Task-oriented adaptive learning of robot manipulation skills

Kexin Jin[1],Guohui Tian[1†] ,Bin Huang[1] ,Yongcheng Cui[1] and Xiaoyu Zheng[1]

[1] School of Control Science and Engineering, Shandong University, Jinan, China
(E-mail: kx.jin@mail.sdu.edu.cn; g.h.tian@sdu.edu.cn; huangbin@sdu.edu.cn)

Abstract: In industrial environments, robots are confronted with constantly changing working conditions and manipulation tasks. Traditional approaches that require robots to learn skills from scratch for new tasks are often slow and heavily dependent on human intervention. To address inefficiency and enhance robots' adaptability, we propose a general Intelligent Transfer System (ITS) that enables autonomous and rapid new skill learning in dynamic environments. ITS integrates Large Language Models (LLMs) with transfer reinforcement learning, harnessing both the advanced comprehension and generative capabilities of LLMs and the pre-acquired skill knowledge of robotic systems. First, to achieve comprehensive understanding of unseen task commands and enable autonomous skill learning in robots, we propose a reward function generation method based on task-specific reward components. This approach significantly improves time efficiency and accuracy while eliminating the need for manual design. Secondly, to accelerate the learning speed of new robotic skills, we propose an Intelligent Transfer Network (ITN) within the ITS. Unlike traditional methods that merely reuse or adapt existing skills, ITN intelligently integrates related skill features, significantly enhancing learning efficiency through synergistic knowledge fusion. We systematically evaluate our method in the simulation environment. The results demonstrate that the system can efficiently learn unseen skills without relying on pre-programmed behaviors, achieving significant improvements in time efficiency—specifically, 72.22% and 65.17% faster learning for two major tasks compared to learning from scratch. Supplementary materials are accessible via our project page:*https://jkx-yy.github.io/*

Keywords: LLMs, Transfer reinforcement learning, Industrial robotics, Automation, Manipulation

1 INTRODUCTION

In real-world applications, robots must operate in dynamically changing environments and adapt to continuously evolving tasks. To execute tasks effectively without human intervention, they need to autonomously and rapidly learn new skills. While existing research has explored aspects of autonomy and adaptability separately, designing an effective, automated, and generalisable learning framework for unknown manipulation tasks remains a significant challenge.

Firstly, Integrating Large Language Models (LLMs) with robots has shown great potential in enhancing autonomous learning capabilities. For instance, Code as Policies [1] generates robot action strategies from natural language commands, RobotTool [2] enables robots to interpret natural language instructions and creatively utilize tools to complete complex tasks. However, both methods heavily rely on predefined control primitives, limiting their ability to handle new skills beyond the scope of these primitives and restricting them to merely combining existing skills. To achieve true creativity without relying on pre-programmed code, Text2Reward [3] generates reward functions from language commands to guide robot skill learning through reinforcement learning. While this approach significantly enhances the robot's ability to innovate, its dependence on human feedback compromises the autonomy of the skill-learning process. Eureka [4] addresses this limitation by introducing a self-optimization mechanism to generate optimal reward functions. However, its lack of domain-specific understanding results in low efficiency in autonomously generating reward components, ultimately reducing the overall efficiency of producing executable reward functions. To address the challenges faced by LLMs in generating task reward functions, such as limited creative capacity, heavy reliance on human intervention, and low operational efficiency, we propose a task-oriented general reward function generator. Our approach leverages automatically generated, high-precision reward components to enable more efficient reward function generation.

Secondly, some studies have made significant progress in enhancing robot new skill learning in specific contexts. For complex manipulation tasks, Hierarchical Reinforcement Learning (HRL) utilizes high-level strategies to reuse underlying skills, thereby accelerating the learning process by leveraging knowledge from previously acquired skills [5],[6]. However, such methods lack adaptability to new or unknown tasks, limiting their generalizability. Recent advancements combine HRL with imitation learning to address vision-based long-horizon robot manipulation [7]. Despite their potential, these approaches are constrained by offline dataset limitations and require task-specific meta-

controllers, reducing their flexibility. Another line of research, using synthetic data to augment the training set, enabling the learning of task-parameterized skills with only a small number of expert demonstrations [8]. Nevertheless, the performance of these methods heavily depends on the quality and quantity of demonstration data, making them vulnerable to suboptimal or limited datasets. In summary, existing skill enhancement techniques exhibit several limitations, including a high dependency on pre-training strategies, sensitivity to environmental changes, and significant time consumption. Moreover, they are often tailored to specific tasks, lacking the generalizability for cross-task transfer, and are unable to autonomously learn new tasks involving previously unlearned skills. To address these limitations, this paper proposes a general transfer learning framework that goes beyond simple skill combination. By intelligently fusing relevant features from previously acquired skills, our approach accelerates the learning of new tasks while maintaining the ability to learn entirely novel skill combinations that have never been encountered before. This innovative fusion mechanism enables the system to generalize across tasks and autonomously learn complex, previously unseen skills.

