HyperTASR: Hypernetwork-Driven Task-Aware Scene Representations for Robust Manipulation

Li Sun*, **Jiefeng Wu***, **Feng Chen**, **Ruizhe Liu**, **Yanchao Yang**

Institute of Data Science & Department of Electrical and Electronic Engineering
The University of Hong Kong
{sunlids, jiefengwu, cf24, zrllrz360}@connect.hku.hk, yanchaoy@hku.hk

Abstract: Effective policy learning for robotic manipulation requires scene representations that selectively capture task-relevant environmental features. Current approaches typically employ task-agnostic representation extraction, failing to emulate the dynamic perceptual adaptation observed in human cognition. We present HyperTASR, a hypernetwork-driven framework that modulates scene representations based on both task objectives and the execution phase. Our architecture dynamically generates representation transformation parameters conditioned on task specifications and progression state, enabling representations to evolve contextually throughout task execution. This approach maintains architectural compatibility with existing policy learning frameworks while fundamentally reconfiguring how visual features are processed. Unlike methods that simply concatenate or fuse task embeddings with task-agnostic representations, HyperTASR establishes computational separation between task-contextual and state-dependent processing paths, enhancing learning efficiency and representational quality. Comprehensive evaluations in both simulation and real-world environments demonstrate substantial performance improvements across different representation paradigms. Through ablation studies and attention visualization, we confirm that our approach selectively prioritizes task-relevant scene information, closely mirroring human adaptive perception during manipulation tasks. The project website is at lisunphil.github.io/HyperTASR_projectpage.

Keywords: Representation Learning, Robotic Manipulation, HyperNetworks

1 Introduction

Embodied AI has made significant advances in recent years [1, 2, 3, 4], driven by the mission of creating intelligent agents that can interact with physical environments with both effectiveness and robustness. These capabilities are essential for numerous real-world applications, necessitating the development of generalizable policy learning frameworks that translate perceptual observations into precise motor commands [5]. A *typical policy learning* pipeline comprises a representation extraction module that transforms raw observations into structured scene representations and a policy module that maps these representations to actions [6, 7, 8].

To enable flexible interactions across diverse scenarios, modern policy architectures incorporate task conditioning, enabling multi-task learning capabilities that facilitate the sharing of transferable skills – a critical step toward general-purpose embodied intelligence. When trained end-to-end on demonstration data, these pipelines have demonstrated impressive performance across various manipulation tasks and reasonable robustness in novel scenarios.

$*$: equal contribution.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

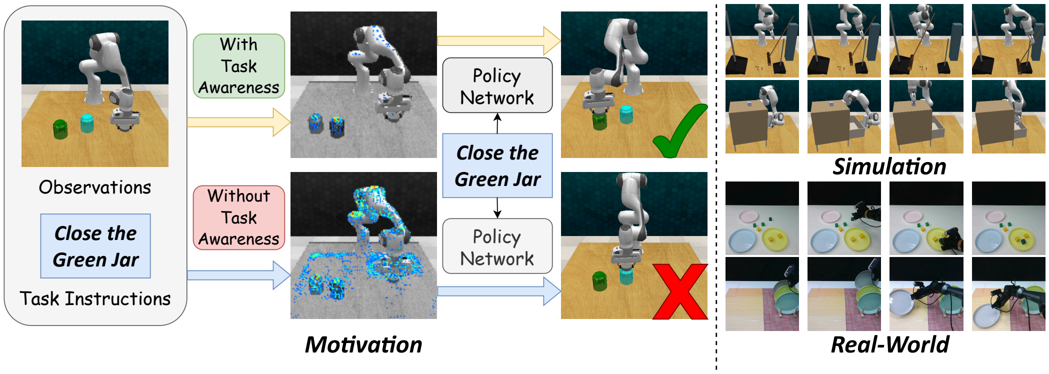


Figure 1: Task-aware representations enable selective attention to task-relevant scene elements, enhancing manipulation performance. *Top-left:* Our proposed HyperTASR pipeline incorporates task-aware scene representation extraction that dynamically modulates feature processing based on both task objectives and execution phase. *Bottom-left:* Conventional pipelines employ fixed, task-agnostic representation extractors that process visual information uniformly across all tasks, limiting representational flexibility. *Right:* Visualization of manipulation tasks in sim & real.

However, current policy learning approaches typically implement scene representation extraction as a *task-agnostic* process, which is decoupled from action prediction (Fig. 1). This separation omits established findings in human cognition, where extensive research demonstrates that visual processing adaptively reconfigures based on task objectives and execution context. Studies by Hayhoe [9, 10], Rothkopf [11], and Foulsham [12] reveal how visual representations dynamically adjust to task demands. This adaptation aligns with Gibson's [13] concept of affordances and the Theory of Event Coding (TEC) [14], as well as neurological evidence on adaptive neural representations [15].

Motivated by these insights, we propose *HyperTASR* – a hypernetwork-driven task-aware scene representation framework that enables policy learning pipelines to selectively focus on task-relevant scene elements, thereby enhancing sample efficiency and generalization capabilities. Consider the progressive nature of cup grasping: initially requiring coarse spatial awareness for localization, then transitioning to fine-grained geometric perception as the gripper approaches the handle. This dynamic modulation of representational focus facilitates precise and efficient scene interaction.

To *implement* this task-conditional scene representation, we introduce a modular transformation framework that adaptively reconfigures representations based on both task objectives and execution phase. *Specifically,* we employ a hypernetwork architecture that dynamically generates the parameters of a representation transformation network conditioned on task specifications and progression state. This approach establishes computational separation between task-contextual and state-dependent gradients [16], significantly enhancing learning efficiency. The framework continuously modulates scene representations throughout task execution, ensuring that the extracted features remain optimally aligned with current manipulation requirements.

We integrate HyperTASR with two representative state-of-the-art policy learning architectures: one employing train-from-scratch representations [17] and another utilizing fixed pre-trained backbones [18]. In **simulation** experiments on RLBench [19], our framework substantially elevates performance across both architectures. *Notably,* integration with GNFactor increases success rates by more than 27%, while implementation with 3D Diffuser Actor achieves success rates exceeding 80% for the first time in single-view configurations. In **real-world** experiments, HyperTASR enables effective multi-task manipulation even with limited demonstration data, *outperforming* baseline methods. Through comparative analysis with ablated models and attention visualization, we demonstrate that our approach selectively prioritizes task-relevant scene information throughout execution.

In summary, our contributions are:

- We propose HyperTASR, a novel framework for extracting task-aware scene representations that enables robotic agents to emulate human-like adaptive perception by focusing on the most task-relevant environmental features throughout execution.

- We introduce a hypernetwork-based representation transformation that dynamically generates adaptation parameters conditioned on both task specifications and progression state, maintaining architectural compatibility with existing policy learning frameworks.
- We demonstrate through comprehensive experiments in both simulation and real-world settings that HyperTASR significantly enhances performance across different representation paradigms, establishing new state-of-the-art results for single-view manipulation.

2 Related Work

Scene Representation for Multi-task Robotic Manipulation. Recent advancements in multi-task robotic manipulation have significantly improved task execution and generalization [20, 21, 22, 23, 24, 25, 26]. The dominant paradigm extracts scene representations from sensory input for action mapping [27, 28, 29]. State-of-the-art approaches [20, 30, 31] leverage foundation models to inject semantic knowledge, with methods like [18] directly utilizing pretrained visual backbones and GN-Factor [17] incorporating feature distillation – enhancing generalizability across environments [32]. However, a fundamental limitation persists: scene representations typically remain task-agnostic and static throughout execution. This contradicts findings in human cognition [9, 11], where visual processing dynamically reconfigures based on task objectives and context. While Vision-Language-Action models [33, 34, 35] incorporate task information through language embeddings, few integrate task context at the representation stage. RT-1 [21] employs FiLM [36] to encode instructions alongside observations but lacks explicit representation learning. Our approach fundamentally differs by conceptualizing representation adaptation as a dynamic transformation process that evolves throughout task execution – a significant advance beyond methods that maintain static representations or implement simple conditioning without accounting for temporal context.

Hypernetworks in Robotic Learning. Hypernetworks [37] provide an efficient framework for implementing our task-aware scene representations. Developed for neural architecture search [38, 39], continual learning [40, 41], generative modeling [42, 43], and reinforcement learning [44, 45, 46, 47], hypernetworks excel at generating specialized parameters conditioned on task-specific information [48]. In robotic applications [49, 50], they enable adaptation across diverse scenarios. We leverage hypernetworks for their advantages in enabling functional transformation of representation spaces rather than simple feature weighting – particularly well-suited for realizing our core contribution of adaptive perceptual processing that continuously evolves throughout task execution.

3 Method

In this section, we present HyperTASR, our novel hypernetwork-based architecture that dynamically modulates scene representation extraction based on task context and execution phase. In Sec. 3.1, we formalize the problem definition within the context of manipulation policy learning, followed by a detailed exposition of our task-conditional representation framework (Sec. 3.2) and the associated training methodologies (Sec. 3.3 & Sec. 3.4). Please refer to Fig. 2 for an overview of the proposed policy learning pipeline and the core components.

