Task-Oriented Adaptive Learning of Robot Manipulation Skills

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

***Abstract*— The robot's operating environment and tasks are dynamically changing. Therefore, developing a mechanism that can infer other things from one fact to adapt to various scenarios is crucial for enhancing the robot's adaptive capabilities. This paper proposes a general Intelligent Transfer System (ITS) to enable rapid skill learning for robots in dynamic tasks. ITS integrate Large Language Models (LLMs) with transfer learning, leveraging LLMs' intelligence and prior skill 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 these task.This approach eliminates the need to design hierarchical sub-processes for complex tasks. In addition, we designed an intelligent transfer network (ITN) in ITS to extract knowledge of relevant skills for the learning of new ones. This paper's method holds promise for enabling robots to independently solve entirely new operational tasks. We conducted a series of evaluations in a simulation environment, and the experimental results show 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. You can get the code with all the experimental videos on our project website: https://jkx-yy.github.io/**

I. INTRODUCTION

Robot operations in the real world are constantly adapting due to dynamic changes in the environment and tasks. Traditional pre-programmed behavioral planning and decision-making methods are difficult to adapt to these changes [1], while reinforcement learning methods need to be re-trained from scratch when the environment and operation objects are slightly changed [2]. Transfer learning, which uses relevant task experience to improve the efficiency of autonomous learning on new tasks, greatly reduces the heavy burden of repetitive learning on robots, and holds new promise for intelligent robot operation [1], [3]. Nevertheless, designing an effective and automated approach to construct a generalized transfer framework for unknown new tasks in the context of long-term robot operation tasks remains a challenge .

Some work has achieved good results in specific settings. When dealing with complex robot manipulation tasks, Hierarchical Reinforcement Learning (HRL) employs high-level strategies to select underlying strategies for performing specific subtasks. The learning process is significantly enhanced by reusing previously learned skills [2], [3]. However, these reused methods cannot be adapted to new, unknown tasks. There is some recent work that combines HRL with imitation learning to solve vision-based long-field-of-view robot manipulation [4], but it suffers from the common limitations of offline datasets and requires task-specific meta-controllers to combine skills for a single task. In another work, expert demonstrations are introduced to guide the strategy search [5], thus accelerating the learning process in sparse reward environments. However, these methods may suffer from reduced algorithmic effectiveness when the demonstration data is very limited or of low quality. In summary, existing skill transfer learning techniques have high requirements for pre-training strategies, are more sensitive to changes in the environment, and rely on human involvement,When new tasks involve unknown underlying strategies, these methods fail due to their inability to autonomously learn new skills.

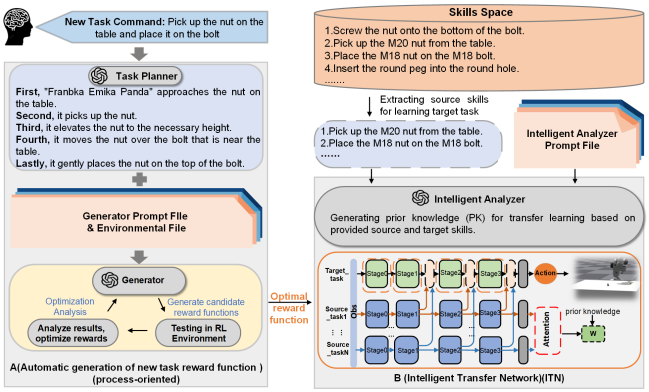


Fig. 1: The Intelligent Transfer System (ITS) is divided into two parts.In A, the generator analyzes the task and environment to generate the optimal reward function through multiple optimization.In B, an Intelligent Transfer Network (ITN) is designed to accelerate the skill learning by fusing the features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

On the other hand, we consider that LLMs excel in code generation capabilities and semantic planning. [6] utilized ChatGPT for problem solving in robot operations to train a reliable agent RobotGPT significantly improved the task success rate. [7] utilized LLMs for fine-grained task planning and improved the efficiency and effectiveness of task planning. There are several other listed tasks that show the great potential of LLMs for robotics applications [8]. However, whether LLMs can be used to guide skill transfer learning remains an unexplored question. In this paper, we fully utilize the analytical capabilities of LLMs to replace the human analytical process and use the results to guide skill transfer.

