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*— Robots' working conditions and manipulation tasks are constantly changing in industrial environments. 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 new skills rapidly and autonomously in dynamic tasks. ITS integrates Large Language Models (LLMs) with transfer reinforcement learning, leveraging LLMs' intelligence and prior skills knowledge. It can comprehend previously unseen task commands and automatically generate a process-oriented reward function based on task reward_components for each task, enabling 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 and accelerate learning new ones. We systematically evaluate our method in the simulation environment. The results demonstrate that it can autonomously and efficiently learn unseen skills without relying on pre-programmed behavior, achieving true creativity while improving the time efficiency of two major tasks by 72.22% and 65.17% compared to learning from scratch. For code and videos, please visit our project website:*https://jkx-yy.github.io/**

I. INTRODUCTION

The operations of robots in the real world adjust continuously due to dynamic changes in the environment and tasks. They must autonomously and quickly learn new skills to operate effectively without human intervention. Most existing work studies these two aspects separately, but designing an effective, automated, and generalizable learning framework for unknown tasks in robot manipulation remains a significant challenge.

Integrating LLMs and robots can enhance the robot's autonomous learning capabilities. Code as Policies [1] is a technology that generates robot action strategies through natural language commands. RobotTool [2] enables robots to understand natural language commands and creatively use tools to complete complex tasks. However, these two methods heavily rely on predefined control primitives and cannot handle new skills that are not included in the primitives, only combining existing skills. In order to achieve true creativity without relying on existing programming code, Text2Reward [3] generates reward functions for reinforcement learning from language commands to guide robot skill learning. Although this method can greatly enhance the robot's ability to innovate in learning new skills, introducing human feedback reduces the autonomy of the robot's skill-learning process. Eureka [4] solved the problem by introducing a self-optimization mechanism to generate the optimal reward function of the task. However, due to its lack of understanding of professional operations, the efficiency of autonomously generating reward components is low, reducing the overall efficiency of producing executable reward functions. To address the issues of lack of creativity and high dependency, we design a general task-oriented reward function generator, which uses automatically generated, highly accurate reward_components to enable efficient reward function generation.

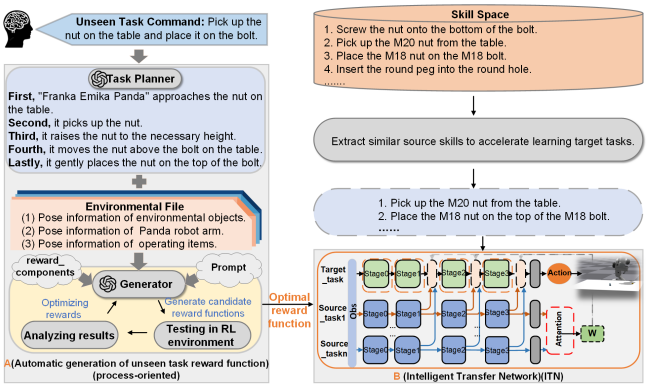


Fig. 1. Overview of ITS. The Intelligent Transfer System (ITS) is divided into two parts. In A, the generator analyzes the task command, environmental file and reward_components to generate the optimal reward function. In B, an Intelligent Transfer Network (ITN) is designed to accelerate the new skill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

Certain studies 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 skills, thereby accelerating the learning process by leveraging previously learned skills [5]. 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 [6]. Yet, it suffers from offline dataset limitations and requires task-specific meta-controllers. Another approach uses expert demonstrations to guide strategy search, accelerating learning in sparse reward environments [7]. However, limited or low-quality demonstration data can reduce the effectiveness of these methods.

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)

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, lack the general ability for cross-task transfer, and fail to learn new tasks involving previously unlearned skills autonomously. To this end, this paper proposes a general transfer learning framework that accelerates the learning process of unexplored skills by utilizing already acquired skills.

