
Hyper-GoalNet: Goal-Conditioned Manipulation Policy Learning with HyperNetworks

Pei Zhou¹ Wanting Yao^{1,2*} Qian Luo¹ Xunzhe Zhou¹ Yanchao Yang¹

¹InfoBodied AI Lab, The University of Hong Kong ²University of Pennsylvania
{pezhou,qianluo1,xunzhe_zhou}@connect.hku.hk
wt Yao@seas.upenn.edu, yanchao@hku.hk

Abstract

Goal-conditioned policy learning for robotic manipulation presents significant challenges in maintaining performance across diverse objectives and environments. We introduce *Hyper-GoalNet*, a framework that generates task-specific policy network parameters from goal specifications using hypernetworks. Unlike conventional methods that simply condition fixed networks on goal-state pairs, our approach separates goal interpretation from state processing – the former determines network parameters while the latter applies these parameters to current observations. To enhance representation quality for effective policy generation, we implement two complementary constraints on the latent space: (1) a forward dynamics model that promotes state transition predictability, and (2) a distance-based constraint ensuring monotonic progression toward goal states. We evaluate our method on a comprehensive suite of manipulation tasks with varying environmental randomization. Results demonstrate significant performance improvements over state-of-the-art methods, particularly in high-variability conditions. Real-world robotic experiments further validate our method’s robustness to sensor noise and physical uncertainties. Code is available at: <https://github.com/wantingyao/hyper-goalnet>.

1 Introduction

Goal-conditioned policy learning enables embodied agents to adjust their actions based on current state observations and specified goals [8, 27, 46]. By integrating goal information into decision making, agents leverage knowledge across various tasks, enhancing adaptability [7, 12, 35] in hierarchical reinforcement learning and complex imitation learning [4, 17, 49].

Conventional approaches typically concatenate goal observations with current states as input to a fixed-parameter network [9, 56, 53]. This design creates a fundamental limitation: the network must process all possible goal-current state combinations using the same fixed weights, conflating “what” to process (current state) with “how” to process it (goal-dependent strategy). Consequently, these architectures often struggle with generalization to novel goals and complex manipulation tasks that require different processing strategies depending on the goal specification.

We *aim to* rethink this relationship by treating goals *not* as additional input features but as specifications that determine *how* current observations should be processed. Hypernetworks – neural networks that generate weights for another network – offer a natural implementation of this perspective. By explicitly modeling goals as determinants of policy parameters rather than as inputs, hypernetworks effectively disentangle task-dependent processing (defined by goals) from state-dependent processing (applied to current observations) [19, 45]. This approach better aligns with biological goal-directed

*Work done as an intern at HKU.

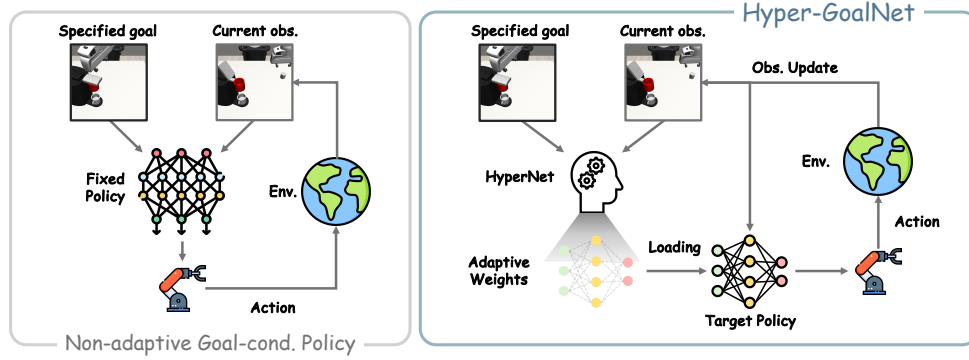


Figure 1: **The proposed Goal-Conditioned Policy Generation framework (Hyper-GoalNet) and conventional goal-conditioned policies.** Existing methods typically employ a fixed-parameter policy network that processes concatenated current observations and goal states, treating goals mostly as additional inputs. In *contrast*, our approach formulates policy learning as an adaptive generation task, where the goal image determines the parameters of the policy network itself – transforming goals from inputs into specifications that define *how* current observations should be processed. This allows for more effective handling of diverse goals and complex manipulation tasks.

behavior, where prefrontal regions interpret task goals and dynamically modulate processing in sensorimotor circuits accordingly [32, 51].

To address these challenges, we present **Hyper-GoalNet**, a hypernetwork-based framework for robotic manipulation illustrated in Fig. 1. Our approach employs a hypernetwork to dynamically generate target policy parameters conditioned on specified goals. When loaded into the policy network, these parameters enable the processing of current observations without requiring further access to goal information. This architecture creates a clear separation of concerns: the hypernetwork interprets what the goal means for processing strategy, while the generated policy focuses exclusively on transforming current observations into appropriate actions. A key advantage of this goal-aware design is that it enables the system to autonomously detect task completion during execution. By training the hypernetwork to model the conditional distribution of effective policy weights given goal specifications, we obtain a system that can adapt its processing strategy to diverse manipulation tasks.

Our technical contributions center on effectively applying hypernetworks for parameter-adaptive goal-conditioned policies learning. *First*, we adapt optimization-inspired hypernetwork architectures for generating policy parameters conditioned on goal specifications, creating a framework that dynamically determines how current observations should be processed. *Second*, we introduce an effective latent space shaping technique that imposes two critical properties: (1) predictability of future states through a learned dynamics model, and (2) preservation of physical relationships through distance constraints that ensure monotonic progression toward goals. These properties create an ideal representation space for our hypernetwork, providing clear signals about how policy parameters should change as states approach goals.

Our extensive experiments across multiple manipulation tasks show that Hyper-GoalNet significantly outperforms state-of-the-art methods, achieving higher success rates on complex contact-rich manipulations. Notably, while conventional approaches fail almost completely in high-variability environments, our method maintains robust performance. Ablation studies confirm the critical importance of our proposed components in the parameter-adaptive goal-conditioned policy learning framework. Finally, real-robot experiments demonstrate that our parameter-adaptive approach succeeds in physical environments where conventional methods struggle with sensor noise and environmental variations. These results confirm that explicitly modeling goals as determinants of processing strategy rather than as additional inputs creates a more effective and robust framework for goal-conditioned manipulation.

2 Related Work

Goal-conditioned policy. Goal-conditioned policy learning has attracted significant attention for developing versatile and generalizable agents [24, 52, 50, 57, 25]. Traditional methods augment

state spaces with goal information and train policies that condition on these augmented states [13, 44, 30, 54]. Hindsight Experience Replay (HER) [1] exemplifies this approach by allowing agents to learn from failures by reinterpreting unsuccessful outcomes as alternative goals. However, these methods typically suffer from increased complexity and require extensive tuning to manage the parameters associated with goal conditioning [39, 33, 48]. Our approach distinguishes itself by utilizing hypernetworks to dynamically generate policy weights, thereby reducing parameters in the policy network while enhancing scalability without extensive retraining [20, 43].

In the realm of imitation learning for goal-conditioned policies, several frameworks have demonstrated promising results by learning from pre-collected demonstrations [29, 9, 53]. Recent goal-conditioned behavior cloning approaches such as C-BeT [9] and MimicPlay [53] have advanced long-horizon manipulation tasks. However, these methods typically require sequences of achievable goal images, which are challenging to obtain in practice. Moreover, while performing well in basic pick-and-place scenarios, they often struggle with contact-rich tasks that demand precise environmental awareness [31]. Our method overcomes these limitations through effective latent space shaping, requiring only a single goal image while maintaining robustness across diverse manipulation scenarios.

Hypernetworks and Cognitive science insights for goal-directed behavior. Our work draws inspiration from cognitive science research on human goal-directed behavior, where meta-cognitive strategies and higher-level planning mechanisms enable adaptive actions [6, 36, 37]. Studies show that humans efficiently manage cognitive resources and flexibly adapt to different goals through higher-level representations [18, 11, 10]. Current policy learning methods incorporating cognitive principles often focus on imitation learning to mimic human strategies [2, 16, 14], but can be limited by demonstration quality and diversity [38, 42, 22]. Hypernetworks have been explored in robotic control [21, 55, 3], though primarily within reward-driven reinforcement learning (RL) settings [23, 5]. This fundamental difference in training paradigms, RL versus our reward-free behavior cloning (BC), means their end-to-end algorithms are not directly adaptable. We clarify, however, that their core hypernetwork architectures can be decoupled from the RL framework. By embedding cognitive insights into our hypernetwork architecture, we emulate human-like flexible adaptation [6, 36, 37]. Hyper-GoalNet’s capacity to generate goal-specific policy parameters without extensive retraining addresses practical challenges of goal-conditioned learning [20] while mirroring key cognitive mechanisms, offering a biologically plausible framework for adaptable policy generation.

3 Method

Let $\mathcal{D} = \{\tau_i\}_{i=1}^M$ be a dataset consisting of M robotic manipulation demonstrations, where each trajectory comprises a sequence of observation-action pairs, i.e., $\tau_i = \{(o_j^i, a_j^i)\}_{j=1}^{N_i}$, with a_j denoting a continuous-valued action and o_j representing a tuple containing high-dimensional state observations. Specifically, o_j includes an RGB image I_j captured by a single front-view camera, as well as the proprioceptive information s_j of the embodied agent. Given this formulation, our objective is to develop a *generalizable goal-conditioned policy learning framework* that enables efficient adaptation to diverse manipulation tasks.

We propose a shift from conventional goal-conditioned policies, which typically use fixed parameters while processing both current and goal images. Our key insight is that the goal image inherently specifies *how* the current image should be processed to generate appropriate actions. Therefore, we argue that the policy parameters themselves – which determine the processing mechanism – should adapt based on different goal specifications. To realize this insight, we leverage hypernetworks to dynamically generate task-specific policy parameters conditioned on goal images, rather than directly conditioning a fixed policy network on both current and goal observations. This approach creates a more flexible and efficient framework where the processing of current states is explicitly tailored to the specified goals.

The full pipeline is illustrated in Fig. 2. *Next*, we elaborate on the key components of our approach. In Sec. 3.1, we describe how we adapt hypernetwork architectures to effectively generate varying goal-reaching policies. In Sec. 3.2, we introduce an effective latent space shaping techniques that significantly enhance the performance of goal-conditioned policy generation by enforcing meaningful geometric structure in the representation space. *Finally*, in Sec. 3.3, we present the test-time inference pipeline that integrates our trained model (Hyper-GoalNet) to accomplish diverse manipulation tasks, highlighting the practical advantages of our parameter-adaptive approach.