3.1 Preliminaries

Multi-task robotic manipulation requires agents to act efficiently across a heterogeneous task space, denoted by $\mathcal{T} = \{\tau_k\}_{k=1}^{K}$. Each task τ induces a task-conditioned Markov Decision Process (MDP), characterized by states $s_t \in \mathcal{S}$, actions $a_t \in \mathcal{A}$, and the task specification $\tau \in \mathcal{T}$. The primary objective in multi-task policy learning is to train a policy $\pi : \mathcal{S} \times \mathcal{T} \to \mathcal{A}$ that generates optimal action sequences for completing the task. For embodied agents operating in physical environments, the latent state s_t is rarely directly accessible due to inherent limitations in environmental perception. *Consequently,* policies usually operate on learned representations $z_t = \phi(o_t)$ derived from partial observations o_t, where ϕ denotes the representation extraction network (**representation extractor**) that feeds into the subsequent action prediction module (**policy**). These processes are also known

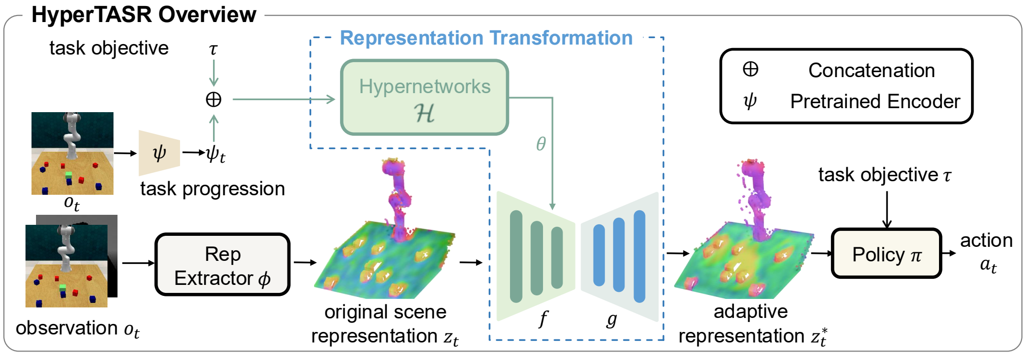


Figure 2: HyperTASR framework overview. Our approach enhances policy learning pipelines by introducing a dynamic scene representation mapping to transform the representation before performing action prediction. The mapping consists of a hypernetwork and task-specific autoencoder (highlighted in blue). The hypernetwork dynamically generates encoder parameters conditioned on both task objectives and progression state, enabling contextual modulation of scene representations throughout task execution.

as POMDPs (Partially Observable Markov Decision Processes). Here, we adopt the term policy to specifically reference the action prediction component of the learning pipeline. In the single-view paradigm, agents perceive the environment through a single RGB-D image at each timestep t, where $o_t = (I_t, D_t)$ comprises RGB imagery (I_t) and corresponding depth data (D_t).

3.2 Task-Aware Scene Representation

To formalize, contemporary frameworks for multi-task robotic learning typically implement a two-stage architecture: first extracting a scene representation $z_t = \phi(o_t)$ from observation o_t, then computing actions via a policy module $a_t = \pi(z_t, \tau)$. The task information is integrated only at the action prediction stage, and the representation extraction process ϕ remains task-agnostic and static throughout task execution.

Specifically, we can categorize current representation extraction methodologies into two main approaches. The *first* approach leverages pre-trained foundational models to extract semantically rich scene representations [18, 30, 51]. While these methods capture general visual semantics effectively, they operate independently of the downstream action prediction objective, potentially extracting features suboptimal for specific manipulation tasks. The *second* approach entails training representation extraction architectures from scratch, typically co-optimized with the policy network [17, 52, 53], yielding representations calibrated for action prediction. However, these approaches typically employ a task-agnostic representation extractor, failing to recognize that different manipulation objectives may require selectively emphasizing distinct aspects of the visual scene.

Cognitive Inspiration. In contrast, human visual processing adaptively modulates scene perception based on both task objectives and the execution phase. Consider the procedural task of preparing tea: during the initial object localization phase, coarse spatial awareness suffices; however, during the pouring action, visual processing sharpens to capture precise spatial relationships and fine-grained geometric details necessary for successful liquid transfer. This dynamic perceptual reconfiguration suggests that effective scene representations should evolve contextually throughout task execution.

Proposed Approach. *Consequently,* we propose that scene representation extraction should be explicitly conditioned on task context, modeled as $z_t = \phi(o_t, \tau)$. This task-conditional representation framework offers several advantages: (1) enhanced effectiveness across tasks through adaptive environmental encoding, (2) improved interpretability as representations selectively highlight task-relevant scene elements, (3) increased computational efficiency by filtering irrelevant environmental information, and (4) closer alignment with the progressive nature of manipulation tasks. The following sections detail our hypernetwork-based implementation that dynamically modulates scene representation extraction according to both task identity and execution phase.

3.3 Hypernetwork-Driven Task-Conditional Scene Representation

Our approach for integrating task and progression awareness into scene representations is designed as a versatile framework applicable across diverse policy learning architectures. Rather than modifying the intrinsic representation extraction mechanisms of each pipeline – which would introduce architectural dependencies and complicate comparative analysis – we propose a modular transformation layer that preserves the original dimensionality of scene representations while enriching them with task-specific context. *Specifically,* we implement this transformation as a lightweight autoencoding structure:

$$z_t^* = g^{\omega} \circ f(z_t; \theta), \tag{1}$$

where f and g denote the encoding and decoding functions, respectively, parameterized by θ and ω. This formulation ensures that the transformed representation z_t^* maintains the same dimensionality as the original z_t, enabling seamless integration with any downstream policy network without architectural modifications.

To incorporate contextual information about the task into the representation transformation process, we dynamically modulate the encoding function parameters $f(\cdot; \theta)$ rather than the features. *Crucially,* we identify two fundamental dimensions that guide adaptive representation: the *task objective* that defines the manipulation goal and the *task progression state* that captures temporal execution context. We formalize these dimensions with task specification τ and progression encoding $\psi_t = \psi(o_t)$ extracted from observation o_t. The conditional transformation process is expressed as:

$$z_t^* = g^{\omega} \circ f(z_t; \theta(\tau, \psi_t)). \tag{2}$$

While the decoding function g^{ω} remains task-invariant and is co-optimized during training, the scene representation encoder parameters θ are dynamically generated to adapt to the task context. *For this dynamic parameter generation,* we leverage a hypernetwork architecture $\mathcal{H}$ [50] that synthesizes task-specific encoding parameters conditioned on both task objectives and execution state:

$$\theta = \mathcal{H}(\tau, \psi_t). \tag{3}$$

This hypernetwork-based parameterization provides three crucial technical advantages: (1) it establishes a clear computational separation between task-contextual and state-dependent gradient flows during backpropagation [16], substantially enhancing learning efficiency; (2) it enables functional transformation of the representation space rather than mere feature weighting or selection, allowing for more expressive adaptation to task requirements; and (3) it facilitates rapid adaptation across tasks without catastrophic forgetting, as task-specific parameters are generated on demand. The resulting architecture dynamically reconfigures its representation extraction strategy for each task and execution phase while maintaining nice compatibility with existing policy learning frameworks.

3.4 Integration and Training Objectives

We integrate our hypernetwork-based task-aware scene representation extraction with two representative state-of-the-art architectures that exemplify distinct approaches to representation learning: GNFactor [17], which trains representations from scratch with the policy, and 3D Diffuser Actor [18], which leverages pre-trained visual backbones.

GNFactor Integration. We insert a 3D autoencoder after GNFactor's volumetric representation extraction, with encoder parameters generated by the proposed HyperTASR. *It is worth noting that* we eliminate the feature distillation component consists of neural rendering. *This modification streamlines the framework* to be optimized end-to-end solely through behavior cloning, formulated as:

$$\mathcal{L} = \lambda_{\text{pos}}\mathcal{L}_{\text{pos}} + \lambda_{\text{rot}}\mathcal{L}_{\text{rot}} + \lambda_{\text{open}}\mathcal{L}_{\text{open}} + \lambda_{\text{collide}}\mathcal{L}_{\text{collide}}, \tag{4}$$

where $\mathcal{L}_{\text{pos}}$, $\mathcal{L}_{\text{rot}}$, $\mathcal{L}_{\text{open}}$, and $\mathcal{L}_{\text{collide}}$ represent the position loss, rotation loss, gripper openness loss, and collision avoidance loss, respectively.

