FLOWING FASTER TO COORDINATE: ONE-STEP ONLINE MULTI-AGENT FLOW POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent reinforcement learning (MARL) provides a powerful framework for learning coordinated behaviors through interactions with the environment. Developing MARL policies requires balancing expressive modeling of complex and multimodal action distributions with efficient training and execution. Generative policies, particularly diffusion-based policies, can faithfully capture complex and multimodal behaviors, but costly iterative sampling hinders their scalability in online multi-agent settings. We propose an **O**nline **MA**RL framework via **one-step F**low **model** (**OMAF**) that combines expressive generative policies with efficient one-step action generation. OMAF employs a Transformer-based flow policy to capture complex coordination behaviors, while its approximate path score surrogate provides a principled route to synchronized flow policy optimization. To enable stable and sample-efficient learning, we further develop a joint optimization scheme coupling softmax Q-value estimation with a joint flow policy objective for coordinated policy learning. By eliminating iterative sampling, OMAF dramatically reduces training overhead without sacrificing policy expressiveness. Extensive experiments across 10 standard tasks from MPE and MAMuJoCo show that OMAF consistently achieves superior performance, with up to $3.4\times$ higher returns and $10.5\times$ sample efficiency improvement compared with baseline methods. These results validate the effectiveness of OMAF as an expressive and computationally efficient one-step flow policy paradigm for online MARL.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) (Oliehoek et al., 2008) harnesses the power of collective intelligence, empowering agents to unlock capabilities beyond those of isolated learners. This collaborative paradigm has driven remarkable advances across diverse domains, from superhuman gameplay (Vinyals et al., 2019; Berner et al., 2019; Ye et al., 2020) to embodied coordination (Chen et al., 2025a; Guo et al., 2026), scientific discovery, e.g., mathematical theorem proving (Su et al., 2025; OpenAI, 2026), and LLM-based agentic systems (Zhao et al., 2026; Yang & Thomason, 2026). Realizing such collective capabilities requires policies that are both sample-efficient and expressive enough to capture diverse coordination behaviors (Lin & Lee, 2026; Li et al., 2026d).

Generative policies, particularly diffusion- and flow-based policies (Ho et al., 2020; Lipman et al., 2023), provide a powerful foundation for expressive decision-making, naturally capturing complex and multimodal action distributions and diverse, coordinated behaviors beyond conventional unimodal policies (Wang et al., 2023; Zhang et al., 2026b). Such advantage has translated into strong performance with diffusion policies (Ma et al., 2025; Celik et al., 2025) in single-agent online RL, but their iterative sampling remain computationally costly. One-step flow policies overcome this bottleneck with single-step action generation while retaining the benefits of generative modeling (Zhan et al., 2026; Dong et al., 2026).

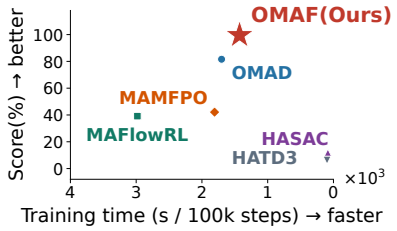


Figure 1: Superior performance and improved training efficiency of OMAF over 10 standard tasks.

Yet, efficient generative policies remain underexplored in multi-agent RL, despite being particularly important in this setting. The exponentially larger joint state-action spaces of multiple interacting

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

agents demand expressive policies while amplifying the cost of iterative generation as the number of agents grows (Lee et al., 2026a; Li et al., 2026d). Meanwhile, limited interaction data and complex coordination further require high sample efficiency. More critically, few-step flow generation faces a trade-off between efficiency and policy expressiveness, making it challenging in multi-agent settings. Developing expressive generative policies that retain strong coordination performance with efficient one-step generation therefore remains an open challenge for online MARL.

To tackle these challenges, we propose **O**nline **MA**RL via **one-step F**low model (**OMAF**), a framework specifically designed for efficient and expressive policy learning in online MARL. OMAF introduces a Transformer-based one-step flow policy for efficient generative policy learning, retaining the expressive coordination while eliminating iterative denoising. We further develop an approximate path score surrogate and synchronized policy optimization with a joint softmax Q-value (Pan et al., 2020) for stable and sample-efficient training from scratch. Together, these designs jointly address the efficiency–performance challenge of generative policy learning in online MARL. As illustrated in Figure 1, OMAF simultaneously achieves state-of-the-art performance and training efficiency across multi-agent environments. Our contributions are as follows:

- We propose **OMAF**, an efficient one-step flow policy for online MARL under the centralized training and decentralized execution (CTDE) framework. The policy incorporates a Transformer-based velocity network to generate continuous actions through a single denoising step, retaining expressive multi-agent coordination without iterative sampling.
- We develop a stable joint optimization scheme to effectively train the one-step flow policy. Softmax Q-value estimation provides smooth and reliable value guidance, while synchronized joint flow policy optimization coordinates policy updates across agents and promotes effective multi-agent coordination. Together, these components facilitate stable and sample-efficient learning of expressive one-step flow policies.
- We conduct extensive experiments across 10 standard online MARL environments from Multi-Agent Particle Environments (MPE) and Multi-Agent MuJoCo (MAMuJoCo). OMAF consistently achieves the best performance, with up to $3.4\times$ higher returns and $10.5\times$ greater sample efficiency than baseline methods, while substantially reducing computational overhead by eliminating iterative denoising.

2 Related Work

In this section, we discuss representative works on online MARL and flow-based RL policies, and defer a more comprehensive review to Appendix A.

2.1 Online Multi-Agent Reinforcement Learning

Online MARL has advanced substantially under centralized training with decentralized execution (CTDE) (Oliehoek et al., 2008; Matignon et al., 2012), with representative methods such as MADDPG, MAPPO, VDN, QMIX, and QPLEX (Lowe et al., 2017; Yu et al., 2022; Sunehag et al., 2018; Rashid et al., 2018; Wang et al., 2021) improving coordination through centralized critics and value factorization. Recent studies further enhance factorization, exploration, robustness, scalability, and heterogeneous-agent coordination (Huang et al., 2026; Xu et al., 2026; Qu et al., 2026; Pan et al., 2021; Li et al., 2023; Zhong et al., 2024), while decentralized methods address scalability and non-stationarity (Tampuu et al., 2017; de Witt et al., 2020; Li et al., 2026a; Meulemans et al., 2025). Recent works introduce generative policies, including HAQO (Lin & Lee, 2026), OMAD (Li et al., 2026d) and DDPL (Zhang et al., 2026c), to improve policy expressiveness and exploration.

2.2 Flow Policies in Reinforcement Learning

Flow-based generative policies offer efficient alternatives to diffusion by learning velocity fields that transport noise to actions along straighter trajectories (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023; Geng et al., 2025). In RL, existing works study flow models for value estimation, e.g., flowq (Agrawalla et al., 2026a;b), and expressive policies, including ReinFlow (Zhang et al., 2025b), SAC-Flow (Zhang et al., 2026b), FlowQ (Alles et al., 2025), QIPO (Zhang et al., 2025a), and FQL (Park et al., 2025), as well as efficient one-step generation through SSCP (Koirala

& Fleming, 2026), OFQL (Nguyen & Yoo, 2026), and MVP (Zhan et al., 2026). Recent methods further optimize flow policies via policy gradients, including FPO (McAllister et al., 2026), OSMFPG (Wang et al., 2026), LaMPO (Liu et al., 2026), FlowRL (Lyu et al., 2026), RFO (Zhong et al., 2026), MFPO (Dong et al., 2026), and FPMD (Chen et al., 2025b). For MARL, MAC-Flow (Lee et al., 2026a) and OM^{2}P (Li et al., 2026c) consider one-step flow policies only in offline learning. We address the missing setting of direct one-step flow optimization in online MARL.

3 Background

3.1 Multi-Agent Reinforcement Learning and Efficient Value Estimation

We formulate cooperative MARL as a decentralized partially observable MDP (Dec-POMDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho \rangle$ (Oliehoek & Amato, 2016) with global states, joint actions, transition dynamics, shared rewards, and initial state distribution. The transition function $\mathcal{P}$ specifies the transition dynamics, while each agent i samples $a_t^i \sim \pi_i(\cdot|s_t^i)$ based on s_t^i, inducing the factorized joint policy $\boldsymbol{\pi}(a|s) = \prod_{i=1}^N \pi_i(a^i|s^i)$. Under the shared reward $r_t = \mathcal{R}(s_t, a_t)$, the objective is to maximize $J(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where $\gamma \in [0, 1)$ is the discount factor. To encourage exploration in online MARL, Maximum-entropy RL (Ziebart et al., 2008; Haarnoja et al., 2018) augments the objective with an entropy regularizer to encourage exploration: $J_{\text{ent}}(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t \left(r_t + \alpha \mathcal{H}(\boldsymbol{\pi}(\cdot|s_t))\right)\right]$, where α controls the exploration-exploitation trade-off. The corresponding soft Bellman residual to optimize Q-function Q_ϕ is

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\phi(s,a) - r - \gamma\left(\min_{k=1,2} Q_{\tilde{\phi}_k}(s',a') - \alpha \sum_{i=1}^{N} \log \pi_i(a_i'|s')\right)\right)^2\right], \quad (1)$$

and the joint policy $\boldsymbol{\pi}_\theta$ is updated via

$$L_{\boldsymbol{\pi}}(\theta) = \mathbb{E}_{s\sim\mathcal{D}, \mathbf{a}\sim\pi_\theta(\cdot|s)}[\alpha \log \boldsymbol{\pi}_\theta(\cdot|s) - Q_\phi(s, \mathbf{a})]. \quad (2)$$

The softmax Q-value (Haarnoja et al., 2017; Pan et al., 2020) provides a smooth alternative to the hard maximum by weighting actions according to their exponentiated Q-values, and its estimation via importance sampling is written as

$$Q_{\text{soft}}(s', \pi) = \mathbb{E}_{a'\sim\pi}\left[\frac{\exp\left(\beta Q_{\tilde{\phi}}(s',a')\right) Q_{\tilde{\phi}}(s',a')}{\pi(a'|s')}\right] / \mathbb{E}_{a'\sim\pi}\left[\frac{\exp\left(\beta Q_{\tilde{\phi}}(s',a')\right)}{\pi(a'|s')}\right]. \quad (3)$$

3.2 Flow Matching

Flow matching (Lipman et al., 2023; Li et al., 2026b;e) learns a continuous transport process that maps samples from a tractable source distribution to a desired data distribution. Let $x_0 \sim p(x)$ denote a data sample and $x_1 \sim \mathcal{N}(0, I_d)$ a source sample. A time-dependent path x_t connects the two distributions over $t \in [0, 1]$, with its instantaneous velocity given by $v(t, x_t) = \frac{\mathrm{d}x_t}{\mathrm{d}t}$. Flow matching learns a parameterized velocity field $v_\theta(t, x_t)$ by minimizing

$$\min_\theta \ \mathbb{E}_{x_0,x_1,t}\left[\|v_\theta(t,x_t) - v(t,x_t)\|_2^2\right],$$

where $t \sim U([0, 1])$. Once trained, the learned flow generates samples by solving the corresponding ordinary differential equation (ODE) $x_t = x_r + \int_r^t v_\theta(\tau, x_\tau)\,\mathrm{d}\tau$, which can be discretized using the first-order update:

$$x_{t-\Delta t} = x_t - v_\theta(t, x_t)\Delta t\,. \quad (4)$$

4 Theoretical Bridge to OMAF

While flow matching enables efficient generative modeling through continuous transport, applying flow-based policies to online MARL still presents several challenges.