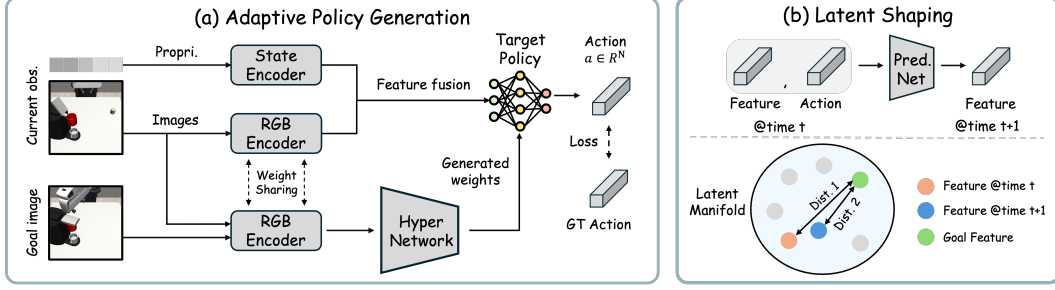


Figure 2: **An overview of the proposed Hyper-GoalNet framework.** (a) *Adaptive Policy Generation*: Unlike conventional approaches with fixed parameters, our hypernetwork dynamically generates task-specific policy parameters conditioned on goal images. This creates a parameter-adaptive target policy that processes current observations (RGB images and proprioception) through a multimodal encoder to produce actions tailored to specific goals. (b) *Latent Shaping*: Our approach enhances performance by explicitly structuring the latent space in two ways: a predictive network models state transitions to improve temporal dynamics, while geometric constraints ensure distances to goal states monotonically decrease during successful trajectories (detailed in Sec. 3.2).

3.1 Goal-Conditioned Hypernetworks

We formulate goal-conditioned policy learning as a parameter generation task rather than a direct conditioning problem. This formulation reframes the challenge from “what action to take given current and goal observations” to “what processing parameters to use given the goal.”

More formally, given a current observation o_c and a desired goal observation o_g , we model the conditional distribution over target policy weights that will transform the scene into the goal configuration. With the set of robotic manipulation demonstrations $\mathcal{D}$, we learn the distribution $\mathcal{H}(\theta \mid o_c, o_g)$:

$$\mathcal{H}(\theta \mid o_c, o_g) := \mathbb{P}_{\mathcal{D}}(\theta \mid o_c = I_t, o_g = I_{t'}), \text{ where } I_t, I_{t'} \in \tau_i \in \mathcal{D} \text{ and } t' > t. \quad (1)$$

For practical implementation, we focus on a single goal state rather than a sequence of goals, and use RGB image observations to condition the hypernetwork. Since our primary objective is to investigate the efficacy of goal-conditioned policy generation for manipulation tasks rather than developing a full probabilistic model, we approximate this as a deterministic mapping:

$$\mathcal{H} : \mathcal{O} \times \mathcal{O} \rightarrow \Theta, \quad (2)$$

where $\mathcal{O}$ denotes the observation space and Θ represents the space of target policy parameters. With a current-goal observation pair (o_c, o_g) , the hypernetwork $\mathcal{H}$ produces a policy that guides the transition from current state o_c to goal state o_g through action execution.

Hypernetwork architecture. To implement our parameter-adaptive approach, we adopt a hypernetwork architecture that efficiently generates policy parameters for achieving specified goals. The architecture must be capable of capturing the complex relationships between current states, goal states, and the required actions to bridge them.

We leverage an optimization-inspired architecture following [40], which provides beneficial inductive bias for our parameter generation task. This approach mimics iterative optimization by refining policy parameters through multiple feed-forward steps:

$$\theta^K = \mathcal{H}(o_c, o_g), \quad (3)$$

where θ^K represents the final policy parameters after K refinement iterations, with each update computed as:

$$\theta^k = \theta^{k-1} + \lambda^k(\theta^{k-1}, \alpha) \psi^k(\theta^{k-1}, \alpha), \quad \alpha = \phi(o_c, o_g). \quad (4)$$

Here, neural modules λ^k and ψ^k serve as learned analogs to step sizes and gradients in optimization, operating on embeddings $\phi(o_c, o_g)$ of the current and goal observations. This mechanism enhances the hypernetwork’s ability to generate effective task-specific policy parameters and improves generalization to new goal specifications. Once we obtain the goal-conditioned policy weights θ , we can process the current observation through the generated policy to predict appropriate actions.

Hypernetwork Training. We train our hypernetwork $\mathcal{H} : \mathcal{O} \times \mathcal{O} \rightarrow \Theta$ to generate parameters for the target visuomotor policy $\pi(\cdot; \theta)$ using behavior cloning (BC) on demonstration data. The generated policy takes the current observation o_t (comprising image I_t and proprioception s_t) and outputs actions for execution.

To enhance robustness, we utilize a sequence of L consecutive observations as input to the policy, capturing temporal dependencies under a non-Markovian assumption. The training objective minimizes the BC loss between demonstrated actions a_t^i and predicted actions $\hat{a}_t^i$:

$$\mathcal{L}_{\text{policy}} = \sum_{i=1}^M \sum_{1 \leq t < t' \leq N_i} \ell(a_t^i, \hat{a}_t^i), \quad \hat{a}_t^i = \pi(o_{t-L:t}^i; \mathcal{H}(o_t^i, o_{t'}^i)). \quad (5)$$

Here, ℓ represents the Mean Squared Error (MSE) loss. The end-to-end training procedure works as follows: for a given current observation o_t^i and a goal image $o_{t'}^i$, the hypernetwork $\mathcal{H}$ generates the weights for the policy network π . This goal-conditioned policy then processes the observation sequence $o_{t-L:t}^i$ to predict the action $\hat{a}_t^i$. The resulting loss is backpropagated through both the policy network and the hypernetwork. We restrict goal representations to image inputs since proprioceptive goal states may not always be available in practical applications. This formulation allows the hypernetwork to learn how to generate goal-specific processing mechanisms (target policy parameters) from visual goals, embodying our key insight that goal images determine *how* current observations should be processed.

3.2 Latent Space Shaping

A critical insight in our approach is that the effectiveness of parameter-adaptive policies depends significantly on the quality of the representation space in which observations are embedded. While our hypernetwork can directly generate policy parameters from raw observations, we find that explicitly shaping the latent representation space substantially enhances performance. Given the high dimensionality and redundant information in RGB images, we employ an image encoder $\mathcal{E}$ to extract task-relevant features and compress them into low-dimensional latents $z = \mathcal{E}(I)$.

We identify two fundamental properties that, when enforced in the latent space, particularly benefit our parameter-adaptive approach: *predictability* and *physical structure preservation*. The first property ensures the latent space facilitates modeling of state transitions, making the hypernetwork’s task of generating appropriate policy parameters more tractable. The second property ensures that the geometric relationships in the latent space meaningfully reflect physical relationships between states, enabling the generated policies to exploit these structured representations.

Enhancing predictability through dynamic modeling. To improve the predictability of latent representations, we introduce a dynamics model that forecasts future states in the latent space. By training this model to predict state transitions while simultaneously shaping the encoder $\mathcal{E}$ through backpropagation, we create a latent space where sequential relationships are explicitly captured. This significantly benefits our hypernetwork, as it needs to generate policy parameters that leverage these sequential relationships to guide transitions from current to goal states.

Formally, consider a discrete-time dynamical system with state representation $z_t \in \mathcal{Z}$ and control input $a_t \in \mathcal{A}$. The forward dynamic model is:

$$\hat{z}_{t+1} \sim p_{\Phi}(z_{t+1} \mid z_t, a_t), \quad (6)$$

where $z_t = \mathcal{E}(I_t)$ represents the latent encoding at time t , a_t is the executed action, and p_{Φ} is the transition dynamics parameterized by Φ . For practical implementation, we approximate this with a deterministic model $\Phi : \mathcal{Z} \times \mathcal{A} \rightarrow \mathcal{Z}$. The corresponding learning objective minimizes the prediction loss:

$$\mathcal{L}_{\text{pred}} = \mathbb{E}_{\tau \sim \mathcal{D}} [\ell(\Phi(z_t, a_t), z_{t+1})], \quad (7)$$

where ℓ is a distance metric in the latent space. When Φ is well-trained, we further finetune $\mathcal{E}$ through Φ to shape representations that capture both current state and potential transition information.

Preserving physical structure through distance constraints. For our parameter-adaptive approach to be effective, the latent space must preserve the physical structure of the task, particularly the progression towards goals. We formalize this as a requirement that the distance between the current state and goal state should monotonically decrease along goal-reaching trajectories. This property is

especially valuable for our hypernetwork, as it provides a clear signal about how policy parameters should change as states approach goals.

Specifically, for any goal-reaching trajectory $\tau_i \in \mathcal{D}$, where $\tau_i = \{(o_j^i, a_j^i)\}_{j=1}^{N_i}$, we enforce:

$$d_{\mathcal{E}}(o_j^i, o_{j'}^i) \geq d_{\mathcal{E}}(o_{j+1}^i, o_{j'}^i), \quad \forall j < j', \quad (8)$$

where $d_{\mathcal{E}}$ denotes a distance metric in the latent space. To explicitly model this behavior, we propose the following loss function:

$$\mathcal{L}_{\text{dist}} = \mathbb{E}_{\tau \sim \mathcal{D}} \sum_j \max(0, \beta + d(z_{j+1}, z_g) - d(z_j, z_g)), \quad (9)$$

where $d(z_1, z_2) = \|z_1 - z_2\|_2$ represents the Euclidean distance between latent features, $z_j = \mathcal{E}(I_j)$ and $z_g = \mathcal{E}(I_g)$ denote the image and goal image latents, respectively. The margin parameter $\beta \geq 0$ enforces a minimum decrease in distance between consecutive states and the goal. Empirically, setting $\beta = 0$ suffices to induce the desired monotonic progression.

This shaped latent space creates an ideal foundation for our parameter-adaptive approach, as it encodes both the predictive dynamics and geometric structure needed for the hypernetwork to effectively generate goal-tailored policy parameters. Figure 3 illustrates how our latent space shaping approach compares to alternative methods, showing the enhanced structure that benefits our goal-conditioned policy generation.

3.3 Hyper-GoalNet for Manipulation

Having established our parameter-adaptive architecture and latent space shaping techniques, we now present our complete framework, Hyper-GoalNet, which synthesizes these components for effective goal-conditioned manipulation. The **overall training objective** combines our policy generation loss with the latent space shaping terms:

$$\mathcal{L}_{\text{Hyper-GoalNet}} = \mathcal{L}_{\text{policy}} + \lambda_{\text{pred}} \mathcal{L}_{\text{pred}} + \lambda_{\text{dist}} \mathcal{L}_{\text{dist}}, \quad (10)$$

where λ_{pred} and λ_{dist} are weight coefficients balancing the contributions of predictability and structural constraints. The framework is trained end-to-end using gradient descent, allowing all components to co-adapt for optimal performance.

