Hierarchical Reinforcement Learning with Optimal Level Synchronization Based on Flow-Based Deep Generative Model

JAEYOON KIM*, JUNYU XUAN, JIE LIANG, and FAROOKH HUSSAIN, University of Technology Sydney, Australia

High-dimensional state and action spaces com- bined with sparse reward structures in reinforcement learning (RL) environments typically require advanced control architec- tures. Hierarchical Reinforcement Learning (HRL) demonstrates superior performance compared to atomic RL approaches in these challenging scenarios. HRL can manage the complexity of commands to achieve task objectives through its hierarchical structure. One of the key challenges in HRL is efficiently training each level's policy with optimal data collection from its experience. Off-policy correction is a critical technique for facilitating sample-efficient off-policy training in HRL, as it addresses the non-stationary issue of higher-level policy training. However, existing methods typically employ indirect probabilistic approaches that fail to accurately capture the current capability of the lower-level policy. This mismatch ultimately constrains the effectiveness of higher-level policy training. In this paper, we propose a novel HRL model that supports direct off-policy correction based on a Flow-based Deep Generative Model (FDGM). This approach leverages the inverse operation of FDGM to achieve goals aligned with the current knowledge of the lower-level policy. Additionally, our model addresses the limitations of FDGM to enable its effective use in HRL. Through comparative experiments on benchmark environments, our model demonstrates superior performance over existing models.

JAIR Associate Editor: Pablo Samuel Castro

JAIR Reference Format:
JaeYoon Kim, Junyu Xuan, Jie Liang, and Farookh Hussain. 2026. Hierarchical Reinforcement Learning with Optimal Level Synchronization Based on Flow-Based Deep Generative Model. *Journal of Artificial Intelligence Research* 86, Article 28 (July 2026), 26 pages. DOI: 10.1613/jair.1.19264

1 Introduction

Among many reinforcement learning (RL) algorithms, Hierarchical Reinforcement Learning (HRL) stands out by addressing the limitations of atomic RL algorithms such as Deep Q-Network (DQN) (Mnih, Kavukcuoglu, et al. 2013), Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2015), Soft Actor-Critic (SAC) (Haarnoja, Zhou, et al. 2018), and Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al. 2018). HRL demonstrates a strong capability to efficiently seek optimal solutions in high-dimensional state and action spaces, as well as sparse reward environments. This efficiency is achieved through temporal abstraction goals between the policies of HRL. Since HRL is composed of hierarchical policies, effective communication between higher-level and lower-level policies is crucial for its performance.

In a two-level hierarchical structure, the higher-level policy focuses on abstract and generalized features, while the lower-level policy handles primitive actions in interaction with the environment. For instance, in the MuJoCo Ant environment, the hierarchical policy operates through coordinated temporal abstraction: (1) the higher-level

*Corresponding Author.

Authors' Contact Information: JaeYoon Kim, ORCID: 0000-0003-0898-5097, JaeYoon.Kim@student.uts.edu.au; Junyu Xuan, ORCID: 0000-0002-8367-6908, Junyu.Xuan@uts.edu.au; Jie Liang, ORCID: 0000-0001-7179-5208, Jie.Liang@uts.edu.au; Farookh Hussain, ORCID: 0000-0003-1513-8072, Farookh.Hussain@uts.edu.au. University of Technology Sydney, Sydney, New South Wales, Australia.

DOI: 10.1613/jair.1.19264

policy generates macro-level movement directions visualized as arrows in Fig. 1, which serve as subgoals, while (2) the lower-level policy executes corresponding primitive actions at each timestep to achieve these directional objectives.

In goal-conditioned HRL, a policy is conditioned on an additional input called a goal, which is provided by its higher-level policy. The communication between each HRL policy is crucial, and there are several approaches to managing this in an HRL algorithm. For example, an HRL algorithm can control a goal using an atomic RL algorithm (Haarnoja, Hartikainen, et al. 2018; J. Li et al. 2022; Nachum et al. 2018a; Schmidt et al. 2024) or various machine learning algorithms (Osa et al. 2019; Rafati and Noelle 2019; Co-Reyes et al. 2018).

Meanwhile, the goal is also used for training its own policy. One of the key challenges in hierarchical RL, which is typically trained using a bottom-up layer-wise approach, is how to train each policy with optimal data collection for the task. Fig. Off-policy correction for HRL, first introduced in HIRO (Nachum et al. 2018a), provides an effective solution to the optimal training challenge in HRL. As illustrated in Fig. 2, HIRO's architecture implements this through a two-level policy framework with periodic goal correction.

HIRO combines off-policy training and correction to address the challenge of training a higher-level policy from the perspective of the current lower-level policy. Off-policy methods have been highlighted for their ability to overcome the local minima issues associated with on-policy methods. However, off-policy training also faces a non-stationary issue when training the higher-level policy of HRL due to its inherent algorithmic characteristics.

The correction component requires careful consideration and examination. HIRO has attempted to address this issue through various indirect estimation methods, proposing an off-policy correction technique to mitigate the challenges of off-policy learning. This is achieved by relabeling past experiences, specifically the actions of the higher-level policy, as goals.

However, the estimation process, which relies on an indirect probabilistic approach, has proven to be insufficient or imprecise due to its inherent limitations. The method's inability to accurately reflect the current knowledge of the lower-level policy stems from its reliance on indirect estimation.

Our research emphasizes the importance of direct estimation, which accurately reflects the current knowledge of the lower-level policy. This direct approach enhances the effectiveness of training the higher-level policy. In this study, we propose a novel HRL model with a direct off-policy correction strategy. This strategy leverages the current knowledge of the lower-level policy using a Flow-Based Deep Generative Model (FDGM), eliminating the need for superficial data derived from indirect estimations.

To reflect the current knowledge of the lower-level policy in the off-policy correction after training, we utilize the current internal parameters of its neural network (NN) to determine an exact goal for training the higher-level policy. By leveraging the inverse of the policy, we can directly compute the goal, effectively relabeling a previous goal of the higher-level policy.

When the result of an inverse operation of the lower-level policy is evaluated as a goal for the higher-level policy, reflecting the current knowledge of the lower-level policy, the estimation using the inverse model of the policy can serve as an accurate and effective off-policy correction method for HRL. In this context, determining the inverse value of a policy in a model-free HRL is defined as the inverse model of the policy.

The FDGM, capable of supporting the exact inverse value of a policy, is a promising candidate for defining an inverse model in HRL (Albergo and Vanden-Eijnden 2022; Kobyzev et al. 2020; Y. Li et al. 2025; Lipman et al. 2022; Liu 2022; Xu et al. 2022). However, FDGM has inherent limitations, including a restriction between input and output dimensions, as well as a chronic issue—biased log-density estimation (Erdogan 2025; C. Li et al. 2025). The former limitation makes it challenging to define an inverse model that uses a general method to support the flexible choice of goal dimensions. Meanwhile, ongoing research is addressing the latter issue in FDGM (Du et al. 2025; Liao et al. 2025; Wang et al. 2025). Our research proposes an architecture inspired by (Ma et al. 2020), which is shown in Fig. 2, to leverage the advantages of FDGM in HRL while mitigating its inherent limitations.

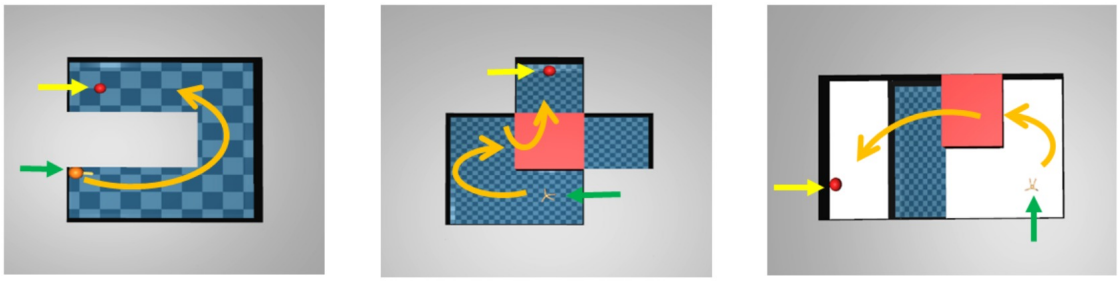


Fig. 1. An example of several tasks in the Ant environment used in this research is shown above. The target location indicated by the yellow arrow serves as the reward point for the agent's approach. The agent, represented by the green arrow, must navigate to this target. The trained policy requires the agent to: (1) combine multiple action sequences while (2) interacting with a pink obstacle block in two out of the three above tasks. These tasks include: a series of directional movements (third from the right), clearing away an obstacle (second from the right), and constructing a bridge using an object (rightmost task). The higher-level policy guides its lower-level policy by providing abstract actions represented by the orange arrow, which specify the desired positions and orientation of the ant and its limbs to achieve the reward.

Our model is compared to HIRO in terms of the methodology for off-policy correction and the issue of biased log-density estimation in FDGM. Additionally, our model is compared to Latent Space Policies (LSP) (Haarnoja, Hartikainen, et al. 2018), which uses FDGM for HRL shown in Fig. 2, regarding the issue of the invertible bijective transformation function in FDGM mentioned above. A comparative analysis between our proposed model and the reference models is presented in Table 2.

In utilizing optimal off-policy correction to synchronize each level of HRL, several key questions arise. How can off-policy correction in HRL be effectively implemented using the current internal knowledge of a newly trained lower-level policy to optimize the training of its higher-level policy? Additionally, how can the limitations of FDGM be addressed when considering it for a lower-level policy, particularly when leveraging its inverse operation to relabel goals for off-policy correction?

Our experimental results on benchmark MuJoCo environments demonstrate that our approach outperforms HIRO and LSP in terms of both speed and performance accuracy. In summary, the contributions of this research are as follows:

- We propose a novel hierarchical RL model architecture that defines an inverse model for model-free HRL.
- Our model adopts a direct method based on FDGM to identify an exact goal that reflects the current knowledge of a recently updated lower-level policy.
- Despite utilizing FDGM in the lower-level policy, our model effectively addresses the limitations associated with FDGM in HRL.

The remainder of this paper is organized as follows. Section 2 introduces the preliminaries of HRL and off-policy correction. Section 3 analyzes related research. Section 4 describes our proposed HRL model using FDGM. Section 5 presents the experimental results comparing the performance of our HRL model with HIRO and LSP. Section 6 discusses the insights gained from the experiments. Finally, Section 7 concludes this paper and outlines future work.

2 Preliminary Knowledge

2.1 Hierarchical Reinforcement Learning

The structure of HRL is composed of several unit agents stacked hierarchically. A policy in goal-conditioned HRL typically has two inputs: the state of the environment s_t and a goal g_t, which is a temporal abstraction

provided by its higher-level policy μ_{hi}. A unit agent is usually constructed based on an atomic RL framework, as it functions as a policy within a level of HRL. The lowest-level policy interacts with the environment by generating an action $a_t \sim \mu_{lo}(s_t, g_t)$ based on the goal g_t from its higher-level policy, where $g_t \sim \mu_{hi}$ is updated every c time steps (c is the horizon of μ_{hi}) or determined by a fixed goal transition function $h(s_{t-1}, g_{t-1}, s_t)$. Due to the hierarchical structure, the training period c of a higher-level policy μ_{hi} is longer than that of its lower-level policy μ_{lo}. The higher a policy is positioned in the HRL hierarchy, the more abstract its role becomes.