	Avg. Success ↑	Avg. Rank ↓	close jar	open drawer	sweep to dustpan	meat off grill	turn tap	slide block	put in drawer	drag stick	push buttons	stack blocks
Peract [20]	20.4	5.4	18.7$_{\pm 8.2}$	54.7$_{\pm 18.6}$	0.0$_{\pm 0.0}$	40.0$_{\pm 17.0}$	38.7$_{\pm 6.8}$	18.7$_{\pm 13.6}$	2.7$_{\pm 3.3}$	5.3$_{\pm 5.0}$	18.7$_{\pm 12.4}$	6.7$_{\pm 1.9}$
GNFactor [17]	33.3	4.8	32.8$_{\pm 0.6}$	36.0$_{\pm 0.2}$	48.0$_{\pm 0.3}$	51.2$_{\pm 0.1}$	56.8$_{\pm 0.3}$	20.0$_{\pm 1.2}$	8.8$_{\pm 1.3}$	69.6$_{\pm 0.4}$	5.6$_{\pm 3.7}$	4.0$_{\pm 4.0}$
Act3D [30]	65.3	3.1	52.0$_{\pm 5.7}$	84.0$_{\pm 8.6}$	80.0$_{\pm 9.8}$	66.7$_{\pm 1.9}$	64.0$_{\pm 5.7}$	**100.0**$_{\pm 0.0}$	54.7$_{\pm 3.8}$	86.7$_{\pm 1.9}$	64.0$_{\pm 1.9}$	0.0$_{\pm 0.0}$
3D Diffuser Actor [18]	79.0	1.8	63.2$_{\pm 1.6}$	**88.8**$_{\pm 7.8}$	94.4$_{\pm 4.1}$	**84.8**$_{\pm 4.7}$	72.8$_{\pm 4.7}$	94.4$_{\pm 2.0}$	88.8$_{\pm 4.7}$	98.4$_{\pm 2.0}$	87.2$_{\pm 1.6}$	**17.4**$_{\pm 5.1}$
GNFactor w/ HyperTASR	42.6	4.5	32.0$_{\pm 0}$	75.2$_{\pm 0.5}$	66.4$_{\pm 0.4}$	48.8$_{\pm 0.2}$	54.4$_{\pm 2.0}$	23.2$_{\pm 4.6}$	22.4$_{\pm 0.9}$	83.2$_{\pm 0.3}$	17.6$_{\pm 1.2}$	3.2$_{\pm 3.5}$
3D DA w/ HyperTASR	**81.3**	**1.4**	**68.0**$_{\pm 2.5}$	87.2$_{\pm 1.6}$	**98.4**$_{\pm 2.0}$	82.4$_{\pm 3.2}$	**85.6**$_{\pm 3.2}$	98.4$_{\pm 2.0}$	**89.6**$_{\pm 6.5}$	**100.0**$_{\pm 0.0}$	**92.0**$_{\pm 0.0}$	11.2$_{\pm 1.6}$

Table 1: **Evaluation on RLBench in the single-view setting.** Success rates on 10 RLBench tasks using only the *front* camera view. All models are trained with 20 demonstrations per task and evaluated across 5 seeds with 25 episodes per task. HyperTASR significantly improves performance when integrated with both GNFactor [17] and 3D Diffuser Actor (3D DA) [18], demonstrating the effectiveness of task-aware scene representations.

3D Diffuser Actor Integration. For 3D Diffuser Actor, which utilizes point cloud features derived from a pre-trained 2D visual backbone, we integrate HyperTASR by inserting a 2D autoencoder after the pre-trained feature extraction stage. This placement allows our hypernetwork to modulate the rich semantic features before they are projected into the 3D point cloud representation. We maintain the original training objectives, optimizing the complete architecture through behavior cloning loss.

Our hypernetwork, autoencoder components, and policy networks are jointly optimized through gradient backpropagation. This integration approach demonstrates the versatility of our framework, as it seamlessly enhances both learned-from-scratch and pre-trained representation architectures without requiring a fundamental redesign of their core components.

4 Experiments

We evaluate HyperTASR by integrating it with GNFactor [17] and 3D Diffuser Actor [18] on multi-task manipulation benchmarks in both simulation and real-world settings.

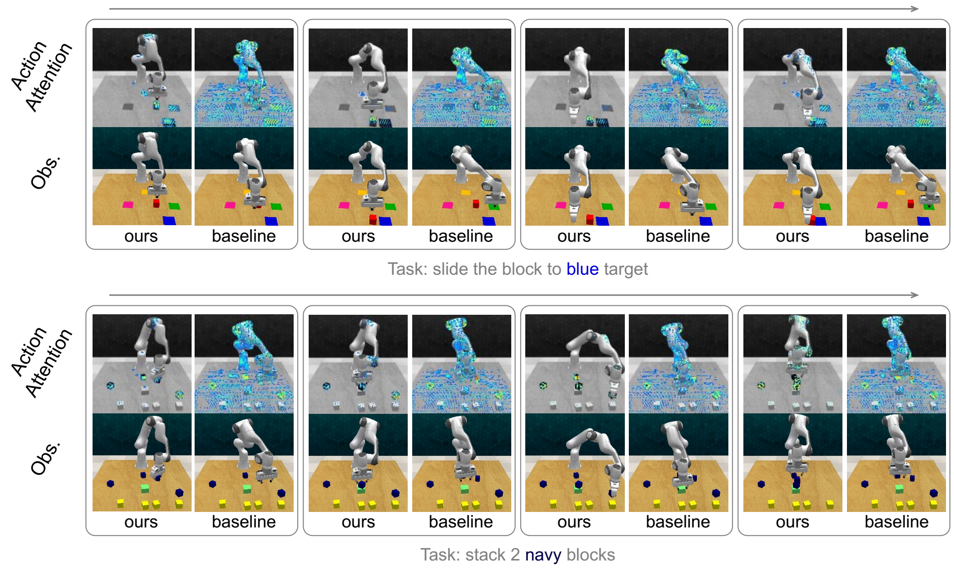
4.1 Experiment Setting

Our experiments use RLBench [19], a large-scale benchmark with over 100 manipulation tasks in realistic simulated environments using a Franka Panda robot. Following [17], we evaluate on 10 language-conditioned tasks comprising 166 variations. All methods predict the next keypose for the end-effector and use BiRRT [54] for motion planning. To emphasize the impact of scene representations under practical deployment constraints, we also conduct all experiments in the challenging single-view setting, using only the front camera view RGB-D sensory data.

Baselines. We compare against state-of-the-art policy learning frameworks: PerAct [20], which voxelizes the 3D workspace; GNFactor [17], which constructs a 3D feature volume from a single RGB-D view; and Act3D [30] and 3D Diffuser Actor [18], which represent the state of the art on RLBench. Results for PerAct and Act3D are adopted from published work [17, 18], while we retrained GNFactor and 3D Diffuser Actor under identical single-view conditions to ensure fair comparison. All models are trained on the same keypose demonstrations, and we report results across five random seeds to ensure statistical reliability.

4.2 Implementation Details

Our HyperTASR implementation uses a UNet-based [55] autoencoder with skip connections. Following [50], we employ an optimization-biased hypernetwork that predicts parameter updates iteratively rather than directly generating encoder weights via fully connected layers. For task objective conditioning (τ), we utilize the language features already present in the original policy pipelines. Task progression information (ψ_t) is extracted using a frozen pretrained VAE Encoder from Stable Diffusion [56]. For GNFactor integration, we directly apply our HyperTASR to predict parameters of the original lightweight 3D UNet voxel encoder. The model is trained for 200k iterations on a single NVIDIA H800 GPU. For 3D Diffuser Actor, we maintain the fixed backbone and add a 2D UNet with nine convolutional layers, training for 600k iterations on four H800 GPUs.



Task: slide the block to blue target

Task: stack 2 navy blocks

Figure 3: Attention visualization comparing policies with and without task-aware scene representations on ‘slide block’ (top) and ‘stack blocks’ (bottom) tasks. HyperTASR consistently focuses on task-relevant objects, while the baseline attention disperses across irrelevant scene elements, explaining the performance gain of HyperTASR across approaches.

4.3 Results in Simulation

We present quantitative results in Table 1, evaluating performance through both average success rate and average rank across tasks. The experimental findings demonstrate HyperTASR’s substantial impact on both integration frameworks. When integrated with GNFactor, HyperTASR achieves a remarkable 27.9% relative improvement (9.3% absolute increase) over the baseline across the 10 evaluation tasks. Notably, this performance gain comes with reduced computational requirements – our approach eliminates the need for feature distillation, removes multi-view supervision dependencies, and results in a smaller network with faster training convergence. This combination of enhanced performance with reduced computational overhead highlights the efficiency of task-aware scene representations. For the 3DDA integration, HyperTASR surpasses not only the baseline but all current state-of-the-art methods, establishing new benchmark performance for single-view manipulation. The consistent improvement across both frameworks demonstrates HyperTASR’s versatility and effectiveness with different scene representation paradigms and policy network architectures. The average rank metric further confirms that models enhanced with HyperTASR perform better across a wider range of tasks, indicating superior cross-task generalization capabilities – a critical attribute for real-world robotic applications in diverse manipulation scenarios.

Visualization of HyperTASR. To provide qualitative insights into how task-aware scene representations transform attentional dynamics, we visualize action attention patterns in Fig. 3. Compared to baseline models, HyperTASR produces significantly more focused attention maps that precisely target task-relevant objects. Furthermore, these attention patterns evolve dynamically throughout task execution. In the sliding blocks task, for example, attention initially concentrates on the block while the gripper approaches. Once contact is established, attention shifts to the target area where the block should be placed. This progressive adaptation of perceptual focus closely mirrors human visual processing during manipulation tasks, providing a clear mechanism for the performance improvements observed in our quantitative results.

4.4 Real-World Evaluation

To validate HyperTASR’s effectiveness in physical environments, we conduct experiments using a Piper robotic arm equipped with a parallel gripper. We design six diverse manipulation tasks with variations in object colors, counts, placements, and categories to assess generalization capabilities.