To address the challenge of enabling robots to learn new skills autonomously and rapidly in dynamic environments, we developed a task-oriented intelligent transfer method, as

This work is Supported by National Natural Science Foundation of China(U22A2059), National Natural Science Foundation of China under Grant 62273203, the National Key RD Program of China under Grant 2018YFB1307101, and the Tais- han Scholars Program of Shandong Province(ts201511005). (Corresponding author: Guohui Tian.)

KeXin Jin, Guohui Tian, Bin Huang, XiaoYu Zheng are with the School of Control Science and Engineering, Shandong University, Jinan, 250061, China. (email:jkx983008880@163.com; g.h.tian@sdu.edu.cn;)

shown in Figure 1. Given that the learning of robotic skills heavily relies on the design of reward function, we designed an automatic reward function generation scheme. To store learned skills, we constructed a dynamically expandable skill space that assigns source strategies to target task based on the semantic information of skills. Additionally, we integrated transfer learning with LLMs to form an intelligent knowledge-guided ITN. Inspired by [9], we horizontally connected the feature extraction modules of source and target tasks and fused the features of different skills.

We summarize four key contributions of this work: (1)We propose an innovative and generalized ITS that leverages the code generation, problem analysis, and planning capabilities of LLMs to assist robots in autonomous skill learning for new tasks. (2)Since manually designing separate reward functions for each subtask of the target task is labor-intensive and limits the robot's autonomy in handling complex tasks, we developed a reward function generator dominated by LLMs. This approach aims to create a comprehensive reward function for the operational process and was evaluated across tasks of varying complexity. (3)To fully utilize empirical knowledge, we developed an intelligent analyzer, dominated by LLMs, that analyzes the contribution relationship matrix between source and target tasks, serving as empirical knowledge for the attention module. (4)To enable the robot to utilize previously acquired skills, we designed an ITN that incorporates prior knowledge from LLMs. We developed feature fusion modules at three different dimensions to be inserted at various locations within the network, allowing for the extraction of features at varying levels of source skills.

II. RELATED WORK

A. Transfer Reinforcement Learning

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 a set of the state of the environment and the state of the agent.$A = a_0 \times a_1 \times \cdots \times a_n$ is the set of all possible actions, and the action represents the operation that can be performed by the agent in a certain situation. $R = r_0, r_1 \cdots, r_n$ is the agent's reward function, $R(s,a)$ is the immediate reward obtained after taking action a in state s. The reward function provides feedback to the agent, guiding it to learn which actions are beneficial and which are harmful; $\gamma \in [0,1]$ is the discounting factor used to weigh the importance of current and future rewards. The goal of reinforcement learning is to learn an optimal policy $\pi^*(a|s)$ to maximize the agent's cumulative reward.Transfer learning is systematically presented and summarized in the article [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 optimal strategy.

B. Transfer Reinforcement Learning of Robot Skills

Some typical transfer learning methods effectively address the transfer problem between source and target domains. [10], [11] utilized knowledge from the source domain for action generation through strategy reuse or strategy distillation [12] [13] [14] 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. [15] proposed a method for reconfiguring robot assembly control using Equivalent Compliance Theory (ECT) combined with Weighted Dimensional Policy Distillation (WDPD). [16] 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. [17] employed feature selection and adaptive transfer learning to generalize robot peg-in-hole assembly skills across different geometric features. [18] investigated meta-reinforcement learning to solve simulated industrial insertion tasks and quickly adapt policies in real-world scenarios. [19] and [20] aimed to use transfer learning with reinforcement learning for tasks like stacking and grasping. However, these approaches primarily focus on generalizing parts within the same operational 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 Robotics