A task-oriented ITS is developed to enable robots to rapidly and autonomously learn new skills in dynamic environments (Fig. 1). An automatically generated process-oriented reward function based on reward_components enhances the robot's ability to solve new tasks autonomously. A dynamically expandable Skill Space stores learned skills and assigns skills to target tasks. An ITN is constructed utilizing source skills to accelerate learning target tasks.

We summarize three key contributions of this work:

(1) In order to realize the autonomous and fast learning of unknown new tasks for robots, an ITS integrating LLMs and transfer learning is designed—the system without relying on human help and predefined action primitives.

(2) In order to realize the autonomous learning of robot skills, a generator based on reward_component acceleration is designed, which can generate the optimal reward function in a shorter time than previous work.

(3) In order to equip the system with general cross-task transfer capabilities and accelerate the learning of new skills, an ITN is proposed to integrate shared skill features. Notably, this new task can either combine existing tasks within the Skill Space or involve operations that have never been encountered before in the Skill Space.

II. RELATED WORK

A. Application of LLMs's Generative Ability and Comprehension Ability in Robots

With their comprehensive knowledge base and powerful utilization and generation capabilities, LLMs are increasingly applied in robotics. [8] demonstrated the great potential of LLMs in robotics applications. [9] utilized LLMs for fine-grained task planning and improved the efficiency and effectiveness of task planning. [10] utilized ChatGPT for problem-solving in robot operations to train a reliable agent-RobotGPT significantly improved the task success rate. [11] reduced the sample complexity of reinforcement learning in robotics tasks by prompt engineering to extract prior knowledge from LLMs. In this paper, we leverage LLMs' task planning, problem analysis, and code generation to design a planner and generator, replacing human intervention and enabling automatic implementation of ITS.

B. Transfer Reinforcement Learning in Robotic Skills

Transfer reinforcement learning combines the ideas of reinforcement learning and transfer learning. A reinforcement learning task is a tuple $\langle S, A, P, R, \gamma \rangle$, S is the set of states representing the environment and agent, A represents the actions an agent can take in a given state, $P(s' \mid s, a)$ is the probability of transitioning from state s to state s' after action a, $R = r_0, r_1 \cdots, r_n$ is the reward function, and $R(s,a)$ is the immediate reward obtained after taking action a in state s, $\gamma \in [0,1]$ is the discount factor. It aims to learn an optimal policy $\pi^*(a|s)$ to maximize cumulative rewards. Extensive research exists on using reinforcement learning for machine manipulation tasks. [12] used a graph neural network to represent states and actions, optimizing collaborative human-machine assembly with a reinforcement learning algorithm. [13] investigated meta-reinforcement learning to solve simulated industrial insertion tasks and quickly adapt policies in real-world scenarios. [14] and [15] applied reinforcement learning to stacking and grasping tasks. 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, [16] [17] used source domain knowledge for action generation through strategy reuse or distillation. [18] [19] [20] improved the training process of the target task by decoupling or reusing task-invariant features. Recent studies have explored combining transfer learning with reinforcement learning to address real-world robotics problems. [21] applied feature selection and adaptive transfer learning to generalize robot peg-in-hole assembly across varying geometric features. Most existing 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, we aim to handle various tasks that evolve efficiently.

III. METHOD

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

A. Automatic Generation of Task-oriented Reward Functions

We propose a general LLMs prompt framework to generate an optimal reward function for unseen tasks. Dataset info, the reward optimization process, failure examples, and ITS pseudocode are available on the project website.

1) Reward_components Generation Stage: To generate executable reward functions more efficiently, we constructed a dataset containing 1000 task-reward_components involving manipulation tasks of varying complexity. Fine-tuning the LLaMA [22] gives it accurate reward_component cognitive ability. The task command in Fig. 1 is automatically analyzed into three distance reward_components: $distance\_finger\_to\_nut$, $distance\_nut\_to\_height$, and $distance\_nut\_to\_bolt$. Compared to the random generation methods of Text2Reward [3] and Eureka [4], it is more time-efficient and successful.

