2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Leveraging the efficiency of multi-task robot manipulation via task-evoked planner and reinforcement learning

Haofu Qian[1,2], Haoyang Zhang[1,2], Jun Shao[1,2], Jiatao Zhang[1,2], Jason Gu[2], Wei Song[1,2✉] and Shiqiang Zhu[1✉]

***Abstract*— Multi-task learning has expanded the boundaries of robotic manipulation, enabling the execution of increasingly complex tasks. However, policies learned through reinforcement learning exhibit limited generalization and narrow distributions, which restrict their effectiveness in multi-task training. Addressing the challenge of obtaining policies with generalization and stability represents a non-trivial problem. To tackle this issue, we propose a planning-guided reinforcement learning method. It leverages a task-evoked planner(TEP) and a reinforcement learning approach with planner's guidance. TEP utilizes reusable samples as the source, with the aim of learning reachability information across different task scenarios. Then in reinforcement learning, TEP assesses and guides the Actor towards better outputs and smoothly enhances the performance in multi-task benchmarks. We evaluate this approach within the Meta-World framework and compare it with prior works in terms of learning efficiency and effectiveness. Depending on experimental results, our method has more efficiency, higher success rates, and demonstrates more realistic behavior.**

I. Introduction

In the field of manipulation, tasks often demand continuous control, and progress has been achieved across various scenarios, including opening caps, handwriting, assembly, and manipulating deformable objects [1]–[6]. Researchers have manually specified observation, action, and reward functions [7], [8], guiding the agents through iterative exploration until they establish a stable and effective model [8], [9]. However, in this approach, the action distribution of the model remains narrow, and the transition probabilities are relatively limited.

Previously noted, the specific settings encompassing observation composition, state dimensions, and reward functions confine the scope of skill definition and application, posing challenges when attempting to generalize their effectiveness to different situations. In reality, robots often need to possess a significant number of versatile skills [9] that can perform precise operations across various scenarios and tasks. To enable the widespread application of robots in real-world, reinforcement learning algorithms are needed capable of efficiently acquiring a wide range of skills throughout one learning progress. Such requirements face the difficulty of dealing with various behavioral patterns and more intricate transition models, which obstruct agents achieving multiple tasks. Realizing robust generalization across a diverse task set within a structurally unified policy network currently stands a bottleneck in research. Multi-task reinforcement learning, which leverages parameter sharing and reuse, is considered a promising approach to tackle the issue [10]–[16]. Nevertheless, it also faces problems such as imbalanced action distributions, training instability, and incomplete transfer probabilities [17], [18].

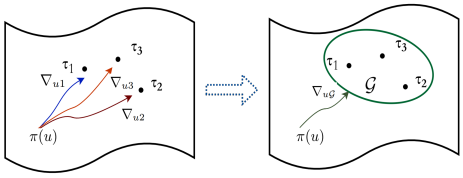

Fig. 1. Faced with a diverse range of tasks, policies tend to diverge and transfer negatively at a certain point. The introduction of a unified target space is anticipated to lead to improvements.

In the meantime, recent research has explored the use of planning-guided reinforcement learning, which select different output modes based on factors such as the length of the visual horizon or the transferring of modeling methods [10], [15], [19]–[23]. These approaches have demonstrated promising results in robot motion planning and indoor navigation, and have the capability to accommodate the inclusion of new tasks to the existing task set. Such methods are proved highly valuable in the initial stages of reinforcement learning, as they facilitate effective exploration of the state space, providing a substantial of exploration samples, and contribute to stabilizing the training process. This offers a novel approach for addressing the issues in multi-task learning mentioned above.

In this work, we introduce the TEP (**T**ask-**e**voked **P**lanner) and the TERL (**T**ask-**e**voked Planner with **R**einforcement **L**earning) algorithm to address policy conflicts between different tasks and mitigate negative transfer during the learning process and endeavor to establish a shared action space across multiple tasks, encompassing the states and actions relevant to accomplishing these tasks. The method combines the strengths of planning and multi-task learning by leveraging historical reusable information when learning new tasks. It guides the agent to explore a shared goal space, reducing behavioral divergence when unnecessary, as shown in Fig.

*This work was supported by National Natural Science Foundation of China(Grant No. U21A20488)[1] and Key Research Project of Zhejiang Lab (Grant No. G2021NB0AL03)[2].

[1] Zhejiang University, Hangzhou 310030, China
[2] Research Center for Intelligent Robotics,Research Institute of Interdisciplinary Innovation, Zhejiang Lab, Hangzhou 311100,China
✉ Correspondence: Wei Song(weisong@zhejianglab.com) and Shiqiang Zhu(zhusq@zhejianglab.com)

1. Specifically, TEP is a motion planner trained on samples with high task-agnostic priority and provides reachability information between two spatial positions. Subsequently, we incorporate this planner into the multi-task reinforcement learning framework, where the policy network and the planner simultaneously receive inputs consisting of the current state and a one-hot encoded vector representing the task ID. The planner evaluates the attainability of the objective by considering the reachability between the current state and the goal state and gives the alternative actions when the policy network's decision proved to be inefficient. Through this approach, we preserve reusable samples from previous tasks and transfer them when learning new tasks, thereby enhancing training efficiency and stability through various tasks. We evaluate the proposed approach in the Meta-World environment, which involves simultaneous learning of up to 50 tasks, and achieve competitive performance.