We propose an Intelligent Transfer System (ITS), as illustrated in **Fig. 1**. The system comprises three core components: (1) an automatically generated process-oriented reward function, which dynamically constructs task-specific reward functions based on reward components to guide autonomous task solving; (2) a dynamically expandable Skill Space that organizes and stores learned skills, enabling efficient extraction of source skills relevant to target tasks; and (3) an Intelligent Transfer Network (ITN) that extracts and transfers knowledge from source skills to accelerate the learning of target tasks.

We summarize the three key contributions of this work:

(1) A novel Intelligent Transfer System (ITS) integrating LLMs and transfer learning, enabling robots to autonomously and rapidly learn unknown tasks without human intervention or predefined action primitives.

(2) A reward function generation method leveraging highly accurate reward components, outperforming state-of-the-art approaches in task comprehension, executability, and accuracy, while significantly reducing generation time.

(3) An Intelligent Transfer Network (ITN) is proposed, incorporating federated learning principles and featuring a dynamically adaptable structure. It fuses features from diverse source skills to accelerate new skill learning, supporting both the recombination of existing skills and the learning of entirely novel task combinations, even for previously unseen scenarios.

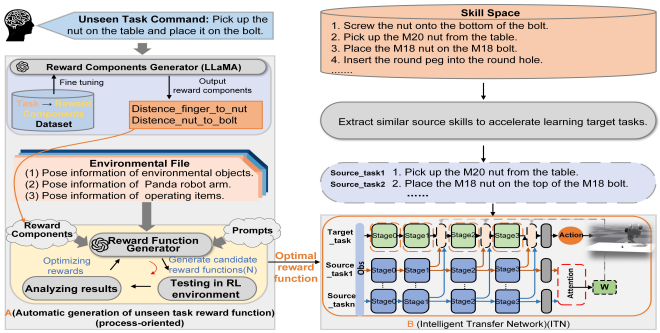


Fig. 1. Overview of ITS

2 RELATED WORK

2.1 Reward function generation based on LLMs

In reinforcement learning, the reward function serves as a critical mechanism that guides the agent's behavior by providing immediate feedback, enabling the optimization of its policy. An effectively designed reward function plays a pivotal role in guiding the agent to accomplish task goals, while an inadequately designed one can result in suboptimal decision-making or inefficient actions [9]. Recently, the automatic generation of reward functions using LLMs has emerged as a promising research direction. This approach reduces reliance on manual design. For example, Ma et al. [4] leveraged the generative capabilities of LLMs to achieve human-level reward generation through iterative trial-and-error refinement, while Yu et al. [10] employed template-based prompting to enhance the success rate and accuracy of reward function generation. Additionally, Li et al. [11] introduced LLM-based modules, including reward designers, critics, and trajectory analyzers, to create, validate, and debug reward functions. However, these methods often fail to address LLMs' limitations in understanding complex, domain-specific, or ambiguous task commands. The time-consuming process of trial-and-error corrections, along with excessive reliance on human feedback, further undermines the efficiency of autonomous reward function generation. To overcome these issues, we propose a reward component abstraction method based on task command. This method enhances higher-level task understanding, simplifies reward design, and significantly improves efficiency in handling complex and ambiguous commands, while reducing the time required to generate high-quality reward functions.

2.2 Transfer reinforcement learning in robotic skills

Transfer reinforcement learning integrates the principles of reinforcement learning and transfer learning. A reinforcement learning task is formally defined as a tuple $< S, A, P, R, \gamma >$ where S denotes the set of states representing the environment and agent, A represents the set of actions available to the agent, $P(s'|s,a)$ is the transition probability from state s to state s' after taking action a, $R(s,a)$ is the immediate reward obtained after executing action a in state s, and $\gamma \in [0,1]$ is the discount factor. The objective is to learn an optimal policy $\pi^*(a|s)$ that maximizes the cumulative rewards over time. Reinforcement learning has been extensively applied to robotic manipulation tasks. For instance, Zhang et al. [12] employed graph neural networks to represent states and actions, optimizing collaborative human-robot assembly through reinforcement learning. Schoettler et al. [13] explored meta-reinforcement learning to solve simulated industrial insertion tasks and adapt policies to real-world scenarios efficiently. Additionally, Gao et al. [14] and Xu et al.[15] demonstrated the effectiveness of reinforcement learning in stacking and grasping tasks.