2) Planning Stage: The robot receives a new, concise task command when task requirements change. However, this initial command is often unclear. To obtain a complete and accurate description of the task, we design a task planner capable of analyzing the execution order of subtasks and

generating a fine-grained task description. Fig. 2 illustrates the prompts used in the task planner.

3) Generation Stage: An effective prompting approach can significantly enhance the speed of generating executable reward functions and the success rate of task execution. We design an effective four-part prompt as shown in Fig. 2. The system information describes the system's identity and goals. Environment information comprises environmental observation data and the positional information of manipulated objects and the Panda arm. The task description uses a fine-grained description generated by the task planner. In order to ensure that the generated reward function has a high enforceability and success rate, useful prompts are designed, including the proposed reward form and constraint hint. The strategy optimization information includes several analyzable metrics for generating improved reward functions.

4) Reward Optimization Stage: In each iteration, the generator generates multiple reward function candidates based on the sample size N and regenerates them if the initial generation fails. To ensure the generated reward function correctly guides the robot to complete skill sequentially, we record the maximum task success rate, the value of each reward_component, and the total reward for complex skill in each iteration for each reward candidate. Then, based on these N sets of data, LLM analyzes the parameter values of each component in the next iteration to maximize the total cumulative reward to achieve the purpose of optimization. Through this self-optimization process, the success rate and accuracy of the reward function are continuously improved.

B. Skill Space Construction and Source Skills Extraction

In order to speed up the learning of new tasks, useful source tasks are stored and extracted in a real-time expandable Skill Space that encompasses both basic and complex skills learned by the robot. Basic skills like Pick, Place, Screw, and Insert are used to manipulate various mechanical parts. Complex skills are assembly tasks composed of multiple subskills. Each skill in the Skill Space consists of four attributes: serial number, task description, pre-trained model, and task execution action sequence. We designed a skill extraction method based on SBERT [23] to extract source skills similar to the target task at the semantic level. This method is robust, tends to extract deeper source skills, and has strong interpretability compared with other methods.

C. Intelligent Transfer Network (ITN)

In this section, we use the reward function generated in III-A and show the implementation of cross-task skill transfer, as illustrated in Fig. 3. This approach aims to accelerate learning new skills by leveraging existing knowledge. Firstly, the multi-task attention allocation mechanism is introduced. Then, a skill transfer method based on feature extraction and fusion is proposed.

1)Intelligent Attention: Although source skills similar to the target task have been extracted from the Skill Space, these skills focus on different actions. Different stages of the target task execution emphasize different source skills.

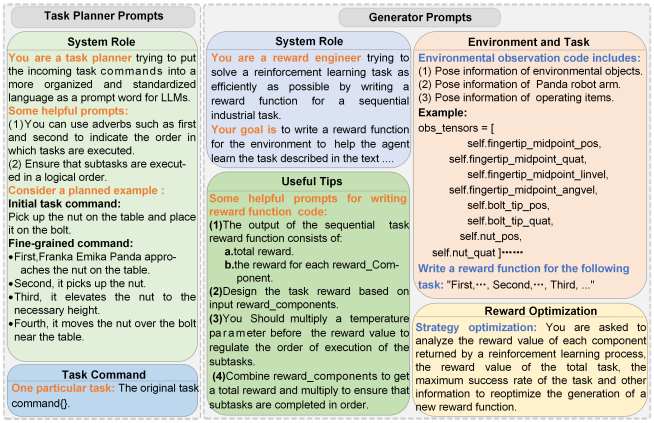


Fig. 2. Prompts for Task Planner and Generator. Task Planner refines user commands. The generator manages the reward function design.

This paper uses Intelligent Attention to assign contributions to different source strategies. The structure of the Intelligent Attention is shown in the bottom right of Fig. 3.