II. RELATED WORK

A. Multi-task learning

Multi-task learning has received significant attention in computer vision [24], [25], NLP [28], [29], bioinformatics [26], [27], and health informatics. In recent years, with the development of robot vision and speech technologies, multi-task learning in robotics has also become a research hotspot [10]–[16], [30], [31]. While sharing parameters at the lower levels among different models has shown some success in task expansion, it comes with the challenge of task interference and negative transfer [11], [14], [32]–[35]. Distillation methods [34], [35] have been effective in avoiding these issues, but they often significantly increase the size and complexity of the network. Approaches like curriculum learning or iterative learning [17], [36] attempt to strike a balance between these factors but may not perform optimally when expanding the task set. A combination [10] of planning and reinforcement learning has been applied successfully in motion and path planning, but it faces greater challenges in multi-task planning because of the diversity and complexity of the manipulation. Therefore, in this paper, we introduce the TEP (Task-evoked Planner) to gather shared experiences among relevant tasks and further to guide and improve training for rapid acquisition of new skills.

B. Reinforcement learning with planning

The current challenge lies in how to integrate the strengths of motion planning and reinforcement learning. One approach is to partition the problem artificially, allowing planning and RL to address different aspects [10], [19], [22], [23]. However, this method requires significant engineering efforts to decompose tasks, limiting its scalability to new tasks. Another approach involves incorporating motion planning into RL frameworks in a modular structure, forming HRL (Hierarchical Reinforcement Learning), and designing corresponding boundaries and transition rules [37], [38]. Such methods are often used for tasks with clear target objects and low motion complexity. In addition, there are methods involving heuristic weights [39], policy distillation with motion planner augmentation [40], and suboptimal solution mapping in abstract environments [41], among others.

III. PRELIMINARIES

A. Reinforcement learning

Our approach builds upon a modeled system of Markov decision processes in reinforcement learning, which is represented by a tuple $\langle S, A, P, R, \rho, \gamma \rangle$, where S and A are the state and action spaces, respectively. Reinforcement learning algorithms aim to learn a policy $\pi(a|s)$ such that the expected sum of rewards $R(s_t, a_t)$ obtained under this policy is maximized, starting from an initial state distribution ρ and following the transition probabilities $P(s_{(t+1)}|s_t; a_t)$. The discount factor γ is used to balance the trade-off between immediate and future rewards. Our objective is maximizing the expected discounted accumulated rewards, which is given by:

$$\theta = argmax \left(\mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \right) \tag{1}$$

B. Soft Actor-Critic

Soft Actor-Critic (SAC) [5] is a representative offline policy learning algorithm that encourages exploration and uniform sampling by adding a component of behavior entropy to the cumulative reward. It learns by alternately optimizing a value network $Q_\varphi(s, a)$ and a policy network $\pi_\theta(a|s)$, as metioned before. φ and θ represent parameters of Q network and policy network, respectively. The soft Q function can be learned by minimizing the soft Bellman residuals,

$$\begin{aligned} J_Q(\varphi) = \mathbb{E}_{(s,,a_i)-\mathcal{R}}[\tfrac{1}{2}(Q_\varphi(s_t, a_t) \\ -(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}}[V_{\bar{p}}(s_{t+1})]))^2], \end{aligned} \tag{2}$$

where

$$V_{\bar{\phi}}(s_t) = \mathbb{E}_{\pi_{\bar{\theta}}}\left[(Q_{\bar{p}}(s_t, a_t) - \alpha \log \pi_\theta(a_t \mid s_t)) \right] \tag{3}$$

and $J_Q(\varphi)$ is the objective function of the Q network. $Q_{\bar{\phi}}$ denotes the target Q network with parameter $\bar{\varphi}$ which is derived by the exponential moving average of φ in the Q network with a smoothing constant τ. $\mathcal{R}$ is a replay buffer. The policy function π_θ can be obtained by minimizing the following loss function,

$$J_\pi(\theta) = \mathbb{E}_{s_t-\mathcal{R}}\left[\mathbb{E}_{\mathrm{a}_t \sim \pi_\theta}\left[\alpha \log \pi_\theta(a_t \mid s_t) - Q_\varphi(s_t, a_t) \right] \right]. \tag{4}$$

IV. MULTI-TASK REINFORCEMENT LEARNING WITH TASK-EVOKED PLANNER

In multi-task learning, tasks are uniformly sampled from the task set $\mathcal{D}$, and the objective is to maximize the average return across all tasks. However, due to the misalignment of task-specific goal spaces, this can lead to policy distribution fluctuations during training. Notably these fluctuations are often more pronounced in the early stages of training and can subsequently impact the final model's performance.