The goal of transfer learning is to leverage the agent's source domain policies $\pi_s = \{\pi_1, \ldots, \pi_n\}$ to facilitate learning in the target domain π_t. For example, Shin et al. [16] and Tao et al. [17] utilized source domain knowledge for action generation through policy reuse or distillation. Barreto et al. [18] and Zhang [19] improved target task training by decoupling or reusing task-invariant features. Recent studies have further explored the integration of transfer learning with reinforcement learning to tackle real-world robotics problems. Men et al. [20] proposed a Policy Fusion Transfer framework to enhance generalization and efficiency in robotic assembly. Jin et al. [21] applied feature selection and adaptive transfer learning to generalize peg-in-hole assembly across varying geometric features. Most existing approaches focus on generalizing within the same manipulation tasks and fail to address cross-task transfer between different task types. In contrast, our work aims to handle a variety of evolving tasks, enabling efficient and effective knowledge transfer across diverse manipulation.

3 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. Additional details, such as dataset information, the reward optimization process, ITS pseudocode, and experimental video materials, are available on the project website.

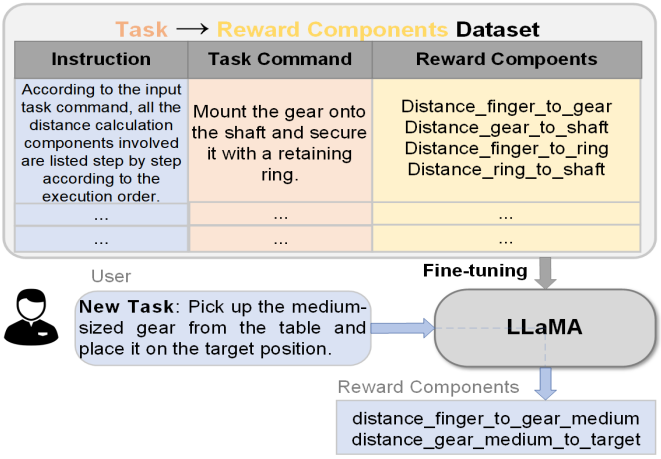


Fig. 2. Schematic of reward components generation

3.1 Automatic generation of task-oriented reward

3.1.1 Reward components generation

LLMs face challenges in processing task commands involving ambiguous dependencies or missing action subjects. For instance, given a command like "Assemble the bolt and nut on the table," they often fail to infer implicit sequences (e.g., "finger → nut, nut → bolt") or capture temporal logic in more complex tasks. While LLMs excel at task decomposition, they are less effective at fine-grained atomic action planning. As a result, directly using LLM-generated reward functions can lead to unreliable outcomes and improper execution, reducing the efficiency of reward function generation. To obtain precise atomic action mappings for tasks (e.g., " finger → nut, nut → bolt"), we constructed a task-to-reward components mapping dataset. This dataset comprises approximately 10,000 industrial assembly commands, covering common items such as gears and connectors. The commands are sourced from search engines, human-created examples, and GPT-generated inputs, with all mapping pairs rigorously reviewed and filtered. As shown in **Fig. 2**, the reward components for each task consist of computation components (e.g., distances between the manipulator, object and target). Through fine-tuning the open-source LLaMA model [22] on our carefully constructed dataset, we enhance its capability to generate precise reward components. These components are defined as atomic action sequences that represent the correct execution order of tasks, thereby significantly improving the efficiency of reward function generation.

3.1.2 Generation of reward function

We design a four-part prompt for the reward function generator (using a GPT-4-0613 variant) in **Fig. 1**, with the corresponding prompt illustrated in **Fig. 4**. The system

information describes the system's identity and goals, the environment information includes observation data and the positional details of manipulated objects and the Panda arm. To ensure high enforceability and success rates, useful prompts are designed, including the proposed reward form, constraint hints, and guidance for generating reward functions based on reward components. Additionally, strategy optimization information provides analyzable metrics for generating improved reward functions.

3.1.3 Optimization of reward function

In each iteration, the generator produces multiple reward function candidates based on the sample size N, and regenerates them if the initial generation fails. To ensure the reward function correctly guides the robot to complete skills sequentially, we record the maximum task success rate, the value of each reward component, and the total reward for complex skills in each iteration. Using these N sets of data, the LLM analyzes the parameter values of each component in the next iteration to maximize the total cumulative reward, achieving optimization. Through this self-optimization process, the success rate and accuracy of the reward function are continuously improved.

3.2 Skill space construction & source skills extraction

To accelerate the learning of new tasks, useful source skills are stored and extracted in a real-time expandable Skill Space, which includes both basic (e.g., Pick, Place, Screw, Insert) and complex skills (e.g., assembly tasks composed of multiple basic skills). Each skill in the Skill Space is defined by four attributes: serial number, task description, pre-trained model, and action sequence. We design a skill extraction method based on SBERT [23], which extracts semantically similar source skills for the target task. Compared to other methods, this approach is robust, extracts deeper source skills, and offers strong interpretability.