The source policies related to the target task are $\langle \pi_1, \ldots, \pi_n \rangle$. Given the input environmental observation Obs (including the pose information of the objects and the Panda arm), the output is $\langle \pi_1(s), \ldots, \pi_n(s) \rangle$. 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 MLP. The output of the MLP is then processed through a Fully Connected Layer (FC) and softmax to obtain the contribution value matrix W.

$$(w_1, \ldots, w_n) = Softmax[Linear(h_1, \ldots, h_n); \theta_l] \tag{2}$$

The θ_l is the parameter of the FC.

2)Feature Extraction and Fusion: The Intelligent Transfer Network extracts features from the source tasks and transfers them to the target task. A lateral connection is used to transfer features from the source tasks more efficiently. This structure allows the agent to adopt the experience of the source tasks while learning new features. This is why ITN can learn new skills that are not within the Skill Space.

We use the A2C [24] reinforcement learning algorithm in our experiments. Fig. 3 illustrates part of the network structure, showing as $Source\_task1$ or $Source\_taskn$. We divide the transfer process into four stages, generating lateral connections only in Stage1, Stage2, and Stage3 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 flexible connection structure allows you to replace the entire network with different models according to your needs.

$Source\_task1$,..,$Source\_taskn$ are pre-trained strategies $\langle \pi_1, \ldots, \pi_n \rangle$ 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

$$SO^j = \left(SO_1^j, \cdots, SO_n^j \right) \tag{3}$$

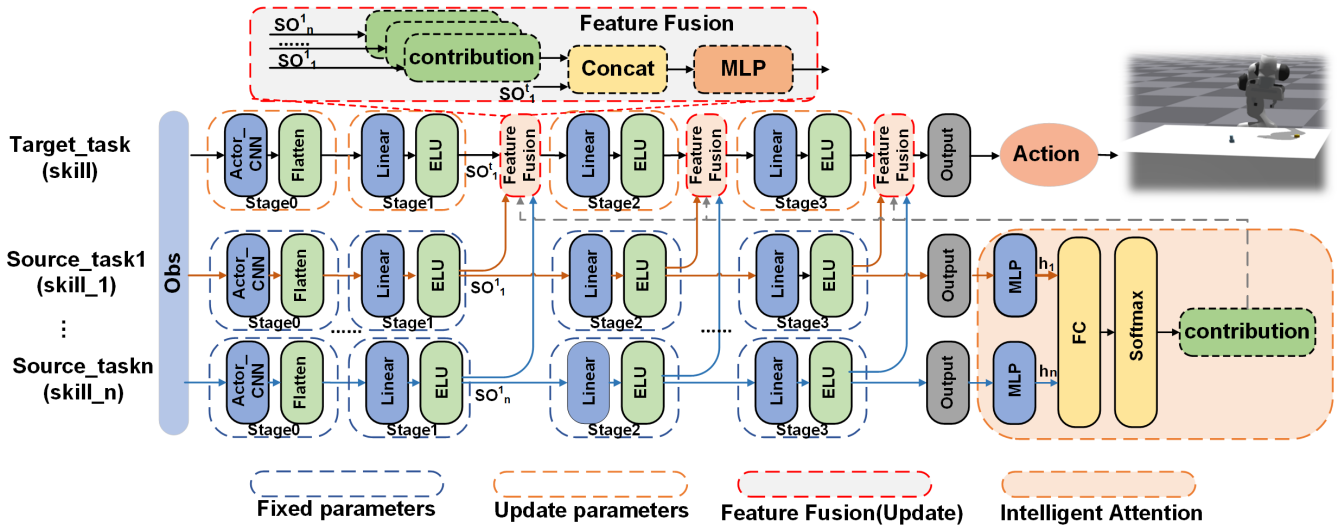


Fig. 3. Overview of the ITN. ITN connects the models of source tasks horizontally. The Intelligent Attention module processes source strategies and outputs contributions. The Feature Fusion model fuses features from each source task. The final output of the network is the strategy for the target task.

