Task-Oriented Adaptive Learning of Robot Manipulation Skills

KeXin Jin[1] ,GuoHui Tian[2],Member,IEEE,Bin Huang[2],YongCheng Cui[3],XiaoYu Zheng[4]

***Abstract*— The working environment and manipulation tasks of robots are always dynamically changing. For robots to learn skills from scratch based on new task commands is a slow process that relies heavily on human assistance. Therefore, developing a mechanism that can adapt to different manipulation tasks and autonomously and quickly learn new skills can effectively address the issue of low efficiency in robot skill learning and enhance the robot's adaptive capabilities. This paper proposes a general Intelligent Transfer System (ITS) that enables robots to learn skills rapidly and autonomously in dynamic tasks. ITS integrates Large Language Models (LLMs) with transfer reinforcement learning, leveraging LLMs' intelligence and prior skills knowledge to expedite the learning of new skills. It is capable of comprehending new and previously unseen task conmmands and automatically generating a process-oriented reward function for each of these tasks, which enables the autonomous learning of new skills while eliminating the need to design hierarchical sub-processes for complex tasks. In addition, an Intelligent Transfer Network (ITN) is designed within ITS to extract knowledge of relevant skills in order to accelerate the learning of new ones. We systematically evaluate our method in simulation environment. The results demonstrate that our method improves the time efficiency of two major tasks by 72.22% and 65.17%, respectively, compared with learning from scratch. Additionally, we compare our method with several outstanding works in the field. For videos, please visit our project website:*https://jkx-yy.github.io/**

I. INTRODUCTION

The operations of robots in the real world adjusts continuously due to dynamic changes in the environment and tasks. Traditional pre-programmed behavior planning and decision-making methods struggle to adapt to these changes [1]. Although reinforcement learning can enable robots to learn new skills, subtle changes in the environment or objects often necessitate retraining from scratch, requiring much repetition [2]. Transfer learning leverages relevant task experience to improve the efficiency of autonomous learning for new tasks, significantly reducing repetitive learning burdens [3]. Nevertheless, designing an effective, automated, and generalizable transfer framework for unknown new tasks in long-term robot manipulation remains a significant challenge.

Some works have effectively enhanced the learning of robot skills in specific settings. For complex manipulation tasks, Hierarchical Reinforcement Learning (HRL) employs high-level strategies to reuse underlying strategies for subtasks, enhancing the learning process by leveraging previously learned skills [4]. However, these reused methods cannot be adapted to new, unknown tasks. Recent work combines HRL with imitation learning to tackle vision-based long-field-of-view robot manipulation [5], but it suffers from offline dataset limitations and requires task-specific meta-controllers. Another approach uses expert demonstrations to guide strategy search [6], accelerating learning in sparse reward environments. However, limited or low-quality demonstration data can reduce the effectiveness of these methods. In summary, existing skill enhancement techniques have high requirements for pre-training strategies, are sensitive to environmental changes, and depend on human involvement. They are designed for specific tasks, lacking the general ability for cross-task transfer, and they fail to autonomously learn new tasks that involve previously unlearned skills.

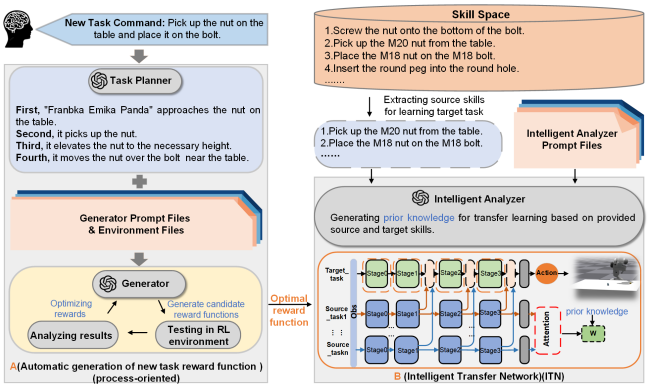


Fig. 1. Overview of ITS. The Intelligent Transfer System (ITS) is divided into two parts, In A, the generator analyzes the task and environment to generate the optimal reward function, In B, an Intelligent Transfer Network (ITN) is designed to accelerate the skill learning by fusing the features from different source skills.

Robots learning new skills heavily depends on human-designed reward functions, which hinders autonomous learning. Although human experts can design reward functions for simple skills, they struggle with complex tasks. LLMs excel in semantic planning and code generation due to their powerful computational, analytical, and memory capabilities. Extensive experiments show that LLMs outperform humans in generating reward functions [7]. However, this study doesn't address designing general reward functions for sequential complex tasks that may occur under dynamic tasks. A task-adaptive reward generator is crucial for enabling autonomous learning of new skills and improving task adaptability. Additionally, using LLMs' common sense knowledge to guide skill transfer learning in robots is an

This work is supported by National Natural Science Foundation of China under Grant(U22A2059,62273203) ,in part by the National Key R&D Program of China under Grant 2018YFB1307101, and in part by the Taishan Scholars Program of Shandong Province under Grant ts201511005. *(Corresponding author: Guohui Tian.)*

All the authors are affiliated with the School of Control Science and Engineering, Shandong University, Jinan, 250061, China. (email:kx.jin@mail.sdu.edu.cn; g.h.tian@sdu.edu.cn; huangbin@sdu.edu.cn; cuiyc@mail.sdu.edu.cn;202334997@mail.sdu.edu.cn)

industry concern that remains unexplored.