First, entropy regularization is hindered by intractable policy likelihoods. In contrast to Gaussian policies, flow-based policies involve densities induced by multi-step generative processes, making likelihood evaluation difficult (Lipman et al., 2023). This complicates entropy-based exploration in continuous multi-agent action spaces (Zhong et al., 2024).

Second, flow policies face a fundamental conflict between joint expressiveness and decentralized execution under CTDE. Joint flows capture inter-agent dependencies but couple agents, while independent flows preserve decentralization at the cost of essential coordination (Zhu et al., 2024).

Finally, one-step generation exposes an intrinsic mismatch between expressive flow policies and online reward optimization. Aggressive step reduction can degrade action quality due to transport errors (Li et al., 2026d), while RL must reshape the flow toward high-value joint actions (Pan et al., 2021; Li et al., 2025b). Efficiently reconciling expressive one-step generation with global coordination remains an open challenge in online MARL.

These challenges motivate our design of Online MARL via one-step Flow model (**OMAF**). We next provide theoretical guarantees and insights to characterize the proposed one-step flow policy and its optimization, offering a principled understanding of its expressiveness, value estimation stability, and coordination capability.

4.1 Theoretical Guarantees

We provide population-level theoretical guarantees for OMAF under practical approximations. We first establish convergence of flow policy optimization, showing that the learned policy approaches the soft-optimal decentralized policy. We then bound the approximation error induced by collapsing the flow trajectory into a single Euler step, providing a theoretical basis for one-step generation. Proofs are deferred to Appendix B.

Theorem 1 (Approximate convergence to the soft-optimal decentralized policy)**.** *Let $\rho(\boldsymbol{s})$ be the population observation distribution and let $\boldsymbol{\pi}^\star$ be the factorized maximum-entropy optimal policy induced by $Q^\star$. Suppose that, uniformly over the flow-policy class,*

$$\left|\mathcal{L}_\pi(\boldsymbol{\theta}) - \overline{\mathcal{L}}_\pi(\boldsymbol{\pi}_{\boldsymbol{\theta}})\right| \le \varepsilon_{\mathrm{obj}} \coloneqq \varepsilon_Q + \alpha\varepsilon_\ell,$$

where ε_Q bounds the critic error and ε_ℓ bounds the discrepancy between the path score surrogate and the exact joint log-density. Assume that the policy class has approximation error

$$\inf_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{s}\sim\rho}[\mathrm{KL}(\boldsymbol{\pi}_{\boldsymbol{\theta}} \,\|\, \boldsymbol{\pi}^\star)] \le \varepsilon_{\mathrm{app}}.$$

For an iterate $\boldsymbol{\theta}_k$, define $\varepsilon_{\mathrm{opt},k} \coloneqq \mathcal{L}_\pi(\boldsymbol{\theta}_k) - \inf_{\boldsymbol{\theta}} \mathcal{L}_\pi(\boldsymbol{\theta})$. Then,

$$\mathbb{E}_{\boldsymbol{s}\sim\rho}[\mathrm{KL}(\boldsymbol{\pi}_{\boldsymbol{\theta}_k} \,\|\, \boldsymbol{\pi}^\star)] \le \varepsilon_{\mathrm{app}} + \frac{\varepsilon_{\mathrm{opt},k} + 2\varepsilon_{\mathrm{obj}}}{\alpha}. \tag{5}$$

Consequently, $\boldsymbol{\pi}^\star$ is the unique policy-space minimizer of $\overline{\mathcal{L}}_\pi$ when all three errors vanish. Moreover, if $\mathcal{L}_\pi$ is L-smooth, satisfies a local Polyak–Łojasiewicz inequality with constant $\mu > 0$, and its unbiased stochastic gradient has variance at most σ_g^2, then for $0 < \eta \le 1/L$,

$$\mathbb{E}[\varepsilon_{\mathrm{opt},k}] \le (1-\eta\mu)^k \varepsilon_{\mathrm{opt},0} + \frac{L\eta\sigma_g^2}{2\mu}. \tag{6}$$

This theorem establishes that the flow policy converges to the soft-optimal policy up to approximation and optimization errors. The next result bounds the single-step Euler approximation error of the stochastic flow policy, where $z \sim \mathcal{N}(0, I)$ is the joint base noise and $a_t^\star$ denotes the optimal latent flow characteristic induced by the optimal policy $\boldsymbol{\pi}^*$.

Theorem 2 (One-step near-optimality of the stochastic flow policy)**.** *Let $v^\star$ be the velocity field of the optimal latent flow, with material acceleration $D_t v^\star \coloneqq \partial_t v^\star + v^\star \nabla_a v^\star$. Assume that the target flow has bounded temporal variation and that the learned initial velocity satisfies bounded mean approximation and stochastic errors:*

$$\begin{aligned}
\left(\mathbb{E}_{\boldsymbol{s},z}\left[\|D_t v^\star(a_t^\star, t \mid \boldsymbol{s})\|_2^2\right]\right)^{1/2} &\le \kappa, \qquad t \in [0,1], \\
\left(\mathbb{E}_{\boldsymbol{s},z}\left[\|\mu_{\boldsymbol{\theta}}(z,\boldsymbol{s},0) - v^\star(z,0 \mid \boldsymbol{s})\|_2^2\right]\right)^{1/2} &\le \varepsilon_\mu, \\
\left(\mathbb{E}_{\boldsymbol{s},z,\boldsymbol{\xi}}\left[\|\sigma_{\boldsymbol{\theta}}(z,\boldsymbol{s},0) \odot \boldsymbol{\xi}\|_2^2\right]\right)^{1/2} &\le \varepsilon_\sigma, \qquad \boldsymbol{\xi} \sim \mathcal{N}(0, I).
\end{aligned}$$

Let $\boldsymbol{\pi}_{\boldsymbol{\theta}}^{(1)}$ *be induced by the stochastic one-step action*

$$\widehat{\boldsymbol{a}} = \tanh(z + \mu_{\boldsymbol{\theta}}(z, \boldsymbol{s}, 0) + \sigma_{\boldsymbol{\theta}}(z, \boldsymbol{s}, 0) \odot \boldsymbol{\xi}) .$$

Then, with $\Delta_1 := \varepsilon_\mu + \varepsilon_\sigma + \kappa/2$,

$$\left(\mathbb{E}_{\boldsymbol{s}\sim\rho}\left[W_2^2\left(\boldsymbol{\pi}_{\boldsymbol{\theta}}^{(1)}(\cdot \mid \boldsymbol{s}), \boldsymbol{\pi}^\star(\cdot \mid \boldsymbol{s})\right)\right]\right)^{1/2} \le \Delta_1. \tag{7}$$

If $Q^\star(\boldsymbol{s}, \boldsymbol{a})$ *is* K_Q*-Lipschitz in the joint action and the discounted observation occupancy of* $\boldsymbol{\pi}_{\boldsymbol{\theta}}^{(1)}$ *is dominated by* ρ *with constant* C_ρ*, then*

$$J(\boldsymbol{\pi}^\star) - J(\boldsymbol{\pi}_{\boldsymbol{\theta}}^{(1)}) \le \frac{C_\rho K_Q}{1-\gamma}\Delta_1. \tag{8}$$

Thus, exact one-step recovery follows in the special case $\varepsilon_\mu = \varepsilon_\sigma = \kappa = 0$*; otherwise, the performance gap is controlled by the velocity-estimation error, the injected velocity noise, and the curvature of the optimal flow trajectory.*

This theorem establishes the near-optimality of one-step flow policies under bounded approximation errors. Together, the two theorems underscore the critical role of accurate value estimation and one-step action generation, motivating OMAF as a principled framework specifically designed to enable efficient one-step flow policy learning.

5 The Online MARL via One-step Flow Model Method

This section introduces our flow policy algorithm for online multi-agent reinforcement learning, with an overview shown in Figure 2. To enable efficient one-step flow policy learning in online MARL, OMAF incorporates two key designs: (i) a Transformer-based velocity network that directly parameterizes the flow policy, enabling expressive policy modeling and one-step action generation without iterative sampling or policy distillation; and (ii) a stable joint optimization scheme that combines softmax Q-value estimation with joint policy optimization, providing reliable value guidance while facilitating coordinated policy learning. We detail these components below.

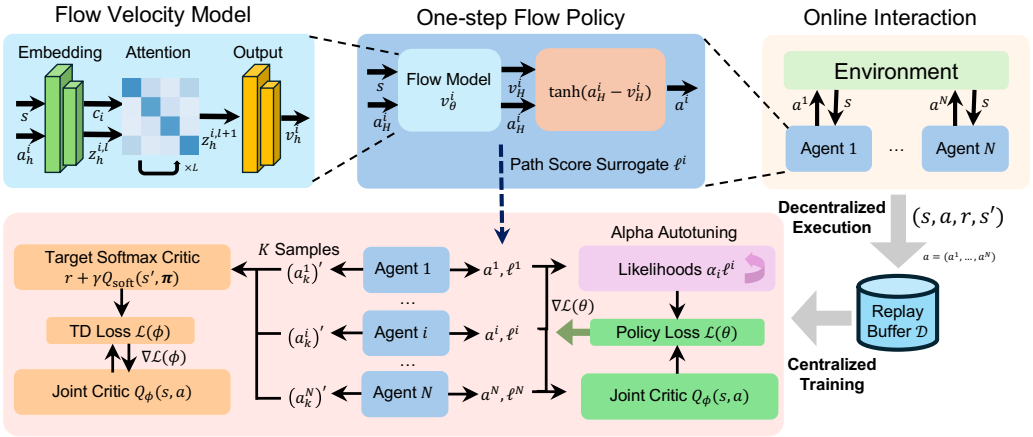


Figure 2: The CTDE framework of OMAF. Each agent generates a one-step flow action a^i and path score surrogate ℓ^i through embedding attention (up), while jointly optimizes the flow policies (down) with a shared critic $Q_\phi(s, a)$ and synchronized policy loss $\mathcal{L}(\theta)$.