Based on these challenges, we propose TEP(Task-evoked Planner) and aim to address the following questions:

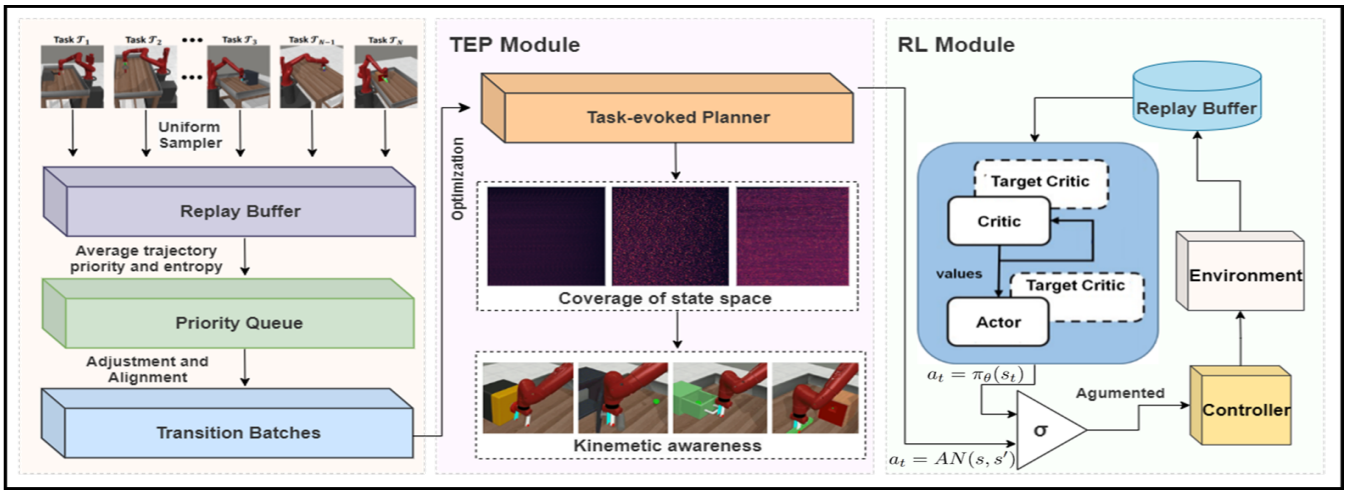

Fig. 2. Semantic Illustration of the framework.

1)How to extract valuable task-agnostic samples from the stable training of individual tasks.

2)How to utilize a planner to extract and transform these samples into a usable form.

3)How to effectively incorporate the planner into the multi-task reinforcement learning framework to fully leverage its sample knowledge.

A. Sample selection via priority

In Meta-World, the training of individual tasks is not particularly challenging. We initiate the sample selection process when the training task achieves a success rate of 85%. Our selection method is based on prioritized trajectory replay (PTR) [42], primarily utilizing the criteria of the average trajectory priority with the generalized advantage estimation (GAE) and the distribution entropy. Optimization is performed per batch, with each consisting of 512 samples. The definition of the average trajectory priority A_j is as follows:

$$p_\tau^i = \text{mean}\left\{|A_j|\right\}_\tau^i \tag{5}$$

$$A_j = \delta_j + (\gamma\lambda)\delta_{j+1} + \cdots + (\gamma\lambda)^{T-j-1}\delta_{T-1} \tag{6}$$

$$\delta_t = r_t + \gamma V\left(s_{t+1}\right) - V\left(s_t\right) \tag{7}$$

where A_j is General Advantage Estimation based on multiple continuous experience. The trajectory $\tau_i = (s_i, a_i, r_i, s_{i+1}, p_i)$ encompasses multiple steps of experience. γ and λ are hyperparameters, which is 0.99 and 0.95, respectly. The entropy item is defined as:

$$p_{E(i)} = -\int_\pi p(s|a)\log p(s|a)da \tag{8}$$

After balancing with factor $\alpha = 0.6$, the sample priority becomes:

$$P(i) = \alpha p_\tau^i + (1-\alpha)p_{E(i)} \tag{9}$$

B. Task-evoked Planner

Our approach was inspired by Eysenbach [41], [43], who introduced a method for assessing distance in the state space. Expanding on the approach and taking into account the unique attributes of robotic tasks, we introduce TEP and develop a novel evaluation method for estimating the reachability information between states. In essence, the design of the planner adheres to three guiding principles: first, the distance between states of two consecutive samples is set to 1; second, for two consecutive samples with the same goal, the difference in distance between their current states and the goal state should be 1; finally, if two states are swapped, the distance should remain unchanged. The transformation of the planner is as follows:

$$\mathcal{F}: \mathcal{S}^2 \times \mathcal{A} \to \mathbb{R}_+ \tag{10}$$

in application, we will employ $\mathcal{F}(s, s', a)$ to denote the reachability from s to s' when starting with action a, expressed as the required number of steps.