To enable robots to rapidly and autonomously learn new skills in dynamic environments, a task-oriented intelligent skills transfer system is developed (Figure 1). An automatically generated process-oriented reward function enhances the robot's ability to autonomously solve new tasks. A dynamically expandable skill space stores learned skills and assigns them to target tasks based on semantic information. An intelligent knowledge-guided ITN is formed by integrating transfer reinforcement learning with LLMs, utilizing source skills to accelerate the learning of target tasks.

We summarize four key contributions of this work:

(1) To assist robots in autonomous skill learning for new tasks, an innovative and generalized ITS is proposed that leverages the code generation, problem analysis, and planning capabilities of LLMs.

(2) To avoid manually designing separate reward functions for each subtask within a complex task, an LLM-driven generator is developed to automatically create a comprehensive reward function for the operation process.

(3) To fully utilize empirical knowledge, an intelligent analyzer dominated by LLMs is developed. This analyzer is used to analyze the contribution relationship matrix between source and target tasks, providing empirical knowledge for the attention module.

(4) To enable the system to possess general cross-task transfer capabilities and accelerate learning of new skills, an ITN is proposed, focusing on extracting shared skill features to adapt to various manipulation tasks.

II. RELATED WORK

A. Reinforcement Learning in Robot Operations

A reinforcement learning task can be described as a tuple $\langle S, A, P, R, \gamma \rangle$, $S = s_0 \times s_1 \times \cdots \times s_n$ is the set of environment and agent states. $A = a_0 \times a_1 \times \cdots \times a_n$ is the set of all possible actions, representing manipulations the agent can perform. $R = r_0, r_1 \cdots, r_n$ is the agent's reward function, where $R(s,a)$ is the immediate reward for action a in state s. The reward function provides feedback, guiding the agent to learn beneficial actions. The design of the reward function is crucial for the robot to master new skills. $\gamma \in [0,1]$ is the discount factor for current and future rewards. The goal of reinforcement learning is to learn an optimal policy $\pi^*(a|s)$ to maximize cumulative rewards. Extensive research exists on using reinforcement learning for machine manipulation tasks. [8] employd a graph neural network to represent states and actions during the assembly process, optimizing the strategy for collaborative human-machine assembly using a reinforcement learning algorithm. [9] investigated meta-reinforcement learning to solve simulated industrial insertion tasks and quickly adapt policies in real-world scenarios. [10] and [11] applied reinforcement learning to stacking and grasping tasks.

B. Transfer Learning in Robot Skills

Transfer learning is systematically presented and summarized in [1]. The purpose of transfer learning is to use the agent's source domain policy $\pi = \pi_1, \cdots, \pi_n$ to aid in learning the target strategy. Typical methods address the transfer problem between source and target domains effectively. For instance, [12] [13] used source domain knowledge for action generation through strategy reuse or distillation. [14] [15] [16] improved the training process of the target task by decoupling or reusing task-invariant features. Recent studies have explored various ways to combine transfer learning with reinforcement learning to address real-world robotics problems. [17] proposed a method for reconfiguring robot assembly control using Equivalent Compliance Theory (ECT) combined with Weighted Dimensional Policy Distillation (WDPD). [18] employed feature selection and adaptive transfer learning to generalize robot peg-in-hole assembly skills across different geometric features. However, these approaches primarily focus on generalizing parts within the same manipulation tasks and do not address cross-task transfer between different types of tasks. In contrast, our aim is to efficiently handle various classes of tasks that evolve over time.

C. Application of LLMs in Robots

LLMs, with their comprehensive knowledge reserves and powerful utilization and generation capabilities, are increasingly being applied in the field of robotics. [19] utilized ChatGPT for problem solving in robot operations to train a reliable agent RobotGPT significantly improved the task success rate. [20] utilized LLMs for fine-grained task planning and improved the efficiency and effectiveness of task planning. [21] reduced the sample complexity of reinforcement learning in robotics tasks by prompt engineering to extract prior knowledge from LLMs. [22] demonstrated the great potential of LLMs in robotics applications. In this paper, we leverage the task planning, problem analysis, and code generation capabilities of LLMs to design planner, analyzer, and generator, thereby replacing human intervention and powering the automatic implementation of ITS.

III. METHOD

The model schematic of the ITS is illustrated in Figure 1. It comprises two modules: The reward function automatic generation module and the intelligent transfer module.

A. Automatic Generation of New Task Reward

When robots learn complex tasks, traditional hierarchical reinforcement learning methods decompose these tasks into subtasks, designing separate reward for each, which presents a challenge in ensuring connectivity among subtasks. We propose a generalized reward prompt framework for LLMs to generate a comprehensive reward function for complex tasks, thereby accommodating task variations. Overall, we design the following three stages for generating an optimal reward function.

1) Planning Stage: When task requirements change, the robot receives a new, concise task command. However, this initial command is often unclear. To obtain a complete and accurate description of the task, we design a task planner

