 100 scenarios. The playbook achieves a success rate of 24.4% for two sub-task chain problems, which indicates that the extended playbook can generate proper mixed-play plans for compounded tasks.

We plan play sequences using a MCTS-based planner described in Section V-B. We infer 32 future state-play trajectories through tree search. On average, the MCTS-based planner takes 0.23 seconds to infer a single raw action using an RTX 4080 GPU.

D. Analysis of Play Reuse Ratio

To robustly and efficiently solve new tasks, it is important not only to learn new skills but also to reuse existing skills. Therefore, we analyze the selection rates of old and new plays when solving new tasks using the extended playbook. We use the final model that has completed all continual learning steps in Section VII-C and experiment in 25 episodes per task. The results of reuse ratios are shown in Figure 6. On average, when solving newly learned tasks, the agent chooses old plays at a rate of 28.8%, which means that old plays are useful in solving new tasks. Note that in the case of *turn on LED*, since *turn off LED* and *turn on LED* are performed by similar action sequences due to the desk structure of CALVIN, it is reasonable to use old plays more than new ones.

Sub-Tasks	CQL	IQL	TT+IQL	Play-LMP	RIL	TACO-RL	OPAL+IQL	SkiMo+IQL	XSkill+IQL	Playbook
First Sub-Task	0.143 ± 0.035	0.198 ± 0.041	0.402 ± 0.119	0.427	0.678	0.414	0.734 ± 0.074	0.594 ± 0.212	0.261 ± 0.014	**0.866 ± 0.021**
Second Sub-Task	0.000 ± 0.000	0.000 ± 0.000	0.042 ± 0.015	0.039	0.221	0.165	0.350 ± 0.118	0.226 ± 0.183	0.008 ± 0.005	**0.508 ± 0.037**
Average Length	0.143	0.198	0.444	0.466	0.899	0.579	1.084	0.820	0.269	**1.374**

(a) Success rates for two sub-tasks chain problems

Sub-Tasks	CQL	IQL	TT+IQL	Play-LMP	RIL	TACO-RL	OPAL+IQL	SkiMo+IQL	XSkill+IQL	Playbook
First Sub-Task	0.108 ± 0.021	0.135 ± 0.023	0.372 ± 0.057	0.400	0.701	0.213	0.741 ± 0.084	0.629 ± 0.230	0.304 ± 0.035	**0.901 ± 0.011**
Second Sub-Task	0.000 ± 0.000	0.000 ± 0.000	0.082 ± 0.019	0.029	0.254	0.028	0.311 ± 0.105	0.233 ± 0.193	0.018 ± 0.001	**0.563 ± 0.027**
Third Sub-Task	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000	0.013	0.000	0.084 ± 0.059	0.067 ± 0.070	0.000 ± 0.000	**0.214 ± 0.021**
Average Length	0.108	0.135	0.454	0.429	0.968	0.241	1.136	0.929	0.322	**1.678**

(b) Success rates for three sub-tasks chain problems

TABLE II: Results for sub-task chains in CALVIN. Each mean and standard deviation of success rates are calculated over 1,000 scenarios with three random seeds. The average length indicates the average number of completed sub-tasks.

Model	Open Drawer	Move Slider Right	Turn off LED	Turn off Lightbulb	Close Drawer	Move Slider Left	Turn on LED	Turn on Lightbulb	Average
Init	1.00 ± 0.00	0.96 ± 0.04	0.79 ± 0.02	0.95 ± 0.02	0.00 ± 0.00	0.05 ± 0.02	0.03 ± 0.05	0.04 ± 0.07	0.48
Step 1	0.97 ± 0.02	0.96 ± 0.04	0.83 ± 0.06	0.89 ± 0.02	0.87 ± 0.02	0.05 ± 0.02	0.12 ± 0.11	0.05 ± 0.02	0.59
Step 2	0.97 ± 0.05	0.96 ± 0.04	0.77 ± 0.02	0.92 ± 0.04	0.88 ± 0.04	0.76 ± 0.04	0.08 ± 0.00	0.07 ± 0.02	0.68
Step 3	0.99 ± 0.02	0.96 ± 0.04	0.61 ± 0.02	0.91 ± 0.02	0.87 ± 0.02	0.76 ± 0.07	0.63 ± 0.12	0.04 ± 0.04	0.72
Final	0.99 ± 0.02	0.93 ± 0.06	0.60 ± 0.16	0.93 ± 0.06	0.77 ± 0.02	0.73 ± 0.06	0.56 ± 0.18	0.64 ± 0.28	**0.77**

TABLE III: The success rate of the extended playbook for eight sub-tasks in CALVIN. We highlight the cell if the corresponding task dataset is not used for learning of each model. Each mean and standard deviation of success rate are averaged over 25 scenarios with three random seeds.