In the feature fusion layer, $W = w_1, \ldots, w_n$ is the output of Intelligent Attention in Eq. 2, representing the contribution values of the different source tasks. W acts on SO^j and U_j is generated

$$U_j = SO^j \otimes W = \left(SO_1^j \cdot w_1, \ldots, SO_n^j \cdot w_n\right) \tag{4}$$

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

$$F_j = \left(SO_1^j \cdot w_1, \ldots, SO_n^j \cdot w_n, SO_t^j\right) \tag{5}$$

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

$$I_{j+1} = MLP\left(F_j; \theta_{MLP_j}\right) \tag{6}$$

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 tasks. The features are propagated backward, layer by layer, until the output layer outputs the action

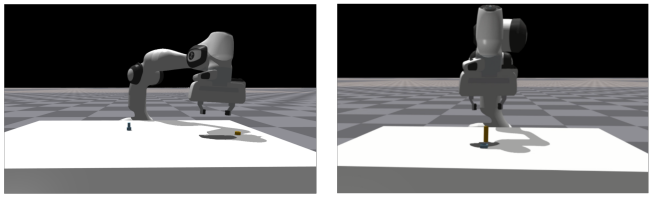
$$A = f\left(MLP\left(I_l\right); \theta_{out}\right) \tag{7}$$

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 target task.

IV. EXPERIMENTS

Our experiments focus on answering the following questions: 1) Is the reward function generated by the generator more effective? 2) How is the performance and generality of ITN? 3) How does ITN's performance in operation tasks compare to other state-of-the-art methods?



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

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

TABLE I. DETAILED INFORMATION ABOUT TASKS

Type of task	Task commands	Sub-skills
B-Skill	Pick up the nut.	Pick
B-Skill	Place the nut on the bolt.	Place
B-Skill	Screw the nut onto the bottom of the bolt.	Screw
…	…	…
B-Skill	Insert peg into the hole.	Insert
C-Skill	Pick up the nut and place it on the bolt.	Pick-Place
C-Skill	Pick up the nut, place it on the bolt and screw it to the bottom of the bolt.	Pick-Place-Screw
C-Skill	Pick up the nuts and place them on the bolts.	Pick-Place-Pick-Place
…	…	…
C-Skill	Pick up the round peg and insert it into the round hole.	Pick-Place-Insert

Experimental Environment: We used the Isaac Gym [25] simulator to implement environmental modeling and conducted reinforcement learning experiments. We designed two main tasks: the *NutBolt* task and the *PegHole* task. Fig. 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. Table I lists some basic skills (B-Skill) and complex skills (C-Skill).

Experimental Setup: To ensure the method's reliability, we used five random seeds for 10 experiments and calculated the 95% confidence interval for each experiment. The A2C [24] algorithm was employed for policy learning. 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. Experiment on Reward Function Generation

We conducted a series of evaluations on various manipulation tasks, testing the generator's ability to generate reward functions for both basic and complex skills. The experiment used the GPT-4-0613 variant as the intelligent generator.

Baseline: We compared the performance of the reward function generated by our method, the manually designed reward function, and the Eureka-generated reward function in reinforcement learning tasks.

Training Details: The generator performed five iterations per task to generate the reward functions. Each iteration involved 10 samples and selected 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.

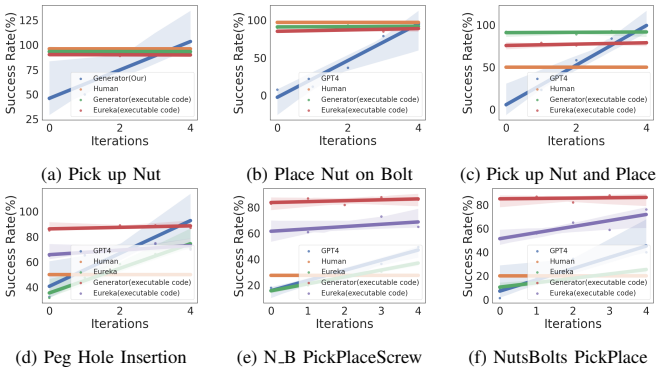
Results: As shown in Fig. 5, comparing the reward functions generated by generator(our) with human-designed ones revealed that generator(our) outperformed in both basic and complex tasks. In complex tasks, the success rate of its reward functions significantly exceeded that of human-designed ones as iterations progressed due to its deep understanding of task requirements and strong analytical, computational, and design capabilities. Compared to Eureka [4], the reward functions generated by the generator(our) achieved higher execution rates because it was based on a fine-tuned LLaMA [22], which generated more accurate reward_components. Especially in complex tasks, the reward functions generated by the generator(our), with precise reward_components, significantly outperformed those from Eureka regarding success rate.