For each task, we collect 15 expert demonstrations using a master-puppet teleoperation system identical to ALOHA [57]. RGB-D observations are captured via an Intel RealSense camera at 640×480 resolution and subsequently downsampled to 256×256 for processing. During inference, target gripper poses are executed using the MoveIt package in ROS. We integrate HyperTASR with the 3D Diffuser Actor framework and evaluate performance across 15 episodes per task.

As shown in Table 2, HyperTASR consistently outperforms the baseline 3D Diffuser Actor across all real-world tasks, demonstrating that the benefits of task-aware scene representations transfer effectively from simulation to physical environments. This performance gain is particularly noteworthy given the limited demonstration data (15 per task) and the inherent challenges of real-world sensing and actuation. Additional visualizations and experimental details are provided in the Appendix.

	Avg. Succ	place dish	clean cups	stack cups	stack blocks	put cups on shelf	place blocks
3D Diffuser Actor	42.2	40.0	53.3	13.3	20.0	46.6	80.0
3D-DA w/ HyperTASR	**51.1**	**53.3**	**66.6**	**20.0**	**26.6**	**53.3**	**86.6**

Table 2: **Real-World Experiment Results.** Success rates across six manipulation tasks with 15 episodes per task. HyperTASR consistently outperforms the baseline 3D Diffuser Actor, demonstrating that task-aware scene representations transfer effectively from simulation to physical environments with limited demonstration data.

4.5 Ablation Study

We conduct comprehensive ablation studies to evaluate key design choices in HyperTASR. *First,* we examine whether simpler conditioning mechanisms could achieve similar benefits. Within the 3D Diffuser Actor framework, we replace our hypernetwork with a cross-attention module that fuses task objectives and progression information with the original scene representation. As shown in Table 3 (upper), this alternative yields only marginal improvements, confirming that effective task-aware representations require sophisticated functional transformation rather than simple feature fusion. We *further* investigate three critical aspects using GNFactor, with results in Table 3 (lower): (i) **Feature Distillation:** Adding explicit distillation supervision constrains representational flexibility, reducing performance by limiting adaptation capabilities – supporting our design choice to eliminate this component. (ii) **Task Progression:** Conditioning only on task objectives (τ) without progression information significantly degrades performance, confirming that effective representations must evolve throughout task execution. (iii) **Hypernetwork Target:** Having the hypernetwork predict only encoder parameters proves more efficient than generating the entire autoencoder, validating our architectural focus on encoder transformation with a fixed decoder.

Ablation	Success Rate (%)
3D Diffuser Actor	79.02 ± 1.65
Task-Awareness by Transformer	79.23 ± 1.10
Task-awareness by HyperTASR (ours)	**81.28 ± 0.82**
GNFactor	33.20 ± 1.22
HyperTASR w/ Feature Distillation	34.00 ± 2.12
HyperTASR conditioned on τ	32.24 ± 0.60
HyperTASR predicting θ and ω	36.32 ± 1.32
HyperTASR (ours)	**42.60 ± 1.35**

Table 3: **Ablation Study Results.** *Upper:* Comparison of hypernetwork vs. attention-based approaches for implementing task awareness in the 3D Diffuser Actor framework. *Lower:* Analysis of feature distillation, task progression conditioning, and hypernetwork target selection within the GNFactor framework.

5 Conclusion

We present HyperTASR, a novel framework for task-aware scene representations in robotic manipulation that dynamically adapts perceptual processing based on both task objectives and execution progression. Our hypernetwork-driven approach enables representations to evolve contextually throughout task execution, focusing on task-relevant environmental features. Evaluations in both simulation and real-world settings demonstrate significant performance enhancements across different representation paradigms. Ablation studies confirm the effectiveness of our design components for task-aware representations extraction. HyperTASR bridges the gap between human-inspired adaptive perception and computational approaches to robotic manipulation, establishing a foundation for more efficient multi-task policy learning.

6 Limitations

While HyperTASR demonstrates substantial improvements in manipulation performance, several opportunities for future enhancement remain. Our experiments primarily focus on behavior cloning, while HyperTASR has the capability of extending to reinforcement learning field.

Our current evaluation uses single-arm grippers as the robotic platform. The principles of task-aware scene representation could potentially extend to more advanced manipulation systems such as bimanual setups and dexterous hands, which would broaden the applicability of our approach to more sophisticated manipulation tasks.

These limitations highlight promising research directions that could build upon the foundation established by HyperTASR. The consistent performance improvements observed across different representation paradigms suggest that task-aware adaptation principles could generalize effectively to these extended capabilities.

Acknowledgments

This work is supported by the Early Career Scheme of the Research Grants Council (RGC) grant # 27207224, the HKU-100 Award, a donation from the Musketeers Foundation, and in part by the JC STEM Lab of Autonomous Intelligent Systems funded by The Hong Kong Jockey Club Charities Trust.

References

[1] R. Pfeifer and C. Scheier. Representation in natural and artificial agents: an embodied cognitive science perspective. *Zeitschrift für Naturforschung C*, 53(7-8):480–503, 1998. 1

[2] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2): 230–244, 2022. 1

[3] Y. Liu, W. Chen, Y. Bai, X. Liang, G. Li, W. Gao, and L. Lin. Aligning cyber space with physical world: A comprehensive survey on embodied ai. *arXiv preprint arXiv:2407.06886*, 2024. 1

[4] Z. Xu, K. Wu, J. Wen, J. Li, N. Liu, Z. Che, and J. Tang. A survey on robotics with foundation models: toward embodied ai. *arXiv preprint arXiv:2402.02385*, 2024. 1

[5] Y. Mu, Q. Zhang, M. Hu, W. Wang, M. Ding, J. Jin, B. Wang, J. Dai, Y. Qiao, and P. Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[6] Z. Xu, Z. He, J. Wu, and S. Song. Learning 3d dynamic scene representations for robot manipulation. *arXiv preprint arXiv:2011.01968*, 2020. 1

[7] A. Billard and D. Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446): eaat8414, 2019. 1

[8] Y. Li, S. Li, V. Sitzmann, P. Agrawal, and A. Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*, pages 112–123. PMLR, 2022. 1

[9] M. M. Hayhoe, A. Shrivastava, R. Mruczek, and J. B. Pelz. Visual memory and motor planning in a natural task. *Journal of vision*, 3(1):6–6, 2003. 2, 3

[10] M. M. Hayhoe. Vision and action. *Annual review of vision science*, 3(1):389–413, 2017. 2

[11] C. A. Rothkopf, D. H. Ballard, and M. M. Hayhoe. Task and context determine where you look. *Journal of vision*, 7(14):16–16, 2007. 2, 3

[12] T. Foulsham, E. Walker, and A. Kingstone. The where, what and when of gaze allocation in the lab and the natural environment. *Vision research*, 51(17):1920–1931, 2011. 2

[13] J. J. Gibson. *The ecological approach to visual perception: classic edition*. Psychology press, 2014. 2

[14] B. Hommel, J. Müsseler, G. Aschersleben, and W. Prinz. The theory of event coding (tec): A framework for perception and action planning. *Behavioral and brain sciences*, 24(5):849–878, 2001. 2

[15] L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater, and J. Santos-Victor. Affordances in psychology, neuroscience, and robotics: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 10(1):4–25, 2016. 2

[16] E. Sarafian, S. Keynan, and S. Kraus. Recomposing the reinforcement learning building blocks with hypernetworks. In *International Conference on Machine Learning*, pages 9301–9312. PMLR, 2021. 2, 5

[17] Y. Ze, G. Yan, Y.-H. Wu, A. Macaluso, Y. Ge, J. Ye, N. Hansen, L. E. Li, and X. Wang. Multi-task real robot learning with generalizable neural feature fields. *CoRL*, 2023. 2, 3, 4, 5, 6, 13, 14, 15, 16, 20, 21

[18] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024. 2, 3, 4, 5, 6, 13, 14, 15, 17, 21

[19] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020. 2, 6, 14

[20] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023. 3, 6

[21] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 3

[22] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022. 3

[23] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pages 894–906. PMLR, 2022. 3

[24] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023. 3

[25] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox. Rvt-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024. 3

[26] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13739–13748, 2022. 3

[27] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of machine learning research*, 22(30): 1–82, 2021. 3

[28] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022. 3

[29] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9601–9610, 2020. 3

[30] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023. 3, 4, 6

[31] G. Lu, S. Zhang, Z. Wang, C. Liu, J. Lu, and Y. Tang. Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation. In *European Conference on Computer Vision*, pages 349–366. Springer, 2025. 3

[32] S. Karamcheti, S. Nair, A. S. Chen, T. Kollar, C. Finn, D. Sadigh, and P. Liang. Language-driven representation learning for robotics. *arXiv preprint arXiv:2302.12766*, 2023. 3

[33] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024. 3

[34] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 3

[35] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024. 3

[36] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 3

[37] D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkpACe1lx. 3

[38] A. Brock, T. Lim, J. M. Ritchie, and N. J. Weston. Smash: One-shot model architecture search through hypernetworks. In *6th International Conference on Learning Representations 2018*, 2018. 3

[39] C. Zhang, M. Ren, and R. Urtasun. Graph hypernetworks for neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rkgW0oA9FX. 3