During **inference for task completion**, Hyper-GoalNet generates goal-specific policy parameters by feeding the concatenated latent features $[z_g, z_t]$ into the hypernetwork $\mathcal{H}$, where $z_g = \mathcal{E}(I_g)$ and $z_t = \mathcal{E}(I_t)$ are the latent representations of the goal and current observations. The generated goal-specific parameters $\theta = \mathcal{H}(z_t, z_g)$ are then loaded into the target policy $\pi(\cdot; \theta)$, which processes the current observation sequence to produce actions that guide the agent toward the goal.

Hyper-GoalNet offers two principal advantages over conventional goal-conditioned policies:

1) Parameter-adaptive policy generation: By dynamically synthesizing policy parameters based on goal specifications, our approach effectively changes how goal-conditioned policies operate. Rather than relying on a fixed network with static parameters to handle all possible goals, Hyper-GoalNet generates compact and efficient processing pathways that are tailored to specific goals.

2) Natural goal completion detection: Our shaped latent space provides an elegant solution to the challenging problem of goal completion detection. The distance metric $d_{\mathcal{E}}(o_t, o_g)$ in the latent space serves as a natural criterion for determining when a goal has been achieved, enabling autonomous goal transitions without external supervision.

The **test-time task evaluation** procedure for Hyper-GoalNet is formalized in Algorithm 1. Given an initial observation I_0 and goal observation I_g , the algorithm iteratively generates policy parameters, samples actions, and applies them until either the goal is reached (as determined by the latent distance falling below a threshold ϵ) or a maximum number of steps T is achieved. This simple yet effective procedure demonstrates how our parameter-adaptive approach seamlessly integrates into practical robotic control scenarios.

4 Experiments

In this section, we present a comprehensive experimental evaluation across a suite of simulated and real-robot manipulation tasks designed to address the following questions: 1) How effectively does

Algorithm 1 Hyper-GoalNet: Test-Time Task Evaluation

Input: Initial observation I_0 , Goal observation I_g
Modules: Encoder $\mathcal{E}$, Policy generation hypernetwork $\mathcal{H}$
Parameters: Max steps T , Goal completion threshold ϵ

```
1:  $I_t \leftarrow I_0, t \leftarrow 0, \text{done} \leftarrow \text{false}$ 
2: while  $t < T$  and not done do
3:    $\theta \leftarrow \mathcal{H}(\mathcal{E}(I_t), \mathcal{E}(I_g))$  ▷ Generate policy weights
4:    $\hat{a}_t \leftarrow \pi(o_{t-L:t}; \theta)$  ▷ Sample action
5:    $I_{t+1}, \text{done} \leftarrow \text{Env}(\hat{a}_t)$  ▷ Apply action and Env. interaction
6:   if  $d(\mathcal{E}(I_{t+1}), \mathcal{E}(I_g)) < \epsilon$  or done then
7:     return SUCCESS
8:   end if
9:    $I_t \leftarrow I_{t+1}, t \leftarrow t + 1$ 
10: end while
11: return TIMEOUT
```

Hyper-GoalNet’s parameter-adaptive approach generate successful policies for diverse manipulation tasks? 2) To what extent does our latent space shaping enhance the performance of the hypernetwork for goal-conditioned policy generation? 3) How does Hyper-GoalNet compare with conventional goal-conditioned methods and alternative representation learning approaches? Through extensive empirical analysis, we validate the effectiveness of our parameter-adaptive framework.

4.1 Experiment Setup

Simulation Environment. We evaluate our approach using Robosuite, a comprehensive robotics benchmark designed for both short and long-horizon manipulation tasks [31, 58]. This framework provides a standardized suite of environments, from which we select multiple contact-rich tabletop manipulation tasks: coffee manipulation, threading, mug cleanup, nut assembly, three-piece assembly, and several long-horizon tasks including coffee preparation and kitchen manipulation. To assess robustness across varying initial conditions, we use three difficulty levels (d_0, d_1, d_2), where higher indices correspond to increased environmental variability, particularly in object pose initialization (position and orientation). Each experimental environment features a robotic manipulator positioned adjacent to a workspace containing task-specific manipulable objects.

Training Protocol. Our approach follows the behavior cloning paradigm, utilizing a dataset based on MimicGen [31]. For each task, we employ a training dataset of 950 demonstrations, where each timestep comprises front-view RGB images (128×128 resolution), robot proprioceptive states $s_t \in \mathcal{S}$, and corresponding ground-truth actions $a_t \in \mathcal{A}$. The training procedure employs the Adam optimizer [26] with a cosine learning rate schedule [28]. We initialize the learning rate at 5×10^{-4} and maintain uniform loss balancing coefficients ($\lambda_i = 1$ for all components). Our model is trained for 500 epochs with a batch size of 256, by default. Detailed implementation and training protocols are provided in the Sec. D.

4.2 Main Results

Baselines. We compare our parameter-adaptive approach against state-of-the-art goal-conditioned methods that use fixed network parameters. All methods are trained on pre-collected demonstrations from MimicGen [31] and modified to operate with a single future image as the goal specification for fair comparison:

- **GCBC** [29, 15]: Goal-Conditioned Behavioral Cloning concatenates current and goal observations as input to a fixed policy network, learning a direct mapping from this concatenated representation to actions through supervised learning on demonstration data.
- **Play-LMP** [29]: Play-supervised Latent Motor Plans learns a latent plan space from demonstration data, then trains a fixed-parameter policy conditioned on both the current state and the inferred latent plan for the specified goal.

Table 1: **Comparison with state-of-the-art goal-conditioned methods.** Success rates (higher is better) are computed over 50 rollouts across various manipulation tasks with increasing difficulty levels (d0-d2). The experimental setup uses only two historical observations and a single goal image, representing a practical deployment scenario. Our method consistently outperforms conventional fixed-parameter approaches, demonstrating the effectiveness of dynamically generating policy parameters based on goal specifications.

Method	Coffee $\uparrow$				Mug-cleanup $\uparrow$			Three piece Assemb. $\uparrow$				Threading $\uparrow$				Nut Assemb. $\uparrow$	
	d0	d1	d2	Avg.	d0	d1	Avg.	d0	d1	d2	Avg.	d0	d1	d2	Avg.	d0	Avg.
GCBC	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Play-LMP	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MimicPlay	0.28	0.28	0.16	0.24	0.26	0.06	0.16	0.06	0.06	0.00	0.04	0.18	0.02	0.00	0.07	0.03	0.12
C-BeT	0.92	0.00	0.74	0.55	0.30	0.50	0.40	0.00	0.02	0.00	0.01	0.62	0.22	0.12	0.32	0.34	0.32
Ours	0.94	0.76	0.62	0.77	0.78	0.46	0.62	0.52	0.20	0.04	0.25	0.82	0.32	0.24	0.46	0.55	0.52

Table 2: **Long-horizon task performance.** Our parameter-adaptive approach excels in complex sequential tasks, outperforming fixed-parameter methods across difficulty levels.

Method	Coffee Preparation $\uparrow$			Kitchen $\uparrow$			Avg.
	d0	d1	Avg.	d0	d1	Avg.	
GCBC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Play-LMP	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MimicPlay	0.34	0.00	0.17	0.86	0.18	0.52	0.35
C-BeT	0.82	0.04	0.43	0.78	0.70	0.74	0.59
Ours	0.80	0.50	0.65	1.00	0.80	0.90	0.78

Table 3: **Component ablation analysis.** Latent space shaping significantly enhances performance, while proper distance metrics and training are crucial for robustness.

Method	Coffee $\uparrow$			
	d0	d1	d2	Avg.
Ours(uf. at epoch 0)	0.92	0.00	0.62	0.51
Ours(w/o shaping)	0.92	0.00	0.00	0.31
Ours(dist $\leftrightarrow$ start img.)	0.50	0.52	0.32	0.45
Ours(cos dist)	0.94	0.36	0.48	0.59
C-BeT(w/ shaping)	0.80	0.64	0.64	0.69
Ours	0.94	0.76	0.62	0.77

- **C-BeT** [9]: Conditional Behavior Transformer uses self-attention to compress observation history into a latent representation, which is combined with the goal state to condition a fixed-parameter transformer that predicts actions.
- **MimicPlay** [53]: MimicPlay is a self-supervised approach that learns general robotic skills from unstructured teleoperation data, which consists of continuous sequences of observations and actions from a human video. For our experiments, this method is adapted into a goal-image-conditioned policy, with implementation details provided in the Appendix.

Evaluation Metrics. We evaluate performance using task completion success rates over 50 independent rollouts with **randomly initialized**, previously unseen environmental configurations. We impose maximum trajectory lengths of $T = 600$ or 800 steps for contact-rich tasks and $T = 1600$ for long-horizon tasks. While our parameter-adaptive approach enables autonomous task completion detection through latent space metrics (Algorithm 1), we use environment-provided terminal signals for standardized evaluation across all methods. Success is indicated as $\mathcal{S}_i = 1$ if rollout i completes within T steps, and $\mathcal{S}_i = 0$ otherwise.

Quantitative Results. Tables 1 and 2 present success rates across multi-step and long-horizon tasks, respectively. For each task, success is determined by task-specific criteria provided by the environment – such as correct object placement, successful insertion, proper assembly configuration, or completion of a sequence of subtasks for long-horizon scenarios. Our parameter-adaptive approach outperforms fixed-parameter methods across these diverse evaluation criteria and difficulty levels. This superior performance stems from our hypernetwork’s ability to dynamically generate task-specific policy parameters tailored to each goal, resulting in more effective goal-directed behavior. Particularly in high-variability environments (difficulty levels d1-d2), our method demonstrates greater robustness and adaptability compared to conventional approaches – highlighting the advantage of having policy parameters explicitly conditioned on goals rather than using fixed parameters for all scenarios.

Analysis of Likelihood-Based Baselines. We diagnose the poor performance of GCBC and Play-LMP as a fundamental issue of their learning objective, not implementation. These methods, adapted from reputable third-party code, aim to maximize the log-likelihood of expert actions. We found this leads to severe overfitting on the training data, evidenced by a large gap between low training loss and high validation loss. Such memorization-based learning fails to generalize to the subtle variations

present in our high-precision test scenarios. This limitation of likelihood-based models in complex settings is corroborated by prior work [9]. In contrast, our hypernetwork’s design imposes a beneficial structural bias: by emulating an optimization process to generate parameters, it is incentivized to learn a functional mapping from goal to policy, ensuring better generalization and avoiding the overfitting issues that plague the baselines.