[21] explored how LLMs generate new tasks and simulate behaviors to enhance robots' learning and adaptability. [22] reduced the sample complexity of reinforcement learning in robotics tasks by prompt engineering to extract prior knowledge from LLMs . Additionally, [23] designed human-level reward systems by encoding LLMs, thereby improving the efficiency and performance of robots in task learning. Our approach is similar to this work, but our goal is to design a generalized reward function generation framework for complex robot manipulation tasks, with a focus on the entire process of robot manipulation. In this paper, we leverage the task planning, problem analysis, and code generation capabilities of LLMs to design planners, analyzers, and generators, 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 Rewards

When robots learn complex tasks, traditional hierarchical reinforcement learning methods decompose these tasks into subtasks, designing separate rewards for each, which presents a challenge in ensuring connectivity among subtasks. We propose a generalized reward prompt framework for LLMs that aims to generate a comprehensive reward function for complex tasks, thereby accommodating task variations.

1) Planning Stage: When task requirements change, the robot receives a new, concise task command. However, these initial commands are often unclear.To obtain a complete and accurate description of the task, we designed a task planner capable of analyzing the execution order of subtasks and

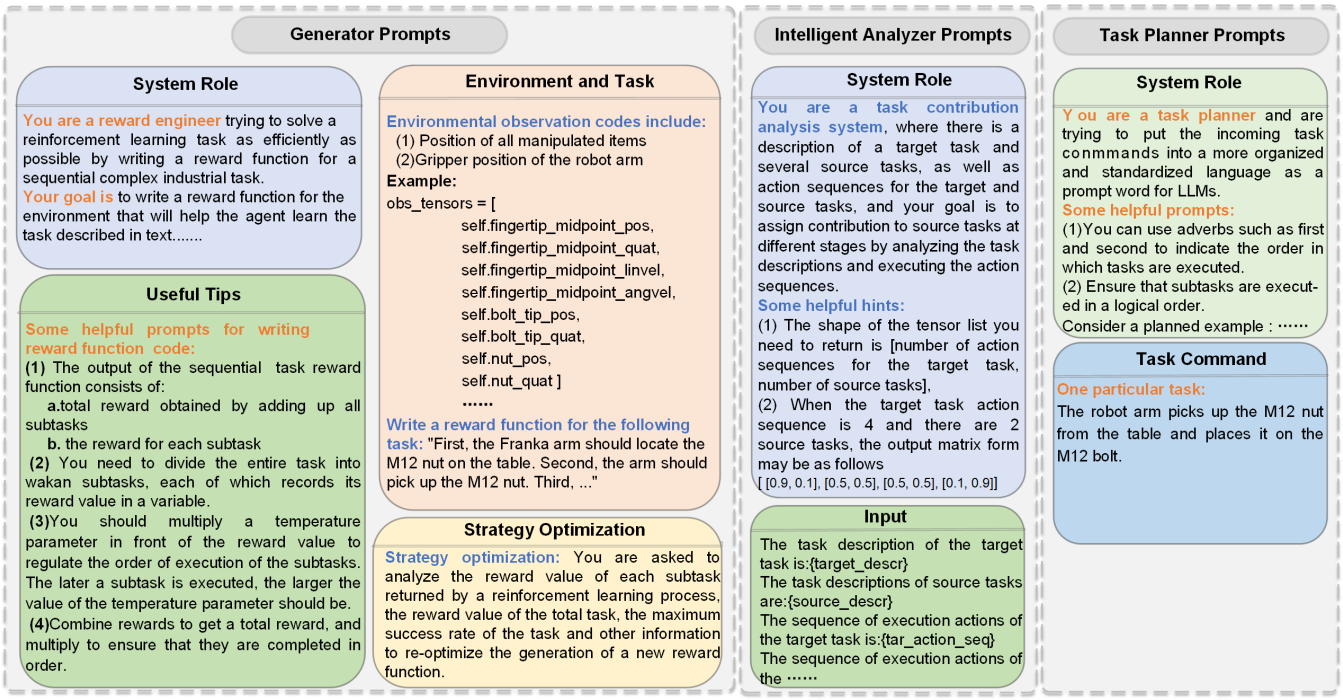


Fig. 2: This paper examines the prompt systems of three LLMs. The first, Reward System Prompts, serves as a management system for designing reward functions for new tasks. The second, Intelligent Analyzer Prompts, enables LLMs to analyze the contribution of different source tasks to the target task at various times through prompts design. The third, Task Planner Prompts, refines user commands.