The environment responds to the action a_t of the lowest-level policy with a reward R_t and a next state s_{t+1}, sampled from the reward function $R(s_t, a_t)$ and the transition function $f(s_t, a_t)$, respectively. The intrinsic reward $r_t = r(s_t, g_t, a_t, s_{t+1})$, where r is a fixed parameterized reward function for the lower-level policy in HRL, is typically provided by the higher-level policy. Notably, the highest-level policy uses the environmental reward R_t instead of an intrinsic reward.

Specifically, in Fig. 1, the higher-level policy devises an abstract strategy (represented by the orange arrow), while the lower-level policy executes physical actions based on the goal from its higher-level policy and the current state. The model architectures, such as the one depicted in Fig. 2, illustrate the outlined structure of HRL. The key notations used in this research are summarized in Table 1.

2.2 Off-Policy Correction

It is crucial to devise an optimal training solution to enhance the performance of HRL. Off-policy methods in HRL have been proposed to address the local minimum issue associated with on-policy methods. In off-policy policy gradient methods, experience replay involves storing sampled trajectory data from past episodes in a replay buffer, which significantly improves sample efficiency. This approach enables better exploration by collecting samples using a behavior policy that differs from the target policy. The objective function of the off-policy policy gradient is defined as follows:

$$J(\theta) = \mathbb{E}_{S\sim d^{\beta}}\left[\sum_{a\in A} Q^{\pi}(s,a)\pi^{\theta}(a|s)\right] \tag{1}$$

where $d^{\beta}(s)$ is the stationary distribution of the behaviour policy $\beta(a|s)$, $\pi^{\theta}(a|s)$ is the target policy and $Q^{\pi}(s,a)$ is the action-value function estimated with respect to the target policy $\pi^{\theta}(a|s)$ (Degris et al. 2012).

However, the off-policy method in goal-conditioned HRL faces a non-stationary issue due to the bottom-up training approach in off-policy RL. Previously used outputs, such as goals from the higher-level policy, may no longer be suitable for training, as they do not reflect the updated parameter values θ of the newly trained lower-level policy. Consider the operation of a lower-level policy μ_{lo} in HRL as follows:

$$a_t \sim \mu_{lo}^{\theta}(s_t, g_t), \text{ at time } t. \tag{2}$$

After μ_{lo}^{θ} is trained for c steps,

$$\begin{gathered} a_{t+c} \sim \mu_{lo}^{\theta'}(s_{t+c}, g_t), \text{ assume } s_{t+c} = s_t \\ a_t \neq a_{t+c} \text{ or } a_t = a_{t+c} \end{gathered} \tag{3}$$

where a_{t+c} and s_{t+c} are the action of $\mu_{lo}^{\theta'}$ and the state of the environment after c steps, respectively. The action a_{t+c} may differ from a_t due to changes in the parameter values of μ_{lo}^{θ} and $\mu_{lo}^{\theta'}$.

To address this issue, an off-policy correction is proposed in HIRO (Nachum et al. 2018a) through an indirect estimation strategy.

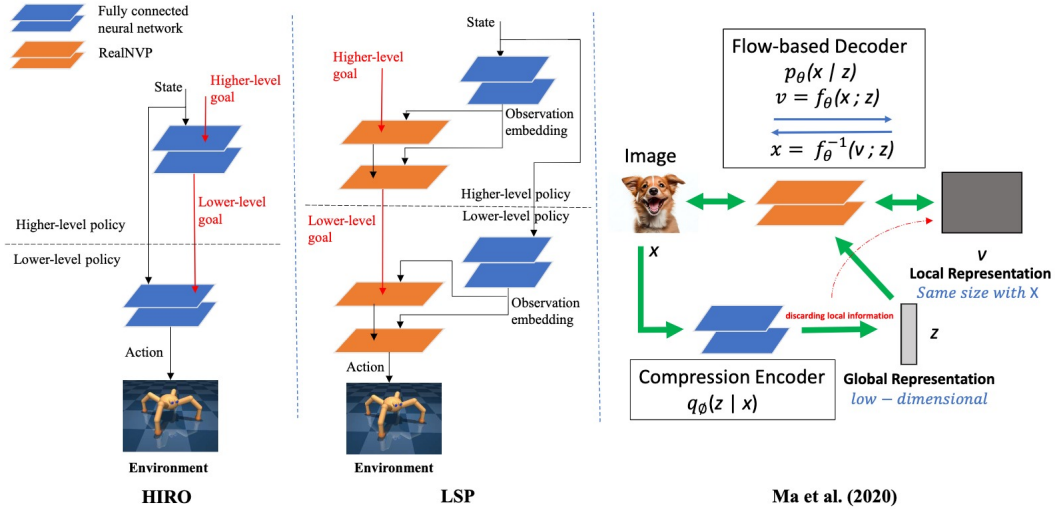


Fig. 2. The model architecture of HIRO (left), LSP (center) and Ma et al. (right)

3 Related Work

3.1 Hierarchical RL

A large body of research has focused on developing optimal methods to connect adjacent level policies in the HRL framework from an action-oriented perspective. In the option-critic architecture, the policy over options selects an option to follow its intra-policy, which continues to operate until the condition for option termination is met (Bacon et al. 2017). The Strategic Attentive Writer, with its HRL structure, offers advantages in sequential decision-making domains due to its use of macro-actions, which are partially organized based on environmental information (Mnih, Agapiou, et al. 2016). FeUdal networks for HRL identify semantically meaningful sub-goals in an environment, which are then pursued by different policies of the manager (Vezhnevets et al. 2017).

(Rafati and Noelle 2019) proposes an efficient and general method for discovering sub-goals using two unsupervised learning techniques: anomaly outlier detection and K-means clustering. The Hierarchical Actor-Critic incorporates DDPG with an actor-critic network and Hindsight Experience Replay (HER) (Andrychowicz et al. 2017) at every policy level (Levy, Platt, et al. 2017). (Levy, Konidaris, et al. 2017) introduces a nested and goal-conditioned HRL framework capable of dividing a task into several sub-tasks using two classes of hindsight transitions. (Nachum et al. 2018b) develops a method to leverage representation learning through sub-optimality, defined as the difference between the value function of an optimal HRL that does not utilize representation learning and the value function of an HRL that does utilize representation learning.

However, the aforementioned studies do not address the optimal training for level synchronization between adjacent level policies. The main focus of HIRO is to develop an effective training method for each HRL policy, addressing the dynamics of training between adjacent level policies to ensure their synchronization. To identify an optimal goal $g_t \in R^{d_s}$ for off-policy correction in HRL, HIRO generates 10 candidate goals. These include eight candidates sampled randomly from a Gaussian distribution centered at the state difference $s_{t+c} - s_t$ (where 'c' is the horizon of the higher-level policy), the original goal g_t, and a goal equivalent to the state difference $s_{t+c} - s_t$.

These candidates are evaluated based on the induced log probability of the lower-level policy:

$$\begin{aligned} &\log \mu_{lo}(a_{t:t+c-1}|s_{t:t+c-1}, \tilde{g}_{t:t+c-1}) \\ &\propto -\frac{1}{2}\sum_{i=t}^{t+c-1} \|a_i - \mu_{lo}(s_i, \tilde{g}_i)\|_2^2 + \text{const} \end{aligned} \tag{4}$$

where μ_{lo} is the lower-level policy, $x_{t:t+c-1}$ represents the sequence $x_t, ..., x_{t:t+c-1}$ collected by the higher-level policy, a_t is the action of μ_{lo} and $\tilde{g}_t$ is the relabeled goal g_t.

After identifying the best candidate, this goal is ultimately used to train the higher-level policy. The research also explores three additional methods, detailed in its Appendix. However, it does not propose a suitable method for off-policy correction in model-free HRL, as it relies on indirect estimation. Our research extends the work initiated by HIRO on off-policy correction, which is a central focus of our study.

LSP proposes that each policy in HRL can be trained in a bottom-up, layerwise manner using a latent variable incorporated into the maximum entropy objective. To ensure that higher layers retain full expressivity, the latent variable and action must be invertible. This is achieved by leveraging a FDGM. In Fig. 2, LSP demonstrates how a latent goal from the higher-level policy and an observation from the environment are provided to the lower-level policy. Two invertible coupling layers (orange), representing the FDGM in LSP, receive the latent goal from the higher-level policy and condition it on an observation embedding. The observation embedding, implemented with two fully connected layers (green), adjusts the dimension of the observation. However, this approach is not suitable for off-policy correction in HRL, as it cannot support a flexible goal space due to the intrinsic limitations of FDGM, particularly the constraints of its invertible bijective transformation function.

For our research, we adopt the use of FDGM in HRL, similar to LSP, while addressing its limitations.

3.2 Deep Generative Models

We have explored various deep generative models for use as an inverse model in HRL. After careful consideration, we concluded that employing a FDGM in our model ensures its validity as an inverse model in HRL.

GAN and VAE (Goodfellow et al. 2014) introduces an innovative deep generative model, Generative Adversarial Networks (GAN), which consists of two models: a discriminator and a generator. A Variational Autoencoder (VAE) (Kingma and Welling 2013) employs a probabilistic encoder to generate a latent variable for a decoder, addressing the limitations of the vanilla Autoencoder (Hinton and Salakhutdinov 2006). Numerous variants of these two generative models have been developed due to their robust unsupervised learning frameworks (Kolouri et al. 2018; Mao et al. 2017; Odena et al. 2017; Radford et al. 2015; Rainforth et al. 2018; Su and Wu 2018; Tolstikhin et al. 2017). However, GAN suffers from a significant drawback: training divergence caused by the difficulty in finding a Nash equilibrium (Salimans et al. 2016). While VAE allows precise control over the latent variable, it often produces blurry outputs due to imperfect reconstruction, making it challenging to train the decoder effectively. Given these inherent disadvantages of GAN and VAE, we have opted to avoid their use in our research.

Normalizing Flow A simple distribution can be transformed into a complex distribution by repeatedly applying an invertible mapping function. The change of variable theorem enables this transformation from one variable to another, resulting in the final distribution of the target variable as follows.

Suppose a probability density function $z \sim p(z)$ describes a random variable z. If there exists an invertible bijective transformation function f between a new variable x and z, such that $x = f(z)$ and $z = f^{-1}(x)$, then the transformation is well-defined. Extending this to the multivariate case, if the change of variable theorem is applied to $\mathbf{z}$ and a final target variable $\mathbf{x}$ through successive distributions $p(\mathbf{z})$ and successive transformation function $f(\mathbf{z})$, we have:

$$\mathbf{z}_{i-1} \sim \ p_{i-1}(\mathbf{z}_{i-1}),\ \mathbf{z}_i = f_i(\mathbf{z}_{i-1}), \text{thus } \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i). \tag{5}$$

Then

$$\begin{aligned} p_i(\mathbf{z}_i) =& p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| det \frac{df_i^{-1}}{d\mathbf{z}_i} \right| \\ =& p_{i-1}(\mathbf{z}_{i-1}) \left| det \frac{df_i}{d\mathbf{z}_{i-1}} \right|^{-1}, \text{thus} \end{aligned} \tag{6}$$

$$\log p_i(\mathbf{z}_i) = \log p_{i-1}(\mathbf{z}_{i-1}) - \log \left| det \frac{df_i}{d\mathbf{z}_{i-1}} \right| \tag{7}$$

where $det \frac{\partial f_i}{\partial \mathbf{z}_i}$ is the Jacobian determinant of the function f.