4.3 Ablation Study

Hypernetwork Architecture Analysis. We evaluate our optimization-inspired hypernetwork design against HyperZero [41], a prominent alternative architecture. Architecturally, HyperZero encodes conditioning information into a meta-embedding that is then transformed to produce the parameters for the target policy network. Because this direct mapping can produce parameters with a numerical range misaligned with that of an optimally trained network, it often requires special initialization to stabilize training. To ensure a stable and robust comparison, we therefore implemented two enhanced initialization schemes for HyperZero. The first, *ScalarInit*, introduces a learnable scalar to control the initial scale of the hypernetwork’s output. The second, *Bias-Init* [5], is designed for high-dimensional conditioning inputs and constrains the parameter range by incorporating learnable biases alongside weights that are initialized to zero. As shown in Table 4, even with these stabilization techniques, our architecture, which requires no special initialization, vastly outperforms HyperZero. This demonstrates that the iterative refinement mechanism in our design is inherently more effective at capturing the relationship between goals and appropriate policy parameters. These results highlight the critical role of architectural choice in developing robust parameter-adaptive policies. More details can be found in Sec. C.

Latent Space Shaping Analysis. Table 3 shows that proper latent space shaping is critical for our parameter-adaptive framework. Removing shaping (“Ours w/o shaping”) severely degrades performance on harder difficulty levels, while our specific choices of using goal-relative distances and Euclidean metrics prove superior to alternatives. Figure 3 visually confirms how our approach creates more consistent monotonic progression toward goals compared to unshaped representations. Notably, applying our shaping techniques to the baseline C-BeT improves its performance, yet its performance lags compared to ours, which in turn signifies the importance of the parameter-adaptive framework as well as the synergy between policy generation and latent shaping. Our training strategy also matters – unfreezing the R3M visual encoder [34] only after 20 epochs ensures stable parameter generation. These results validate our insight that latent spaces should reflect physical progression toward goals to effectively support parameter-adaptive policy learning.

Visualization. Figures 3 and 6 illustrate how our latent space shaping creates representations that directly benefit our parameter-adaptive approach. The plot shows the latent distance to the goal (y-axis) over the execution timesteps of a robotic task rollout (x-axis), where later steps are progressively closer to the goal. Unlike R3M embeddings which show significant fluctuations, our method produces consistently monotonic distance reductions toward goal states. This structured latent space offers two key advantages for our hypernetwork: (1) it provides clearer signals for generating appropriate policy parameters as the agent progresses toward goals, and (2) it enables reliable autonomous detection of task completion based on latent distances. The visualization confirms that our combined predictive modeling and distance constraints create opti-

Table 4: **Ablating hypernetwork architectures.** Our method with standard initialization is compared against HyperZero variants stabilized with enhanced initializations.

Method	Coffee Task (%) $\uparrow$			
	d0	d1	d2	Avg.
HyperZero + <i>ScalarInit</i>	16	18	14	16
HyperZero + <i>Bias-Init</i>	30	18	0	16
Ours (Standard Init.)	94	76	62	77

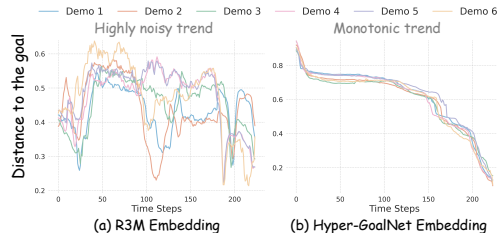


Figure 3: Comparison of (a) unshaped R3M embeddings versus (b) our shaped latent space, showing L_2 distances to goal states along multiple trajectories. Our shaping creates consistent monotonic decreases in distance-to-goal, facilitating more effective parameter generation.

mal representations for goal-conditioned parameter generation, enhancing both performance and interpretability.

Goal Completion Detection. To substantiate our claim that the shaped latent space enables autonomous goal completion detection, we supplement the qualitative evidence from Figures 3, 9, and 10. We evaluate this capability by comparing our autonomous detection, where success is determined by a latent distance threshold (Auto SR), against the environment’s ground-truth signal (Env SR). Table 5 presents the results using two key metrics: *Accuracy*, which measures the agreement between the two signals, and *Recall*, which measures our method’s ability to identify true successes reported by the environment. The strong alignment, evidenced by an average accuracy of 86.6% and recall of 94.3%, provides strong empirical evidence that our latent-distance-based approach is a reliable autonomous completion detector.

Table 5: **Quantitative validation of autonomous goal completion detection.** Our method’s latent distance-based success rate (Auto SR) is compared against the environment’s ground truth (Env SR) on the Coffee tasks. High Accuracy and Recall validate its reliability.

Task	Auto SR	Env SR	Accuracy	Recall
D0	0.96	0.94	94%	98%
D1	0.78	0.76	90%	95%
D2	0.74	0.62	76%	90%
Mean	–	–	86.6%	94.3%

4.4 Real Robot Experiments

We validate our parameter-adaptive approach on physical hardware using the Realman Robotics Platform, featuring a 7-DoF manipulator with a 1-DoF parallel gripper (Figure 7). We evaluate four diverse manipulation tasks – sweep, pick&place, pull, and stack – with 15 trials per task. Due to hardware constraints limiting control to joint angles without end-effector pose information, we exclude MimicPlay, which requires precise 3D end-effector trajectories.

Table 6: **Real-robot experiment results.** (successes/total trials).

Method	Pickplace	Pull	Stack	Sweep
GCBC	0/15	0/15	0/15	2/15
Play-LMP	0/15	0/15	0/15	5/15
C-BeT	2/15	6/15	5/15	8/15
Ours	14/15	15/15	14/15	15/15

As shown in Table 6, conventional fixed-parameter approaches struggle significantly in real-world conditions where environmental noise, perception uncertainties, and imperfect demonstrations create substantial challenges. In contrast, our method maintains high success rates across all tasks, including those requiring precise contact-rich interactions. This real-world performance gap highlights a key advantage of our parameter-adaptive approach: by dynamically generating task-specific policy parameters based on goal images, our method better adapts to real-world variations and demonstration imperfections that weren’t encountered during training. Detailed experimental protocols are provided in the Sec. F.

5 Discussion

Conclusion. Our parameter-adaptive approach represents an effective move in goal-conditioned policy learning by dynamically generating policy parameters based on goal information rather than using fixed parameters with conditioning. The consistent performance improvements across tasks demonstrate that “how” observations should be processed is inherently dependent on the goal specification. Our latent space shaping techniques prove critical for this architecture – imposing physical structure and predictive capacity provides clearer signals for the hypernetwork to generate effective policy parameters. Overall, our results suggest that explicitly modeling the relationship between goals and processing mechanisms offers a promising direction for more flexible and robust robotic control.

Limitations. Our method’s primary limitation is its reliance on a well-structured latent space, which is challenging to form for highly complex tasks and requires demonstration data with clear goal progression. This data dependency creates a key failure mode: out-of-distribution goals can cause the hypernetwork to generate erratic policies. Furthermore, as an offline-trained method, its dynamic generation of parameters for novel goals lacks explicit safety guarantees against unforeseen states, making the integration of robust safety constraints a critical direction for future research.

Acknowledgments and Disclosure of Funding

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] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.
- [2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.
- [3] Sayantan Auddy, Jakob Hollenstein, Matteo Saveriano, Antonio Rodríguez-Sánchez, and Justus Piater. Scalable and efficient continual learning from demonstration via a hypernetwork-generated stable dynamics model. *arXiv preprint arXiv:2311.03600*, 2023.
- [4] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.
- [5] Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in meta-reinforcement learning. In *Conference on Robot Learning*, pages 1478–1487. PMLR, 2023.
- [6] Matthew M Botvinick, Yael Niv, and Andrew G Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *cognition*, 113(3):262–280, 2009.
- [7] Qingwen Bu, Jia Zeng, Li Chen, Yanchao Yang, Guyue Zhou, Junchi Yan, Ping Luo, Heming Cui, Yi Ma, and Hongyang Li. Closed-loop visuomotor control with generative expectation for robotic manipulation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- [8] Xiaoyu Chen, Junliang Guo, Tianyu He, Chuheng Zhang, Pushi Zhang, Derek Cathera Yang, Li Zhao, and Jiang Bian. Igor: Image-goal representations are the atomic control units for foundation models in embodied ai. *arXiv preprint arXiv:2411.00785*, 2024.
- [9] Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.
- [10] Nathaniel D Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12):1704–1711, 2005.
- [11] Peter Dayan and Nathaniel D Daw. Decision theory, reinforcement learning, and the brain. *Cognitive, Affective, & Behavioral Neuroscience*, 8(4):429–453, 2008.
- [12] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.
- [13] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.
- [14] Yan Duan, Xi Chen, Rein Houthoofd, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.
- [15] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

- [16] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29, 2016.
- [17] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.
- [18] Michael J Frank, Lauren C Seeberger, and Randall C O’reilly. By carrot or by stick: cognitive reinforcement learning in parkinsonism. *Science*, 306(5703):1940–1943, 2004.
- [19] Tomer Galanti and Lior Wolf. On the modularity of hypernetworks. *Advances in Neural Information Processing Systems*, 33:10409–10419, 2020.
- [20] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.
- [21] Shashank Hegde, Zhehui Huang, and Gaurav S Sukhatme. Hyperppo: A scalable method for finding small policies for robotic control. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10821–10828. IEEE, 2024.
- [22] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.
- [23] Yizhou Huang, Kevin Xie, Homanga Bharadhwaj, and Florian Shkurti. Continual model-based reinforcement learning with hypernetworks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 799–805. IEEE, 2021.
- [24] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.
- [25] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
- [26] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [27] Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.
- [28] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.
- [29] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pages 1113–1132. PMLR, 2020.
- [30] Jason Yecheng Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. Offline goal-conditioned reinforcement learning via f -advantage regression. *Advances in Neural Information Processing Systems*, 35:310–323, 2022.
- [31] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.
- [32] Earl K Miller and Jonathan D Cohen. An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24(1):167–202, 2001.
- [33] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.
- [34] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

- [35] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. *Advances in neural information processing systems*, 32, 2019.
- [36] Randall C O’Reilly and Michael J Frank. Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*, 18(2):283–328, 2006.
- [37] Giovanni Pezzulo and Paul Cisek. Navigating the affordance landscape: feedback control as a process model of behavior and cognition. *Trends in cognitive sciences*, 20(6):414–424, 2016.
- [38] Dean A Pomerleau. Alvin: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.
- [39] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.
- [40] Hanxiang Ren, Li Sun, Xulong Wang, Pei Zhou, Zewen Wu, Siyan Dong, Difan Zou, Youyi Zheng, and Yanchao Yang. Hypogen: Optimization-biased hypernetworks for generalizable policy generation. In *The Thirteenth International Conference on Learning Representations*.
- [41] Sahand Rezaei-Shoshtari, Charlotte Morissette, Francois R Hogan, Gregory Dudek, and David Meger. Hypernetworks for zero-shot transfer in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9579–9587, 2023.
- [42] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.
- [43] Cicero Nogueira dos Santos, Youssef Mroueh, Inkit Padhi, and Pierre Dognin. Learning implicit generative models by matching perceptual features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4461–4470, 2019.
- [44] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.
- [45] Simon Schug, Seijin Kobayashi, Yassir Akram, João Sacramento, and Razvan Pascanu. Attention as a hypernetwork. *arXiv preprint arXiv:2406.05816*, 2024.
- [46] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575. IEEE, 2021.
- [47] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.
- [48] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.
- [49] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.
- [50] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.
- [51] I Toni. The cerebellum and parietal cortex play a specific role in coordination: a pet study. *Neuroimage*, 14(4):899–911, 2001.