OMAF establishes a powerful online flow-based MARL framework that integrates expressive flow policy, efficient one-step generation, and effective multi-agent coordination. While existing MARL methods predominantly use Gaussian policies, e.g., HARL (Zhong et al., 2024), OMAF adopts a powerful flow policy to capture expressiveness and complicated action distributions. Compared with diffusion-based methods such as OMAD (Li et al., 2026d), OMAF replaces costly diffusion-driven iterative denoising with efficient one-step flow generation, achieving expressive policy modeling without sacrificing generation efficiency.

5.1 Decentralized Flow Policy Formulation

To extend flow-based policies to multi-agent domains, we employ a factorized policy $\boldsymbol{\pi}_\theta(a|s) = \prod_{i=1}^{N} \pi_{\theta_i}(a^i|s)$ (Zhong et al., 2024), enabling decentralized execution while preserving policy expressiveness. For each agent i, we parameterize the velocity field $v_{\theta_i}(a^i_\tau, s, \tau)$ with a Transformer-based network (Vaswani et al., 2017) (up in Figure 2). Given $a^i_1 \sim \mathcal{N}(0, I)$, the flow trajectory follows $\frac{\mathrm{d}a^i_\tau}{\mathrm{d}\tau} = v_{\theta_i}(a^i_\tau, s, \tau)$, where $\tau \in [0, 1]$, and is discretized via the first-order Euler update:

$$a^i_{\tau-\Delta\tau} = a^i_\tau - v_{\theta_i}(a^i_\tau, s, \tau)\Delta\tau. \tag{9}$$

Finally, the output action is $a^i = \tanh(a^i_0)$. In OMAF, we use a single Euler step to directly generate the action, i.e., $a^i = \tanh(a^i_1 - v_{\theta_i}(a^i_1, s, 1))$.

Since the exact likelihood is intractable, we approximate it using a path score surrogate (Zhang et al., 2026b) for policy optimization (see Appendix C for details). Denote the path density as $p_{\theta_i}(a_{\tau^i_0}, \cdots, a_{\tau^i_H})$ for H-step discretization, the path score surrogate ℓ^i is defined as

$$\ell^i \coloneqq p_{\theta_i}(a^i|s) = \int \cdots \int p_{\theta_i}(a_{\tau^i_0}, \cdots, a_{\tau^i_H}) \mathrm{d}a^i_{\tau_0} \cdots \mathrm{d}a^i_{\tau_{H-1}}. \tag{10}$$

The direct integral is difficult to evaluate, so we employ a tractable approximation to its exact value, with the formulation and analysis detailed in Appendix C. This enables efficient policy optimization while retaining expressive flow-based modeling and one-step action generation.

5.2 Efficient Online Centralized Training of the Flow Policy

We next formulate the centralized training objective for OMAF, where the flow policy is optimized jointly across agents with entropy regularization to encourage both high-value actions and sufficient exploration (down in Figure 2).

Softmax Q-value Estimation. The efficient training of the one-step flow policy places a stronger demand on accurate Q-value estimation, as the critic directly guides policy updates. To obtain more reliable Q-value estimates, based on Equation 1 and Equation 3, we introduce the centralized critic using the CrossQ (Bhatt et al., 2024) and the softmax Bellman operator (Pan et al., 2020)

$$\mathcal{L}_Q(\phi) = \mathbb{E}\left[\left(r + \gamma Q_{\mathrm{soft}}(s', \boldsymbol{\pi}) - Q_\phi(s, a)\right)^2\right], \tag{11}$$

where Q_{soft} is approximated using K joint-action samples $\{a'_k\}_{k=1}^K$ generated by the flow policies:

$$Q_{\mathrm{soft}}(s', \boldsymbol{\pi}) \approx \sum_{k=1}^{K} w_k \left(q'_k - e'_k\right).$$

Specifically, these terms are defined as

$$q'_k = \min_{c\in\{1,2\}} Q_{\phi_c}(s', a'_k), \qquad e'_k = \sum_{i=1}^{N} \alpha_i \ell^i_k, \qquad w_k = \frac{\exp(\beta q'_k)}{\sum_{l=1}^{K} \exp(\beta q'_l)}.$$

Here q'_k denotes the original Q-value of the k-th joint-action sample, e'_k denotes the aggregated path score ℓ^i_k across agents via the weights α_i, and w_k denotes the β-softmax weight assigned to the k-th sample based on its Q-value. Together, softmax Q-value estimation and the path score provide complementary guidance for one-step flow learning: accurate Q-values enable effective policy improvement with few steps, while the path score facilitates efficient optimization and exploration.

Synchronized flow policy optimization. To facilitate effective coordination, based on Equation 2, we synchronously optimize all flow policies under a centralized objective:

$$\mathcal{L}_{\boldsymbol{\pi}}(\boldsymbol{\theta}) = \mathbb{E}_{s\sim\mathcal{D}, a_i\sim\boldsymbol{\pi}_{\theta_i}(\cdot|s)}\left[\sum_{i=1}^{N} \alpha_i \ell^i - \min_{c\in\{1,2\}} Q_{\phi_c}(s, a)\right]. \tag{12}$$

Here $a = (a_1, \cdots, a_N)$ denotes the joint action sampled from all flow policies π_{θ_i}. Jointly optimizing the policies with the centralized Q-function accounts for inter-agent behaviors, promoting coordinated policy improvement. Meanwhile, tractable path score surrogate enables direct evaluation and approximated maximum-entropy optimization compared with diffusion policies.

Agent-wise temperature auto-tuning. To provide fine-grained exploration control for each agent, we adaptively tune an individual entropy coefficient α_i. Denote $\bar{\ell}^i := \mathbb{E}[\ell^i]$ as the expected path score surrogate of agent i. The temperature is optimized by

$$\mathcal{L}(\alpha) = \frac{1}{N}\sum_{i=1}^{N} \alpha_i \left(\mathcal{H}^i_{\text{target}} - \text{sg}(\bar{\ell}^i)\right), \tag{13}$$

where the expected path score surrogate is detached from the temperature update. This adaptive scheme automatically adjusts α_i to maintain the desired exploration level during training.

Overall, online MARL faces severe policy instability as the joint state distribution evolves rapidly, making expressive policy modeling and robust value guidance crucial under limited samples. The flow policy provides expressive one-step parameterization, while the path score enables efficient policy optimization and exploration. Building on this foundation, the softmax Q-value objective provides concise and effective guidance against distributional shift, the synchronized policy objective promotes coordinated improvement across agents, and the agent-specific temperature objectives enable fine-grained exploration control. Together, these components form a unified framework for efficient and stable one-step flow policy learning in online MARL.

5.3 OMAF Algorithm

Algorithm 1: Online Multi-Agent Flow Policy (OMAF)

1 **Initialize:** Critic networks $Q_{\phi,1}$, $Q_{\phi,2}$, target networks $Q_{\bar{\phi},1}, Q_{\bar{\phi},2}$, flow policy π_{θ_i} for agents $i = 1, 2, \ldots, N$, temperature $\{\alpha_i\}_{i=1}^N$, policy delay d_l, target entropy $\{\mathcal{H}^i_{\text{target}}\}_{i=1}^N$, replay buffer $\mathcal{D}$, and the threshold buffer size L_{init}. // Initialization
2 **for** $m = 1$ **to** M *steps* **do**
3 **if** *Buffer length* $L_{\mathcal{D}} < L_{\text{init}}$ **then**
4 Sample a trajectory (s_t, a_t, r_t, s_{t+1}) using the random policy and insert it into $\mathcal{D}$.
5 **else**
6 Sample a trajectory (s_t, a_t, r_t, s_{t+1}) using π_{θ_i} and insert it into $\mathcal{D}$.
7 // Trajectory Generation
8 Sample a mini-batch $B = \{(s, a, r, s')\}$ from $\mathcal{D}$. // Sampling
9 Sample K candidates $a'_k = ((a^1_k)', \ldots, (a^N_k)')$ using π_{θ_i}, calculate $\mathcal{L}(\phi)$ using Equation 11 and optimize ϕ. // Critic Optimization
10 Update target networks via $\bar{\phi}_i \leftarrow \rho\bar{\phi}_i + (1-\rho)\phi_i$. // Target Network Optimization
11 **if** $m \bmod d_l = 0$ **then**
12 Sample $a = (a^1, \ldots, a^N)$ using π_{θ_i}, and calculate $(\ell_{\pi_{\theta_1}}, \ldots, \ell_{\pi_{\theta_N}})$ simultaneously, calculate $\mathcal{L}(\theta)$ using Equation 12 and jointly optimize $\{\theta_i\}_{i=1}^N$. // Synchronized Flow Policy Optimization
13 Optimize the temperature α_i via minimizing Equation 13. // Temperature Optimization

The training procedure of the proposed Online Multi-Agent Flow Policy (OMAF) is summarized in Algorithm 1. Line 1 initializes the model parameters and replay buffer. Lines 4- 6 collect trajectories using random policies at the beginning and the learned flow policies thereafter. Once the replay buffer is sufficiently populated, Line 9 updates the centralized critics using K joint actions sampled from the current flow policies and the softmax Q-value objective, followed by a soft update of the target networks in Line 10. Every d_l steps, Line 12 jointly optimizes all flow policies using the centralized Q-function, while Line 13 tunes the agent-wise entropy temperatures.

Overall, OMAF establishes a pioneering flow-based approach to online MARL that jointly addresses the expressiveness-efficiency trade-off in generative policy learning. Its one-step flow policy models complex action distributions without the iterative sampling required by diffusion policies, while the path score surrogate enables effective maximum-entropy optimization and captures complex multi-agent coordination. A detailed discussion of the baseline algorithms is provided in Appendix C.4. We next conduct extensive experiments to evaluate whether these designs translate into improved policy performance and training efficiency across a diverse set of multi-agent tasks.

6 EXPERIMENTS

We evaluate OMAF on diverse continuous-control multi-agent benchmarks and compare it with state-of-the-art baselines in terms of performance and efficiency. We further visualize the learned policies to examine their expressiveness and coordination capability from different perspectives. Finally, we conduct ablation studies to evaluate the contribution of each key component of OMAF.

6.1 EXPERIMENTAL SETUP

Enviroments. We benchmark OMAF on two widely adopted continuous-control MARL suites: Multi-Agent Particle Environments (MPE) (Lowe et al., 2017) and Multi-Agent MuJoCo (MAMuJoCo) (Peng et al., 2021). We use Cooperative Navigation in MPE, and HalfCheetah, Ant, Walker, Hopper, and Swimmer in MAMuJoCo, covering particle coordination and high-dimensional locomotion tasks. Detailed environment configurations are provided in Appendix D.1.