[40] J. von Oswald, C. Henning, B. F. Grewe, and J. Sacramento. Continual learning with hypernetworks. In *8th International Conference on Learning Representations (ICLR 2020)(virtual)*. International Conference on Learning Representations, 2020. 3

[41] C. Henning, M. Cervera, F. D'Angelo, J. Von Oswald, R. Traber, B. Ehret, S. Kobayashi, B. F. Grewe, and J. Sacramento. Posterior meta-replay for continual learning. *Advances in neural information processing systems*, 34:14135–14149, 2021. 3

[42] N. Ratzlaff and L. Fuxin. Hypergan: A generative model for diverse, performant neural networks. In *International Conference on Machine Learning*, pages 5361–5369. PMLR, 2019. 3

[43] I. Skorokhodov, S. Ignatyev, and M. Elhoseiny. Adversarial generation of continuous images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10753–10764, 2021. 3

[44] Y. Huang, K. Xie, H. Bharadhwaj, and F. Shkurti. Continual model-based reinforcement learning with hypernetworks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 799–805. IEEE, 2021. 3

[45] Z. Xian, S. Lal, H.-Y. Tung, E. A. Platanios, and K. Fragkiadaki. Hyperdynamics: Meta-learning object and agent dynamics with hypernetworks. *arXiv preprint arXiv:2103.09439*, 2021. 3

[46] S. Rezaei-Shoshtari, C. Morissette, F. R. Hogan, G. Dudek, and D. Meger. Hypernetworks for zero-shot transfer in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9579–9587, 2023. 3

[47] M. Beukman, D. Jarvis, R. Klein, S. James, and B. Rosman. Dynamics generalisation in reinforcement learning via adaptive context-aware policies. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[48] V. K. Chauhan, J. Zhou, P. Lu, S. Molaei, and D. A. Clifton. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 57(9):250, 2024. 3

[49] J. Beck, M. T. Jackson, R. Vuorio, and S. Whiteson. Hypernetworks in meta-reinforcement learning. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1478–1487. PMLR, 14–18 Dec 2023. 3

[50] H. Ren, L. Sun, X. Wang, P. Zhou, Z. Wu, S. Dong, D. Zou, Y. Zheng, and Y. Yang. Hypogen: Optimization-biased hypernetworks for generalizable policy generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=CJWMXqAnAy`. 3, 5, 6, 13

[51] Y. Wang, Z. Li, M. Zhang, K. Driggs-Campbell, J. Wu, L. Fei-Fei, and Y. Li. D^3 fields: Dynamic 3d descriptor fields for zero-shot generalizable robotic manipulation. *arXiv preprint arXiv:2309.16118*, 2023. 4

[52] T. Ma, J. Zhou, Z. Wang, R. Qiu, and J. Liang. Contrastive imitation learning for language-guided multi-task robotic manipulation. *arXiv preprint arXiv:2406.09738*, 2024. 4

[53] D. Driess, I. Schubert, P. Florence, Y. Li, and M. Toussaint. Reinforcement learning with neural radiance fields. *Advances in Neural Information Processing Systems*, 35:16931–16945, 2022. 4

[54] J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000. 6

[55] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 6

[56] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 6

[57] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024. 8

A Additional Implementation Details

A.1 Model Structure

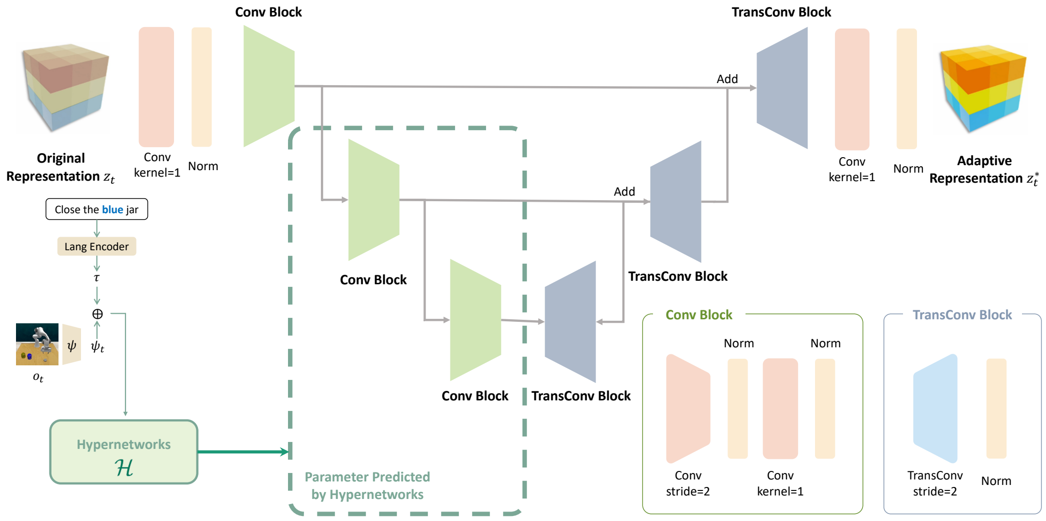


Figure 4: **The detailed model structure of our HyperTASR.** The main diagram shows that the pipeline consists of 2 Convolutional layer and a UNet with skip-connection. *Bottom-right:* The detailed structure of **Conv Block** and **TransConv Block** in the pipeline. These blocks serve as the key components for the encoding and decoding process.

HyperTASR we design mainly consists of three convolutional blocks and three transposed convolutional blocks, as detailed in Fig. 4. Each convolutional block contains two convolutional layers, followed by an InstanceNorm layer and a Leaky ReLU activation function. The first convolutional layer in each block has a kernel size of 3 and a stride of 2, which reduces the resolution of the feature map while increasing the feature channel dimension, effectively encoding the features. The second convolutional layer has a kernel size of 1 and does not change the resolution or channel dimension of the feature map, serving to refine the encoded features.

The transposed convolutional blocks are relatively simpler, consisting of a single transposed convolutional layer followed by InstanceNorm and Leaky ReLU activation. The transposed convolutional layer increases the resolution of the feature map while reducing the channel dimension, effectively decoding the features. This layer has a kernel size of 3 and a stride of 2, ensuring that the spatial dimensions of the feature map are expanded appropriately.

HyperTASR used in GNFactor [17] and 3D Diffuser Actor [18] follow the aforementioned structure. GNFactor directly utilizes a 3D deep volume as its representation, requiring 3D convolutions and 3D transposed convolutions. For the 3D Diffuser Actor, the representation is a point cloud feature, which in a single-view setup combines a 2D feature map and a depth map, leading us to employ 2D convolutions and 2D transposed convolutions as the core elements of the UNet architecture.

In Fig. 5, we detail the implementation of our hypernetworks. Following [50], we adopt an optimization-based hypernetwork, which iteratively predicts the parameter updates rather than directly predicting the final parameter. In our implementation, $K = 8$ represents the parameter update iteration. To control the parameter size of the UNet and effectively manage the parameter size of the Hypernetworks, we introduce optional encoders and decoders. The encoder reduces the dimensionality of the input features before they are fed into the UNet, while the decoder restores the dimensionality after processing. This mechanism is particularly useful for maintaining a balance between model complexity and performance. Specifically, for the 3D Diffuser Actor, we incorporate these optional encoders and decoders to better adapt to the varying input feature requirements.

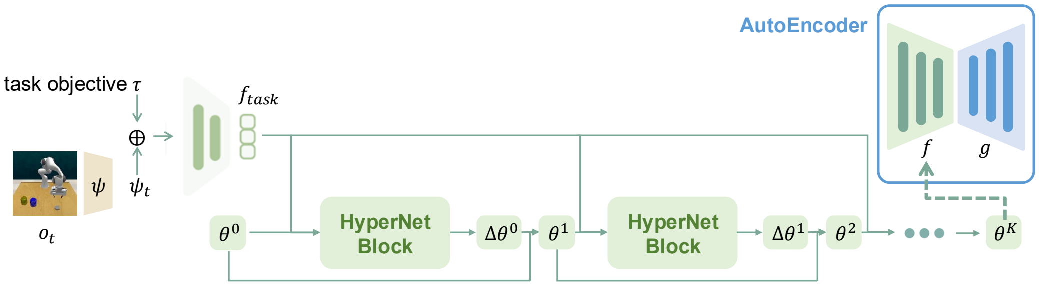


Figure 5: **The detailed structure of hypernetworks we used in HyperTASR.** We employ an optimization-biased hypernetwork that predicts parameter updates iteratively rather than directly generating encoder weights via fully connected layers.