Finally, the chain of K transformations of probability density function f_i—each of which is easily invertible and has a Jacobian determinant that can be efficiently computed—transforms the initial distribution $\mathbf{z}_0$ into the final target variable $\mathbf{x}$. This process is described as follows:

$$\begin{aligned} \mathbf{x} = \mathbf{z}_K &= f_K \circ \circ \circ f_2 \circ f_1(\mathbf{z}_0) \\ \log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left| det \frac{df_i}{d\mathbf{z}_{i-1}} \right| \end{aligned} \tag{8}$$

The flow arises from the path traced by the random variables $\mathbf{z}_i = f_i(\mathbf{z}_{i-1})$. A normalizing flow is defined as the full chain of transformations configured by the successive distributions π_i.

In our research, we focus on the advantages of normalizing flows, particularly their model flexibility and generation speed, despite their inherent drawbacks. The Real-valued Non-Volume Preserving algorithm (RealNVP) leverages normalizing flows, implemented using an invertible bijective transformation function. Each bijection, referred to as an affine coupling layer and denoted as $f : \mathbf{x} \mapsto \mathbf{y}$, decomposes the input dimension into two sections. The intrinsic transformation property of the affine coupling layer ensures that the input dimension remains unchanged, while alternately modifying the two split input sections in each coupling layer. This property enables straightforward inverse operations. Additionally, the Jacobian determinant is easily computed because the Jacobian matrix is lower triangular.

RealNVP employs a multi-scale architecture and batch normalization to enhance performance. To capture the local correlation structure of images, it utilizes two masked convolution methods: the spatial checkerboard pattern mask and the channel-wise mask (Dinh, Sohl-Dickstein, et al. 2016). Non-linear Independent Components Estimation (NICE), a predecessor to RealNVP, uses an additive coupling layer that omits the scale term present in the affine coupling layer (Dinh, Krueger, et al. 2014). Generative Flow with 1×1 convolutions simplifies the architecture by addressing the reverse operation of channel ordering in NICE and RealNVP (Kingma and Dhariwal 2018).

Several studies have attempted to address the chronic drawback of normalizing flows: biased log-density estimation (Chen et al. 2019; Ho et al. 2019). In Fig. 2, (Ma et al. 2020) leverages the intrinsic characteristics of FDGM with an inductive bias based on its model architecture. By doing so, the research effectively utilizes the biased log-density estimation inherent to FDGM. The proposed model architecture combines a VAE, which extracts a global representation of an image, with a FDGM that relies on a local representation. The FDGM is conditioned on the global representation of the image provided by the VAE. As a result, an unbiased log-density estimation of the image can be achieved through the FDGM using the output of the VAE.

The compression encoder $q_\phi(z|x)$ in the VAE framework compresses the high-dimensional image x into a low-dimensional global latent representation z, as follows:

$$q_\phi(z|x) = N(z; \mu(x), \sigma^2(x)) \tag{9}$$

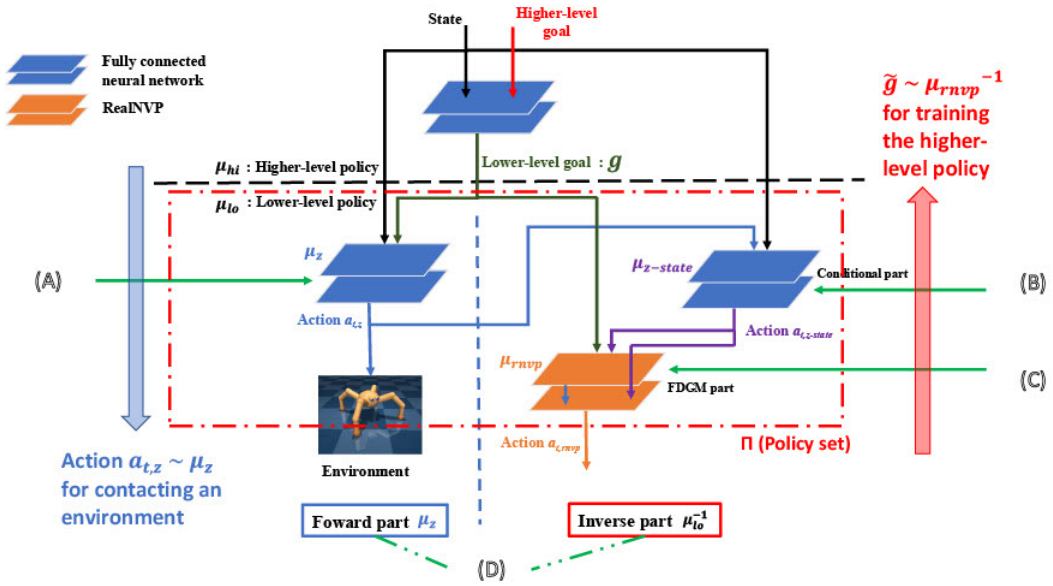


Fig. 3. The architecture of our proposed model utilizes FDGM and incorporates the following key components: (A) A low-level policy from HIRO is employed exclusively for the forward part, which interacts with the environment. (B) The observation embedding of LSP, which functions as an encoder, is replaced by the low-level policy of HIRO, now treated as an independent policy. (C) The RealNVP component of LSP, a neural network integrated into a policy, is repurposed as an independent policy. (D) The combination of the forward and inverse parts is adopted from the model architecture of (Ma et al. 2020).

where $\mu(\cdot)$ and $\sigma(\cdot)$ are neural networks that learn the mean and variance, respectively, and the variational posterior distribution $q_\phi(z|x)$ models the latent variable Z as a diagonal Gaussian.

To address the biased log-density estimation of FDGM in reconstructing the image x, $p_\theta(x|z)$ is considered using z. An invertible flow-based decoder $f_\theta(x;z)$ is employed, with the image x as the main input and the global latent representation z as the conditional input, to obtain a local representation v:

$$v = f_\theta(x;z) \tag{10}$$

where v follows a standard normal distribution, $v \sim N(0,I)$. Finally, the image x is reconstructed using the inverse function of the flow-based decoder:

$$x = f_\theta^{-1}(v;z). \tag{11}$$

We adopt this architecture because it resolves two key challenges: the flexibility of the goal dimension between policies and the biased log-density estimation for the FDGM in our model.

Auto-Regressive Flow An autoregressive flow provides tractable likelihoods because the probability of a full sequence can be expressed as the product of conditional probabilities of past observations. This property enables better density estimation compared to non-autoregressive flow-based models (Germain et al. 2015; Kingma, Salimans, et al. 2016; Oord et al. 2016; Papamakarios et al. 2017; Van Oord et al. 2016). However, the restoration speed of the original data, which is a consequence of its sequential operation, counterbalances this advantage.

Table 1. Key Notations.

Symbol	Meaning
t	action step
μ_{hi}	higher-level policy
μ_{lo}	lower-level policy
μ_z	the policy of forward part
$\mu_{z-\text{state}}$	the policy of conditional part
μ_{rnvp}	the policy of FDGM part
μ_{rnvp}^{-1}	the inverse of the policy of FDGM part
Π	the lower-level policy set composing of $[\mu_z, \mu_{z-\text{state}}, \mu_{\text{rnvp}}]$
L_z	the loss of forward part
$L_{z-\text{state}}$	the loss of conditional part
L_{rnvp}	the loss of FDGM part
L_{common}	the averaged loss over the sum of loss of all three policies
g_t *or* g	the action of higher-level policy called a goal
$\tilde{g}_t$	a goal obtained through off-policy correction for training the higher-level policy μ_{hi}
a_t	the action of lower-level policy
R_t	the extrinsic reward of an environment
r_t	the intrinsic reward of lower-level policy
s_t *or* s	the state of the environment
$a_{t,z}$	the action of forward part
$a_{t,\text{rnvp}}$	the action of FDGM part of inverse part
$a_{t,z-\text{state}}$	the action of conditional part of inverse part
$[g, a]$	the concatenation of $g_{1:d}$ and an action $a_{t,z-\text{state}}$
$[a_{\text{rnvp}}, a_{z-\text{state}}]$	the concatenation of an action $a_{t,\text{rnvp}}$ and an action $a_{t,z-\text{state}}$
$x_{t:t+c-1}$	the sequence $x_t,\ldots,x_{t+c-1}$
c	the horizon of a higher-level policy
D_R	replay buffer

4 Our Model

The aim of our research is not to rely on indirect estimation, as in HIRO, but rather to employ direct estimation that accurately reflects the current knowledge of the lower-level policy. This approach enables us to find an optimal goal for level synchronization when training the higher-level policy. The most direct method is to determine a goal, $\tilde{g}_t$, through the inverse operation of the lower-level policy, which is then used to train the higher-level policy μ_{hi}. This is expressed as:

$$\tilde{g}_t \sim \mu_{lo}^{-1}(a_t) \tag{12}$$

where a_t is the action of lower-level policy, μ_{lo}.

Given that FDGM supports invertible algorithms and fast generation speeds, we adopt an FDGM-based lower-level policy in our research to relabel goals for training the higher-level policy.

However, although FDGM is invertible, its bijective transformation property imposes constraints on goal dimensions, particularly when incorporating an inverse model into model-free HRL. This limitation stems from the requirement that the input and output dimensions of FDGM must be identical, making it difficult to design a

Algorithm 1: Find the goal, $\tilde{g}$, using only μ_{rnvp}^{-1} without forward part (Normal version)

Input: $a_{z-\text{state}}$ and a_{rnvp}
Output: $\tilde{g}$
Init:

- Set the layer dimension of all policies including μ_{hi} and the critic of μ_{rnvp} to be the same except for the actor of μ_{rnvp}
- Set the layer dimension of actor of μ_{rnvp}
- Set the action dimension of $\mu_{z-\text{state}}$

while c **do**
  $a_{z-\text{state}}$ *and* $a_{\text{rnvp}} \sim D_R$;
  $\tilde{g} \sim \mu_{\text{rnvp}}^{-1}(a_{z-\text{state}}, a_{\text{rnvp}})$;

model that accommodates flexible goal dimensions between policies in HRL. Additionally, FDGM suffers from the chronic drawback of biased log-density estimation.

To address these challenges, we define a lower-level policy set, denoted as Π, for the lower-level policy:

$$\Pi = [\mu_z,\ \mu_{z-\text{state}},\ \mu_{\text{rnvp}}] \tag{13}$$

We split the lower-level policy, μ_{lo}, into two parts: a forward part composed of μ_z, which interacts with the environment, and an inverse part, μ_{lo}^{-1}, consisting of $\mu_{z-\text{state}}$ and μ_{rnvp} for the inverse model. The forward part is responsible for generating a real action in the environment. The action of the lower-level policy, denoted as $a_t \sim \mu_{lo}$, is determined by μ_z from Π at the current state s_t. The inverse part includes two components: the μ_{rnvp} part, which uses the output of the conditional part, $\mu_{z-\text{state}}$, and a goal from the higher-level policy as inputs, and the conditional part, $\mu_{z-\text{state}}$, which supports the μ_{rnvp} part.

To obtain the local representation, the main inputs of the inverse part of our model are s_t and g_t for the FDGM part, and the output of the forward part, $a_{t,z}$, for the conditional part. However, the purpose of the inverse part is to find $\tilde{g}_t$ using the inverse operation of the FDGM part. To achieve this, one of the inputs of the FDGM part, s_t, is transferred to the conditional part as an input. This design enhances performance and distinguishes our approach from the observation embedding of LSP, which functions solely as an encoder.

To ensure the exact inverse operation of our model with respect to the output of the corresponding level of the lower-level policy, it is critical to align the output of the inverse part, $a_{t,\text{rnvp}}$, with the output of the forward part, $a_{t,z}$. Essentially, both outputs, $a_{t,z}$ and $a_{t,\text{rnvp}}$, should share a common internal factor under the shared inputs—a goal g_t and a state s_t—despite their differing dimensions. Therefore, $a_{t,z-\text{state}}$ of $\mu_{z-\text{state}}$ serves as the refined global representation.