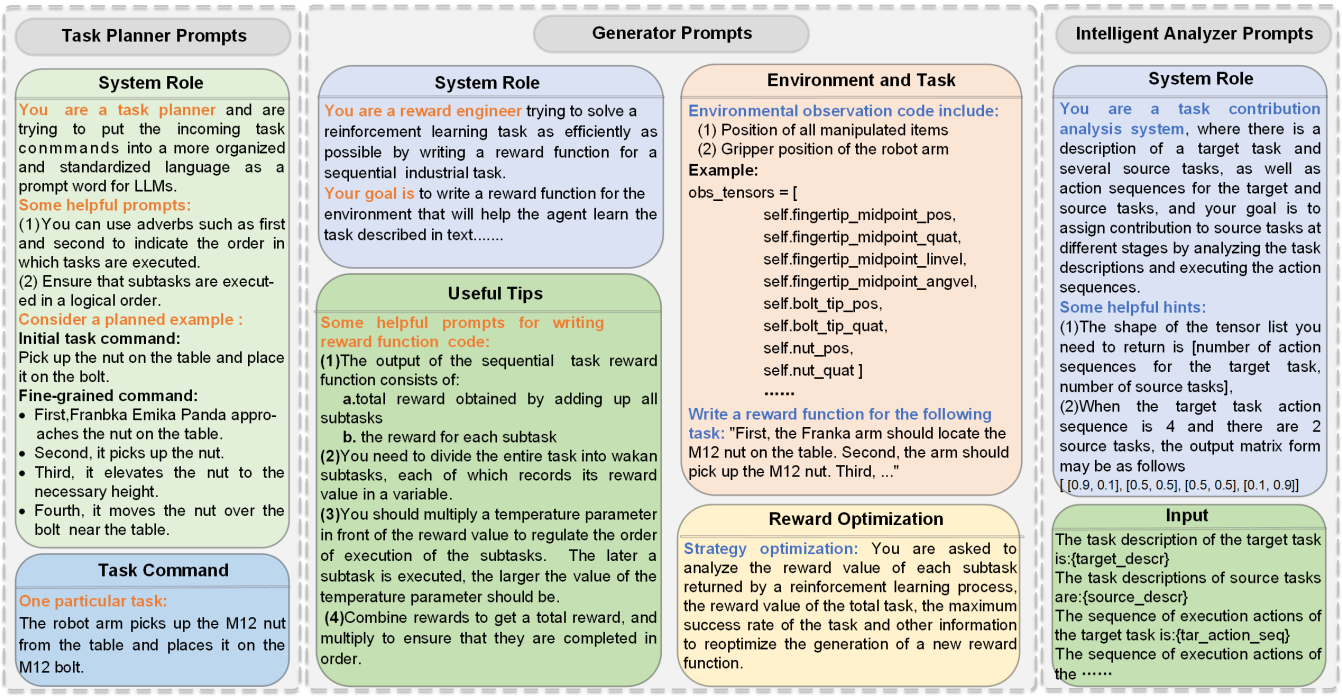


Fig. 2. Prompts for Task Planner, Generator, and Intelligent Analyzer. Task Planner refines user commands. Generator manages reward function design. Intelligent Analyzer enables LLMs to analyze source task contributions.

capable of analyzing the execution order of subtasks and generating a fine-grained task description. This description comprises a sequence of subtasks with adverbial modifiers such as ”First,” ”Second,” and so on. Figure 2 illustrates the LLMs prompts used in task planner .

2) Generation Stage: An effective prompting approach can significantly enhance the speed of generating executable reward function and the success rate of task execution. We design a effective four-part prompts, as shown in Figure 2, the generator prompts. The system information describes the system's identity and goals. Environment information comprises environmental observation data and the positional information of manipulated objects and robotic arm. The task description uses a fine-grained description generated by the task planner. Useful tips includes the proposed form of the reward function, as well as some constraint prompts we provide to ensure that the generated reward function can guide the sequential execution of complex task. The strategy optimization information includes several analyzable metrics for generating improved reward functions. Our prompts are comprehensively designed and received a satisfactory response.

3) Reward Optimization Stage: In each iteration, the LLMs generates multiple reward function candidates based on the sample size and regenerates them if the initial generation fails. To ensure the generated reward function correctly guides the robot to complete complex skill sequentially, we record the maximum task success rate, per subskill reward, and total reward for complex skill in each iteration for each reward candidate. The LLMs then adjust the temperature parameter of each subskill to generate a better reward function in the next iteration based on these data. Through this self-optimization process, the success rate and accuracy of reward function are continuously improved. Our method produces reward function with high interpretability and a higher success rate compared to human-designed ones.

B. Skill Space Construction and Source Skills Extraction

To accelerate learning new tasks, features are extracted from useful source tasks and store source tasks in a real-time expandable skill space that encompasses both basic and complex skills learned by the robot. Basic skills such as Pick, Place, Screw, and Insert are used to manipulate various mechanical parts. Sequential complex skills are designed as assembly task containing multiple subskills. Each skill in the skill space consists of four attributes: serial number, task description, pre-trained model, and task execution action sequence.

Methods such as MK-MMD [23] select source tasks similar to the target task by comparing data distributions. However, these methods face high computational complexity in large skill space and difficulty in kernel function selection. Considering that the source skill can be positively transferred to the target task if the target task command has a high similarity with the task command of the source skill, we design a skill extraction method based on SBERT [24], which extracts source skills similar to the target task at the semantic level. This method is robust, tends to extract deeper source skills, and is highly interpretable compared to other methods.