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 functions and the success rate of task execution. We designed a four-part effective prompt, as shown in Figure 2, The system information describes the system's identity and goals. Environment information comprises environmental observations and the positional information of operating items and robotic arm,the task description uses a fine-grained description generated by the task planner. Generation prompts information 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 tasks. The strategy optimization information we provide includes several analyzable metrics for generating improved reward functions. Our prompts were comprehensively designed and received a satisfactory response.

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

B. Skill Space Construction and Source Skills Extraction

To accelerate learning new tasks, we extract features from useful source tasks and store them in a real-time expandable skill space with over 30 skills, covering basic and complex learned abilities of the robot. Basic skills such as Pick, Place, Screw, and Insert are used to manipulate various mechanical parts. Sequential complex tasks are designed as assembly tasks containing multiple sub-skill. 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 [24] 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 skills operating on different objects can positively transfer to the target skill if the execution semantics are the same, we designed a skill extraction method based on SBERT [25]. This method extracts source skills similar to the target task at the semantic level. It 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. First, we describe the generation

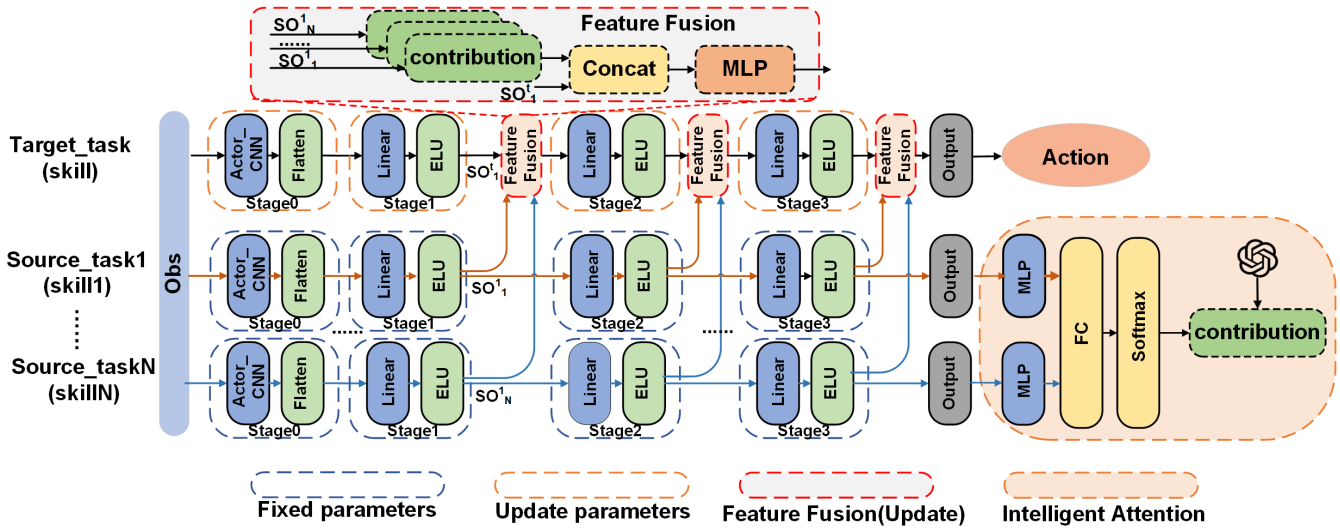


Fig. 3: The Intelligent Transfer Network (ITN) connects the models of source tasks horizontally. The Intelligent Attention module takes the strategies of the source tasks as input and outputs the final contribution values of each subtask in combination with the contribution matrix of the Intelligent Analyzer. The feature fusion layer fuses features from each source task. The final output of the network is the strategy for the new task.

of a priori knowledge based on the intelligent analyzer. Then, we integrate the intelligent analyzer into the attention module to form a intelligent attention mechanism. Finally, we present the intelligent transfer network framework, as shown in Figure 3.