Algorithm	Success Rate of First Sub-Task	Success Rate of Second Sub-Task	Average Length
(Extended) Playbook	0.659 ± 0.058	0.244 ± 0.025	0.903

TABLE IV: The success rate of two sub-task chains among eight tasks in CALVIN using an extended playbook. Each mean and standard deviation of success rates are calculated over 100 scenarios with three random seeds. The average length indicates the average number of completed sub-tasks.

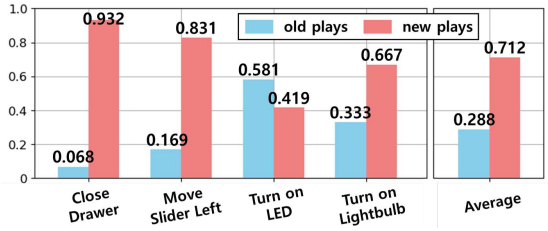


Fig. 6: The play selection ratio for four newly learned tasks in CALVIN. Blue and red bars represent the selection ratio of old and new plays, respectively, when solving each task.

E. Ablation Study

We conduct an ablation study for the playbook on CALVIN. Under the same experiment setting as three sub-task chains in Section VII-B2, we identify the effect of the IB objective and MCP structure on the playbook performance. The first baseline algorithm, *playbook-*α, maximizes only the first term of (4). In other words, the objective becomes the mutual information, excluding the information penalty term for action sequences. Next, the second algorithm, *playbook-*β, does not use the IB objective, and action sequences are simply encoded through the action encoder. The third algorithm, *playbook-*γ, uses non-learnable one-hot vectors as plays to find out the effectiveness of MCP. In other words, the composite policy of the playbook-γ becomes one primitive, not a combination of primitives. The fourth algorithm, *playbook-*δ, forms the composite policy through a normalized linear combination of primitives rather than an exponential combination, i.e., it becomes a Gaussian mixture model. The last algorithm, *playbook+BC*, selects plays using a goal-conditioned behavioral cloning model instead of the planning set. We measure average success rates of 1000 scenarios with three random seeds for all algorithms. As a result, the original playbook shows the best performance of 1.678, as shown in Table V.

VIII. Limitations

Remaining all planning sets for mixed-play plans. As mentioned in Section V-B, we must preserve planning sets for all datasets in order to generate mixed-play plans using an extended playbook. Since planning sets are composed of offline RL algorithms, this limitation can be solved through continual RL for planning sets and will be addressed in our future work for the playbook.

Trade-off between the number of plays and planning efficiency. The playbook has a trade-off between the number of plays and the efficiency of the play plan. As the number of plays increases, the ability of the playbook to cover multi-modal actions improves. But as the complexity of the play plan increases, the required computational cost increases and the efficiency decreases. Conversely, as the number of plays decreases, the play plan becomes simple, but the expressive power of the playbook decreases. Therefore, the playbook must explore the appropriate number of plays adaptively.

Sub-Tasks	playbook	playbook-α	playbook-β	playbook-γ	playbook-δ	playbook+BC
First Sub-task	**0.901 ± 0.011**	0.874 ± 0.012	0.882 ± 0.013	0.745 ± 0.174	0.878 ± 0.018	0.726 ± 0.018
Second Sub-task	**0.563 ± 0.027**	0.434 ± 0.006	0.477 ± 0.013	0.079 ± 0.013	0.533 ± 0.033	0.319 ± 0.026
Third Sub-task	**0.214 ± 0.021**	0.105 ± 0.008	0.154 ± 0.019	0.001 ± 0.001	0.152 ± 0.012	0.070 ± 0.050
Average Length	**1.678**	1.413	1.513	0.825	1.563	1.116

TABLE V: Performance results of the ablation study for three sub-tasks chain problems. Each mean and standard deviation of success rates are calculated over 1,000 scenarios with three random seed The average length indicates the average number of completed sub-tasks.