C. Intelligent Transfer Network (ITN)

In this section, we demonstrate the implementation of cross-task skill transfer as shown in Figure 3. First, we

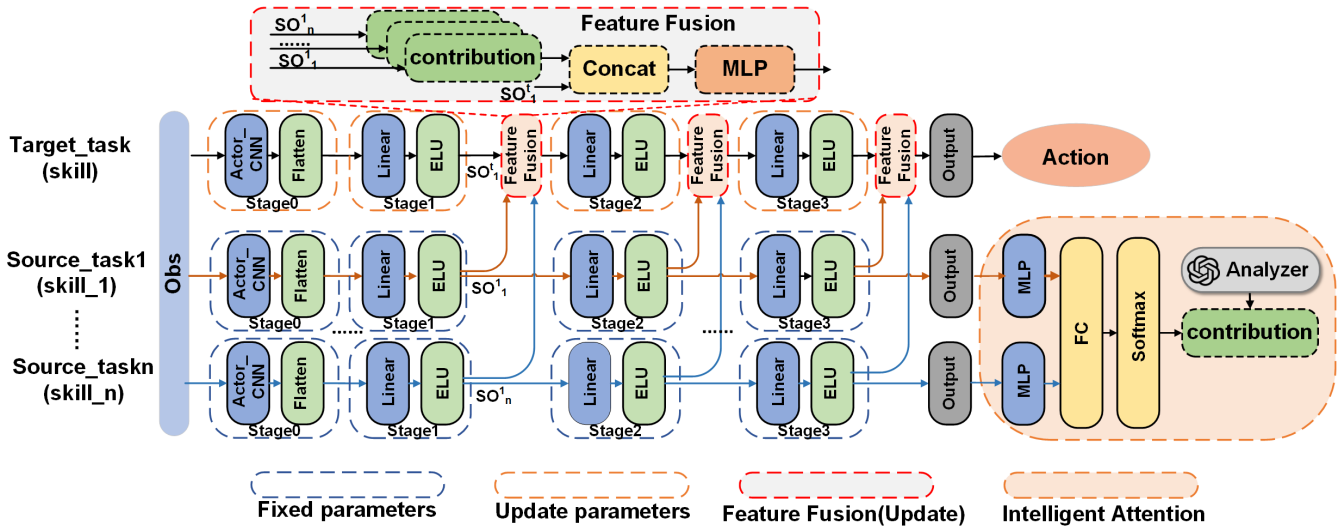


Fig. 3. Overview of the ITN. ITN connects the models of source tasks horizontally. The Intelligent Attention module takes the strategies of the source tasks as input and outputs the final contribution values of each subtask in combination with the contribution matrix of the Analyzer. The Feature Fusion modelu fuses features from each source task. The final output of the network is the strategy for the new task.

describe the generation of prior knowledge based on the intelligent analyzer. Next, we integrate the intelligent analyzer into the attention module to form an intelligent attention mechanism. Finally, we present the method for skill transfer based on feature extraction and fusion.

1)Analyzer-based Prior Knowledge Acquisition: A complex target task is assigned multiple source strategies, each contributing differently at various stages of execution of the target task. Many studies such as [25] used attention networks to learn this contribution from scratch. However, incorporating human-level prior knowledge into the transfer learning process remains an open question. In this section, we aim to have LLMs analyze this contribution matrix and use it as priori knowledge to correct the attention network. To achieve this, we design specialized prompts for the intelligent analyzer, as shown in Figure 2.

The prompts for the intelligent analyzer consist of two parts. The system role information sets the goal of the intelligent analyzer and provides analysis tips. The input information offers a framework for receiving task descriptions and execution sequences for both the target and source tasks. Based on these prompts, the intelligent analyzer can accurately divide the entire contribution matrix into $m \times n$ matrices, where m represents the number of subtasks of the target task and n represents the number of source tasks involved in the transfer process.

2)Intelligent Attention : Although source skills similar to the target task have been extracted from the skill space, these skills focus on different manipulation actions. Different stages of the target task execution emphasize different source skills. In this paper, we use the Intelligent Attention to assign contributions to different source strategies. The structure of the Intelligent Attention is shown in the bottom right of Figure 3.

The input to the attention module consists of the strategy outputs $\langle \pi_1(s), \ldots, \pi_n(s) \rangle$ from the source tasks. Each strategy is processed through the MLP, and the output for the jth source strategy is

$$h_j = MLP(\pi_j(s); \theta_j) \tag{1}$$

The θ_j is the parameter of π_j. The output of the MLP is then processed through a fully connected layer and softmax to obtain the original contribution value matrix.

$$(w'_1, \ldots, w'_n) = Softmax[Linear(h_1, \ldots, h_n); \theta_l] \tag{2}$$

The θ_l is the parameter of the fully connected layer. We add the contribution value term $(w_{G1}, \ldots, w_{Gn})$ of the intelligent analyzer to this, and output the final contribution value matrix of different source tasks as

$$W = (w_1, \cdots, w_n) \tag{3}$$

Since the intelligent analyzer modifies $(w_{G1}, \ldots, w_{Gn})$ at different segments during the target task execution, it affects the adaptive weighting network for the purpose of providing guidance.

3)Feature Extraction and Fusion: The intelligent transfer network extracts features from the source tasks and transfers them to the target task. Figure 3 gives the network structure when there are two source tasks. To transfer features from the source tasks more efficiently, lateral connection is used. This structure allows the agent to adopt the experience of the source tasks while learning new features.

We use the A2C [26] reinforcement learning algorithm in our experiments. Figure3 illustrates part of the network structure, showing as *Source_task*1 or *Source_taskn*. The target task network and the source tasks network have exactly the same structure. We divide the transfer process into four stages, generating lateral connections only in stages 1, 2, and 3 to extract features at different levels of the skills. To fuse the features of the target task with those of different source tasks, a feature fusion module is added after each feature

extraction stage of the target task network. This connection structure is flexible and allows you to replace the entire network with different models according to your needs.