According to the priciples outlined above, given a set of samples $\mathcal{Z}$, the planner should optimize itself from three aspects:

$$\begin{gathered}\left(\mathcal{F}\left(s_t, s_{t+1}, a_t\right) - 1\right)^2 + \\ \left(\mathcal{F}\left(s_t, s', a_t\right) - \left(1 + \min_a \tilde{\mathcal{F}}\left(s_{t+1}, s', a\right)\right)\right)^2 + \\ \left(\mathcal{F}(s_t, s_{t+1}, a_t) - \mathcal{F}(s_{t+1}, s_t, -a_t)\right)^2\end{gathered} \tag{11}$$

Using the above section as the loss function and training it with previous experience sampled in Sec. IV.A will yield a planner that has details about the world models. However, when it needs to be applied to a reinforcement learning, the challenge is that we cannot predict the next state s' when we obtain the Actor's output. Without the necessary input data(s_{t+1}), we cannot make the best use of this planner. Therefore, we further establish an Action Network (AN) for TEP to tranfser the relation between (s_t, s_{t+1}) to (s_t, a). The detailed network structure will be provided in Sec. V.A. The role of AN is to provide the optimal action solution from

the current state to the target state based on the reachability information from the planner, represented as follows:

$$\begin{gathered} AN_{\theta_a}: \mathcal{S}^2 \rightarrow \mathcal{A} \\ AN_{\theta_a}(s, s') \simeq \operatorname{argmin}_a \mathcal{F}_{\theta_{\mathcal{F}}}(s, s', a) \end{gathered} \tag{12}$$

For training, given a batch of tuples (s_t, s_{t+1}, s', a_t) we update TEP's parameter θ_f to reduce

$$\begin{gathered} \mathcal{L}\left(\theta_{\mathcal{F}}; s_t, s_{t+1}, s', a_t, \tilde{\theta}_{\mathcal{F}}\right) = \\ (\mathcal{F}_{\theta_{\mathcal{F}}}(s_t, s_{t+1}, a_t) - 1)^2 + (\mathcal{F}_{\theta_{\mathcal{F}}}(s_t, s', a_t) - \\ \left(1 + \mathcal{F}_{\tilde{\theta}_{\mathcal{F}}}(s_{t+1}, s', AN_{\tilde{\theta}_a}(s_{t+1}, s'))\right))^2 - \\ \left(\mathcal{F}(s_t, s_{t+1}, a_t) - \mathcal{F}(s_{t+1}, s_t, -a_t)\right)^2 \end{gathered} \tag{13}$$

and meanwhile update AN's parameter θ_a to reduce

$$\mathcal{L}(\theta_a) = \mathcal{F}_{\theta_{\mathcal{F}}}(s_t, s', AN_{\theta_a}(s_t, s')) \tag{14}$$

In summary, after training, TEP has obtained precise information regarding the environmental kinematic model. It can guide the agent's actions by determining the necessary number of steps between the current state and the goal state. Moreover, in situations where the agent's decisions are inefficient, TEP has the capability to propose alternative actions.

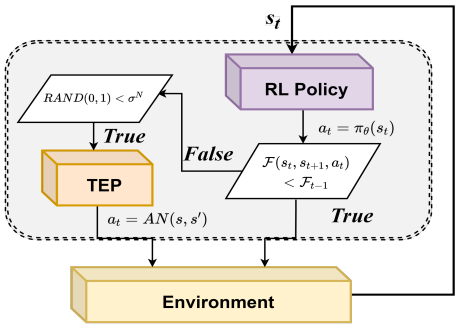

Fig. 3. TEP extended RL policy.

C. Reinforcement learning with TEP

TEP-guided Policy In order to efficiently learn manipulation tasks in a multi-task environment and minimize negative interference between diverse tasks, we propose TERL (Task-evoked Sample guided Reinforcement Learning). Our approach leverages a motion planner to guide the robot in efficient movement towards its goal, avoids infeasible areas in the environment and facilitates effective exploration. As illustrated in Fig. 3, our framework consists of two components: a SAC algorithm with task encoding and the proposed planner, TEP. Decisions about what actions the agent should take are made by comparing reachability before generating a new transition, as detailed below:

$$\text{Reachability} \begin{cases} a_t = AN(s, s'), \text{ if } & \mathcal{F}(s_t, s_{t+1}, a_t) > \mathcal{F}_{t-1} \\ & RAND(0,1) < \sigma^N \\ a_t = \pi_\theta(s_t), & \text{otherwise} \end{cases} \tag{15}$$

$\mathcal{F}(s_t, s_{t+1}, a_t) < \mathcal{F}_{t-1}$ represents the reachability to the goal after the execution of the current action, while f_{t-1} represents the distance to the target when the action is not executed. σ denotes the discount factor, which gradually reduces the guidance provided by the planner. N signifies the number of episodes completed. If the distance decreases after the action is executed, the output of the policy network remains unchanged. Otherwise, a randomly generated number $g \in (0, 1)$ is checked against a threshold value σ^N. If it satisfies the condition, the action recommended by the planner is used. This process is repeated until the completion of an episode. During the training process, σ^N gradually decreases, leading to a synchronized reduction in the probability that g is less than σ^N.