3.3 Intelligent transfer network (ITN)

In this section, we employ the reward function generated in 3.1 to enable cross-task skill transfer, as illustrated in **Fig. 4**. Our proposed framework accelerates the acquisition of new skills by integrating features from semantically related source skills within the Skill Space. The core innovations of our approach include: (1) a novel multi-task attention mechanism that dynamically adjusts the weight values of different source skills., and (2) a dynamically expandable skill transfer network that leverages feature extraction and fusion to establish lateral connections across source tasks, thereby enhancing the adaptability and efficiency of knowledge transfer between tasks.

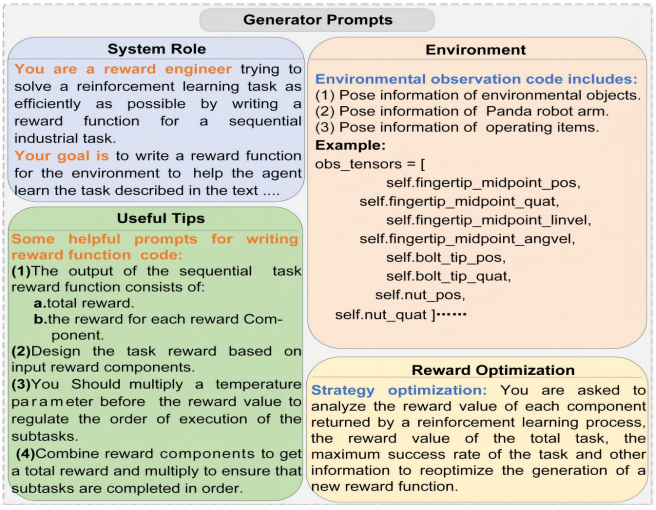


Fig. 3. Prompts for the reward function generator

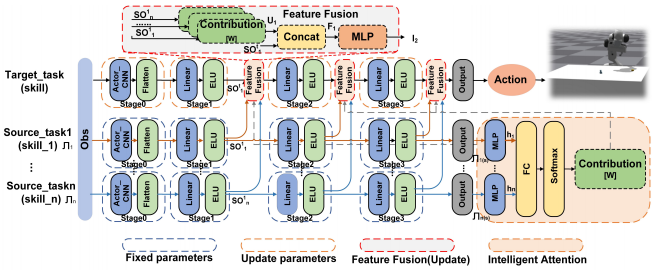


Fig. 4. Overview of the ITN

3.3.1 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. To address this, we propose Intelligent Attention, which dynamically assigns contributions to relevant source strategies. The structure of Intelligent Attention is shown in the bottom right of **Fig. 4**.

The source policies related to the target task are $< \pi_1, \ldots, \pi_n >$. Given the input environmental observation Obs (including the pose information of the objects and the Panda arm), the output is $< \pi_1(s), \ldots, \pi_n(s) >$. Each policy is processed through a Multi-Layer Perceptron (MLP), and the output for the j-th source policy is:

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

the parameter θ_j represents the weights of the MLP. The output of the MLP is further processed through a Fully Connected Layer (FC) and a softmax function 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.

3.3.2 Feature extraction and fusion

The Intelligent Transfer Network (ITN) introduces a novel approach by extracting and transferring features from source tasks to the target task. A key innovation is the use of lateral connections, inspired by federated learning[24], which significantly enhances the efficiency of feature transfer. This unique structure enables the agent to simultaneously leverage the experience of source tasks and learn new features, thereby allowing ITN to acquire skills beyond the existing Skill Space, which is a capability not supported by traditional methods.

We employ the Advantage Actor-Critic (A2C) [25] reinforcement learning algorithm in our experiments. As shown in **Fig. 4**, the network structure is like Source_task1 and Source_taskn. Notably, the number of source tasks is dynamically adjustable based on the target task and the number of source skills extracted from the Skill Space, ensuring scalability and adaptability. The transfer process is divided into four stages, with lateral connections generated only in Stage 1, Stage 2, and Stage 3 to extract features at different skill levels. To fuse features from the target task with those of source tasks, a feature fusion module is added after each feature extraction stage in the target task network. This flexible and modular connection structure allows the replacement of the entire network with different models as needed, providing a highly adaptable framework for cross-task skill transfer.

The components Source_task1,…,Source_taskn represent pre-trained strategies $< \pi_1, \ldots, \pi_n >$ with fixed parameters. For the source tasks, the output of the j-th stage of the i-th task is denoted as SO_i^j, and the total output of the j-th stage across all source tasks is:

$$SO^j = (SO_i^j, ...SO_n^j) \tag{3}$$

In the feature fusion layer, $W = \{w_1, \ldots, w_n\}$ is the output of Intelligent Attention in (2), representing the contribution values of the different source tasks. W acts on SO^j to generate U_j, as follows:

$$U_j = SO^j \otimes W = (SO_i^j.w_1, ..., SO_n^j.w_n) \tag{4}$$

U_j is concatenated with the target output SO_t^j in the j-th stage to obtain the joint feature matrix F_j,as follows:

$$F_j = (SO_i^j.w_1, ..., SO_n^j.w_n, SO_t^j) \tag{5}$$

the joint feature matrix F_j is processed through an MLP module to serve as input to the (j+1)-th stage:

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

the parameter θ_{MLP_j} denotes the weights of the MLP in the j-th feature fusion block. Through the feature fusion block, the features of the source tasks are fused according to task importance while preserving the results of the target task network (BASE). This enables the entire network to learn new tasks effectively. The features are propagated backward layer by layer until the output layer generates the final action:

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

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

During the training of 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 throughout the training process.