*Source_task*1,.....,*Source_taskn* are pre-trained strategies with fixed parameters. For the source tasks, the output of the *j*th stage of the *i*th task is SO_i^j, and the total output of the *j*th stage of the source tasks

$$SO^j = (SO_1^j, \cdots, SO_n^j) \tag{4}$$

The equation 3, output from the Intelligent Attention , represents the contribution values of the different source tasks. W is acts on SO^j and U_j is generated

$$U_j = SO^j \otimes W = \left(SO_1^j \cdot w_1, \ldots, SO_n^j \cdot w_n\right) \tag{5}$$

U_j is concatenated with the target output SO_t^j in the *j*th stage to obtain the joint feature matrix F_j

$$F_j = (SO_1^j \cdot w_1, \ldots, SO_n^j \cdot w_n, SO_t^j) \tag{6}$$

Processing this matrix through an MLP module as input to the $(j+1)$th stage

$$I_{j+1} = MLP(F_j; \theta_{MLP_j}) \tag{7}$$

The θ_{MLP_j} denotes the parameters of *MLP* in the jth feature fusion block. Through the feature fusion block, while preserving the results of the target task network (BASE), the features of the source tasks are fused according to task importance, enabling the entire network to learn new task. The features are propagated backward layer by layer until the output layer outputs the action.

$$A = f\,(MLP(I_l); \theta_{out}) \tag{8}$$

I_l is the input to the last feature fusion layer and θ_{out} is the parameter of the output layer.

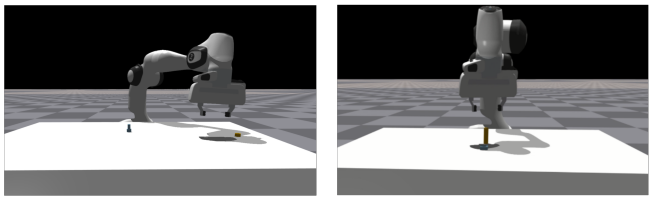
When training the target task, the policy networks of the source tasks are frozen. The parameters of the target network, along with those of the feature fusion module and the intelligent attention module, are continuously updated during the training of the target task.

IV. EXPERIMENTS

Our experiments focus on answering the following questions: 1) Is there practical value in using generator to design reward function for new tasks? 2) How is the performance and generality of ITN? 3) How does ITN perform in manipulation tasks of varying complexity compared to other state-of-the-art methods?

Experimental Environment: We used the Isaac Gym [27] simulator to implement environmental modeling and conduct reinforcement learning experiments. We designed two main tasks: the NutBolt task and the PegHole task. Figure 4 illustrates the modeling of the simulation world. In the simulation, a flexible 7-axis robot, the Franka Emika Panda, was used to manipulate objects, which are various assembled parts used for tasks. Some of these parts are illustrated in Figure 5. Table I lists the experimental tasks,where B-Skill represents basic skill and C-Skill represents complex skill.

Experimental Setup: To ensure the reliability of the method, we used five random seeds for 10 experiments and calculated a 95% confidence interval for each experiment. The A2C algorithm was employed for policy learning and optimization. Computational tasks were performed on a device equipped with an NVIDIA RTX A5000 GPU running Ubuntu 20.04, and the experiments were implemented using the PyTorch framework.



(a) Env-NutBolt (b) Env-PegHole

Fig. 4. Experimental environment. Modeling the simulation world of two primary tasks in the Isaac Gym simulator.

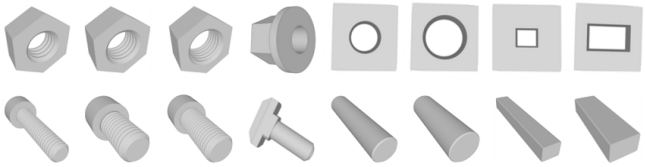


Fig. 5. Manipulate objects for display. The components utilized in two primary tasks.

A. Experiment on Reward Function Generation

We believe it is crucial to design a comprehensive reward function for complex skills in a dynamic environment. We conducted a series of evaluations on manipulation tasks of varying complexity, testing the generator's ability to generate reward functions for both basic and complex skills, as well as its capacity to understand the tasks. The experiment used the GPT-4-0613 variant as an intelligent generator.

Baseline: We compared the performance of GPT-generated reward function in a reinforcement learning task with that of human-designed reward function.

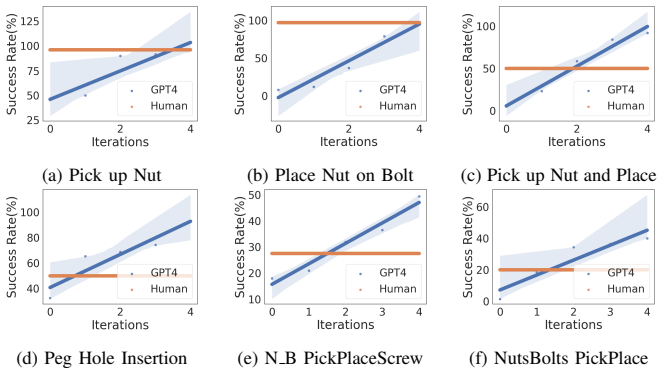
Training Details: The generator performed five iterations per task to generate the reward functions. Each iteration involved 10 samples and selectde the reward function that maximized the average task success rate for the next optimization. After all iterations were completed, the optimal reward function was chosen as the final result of the generator.