A.2 Dataset Composition

Task	Variation Type	# of Variations	Avg. Keyframes	Language Description Example
`close jar`	color	20	6.0	“close the — jar”
`meat off grill`	category	2	5.0	“take the — off the grill”
`open drawer`	placement	3	3.0	“open the — drawer
`sweep to dustpan`	size	2	4.6	“sweep dirt to the — dustpan”
`turn tap`	placement	2	2.0	“turn — tap”
`slide block`	color	4	4.7	“slide the block to — target”
`put in drawer`	placement	3	12.0	“put the item in the — drawer”
`drag stick`	color	20	6.0	“use the stick to drag the cube onto the — — target”
`push buttons`	color	50	3.8	“push the — button, [then the — button]”
`stack blocks`	color, count	60	14.6	“stack — — blocks”

Table 4: **Dataset composition of 10 manipulation tasks in RLBench [19].**

Task	Variation Type	Language Description Example
`place dish`	color	“place the — dish on the — tablecloth”
`clean cups`	color, placement	“Put the — cup into the — basket”
`stack cups`	color, placement	“stack the — cup on the — cup
`stack blocks`	color, count	“stack the — cubes”
`put cups on shelf`	placement	“put the — cup on the shelf next to — cup”
`place blocks`	color	“Place the — block on the — plate”

Table 5: **Dataset composition of 6 manipulation tasks in real robot experiments.**

We conduct experiments on 10 language-conditioned manipulation tasks from RLBench [19], which align with the experimental setup of GNFactor [17]. The task variations include randomly sampled attributes such as colors, sizes, counts, placements, and object categories. Detailed descriptions of the variation types, variation numbers, average keyframes, and sample language descriptions for these tasks are provided in Tab. 4.

For real robot experiments, we design 6 tasks that cover diverse tasks for ”pick and place”. We give our sample task description in Tab. 5.

A.3 Hyperparamters

We provide detailed hyperparameters for our experiments in Tab. 6 and Tab. 7, with some parallel settings and input data differences compared to GNFactor [17] and 3D Diffuser Actor [18]. To ensure a fair comparison, we reproduce the experiments using the same hyperparameters as the original codebase and report the corresponding results in Tab. 1. These results serve as a benchmark for understanding the impact of our modifications.

Impact of Hyperparameter Changes on Experimental Results. For the GNFactor framework, we opt not to use distributed data parallel (DDP) training. Instead, we utilize a single GPU, halve

Variable Name	Value
training iteration	200k
image size	$128 \times 128 \times 3$
batch size	1
optimizer	LAMB
learning rate	0.0005
input voxel size	$100 \times 100 \times 100$
number of transformer blocks	6
number of latents in PerceiverI/O	2048
dimension of CLIP language features	512

Table 6: **Hyperparameters** in GNFactor [17] Framework.

Variable Name	Value
training iteration	$800k$
image size	$256 \times 256 \times 3$
batch size	240
optimizer	Adam
learning rate	0.0001
embedding dim	120
diffusion timestep	100
loss weight of position and rotaion	30 : 20
maximal # of keyposes	25

Table 7: **Hyperparameters** in 3D Diffuser Actor [18] Framework.

the batch size, and double the number of training steps. Despite this adjustment, the final reproduced results fall within the range of multiple experimental outcomes reported in [17]. For the 3D Diffuser Actor, we train both our modified pipeline and the original codebase with the training data provided in the author's released repository, using an RGB image resolution of 256×256. Due to the lower resolution compared to the original paper (256×256) [18], our reproduced results (77.0%) are slightly below the original results (78.4%). Additionally, slight adjustments to the loss weights are made to account for the resolution difference, and the best configuration is chosen as the unified hyperparameter setting for all our experiments.

A.4 Computation Cost

We calculate the computation cost of our experiments on GNFactor [17] by measuring both the total number of parameters in the network and the training time, as shown in Tab. 8. For training time, we use an unloaded GPU to train for 1k steps and record the time taken. From the results, we observe that while our network has more parameters, it achieves higher training efficiency. The increased parameter count results from the inclusion of Hypernetworks, but this does not negatively impact training efficiency. On the contrary, by removing the neural renderer used for feature distillation, the overall training time is reduced. This demonstrates that HyperTASR does not impose a significant computational burden on the network and, in some cases, even improves efficiency by eliminating certain supervisory components.

	Model Params	Training Time for 1k steps (s)
GNFactor	64.66M	976.5
Adapter (ours)	99.12M	844.8

Table 8: **Computation Cost.**

B Additional Results and Analysis

B.1 Additional Ablations Results

In Tab. 3, we only present the success rate data for the ablation studies on the GNFactor [17] framework. Here, we further provide detailed results of these ablation studies across all 10 tasks in Tab. 9. Additionally, we conduct ablation experiments on the 3D Diffuser Actor, and the corresponding results are shown in Tab. 10. These results can validate the effectiveness of our design of HyperTASR.

	Avg. Success	close jar	open drawer	sweep to dustpan	meat off grill	turn tap	slide block	put in drawer	drag stick	push buttons	stack blocks
GNFactor	33.3	**32.8**	36.0	48.0	51.2	**56.8**	20.0	8.8	69.6	5.6	4.0
HyperTASR w/ Feature Distillation	34.0	29.6	69.6	35.2	50.4	44.0	8.0	1.6	48.6	**43.2**	8.8
HyperTASR conditioned on τ	32.2	10.4	47.2	29.6	34.4	51.2	16.8	10.4	**92.0**	22.4	8.0
HyperTASR predicting θ and ω	36.3	28.0	67.2	20.0	**60.8**	49.6	20.0	18.4	78.4	7.2	**13.6**
GNFactor w/ HyperTASR	**42.6**	32.0	**75.2**	**66.4**	48.8	54.4	**23.2**	**22.4**	83.2	17.6	3.2

Table 9: **Detailed Ablation Study Results in GNFactor framework**. We report the average success rate across 5 evaluation seeds.

	Avg. Success	close jar	open drawer	sweep to dustpan	meat off grill	turn tap	slide block	put in drawer	drag stick	push buttons	stack blocks
3DDA	79.0	63.2	**88.8**	94.4	**84.8**	72.8	94.4	88.8	98.4	87.2	**17.4**
HyperTASR conditioned on τ	75.4	60.8	75.4	88.8	82.4	59.2	83.2	80.8	89.6	83.2	4.6
HyperTASR predicting θ and ω	79.2	66.4	86.4	96.8	81.4	79.8	84.0	87.2	99.2	89.6	10.4
3DDA w/ HyperTASR	**81.3**	**68.0**	87.2	**98.4**	82.4	**85.6**	**98.4**	**89.6**	**100.0**	**92.0**	11.2

Table 10: **Detailed Ablation Study Results in 3D Diffuser Actor (3DDA) framework**. We report the average success rate across 5 evaluation seeds.

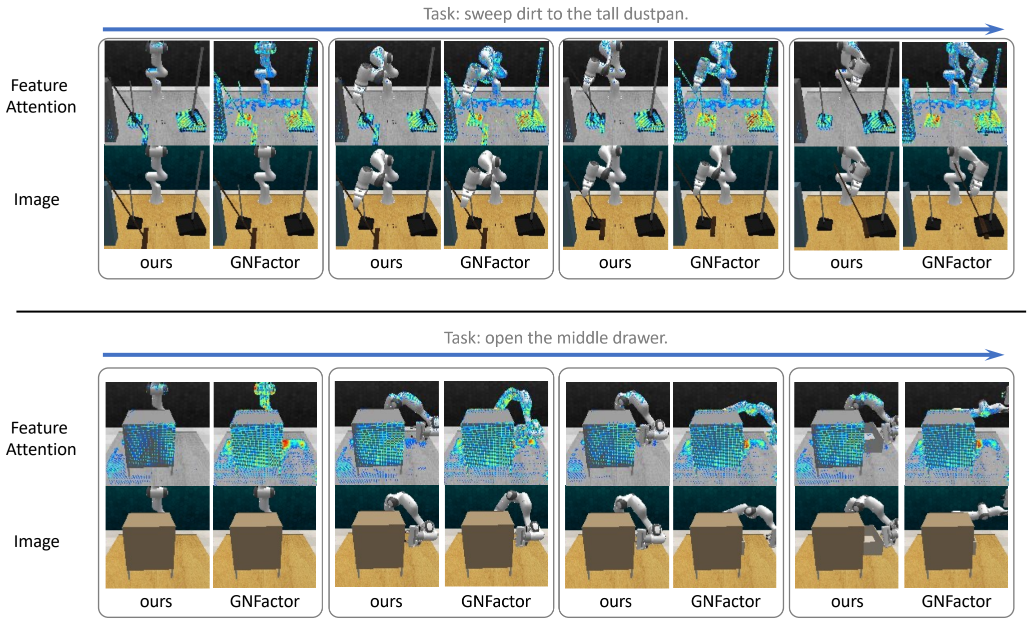


Figure 6: **Visulization Comparison of GNFactor [17] and Ours.**

B.2 Further Experiment Analysis

Experiments on Episode Length. While analyzing the success rate, we also collected statistics on the episode length of the evaluation episodes. The episode length refers to the average number of predicted keyposes or steps in each episode. A shorter episode length indicates fewer steps needed

	Avg. Length	close jar	open drawer	sweep to dustpan	meat off grill	turn tap	slide block	put in drawer	drag stick	push buttons	stack blocks
GNFactor	17.0	19.4	9.9	16.0	15.1	**11.0**	21.1	**16.8**	12.8	23.8	23.8
GNFactor w/ Adapter	**15.9**	**19.3**	**6.7**	**12.1**	**14.4**	11.7	**19.4**	20.7	**9.1**	**21.6**	**23.6**

Table 11: **Episode Length**. We report the average episode length across 5 evaluation seeds. As observed, by focusing on the more relevant portion of the scene with the task-aware representations, action efficiency is also improved.

to complete a task, suggesting a more efficient prediction method. As shown in Tab. 11, our method achieves a shorter episode length than GNFactor in most tasks, on average, 6.4% fewer steps across 10 tasks, which implies that the policy network makes more precise predictions and thus performs more efficiently.