Through Π, our model ensures the flexibility to choose the dimension of the goal at any level while addressing the issue of biased log-density estimation in FDGM. Since off-policy correction is not required at the highest level, all levels except the highest can adopt the two-part architecture of our model: a forward part and an inverse part.

Fig. 3 illustrates the full architecture of our model, which is based on a two-level hierarchical structure. Each component of the lower-level policy, μ_{lo}, including the off-policy correction employed in our model, is explained in detail in the following section.

4.1 Forward Part μ_z

Similar to the lower-level policy in HRL, the action of the forward part interacts with either the environment or another lower-level policy. We model the forward part using the VAE framework proposed by (Ma et al. 2020),

Algorithm 2: Find the goal, $\tilde{g}$, using only μ_{rnvp}^{-1} with forward part (Variant version)

Input: s, g *and* a_{rnvp}
Output: $\tilde{g}$
Init:

- Set the layer dimension of all policies (a forward part and an inverse part) inside Π to be the same except for μ_{hi}
- Set the layer dimension of μ_{hi}
- Set the action dimension of $\mu_{z-\text{state}}$

while c **do**

s, g *and* $a_{\text{rnvp}} \sim D_R$;
$a_z^{'} \sim \mu_z(s, g)$;
$a_{z-\text{state}}^{'} \sim \mu_{z-\text{state}}(s, a_z^{'})$;
$\tilde{g} \sim \mu_{\text{rnvp}}^{-1}(a_{z-\text{state}}^{'}, a_{\text{rnvp}})$

which extracts the global representation of a goal g_t and a state s_t in HRL as follows:

$$a_{t,z} \sim \mu_z(s_t, g_t) \text{ at time } t \tag{14}$$

where μ_z is the policy of the forward part, and $a_{t,z}$ is the action of μ_z. The action $a_{t,z}$ serves as the global representation, which aids in finding the goal $\tilde{g}_t$ in the inverse part for relabeling $g_t \sim D_R$.

The fixed dimension of the action $a_{t,z}$, determined by the given environment and distinct from the dimension of g_t, compensates for a potential limitation of the inverse part, which relies on an FDGM and requires dimension matching with g_t. Since the outputs of both parts, $a_{t,z}$ and $a_{t,\text{rnvp}}$, are equivalent from the perspective of the higher-level policy, we can treat $a_{t,z}$—the output of the forward part—as a global representation corresponding to the output of the inverse part, $a_{t,\text{rnvp}}$.

4.2 Inverse Part μ_{lo}^{-1}

We aim to utilize the inverse operation of FDGM to compute a goal $\tilde{g}_t$ that aligns with the updated lower-level policy, μ_{lo}. Our model architecture is inspired by the approach of (Ma et al. 2020), which employs two distinct representations to balance the strengths and weaknesses of FDGM.

4.2.1 Conditional Part $\mu_{z-\text{state}}$

While the conditional part serves a role similar to the observation embedding in LSP as a conditional input for FDGM, it differs in two key aspects. First, unlike the observation embedding in LSP, which is not trained as an encoder using a fully connected neural network, the conditional part operates as an independent policy to enhance performance. Second, $\mu_{z-\text{state}}$ takes both the state s_t and action $a_{t,z}$ as inputs to process the global representation, whereas the observation embedding in LSP only takes the state s_t to adjust its dimension. This is expressed as:

$$a_{t,z-\text{state}} \sim \mu_{z-\text{state}}(s_t, a_{t,z}) \text{ at time } t \tag{15}$$

where $\mu_{z-\text{state}}$ is the policy of the conditional part and $a_{t,z}$ is the action of the forward part.

There are three reasons why we incorporate the conditional part. First, it provides a refined global representation $a_{t,z}$ while preserving the local representation $a_{t,\text{rnvp}}$ as a conditional input for FDGM. Second, to address the chronic weakness of the FDGM part—biased log-density estimation—we account for the influence of the conditional part, which includes the global representation of a goal g_t and a state s_t.

The second reason is that controlling the dimension of the action $a_{t,z-\text{state}}$ enhances the effectiveness of $a_{t,z}$ in the FDGM part. Although $a_{t,z}$ is fixed by the environment, we aim to indirectly influence its dimension through $a_{t,z-\text{state}}$ to optimize its impact on the output of the FDGM part.

The third reason is to adjust the dimension of the state s_t if it is greater or less than that of the goal g_t, as both are original inputs to the FDGM part. This adjustment helps achieve a more accurate $\tilde{g}_t$ through μ_{lo}^{-1}.

Finding the optimal dimension for $a_{t,z-\text{state}}$ involves a trade-off process that can be refined through experimentation.

4.2.2 FDGM Part μ_{rnvp}

The inputs to the FDGM part are a goal g_t, provided by the higher-level policy as the main input, and an action $a_{t,z-\text{state}}$, provided by the conditional part as the conditional input. This is expressed as:

$$a_{t,\text{rnvp}} \sim \mu_{\text{rnvp}}(g_t, a_{t,z-\text{state}}) \text{ at time } t \tag{16}$$

where μ_{rnvp} is the policy of FDGM part and $a_{t,\text{rnvp}}$ is the action of μ_{rnvp}. The principle of the FDGM part can be explained using the forward transformation property of RealNVP (Dinh, Sohl-Dickstein, et al. 2016) in a single layer:

$$\begin{aligned} a\_\text{rnvp}_{1:d} &= g_{1:d} \\ a\_\text{rnvp}_{d+1:D} &= g_{d+1:D} \odot \exp(s([g,a])) + t([g,a]) \end{aligned} \tag{17}$$

where $\odot$ represents the element-wise product. A goal g_t is split into a $g_{1:d}$ and a $g_{d+1:D}$. The concatenation of $g_{1:d}$ and the action $a_{t,z-\text{state}}$ forms $[g,a]$. The outputs of a coupling layer, $a\_\text{rnvp}_{1:d}$ and $a\_\text{rnvp}_{d+1:D}$ are concatenated for the next layer. Furthermore, $s(\cdot)$ and $t(\cdot)$ are scale and translation functions that map $R^d \mapsto R^{D-d}$. These functions are implemented using a deep neural network, as the inverse function of each bijection and the Jacobian determinant do not require explicit computation of s and t. The inverse operation, along with the forward operation, can be calculated as follows:

$$\begin{cases} a\_\text{rnvp}_{1:d} = g_{1:d} \\ a\_\text{rnvp}_{d+1:D} = g_{d+1:D} \odot \exp(s([g,a])) + t([g,a]) \end{cases} \tag{18}$$

$$\Longleftrightarrow \begin{cases} g_{1:d} = a\_\text{rnvp}_{1:d} \\ g_{d+1:D} = (a\_\text{rnvp}_{d+1:D} - t([a_{\text{rnvp}}, a_{z-\text{state}}])) \\ \qquad \odot \exp(-s([a_{\text{rnvp}}, a_{z-\text{state}}])) \end{cases} \tag{19}$$

Here, an action $a_{t,\text{rnvp}}$ is split into $a\_\text{rnvp}_{1:d}$ and $a\_\text{rnvp}_{d+1:D}$. The concatenation of $a_{t,\text{rnvp}}$ and $a_{t,z-\text{state}}$ forms an $[a_{\text{rnvp}}, a_{\text{z-state}}]$. The outputs of a coupling layer in the inverse operation are $g_{1:d}$ and $g_{d+1:D}$.

The dimension of $a_{t,\text{rnvp}}$ is determined by the dimension of g_t due to the bijective transformation property of FDGM. The output of the FDGM part, $a_{t,\text{rnvp}}$, is not used to provide an action to the environment but rather for the off-policy correction of its higher-level policy after being stored in the replay buffer D_R.

4.3 Off-policy Correction, Convergence and Algorithm

A replay buffer collected by the higher-level policy for off-policy RL is filled with state-action-reward transitions (s_t, g_t, $a_{t,z}$, $a_{t,z-\text{state}}$, $a_{t,\text{rnvp}}$, $\Sigma R_{t:t+c-1}$, s_{t+c}) based on a higher-level transition tuple ($s_{t:t+c-1}$, $g_{t:t+c-1}$, $a_{t:t+c-1,z}$, $a_{t:t+c-1,z-\text{state}}$, $a_{t:t+c-1,\text{rnvp}}$, $R_{t:t+c-1}$, s_{t+c}). During off-policy training with the step t transition of the replay buffer at time $t+c$ steps, the following inequality holds:

$$J_{(\ldots,\, g_t, \ldots) \sim D_R}(\theta)_{t+c} \neq J(\theta)_{t+c}^{OPT} \tag{20}$$

where $J_{(\ldots,\, g_t, \ldots) \sim D_R}(\theta)_{t+c}$ is the objective function of μ_{hi}^{θ} and $J(\theta)_{t+c}^{OPT}$ is the optimal objective function of μ_{hi}^{θ} under the condition that $a_t \neq a_{t+c}$ as mentioned at time $t+c$ in Section 2.2.

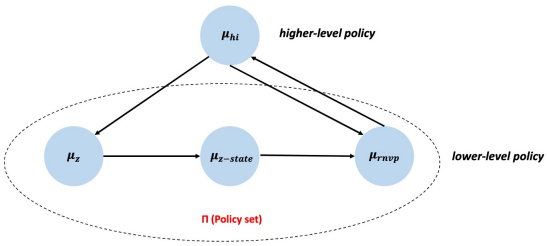


Fig. 4. Communication graph representing the dependence of our model

In this research, we first address the input-output dimension issue and biased log-density estimation of the FDGM through the coordinated use of μ_z and $\mu_{z-\text{sate}}$. Next process begins when μ_{rnvp} generates an action $a_{t+c,\text{rnvp}}$ conditioned on the outputs of $\mu_{z-\text{state}}$. These three components, which are μ_z, $\mu_{z-\text{sate}}$ and μ_{rnvp}, are then jointly trained using a shared loss function $L_{\text{common actor}}$ and $L_{\text{common critic}}$ explained in Section 5. This unified training scheme enables stable inversion of the FDGM in the lower-level policy through the following procedure:

$$\tilde{g}_{t+c} \sim \mu^{-1}_{\text{rnvp}}(a_{t,\text{rnvp}}, a_{t,z-\text{state}}) \tag{21}$$

using $a_{t,\text{rnvp}}$ and $a_{t,z-\text{state}}$ taken from the replay buffer to obtain $\tilde{g}_{t+c}$. To relabel g_t, $\tilde{g}_{t+c}$ is utilized for the level synchronization of adjacent policies when the higher-level policy is trained after the lower-level policy is trained in a bottom-up, layerwise fashion within HRL. As g_t is relabelled with $\tilde{g}_{t+c}$ through the off-policy correction of our model, it approaches $g_{t+c,ideal}$ which ideally reflects the current internal parameter μ^{θ}_{rnvp} of μ_{lo}. Consequently, the objective function of $\mu_{t+c,hi}$ converges to its optimal value, provided the agent experiences all states of the environment. This relationship is expressed as:

$$\begin{aligned} &\text{If } \lim_{s\to\infty} \tilde{g}_{t+c} = g_{t+c,ideal}, \\ &\text{then } \lim_{s\to\infty} J_{(\ldots,\tilde{g}_{t+c},\ldots)\sim D_R}(\theta)_{t+c} = J(\theta)^{OPT}_{t+c}. \end{aligned} \tag{22}$$

Finally, the goal $\tilde{g}_{t+2c} \sim \mu_{t+2c,hi}$ represents the optimal target for both $\mu_{t+2c,z}$ *and* $\mu_{\text{t+2c, rnvp}}$, as the higher-level policy $\mu^{\theta}_{t+2c,hi}$ is dynamically aligned with the current lower-level policy $\mu^{\theta}_{t+2c,lo}$.