4 EXPERIMENT

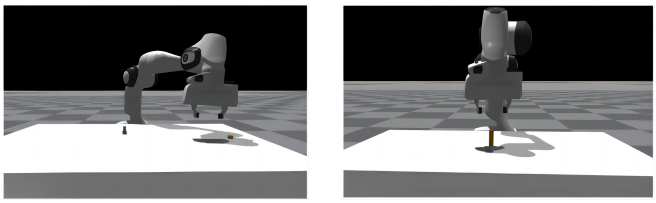
Our experiments aim to address the following key questions: 1. Is the reward function generation method based on reward components more effective than traditional and state-of-the-art approaches? 2. How does our proposed ITN perform in accelerating skill learning and generalizing across tasks, particularly its ability to support both similar-task and cross-task transfer, even for unseen composite tasks? 3. How does ITN compare to other state-of-the-art methods in terms of efficiency and superiority?

4.1 Experimental environment

In this study, we employed the Isaac Gym [26] simulator for environmental modeling and conducted reinforcement learning experiments. To comprehensively evaluate the system's performance and adaptability, we designed two core tasks: the Nut-Bolt Assembly Task (*NutBolt*) and the Peg-Hole Insertion Task (*PegHole*). **Fig. 5** presents a detailed illustration of the simulation world modeling. In the simulation environment, a highly dexterous 7-degree-of-freedom robotic manipulator, the Franka Emika Panda, was employed to perform object manipulation tasks. **Table 1** provides a list of Basic skills (B-Skill) and complex skills (C-Skill).

4.2 Experimental setup

To rigorously validate the reliability of the proposed method, we conducted 10 independent experiments, each initialized with five distinct random seeds, and calculated the 95% confidence intervals for the results. Policy optimization was carried out using the A2C algorithm. All computations were performed on a high-performance workstation equipped with an NVIDIA RTX A5000 GPU and the Ubuntu 20.04 operating system. The experimental framework was implemented using the PyTorch library.



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

Fig. 5. Experimental environment

Table 1. 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

4.3 Experiment on reward function generation

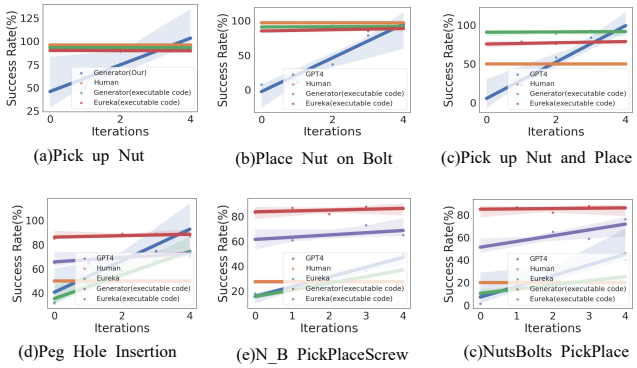
We conducted a comprehensive series of evaluations across a range of manipulation tasks to assess the generator's capability in generating reward functions for both B-skills and C-skills. The experiments were conducted using the GPT-4-0613 variant as the intelligent reward function generator.

4.3.1 Baseline

We conducted a comparative analysis of the performance of reward functions generated by our proposed method (reward functions generated based on reward components (**GPT4**)), manually designed reward functions(**Human**), and those generated by the state-of-the-art **Eureka [4]** framework in reinforcement learning tasks.

4.3.2 Training details

The generator executed five iterations per task to generate reward functions. In each iteration, 10 candidate reward functions were sampled, and the one that maximized the average task success rate was selected for further optimization. Upon completion of all iterations, the optimal reward function was identified and designated as the final output of the generator.



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

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

Fig. 6 . The reward functions generate experimental quantitative results

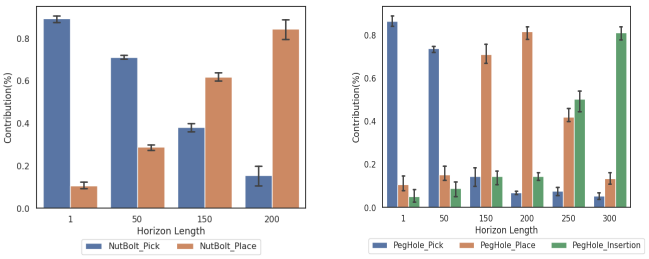


Fig. 7. Attention allocation.