IX. Conclusion

In this paper, we introduced the playbook, a scalable method for skill discovery that addresses the challenge of learning new skills without forgetting existing ones. By retaining only a small subset of previous datasets, the playbook expands structurally with new plays and primitives, enhancing its action diversity. This approach enables the playbook to efficiently accommodate new datasets, treating play selection as a sequential classification problem managed through class incremental learning. Experimentally, we have confirmed that the playbook with a finite number of plays performs better than existing skill discovery methods. In addition, we have verified that the extended playbook not only successfully solves tasks included in new datasets but also performs compounded tasks, which are a combination of old and new tasks.

References

[1] K. Pertsch, Y. Lee, and J. Lim, "Accelerating reinforcement learning with learned skill priors," in *Proc. of the Conference on Robot Learning (CoRL)*, Dec. 2020.

[2] L. X. Shi, J. Lim, and Y. Lee, "Skill-based model-based reinforcement learning," in *Proc. of the Conference on Robot Learning (CoRL)*, Dec. 2022.

[3] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised discovery of skills," in *Proc. of the International Conference on Learning Representations (ICLR)*, Apr. 2020.

[4] J. Kim, S. Park, and G. Kim, "Unsupervised skill discovery with bottleneck option learning," in *Proc. of the International Conference on Machine Learning (ICML)*, Nov. 2021.

[5] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum, "OPAL: Offline primitive discovery for accelerating offline reinforcement learning," in *Proc. of the International Conference on Learning Representations (ICLR)*, May. 2021.

[6] M. Hong, M. Kang, and S. Oh, "Diffused task-agnostic milestone planner," in *Proc. of the Neural Information Processing Systems (NeurIPS)*, Dec. 2024.

[7] Z. Jiang, J. Gao, and J. Chen, "Unsupervised skill discovery via recurrent skill training," in *Proc. of the Neural Information Processing Systems (NeurIPS)*, Dec. 2022.

[8] P. Mazzaglia, T. Verbelen, B. Dhoedt, A. Lacoste, and S. Rajeswar, "Choreographer: Learning and adapting skills in imagination," in *Proc. of the International Conference on Learning Representations (ICLR)*, May. 2023.

[9] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Proc. of the Conference on Robot Learning (CoRL)*, Dec. 2019.

[10] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Proc. of the Conference on Robot Learning (CoRL)*, Dec. 2019.

[11] E. Rosete-Beas, O. Mees, G. Kalweit, J. Boedecker, and W. Burgard, "Latent plans for task-agnostic offline reinforcement learning," in *Proc. of the Conference on Robot Learning (CoRL)*, Dec. 2022.

[12] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[13] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, Jun. 2016.

[14] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[15] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine, "MCP: Learning composable hierarchical control with multiplicative compositional policies," in *Proc. of the Neural Information Processing Systems (NeurIPS)*, Dec. 2019.

[16] S. Park, K. Lee, Y. Lee, and P. Abbeel, "Controllability-aware unsupervised skill discovery," in *Proc. of the International Conference on Machine Learning (ICML)*, Nov. 2023.

[17] F.-Y. Wang, D.-W. Zhou, H.-J. Ye, and D.-C. Zhan, "FOSTER: Feature boosting and compression for class-incremental learning," in *Proc. of the European Conference on Computer Vision (ECCV)*, Oct. 2022.

[18] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 3, pp. 7327–7334, Jun. 2022.

[19] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine, "Parrot: Data-driven behavioral priors for reinforcement learning," in *Proc. of the International Conference on Learning Representations (ICLR)*, May. 2021.

[20] M. Xu, Z. Xu, C. Chi, M. Veloso, and S. Song, "XSkill: Cross embodiment skill discovery," in *Proc. of the Conference on Robot Learning (CoRL)*, Nov. 2023.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[23] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, Jun. 2016.

[24] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, Mar. 2015.

[25] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. of the Neural Information Processing Systems (NeuIPS)*, Dec. 2017.

[26] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, Oct. 2018.

[27] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Proc. of the Neural Information Processing Systems (NeurIPS)*, Dec. 2017.

[28] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *Proc. of the Neural Information Processing Systems (NeurIPS)*, Dec. 2021.

[29] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit Q-learning," in *Proc. of the International Conference on Learning Representations (ICLR)*, Apr. 2022.

[30] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. of the International Conference on Learning Representations (ICLR)*, Apr. 2017.

[31] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in *Proc. of the Neural Information Processing Systems (NeurIPS)*, Dec. 2020.