Our model combines three policies of the same type, along with the common losses $L_{\text{common actor}}$ and $L_{\text{common critic}}$ which are explained in Section 5, into a unified lower-level policy μ_{lo}. The dependencies between these policies are illustrated in Fig. 4. Our research leverages TD3, which has been validated for its convergence (Fujimoto et al. 2018), for all policies. As a result, the convergence of each policy level in our model depends on the interplay of the three sub-policies and the higher-level policy, adhering to the convergence characteristics of the atomic policies in off-policy training.

The dynamics of the inverse operation using our inverse model vary for each task within the environment. To address this, we utilize Algorithm 1 and Algorithm 2 to find an optimal goal for off-policy correction using our inverse model. The process is as follows:

- Step 1: Set the layer dimensions (dim.) of all policies in our model and the action dimension of $\mu_{z-\text{state}}$.
- Step 2: Employ their respective methods to find a goal through μ_{lo}. Section 5 provides the layer dimensions of all policies and the action dimension of $\mu_{z-\text{state}}$ based on the algorithm used in each experimental result.

5 Experiments

In this research, we investigate whether our inverse model in HRL, which employs direct estimation with FDGM for off-policy correction, can achieve competitive performance compared to HIRO and LSP. Our implementation is based on the HIRO open code[1] and the RealNVP open code[2] from LSP, using the same optimization algorithm to ensure transparent benchmarking. As a result, we integrate our model and LSP into the HIRO code framework. We have also released our code[3] along with detailed explanations.

We employ a two-level hierarchical structure for HRL, as it aligns with the structure used in HIRO. The intrinsic reward function of the lower-level policy, as defined in HIRO, is used without modification. All neural networks, including the critic of the FDGM part, consist of fully connected layers, consistent with the implementation in HIRO. The only exception is the actor of the FDGM part, which is implemented as a RealNVP.

HIRO utilizes the TD3 policy gradient algorithm, a variant of DDPG designed for continuous control. Since our model is built on the HIRO source code, all policies in our model also employ TD3. Consequently, all policies within the lower-level policy set, Π, leverage the actor and critic losses of DDPG for their learning. Instead of computing separate losses for each policy in Π, the losses used for training the actor and critic of the three policies are combined into common losses, $L_{\text{common actor}}$ and $L_{\text{common critic}}$. These common losses are calculated as the average of the individual losses of all three policies, as follows:

$$\begin{aligned} L_{\text{common actor}} &= (L_{z\,actor} + L_{z-\text{state actor}} + L_{\text{rnvp actor}})/3 \\ L_{\text{common critic}} &= (L_{z\,critic} + L_{z-\text{state critic}} + L_{\text{rnvp critic}})/3 \end{aligned} \tag{23}$$

where:

- $L_{z\,actor}$ and $L_{z\,critic}$ are the actor loss and critic loss of μ_z
- $L_{z-\text{state actor}}$ and $L_{z-\text{state critic}}$ are the actor loss and critic loss of $\mu_{z-state}$
- $L_{\text{rnvp actor}}$ and $L_{\text{rnvp critic}}$ is the actor loss and critic loss of μ_{rnvp}

The implementation of the lower-level policy and off-policy correction in the HIRO open-source code has been modified. To achieve the primary objective of the experiment, we introduce two specific modifications to the HIRO code:

(1) We replace the neural network of the lower-level policy in HIRO with the architecture of our model.
(2) We replace the indirect probabilistic method used in HIRO with the inverse operation of our model, as outlined in Algorithm 1 or Algorithm 2, for off-policy correction. All other parts of the code remain unchanged.

For LSP, we replace all of its policies with the HIRO model architecture on the goal space, as designed in HIRO, except for the actor of the FDGM part, as previously mentioned. Although LSP does not originally incorporate off-policy correction, we apply off-policy correction using the inverse operation of the FDGM in the lower-level policy to ensure a fair comparison.

Since HIRO investigates the Ant environment in MuJoCo, which is part of the OpenAI Gym, the HIRO open-source code defines optimal maximum and minimum tuple values for the goals of both the higher-level and lower-level policies. In this experiment, we adopt these values as the standard goal spaces. For further details, refer to Appendix A and B.

In our evaluation, we focus on the more challenging tasks within the Ant environment, such as Ant Push and Ant Fall, which include different modes (Multi and Single) among the tasks available in HIRO.

[1] https://github.com/tensorflow/models/tree/master/research/efficient-hrl
[2] https://github.com/haarnoja/sac
[3] https://github.com/jangikim2/Hierarchical_Reinforcement_Learning/tree/master/distributed_inverse-efficeint-hrl-sac_210127_tfp_bijectors.bijector_dataefficient_three_reflected_888_0420

We conduct two comparisons between our model and HIRO, focusing on the methodology for off-policy correction and the biased log-density estimation related to FDGM:

(1) Optimal Goal Space Comparison in Section 5.1: We use the optimal maximum and minimum tuple values for the goals of the higher-level and lower-level policies, as defined in HIRO, as default parameters.
(2) **Non-Optimal Goal Space Comparison** in Section 5.2: We modify some dimensions of the optimal maximum and minimum tuple values for the goals of the higher-level and lower-level policies, as defined in HIRO.

Our model is also compared with LSP in Section 5.3 in terms of the average reward achieved by the higher-level policy, focusing on the inherent restriction between the input and output dimensions of FDGM. Additionally, an ablation study is conducted to compare a variant of our model with the original model in Section 5.4. Both our model and HIRO are evaluated on tasks from the MuJoCo Ant environment in OpenAI Gym, including Ant Push Multi, Ant Fall Multi, Ant Push Single, and Ant Fall Single, to assess the performance of our research. Specifically, our model is evaluated against LSP in Ant Push Single and Ant Fall Single. Further details of the experiments are provided in Appendix A and B.

The objectives of our experiments are as follows:

(1) To verify that our model improves agent performance more effectively than HIRO in terms of off-policy correction.
(2) To determine the extent to which our model addresses the chronic issue of FDGM in HRL.
(3) To evaluate the impact of the flexible goal dimension in our model on the agent's performance compared to LSP.
(4) To compare our model with a variant of our model.

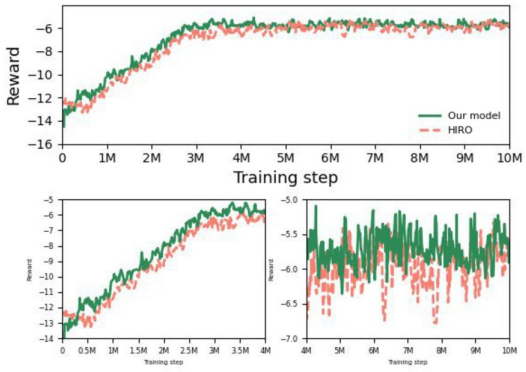
5.1 Comparative Analysis with the Optimal Goal Space

Algorithm 1 and 2 aim to find the optimal $\tilde{g}_t$ from the FDGM part, reflecting the current knowledge of the lower-level policy in each task. In Fig. 5, the average results of our model for the tasks of the higher-level policy are compared with those of HIRO using default parameters, including the optimal goal space. Our model demonstrates significantly better performance in most tasks.

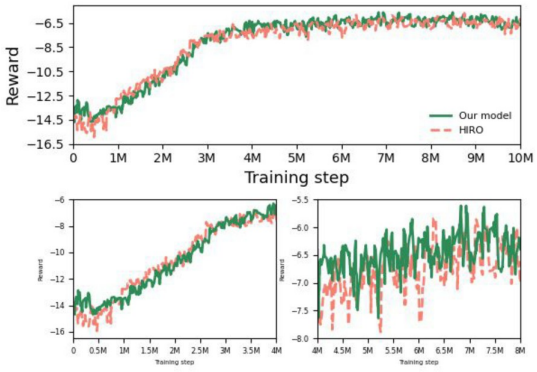
- Ant Push Multi — Actor of FDGM : 145 dim., All other NNs : 170 dim., $a_{t,z-\text{state}}$: 16 dim. — in Fig. 5a: Our model achieves better scores than HIRO with default parameters until 4.5M. Beyond this point, our results marginally outperform HIRO up to 10M. Algorithm 1 is used for our model.
- Ant Fall Multi — Actor of FDGM : 150 dim., All other NNs : 170 dim., $a_{t,z-\text{state}}$: 18 dim. — in Fig. 5b: Our model and HIRO with default parameters achieve similar results until 4.8M. After this, our model's performance is slightly better up to 8M. Algorithm 1 is used for our model.
- Ant Push Single — Actor of FDGM : 130 dim., All other NNs : 150 dim., $a_{t,z-\text{state}}$: 19 dim. — in Fig. 5c: Our model achieves considerably better performance than HIRO with default parameters until 10M, although our model's results are not as strong as HIRO's up to 3M. Algorithm 1 is used for our model.
- Ant Fall Single — All lower policy's NNs : 135 dim., All higher policy's NNs : 160 dim., $a_{t,z-\text{state}}$: 24 dim. — in Fig. 5d: Our model achieves a much faster learning speed up to 5.25M compared to HIRO with default parameters. Our results are similar to HIRO's up to 8M, after which they are slightly lower until 10M. Algorithm 2 is used for our model.

5.2 Comparative Analysis with a Non-Optimal Goal Space

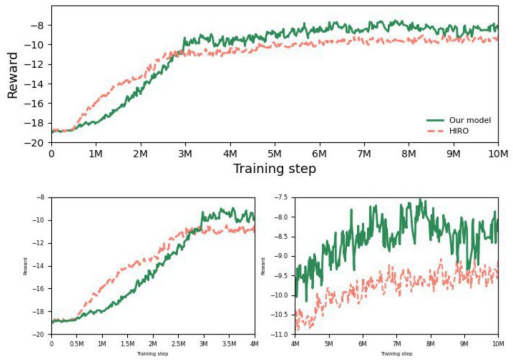
A non-optimal goal space is required for a fair comparison between the two models, HIRO and our model. We consider two cases for the non-optimal goal space:



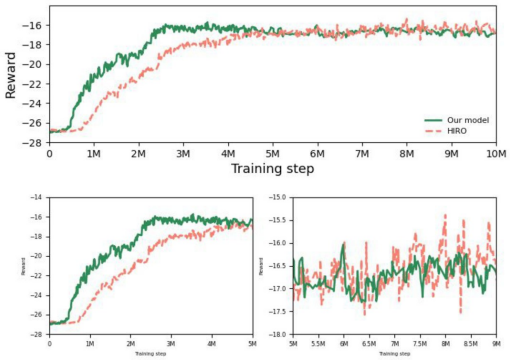
(a) Ant Push Multi



(b) Ant Fall Multi



(c) Ant Push Single



(d) Ant Fall Single

Fig. 5. For the task of the higher-level policy, the average rewards of our model (green) are compared with that of HIRO (salmon) with the same NN size on condition of the optimal goal space.

(1) First Case: The higher-level policy's goal dimension is set to 8, replacing its default goal dimension of 2 or 3 depending on the task, with the first 8 goals of the lower-level policy of HIRO, whose default goal dimension is 15. This configuration is represented as (Our model or HIRO)_8_15, where the notation 'Model name_goal dimension of the higher-level policy_goal dimension of the lower-level policy' is used. As a result, the first 8 default goals of the lower-level policy of HIRO are selected as the goals for the higher-level policy of (Our model or HIRO)_8_15. The goal space of the lower-level policy in (Our model or HIRO)_8_15 remains the same as the default goal space of the lower-level policy of HIRO.
(2) Second Case: The configuration (Our model or HIRO)_8_8 removes the last 7 goals of the lower-level policy of (Our model or HIRO)_8_15, reducing the lower-level policy's goal dimension to 8.