Multi-task Reinforcement Learning with TEP In the context of multi-task reinforcement learning (MRL), we examine a task set denoted as $\mathcal{D} = \{\mathcal{T}_i\}_{i=1}^N$, and assume a uniform distribution $p(\mathcal{T})$ over this task set $\mathcal{D}$. Each task $\mathcal{T}_i$ is associated with a distinct Markov Decision Process (MDP) represented as $M_i = (\mathcal{S}, \mathcal{A}, \mathcal{P}_i, r_i, \rho_i, \gamma)$, capturing variations in reward functions r_i and transition probabilities $\mathcal{P}_i$ among the tasks. Every task $p(\mathcal{T})$ sampled from the distribution $p(\mathcal{T})$ possesses a maximization objective defined as the expected discounted cumulative rewards, denoted as $J(\pi, \mathcal{T})$. The primary objective of multi-task RL is to acquire a policy that maximizes the overall expected returns across the tasks, expressed as $\mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})}\left[J_{SAC}(\pi, \mathcal{T})\right]$.

Furthermore, compared to other methods [11], [44], which an individual soft Q-function $Q_i^\pi(s, a)$ for each task $\mathcal{T}_i$ based on Eq. 2, we adopt TEP which confront these variations, as the kinematic model in space remains largely unchanged across different tasks when not interacting with objects. And due to the involvement of planned actions, samples in the state space often avoid to exhibit significant discrepancies from the Q-value network distribution thus demonstrate a certain degree of robustness to in off-policy algorithms.

V. EXPERIMENTS

A. Benchmarks and Baselines

Benchmarks Our chosen benchmarks are as follows: 1)MT1-Ind: A meta-reinforcement learning (meta-RL) benchmark environment designed to evaluate few-shot adaptation to goal variation within a single task. We have selected five classical robot tasks for this benchmark, including door opening, button pressing, picking and placing, window opening, and sweeping into a hole. 2)MT10: A meta-RL benchmark consisting of ten meta-train tasks that assess few-shot adaptation. This benchmark includes a one-hot vector for task identification. 3)MT-Test: A testing set built upon the MT10 benchmark, comprising four new tasks. 4)MT20: Similar in design to MT10 but with tasks incresing to 20. All tasks are comprehensively detailed within [44].

Baselines We have conducted comparisons between our proposed method and the following baseline methods: 1)Multi-Task SAC(MTSAC) [44]: All tasks share a single policy, with task identification accomplished through a single-head encoding and the concatenation of the current

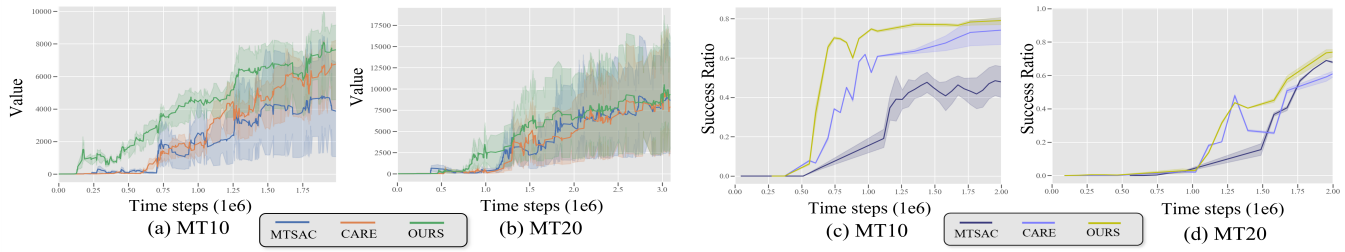

Fig. 4. Experimental Results with MTSAC and CARE.

state vector as input. 2)Context-Aware Reinforcement Learning with Shared Contextual Encoding (CARE) [11]: Utilizes the block contextual Markov decision process (BC-MDP) to address scenarios where the state space may differ among tasks, also the SOAT algorithm in MT10.

Environment Setup In simulation experiments, a task set $\mathcal{D}$ was initialized, from which we select 1, 10 or 20 tasks for learning, depending on different configurations. Each task within the set has its unique reward function and world model, while sharing the same action dimension. Regarding state dimensions, there are slight variations among different tasks. For tasks that exclusively involve movement, the state space requires only target position information, historical data, and robot information. However, for tasks involving rotation, the robot not only needs aforementioned conditions but also the orientation of the target to ensure the effectiveness of its actions.