Baselines. We consider four representative online MARL baselines for comparison. HATD3 (Zhong et al., 2024) and HASAC (Liu et al., 2024) are two widely used actor-critic methods for online MARL. OMAD (Li et al., 2026d) represents the state-of-the-art multi-agent diffusion policy, while MAFlowRL (Lyu et al., 2026) and MAMFPO (Dong et al., 2026) extend flow-based RL methods to the multi-agent setting. All methods are trained with 3 random seeds, and we report the mean performance and standard deviation over 10 evaluation episodes. Implementation details, including hyperparameters and network architectures, are provided in Appendix D.2.

6.2 EXPERIMENT RESULTS

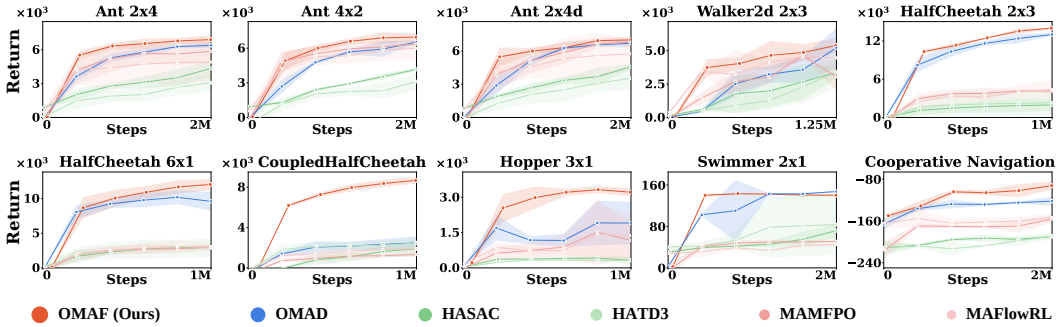


Figure 3: Performance comparison of OMAF against representative online MARL methods HATD3 and HASAC, diffusion-based OMAD, and extensions of flow-based methods MAFlowRL and MAMFPO on the MPE and MAMuJoCo benchmarks. Curves show average episode return over training steps across 3 random seeds, with shaded regions denoting one standard deviation. OMAF achieves superior performance, validating one-step flow policies for online MARL.

The overall performance of OMAF is presented in Figure 3. We evaluate all methods under a limited training budget of fewer than 2×10^6 environment interaction steps to assess both performance and sample efficiency. Across all 10 MPE and MAMuJoCo tasks, OMAF consistently achieves superior performance compared with the baseline algorithms. This advantage holds for both the MPE tasks with complex coordination and the higher-dimensional MAMuJoCo tasks, demonstrating that OMAF can achieve strong performance with substantially fewer environment interactions. Besides, OMAF achieves up to a $10.5\times$ improvement in sample efficiency, reaching baseline-level performance with substantially fewer environment interactions.

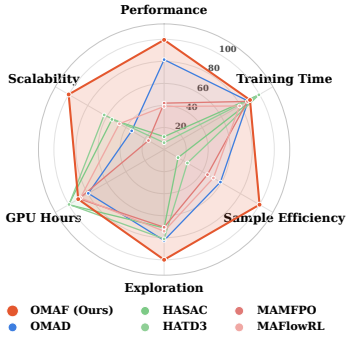


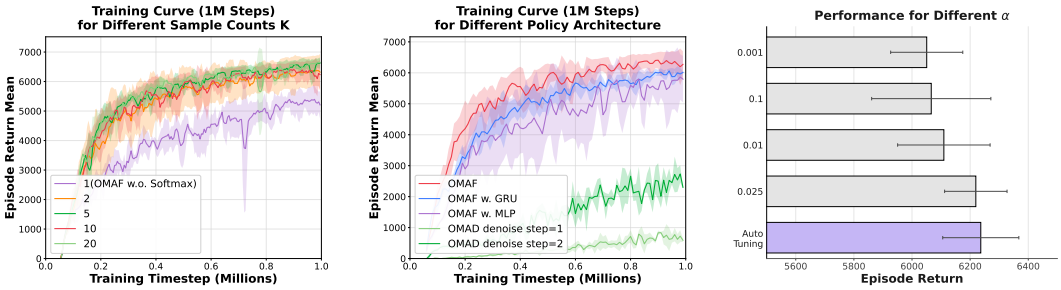
Figure 4: OMAF achieves superior performance across different metrics shown in the radar chart.

Compared with Gaussian-policy methods, including HASAC and HATD3 (Liu et al., 2024; Zhong et al., 2024), OMAF achieves superior performance, demonstrating the advantage of expressive flow-based policies. In contrast to diffusion-based OMAD (Li et al., 2026d), OMAF achieves better performance without

iterative denoising, balancing policy expressiveness and generation efficiency. Against representative multi-agent flow-policy methods, MAFlowRL (Lyu et al., 2026) and MAMFPO (Dong et al., 2026), OMAF consistently performs better, highlighting the effectiveness of our specific flow-policy design and synchronized value and policy optimization.

As shown in Figure 4 (details refer to Appendix D), OMAF achieves superior and well-rounded performance across multiple metrics, outperforming representative baselines algorithms. OMAF consistently performs well across task performance, training and sample efficiency, exploration, and scalability, achieving stronger policies with substantially lower computational resource requirements than diffusion-based OMAD (Li et al., 2026d). Such computational efficiency is crucial for practical and scalable online MARL, where training and inference cost can become a major bottleneck.

6.3 Ablation Study



(a) Sample count K (b) Policy architecture (c) Coefficient term α_i

Figure 5: Ablation studies of OMAF on the Ant 2×4 task.

To evaluate the individual contributions of key components in OMAF, we conduct comprehensive ablation studies on the Ant 2×4 task (Figure 5) about the value estimation, policy architecture and coefficient term sensitivity. More results are shown in Appendix D.

Impact of Sample Count K in Softmax Q (Figure 5a). Setting $K = 1$ (equivalent to OMAF without Softmax Q) leads to a substantial performance degradation, validating the necessity of softmax target value estimation. Selecting an appropriate sample count K significantly stabilizes learning and boosts overall return. However, an excessively large K introduces redundant action sampling, which slightly reduces training efficiency.

Policy Architecture and One-Step Efficiency Figure 5b). OMAF utilizing a Transformer backbone consistently outperforms variants based on GRU and MLP, demonstrating that architectural choice plays a critical role in final performance. Furthermore, when reducing the denoising steps of OMAD (denoise step 1 or 2), its performance collapses significantly. This highlights OMAF's superior capability to achieve highly efficient, one-step training without compromising policy quality.

Temperature Coefficient α Auto-Tuning (Figure 5c): Dynamically auto-tuning α yields higher returns than fixed coefficient settings (e.g., 0.001, 0.01, 0.025, and 0.1). This confirms that the auto-tuning scheme serves as an effective mechanism to dynamically govern policy exploration.

7 Conclusion

We present OMAF, an efficient one-step flow policy framework for online multi-agent reinforcement learning. By integrating expressive flow-based policy modeling with synchronized joint policy optimization and softmax Q-value estimation, OMAF enables efficient action generation while effectively capturing multi-agent coordination. Experiments on MPE and MAMuJoCo demonstrate that OMAF achieves superior performance against strong online diffusion-based and flow-based baselines, while requiring only a single flow step for action generation. These results highlight the potential of one-step flow policies as an efficient and expressive approach for online MARL.

AI USE STATEMENT

During the preparation of this work, we used generative AI tools, specifically ChatGPT, to assist with proofreading, language polishing, and improving the clarity and readability of the manuscript. We did not use generative AI tools for research ideation, methodology development, experimental design, implementation, data generation or analysis, or the generation of research results.

All AI-assisted text was carefully reviewed and revised by the authors to ensure its accuracy, clarity, and consistency with the underlying research. The authors take full responsibility for the final content of this work, including all claims, analyses, and artifacts.

REFERENCES

Johannes Ackermann, Volker Gabler, Takayuki Osa, and Masashi Sugiyama. Reducing overestimation bias in multi-agent domains using double centralized critics. *Advances in Neural Information Processing Systems: Deep Reinforcement Learning Workshop*, 2019.

Bhavya Kumar Agrawalla, Michal Nauman, Khush Agrawal, and Aviral Kumar. floq: Training critics via flow-matching for scaling compute in value-based RL. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=m14YNdmPAh.

Bhavya Kumar Agrawalla, Michal Nauman, and Aviral Kumar. What does flow-matching bring to TD-learning? In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=e7EKA7Y9Im.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023.

Marvin Alles, Nutan Chen, Patrick van der Smagt, and Botond Cseke. Flowq: Energy-guided flow policies for offline reinforcement learning, 2025. URL https://arxiv.org/abs/2505.14139.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal JĂłzefowicz, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *The Twelfth International Conference on Learning Representations*, volume 2024, pp. 55293–55311, 2024.

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

Junhong Chen, Ziqi Yang, Haoyuan G Xu, Dandan Zhang, and George Mylonas. Multi-agent systems for robotic autonomy with llms. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 4194–4204, 2025a.

Tianyi Chen, Haitong Ma, Na Li, Kai Wang, and Bo Dai. One-step flow policy mirror descent. *arXiv preprint arXiv:2507.23675*, 2025b.

Yangkun Chen, Kai Yang, Jian Tao, and Jiafei Lyu. Novelty-guided data reuse for efficient and diversified multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 15930–15938, 2025c.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Xiaoyi Dong, Xi Sheryl Zhang, and Jian Cheng. Mean flow policy optimization. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=88H3V90Wzo.

Wenchang Duan, Yaoliang Yu, Jiwan He, and Yi Shi. Adaptive context length optimization with low-frequency truncation for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 38:97685–97716, 2026.

Xintong Duan, Yutong He, Fahim Tajwar, Ruslan Salakhutdinov, J Zico Kolter, and Jeff Schneider. Accelerating diffusion models in offline RL via reward-aware consistency trajectory distillation. In *NeurIPS 2025 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2025. URL https://openreview.net/forum?id=ZDYsRGOZ4F.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Lihao Guo, Louis Liu, Jiahao Tang, Bo Liu, and Siyang Cao. Opencda-marl: A unified benchmarking framework for cooperative autonomous intersection management with multi-agent reinforcement learning. *IEEE Robotics and Automation Letters*, 2026.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Chang Huang, Shatong Zhu, Junqiao Zhao, Hongtu Zhou, Hai Zhang, Di Zhang, Chen Ye, Ziqiao Wang, and Guang Chen. Potentially optimal joint actions recognition for cooperative multi-agent reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 47566–47589, 2026.