1)Intelligent Analyzer Acquires Prior Knowledge: A complex target task is assigned multiple source strategies, each contributing differently at various stages of execution. Many studies such as [8] use 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 a priori knowledge to correct the attention network. To achieve this, we designed 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 to guide the attention network, where m represents the number of segments 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 operational 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 output $\langle \pi_1(s), \ldots, \pi_n(s) \rangle$ of 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 moments of the target task execution, it affects the adaptive weighting network for the purpose of guidelines.

3)Intelligent Transfer Network: The intelligent transfer network extracts features on the source tasks and transfers them to the target task. Figure 3 gives the structure of the network when the source tasks are two. In order to transfer features from the source tasks more efficiently, horizontal connections are 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 to *Source_taskN*. The target task network and the source tasks network have exactly the same structure. We divide the transfer process into four phases, generating lateral connections only in stage 1, 2, and 3 to extract features at different levels of the task. To fuse the features of the target task with those of different source tasks, we add a feature fusion module 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 jth stage of the ith task is SO_i^j, and the total output of the jth 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 jth 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.

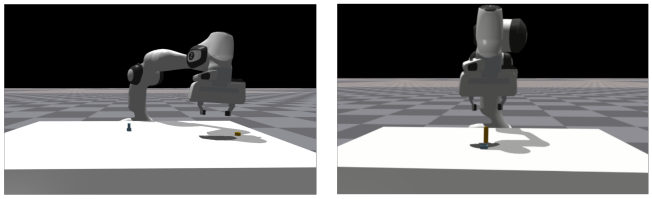
When training the target task, the policy networks of the source tasks are frozen. The parameters of the target network, along with those of the feature fusion module and the intelligent attention module, are continuously updated during the training of the target task.

IV. EXPERIMENTS

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

Experimental Environment: We use the Isaac Gym [27] simulator to implement environmental modeling and conduct reinforcement learning experiments. We designed two main types of tasks: the *NutBolt* task and the *PegHole* task. Figure 4 illustrates the modeling of the simulation world. In the simulation, flexible, lightweight, 7-axis robot Franka Emika Panda used to manipulate objects. The manipulated items are various assembled parts used for tasks, some of which are illustrated in Figure 5. Table I lists the experimental tasks, with B-Skills representing basic skill and C-Skills representing complex skill.

Experimental Setup: To ensure the reliability of the method, we used five random seeds for 10 experiments and calculated 95% confidence intervals for each experiment. All experiments were conducted using the A2C algorithm for policy learning and optimization. Other experimental parameters are provided in the PPO file of the task. All 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: Modeling the simulation world of two primary tasks in the Isaac Gym simulator.

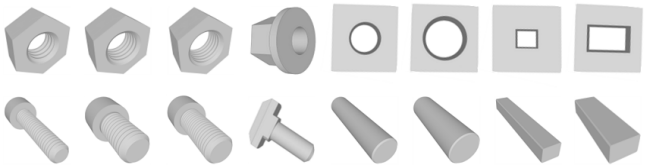


Fig. 5: The figure illustrates the components utilized in the experiment's two primary tasks: the *NutBolt* task and *PegHole* task.

A. Experiments on Reward Function Generation

We believe it is crucial to design a synthesized reward function for complex skills in a dynamic environment. We conduct a comprehensive evaluation 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 uses the GPT-4-0613 variant as an intelligent generator.

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

Training Details: The generator performs five iterations for each task to generate the reward function. Each iteration performs 10 samples and selects the reward function that maximizes the average task success rate for the next optimization. After all iterations are completed, the optimal reward function is chosen as the final result of the generator.