- [52] Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.
- [53] Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.
- [54] Huilin Xu, Jian Ding, Jiakun Xu, Ruixiang Wang, Jun Chen, Jinjie Mai, Yanwei Fu, Bernard Ghanem, Feng Xu, and Mohamed Elhoseiny. Diffusion-based imaginative coordination for bimanual manipulation. *arXiv preprint arXiv:2507.11296*, 2025.
- [55] Hongxiang Yu, Anzhe Chen, Kechun Xu, Zhongxiang Zhou, Wei Jing, Yue Wang, and Rong Xiong. A hyper-network based end-to-end visual servoing with arbitrary desired poses. *IEEE Robotics and Automation Letters*, 8(8):4769–4776, 2023.
- [56] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.
- [57] Pei Zhou, Ruizhe Liu, Qian Luo, Fan Wang, Yibing Song, and Yanchao Yang. Autocgp: Closed-loop concept-guided policies from unlabeled demonstrations. In *The Thirteenth International Conference on Learning Representations*.
- [58] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

Appendix

In this section, we present supplementary materials detailing the methodological framework and experimental procedures used in this study.

A Manipulation Task Details

Task Details. Our experimental evaluation was conducted within the Robosuite [58] simulation environment, utilizing the benchmark dataset from Mimicgen [31]. We primarily investigated complex tabletop manipulation tasks that encompass diverse robotic skills. The selected tasks are characterized as follows:

- **Coffee:** A multi-step manipulation task requiring precise object handling, where the robot must grasp a coffee capsule, insert it into the designated slot of the coffee machine, and securely close the machine’s lid.
- **Mug cleanup:** A sequential task involving both articulated object interaction and object placement. The robot must coordinate drawer manipulation and object transportation, culminating in storage of a mug.
- **Three piece assembly:** A multi-step assembly task demanding spatial reasoning and precise manipulation. The robot must stack three components in a specific sequence to achieve compact assembly configuration.
- **Threading:** A high-precision manipulation task requiring fine motor control. The robot must accurately orient and manipulate a needle for successful insertion through a minimal aperture.
- **Nut Assembly:** A manipulation task involving precise grip control and spatial alignment for successful mechanical assembly. In this task, the success rates for the two nuts are measured separately, and the overall success rate is then calculated.
- **Coffee Preparation:** An extended sequential task combining multiple sub-goals, including cup positioning, drawer manipulation, capsule retrieval, and coffee maker operation, culminating in a fully prepared coffee setup.
- **Kitchen:** A complex sequence involving appliance interaction, object manipulation, and spatial reasoning. The task includes stove operation, cookware handling, and precise object placement.

These tasks are specifically selected for their comprehensive representation of challenging robotic manipulation scenarios, featuring contact-rich interactions, precise object manipulation, and complex multi-step sequences. Each task requires a combination of skills including spatial reasoning, and sequential decision-making. The visual representation of these manipulation tasks and their key phases are illustrated in Figure 4.

Data Processing and Observation Space. The demonstration data from Mimicgen is initially preprocessed by segmenting complete demonstrations into trajectory subsequences to facilitate learning. Our framework implements a constrained observation context with a length of 2, including front-view RGB images and the agent’s proprioceptive state information. This design choice means that the agent’s decision-making process is based solely on the current frame and one historical frame, deliberately limiting the temporal horizon to enhance real-world applicability. Given our focus on goal-conditioned policy learning, the observation space of the hypernetwork is augmented with a single RGB goal image representing a feasible target state. The input modalities are structured as follows:

- Visual observations: RGB images with dimensions 128×128 pixels for both contextual and goal representations.
- Proprioceptive state: A compact 9-dimensional vector encoding essential agent state information.

This deliberately constrained observation space creates a partially observable environment that closely aligns with real-world robotics scenarios, where complete state information is rarely available. While

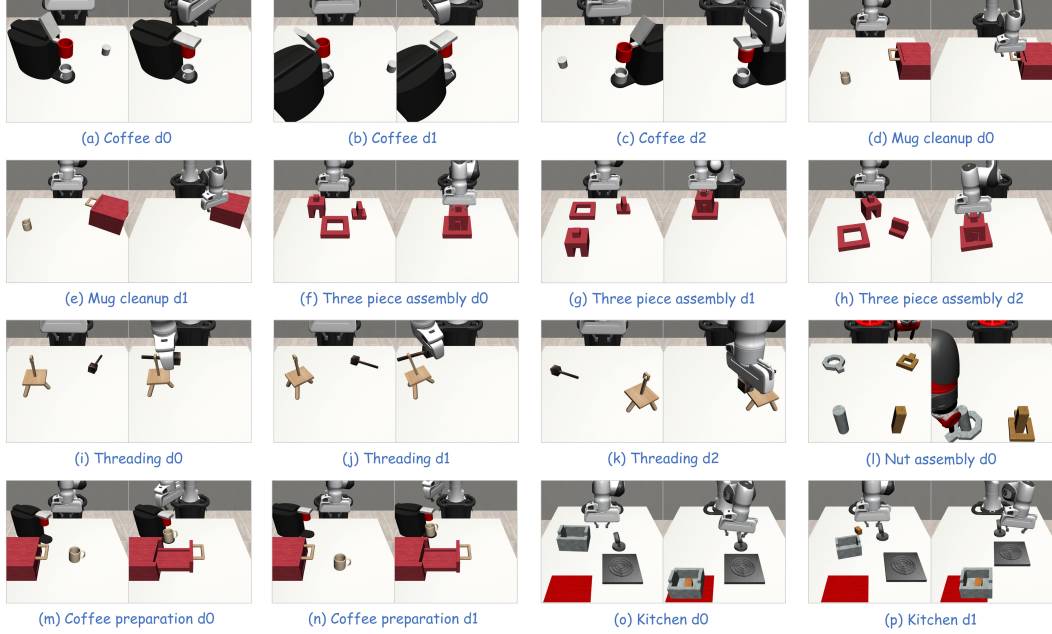


Figure 4: Key phases of diverse manipulation tasks in experimental evaluation.

this design choice enhances the practical applicability of our approach, it also introduces significant challenges:

- Single-goal scenarios necessitate hypernetwork architectures to efficiently handle the demanding requirements of goal-conditioned policy generation.
- Limited temporal context requiring efficient use of historical information.
- Partial observability demanding robust state estimation and feature extraction.
- Complex vision-based reasoning with constrained visual information.

Such challenging conditions serve to validate our method’s effectiveness under realistic constraints, demonstrating its potential for real-world deployment.

Goal Specification Strategy. Our approach implements a systematic strategy for goal specification across both training and evaluation phases. During training, we employ a dynamic goal sampling mechanism where the goal image is stochastically selected from future frames within the same demonstration, subsequent to the current timestep. This design offers two key advantages:

- **Goal Feasibility:** By sampling from actual demonstration frames, we inherently guarantee the physical feasibility and reachability of the specified goals.
- **Goal Diversity:** The random sampling mechanism ensures sufficient variation in goal states, promoting the learning of a robust and generalizable policy.

During the evaluation phase, the goal specification mechanism leverages the Mimicgen framework to generate feasible goal states, facilitating potential transfer from simulation to physical systems.

B Comparison with Baselines

Given the challenging nature of our experimental setting as shown in Table 1, existing baseline methods demonstrate limited success in task completion. To ensure comprehensive evaluation, we introduce modified versions of existing approaches and additional baseline methods adapted for our scenario. The following section details these enhanced baseline implementations and their comparative performance.

Table 7: **Additional quantitative experiments with more baseline methods across different tasks.**

† indicates methods using a **sequence of visual frames** as goals rather than a single goal image, with an **extended observation length** of 10 frames versus the standard 2. ‡ indicates methods with access to **extra** wrist-mounted camera images. The wrist views, extended observation sequence, and extended goal sequences provide richer observational and task-guidance information for augmented baselines. Hyper-GoalNet leverages only one goal image and two front-view observations, indicating a more easy-to-be-applied setup while being more effective.

Method	Coffee †				Mug-cleanup †			Three piece Assemb. †				Threading †				Nut Assemb. †		Avg.
	d0	d1	d2	Avg.	d0	d1	Avg.	d0	d1	d2	Avg.	d0	d1	d2	Avg.	d0		
MimicPlay-O ^{†‡}	0.80	0.84	0.88	0.84	0.68	0.58	0.63	0.50	0.38	0.02	0.30	0.32	0.06	0.06	0.15	0.10		0.44
MimicPlay-M [†]	0.28	0.28	0.16	0.24	0.26	0.06	0.16	0.06	0.06	0.00	0.04	0.18	0.02	0.00	0.07	0.03		0.12
C-BeT	0.92	0.00	0.74	0.55	0.30	0.50	0.40	0.00	0.02	0.00	0.01	0.62	0.22	0.12	0.32	0.34		0.32
Hyper-GoalNet(G)	0.94	0.60	0.62	0.72	0.72	0.54	0.63	0.46	0.22	0.02	0.23	0.78	0.20	0.18	0.39	0.67		0.50
Hyper-GoalNet	0.94	0.76	0.62	0.77	0.78	0.46	0.62	0.52	0.20	0.04	0.25	0.82	0.32	0.24	0.46	0.55		0.52

We also conducted additional comparative experiments with BeT [47], C-BeT [9], and MimicPlay [53] (implemented in both its original setting and a modified setting). And a variant of our method, Hyper-GoalNet(G) is also introduced. Experiments were performed across all 16 MimicGen tasks, with results for contact-rich tasks and long-horizon tasks presented in Tables 7 and 8, respectively. Please note that the success rates reported in Table 8 reflect a modified evaluation criterion in the simulation environment, resulting in slight variations from the kitchen task results presented in Table 1. The implementation details are explained below.