4.3.3 Results

As illustrated in **Fig. 6**, a comparison between the reward functions generated by our proposed method and those manually designed by humans demonstrates that our approach consistently outperforms human-designed reward functions in both basic and complex tasks. In complex tasks, the success rate of the reward functions generated by our method significantly surpasses that of human-designed ones as iterations progress. This is attributed to our method's profound understanding of task requirements, as well as its robust analytical, computational, and design capabilities. Furthermore, compared to the Eureka framework, the reward functions generated by our method achieve higher execution rates. Executability rate measures the syntactical correctness and compatibility of the reward function as the ratio of executable reward functions to the total generated This improvement stems from the fine-tuned LLaMA model employed in our method, which generates more accurate reward components. Notably, in complex tasks, the reward functions generated by our method, leveraging precise reward components, exhibit a significantly higher success rate than those generated by Eureka.

Fig. 7 illustrates ITN's policy allocation dynamics, showing the changes in attention values assigned to different subskills during a cycle in the *NutBolt* and *PegHole* tasks. Data from a single interaction cycle, collected after training stabilized, reveal that ITN

dynamically shifts its focus to different atomic skills at various stages of the target task. This attention distribution closely mirrors human experiential patterns, demonstrating ITN's ability to emulate human-like task adaptation. By dynamically adjusting attention values for source skills, ITN effectively adapts to diverse tasks, significantly accelerating the robot's mastery of new tasks.

4.4 Intelligent transfer experiment

This section presents a comprehensive evaluation of the ITN framework through a series of systematically designed experiments. Ablation experiment was conducted to evaluate the contribution of each module, while generalization experiments were performed to validate the framework's generalization ability and cross-skill transfer capability. Additionally, comparative experiments were carried out to benchmark ITN against other state-of-the-art methods. All experiments utilized reward functions generated by the proposed generator, and the results are visualized in graphical form, focusing on the *NutBolt_PickPlace* task and the *PegHole_PickPlaceInsert* task (the first and last complex tasks listed in **Table 1**).

4.4.1 Ablation experiment

For comparison, we adopted several widely recognized baselines commonly employed in transfer reinforcement learning research.

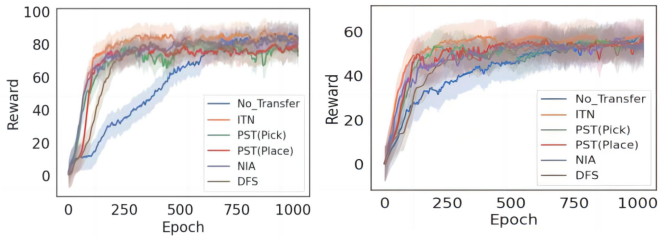
(1) *No_Transfer* does not use pre-learned skills but learns from scratch through exploration.

(2) *No Intelligent Attention (NIA)* is the proposed ITN model without the Intelligent Attention component, assigning a fixed contribution value to each subtask.

(3) *Direct Feature Summation (DFS)* is the proposed ITN model without Intelligent Attention and Feature Fusion, directly summing features from different source skills.

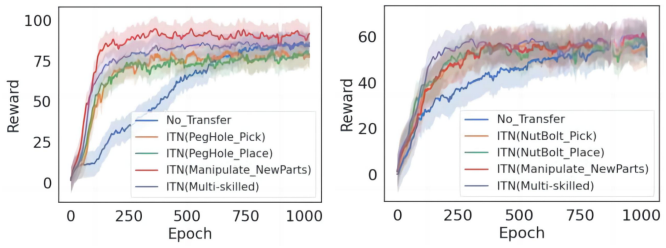
(4) *Partial Skills Transfer (PST)* is the proposed ITN method that transfers only a few subskills—imitating the lack of prior knowledge in the Skill Space. Specifically, *PST(Pick/Place)* indicates that only the atomic Pick or Place skill from the Skill Space is utilized for transfer to the target task.

The experimental results are presented in **Fig. 8**. ITN exhibited significantly faster learning speeds (measured by the epoch at which the curve stabilized) compared to *No_Transfer* in both tasks. *PST*, which simulates scenarios with limited prior skills, demonstrated that ITN significantly outperforms *No_Transfer* in learning speed, by efficiently leveraging a small subset of mastered skills to accelerate the acquisition of unseen tasks. This capability represents a significant advancement and a key contribution