Results: The generator-generated reward function outperforms the human-designed reward function. Figure 6 shows the variation in the success rate 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 is evident that the GPT-4-generated reward function can match or even surpass the performance of the human design after a few iterations. In the complex tasks (c), (d), (e), and (f), as the number of iterations increases, the reward function

TABLE I: DETAILED INFORMATION ABOUT THE TASK

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

designed by GPT-4 proves to be significantly more effective than the human design. This effectiveness arises because the generator can fully understand the task requirements based on prompts and possesses superior analytical, computational, and memory abilities compared to humans. These results demonstrate 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 is lower due to the higher complexity. To improve the success rate, increasing the number of iterations and samples is one solution. However, for complex tasks containing multiple subtask, combining the method of this paper with hierarchical learning could yield unexpected results. Our approach has made significant progress on tasks of moderate complexity and successfully implemented a reward function that represents the entire task, eliminating the need to design separate reward functions for each sub-task. This is particularly important for sequential operating skills.

(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 Experiment

In this section, we provide a comprehensive evaluation of the performance of ITN. First, evaluating the performance of each ITN module through designed validation experiments. Next, testing the generalization and cross-skill transfer capabilities of ITN. Finally, analyzing ITN's performance in comparison with other leading approaches. All experiments in this section utilize generator-generated reward functions. All priori skills are derived from the skill space we constructed. We will present the results in the form of graphs for two complex tasks: the *NutBolt_PickPlace* task and the *PegHole_Insert* task (the first and last complex tasks in the table) from Table I.

1)Verification Experiment: We used the baseline commonly used for transfer reinforcement learning as a comparison, which is 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 sub-skills.

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

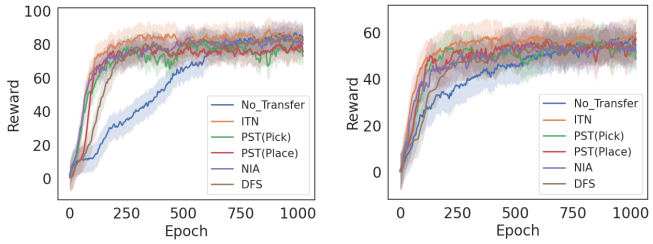


Fig. 7: The left figure is the validation experiment of *NutBolt_PickPlace* and the right figure is the validation experiment of *PegHole_Insert*.

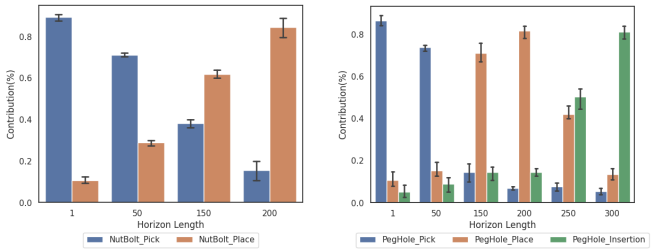


Fig. 8: The left and right graphs show the variation of subskill attention values over an epoch for ITN in the NutBolt and PegHole experiments.

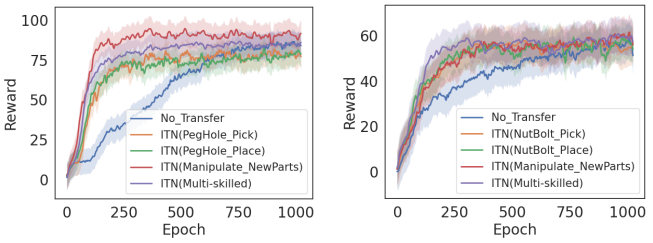


Fig. 9: ITN cross-task transfer experiment with *NutBolt_PickPlace* task on the left and *PegHole_Insert* task on the right.