Junyong Jiang, Xin Yuan, Longhe Lin, Songze Li, and Lu Dong. Budget-efficient attacks and robustness training for cooperative MARL. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=k3T8Cwv1ly.

Whiyoung Jung, Sunghoon Hong, Deunsol Yoon, Kanghoon Lee, and Woohyung Lim. Agent-centric actor-critic for asynchronous multi-agent reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

Prajwal Koirala and Cody Fleming. Flow-based single-step completion for efficient and expressive policy learning. In *International Conference on Learning Representations*, volume 2026, pp. 145668–145701, 2026.

Andreas Kontogiannis, Konstantinos Papathanasiou, Yi Shen, Giorgos Stamou, Michael M Zavlanos, and George Vouros. Enhancing cooperative multi-agent reinforcement learning with state modelling and adversarial exploration. In *Forty-second International Conference on Machine Learning*, 2025.

Dongsu Lee, Daehee Lee, and Amy Zhang. Multi-agent coordination via flow matching. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=2L6MffR0ut.

Sunwoo Lee, Mingu Kang, Yonghyeon Jo, and Seungyul Han. Interaction-breaking adversarial learning framework for robust multi-agent reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=ugeI4yowce.

Chao Li, Bingkun BAO, and Yang Gao. In-context fully decentralized cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 38:147796–147818, 2026a.

Pengyi Li, Jianye Hao, Hongyao Tang, Yan Zheng, and Xian Fu. Race: improve multi-agent reinforcement learning with representation asymmetry and collaborative evolution. In *International Conference on Machine Learning*, pp. 19490–19503. PMLR, 2023.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=XUks1Y96NR.

Yueheng Li, Guangming Xie, and Zongqing Lu. Revisiting cooperative off-policy multi-agent reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025b.

Zeyang Li, Sunbochen Tang, and Navid Azizan. Reverse flow matching: A unified framework for online reinforcement learning with diffusion and flow policies. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=vUpEe4yd1T.

Zhuoran Li, Xun Wang, Hai Zhong, Qingxin Xia, Lihua Zhang, and Longbo Huang. Om2p: Offline multi-agent mean-flow policy. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '26, pp. 3767–3776, Richland, SC, 2026c. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400723179. doi: 10.65109/WULT4244. URL https://doi.org/10.65109/WULT4244.

Zhuoran Li, Hai Zhong, Xun Wang, Qingxin Xia, Lihua Zhang, and Longbo Huang. Diffusing to coordinate: Efficient online multi-agent diffusion policies. In *Forty-third International Conference on Machine Learning*, 2026d. URL https://openreview.net/forum?id=tHY0G4PhCR.

Ziyun Li, Huancheng Hu, Soon Hoe Lim, Xuyu Li, Fei Gao, Enmao Diao, Zezhen Ding, Michalis Vazirgiannis, and Henrik Boström. A kinetic energy perspective of flow matching. In *Forty-third International Conference on Machine Learning*, 2026e. URL https://openreview.net/forum?id=d6IyHrX48y.

Bor-Jiun Lin and Chun-Yi Lee. Heterogeneous agent q-weighted policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=kPrvXjUZPG.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Jiarong Liu, Yifan Zhong, Siyi Hu, Haobo Fu, QIANG FU, Xiaojun Chang, and Yaodong Yang. Maximum entropy heterogeneous-agent reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di ZHANG, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=oCBKGw5HNf.

Tenglong Liu, Xin Xu, Yixing Lan, Jiacheng Huang, Xinglong Zhang, and Ling-Li Zeng. Latent meanflow policy optimization for offline reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Lei Lyu, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. *Advances in Neural Information Processing Systems*, 38:93967–93990, 2026.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *Forty-second International Conference on Machine Learning*, 2025.

Laëtitia Matignon, Laurent Jeanpierre, and Abdel-Illah Mouaddib. Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. In *Twenty-sixth AAAI conference on artificial intelligence*, 2012.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *International Conference on Learning Representations*, volume 2026, pp. 36352–36372, 2026.

Alexander Meulemans, Seijin Kobayashi, Johannes von Oswald, Nino Scherrer, Eric Elmoznino, Blake A Richards, Guillaume Lajoie, Blaise Aguera y Arcas, and Joao Sacramento. Multi-agent cooperation through learning-aware policy gradients. In *International Conference on Learning Representations*, volume 2025, pp. 45978–46009, 2025.

Nabuat Zaman Nahim, Fairoz Nower Khan, and Peizhong Ju. A survey of flow matching in reinforcement learning. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=P6e5IC4gPe. Survey Certification.

Thanh Xuan Nguyen and Chang D. Yoo. One-step flow q-learning: Addressing the diffusion policy bottleneck in offline reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=60VgwdzxDM.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

OpenAI. Finite time blowup for navier–stokes. OpenAI Technical Report, 2026. URL https://cdn.openai.com/pdf/32d9f210-8b73-45e0-91bc-82a30aef8a9a/navier-stokes.pdf. Accessed September 2026.

Ling Pan, Qingpeng Cai, and Longbo Huang. Softmax deep double deterministic policy gradients. *Advances in neural information processing systems*, 33:11767–11777, 2020.

Ling Pan, Tabish Rashid, Bei Peng, Longbo Huang, and Shimon Whiteson. Regularized softmax deep multi-agent q-learning. *Advances in Neural Information Processing Systems*, 34:1365–1377, 2021.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KVf2SFL1pi.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34:12208–12221, 2021.

Chengrui Ray Qu, Christopher Yeh, Kishan Panaganti, Eric Mazumdar, and Adam Wierman. Distributionally robust cooperative multi-agent reinforcement learning with value factorization. In *International Conference on Learning Representations*, volume 2026, pp. 9420–9446, 2026.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, pp. 4295–4304. PMLR, 2018.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pp. 5887–5896. PMLR, 2019.

Qi Su, Hongyu Wang, Yu Xia, and Long Wang. A multi-agent reinforcement learning framework for exploring dominant strategies in iterated and evolutionary games. *Nature Communications*, 17(1):490, 2025.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2085–2087, 2018.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

Franki NGUIMATSIA TIOFACK, Théotime Le Hellard, Fabian Schramm, Nicolas Perrin-Gilbert, and Justin Carpentier. Guided flow policy: Learning from high-value actions in offline reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=EBjy1rmpv0.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2021.

Zeyuan Wang, Da Li, Yulin Chen, Ye Shi, Liang Bai, Tianyuan Yu, and Yanwei Fu. One-step generative policies with q-learning: A reformulation of meanflow. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 26751–26759, 2026.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Yulong Xia and Fuchun Sun. Behavior regularization with flow latent policy for offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27028–27036, 2026.

Guangming Xie, Zongqing Lu, et al. Multi-agent guided policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 100398–100418, 2026.

Qinyu Xu, Yuanyang Zhu, Xuefei Wu, and Chunlin Chen. High-order interactions modeling for interpretable multi-agent q-learning. *Advances in Neural Information Processing Systems*, 38: 156614–156646, 2026.

Guang Yang, Jingwen Qiao, Tianpei Yang, Yanqing Wu, Jing Huo, Xingguo Chen, and Yang Gao. Multi-agent reinforcement learning with communication-constrained priors. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Wei Yang and Jesse Thomason. Learning to deliberate: Meta-policy collaboration for agentic llms with multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 29820–29828, 2026.

Deheng Ye, Guibin Chen, Wen Zhang, Sheng Chen, Bo Yuan, Bo Liu, Jia Chen, Zhao Liu, Fuhao Qiu, Hongsheng Yu, et al. Towards playing full moba games with deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:621–632, 2020.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Guojian Zhan, Letian Tao, Pengcheng Wang, Yixiao Wang, Yuxin Chen, Yiheng Li, Hongyang Li, Masayoshi Tomizuka, and Shengbo Li. Mean flow policy with instantaneous velocity constraint for one-step action generation. In *International Conference on Learning Representations*, volume 2026, pp. 157132–157154, 2026.

He Zhang, Ying Sun, and Hui Xiong. Goldenstart: Q-guided priors and entropy control for distilling flow policies. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=UqF3d6yM0S.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *International conference on learning representations*, volume 2025, pp. 17943–17970, 2025a.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=ACagRwCCqu.

Yixian Zhang, Shu-ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. Sac flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. In *International Conference on Learning Representations*, volume 2026, pp. 121983–122016, 2026b.

Yuyang Zhang, Haldun Balim, and Na Li. Decentralized diffusion policy learning for enhanced exploration in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2605.07101*, 2026c.

Wenshuai Zhao, Yi Zhao, Zhiyuan Li, Juho Kannala, and Joni Pajarinen. Optimistic multi-agent policy gradient. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=MWTicAxmRP.

Wenshuai Zhao, Zhiyuan Li, and Joni Pajarinen. Learning progress driven multi-agent curriculum. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Al6qG8BlKg.

Yujie Zhao, Lanxiang Hu, Yang Wang, Minmin Hou, Hao Zhang, Ke Ding, and Jishen Zhao. Stronger-mas: Multi-agent reinforcement learning for collaborative llms. In *International Conference on Learning Representations*, volume 2026, pp. 150619–150651, 2026.

Hai Zhong, Zhuoran Li, Xun Wang, and Longbo Huang. Reparameterization flow policy optimization. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Wx5XHJoeyn.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25(32): 1–67, 2024.

Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. Madiff: Offline multi-agent learning with diffusion models. *Advances in Neural Information Processing Systems*, 37:4177–4206, 2024.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A Detailed Related Works

Here, we review related works on online MARL and diffusion- and flow-based policy optimization, covering both single- and multi-agent RL settings. We further discuss the challenges of one-step flow policy optimization in online MARL and compare OMAF with existing approaches.

A.1 Online Multi-Agent Reinforcement Learning

Online multi-agent reinforcement learning (MARL) has achieved substantial progress in recent years. Under the centralized training with decentralized execution (CTDE) paradigm (Oliehoek et al., 2008; Matignon et al., 2012), a series of influential algorithms, including MADDPG (Lowe et al., 2017), MAPPO (Yu et al., 2022), VDN (Sunehag et al., 2018), COMA (Foerster et al., 2018), QTRAN (Son et al., 2019), QMIX (Rashid et al., 2018), and QPLEX (Wang et al., 2021), have significantly improved coordination and stability in cooperative tasks. In parallel, fully decentralized approaches such as IQL (Tampuu et al., 2017), IPPO (de Witt et al., 2020), and MATD3 (Ackermann et al., 2019) achieve competitive performance without centralized critics, underscoring the scalability of online MARL; recent studies further tackle non-stationarity and co-adaptation in fully decentralized settings via return-aware context modeling (Li et al., 2026a) and learning-aware policy gradients (Meulemans et al., 2025).