TEP itself employs a multi-layer perceptron (MLP) with a ReLU activation function, consisting of $2d_s + d_a$ input units, corresponding to the concatenation of two states and an action. It comprises three hidden layers, each containing 512 units, and a single output unit. The AN (Action Network) component has $2d_s$ input units, three hidden layers with 512 units each, and d_a output units with a tanh non-linear activation function. In Meta-World d_s is 39 and d_a is 4. We sampled a buffer of $6 * 10^6$ transitions and performed TEP's optimization on mini-batches of size 1024.

We adopted the same network architecture for SAC, with a discount factor σ set to 0.98 and the number of steps in one episode was set to 4096. Initially, we sample tasks from those with a training success rate of 85% or higher. Because of their stable action distribution and transition probabilities, collecting effective samples becomes straightforward. Building upon this, the acquired planner serves as an additional critic in reinforcement learning, providing guidance for policy decisions and enhancing interactions with the environment to achieve improved behavior.

B. Evaluation Tasks in simulation

In this section, we utilize Task-evoked planner to evaluate the performance of robots on a variety of robotic manipulation tasks within the framework of multi-task learning. To start, we compare the proposed approach against prior works [11], [44] and the conventional benchmark using various metrics, which include cumulative rewards(Value), Success Ratio, Zero-shot Transfer performance to new tasks, and qualitative examples of different strategies.

TABLE I
AVERAGE SUCCESS RATIO OF MT1, MT10, MT-TEST AND MT20 TASKS

Algorithms	MT1-Ind	MT10	MT-Test	MT20
MTSAC	95.0%	68.4%	16.4%	63.4%
CARE	95.2%	75.6%	**23.4%**	64.6%
Ours	**95.4%**	**82.8%**	19.4%	**68.8%**

We employ the MT1-Ind, MT10, MT-Test and MT20 benchmarks for comparative analysis against baseline algorithms. Fig. 4 depicts the evolution of cumulative rewards (value) and success rates throughout the experiments. Tab. 1 presents the average successful rate across different algorithms. The success rate, refers to the average success rate of all tasks in a given task set, with each task uniformly sampled. Our experiments were conducted with five different seeds and reported average success ratio based on 500 trials as a standard. Training samples for MT10 and MT20 vary between 2 million and 2.5 million for different tasks.For a fair comparison, we mostly adhere to previous work [11] regarding hyper parameters and general model settings. The weighting factors τ, γ, lr_{act}, lr_{cri} and τ_{encode} are set to 0.005, 0.99, 3e-4, 3e-4, and 0.05, respectively.

Fig. 4(a), (c) and Tab. 1 show the results on MT10, which affirm the enhancements brought about by our method. As depicted, multi-task learning facilitated by Task-evoked Planners (TEP) yields superior strategies, exhibiting improved performance both in the initial and later stages of the reward curve, when compared to the baselines. Notably, the involvement of TEP contributes to a heightened task completion rate and from the data presented in Tab. 1, our approach has led to an increase of up to 14% in success rates over the other two methods.

Fig. 4(b)(d) further underscore our method's superior performance. Despite the inherent challenges associated with the MT20 task, characterized by its more complex and diverse nature, our approach continues to optimize the learning process. Through the average rewards in this demanding task exhibit significant variance throughout training, our approach consistently maintains a lead in success rate and ultimately leads by up to 4 percentage points.

In addition, the MT-Test demonstrates the performance in terms of Zero-shot Transfer, while the MT1-Ind showcases

the ability to share experiences among related tasks. Incorporating tasks into MT-Test that differ in action types from the existing task set leads to compromised performance due to the absence of relevant samples in TEP.

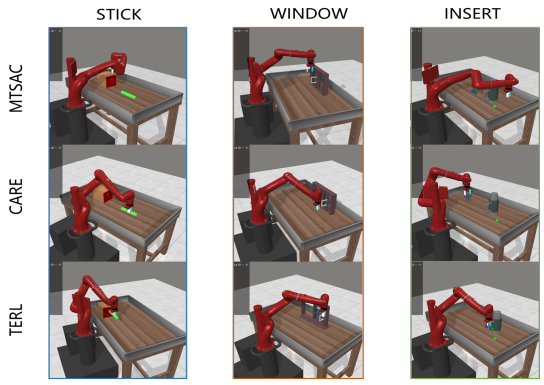

Fig. 5. **Qualitative examples.** Left: We show selected episodes for the stick-insert task. We see that our TERL policy (bottom) is considerably more realistic to the state-action CARE policy (middle) while being marginally more realistic than the pure RL policy (top).