Figure 9 in the Appendix provides further details about these two goal spaces.

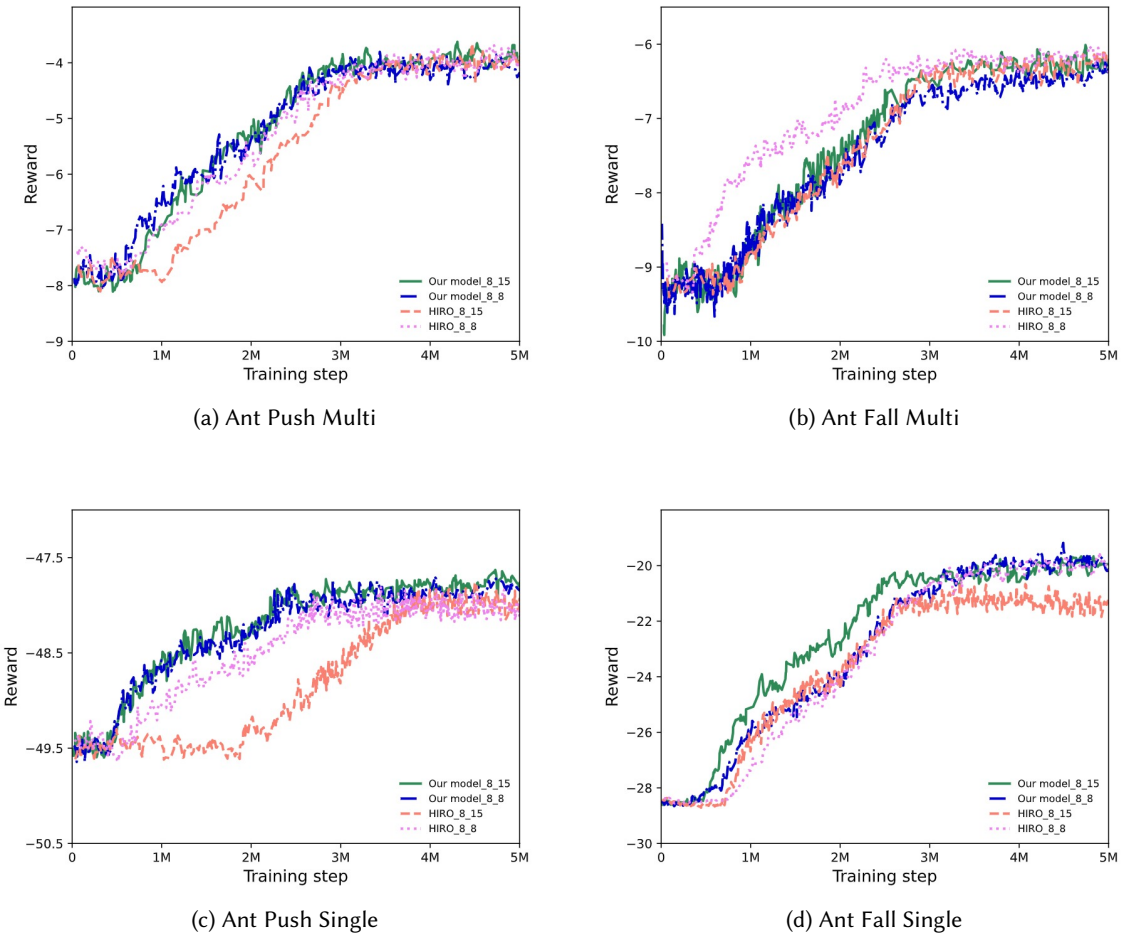


Fig. 6. The average rewards of our model (green, blue) for the task of the higher-level policy are compared with that of HIRO (salmon, violet) on condition of a non-optimal goal space. The numbers of (our model or HIRO)_8_(15 or 8) are the goal dimension of higher-level policy and that of lower-level policy respectively.

For the comparison with HIRO, we first tune the appropriate parameters—specifically, the layer dimensions of all policies at each level and the dimension of $a_{t,z-\text{state}}$ in the conditional part—to optimize the performance of our model in each task using Algorithm 1 or 2. Once the best-performing parameters for our model are determined, we compare the performance of HIRO, configured with the same layer dimensions, against our model for each task.

The conditions for achieving the best performance with our model are as follows:

- Ant Push Multi — Our model - Actor of FDGM : 135 dim., All other NNs : 160 dim., Our model_8_15 - $a_{t,z-\text{state}}$: 12 dim., Our model_8_8 - $a_{t,z-\text{state}}$: 20 dim. — in Fig. 6a: Our model_8_8 and Our model_8_15 show better performance than HIRO_8_8, which is the best among HIRO's two models. Algorithm 1 is used for our model.

- Ant Fall Multi — Our model - Actor of FDGM : 140 dim., All other NNs : 165 dim., Our model_8_15 - $a_{t,z-\text{state}}$: 12 dim., Our model_8_8 - $a_{t,z-\text{state}}$: 15 dim. — in Fig. 6b: HIRO_8_8 shows the best performance among all other models. Algorithm 1 is used for our model.
- Ant Push Single — Our model - Actor of FDGM : 130 dim., All other NNs : 150 dim., Our model_8_15 - $a_{t,z-\text{state}}$: 9 dim., Our model_8_8 - $a_{t,z-\text{state}}$: 22 dim. — in Fig. 6c: Both of our models show better performance than HIRO's two models. Algorithm 1 is used for our model.
- Ant Fall Single — Our model - Actor of FDGM : 135 dim., All other NNs : 160 dim., Our model_8_15 - $a_{t,z-\text{state}}$: 26 dim., Our model_8_8 - $a_{t,z-\text{state}}$: 22 dim. — in Fig. 6d: Our model_8_15 shows the best performance among all other models. The inverse operation of Algorithm 2, combined with the NN size from Algorithm 1, is used for our model.

Our model demonstrates the best performance in most tasks. However, in Fig. 6b, our model's performance falls slightly behind HIRO's. Compared to tasks with a high reward shape, such as Ant Push Multi and Ant Fall Multi, our model shows significantly superior performance over HIRO in tasks with a low reward shape, such as Ant Push Single in Fig. 6c and Ant Fall Single in Fig. 6d, where there is ample room for improvement. Notably, our model performs impressively in the early stages, even when the replay buffer lacks sufficient data for off-policy correction. This is primarily because our model can leverage an exact goal, $\tilde{g}_t$, through direct inference from the inverse operation of FDGM.

5.3 Comparative Analysis Between Our Model and LSP

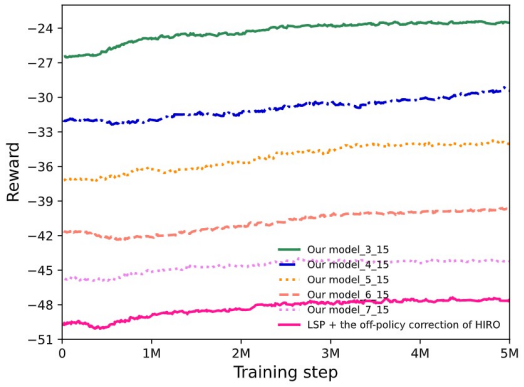
One of the primary objectives of our model is to enable flexible goal spaces for all policies in HRL without any restrictions. Our model_(2,3,4,5,6 or 7)_(8 or 15) exemplifies this goal flexibility. These non-optimal goal spaces are constructed using the same method as (Our model or HIRO)_8_(8 or 15). Specifically, the goal space of the higher-level policy is replaced with the first x goals (where x = 2,3,4,5,6 or 7) of the lower-level policy of HIRO, whose default goal dimension is 15. Fig. 9 in the Appendix provides examples of these goal spaces.

In Fig. 7, our model_(2,3,4,5,6 or 7)_(8 or 15) is compared with LSP to demonstrate the superiority of our model's goal flexibility. The layer dimensions of our model are the same as those in the corresponding tasks of our model_8_(8 or 15) in Fig. 6.

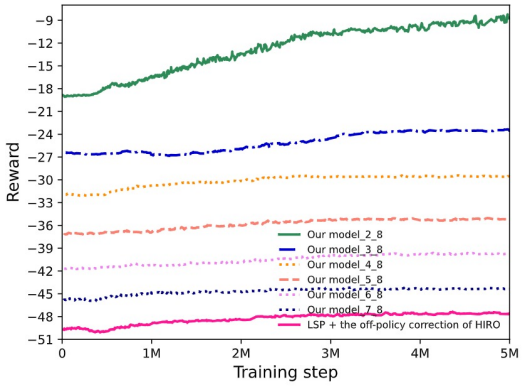
LSP, due to the invertible bijective transformation property of FDGM, requires that the output dimension for selecting a goal space for all policies in HRL must match the action space dimension of the environment, which is 8 in the MuJoCo Ant environment. As a result, LSP is constrained to LSP_8_8. The layer dimensions of LSP are configured based on the settings of our model in Ant Push Single in Fig. 6c and Ant Fall Single in Fig. 6d. Additionally, the observation embedding dimension of LSP is tuned to achieve optimal performance, as follows:

- Ant Push Single — Higher policy's NN : 150 dim., Lower policy's NN : 130 dim., Output dimension of each level's observation embedding : 23 dim. — in Fig. 7a and 7b
- Ant Fall Single — Higher policy's NN : 160 dim., Lower policy's NN : 135 dim., Output dimension of each level's observation embedding : 14 dim. — in Fig. 7c and 7d

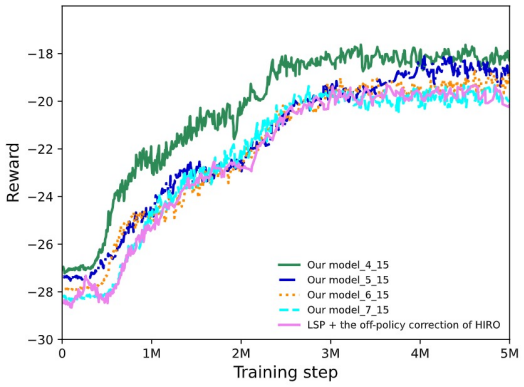
Fig. 7 demonstrates the effectiveness of selecting the goal dimension for the higher-level policy of our model. It reveals that smaller goal dimensions for the higher-level policy of our model lead to better performance. This trend suggests that the goal dimension of the higher-level policy of our model approaches the default goal dimension of HIRO's higher-level policy. LSP performs significantly worse than our model, underscoring the superiority of our model's flexibility in goal dimension.



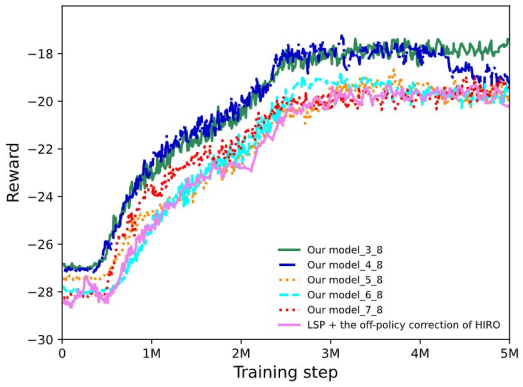
(a) Ant Push Single



(b) Ant Push Single



(c) Ant Fall Single



(d) Ant Fall Single

Fig. 7. For the task of the higher-level policy, the average rewards of our model (green, blue, orange, salmon, violet and navy for Ant Push Single and green, blue, orange, cyan and red for Ant Fall Single) are compared with that of LSP (pink for Ant Push Single and violet for Ant Fall Single). They are based on the change of higher-level goal dimension of a) our model_8_15, (b) our model_8_8 (c) our model_8_15 (d) our model_8_8 used in the corresponding task of Fig. 6.