Building on these foundations, recent centralized methods further improve online MARL from several perspectives, including value factorization, exploration, robustness, and scalability. Value decomposition and the IGM principle have been enhanced through more expressive factorization (Huang et al., 2026; Xu et al., 2026; Qu et al., 2026), while centralized policy optimization and exploration are improved through joint-policy alignment and adaptive context modeling (Xie et al., 2026; Duan et al., 2026). Other works study exploration, robustness, and scalability through regularization, curriculum learning, communication constraints, and adversarial training (Pan et al., 2021; Li et al., 2023; 2025b; Jung et al., 2025; Chen et al., 2025c; Kontogiannis et al., 2025; Zhao et al., 2025; 2024; Yang et al., 2025; Lee et al., 2026b; Jiang et al., 2026). Heterogeneity-aware methods, including HARL and its variants (Zhong et al., 2024; Liu et al., 2024) and HQP (Lin & Lee, 2026), further extend coordination to heterogeneous agents.

Despite these advances, most online MARL methods use unimodal policies, limiting their ability to represent multimodal coordination strategies. Recent works introduce diffusion policies into online MARL (Li et al., 2026d; Zhang et al., 2026c; Lin & Lee, 2026), but iterative denoising incurs substantial sampling and training overhead. Our approach instead employs one-step flow policies, retaining the expressiveness of generative policies while enabling efficient online MARL.

A.2 Flow Policy in Offline Reinforcement Learning

Flow-based policies provide an efficient module to achieve iterative denoising by learning velocity fields that enable direct or few-step action generation (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023; Lipman et al., 2024; Li et al., 2026e). In offline RL, representative approaches such as FQL (Park et al., 2025), FlowQ (Alles et al., 2025), QIPO (Zhang et al., 2025a), SSCP (Koirala & Fleming, 2026), GFP (TIOFACK et al., 2026), FLP (Xia & Sun, 2026) and OFQL (Nguyen & Yoo, 2026) explore flow-based policy learning through one-step generation, energy-guided training, or direct velocity-field learning. Subsequent works further improve flow policies through value-aware generation, behavior regularization, and efficient distillation (TIOFACK et al., 2026; Zhang et al., 2026a; Xia & Sun, 2026; Duan et al., 2025; Li et al., 2025a).

These developments naturally extend to multi-agent settings, where expressive generative policies can model diverse joint behaviors and coordination patterns. In offline MARL, early flow-based MARL methods include MAC-Flow (Lee et al., 2026a) and OM^{2}P (Li et al., 2026c), both of which operate offline and rely on multi-stage training or distillation. To the best of our knowledge, no prior work directly learns flow policies within online MARL, which is the setting addressed by our work.

A.3 Flow Policy in Online RL

Flow policies offer an expressive and efficient parameterization for reinforcement learning (Nahim et al., 2026), with representative methods including ReinFlow (Zhang et al., 2025b), SAC-

Flow (Zhang et al., 2026b), GS-Flow (Zhang et al., 2026a), and MVP (Zhan et al., 2026). Recent studies further enable direct policy-gradient optimization of flow policies, including FPO (McAllister et al., 2026), OSMFPG (Wang et al., 2026), LaMPO (Liu et al., 2026), FlowRL (Lyu et al., 2026), RFO (Zhong et al., 2026), MFPO (Dong et al., 2026), RFM (Li et al., 2026e), and FPMD (Chen et al., 2025b). Despite these advances, existing flow policy optimization methods are predominantly developed for single-agent RL and do not address the challenges of one-step flow optimization in online MARL. Our work extends flow-based policy optimization to this setting by enabling direct one-step flow policy optimization under CTDE.

B Proofs of Theoretical Results

Proof of Theorem 1. Write $F(\boldsymbol{\theta}) \coloneqq \mathcal{L}_\pi(\boldsymbol{\theta})$ and $\overline{F}(\boldsymbol{\pi}) \coloneqq \overline{\mathcal{L}}_\pi(\boldsymbol{\pi})$.

Step 1: variational representation. For a fixed joint observation, the soft-optimal policy obeys

$$\boldsymbol{\pi}^\star(\boldsymbol{a} \mid \boldsymbol{s}) = \frac{\exp(Q^\star(s, \boldsymbol{a})/\alpha)}{Z(\boldsymbol{s})}, \qquad Z(\boldsymbol{s}) = \int \exp(Q^\star(s, \boldsymbol{a})/\alpha)\, \mathrm{d}\boldsymbol{a}. \tag{14}$$

Substitution of $\alpha \log \boldsymbol{\pi}^\star = Q^\star - \alpha \log Z$ into the ideal actor objective gives the exact variational identity

$$\overline{F}(\boldsymbol{\pi}) - \overline{F}(\boldsymbol{\pi}^\star) = \alpha\, \mathbb{E}_{\boldsymbol{s}\sim\rho}[\mathrm{KL}(\boldsymbol{\pi}(\cdot \mid \boldsymbol{s}) \,\|\, \boldsymbol{\pi}^\star(\cdot \mid \boldsymbol{s}))]\,. \tag{15}$$

The normalizing term $-\alpha \mathbb{E}_\rho[\log Z(\boldsymbol{s})]$ is independent of the candidate policy and cancels in this difference. Thus the ideal actor objective is an information projection onto the soft-optimal policy within the decentralized factorized policy class.

Step 2: objective perturbation. Let $\widehat{Q}$ and $\widehat{\ell}_{\boldsymbol{\theta}}$ denote the critic and joint log likelihood used by the implementation. Before taking absolute values, their cumulative deviation from the ideal objective is

$$F(\boldsymbol{\theta}) - \overline{F}(\boldsymbol{\pi}_{\boldsymbol{\theta}}) = \mathbb{E}_{\rho, \boldsymbol{\pi}_{\boldsymbol{\theta}}}\left[\alpha\big(\widehat{\ell}_{\boldsymbol{\theta}} - \log \boldsymbol{\pi}_{\boldsymbol{\theta}}\big) - \big(\widehat{Q} - Q^\star\big)\right]. \tag{16}$$

Hence uniform expected errors ε_ℓ and ε_Q imply equation 5 by the triangle inequality. In particular, the theorem does not identify the practical log likelihood with an exact likelihood; its discrepancy is carried explicitly by ε_ℓ.

Step 3: transfer from objective error to policy error. Let $\boldsymbol{\theta}_{\mathrm{app}}$ be an arbitrarily accurate minimizer of the left-hand side of equation 5. Applying equation 5 at $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}_{\mathrm{app}}$ yields the following four-step transfer:

$$\begin{aligned}
\overline{F}(\boldsymbol{\pi}_{\boldsymbol{\theta}_k}) &\le F(\boldsymbol{\theta}_k) + \varepsilon_{\mathrm{obj}} \\
&= \inf_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) + \varepsilon_{\mathrm{opt},k} + \varepsilon_{\mathrm{obj}} \\
&\le F(\boldsymbol{\theta}_{\mathrm{app}}) + \varepsilon_{\mathrm{opt},k} + \varepsilon_{\mathrm{obj}} \\
&\le \overline{F}(\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{app}}}) + \varepsilon_{\mathrm{opt},k} + 2\varepsilon_{\mathrm{obj}}.
\end{aligned}$$

Subtracting $\overline{F}(\boldsymbol{\pi}^\star)$, then applying the approximation assumption and equation 15, gives

$$\alpha\, \mathbb{E}_\rho[\mathrm{KL}(\boldsymbol{\pi}_{\boldsymbol{\theta}_k} \| \boldsymbol{\pi}^\star)] \le \varepsilon_{\mathrm{opt},k} + 2\varepsilon_{\mathrm{obj}} + \alpha\varepsilon_{\mathrm{app}}. \tag{17}$$

Dividing by α proves equation 5. In the realizable and exact-objective limit, the KL divergence vanishes only if the two policies agree ρ-almost surely, proving uniqueness.

Step 4: stochastic optimization recursion. Let g_k be the mini-batch gradient, satisfying $\mathbb{E}[g_k \mid \boldsymbol{\theta}_k] = \nabla F(\boldsymbol{\theta}_k)$ and $\mathbb{E}[\|g_k - \nabla F(\boldsymbol{\theta}_k)\|_2^2 \mid \boldsymbol{\theta}_k] \le \sigma_g^2$. From L-smoothness and the update $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta g_k$,

$$\begin{aligned}
\mathbb{E}[F(\boldsymbol{\theta}_{k+1}) \mid \boldsymbol{\theta}_k] &\le F(\boldsymbol{\theta}_k) - \eta \|\nabla F(\boldsymbol{\theta}_k)\|_2^2 + \frac{L\eta^2}{2}\mathbb{E}[\|g_k\|_2^2 \mid \boldsymbol{\theta}_k] \\
&\le F(\boldsymbol{\theta}_k) - \eta\left(1 - \frac{L\eta}{2}\right)\|\nabla F(\boldsymbol{\theta}_k)\|_2^2 + \frac{L\eta^2\sigma_g^2}{2}.
\end{aligned} \tag{18}$$

Because $\eta \le 1/L$, the coefficient of the gradient norm is at least $\eta/2$. The PL inequality $\|\nabla F(\boldsymbol{\theta}_k)\|_2^2 \ge 2\mu\varepsilon_{\mathrm{opt},k}$ consequently gives

$$\mathbb{E}[\varepsilon_{\mathrm{opt},k+1} \mid \boldsymbol{\theta}_k] \le (1 - \eta\mu)\varepsilon_{\mathrm{opt},k} + \frac{L\eta^2\sigma_g^2}{2}. \tag{19}$$

Taking total expectations and unrolling the affine recursion,

$$\mathbb{E}[\varepsilon_{\mathrm{opt},k}] \le (1-\eta\mu)^k \varepsilon_{\mathrm{opt},0} + \frac{L\eta^2\sigma_g^2}{2}\sum_{j=0}^{k-1}(1-\eta\mu)^j$$
$$\le (1-\eta\mu)^k \varepsilon_{\mathrm{opt},0} + \frac{L\eta\sigma_g^2}{2\mu}, \tag{20}$$

which establishes equation 6. □

Proof of Theorem 2. We execute the proof following these steps.