Details about the Baselines. We selected four task-specific baselines and reimplemented them under settings generally consistent with Hyper-GoalNet. The reimplementation details are as follows.

- **GCBC** [29, 15]: Goal-Conditioned Behavioral Cloning (GCBC) is the most general framework for learning goal-conditioned policies. It consists of a perception module, a visual encoder, and a RNN-based goal-conditioned policy module. GCBC takes in a 9-dimensional proprioceptive state, a current front-view RGB image, and a goal RGB image as input and predicts the action distribution to transfer the current state to the goal state. The model is trained end-to-end with the objective of maximizing the log-likelihood of the ground-truth action in the predicted distribution. The observation sequence length and the predicted action sequence length are both restricted to 5 steps.
- **Play-LMP** [29]: Play-Supervised Latent Motor Plans (Play-LMP) builds upon the foundation of GCBC, aiming to learn reusable plan representations and task-agnostic control from play data. Play-LMP consists of three main components: 1) Plan recognition module: maps the input sequence to a distribution in the latent plan space. 2) Plan proposal module: generates multiple conditional prior solutions based on the current and goal states. 3) Plan and goal-conditioned policy: predicts actions conditioned on the current state, goal state, and a latent plan sampled from the plan proposals. Similar to GCBC, both the observation sequence length and the predicted action sequence length are restricted to 5 steps. The model is trained end-to-end.
- **MimicPlay** [53]: MimicPlay employs a hierarchical learning framework consisting of two training stages. In the high-level training stage, the model takes the robot’s end-effector pose, along with the current visual observation and goal observation, as input to predict the future pose trajectory of the robot’s end-effector. This component is referred to as the high-level planner. In the low-level training stage, the high-level planner with the best validation performance from the previous stage is loaded and its parameters are frozen. The model then continues training using a 9-dimensional robot proprioceptive state and visual observations (both current and goal) as input to predict the robot’s actions.

Please note that in the original MimicPlay low-level training setup, in addition to the current front-view RGB image, a wrist-mounted RGB image is also used as input, which may contribute to its higher success rate. To ensure a fair comparison with our method, we modify this setup by *replacing* the wrist-mounted image with a duplicate front-view image

Table 8: **Additional evaluation on long-horizon tasks.** † indicates methods using a **sequence of visual frames** as goals rather than a single goal image, with an **extended observation length** of 10 frames versus the standard 2. ‡ indicates methods with access to **extra** wrist-mounted camera images. The wrist views, extended observation sequence, and extended goal sequences provide richer observational and task-guidance information for augmented baselines. Ours achieves similar performance compared to the baseline with access to wrist view images, extended observation sequences and a sequence of goal images, which demonstrates the effectiveness of our method in achieving long-horizon tasks with much less guidance information (effort).

Method	Coffee Prep.↑			Kitchen↑			Avg.
	d0	d1	Avg.	d0	d1	Avg.	
MimicPlay-O ^{†‡}	0.86	0.68	0.77	1.00	0.70	0.85	0.81
MimicPlay-M [†]	0.34	0.00	0.17	0.86	0.18	0.52	0.35
C-BeT	0.82	0.04	0.43	0.78	0.70	0.74	0.59
Hyper-GoalNet(G)	0.80	0.50	0.65	1.00	0.88	0.94	0.80
Hyper-GoalNet	0.80	0.50	0.65	1.00	0.80	0.90	0.78

Table 9: **Comparison with augmented BeT.** † represents **extended** context length and **additional** wrist-view images. Although BeT is not originally designed as a goal-conditioned policy method, its augmented version serves as a strong baseline. Our method still outperforms the augmented BeT.

Method	Cof. d0 ↑	Cof. d2 ↑	Mug. d1 ↑	Avg.
BeT [†]	0.66	0.42	0.26	0.45
Ours	0.94	0.62	0.46	0.67

during the low-level training process. Additionally, we adopt the same goal-specified strategy as described above, rather than providing the entire prompt video as used in MimicPlay’s original test-time evaluation setting.

Hyper-GoalNet(G) We introduce another variant of our method, Hyper-GoalNet(G), where the hypernetwork backbone takes only a single goal image as input, without requiring the current image during both training and testing phases. All other settings remain unchanged. During inference, Hyper-GoalNet(G) generates the weights for the lightweight target policy only once and maintains them fixed during rollouts, resulting in improved computational efficiency. As shown in Table 7 and Table 8, Hyper-GoalNet(G) still outperforms the baselines while utilizing a much smaller lightweight policy network, demonstrating both the effectiveness and efficiency of our approach.

Comparison with BeT [47]. We conducted comparative experiments with Behavior Transformer (BeT), a state-of-the-art approach for multi-modal behavioral learning. BeT employs k-means clustering to discretize continuous actions and utilizes transformers to model categorical distributions across action bins, incorporating an action correction head to refine discretized actions into continuous ones. Despite not being explicitly designed as a goal-conditioned policy, BeT has emerged as a robust baseline in current robot learning literature. Given that the original BeT implementation for the Franka Kitchen task was limited to state space observations and lacked compatibility with Robosuite tasks, we enhanced its architecture by incorporating a pretrained image encoder [34]. To strengthen the baseline comparison, we augmented BeT with additional wrist camera observations—a feature absent in our method—and extended the context length from 2 (used in our approach) to 4, which typically facilitates more effective policy learning for BeT. Consequently, BeT acquires more image observations per timestep than our method, while maintaining the same image resolution of 128×128 pixels.

Table 9 presents experimental results for some randomly selected tasks, where Cof. d0 and Cof. d2 represent Coffee d0 and Coffee d2 tasks, respectively, and Mug. d1 denotes the Mug cleanup d1 task. Notably, despite utilizing reduced contextual information, our method demonstrates superior performance across all tasks in terms of success rates. These results substantiate our method’s robust capability in behavior learning, even under more constrained observational conditions.

Comparison with C-BeT [9]. Conditional Behavior Transformer (C-BeT) is a goal-conditioned version of BeT, a behavior prediction model that compresses an agent’s observation history and the goal state into a compact latent representation using self-attention, which is then transformed along with discrete action representations to efficiently predict the agent’s future behaviors. To ensure fair comparison, we configured C-BeT with identical observation settings to our method, maintaining a context length of 2 and utilizing a single goal image. The comparative results are presented in Table 7 and Table 8.

Comparison with Original MimicPlay (MimicPlay-O). In the original MimicPlay experiment setting, we *retained* the wrist image as an input during the low-level training stage. For the test-time evaluation process, we provided the pretrained model with prompt videos in HDF5 format generated by MimicGen. In contrast to our approach, MimicPlay-O has access to wrist-mounted camera images, uses **an extended observation length of 10 frames** instead of 2, and utilizes **a sequence of visual frames as goals**, rather than a single goal image configuration.

Comparison with another Modified MimicPlay (MimicPlay-M). In this experimental setting, we *removed* the wrist image as an input during the low-level training stage, since the wrist image is not easy to obtain for goal specification. For the test-time evaluation process, we provided the pretrained model with prompt videos in HDF5 format generated by MimicGen. Please note that this experimental setup slightly differs from the one in our baseline setting. In contrast to our approach, MimicPlay-M uses an extended observation length of 10 frames instead of 2, and utilizes a sequence of visual frames as goals, rather than a single goal image configuration.

Efficiency Analysis. We evaluate the computational efficiency by measuring the average inference time per action step during deployment. Table 10 presents the average inference latency per step across different methods, measured over 40,000 steps on a single NVIDIA RTX 3090 GPU. Our proposed Hyper-GoalNet(G) demonstrates superior computational efficiency while maintaining state-of-the-art performance. This efficiency stems from our novel approach of dynamically generating weights for a lightweight target policy. Specifically, Hyper-GoalNet(G) generates a suitable set of policy weights at the beginning of each rollout based on the goal image. These weights remain fixed throughout the execution, eliminating the need for repeated weight generation and thus significantly reducing the computational overhead during deployment.

Table 10: Average Inference Time Per Step

Method	GCBC	Play-LMP	C-BeT	Hyper-GoalNet	Hyper-GoalNet(G)
Time (ms)	15.47	22.78	13.61	6.33	1.46

C Comparison with Other Hypernetworks

This section provides additional details on the comparison between our proposed hypernetwork architecture and HyperZero, particularly regarding computational requirements.

C.1 Training Time & Memory Requirements

To ensure a fair and direct comparison, all experiments were conducted on a single NVIDIA RTX 4090 GPU. The training hyperparameters, including batch size, learning rate, and optimizer settings, were kept identical for both our method and the HyperZero baseline. The only modification was the hypernetwork architecture itself. This controlled setup ensures that any observed differences in resource consumption are directly attributable to the design of the hypernetwork module.

As detailed in Table 11, our method requires slightly more resources than HyperZero in terms of per-epoch training time and memory usage. However, this modest computational overhead is coupled with the substantial performance gains documented in the main paper, highlighting the efficiency and effectiveness of our architectural design.

Table 11: **Computational resource comparison.** The table shows per-epoch training time and memory footprint for our method versus HyperZero under identical hyperparameter settings. “Frozen” and “Unfrozen” refer to the state of the visual encoder.

Metric	HyperZero	Ours
Training Time / Epoch	~90s	~104s
Memory (Frozen Encoder)	3,038 MB	4,916 MB
Memory (Unfrozen Encoder)	13,452 MB	14,844 MB

D Policy Learning Details

Overview. A conventional sequential decision-making problem can be formalized as a discrete-time finite Markov decision process (MDP) defined by a 7-tuple $M = (\mathcal{O}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, H)$, where:

- $\mathcal{O}$ denotes the observation space,
- $\mathcal{A}$ represents the action space,
- $\mathcal{P} : \mathcal{O} \times \mathcal{A} \times \mathcal{O} \rightarrow \mathbb{R}_+$ defines the transition probability distribution,
- $\gamma \in [0, 1]$ is the discount factor,
- H specifies the temporal horizon of the process.

In the context of imitation learning, we define a complete state-action trajectory as $\tau = (o_0, a_0, \dots, o_t, a_t)$, where the initial state is sampled as $o_0 \sim \rho_0(o_0)$, actions are generated by the policy $a_t \sim \pi_\theta(\cdot|o_t)$, and state transitions follow $o_{t+1} \sim \mathcal{P}(\cdot|o_t, a_t)$.