Qualitative examples Relying solely on values or success rates cannot provide a complete view of the policy. A selection of episodes has been chosen in Fig. 5 for the purpose of demonstrating the differences among the learned policies. Firstly, MTSAC sometimes encounters the operational constraint boundaries of the robot due to insufficiently realistic actions, thereby impeding its ability to accomplish tasks(top). For instance, in the STICK task, it collides with the box, in the WINDOW task, it collides with the window, and in the INSERT task, it reaches the joint angle limit. CARE exhibits some realistic actions, but still has certain deviations when interacting with objects(middle). Consequently, in the STICK task, it lacks accuracy in grasping the stick, in the WINDOW task, it uses its joints instead of the end-effector to manipulate the handle, and in the INSERT task, it grasps the bracket off-center, leading to a loss of horizontal balance. Lastly, the TERL strategy(ours) outperforms the other two and displays a higher degree of realism (bottom).

C. Real world applications

Furthermore, the existence of intelligent robots capable of performing a variety of tasks using a single model should not be confined solely to simulation but extend to interactions within the physical world. Real world experiments are needed. Mask-RCNN was utilized for object instance segmentation, acquiring the pose and spatial position of target objects. Subsequently, through the retrieval of robot body positioning and joint information, we obtained relative positioning with respect to the target objects through which our policy can be executed. Fig. 6 illustrates partial results of the manipulation. When confronted with variations in physical parameters(object size, weight, and friction coefficients) between real-world and simulation, our model consistently generates effective manipulation sequences.

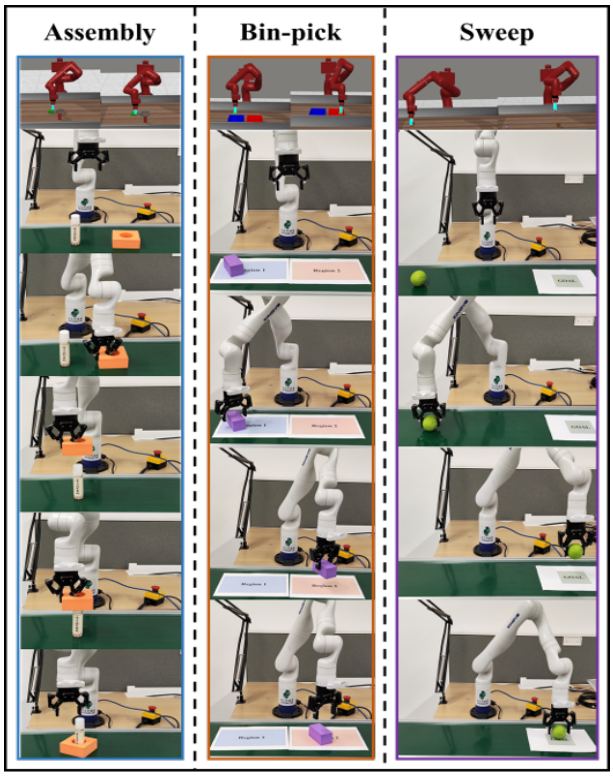

Fig. 6. Evaluate our policy on various tasks in real world.

VI. CONCLUSION

In this work, we introduce TEP to optimize policy's narrow distribution and action's generalization capability in multi-task environments and harness its advantages with reinforcement learning to achieve more efficient and realistic manipulation. Specifically, we employ a hybrid prioritized sample selection approach, devise objectives for planner updates, and propose a planning-guided reinforcement learning algorithm. Our method minimizes the need for task-specific knowledge and effectively leverages the utility of historical samples. Experimental results demonstrate the enhanced training efficiency of multi-task learning and the improved manipulation performance. The effectiveness of the framework in real-world scenarios has also been validated. The limitations is the additional time and space required for training the planner separately. In the future, we plan to integrate the planner into the RL framework for simultaneous training and develop more general planning methods adapted to more complex skills.

REFERENCES

[1] Gu, Shixiang, et al. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates." 2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017.

[2] Rajeswaran, Aravind, et al. "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations." arXiv preprint arXiv:1709.10087 (2017).

[3] Chen, Tao, Jie Xu, and Pulkit Agrawal. "A system for general in-hand object re-orientation." Conference on Robot Learning. PMLR, 2022.

[4] Qin, Yuzhe, et al. "DexPoint: Generalizable Point Cloud Reinforcement Learning for Sim-to-Real Dexterous Manipulation." arXiv preprint arXiv:2211.09423 (2022).

[5] Wu, Yueh-Hua, Jiashun Wang, and Xiaolong Wang. "Learning generalizable dexterous manipulation from human grasp affordance." arXiv preprint arXiv:2204.02320 (2022).

[6] Meng, Qiwei, et al. "Kgnet: Knowledge-guided networks for category-level 6d object pose and size estimation." 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023.

[7] Jangir, Rishabh, Guillem Alenya, and Carme Torras. "Dynamic cloth manipulation with deep reinforcement learning." 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

[8] Singh, Avi, et al. "End-to-end robotic reinforcement learning without reward engineering." arXiv preprint arXiv:1904.07854 (2019).

[9] Groth, Oliver, et al. "Goal-conditioned end-to-end visuomotor control for versatile skill primitives." 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.

[10] Yamada, Jun, et al. "Motion planner augmented reinforcement learning for robot manipulation in obstructed environments." Conference on Robot Learning. PMLR, 2021.