Analysis of Reproducing Results of 3D Diffuser Actor. Due to differences in input image resolution, our reproduced results are slightly inferior to those proposed by 3D Diffuser Actor [18]. From the experimental results, it can be observed that for tasks where fine geometric details are crucial, such as ”close jar,” ”meat off grill,” and ”stack blocks,” our reproduced results perform poorly. This is consistent with the lack of detailed information in our data. On the other hand, we find that for tasks that only require determining the general position of an object, such as ”put in drawer” and ”push buttons”, our reproduced results significantly outperform those reported in the original paper. This highlights the significant influence that different representations (determined by input data) have on the current process of robot learning.

Ablation Results Analysis. From the comparison of ablation results, we observe that using only the task objective as the sole condition for the hypernetwork often leads to worse performance than the original codebase. We believe this is because, in RLBench, the ten selected tasks have limited variation in task objectives, with each variation corresponding to a unique task objective. As a result, training tends to lead to the hypernetwork memorizing the task objective rather than generalizing, turning the hypernetwork into a container for memorizing a few sets of parameters instead of a tool for dynamically adjusting the information extraction process. Consequently, the entire network is prone to significant overfitting, leading to poor evaluation results.

Analysis on Limited Improvement Compared to 3D Diffuser Actor Codebase. The experiments show that compared to our significant improvement on GNFactor framework, our method has limited improvement over the 3D Diffuser Actor. Through visualization of the representation compared to the input image, we observe that, compared to the representation of the 3D Diffuser Actor, our representation is primarily focused on task-relevant areas, while the 3D Diffuser Actor’s representation is more dispersed. From this perspective, our representation should significantly outperform that of the 3D Diffuser Actor during task execution. However, the final experimental results show limited improvement. We believe this is because the diffusion policy network has a strong capability for information extraction. During training, the diffusion policy not only extracts task-relevant information from the pre-trained backbone features but also further predicts action outcomes based on this information. Therefore, although our representation is better suited for learning manipulation tasks, the powerful policy network largely bridges the gap. In contrast, when using a relatively less powerful policy network, such as the Perceiver Actor, the performance improvement brought by the HyperTASR becomes much more significant.

Failure Case Analysis. In simulation experiments, the failure usually appears when accurate operation on tiny objects is needed. We conduct experiments on 128×128 resolution and 256×256 resolution, from which we observe that with higher resolution, the average success rate increased from 78.5% to 81.3%. Therefore, we believe that the capability of manipulating tiny objects are highly related to the input sensory data resolution. For real robot experiments, we define task success by finishing the task without significantly changing the position of other unrelated objects in the scene. In actual evaluation, many failures are caused by changing the position of other unrelated objects due to we do incorporate collision loss in real robot experiments.

B.3 Additional Visualization

We provide additional visualization results to further demonstrate the effectiveness of our approach. First, we compare the gradient visualizations of our method and GNFactor across more tasks in Fig. 6. From these results, it can be observed that, compared to GNFactor, our representation's attention map is more focused on task-relevant objects, whereas GNFactor's representation tends to allocate some attention to the background and objects unrelated to the task.

Next, we present a comparison of task execution between our method and GNFactor in Fig. 8. We provide RGB image sequences of the action execution. It can be seen that, compared to GNFactor, our approach more accurately identifies the locations of task-relevant objects, enabling more precise action execution and ultimately leading to successful task completion. In contrast, GNFactor often fails to complete the task due to getting stuck after an incorrect action execution. Meanwhile, we present a comparison of the real-world task execution of 3D Diffuser Actor and our HyperTASR in Fig. 7. More comparison are shown in Supplementary Video.

We also present the change in our representation during the action execution process in GNFactor tasks in Fig. 9. Specifically, for the ”stack blocks” task, our method shows high attention on a target block before placing it, and once the block is successfully placed, the attention on it significantly decreases. This indicates that the information regarding the block becomes less important after its placement in the context of completing the task.

Meanwhile, we provide visualizations of the gradients of our representation versus the input image for the 3D Diffuser Actor in Fig. 10. It is evident that, compared to the 3D Diffuser Actor, our method's attention is much more concentrated.

In addition, we provide attention visualization of real world experiments in Fig. 11. We compute the gradient of the representation with respect to the input image. We can observe that, compared with the 3D Diffuser Actor with the attention spread through the entire image, HyperTASR is much more focused on task-related objects. Meanwhile, during the task execution, we can observe that initially, attention is focused on the yellow cup and the gripper. As the yellow cup has been picked, the attention switches to the grey cup and the robotic arm. Finally, in the stacking process, the representation focuses on two cups again. This proves our HyperTASR generates representations that dynamically adapt as the task progresses. In Fig. 11, we visualize attention in real robot experiments by computing the gradient of the learned representation with respect to the input image of the training set. Unlike the 3D Diffuser Actor, whose attention is diffusely distributed across the scene, HyperTASR concentrates its attention on task-relevant objects. During the early grasping phase, attention is tightly focused on the yellow cup and the gripper. Once the yellow cup is lifted, attention shifts to the grey cup and the robotic arm. Finally, as the stacking motion commences, the model's attention returns to both cups. These observations demonstrate that HyperTASR produces dynamic, task-aware representations that track the evolving focus requirements throughout task execution.

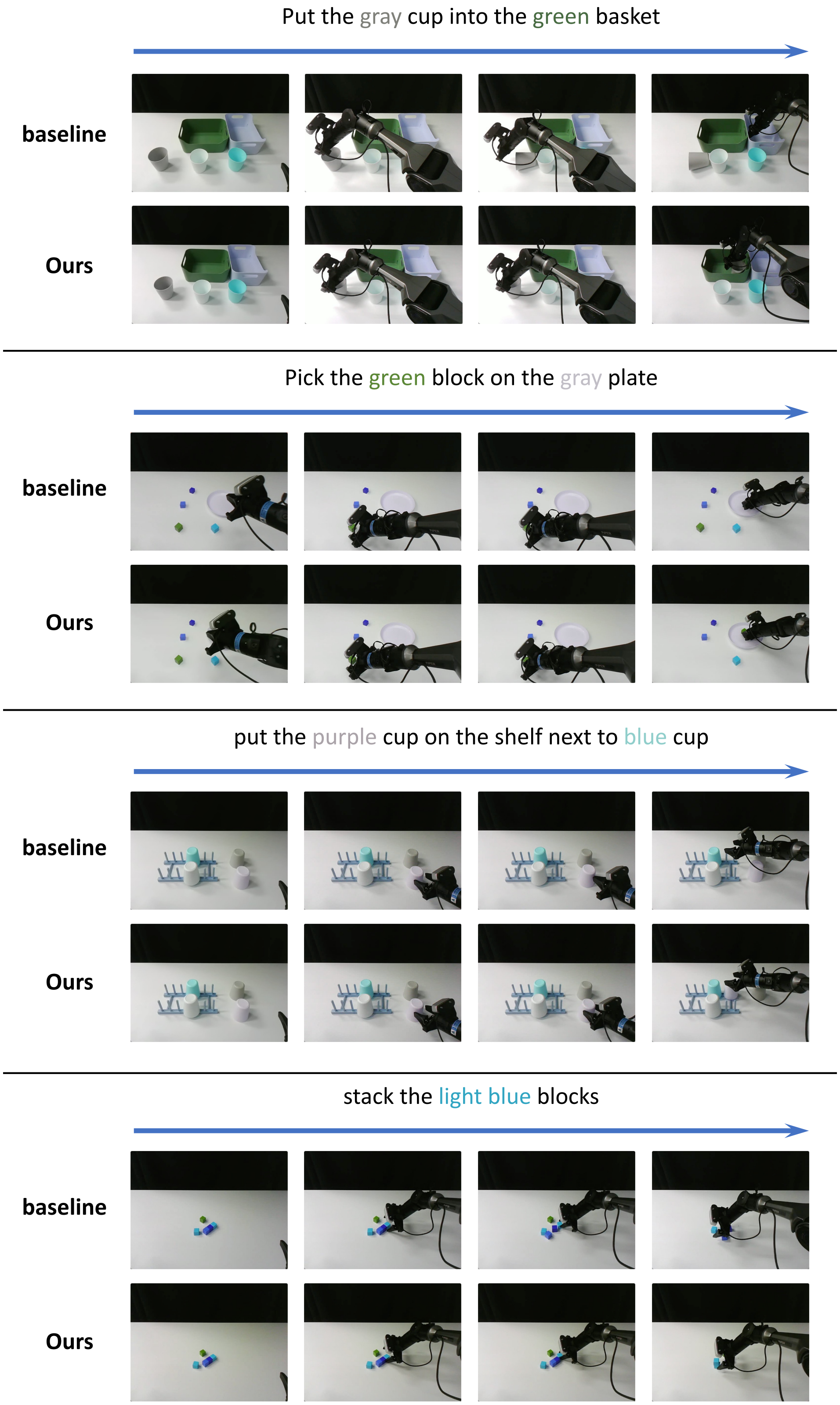

Put the gray cup into the green basket
baseline
Ours
Pick the green block on the gray plate
baseline
Ours
put the purple cup on the shelf next to blue cup
baseline
Ours
stack the light blue blocks
baseline
Ours