Traditionally, the objective in goal-conditioned decision-making problems is to identify an optimal goal-conditioned policy π_θ that maximizes the expected discounted reward:

$$\eta(\pi_\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^T \gamma^t r(o_t, a_t, o_{t+1}|o_g) \right] \quad (11)$$

However, our approach diverges from this conventional framework in several key aspects: *Reward-Free Learning:* Operating within a behavior cloning paradigm, we lack access to explicit reward signals. Instead, we aim to learn a goal-specific policy π_θ that maps states to optimal actions purely from demonstrations. *Goal-Specific Policy Generation:* Rather than learning a universal goal-conditioned policy, our hypernetwork architecture generates specialized policies for specific goals, conditioned on the current RGB observation and a target goal image. *Non-Markovian Extension:* We relax the Markovian assumption to incorporate temporal dependencies. The resulting policy formulation becomes:

$$\pi_\theta(a_t|o_{t-1}, o_t) \quad (12)$$

This extended formulation enables the policy to leverage information from a context window of length 2, enhancing its capacity to handle complex, temporally-dependent manipulation sequences.

Model Architecture. Our architectural design addresses the challenges of visuomotor manipulation tasks, which require processing of high-dimensional visual inputs rather than simple state-based representations. The architecture comprises several key components integrated to handle visual and proprioceptive information effectively. *Visual Processing Pipeline:* To bridge the gap between high-dimensional visual inputs and hypernetwork processing capabilities, we employ a pre-trained visual encoder [34] to compress RGB images into compact latent representations. This encoder’s training follows a two-phase strategy: *Initial phase (first 20 epochs):* Parameters remain frozen to establish stable feature representations; *Fine-tuning phase:* Parameters become trainable to optimize task-specific visual features.

Hypernetwork Configuration. Our hypernetwork uses the HyPoGen architecture [40] with 8 optimization blocks. It processes encoded current and goal images to generate the parameters for a 3-layer MLP target policy, an architecture that can be flexibly extended in depth and width. This

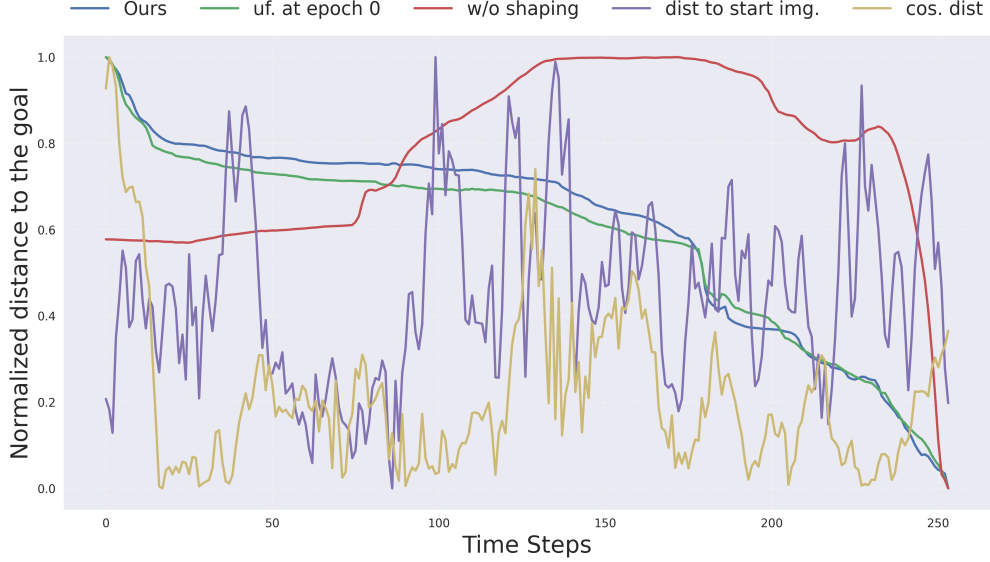


Figure 5: Normalized distance between current and goal states computed with **different latent spaces** along policy rollouts.

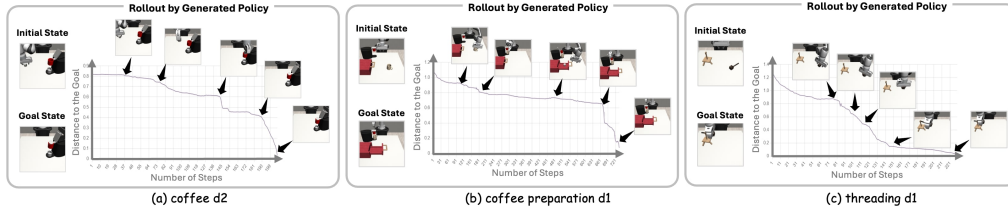


Figure 6: **Distance to the goal in the latent space along policy rollouts across different manipulation tasks.** The consistent decreasing trend across diverse tasks demonstrates that our learned latent representations effectively capture the physical progress toward task goals, establishing meaningful correspondence between latent-space distances and real-world task completion.

meta-learning approach enables dynamic policy adaptation based on specified goals while maintaining computational efficiency. Beside the image encoder mentioned above, the action generation incorporates several specialized components: *Predictive Model*: Implemented as an MLP operating in the compressed latent space, leveraging the reduced dimensionality for efficient dynamics modeling, *Proprioceptive Encoder*: A compact MLP processes low-dimensional proprioceptive states, providing essential agent state information, *Feature Integration*: Temporal image features are concatenated with proprioceptive information at each timestep, *Target Policy*: A lightweight MLP processes the integrated features to generate appropriate control actions. This architecture efficiently handles the complexity of visuomotor tasks while maintaining computational tractability through dimensionality reduction and feature integration.

Training Details. Our framework implements an end-to-end training paradigm with controlled experimental conditions for reproducibility. Using a fixed random seed, we partition the dataset into 950 training and 50 validation demonstrations across all tasks. Training employs a batch size of 256 and the Adam optimizer with an initial learning rate of 5×10^{-4} , coupled with cosine annealing for learning rate decay. The model trains for 500 epochs without weight decay or dropout regularization in the hypernetwork component. The training and evaluation procedures were performed on a single NVIDIA GeForce RTX 3090 or RTX 4090 GPU. To ensure a fair comparison, all methods evaluated in our experiments were trained using this identical configuration.



Figure 7: The real robot workspace.

E Ablation Studies

We conducted ablation experiments to evaluate the impact of various methodological choices, with quantitative results presented in Table 3. Figure 6 and Figure 5 illustrates the normalized distance between current and goal states under different experimental configurations. Our analysis includes several key variations:

- *uf.* at epoch 0. Full parameter unfreezing from epoch 0, where all model components are trainable from initialization.
- *w/o shaping*. Removal of latent shaping technique detailed in Section 3.2.
- *dist $\leftrightarrow$ start img.* Alternative distance computation between current and start images, rather than current and goal images.
- *cos. dist.* Implementation of cosine distance metric in place of Euclidean distance.
- *C-Bet(w/ shaping)*. We implement C-Bet and incorporate our proposed additional shaping method while maintaining the same experimental setup.

These systematic variations enable us to quantify the contribution of each design choice to the overall system performance.

F Real Robot Experiment Setting

Real Robot Platform. All real-world experiments were conducted on a RealMan RMC-DA dual-arm manipulator, which comprises two 7-degree-of-freedom robotic arms, each rated for a 5 kg payload and fitted with a parallel-jaw gripper. While the platform supports bimanual manipulation, this work focuses specifically on evaluating Hyper-GoalNet’s capabilities in single-arm tabletop manipulation tasks. The extension to bimanual manipulation remains as future work. The robot is mounted in front of a 1.25 m $\times$ 0.75 m tabletop, which serves as the exclusive workspace for all manipulation tasks. A standardized set of test objects—ranging from simple geometric primitives (e.g., cubes) to more complex shapes—is placed on the table according to predefined configurations. And on the tabletop there might be some distractor object. To support perception, we employ an overhead RGB-D sensor (Intel RealSense D435i). The entire workspace is shown in Figure 7

Task Details. To validate the versatility and robustness of our method across a broad spectrum of manipulation skills, we selected four representative tabletop tasks. Each task emphasizes a different

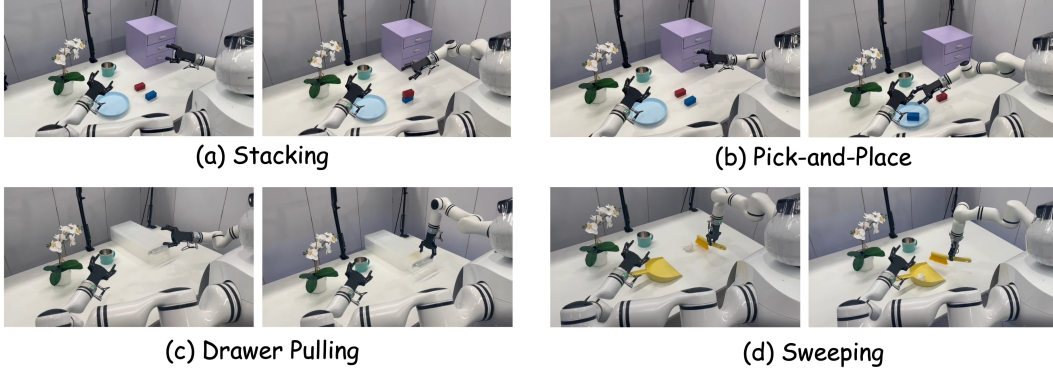


Figure 8: Key phases of diverse manipulation tasks in real robot experimental evaluation.

core competency—object localization, precision grasping, and surface contact manipulation—and is defined as follows:

- **Pick-and-Place.** The robot must perceive and localize a specified cubic object within the workspace, plan a collision-free trajectory, execute a stable grasp, and transport the object to a predefined target location (e.g., a plate). Success is measured by the accuracy of the final placement and the repeatability across trials.
- **Stacking.** Extending the pick-and-place paradigm, this task requires the robot to grasp a source cube, position it directly above a target cube resting on the tabletop, lower it until gentle contact is detected via proximity or vision-based cues, and then release to complete the stack. Success is defined by the source cube being neatly aligned atop the target cube.
- **Drawer Pulling.** The robot must detect drawer handle, plan an approach to engage the handle with its gripper, and execute a controlled pulling maneuver to extend the drawer along its linear guide. Performance is evaluated by the final extension distance achieved without stalling.
- **Sweeping.** The end-effector is equipped with a broom attachment. The robot must locate and gather a target object scattered on the tabletop, then sweep it into a designated collection zone (e.g., a dustpan). Success is defined by the target object being fully contained within the collection zone at the end of each trial.