B. Intelligent Transfer Experiments

This section provided a comprehensive evaluation of ITN. An 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 conducted to analyze ITN's performance compared to other leading methods. All experiments in this section utilized generator-generated reward functions. We present the results in the form of graphs for the *NutBolt_PickPlace* task and the *PegHole_PickPlaceInsert* 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:

- $No\_Transfer$ does not use pre-learned skills but learns from scratch through exploration.



(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. 5. The reward functions generate experimental quantitative results.

- *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—imitating the lack of prior knowledge in the Skill Space.

The results are shown in Fig. 6. The learning speed (the epoch at which the curve reached stability) of ITN was significantly better than No_Transfer in both tasks.PST simulates the absence of prior skills. The results demonstrated that ITN learns much faster than No_Transfer with just a few prior skills, indicating that ITN can rapidly acquire new skills that are not based on existing ones. This capability is a significant goal and breakthrough of our work, as most previous studies have relied on fixed skill combinations. Comparing the performance of ITN with NIA demonstrated the superior effectiveness of integrating an Intelligent Attention model over a fixed value. The performance comparison between NIA and DFS confirmed the strong feature fusion capability of the feature fusion layer. ITN improved time efficiency by approximately 72.22% and 65.17% compared to No_Transfer in the respective tasks.

2)Generalization Experiment: This section examined the cross-task transfer capability and generalization of ITN in a dynamic environment. We designed five scenarios for each task. Take the *NutBolt_PickPlace* task as an example:

- $ITN(PegHole\_Pick)$ refers to transferring only the $PegHole\_Pick$ skill to the $NutBolt$ task.
- $ITN(PegHole\_Place)$ refers to transferring only the $PegHole\_Place$ skill to the $NutBolt$ task.
- $ITN(Manipulate\_NewParts)$ uses the learned $PegHole\_Pick$ and $PegHole\_Place$ skills in the $NutBolt$ task to operate a new part of the same type.
- $ITN(Multi-skilled)$ is to simultaneously transfer the previously learned $NutBolt\_Pick$ and $NutBolt\_Place$ skills in the new $NutBolt$ task.

The experimental results are illustrated in Fig. 7. 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. Based on the ac-

tion feature level, this transfer effectively addressed various manipulation problems encountered by robots in dynamic tasks. Curve 4 indicated that the robot quickly achieved high rewards when operating unseen new parts, highlighting ITN's efficiency in handling task variations. 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 multiple-skill fusion strategy was the most effective, significantly enhancing the learning efficiency and performance of the robot manipulation tasks.

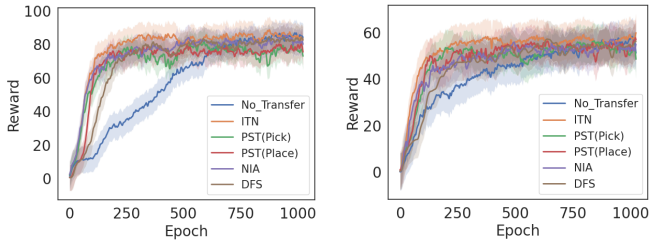


Fig. 6. Results of ablation experiment. Left: *NutBolt_PickPlace* task. Right: *PegHole_PickPlaceInsert* task.