Figure 7: **Real World Task Execution Comparison of 3D Diffuser Actor and Ours.**

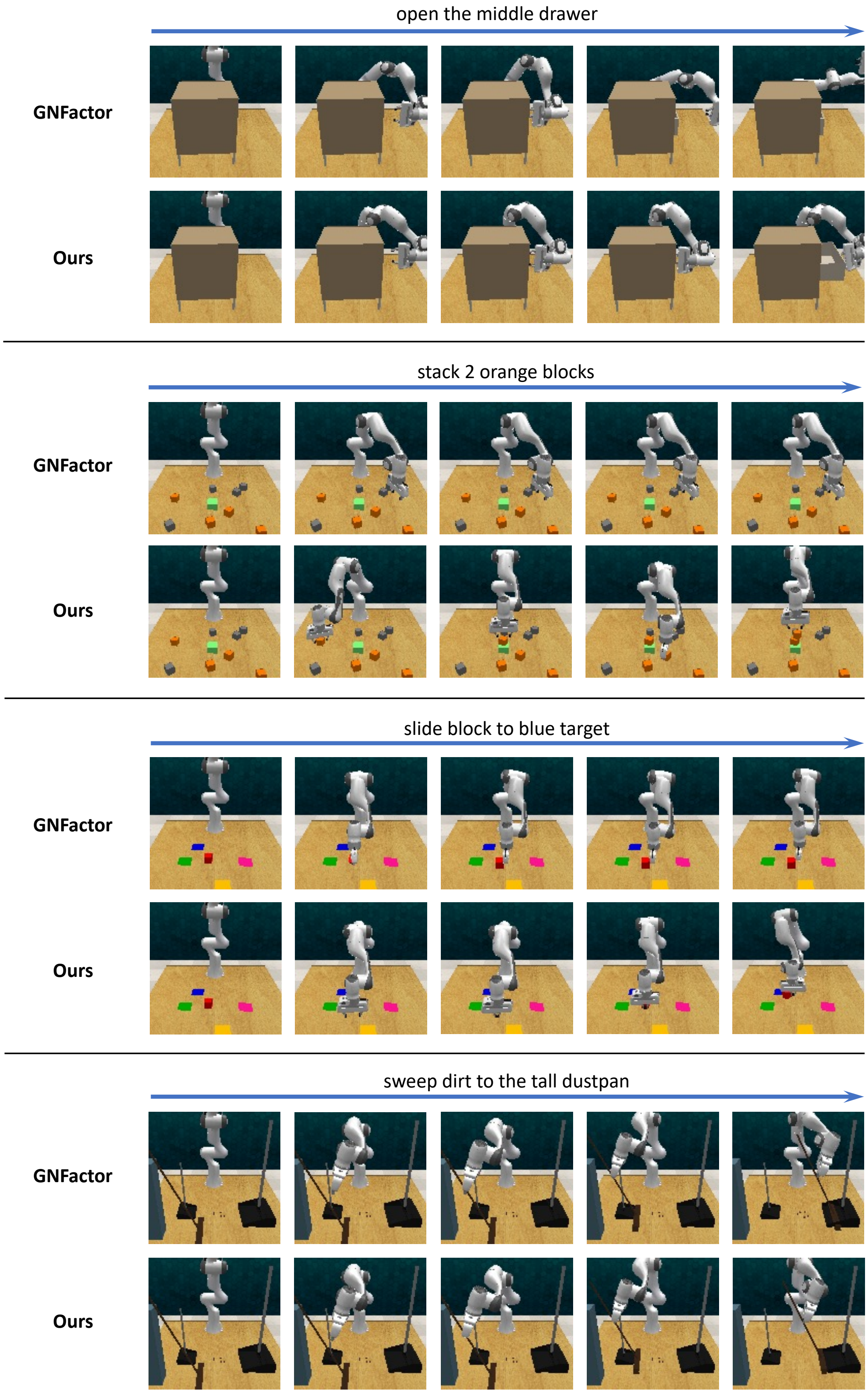


Figure 8: **Task Execution Comparison of GNFactor [17] and Ours.**

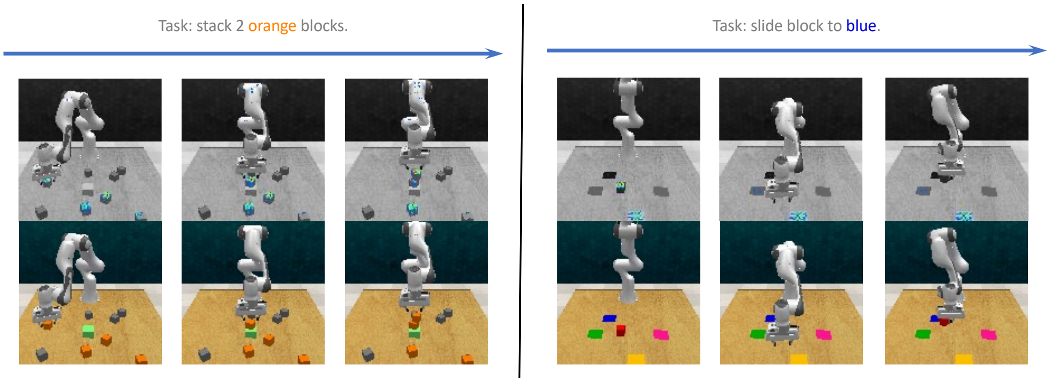


Figure 9: **Visualization Comparison regarding Task Progress for GNFactor [17] with our HyperTASR.**

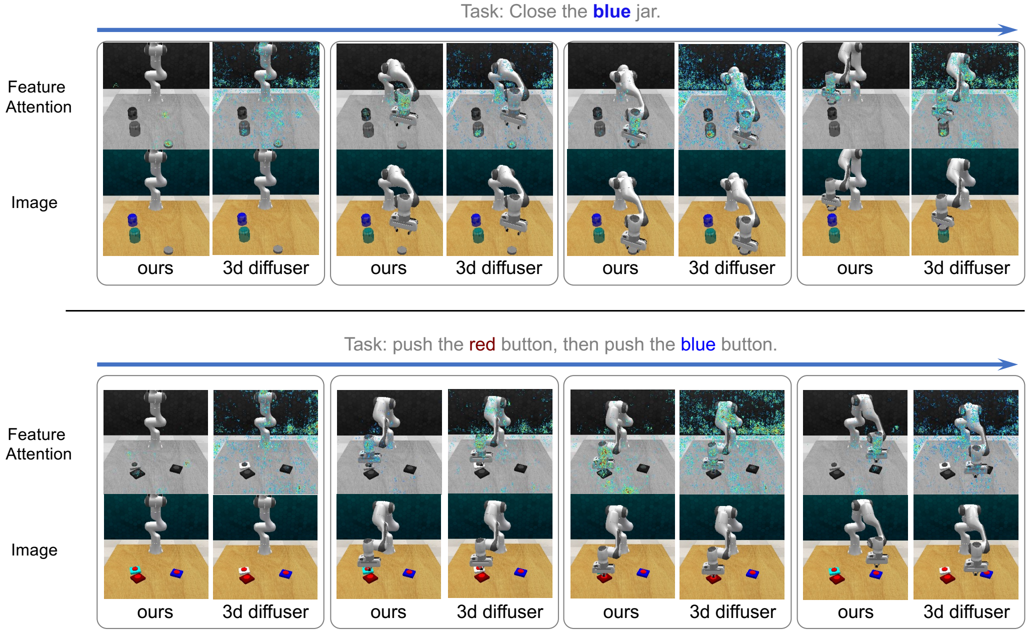

Task: Close the blue jar.
Feature Attention
Image
ours
3d diffuser
Task: push the red button, then push the blue button.

Figure 10: **Visualization Comparison of 3D Diffuser Actor [18] and Ours.**

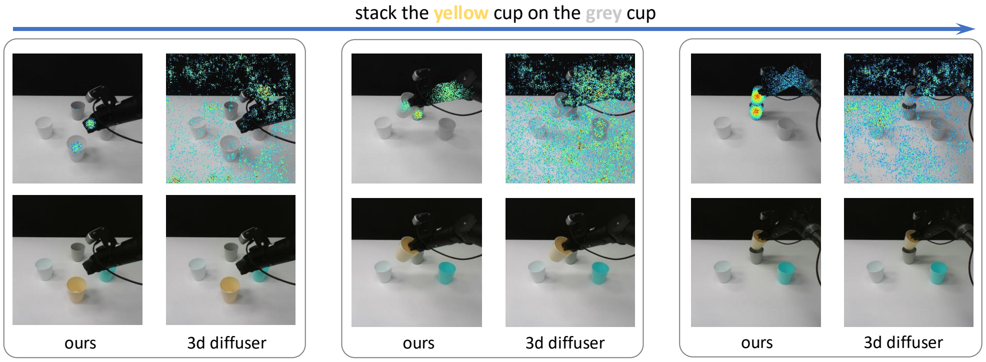


Figure 11: **Real Robot Visualization Comparison of 3D Diffuser Actor [18] and Ours.**