Data Processing and Observation Space. Our real-robot evaluation follows the same protocol as in simulation: at each timestep, the policy receives the two most recent observations and a single goal image. We collect approximately 70–100 human teleoperation trajectories per task for training. Vision is acquired with an Intel RealSense D435i depth camera; we concatenate its depth channel with the RGB channels to form 4-channel images of resolution 128×128 , which are normalized to $[0, 1]$ before input to the encoder. Since explicit end-effector poses are unavailable, we represent the current proprioceptive state by the previous action—comprising seven joint angles and one gripper command—resulting in an 8-dimensional vector. All modalities are synchronized identically to the simulation setting, ensuring a seamless transfer between simulated and real-world experiments.

G Additional Visualization

We provide additional visualization of the shaping. Figure 9 and Figure 10 demonstrate the approximate monotonic trend of the distance between the current state and the goal state is consistent across different task scenarios. This consistent pattern substantiates the robustness of our shaping mechanism and validates its task-agnostic applicability.

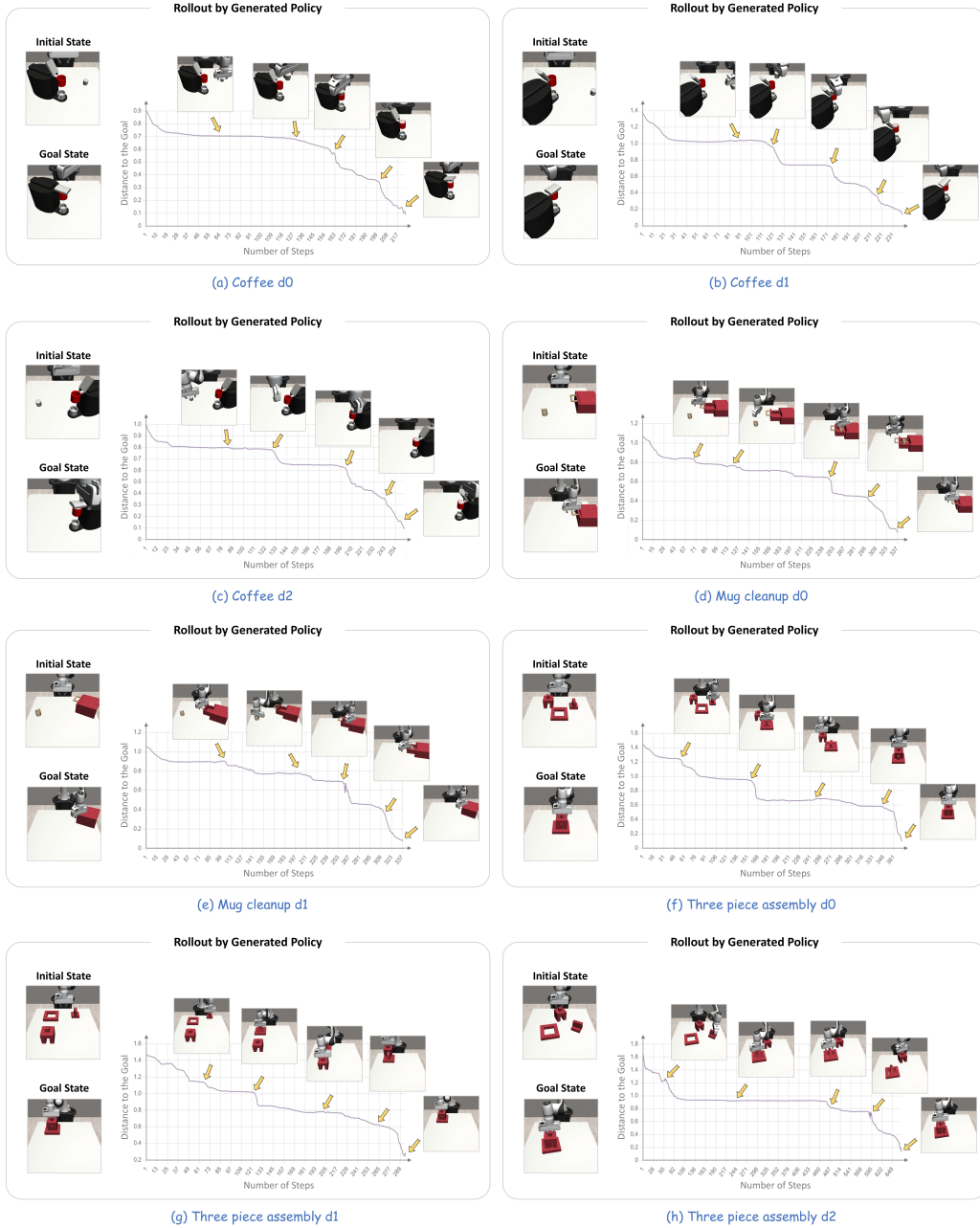


Figure 9: Additional visualization: Distance to the goal in the latent space along policy rollouts across different manipulation tasks. The consistent decreasing trend across diverse tasks demonstrates that our learned latent representations effectively capture the physical progress toward task goals, establishing meaningful correspondence between latent-space distances and real-world task completion.

H Future Work

Building upon our current findings, we identify three primary directions for future research: First, we aim to incorporate multi-goal reasoning to handle complex sequential tasks. Second, we plan to extend our framework by integrating foundation models to develop a more generalized policy generation framework, potentially enabling broader task generalization and enhanced adaptability.

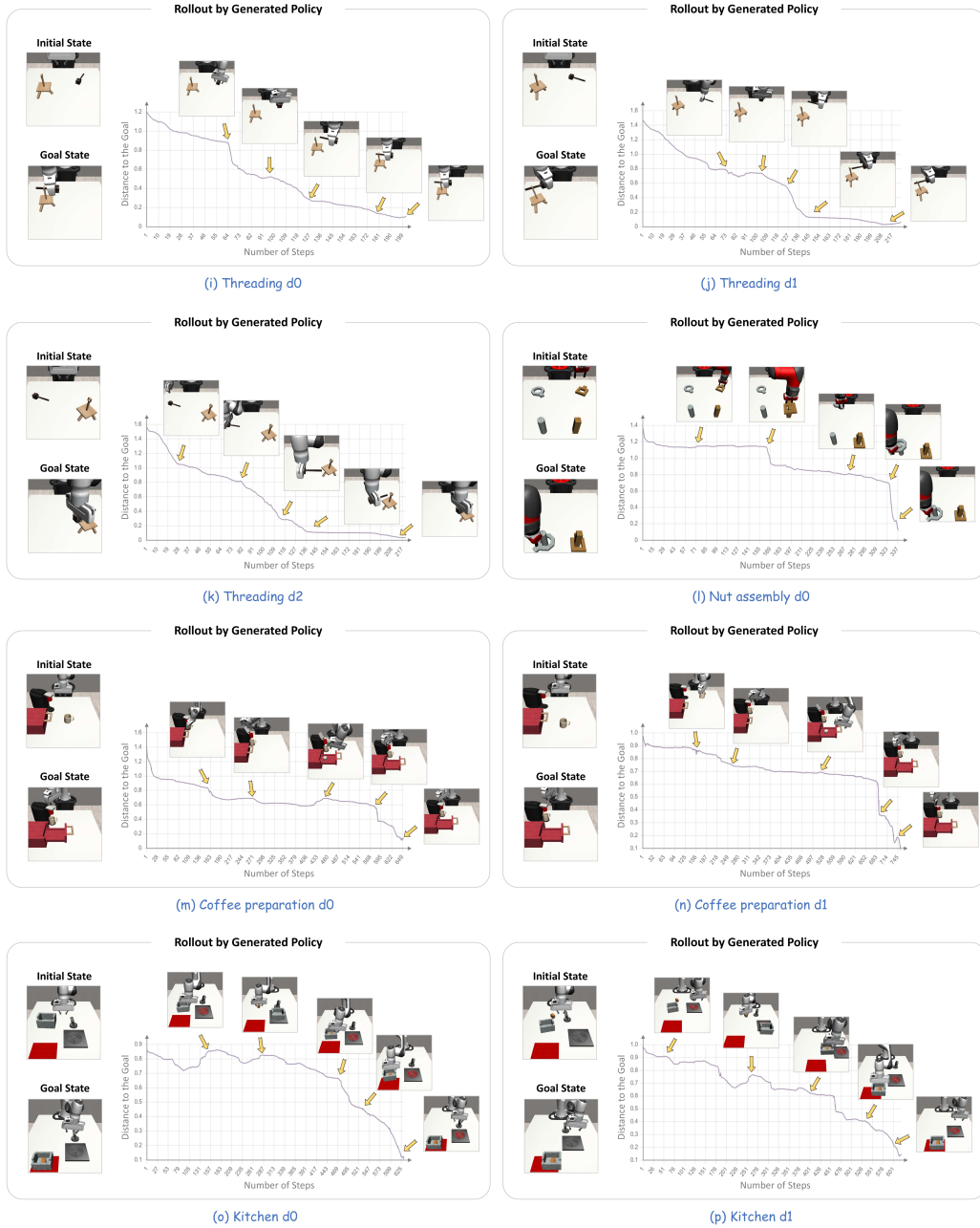


Figure 10: **Additional visualization: Distance to the goal in the latent space along policy rollouts across different manipulation tasks.** The consistent decreasing trend across diverse tasks demonstrates that our learned latent representations effectively capture the physical progress toward task goals, establishing meaningful correspondence between latent-space distances and real-world task completion.

across diverse manipulation scenarios. Third, we plan to combine our parameter-adaptive approach with reinforcement learning to reduce dependence on demonstration data.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [NA].
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes]" is generally preferable to "[No]", it is perfectly acceptable to answer "[No]" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No]" or "[NA]" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist",**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers.**

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: We claim our contribution in the introduction and method section.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We discuss limitations including challenges in long-term planning, further work on scaling up, and multi-goal reasoning design.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper does not does not involve the theoretical part and the proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We include implementation details in the experiment section and training details in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We don't provide open access to the code now, but the data is publicly available. We will open the source code upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: We show the training and testing details, as well as model architecture in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[No\]](#)

Justification: We do not show error bars in paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: We mention the compute resources in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: There is no societal impact of the work performed.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: In our manipulation tasks, we have not encountered any safeguard issues.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: All the methods, datasets and models originated from other works are cited in reference.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[NA\]](#)

Justification: We will release relative assets upon acceptance. We welcome new creation based on our system.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The development of our core methods does not incorporate LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.