[11] Sodhani, Shagun, Amy Zhang, and Joelle Pineau. "Multi-task reinforcement learning with context-based representations." International Conference on Machine Learning. PMLR, 2021.

[12] Micheli, Vincent, Karthigan Sinnathamby, and François Fleuret. "Multi-task reinforcement learning with a planning quasi-metric." arXiv preprint arXiv:2002.03240 (2020).

[13] Nasiriany, Soroush, Huihan Liu, and Yuke Zhu. "Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks." 2022 International Conference on Robotics and Automation (ICRA). IEEE, 2022.

[14] Angelov, Daniel, et al. "Composing diverse policies for temporally extended tasks." IEEE Robotics and Automation Letters 5.2 (2020): 2658-2665.

[15] Haarnoja, Tuomas, et al. "Soft actor-critic algorithms and applications." arXiv preprint arXiv:1812.05905 (2018).

[16] Zentner, K. R., et al. "Efficient multi-task learning via iterated single-task transfer." 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2022.

[17] Driess, Danny, et al. "Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning." 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

[18] Englert, Peter, and Marc Toussaint. "Learning manipulation skills from a single demonstration." The International Journal of Robotics Research 37.1 (2018): 137-154.

[19] Meng, Qiwei, et al. "Rffce: Residual feature fusion and confidence evaluation network for 6dof pose estimation." 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023.

[20] Singh, Satinder, et al. "Robust reinforcement learning in motion planning." Advances in neural information processing systems 6 (1993).

[21] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis. Learning navigation behaviors end-to-end with autorl. IEEE Robotics and Automation Letters, 4(2):2007–2014, 2019.

[22] Xia, Fei, et al. "Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation." arXiv preprint arXiv:2008.07792 (2020).

[23] Xia, Fei, et al. "Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation." arXiv preprint arXiv:2008.07792 (2020).

[24] M. Lapin, B. Schiele, and M. Hein, "Scalable multitask representation learning for scene classification," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2014, pp. 1343–1441

[25] M. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," in Proc. Int. Conf. Learn. Representations, 2016

[26] X. Lu, Y. Wang, X. Zhou, Z. Zhang, and Z. Ling, "Traffic sign recognition via multi-modal tree-structure embedded multitask learning," IEEE Trans. Intell. Transp. Syst., vol. 18, no. 4, pp. 960–972, Apr. 2017.

[27] J. Xu, P. Tan, J. Zhou, and L. Luo, "Online multi-task learning framework for ensemble forecasting," IEEE Trans. Knowl. Data Eng., vol. 29, no. 6, pp. 1268–1280, Jun. 2017.

[28] A. H. Abdulnabi, G. Wang, J. Lu, and K. Jia, "Multi-task CNN model for attribute prediction," IEEE Trans. Multimedia, vol. 17, no. 11, pp. 1949–1959, Nov. 2015.

[29] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King, "Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2015, pp. 4460–4464.

[30] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In International Conference on Learning Representations, 2018.

[31] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. arXiv preprint arXiv:2003.13661, 2020.

[32] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4238– 4245. IEEE, 2018.

[33] Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In 2017 IEEE international conference on robotics and automation (ICRA), pp. 2161– 2168. IEEE, 2017.

[34] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. arXiv preprint arXiv:1511.06295, 2015.

[35] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In Advances in Neural Information Processing Systems, pp. 4496–4506, 2017.

[36] Huang, Hanchi, et al. "Curriculum-based asymmetric multi-task reinforcement learning." IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

[37] Gieselmann, Robert, and Florian T. Pokorny. "Planning-augmented hierarchical reinforcement learning." IEEE Robotics and Automation Letters 6.3 (2021): 5097-5104

[38] Angelov, Daniel, et al. "Composing diverse policies for temporally extended tasks." IEEE Robotics and Automation Letters 5.2 (2020): 2658-2665..

[39] Cao, Yuxue, et al. "Reinforcement learning with prior policy guidance for motion planning of dual-arm free-floating space robot." Aerospace Science and Technology 136 (2023): 108098.

[40] Liu, I-Chun Arthur, et al. "Distilling motion planner augmented policies into visual control policies for robot manipulation." Conference on Robot Learning. PMLR, 2022.

[41] Eysenbach, Ben, Russ R. Salakhutdinov, and Sergey Levine. "Search on the replay buffer: Bridging planning and reinforcement learning." Advances in Neural Information Processing Systems 32 (2019).

[42] Liang, Xingxing, et al. "Ptr-ppo: Proximal policy optimization with prioritized trajectory replay." arXiv preprint arXiv:2112.03798 (2021).

[43] Micheli, Vincent, Karthigan Sinnathamby, and François Fleuret. "Multi-task reinforcement learning with a planning quasi-metric." arXiv preprint arXiv:2002.03240 (2020).

[44] Yu, Tianhe, et al. "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning." Conference on robot learning. PMLR, 2020.