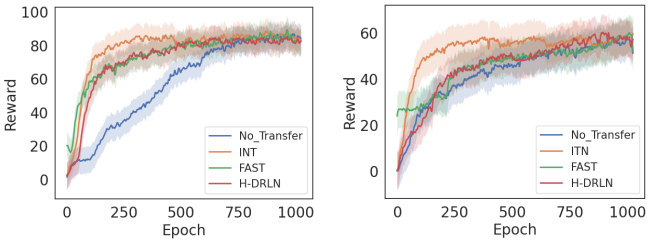


Fig. 10: On the left is a comparison experiment for the task *NutBolt_PickPlace* and on the right is a comparison experiment for the task *PegHole_PickPlaceInsert*.

Figure 8 shows ITN's policy allocation concerns. The data, derived from a single interaction cycle after training has stabilized, indicates that the Intelligent Attention allocates attention to source skills differently at various stages of the target task. The similarity of this distribution to human experience aptly demonstrates the value of introducing the a priori knowledge of the intelligent analyzer to guide the attention module to learning. ITN can successfully adapt to various tasks by adjusting the attention values of source skills, laying the foundation for robots to quickly master new tasks.

2)Generalized Experiment: This section examines the cross-task transfer capability and generalization of ITN in dynamic environments. We designed five scenarios for each task to test the generalizability of ITN by exchanging the source skills of the two target tasks. Additionally, we assessed the cross-task transfer capability of ITN on new tasks by manipulating previously unseen parts.

The experimental results are illustrated in Figure 9. Curves 2 and 3 demonstrate that, although the target task and the source skills have different operational goals, the ITN can effectively utilize shared action features for transfer.This transfer, based on the action feature level, can effectively address a range of manipulation problems encountered by robots in dynamic tasks. Curve 4 shows that the robot quickly achieves high rewards when operating unseen new parts, indicating ITN's efficiency in handling variations of the task. Curve 5 demonstrates 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 is the most effective, significantly improving the learning efficiency and performance of the robot manipulation task.

3)Comparison Experiment : The experiment aims to evaluate the competitiveness of ITNs in robotics skill transfer learning. We have chosen to compare two advanced and representative approaches in the field of transfer learning.

Hierarchical Deep Reinforcement Learning Network (H-DRLN) [28] efficiently retains knowledge through skill distillation and deep skill arrays. It learns strategies to determine when to perform original operations or reuse pre-learned skills. The original paper evaluates this method in Minecraft. Here, we replicated the approach in our experiments.

Feature Selection Adaptive Transfer (FAST) facilitates knowledge transfer between old and new tasks by extracting shared features of the source and target domains [17]. It performs a series of peg-hole tasks in a simulation. We replicated this approach and conducted an experimental comparison in our own experimental setup.

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

In table II, Our (ITN) method excels in the early and middle stages, achieving initial success rates of 86.64% and 79.56% in two tasks, significantly higher than the other methods. As training progresses, success rates of all methods

rise,but ITN achieves higher success rates at an earlier stage ,demonstrating superior time efficiency . The method is highly anticipated for its practical value.

TABLE II: THIS TABLE DISPLAYS HOW THE TASK SUCCESS RATES OF FOUR METHODS CHANGE WITH TRAINING CYCLES

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 letter, we present a method for robots to autonomously and quickly learn new tasks in dynamic environments. To enable autonomous problem-solving, we divide the system into two parts. First, we utilize LLMs to automatically generate a process-oriented reward function for new tasks. Second, we construct an intelligent transfer module that uses LLMs to generate prior knowledge for the transfer process and intelligently fuse features from source skills. We conducted a series of simulation experiments to verify the efficiency of the method. While our approach performs well in handling tasks of moderate complexity, designing a generalized reward function for highly complex tasks remains challenging. We have high expectations for the transfer performance of our framework on more complex tasks. In the future, we plan to further combine our approach with methods such as HRL to conduct experiments on more challenging robot manipulation tasks.

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] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer, "Grounding large language models in interactive environments with online reinforcement learning," 2023, arXiv preprint arXiv:2302.02662, 2023.

[22] 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.

[23] 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.

[24] 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.

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

[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.