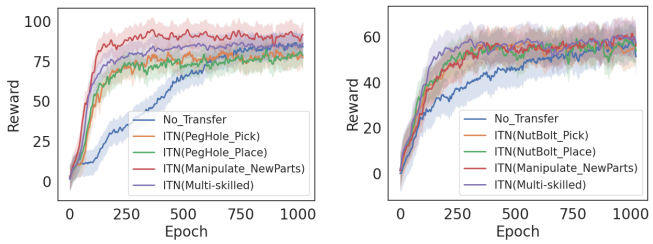


Fig. 7. Results of generalization experiment. Left: *NutBolt_PickPlace* task. Right: *PegHole_PickPlaceInsert* task.

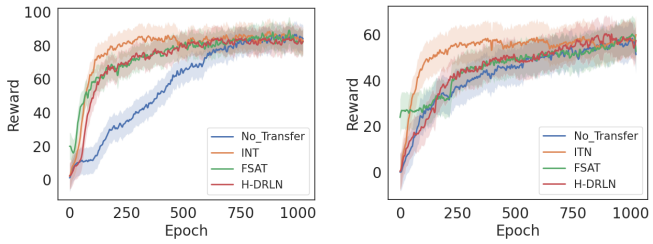


Fig. 8. Results of comparison experiment. On the left is the comparison experiment for the task *NutBolt_PickPlace*, and on the right is the comparison experiment for the task *PegHole_PickPlaceInsert*.

3)Comparison Experiment: For comparison, we chose two representative advanced methods significantly enhancing robot skill learning.

Hierarchical Deep Reinforcement Learning Network (H-DRLN) [26] 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 (FSAT) [21] facilitated knowledge transfer by extracting shared features between old and new tasks. We replicated these approaches and conducted the comparison in our experimental setup.

The results are shown in Fig. 8. Several methods of learning significantly outperformed direct learning. Although FSAT 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. FSAT 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 learning 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, 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%**
FSAT	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_PickPlaceInsert					
No_Transfer	50.49%	65.79%	71.91%	78.69%	**90.73%**
FSAT	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

This paper presents an Intelligent Transfer System (ITS) that allows robots to autonomously and quickly learn new tasks in a dynamic environment. The system is divided into two parts to enable autonomous problem-solving. Firstly, LLMs generate a process-oriented reward function based on the reward_components for the new task. Secondly, an Intelligent Transfer Network (ITN) is constructed to fuse features from source skills, accelerating the learning of the new task. We conducted a series of simulation experiments to verify the efficiency and superiority of our method. ITS can autonomously learn new and unseen skills and utilizes related task experience to accelerate the learning process, a capability humans have long hoped robots could possess. In addition, ITS has strong cross-task transfer ability, allowing it to adapt well to task changes. It can handle combinations of existing tasks and efficiently learn skills it has never encountered before. For future work, we will first improve the generator's prompt design to enhance the reward design's speed and success rate for more complex robotic tasks. Then, we plan to conduct cross-domain experiments by applying ITS to more challenging service robotics fields.

REFERENCES

[1] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9493–9500.

[2] M. Xu, P. Huang, W. Yu, S. Liu, X. Zhang, Y. Niu, T. Zhang, F. Xia, J. Tan, and D. Zhao, "Creative robot tool use with large language models," arXiv preprint arXiv:2310.13065, 2023.

[3] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu, "Text2reward: Reward shaping with language models for reinforcement learning," arXiv preprint arXiv:2309.11489, 2024.

[4] 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:2310.12931, 2024.

[5] 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.

[6] 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.

[7] 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," arXiv preprint arXiv:1707.08817, 2018.

[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," arXiv preprint arXiv:2401.04334, 2024.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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, pp. 10 141–10 152.

[18] 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," arXiv preprint arXiv:1901.10964, 2019.

[19] 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, 21st IFAC World Congress.

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

[21] 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.

[22] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," arXiv preprint arXiv:2302.13971, 2023.

[23] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," arXiv preprint arXiv:1908.10084, 2019.

[24] 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.

[25] 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," arXiv preprint arXiv:2108.10470, 2021.

[26] 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.