5.4 Ablation Study

The importance of $a_{t,z}$ is of particular interest due to its significant role in addressing the chronic issue of FDGM, specifically the biased log-density problem. To explore its impact on our model's performance, we investigate how replacing $a_{t,z}$ with the goal g_t as the input to the conditional part affects the model's performance.

- Ant Fall Multi of variant model – Actor of FDGM : 150, All other NNs : 180, $a_{t,z-\text{state}} = 19$

In this experiment, we do not set the parameters of each policy to match those of our model in the comparative analysis. Instead, we first identify the parameters of the variant model with the optimal goal space to achieve its best performance in the Ant Fall Multi task from Section 5.1. We then compare the performance of our original

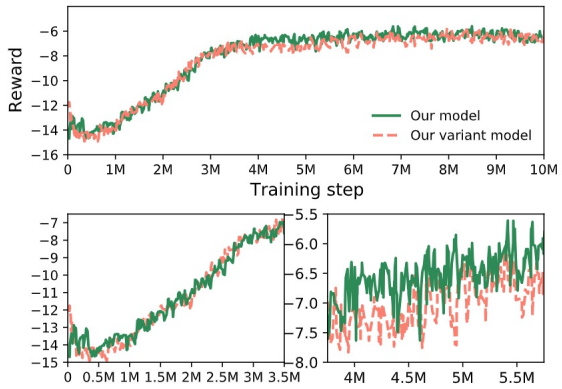


Fig. 8. Average result of our method (green) compared with that of the variant model (salmon) in Ant Fall Multi

model from the previous Ant Fall Multi experiment with the optimal goal space to the variant model with the newly determined parameters.

The results show that while both models perform similarly until 3.5M, our original model marginally outperforms the variant model up to 7.5M. After this point, both models exhibit similar performance up to 10M. The performance comparison between our model and the variant model is illustrated in Fig. 8.

6 Discussion

6.1 The Performance of the Higher-Level Policy Based on Our Model in All Tasks

In most experiments, our model outperforms both HIRO and LSP. Despite not relying on indirect estimation methods, such as the indirect probabilistic approach used in HIRO, our model achieves competitive performance against HIRO and LSP by leveraging the inverse operation of FDGM. Notably, the lower-level policy, which consists of three sub-policies, does not significantly hinder the performance of the higher-level policy during the early learning stages. However, fine-tuning the dimension of $a_{t,z-\text{state}}$ and the layer dimensions of both level policies is essential, as these parameters influence the performance of the higher-level policy.

6.2 The Inference from the Performance of Our Model Compared with LSP

Even with a non-optimal goal space, our model demonstrates competitive performance, underscoring the importance of flexibility in choosing the goal dimension in HRL. In Fig. 7, LSP fails to achieve competitive performance due to its inherent limitation of a fixed goal space. In contrast, our model showcases exceptional performance potential, thanks to its flexibility in selecting the goal space.

6.3 Why is RealNVP Selected as a FDGM in Our Model?

A persistent challenge in FDGMs is biased log-density estimation, which remains an active area of research (Du et al. 2025; Erdogan 2025; C. Li et al. 2025; Liao et al. 2025; Wang et al. 2025). Although RealNVP represents an early research of FDGMs, it has become a standard benchmark in the field due to its foundational contributions. Consequently, RealNVP serves as an appropriate baseline model for our model.

6.4 Can Our Model Function as an Inverse Model in HRL?

Our model demonstrates valid inverse modeling capabilities in HRL, as the lower-level policy μ_{lo} admits an exact invertible representation through three internal components which are μ_z, $\mu_{z-\text{sate}}$ and μ_{rnvp}.

6.5 Is the Role of $\mu_{z-\text{sate}}$ Important for Our Model?

The performance of our model is sensitive to the role of $\mu_{z-\text{sate}}$ supporting an optimized conditional input to μ_{rnvp}. $\mu_{z-\text{sate}}$ decides the optimized synchronization between the global representation of μ_z and the local representation of μ_{rnvp}.

6.6 Is There Any Research in Terms of Off-Policy Correction for HRL Since HIRO?

There is no research regarding off-policy correction for HRL since HIRO. Although we have investigated a relevant research (Cannon and Simsek 2024; J. Li et al. 2022; Schmidt et al. 2024), all researches are different from the off-policy correction of model-free on-line HRL.

6.7 The Inverse Operation and Layer Dim. in Algorithm 1 and Algorithm 2

The rationale for considering Algorithm 1 and 2 is to address the performance level and speed of the higher-level policy, respectively. Since our model employs TD3 for its policies, the FDGM part also incorporates a combination of actor and critic networks. Interestingly, when the layer dimensions of the actor match those of all other neural networks in Algorithm 1, the performance of the higher-level policy falls short of expectations. However, when there is a difference in the layer dimensions, particularly for the actor of FDGM compared to the other neural networks, the performance aligns with our expectations.

Algorithm 1 performs a straightforward inverse operation that reflects only the FDGM part of the lower-level policy. In contrast, Algorithm 2 considers all three sub-parts of the lower-level policy, incorporating different layer dimensions between the higher-level and lower-level policies. Through this research, we observe that the performance level of Algorithm 1 is superior to that of Algorithm 2. However, Algorithm 2 tends to achieve faster performance. In one of our experiments with a non-optimal goal space—specifically, Ant Fall Single—we combine the two algorithms to leverage their respective strengths.

6.8 The Performance Difference Between the Low Reward Shape and the High Reward Shape

Although our model achieves better performance than HIRO across all tasks, there is a noticeable difference in performance between high-reward-shape tasks (Ant Push Multi and Ant Fall Multi) and low-reward-shape tasks (Ant Push Single and Ant Fall Single). This discrepancy may be attributed to two main reasons. First, the chronic issue of FDGM—biased log-density estimation—still affects the inverse operation of FDGM. Specifically, the RealNVP used in our implementation is an early version of FDGM, which may contribute to this limitation. Second, the reconstruction loss of the Autoencoder also plays a significant role. In our model, the forward part and conditional part function as an encoder and decoder, respectively. In environments where the size of the feature space is much larger than that of the action space, the reconstruction loss of the Autoencoder leads to varying performance in tasks with low or high reward shapes.

6.9 The Further Research for an Environment with a Small Action Space and a Big Feature Space

The inputs to the conditional part are s_t and $a_{t,z}$. In environments where the action space is small and the feature space is large, the model architecture of (Ma et al. 2020) for unbiased log-density estimation in our model may not function effectively. This is because the dimension of the global representation, $a_{t,z}$, is significantly smaller than that of s_t. To address this, a decoder for $a_{t,z}$ is required to increase the dimension of the global representation.

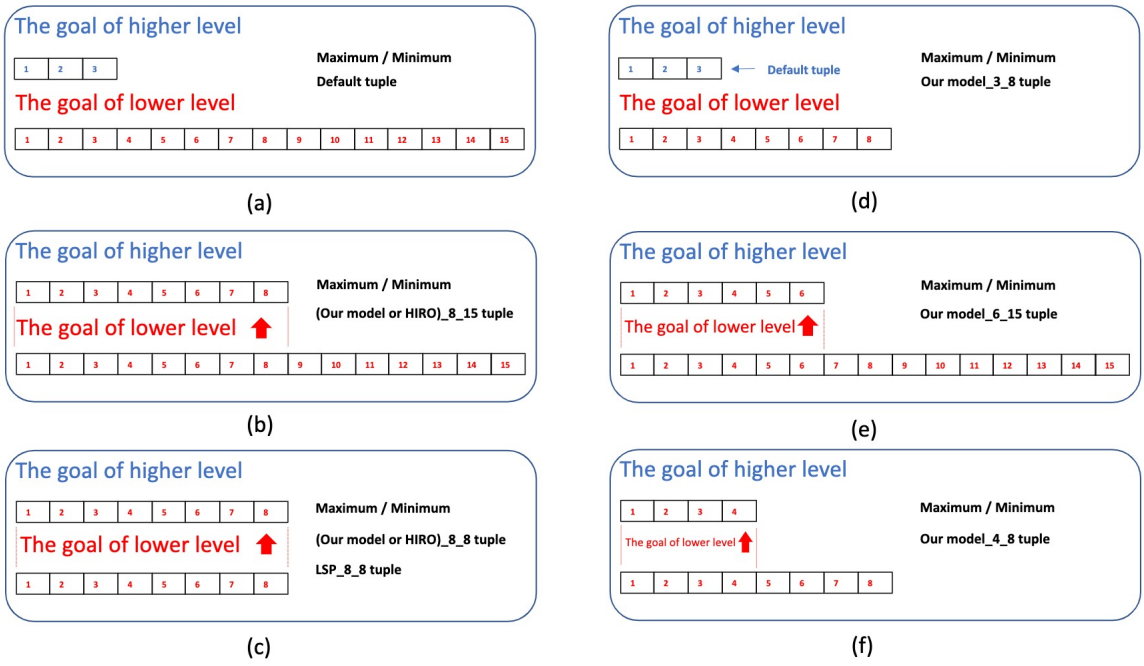


Fig. 9. An example of how to create a non-optimal goal space for the higher-level and lower-level policies of our model, starting from the default optimal goal space tuples of HIRO's higher-level and lower-level policies. Each level's default tuple in (a) is represented with a color—blue for the higher-level policy or red for the lower-level policy—and the position number of the tuple element. Only the tuple elements of the destination which is higher-level's goal are replaced by the corresponding tuple elements of the source which is lower-level's goal. In the case of (Our model, HIRO, or LSP)_8_8 for the lower-level's goal, the last 7 tuple elements of the lower-level goal are removed. (a) The default tuple, which is the optimal tuple, for the higher-level and lower-level policies of HIRO. Appendix A provides the definition of each level's tuple. (b) Our model_8_15 (c) Our model_8_8 (d) Our model_3_8: The goal space of the higher-level policy for 'Our model_2_8' and 'Our model_3_8' is constrained to match the default goal space of Ant Push Single and Ant Fall Single in HIRO, respectively. (d) Our model_6_15 (e) Our model_4_8

7 Conclusion

We have introduced a novel hierarchical reinforcement learning (HRL) inverse model that supports level synchronization using FDGM for off-policy correction in HRL. Our model outperforms HIRO and LSP, as demonstrated in four test tasks within the Ant environment, which includes two reward shapes, high reward shape and low reward shape, in a two-level hierarchical structure. Our model presents a generalizable approach for the inverse model in model-free HRL.

Our research demonstrates that off-policy correction in HRL can be effectively implemented using data acquired from a policy based on FDGM. This approach, known as the direct method, is more practical and broadly applicable, as it relies on natural data computation without requiring peripheral information used in indirect estimations.

To date, there has been limited research on FDGM in the RL domain. Typically, most RL research utilizes general neural networks, such as fully connected neural networks. However, as RL may require additional functionality, such as inverse operations, future RL research could benefit significantly from incorporating generative models.

Further research is needed to enhance our model. Future studies should focus on RL using general FDGM algorithms with feature-based data, rather than image-based data, which is large in size. Additionally, our generalized inverse model could be explored in atomic RL, even though it has primarily been studied in HRL. We also consider extending our work to non-Markovian environments in the future.

References

M. S. Albergo and E. Vanden-Eijnden. 2022. "Building normalizing flows with stochastic interpolants." *arXiv preprint arXiv:2209.15571*.