Step 1: endpoint error of a one-step discretization. Let $\widehat{a}_1$ denote the latent state before action squashing in equation 7. Along the optimal characteristic $\dot{a}_1^\star = v^\star(a_t^\star, t \mid \boldsymbol{s})$,

$$a_1^\star = z + \int_0^1 v^\star(a_t^\star, t \mid \boldsymbol{s})\,\mathrm{d}t. \tag{21}$$

Adding and subtracting $v^\star(z, 0 \mid \boldsymbol{s})$ therefore gives

$$\widehat{a}_1 - a_1^\star = \mu_{\boldsymbol{\theta}}(z, \boldsymbol{s}, 0) - v^\star(z, 0 \mid \boldsymbol{s}) + \sigma_{\boldsymbol{\theta}}(z, \boldsymbol{s}, 0) \odot \boldsymbol{\xi}$$
$$- \int_0^1 [v^\star(a_t^\star, t \mid \boldsymbol{s}) - v^\star(z, 0 \mid \boldsymbol{s})]\,\mathrm{d}t. \tag{22}$$

The fundamental theorem of calculus along this characteristic gives the non-autonomous flow identity

$$v^\star(a_t^\star, t \mid \boldsymbol{s}) - v^\star(z, 0 \mid \boldsymbol{s}) = \int_0^t D_s v^\star(a_s^\star, s \mid \boldsymbol{s})\,\mathrm{d}s. \tag{23}$$

Thus, the third term in equation 22 is exactly the local truncation error caused by replacing the full characteristic integral with its initial velocity. Applying Minkowski's integral inequality twice yields

$$\left(\mathbb{E}_{\boldsymbol{s},z,\boldsymbol{\xi}}\left[\|\widehat{a}_1 - a_1^\star\|_2^2\right]\right)^{1/2} \le \varepsilon_\mu + \varepsilon_\sigma + \int_0^1\int_0^t \kappa\,\mathrm{d}s\,\mathrm{d}t$$
$$= \varepsilon_\mu + \varepsilon_\sigma + \frac{\kappa}{2} = \Delta_1. \tag{24}$$

Step 2: latent error to conditional policy discrepancy. For every fixed $\boldsymbol{s}$, couple $\boldsymbol{\pi}_{\boldsymbol{\theta}}^{(1)}(\cdot \mid \boldsymbol{s})$ and $\boldsymbol{\pi}^\star(\cdot \mid \boldsymbol{s})$ with the same $(z, \boldsymbol{\xi})$, where the optimal endpoint simply ignores $\boldsymbol{\xi}$. Since $\tanh$ is 1-Lipschitz, this is an admissible coupling in the definition of W_2, and hence

$$W_2^2\left(\boldsymbol{\pi}_{\boldsymbol{\theta}}^{(1)}(\cdot \mid \boldsymbol{s}), \boldsymbol{\pi}^\star(\cdot \mid \boldsymbol{s})\right) \le \mathbb{E}_{z,\boldsymbol{\xi}\mid\boldsymbol{s}}\left[\|\tanh(\widehat{a}_1) - \tanh(a_1^\star)\|_2^2\right]$$
$$\le \mathbb{E}_{z,\boldsymbol{\xi}\mid\boldsymbol{s}}\left[\|\widehat{a}_1 - a_1^\star\|_2^2\right]. \tag{25}$$

Integrating equation 25 with respect to ρ and invoking equation 24 proves equation 7.

Step 3: conditional policy discrepancy to return loss. The performance-difference identity, written using the normalized discounted occupancy $d_{\boldsymbol{\theta}}^{(1)}$, is

$$J(\boldsymbol{\pi}^\star) - J(\boldsymbol{\pi}_{\boldsymbol{\theta}}^{(1)}) = \frac{1}{1-\gamma}\,\mathbb{E}_{\boldsymbol{s}\sim d_{\boldsymbol{\theta}}^{(1)}}\left[\mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}^\star} Q^\star(s, \boldsymbol{a}) - \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}}^{(1)}} Q^\star(s, \boldsymbol{a})\right]. \tag{26}$$

Kantorovich–Rubinstein duality and the K_Q-Lipschitz property imply that the inner difference in equation 26 is at most $K_Q W_1 \le K_Q W_2$. The occupancy-ratio assumption and Cauchy–Schwarz give

$$\mathbb{E}_{d_{\boldsymbol{\theta}}^{(1)}}[W_2] = \mathbb{E}_\rho\left[\frac{\mathrm{d}d_{\boldsymbol{\theta}}^{(1)}}{\mathrm{d}\rho} W_2\right] \le \sqrt{C_\rho}\,\left(\mathbb{E}_\rho[W_2^2]\right)^{1/2} \le \sqrt{C_\rho}\Delta_1. \tag{27}$$

Substitution into equation 26 proves equation 8. If all three error terms vanish, the coupling agrees almost surely at the endpoint, establishing exact one-step recovery. □

C Details of the Flow Policy

In this section, we provide both the theoretical formulation and the practical implementation of our flow policy. We first show that flow-based policies admit path score surrogate evaluation through the continuous-time change-of-variables formulation. We next derive the corresponding expression and its computation along the flow trajectory. Finally, we detail the flow policy used in our experiments, including the observation-conditioned Transformer architecture for parameterizing the velocity field and the path-wise likelihood surrogate used in practice to provide the log-policy term for policy optimization.

C.1 Derivation of path score surrogate

We employ a factorized policy $\boldsymbol{\pi}_\theta(a|s) = \prod_{i=1}^N \pi_{\theta_i}(a^i|s)$, where each agent i generates its action independently conditioned on the global state s. Specifically, we parameterize the policy of agent i with a Transformer-based velocity field $v_{\theta_i}(a^i_\tau, s, \tau)$ and generate actions by evolving an initial Gaussian noise $a^i_1 \sim \mathcal{N}(0, I)$ along the reverse-time trajectory from $\tau = 1$ to $\tau = 0$, and finally $a^i = \tanh(a^i_0)$. To introduce stochasticity into the generation process, we formulate the flow policy as a reverse-time SDE that preserves the desired marginal distribution (Zhang et al., 2026b; Liu et al., 2025). For each agent i, the reverse-time dynamics are given by

$$\mathrm{d}a^i_\tau = \left[v_{\theta_i}(a^i_\tau, s, \tau) - \frac{\sigma_\tau^2}{2} \nabla_{a^i_\tau} \log p_\tau(a^i_\tau|s) \right] \mathrm{d}\tau + \sigma_\tau \mathrm{d}w, \tag{28}$$

where w denotes a Wiener process and σ_τ controls the level of stochasticity during generation. For the rectified flow formulation, the score term can be expressed using the velocity field, yielding

$$\mathrm{d}a^i_\tau = \left[v_{\theta_i}(a^i_\tau, s, \tau) + \frac{\sigma_\tau^2}{2\tau} \left(a^i_\tau + (1-\tau) v_{\theta_i}(a^i_\tau, s, \tau) \right) \right] \mathrm{d}\tau + \sigma_\tau \mathrm{d}w. \tag{29}$$

Applying the Euler-Maruyama discretization gives the stochastic action update

$$a^i_{\tau-\Delta\tau} = a^i_\tau - \left[v_{\theta_i}(a^i_\tau, s, \tau) + \frac{\sigma_\tau^2}{2\tau} \left(a^i_\tau + (1-\tau) v_{\theta_i}(a^i_\tau, s, \tau) \right) \right] \Delta\tau + \sigma_\tau \sqrt{\Delta\tau} \epsilon, \tag{30}$$

where $\epsilon \sim \mathcal{N}(0, I)$ provides stochasticity during action generation, and $\Delta\tau > 0$ denotes the step size, such that the reverse-time process evolves from $\tau = 1$ to $\tau = 0$ with $\tau \leftarrow \tau - \Delta\tau$ at each step. Conditioned on a^i_τ and s, the one-step denoised action $a^i_{\tau-\Delta\tau}$ follows a Gaussian distribution:

$$p_{\theta_i}\left(a^i_{\tau-\Delta\tau} \mid a^i_\tau, s \right) \sim \mathcal{N}\left(a^i_{\tau-\Delta\tau}; \mu_{\theta_i}(a^i_\tau, s, \tau), \sigma_\tau^2 \Delta\tau I \right), \tag{31}$$

where $\mu_{\theta_i}(a^i_\tau, s, \tau)$ denotes the deterministic mean specified by the drift term in 30. Therefore, the one-step transition likelihood admits a closed-form expression, making it tractable to evaluate and optimize.

C.2 Detailed derivation of the path score surrogate

Let $\tilde{a}^i = (a^i_{\tau_0}, \ldots, a^i_{\tau_H})$ denote the reverse-time trajectory of agent i, where $\tau_0 = 1 > \tau_1 > \cdots > \tau_H = 0$ and $a^i_{\tau_0} \sim \mathcal{N}(0, I)$. The final action is obtained by applying an element-wise $\tanh$ transformation, $a^i = \tanh(a^i_{\tau_H}) = \tanh(a^i_0)$. Based on the factorized policy $\boldsymbol{\pi}_\theta(a|s) = \prod_{i=1}^N \pi_{\theta_i}(a^i|s)$, the joint path density of all agents $\tilde{a} = (\tilde{a}^1, \ldots, \tilde{a}^N) \sim p_\theta(\tilde{a}|s) = \prod_{i=1}^N \pi_{\theta_i}(\tilde{a}^i|s)$ is written as:

$$p_\theta(\tilde{a}|s) = \prod_{i=1}^N \left[\zeta(a^i_{\tau_0}) \prod_{h=0}^{H-1} p_{\theta_i}\left(a^i_{\tau_{h+1}} \mid a^i_{\tau_h}, s; \Delta\tau_h \right) \left\| \det(\mathcal{J}(\tanh(a^i_{\tau_H}))) \right\|^{-1} \right], \tag{32}$$

where $\zeta(\cdot)$ denotes the standard Gaussian base density, and $p_{\theta_i}(\cdot \mid a^i_{\tau_h}, s; \Delta\tau_h)$ is the Gaussian one-step transition induced by the Euler-Maruyama discretization. For a D-dimensional vector a, since the $\tanh$ transformation is applied independently to each action dimension, its Jacobian determinant admits the closed-form expression $\|\det J(\tanh(a))\| = \prod_{d=1}^D \left(1 - (a_d)^2\right)^{-1}$. The marginal policy density of $a = (a^1, \ldots, a^N)$ is written as integrating the intermediate pre-activations:

$$\ell_\theta(a|s) = \int \cdots \int \ell_\theta(\tilde{a}|s) \mathrm{d}a^1_{\tau_0} \cdots \mathrm{d}a^1_{\tau_{H-1}} \cdots \mathrm{d}a^N_{\tau_0} \cdots \mathrm{d}a^N_{\tau_{H-1}}, \qquad a^i = \tanh(a^i_0). \tag{33}$$

We denote $\ell^i \approx \pi_{\theta_i}(a^i|s)$ as the path score surrogate of the agent i and it can be written as

$$\ell^i = \int \cdots \int p_{\theta_i}(\tilde{a}^i|s)\mathrm{d}a^i_{\tau_0} \cdots \mathrm{d}a^i_{\tau_{H-1}}. \tag{34}$$

The path score surrogate enables efficient value estimation and policy optimization compared with the diffusion-based policies, e.g., OMAD (Li et al., 2026d).