Results: The generator-generated reward function outperformed the human-designed reward function. Figure 6 shows the variation in the success rates of task with the number of iterations for both the expert-designed and GPT-4-generated reward functions in guiding the robot in a reinforcement learning task. Comparing the basic experiments (a) and (b), it was evident that the GPT-4-generated reward function could match or even surpass the performance of the human-designed after a few iterations. In the complex tasks (c), (d), (e), and (f), as the number of iterations increased, the reward function designed by GPT-4 proved to be significantly

TABLE I. DETAILED INFORMATION ABOUT THE TASK

Type of task	Task conmmands	Sub-skills	Action	O-object	Model of O-object
B-Skill	Pick up the nut.	Pick	approach-grasp-lift	nut	HexNut-M12,HexNut-M16...
B-Skill	Place the nut on the bolt.	Place	approach-place	nut, bolt	HexNutBolt-M12,HexNutBolt-M16...
B-Skill	Screw the nut onto the bottom of the bolt.	Screw	screw	nut,bolt	HexNutBolt-M12,HexNutBolt-M16...
...	...	...	...	...	...
B-Skill	Insert peg into the hole.	Insert	insert	peg,hole	RoundPegHole-4mm,RectrPegHole-16mm...
C-Skill	Pick up the nut and place it on the bolt.	Pick-Place	approach-grasp-lift-approach-place	nut, bolt	HexNutBolt-M12,HexNutBolt-M16...
C-Skill	Pick up the nut, place it on the bolt, and screw it to the bottom of the bolt.	Pick-Place- Screw	approach-grasp-lift-approach-place-screw	nut, bolt	HexNutBolt-M12,HexNutBolt-M16...
C-Skill	Pick up the nuts and place them on the bolts.	Pick-Place-Pick-Place	approach-grasp-lift-approach-place	nuts, bolts	HexNutBolt-M12,HexNutBolt-M16...
...	...	...	...	...	...
C-Skill	Pick up the round peg and insert it into the round hole.	Pick-Place-Insert	approach-grasp-lift-approach-place-insertion	peg, hole	RoundPegHole-4mm,RectPegHole-16mm...

more effective than the human design. This effectiveness arose because the generator can fully understood the task requirements based on prompts and possessed superior analytical, computational, and memory abilities compared to humans. These results demonstrated the higher efficiency and performance of LLM-based methods for task reward design. In tasks (e) and (f), the success rate of the reward function generated with only five iterations was lower due to the higher complexity. To improve the success rate, increasing the number of iterations and samples was one solution. Our approach made significant progress on tasks of moderate complexity and successfully implemented a reward function that represented the entire task, eliminating the need to design separate reward functions for each subtask. This was particularly important for sequential operating skills. A key to autonomous robot operation was designing a reward function that could guide the entire operation process by automatically generating new tasks for dynamic environments.



(a) Pick up Nut (b) Place Nut on Bolt (c) Pick up Nut and Place

(d) Peg Hole Insertion (e) N_B PickPlaceScrew (f) NutsBolts PickPlace

Fig. 6. Comparison of success rates for different tasks using GPT-4 and human-designed reward functions.

B. Intelligent Transfer Experiments

This section provided a comprehensive evaluation of ITN. Ablation experiment was conducted to assess the performance of each module. The generalization experiment was performed to verify generalization and cross-skill transfer capability. The comparison experiment was carried out to analyze ITN's performance compared to other leading methods. All experiments in this section utilized generator-generated reward functions, with all prior skills derived from the skill space we constructed. Results were presented in the form of graphs for two complex tasks: the *NutBolt_PickPlace* task and the *PegHole_Insert* task (the first and last complex task in the table) from Table I.

1)Ablation Experiment: We used the baselines commonly used for transfer reinforcement learning for comparison, which are as follows:

- *No_Transfer* does not use pre-learned skills but learns from scratch through exploration.
- *No Intelligent Attention (NIA)* is the proposed ITN model without the Intelligent Attention component, assigning a fixed contribution value to each subtask.
- *Direct Feature Summation (DFS)* is the proposed ITN model without Intelligent Attention and Feature Fusion, directly summing features from different source skills.
- *Partial Skills Transfer (PST)* is the proposed ITN method that transfers only a few subskills.

The results are shown in Figure 7. The learning speed (the epoch at which the curve reached stability) of ITN was significantly better than No_Transfer in both tasks. PST assigned different stages of subskills to target tasks, with transferring any stage of subskill significantly enhancing learning efficiency. The independent nature of subskill supported cross-skill transfer in dynamic tasks. Additionally, experiment confirmed that ITN could independently and autonomously learn unknown subskills in complex tasks with only partial subskills available. Comparing ITN with NIA demonstrated the superior effectiveness of integrating an Intelligent Adaptive model over a fixed value. NIA and DFS confirmed the robust feature fusion capability of the feature fusion layer. ITN improved time efficiency by around 72.22% and 65.17% compared to No_Transfer in the respective tasks.

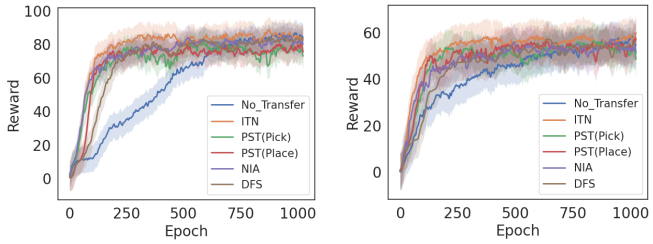


Fig. 7. Results of ablation experiment. Left: NutBolt_PickPlace task. Right: PegHole_Insert task.