M. Andrychowicz et al.. 2017. "Hindsight experience replay." *arXiv preprint arXiv:1707.01495*.

P.-L. Bacon, J. Harb, and D. Precup. 2017. "The option-critic architecture." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 1. Vol. 31.

T. P. Cannon and Ö. Simsek. 2024. "Hierarchical Orchestra of Policies." *arXiv preprint arXiv:2411.03008*.

R. T. Chen, J. Behrmann, D. Duvenaud, and J.-H. Jacobsen. 2019. "Residual flows for invertible generative modeling." *arXiv preprint arXiv:1906.02735*.

T. Degris, M. White, and R. S. Sutton. 2012. "Off-policy actor-critic." *arXiv preprint arXiv:1205.4839*.

L. Dinh, D. Krueger, and Y. Bengio. 2014. "Nice: Non-linear independent components estimation." *arXiv preprint arXiv:1410.8516*.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. 2016. "Density estimation using real nvp." *arXiv preprint arXiv:1605.08803*.

H. Du, C. Krause, V. Mikuni, B. Nachman, I. Pang, and D. Shih. 2025. "Unifying simulation and inference with normalizing flows." *Physical Review D*, 111, 7, 076004.

E. Erdogan. 2025. "Geometric Flow Models over Neural Network Weights." *arXiv preprint arXiv:2504.03710*.

S. Fujimoto, H. Hoof, and D. Meger. 2018. "Addressing function approximation error in actor-critic methods." In: *International Conference on Machine Learning*. PMLR, 1587–1596.

M. Germain, K. Gregor, I. Murray, and H. Larochelle. 2015. "Made: Masked autoencoder for distribution estimation." In: *International Conference on Machine Learning*. PMLR, 881–889.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. "Generative adversarial networks." *arXiv preprint arXiv:1406.2661*.

T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine. 2018. "Latent space policies for hierarchical reinforcement learning." In: *International Conference on Machine Learning*. PMLR, 1851–1860.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. 2018. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In: *International Conference on Machine Learning*. PMLR, 1861–1870.

G. E. Hinton and R. R. Salakhutdinov. 2006. "Reducing the dimensionality of data with neural networks." *science*, 313, 5786, 504–507.

J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel. 2019. "Flow++: Improving flow-based generative models with variational dequantization and architecture design." In: *International Conference on Machine Learning*. PMLR, 2722–2730.

D. P. Kingma and P. Dhariwal. 2018. "Glow: Generative flow with invertible 1x1 convolutions." *arXiv preprint arXiv:1807.03039*.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. 2016. "Improving variational inference with inverse autoregressive flow." *arXiv preprint arXiv:1606.04934*.

D. P. Kingma and M. Welling. 2013. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114*.

I. Kobyzev, S. J. Prince, and M. A. Brubaker. 2020. "Normalizing flows: An introduction and review of current methods." *IEEE transactions on pattern analysis and machine intelligence*, 43, 11, 3964–3979.

S. Kolouri, P. E. Pope, C. E. Martin, and G. K. Rohde. 2018. "Sliced-wasserstein autoencoder: An embarrassingly simple generative model." *arXiv preprint arXiv:1804.01947*.

A. Levy, G. Konidaris, R. Platt, and K. Saenko. 2017. "Learning multi-level hierarchies with hindsight." *arXiv preprint arXiv:1712.00948*.

A. Levy, R. Platt, and K. Saenko. 2017. "Hierarchical actor-critic." *arXiv preprint arXiv:1712.00948*, 12.

C. Li, B. Huggins, P. Mikkola, and L. Acerbi. 2025. "Normalizing Flow Regression for Bayesian Inference with Offline Likelihood Evaluations." *arXiv preprint arXiv:2504.11554*.

J. Li, C. Tang, M. Tomizuka, and W. Zhan. 2022. "Hierarchical planning through goal-conditioned offline reinforcement learning." *IEEE Robotics and Automation Letters*, 7, 4, 10216–10223.

Y. Li et al.. 2025. "Generative Models in Decision Making: A Survey." *arXiv preprint arXiv:2502.17100*.

B. Liao, X. Chen, J. Xu, J. Zhou, and H. Sun. 2025. "A novel Bayesian geophysical inversion method to address loss function bias: The iterative normalizing flows model." *Journal of Geophysical Research: Machine Learning and Computation*, 2, 1, e2024JH000479.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. 2015. "Continuous control with deep reinforcement learning." *arXiv preprint arXiv:1509.02971*.

Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. 2022. "Flow matching for generative modeling." *arXiv preprint arXiv:2210.02747*.

Q. Liu. 2022. "Rectified flow: A marginal preserving approach to optimal transport." *arXiv preprint arXiv:2209.14577*.

X. Ma, X. Kong, S. Zhang, and E. Hovy. 2020. "Decoupling Global and Local Representations from/for Image Generation." *arXiv preprint arXiv:2004.11820*.
X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley. 2017. "Least squares generative adversarial networks." In: *Proceedings of the IEEE international conference on computer vision*, 2794–2802.
V. Mnih, J. Agapiou, S. Osindero, A. Graves, O. Vinyals, K. Kavukcuoglu, et al.. 2016. "Strategic attentive writer for learning macro-actions." *arXiv preprint arXiv:1606.04695*.
V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. 2013. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602*.
O. Nachum, S. Gu, H. Lee, and S. Levine. 2018a. "Data-efficient hierarchical reinforcement learning." *arXiv preprint arXiv:1805.08296*.
O. Nachum, S. Gu, H. Lee, and S. Levine. 2018b. "Near-optimal representation learning for hierarchical reinforcement learning." *arXiv preprint arXiv:1810.01257*.
A. Odena, C. Olah, and J. Shlens. 2017. "Conditional image synthesis with auxiliary classifier gans." In: *International conference on machine learning*. PMLR, 2642–2651.
A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. 2016. "Wavenet: A generative model for raw audio." *arXiv preprint arXiv:1609.03499*.
T. Osa, V. Tangkaratt, and M. Sugiyama. 2019. "Hierarchical reinforcement learning via advantage-weighted information maximization." *arXiv preprint arXiv:1901.01365*.
G. Papamakarios, T. Pavlakou, and I. Murray. 2017. "Masked autoregressive flow for density estimation." *arXiv preprint arXiv:1705.07057*.
A. Radford, L. Metz, and S. Chintala. 2015. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434*.
J. Rafati and D. C. Noelle. 2019. "Learning representations in model-free hierarchical reinforcement learning." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 01. Vol. 33, 10009–10010.
T. Rainforth, A. Kosiorek, T. A. Le, C. Maddison, M. Igl, F. Wood, and Y. W. Teh. 2018. "Tighter variational bounds are not necessarily better." In: *International Conference on Machine Learning*. PMLR, 4277–4285.
J. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine. 2018. "Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings." In: *International Conference on Machine Learning*. PMLR, 1009–1018.
T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. 2016. "Improved techniques for training gans." *arXiv preprint arXiv:1606.03498*.
C. Schmidt, D. Gammelli, J. Harrison, M. Pavone, and F. Rodrigues. 2024. "Offline Hierarchical Reinforcement Learning via Inverse Optimization." *arXiv preprint arXiv:2410.07933*.
J. Su and G. Wu. 2018. "f-VAEs: Improve VAEs with conditional flows." *arXiv preprint arXiv:1809.05861*.
I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. 2017. "Wasserstein auto-encoders." *arXiv preprint arXiv: 1711.01558*.
A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu. 2016. "Pixel recurrent neural networks." In: *International Conference on Machine Learning*. PMLR, 1747–1756.
A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. 2017. "Feudal networks for hierarchical reinforcement learning." In: *International Conference on Machine Learning*. PMLR, 3540–3549.
L. Wang, Q. Zhang, Y. Yang, and H. Wang. 2025. "EAGLE: Contextual Point Cloud Generation via Adaptive Continuous Normalizing Flow with Self-Attention." *arXiv preprint arXiv:2503.13479*.
C. Xu, X. Cheng, and Y. Xie. 2022. "Normalizing flow neural networks by JKO scheme." *arXiv preprint arXiv:2212.14424*.

8 Appendices

Fig. 9 illustrates an example of creating a non-optimal goal space for the higher-level and lower-level policies of our model, starting from the default optimal goal space tuples of the higher-level and lower-level policies in HIRO.

Additionally, the training parameters for our model and the key points for the experiments are summarized as follows.

A The Training Parameters for Our Model

The training parameters defined in our model for all tasks are the same as those of HIRO, as follows:

- Discount γ = 0.99 for both controllers.
- Adam optimizer; actor learning rate 0.0001; critic learning rate 0.001.
- Soft update targets τ = 0.005 for both controllers.

- Replay buffer of size 200,000 for both controllers.
- Lower-level train step and target update performed every 1 environment step.
- Higher-level train step and target update performed every 10 environment steps.
- No gradient clipping.
- Reward scaling of 1.0 for lower-level; 0.1 for higher-level.
- Lower-level exploration is Gaussian noise with σ = 1.0.
- Higher-level exploration is Gaussian noise with σ = 1.0.
- state s_t : 30 dim.
- action $a_{t,z}$: 8 dim.
- Training step : 10M steps
- Higher-level goal space for ant_fall_(multi or single)
  - meta_context_range = ((-4, -4, 0), (12, 28, 5))
  - goal dim. in higher-level policy : 3 dim.
- Higher-level goal space for ant_push_(multi or single)
  - meta_context_range = ((-16, -4), (16, 20))
  - goal dim. in higher-level policy : 2 dim.
- lower-level goal space :
  - CONTEXT_RANGE_MIN = (-10, -10, -0.5, -1, -1, -1, -1, -0.5, -0.3, -0.5, -0.3, -0.5, -0.3, -0.5, -0.3)
  - CONTEXT_RANGE_MAX = (10, 10, 0.5, 1, 1, 1, 1, 0.5, 0.3, 0.5, 0.3, 0.5, 0.3, 0.5, 0.3)
  - goal dim. in lower-level policy : 15 dim.

B The Key Points for Experiments

The key points to note during the experiment are reiterated below:

- Although the dimensions of a_z and a_{rnvp} are fixed based on the environment and the input of FDGM, the dimension of $a_{t,z-\text{state}}$ is determined through a trade-off process involving experimentation to identify the optimal dimension.
- The layer dimensions of all policies of 'Our model_8_8' in Fig. 6 are the same as those in 'Our model_8_15'.
- In Fig. 7, the goal space of the higher-level policy for 'Our model_2_8' and 'Our model_3_8' is constrained to match the original goal space of Ant Push Single and Ant Fall Single in HIRO, respectively.
- The layer dimensions of all policies in our model in Fig. 7 are the same as those in our model in Fig. 6.
- The optimal size of $a_{t,z-\text{state}}$ for each experiment in Fig. 7 is selected through experimentation.
- Similar to Ant Fall Single in Fig. 6, our model in Ant Fall Single in Fig. 7 leverages the combination of the inverse operation from Algorithm 2 and the neural network size from Algorithm 1.
- Although the LSP framework does not inherently include off-policy correction, LSP in Fig. 7 incorporates off-policy correction to ensure a fair comparison in the experiment.

C The Comparison Table

Our proposed model and all reference models are evaluated based on two key criteria: (1) off-policy correction capability and (2) FDGM integration. In Table 2, 'O' indicates that a model satisfies a given criterion, while 'X' denotes its absence.

Table 2. Model comparison

	Off-policy correction	FDGM
Our model	O	O
HIRO	O	X
LSP	X	O
Ma et al.	X	O

Received 25 May 2025; accepted 14 May 2026