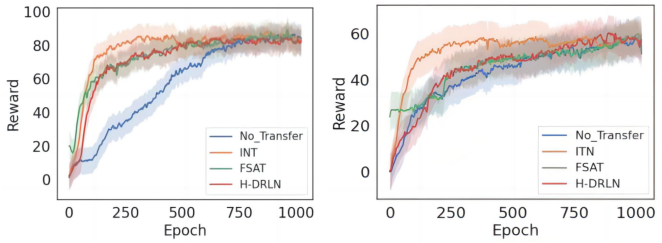
(a) NutBolt_PickPlace (b) PegHole_PickPlaceInsert

Fig. 8. Results of ablation experiment



(a) NutBolt_PickPlace (b) PegHole_PickPlaceInsert

Fig. 9. Results of generalization experiment



(a) NutBolt_PickPlace (b) PegHole_PickPlaceInsert

Fig. 10. Results of comparison experiment

of our work. Unlike most prior studies that rely on fixed skill combinations, ITN introduces a dynamic and adaptive approach to skill acquisition, enabling more flexible and efficient learning in complex tasks. Furthermore, the comparison between ITN and *NIA* underscores the superior effectiveness of integrating an Intelligent Attention mechanism over fixed-value assignments. Additionally, the performance gap between *NIA* and *DFS* validates the robust feature fusion capability of the feature fusion layer. In terms of time efficiency, ITN achieved improvements of approximately 72.22% and 65.17% over *No_Transfer* in the respective tasks.

4.4.2 Generalization experiment

This section investigates the cross-task transfer capability and generalization performance of ITN in dynamic environments. For each task, five distinct scenarios were designed. Taking the *NutBolt_PickPlace* task as an example:

(1) *ITN(PegHole_Pick)* refers to transferring only the *PegHole_Pick* skill to the *NutBolt_PickPlace* task.

(2) *ITN(PegHole_Place)* refers to transferring only the *PegHole_Place* skill to the *NutBolt_PickPlace* task.

(3) *ITN(Manipulate_NewParts)* transfers the learned *PegHole_Pick* and *PegHole_Place* skills to the *NutBolt_PickPlace* task to manipulate a new part of the same type.

(4) *ITN(Multi-skilled)* transfers the previously learned *NutBolt_Pick* and *NutBolt_Place* skills to the new *NutBolt_PickPlace* task.

The experimental results are illustrated in **Fig. 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. Based on the action feature level, this transfer can effectively address 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.

4.4.3 Comparison experiment

For comparison, we chose two representative advanced methods significantly enhancing robot skill learning. The Hierarchical Deep Reinforcement Learning Network (H-DRLN) [27] efficiently retains knowledge through skill distillation and deep skill arrays, enabling the model to dynamically decide whether to execute original actions or reuse pre-learned skills. The Feature Selection Adaptive Transfer (FSAT) [21] facilitates knowledge transfer by extracting shared features between source and target tasks. We replicated these methods and integrated them into our experimental framework for comparative analysis.

The experimental results are presented in **Fig. 10**. Several methods significantly outperformed direct learning. Although FSAT achieved the best performance in the early training phase by selecting data closer to the target domain to initialize the policy, it was surpassed by ITN after several dozen iterations. This highlights a key advantage of ITN: while FSAT excels in extracting shared features and solving transfer problems across manipulated objects, ITN focuses on a generalized transfer approach across tasks, enabling broader applicability and adaptability. H-DRLN attempted to map the current environment state to the probabilities of new actions and source skills, reusing the subskill with the highest probability value. However, this approach, while incorporating new strategy learning, was limited in its applicability due to its reliance on specific skills. In contrast, ITN is not constrained by specific skills and demonstrates full adaptability to task changes, representing a significant advancement over existing methods.

As shown in **Table 2**, the success rates of all methods improved as training progressed. However, ITN achieved a higher success rate at an earlier stage, demonstrating significantly superior time efficiency. This early performance advantage highlights ITN's practical value and its potential for real-world applications.

Table 2. The task success rate of the four methods va ries with the training cycle.

Method	200epo	400epo	600epo	800epo	1024epo
NutBolt_PickPlace					
No_Transfer	34.66%	56.32%	81.03%	90.64%	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_PickPlaceInsert					
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%**

5 CONCLUSION

This paper introduces an Intelligent Transfer System (ITS), a novel framework enabling robots to autonomously and efficiently learn new tasks in dynamic environments. The key innovation is its process-oriented reward function generation based on reward components, significantly improving reward design efficiency. We also propose the Intelligent Transfer Network (ITN), a task-adaptive, universally transferable architecture with a horizontally connected federated learning framework and attention mechanisms, enabling effective feature fusion and cross-task adaptability. ITS is the first general-purpose cross-task transfer method we propose, overcoming the domain limitations of previous approaches. By avoiding fixed skill combinations and domain-specific constraints, it enables seamless knowledge transfer across diverse tasks. Experiments demonstrate ITS's faster learning speeds and robust adaptability to task variations. Future work will focus on extending ITS to other robotics domains, and addressing the sim2real challenge to enable real-world applications.

Acknowledgments 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.