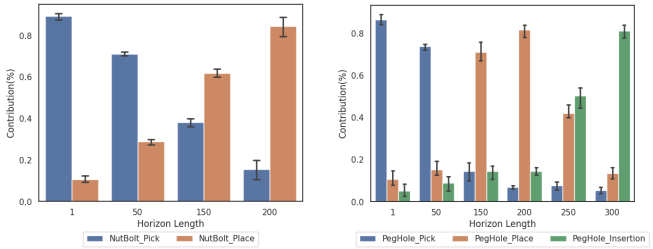


Fig. 8. Attention allocation. These graphs illustrate the changes in the attention values assigned by ITN to different subskills in a cycle in the NutBolt and PegHole tasks.

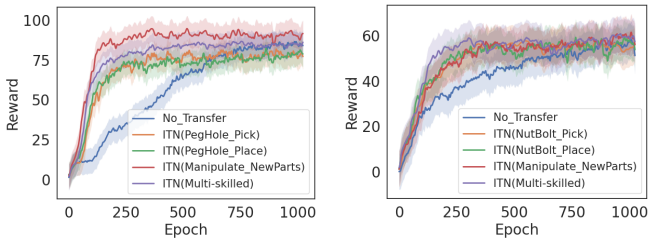


Fig. 9. Results of generalization experiment. Left: NutBolt_PickPlace task. Right: PegHole_Insert task.

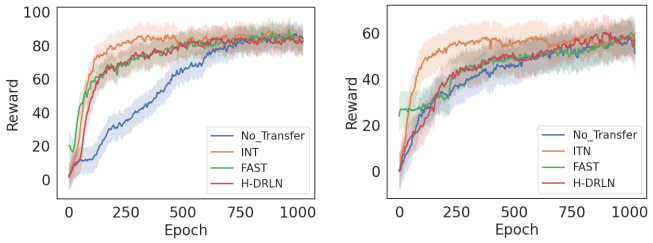


Fig. 10. Results of comparision experiment. On the left is the comparison experiment for the task *NutBolt_PickPlace* and on the right is the comparison experiment for the task *PegHole_Insert*.

Figure 8 shows ITN's policy allocation concerns. The data, derived from a single interaction cycle after training had stabilized, revealed that Intelligent Attention allocated attention to source skills differently at various stages of the target task. The similarity of this distribution to human experience aptly demonstrated the value of introducing the a priori knowledge of the intelligent analyzer to guide the attention module in learning. ITN successfully adapted to various tasks by adjusting the attention values of source skills, enabling robots to quickly master new tasks .

2)Generalization Experiment: This section examined the cross-task transfer capability and generalization of ITN in a dynamic environment. To assess the generalizability of ITN, we designed five scenarios for each task, exchanging the source skills between the two target tasks. Additionally, we evaluated ITN's cross-task transfer capability on new tasks involving previously unseen parts.These evaluations provided comprehensive insights into ITN's ability to adapt and generalize across different robotic manipulation scenarios.

The experimental results are illustrated in Figure 9. Curves 2 and 3 revealed that, although the target task and the source skills had different manipulation goals, ITN effectively utilized shared action features for transfer. This transfer, based on the action feature level, effectively addressed a range of manipulation problems encountered by robots in dynamic tasks. Curve 4 indicated that the robot quickly achieved high rewards when operating unseen new parts, ihighlighting ITN's efficiency in handling variations of the task. Curve 5 showed the best learning speed and the most stable learning curve throughout the training process compared to the other curves, suggesting that the multi-skill fusion strategy was the most effective, significantly enhancing the learning efficiency and performance of the robot manipulation tasks.

3)Comparison Experiment: The experiment aimed to evaluate the competitiveness of ITN in robotics skill transfer learning. We chose two representative advanced methods that can significantly enhance robot skill learning for comparison.

Hierarchical Deep Reinforcement Learning Network (H-DRLN) [28] efficiently retained knowledge using skill distillation and deep skill arrays. It learned when to perform original actions or reuse pre-learned skills. Feature Selection Adaptive Transfer (FAST) [18] facilitated knowledge transfer by extracting shared features between old and new tasks.It performed a series of peg-hole tasks in a simulation. We replicated these approaches and expanded comparison in our experimental setup.

The results are shown in Figure 10. Several methods of learning significantly outperformed direct learning. Although FAST performed best in early training by selecting data closer to the target domain to initialize the policy, it was surpassed by ITN after a few dozen iterations. FAST focused on extracting shared features and excelled in solving transfer problems across manipulated objects, whereas ITN focused on exploring a generalized transfer approach across tasks. H-DRLN attempted to receive the current environment state as input and output the probabilities of new actions and source skills, reusing the subskill with the highest probability value. Although this approach incorporated the learning of new strategies, the reuse approach was very limited in terms of usage scenarios. ITN was not dependent on specific skills and could fully adapt to task changes.

In Table II, our (ITN) method excelled in the early and middle stages, achieving an initial success rates of 86.64% and 79.56% in two tasks, significantly higher than the other methods. As training progressed, the success rate of all methods rose, but ITN achieved a higher success rate at an earlier stage, demonstrating superior time efficiency. The method is highly anticipated for its practical value.

TABLE II. THE TASK SUCCESS RATE OF THE FOUR METHODS VARIES WITH THE TRAINING CYCLE.

Methods	200 epo	400 epo	600 epo	800 epo	1024 epo
NutBolt-PickPlace					
No_Transfer	34.66%	56.32%	81.03%	90.46%	**91.6%**
FAST	71.48%	83.47%	89.91%	91.67%	**91.78%**
H-DRLN	70.96%	85.56%	91.02%	91.54%	**92.02%**
Our(ITN)	86.64%	**92.20%**	**92.20%**	**92.20%**	**92.21%**
PegHole-Insert					
No_transfer	50.49%	65.79%	71.91%	78.69%	**90.73%**
FAST	48.96%	71.91%	76.81%	80.09%	**91.03%**
H-DRLN	58.14%	71.96%	77.52%	87.97%	**91.82%**
Our(ITN)	79.56%	**88.74%**	**89.66%**	**90.27%**	**92.10%**

V. CONCLUSIONS

In this paper, Intelligent Transfer System (ITS) is presented for robots to autonomously and quickly learn new tasks in a dynamic environment. To enable autonomous problem-solving, the system is divided into two parts. Firstly, LLMs is utilized to generate a process-oriented reward function for new task. Secondly, an intelligent transfer module is constructed that leverages LLMs to generate prior knowledge for the transfer process and fuses features from source skills. We conducted a series of simulation experiments to verify the efficiency and superiority of our method. ITS not only has the ability to autonomously learn new and unseen skills but also utilizes related task experience to accelerate learning, which has always been a capability humans hope robots can possess. Additionally, ITS demonstrates superior cross-task transferability, allowing it to adapt well to changes in tasks. For future work, we will first improve the generator's prompt design to enhance the speed and success rate of reward design for complex robotic tasks. Then, we plan to conduct cross-domain experiments by applying ITS to more challenging service robotics fields.

REFERENCES

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[2] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, pp. 1–40, 2016.

[3] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International Conference on Artificial Neural Networks*, 2018.

[4] M. Hutsebaut-Buysse, K. Mets, and S. Latré, "Hierarchical reinforcement learning: A survey and open research challenges," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 172–221, 2022.

[5] Y. Zhu, P. Stone, and Y. Zhu, "Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4126–4133, 2022.

[6] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2018, arXiv preprint arXiv:1707.08817, 2018.

[7] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv: Arxiv-2310.12931*, 2023.

[8] R. Zhang, J. Lv, J. Li, J. Bao, P. Zheng, and T. Peng, "A graph-based reinforcement learning-enabled approach for adaptive human-robot collaborative assembly operations," *Journal of Manufacturing Systems*, vol. 63, pp. 491–503, 2022.

[9] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow, "Meta-reinforcement learning for robotic industrial insertion tasks," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9728–9735.

[10] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6023–6029.

[11] K. Xu, H. Yu, Q. Lai, Y. Wang, and R. Xiong, "Efficient learning of goal-oriented push-grasping synergy in clutter," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6337–6344, 2021.

[12] Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, "Distral: Robust multitask reinforcement learning," 2017, arXiv preprint arXiv:1707.04175, 2017.

[13] Y. Tao, S. Genc, J. Chung, T. Sun, and S. Mallya, "Repaint: Knowledge transfer in deep reinforcement learning," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 10 141–10 152.

[14] A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Žídek, and R. Munos, "Transfer in deep reinforcement learning using successor features and generalised policy improvement," 2019, arXiv preprint arXiv:1901.10964, 2019.

[15] H. Chang, Z. Xu, and M. Tomizuka, "Cascade attribute network: Decomposing reinforcement learning control policies using hierarchical neural networks," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8181–8186, 2020, 21st IFAC World Congress.

[16] A. Zhang, H. Satija, and J. Pineau, "Decoupling dynamics and reward for transfer learning," 2018, arXiv preprint arXiv:1804.10689, 2018.

[17] Y. Gai, B. Wang, J. Zhang, D. Wu, and K. Chen, "Robotic assembly control reconfiguration based on transfer reinforcement learning for objects with different geometric features," *Engineering Applications of Artificial Intelligence*, vol. 129, p. 107576, 2024.

[18] L. Jin, Y. Men, R. Song, F. Li, Y. Li, and X. Tian, "Robot skill generalization: Feature-selected adaptation transfer for peg-in-hole assembly," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 3, pp. 2748–2757, 2024.

[19] Y. Jin, D. Li, Y. A, J. Shi, P. Hao, F. Sun, J. Zhang, and B. Fang, "Robotgpt: Robot manipulation learning from chatgpt," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2543–2550, 2024.

[20] X. Li, G. Tian, and Y. Cui, "Fine-grained task planning for service robots based on object ontology knowledge via large language models," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 6872–6879, 2024.

[21] L. Chen, Y. Lei, S. Jin, Y. Zhang, and L. Zhang, "Rlingua: Improving reinforcement learning sample efficiency in robotic manipulations with large language models," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6075–6082, 2024.

[22] J. Wang, Z. Wu, Y. Li, H. Jiang, P. Shu, E. Shi, H. Hu, C. Ma, Y. Liu, X. Wang, Y. Yao, X. Liu, H. Zhao, Z. Liu, H. Dai, L. Zhao, B. Ge, X. Li, T. Liu, and S. Zhang, "Large language models for robotics: Opportunities, challenges, and perspectives," 2024, arXiv preprint arXiv:2401.04334, 2024.

[23] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *J. Mach. Learn. Res.*, vol. 13, no. null, p. 723–773, mar 2012.

[24] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019.

[25] H. Shi, J. Li, J. Mao, and K.-S. Hwang, "Lateral transfer learning for multiagent reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 53, no. 3, pp. 1699–1711, 2023.

[26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, p. 1928–1937.

[27] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.

[28] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "A deep hierarchical approach to lifelong learning in minecraft," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. AAAI Press, 2017, p. 1553–1561.