REFERENCES

[1] Liang J, Huang W, Xia F, et al (2023), Code as policies: Language model programs for embodied control. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 9493–9500

[2] Xu M, Huang P, Yu W, et al (2023) Creative robot tool use with large language models. arXiv [cs.RO]

[3] Xie T, Zhao S, Wu CH, et al (2024) Text2Reward: Reward shaping with language models for reinforcement learning. In: The Twelfth International Conference on Learning Representations (ICLR 2024).

[4] Ma YJ, Liang W, Wang G, et al (2024) Eureka: Human-level reward design via coding large language models. In: The Twelfth International Conference on Learning Representations (ICLR 2024).

[5] Hutsebaut-Buysse M, Mets K, Latr´e S (2022) Hierarchical reinforcement learning: A survey and open research challenges. Machine Learning and Knowledge Extraction 4:172–221

[6] Takeda, S., Yamamori, S., Yagi, S. et al (2025). An empirical evaluation of a hierarchical reinforcement learning method towards modular robot control. Artif Life Robotics 12:154514–154525.

[7] Zhu Y, Stone P, Zhu Y (2022) Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. IEEE Robotics and Automation Letters 7:4126–4133

[8] Zhu J, Gienger M, Kober J (2022) Learning task-parameterized skills from few demonstrations. IEEE Robotics and Automation Letters 7:4063–4070

[9] Booth S, Knox WB, Shah J et al (2023) The perils of trial-and-error reward design: Misdesign through overfitting and invalid task specifications. In: Proceedings of the AAAI Conference on Artificial Intelligence, 37(5):5920–5929.

[10] Yu W, Gileadi N, Fu C et al (2023) Language to rewards for robotic skill synthesis. In: Proceedings of The 7th Conference on Robot Learning, Proceedings of Machine Learning Research, 229:374–404.

[11] Li H, Yang X, Wang Z et al (2023) Auto mc-reward: Automated dense reward design with large language models for minecraft. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 16426–16435.

[12] Zhang R, Lv J, Li J et al (2022) A graph-based reinforcement learning-enabled approach for adaptive human-robot collaborative assembly operations. Journal of Manufacturing Systems 63:491–503.

[13] Schoettler G, Nair A, Ojea JA et al (2020) Meta-reinforcement learning for robotic industrial insertion tasks. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 9728–9735.

[14] Gao J, Li Y, Chen Y et al (2024) An improved SAC-based deep reinforcement learning framework for collaborative pushing and grasping in underwater environments. IEEE Transactions on Instrumentation and Measurement 73:1–14.

[15] Xu H, Yu Q, Lai Y et al (2021) Efficient learning of goal-oriented push-grasping synergy in clutter. IEEE Robotics and Automation Letters 6:6337–6344.

[16] Shin G, Yun S, Kim WT et al (2024) A novel policy distillation with WPA-based knowledge filtering algorithm for efficient industrial robot control. IEEE Access 12:154514–154525.

[17] Tao S, Genc J, Chung T et al (2018) Repaint: Knowledge transfer in deep reinforcement learning. In: Meila M, Zhang T (eds) Proceedings of the 38th International Conference on Machine Learning (ICML'18). PMLR, pp 10141–10152.

[18] Barreto A, Borsa D, Quan J et al (2018) Transfer in deep reinforcement learning using successor features and generalised policy improvement. In: Proceedings of the International Conference on Machine Learning. PMLR, pp 501–510..

[19] Zhang A, Satija H, Pineau J (2018) Decoupling dynamics and reward for transfer learning. arXiv [cs.AI].

[20] Men Y, Jin L, Cui T et al (2023) Policy fusion transfer: The knowledge transfer for different robot peg-in-hole insertion assemblies. IEEE Transactions on Instrumentation and Measurement 72:1–10

[21] Jin L, Men Y, Song R et al (2024) Robot skill generalization: Feature-selected adaptation transfer for peg-in-hole assembly. IEEE Transactions on Industrial Electronics 71:2748–2757.

[22] Touvron H, Lavril T, Izacard G et al (2023) Llama: Open and efficient foundation language models. arXiv [cs.CL].

[23] Reimers N, Gurevych I (2019) Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv [cs.CL].

[24] Shi H, Li J, Mao J et al (2021) Lateral transfer learning for multiagent reinforcement learning. IEEE Transactions on Cybernetics 53:1699–1711.

[25] Mnih V, Badia AP, Mirza M et al (2016) Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning (ICML'16). JMLR.org, pp 1928–1937.

[26] Makoviychuk V, Wawrzyniak L, Guo Y et al (2021) Isaac gym: High performance GPU based physics simulation for robot learning. In: Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)

[27] Tessler C, Givony S, Zahavy T et al (2017) A deep hierarchical approach to lifelong learning in minecraft. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17). AAAI Press, pp 1553–1561.