We give a theorem to guarantee that the use of the path score surrogate is reasonable for efficient exploration for value estimation and synchronized policy optimization.

Theorem 3 (Bounded discrepancy of the path score surrogate)**.** *Given that $z^i = (a^i_{\tau_0}, \ldots, a^i_{\tau_{H-1}})$, assume that the conditional differential entropy of the latent flow path given the final action and state is finite and bounded by*

$$\left|\mathcal{H}_{\theta_i}(Z^i \mid A^i, s)\right| \leq \varepsilon_i. \tag{35}$$

Then the expected discrepancy between the exact action log-likelihood and the path score surrogate satisfies

$$\left|\mathbb{E}_{p_{\theta_i}(a^i, z^i|s)}\left[\log \pi_{\theta_i}(a^i \mid s) - \ell_{\theta_i}(a^i, z^i \mid s)\right]\right| \leq \varepsilon_i. \tag{36}$$

In particular, if the latent path is conditionally concentrated given the final action, i.e., $\mathcal{H}_{\theta_i}(Z^i \mid A^i, s) \leq \varepsilon_i$ with small ε_i, then the path score provides a close approximation to the exact action log-likelihood in expectation.

For the factorized multi-agent policy $\boldsymbol{\pi}_\theta(a \mid s) = \prod_{i=1}^N \pi_{\theta_i}(a^i \mid s)$, the discrepancy satisfies

$$\left|\mathbb{E}\left[\log \boldsymbol{\pi}_\theta(a \mid s) - \sum_{i=1}^N \ell_{\theta_i}(a^i, z^i \mid s)\right]\right| \leq \sum_{i=1}^N \varepsilon_i. \tag{37}$$

Proof. For any (a^i, z^i) with positive density, Bayes' rule gives

$$p_{\theta_i}(z^i \mid a^i, s) = \frac{p_{\theta_i}(a^i, z^i \mid s)}{\pi_{\theta_i}(a^i \mid s)}. \tag{38}$$

Taking logarithms,

$$\ell_{\theta_i}(a^i, z^i \mid s) = \log \pi_{\theta_i}(a^i \mid s) + \log p_{\theta_i}(z^i \mid a^i, s). \tag{39}$$

Therefore,

$$\log \pi_{\theta_i}(a^i \mid s) - \ell_{\theta_i}(a^i, z^i \mid s) = -\log p_{\theta_i}(z^i \mid a^i, s). \tag{40}$$

Taking expectation with respect to $p_{\theta_i}(a^i, z^i \mid s)$ yields

$$\mathbb{E}\left[\log \pi_{\theta_i}(a^i \mid s) - \ell_{\theta_i}(a^i, z^i \mid s)\right] = -\mathbb{E}\left[\log p_{\theta_i}(z^i \mid a^i, s)\right] = \mathcal{H}_{\theta_i}(Z^i \mid A^i, s). \tag{41}$$

Hence,

$$\left|\mathbb{E}\left[\log \pi_{\theta_i}(a^i \mid s) - \ell_{\theta_i}(a^i, z^i \mid s)\right]\right| = \left|\mathcal{H}_{\theta_i}(Z^i \mid A^i, s)\right| \leq \varepsilon_i. \tag{42}$$

For the factorized policy,

$$\log \boldsymbol{\pi}_\theta(a \mid s) = \sum_{i=1}^N \log \pi_{\theta_i}(a^i \mid s), \tag{43}$$

and thus

$$\mathbb{E}\left[\log \boldsymbol{\pi}_\theta(a \mid s) - \sum_{i=1}^N \ell_{\theta_i}(a^i, z^i \mid s)\right] = \sum_{i=1}^N \mathcal{H}_{\theta_i}(Z^i \mid A^i, s), \tag{44}$$

which gives

$$\left|\mathbb{E}\left[\log \boldsymbol{\pi}_\theta(a \mid s) - \sum_{i=1}^N \ell_{\theta_i}(a^i, z^i \mid s)\right]\right| \leq \sum_{i=1}^N \varepsilon_i. \tag{45}$$

This completes the proof. □

Corollary 1 (Validity for entropy-regularized policy optimization)**.** *Consider an entropy-regularized objective with temperature α_i. Replacing the intractable term $\log \pi_{\theta_i}(a^i \mid s)$ by the path score surrogate $\ell_{\theta_i}(a^i, z^i \mid s)$ incurs at most an expected objective discrepancy of $\alpha_i \varepsilon_i$ for agent i. Hence, when the conditional path entropy is small, path-score optimization provides a controlled approximation to maximum-entropy policy optimization.*

C.3 Design of the Flow Policy

Observation-conditioned flow policy. For each agent i, we encode its local observation into a context representation and use a Transformer decoder to parameterize the flow velocity. Specifically, the observation is first normalized and mapped into the model dimension through a two-layer MLP:

$$c^i = E_o^i\left(\mathrm{BR}(s)\right), \tag{46}$$

where $\mathrm{BR}(\cdot)$ denotes Batch Renormalization and E_o^i consists of two linear layers with a SiLU activation. The resulting context vector is reshaped into a singleton sequence and serves as the memory input to the Transformer decoder.

At flow step h, the current latent action $a_h^i := a_{\tau_h}^i$ is first normalized and projected into the model dimension. In parallel, the normalized flow time $\tau_h = 1 - h/H$ is encoded by a three-layer MLP with SiLU activations. The action and time embeddings are then added to form the decoder input:

$$z_h^i = E_a^i\left(\mathrm{BR}(a_h^i)\right) + E_\tau^i(\tau_h). \tag{47}$$

Both z_h^i and the observation context c^i contain a single token.

Transformer decoder. The input token is processed by L Transformer decoder layers. Each layer contains a self-attention module, an observation-conditioned cross-attention module, and a feed-forward network, with residual connections and Batch Renormalization after each sublayer. For the l-th layer, let $z_h^{i,(l-1)}$ denote the input representation. The computation is

$$\begin{aligned}
u_h^{i,(l)} &= z_h^{i,(l-1)} + \mathrm{SA}_l\left(z_h^{i,(l-1)}\right),\\
\bar{u}_h^{i,(l)} &= \mathrm{BR}\left(u_h^{i,(l)}\right),\\
q_h^{i,(l)} &= \bar{u}_h^{i,(l)} + \mathrm{CA}_l\left(\bar{u}_h^{i,(l)}, c^i, c^i\right),\\
\bar{q}_h^{i,(l)} &= \mathrm{BR}\left(q_h^{i,(l)}\right),\\
z_h^{i,(l)} &= \bar{q}_h^{i,(l)} + \mathrm{FFN}_l\left(\bar{q}_h^{i,(l)}\right),\\
z_h^{i,(l)} &\leftarrow \mathrm{BR}\left(z_h^{i,(l)}\right).
\end{aligned} \tag{48}$$

Here, SA_l denotes self-attention over the action token, while CA_l uses the decoder representation as the query and the observation context as both key and value. Since both sequences contain a single token, the self-attention does not perform token-to-token interaction; instead, the observation-dependent cross-attention injects the local observation information into the action representation. The feed-forward network consists of two linear projections with a GELU activation and expands the hidden dimension by a factor of four.

The multi-head attention modules in each Transformer layer are implemented using scaled dot-product attention. Specifically, for an attention module with N_h heads, the input sequences are first projected into query, key, and value representations. For head j, we compute

$$Q_j = X_q W_{Q,j}, \qquad K_j = X_h W_{K,j}, \qquad V_j = X_v W_{V,j}, \tag{49}$$

and apply scaled dot-product attention:

$$\mathrm{Attn}_j(Q_j, K_j, V_j) = \mathrm{softmax}\left(\frac{Q_j K_j^\top}{\sqrt{d_h}}\right) V_j, \tag{50}$$

where d_h is the dimension of each attention head. The outputs from all heads are concatenated and mapped back to the model dimension by an output projection.

In our flow policy, this attention operation is used in two different ways. First, the self-attention module applies attention to the action representation itself:

$$\mathrm{SA}_l(z) = \mathrm{MHA}_l(z, z, z), \tag{51}$$

where the same representation is used as query, key, and value. Second, the observation-conditioned cross-attention module uses the current action representation as the query and the observation context as both key and value:

$$\mathrm{CA}_l(z, c, c) = \mathrm{MHA}_l(z, c, c). \tag{52}$$

Thus, the cross-attention explicitly conditions the action representation on the local observation.

Velocity and noise prediction. After the final Transformer layer, the resulting representation is passed through two independent linear heads. The first predicts the flow velocity,

$$v_{\theta_i}(a_h^i, o^i, t_h) = W_v^i z_h^{i,(L)} + b_v^i, \tag{53}$$

while the second predicts the diagonal log standard deviation,

$$r_{\theta_i}(a_h^i, o^i, t_h) = W_\sigma^i z_h^{i,(L)} + b_\sigma^i. \tag{54}$$

The predicted log standard deviation is bounded through a hyperbolic tangent and linearly mapped to the prescribed interval $[-5, 2]$, after which the noise scale is obtained as

$$\sigma_{\theta_i} = \exp(r_{\theta_i}). \tag{55}$$

The resulting velocity is sampled as

$$\hat{v}_h^i = v_{\theta_i}(a_h^i, o^i, t_h) + \sigma_{\theta_i}(a_h^i, o^i, t_h)\epsilon_h, \qquad \epsilon_h \sim \mathcal{N}(0, I), \tag{56}$$

and the latent action is updated by

$$a_{k+1}^i = a_h^i - \hat{v}_h^i \Delta t, \qquad \Delta t = \frac{1}{H}. \tag{57}$$

Starting from $a_0^i \sim \mathcal{N}(0, I)$, this iterative flow-matching procedure produces the final latent action a_H^i. The action is then mapped to the environment action space using an element-wise $\tanh$ transformation followed by affine scaling:

$$a^i = \left[\tanh(a_H^i) \odot s^i + b^i\right] \odot m^i, \tag{58}$$

where s^i and b^i denote the action scale and bias, respectively, and m^i is the action-validity mask.

Implemented path score surrogate. While the previous formulation defines the policy path score through the continuous-time change-of-variables integral, we use a path-wise surrogate during sampling to obtain the log-policy term required by the actor objective. Specifically, for a D-dimensional sampled trajectory $\{a_h^i, \epsilon_h^i\}_{h=0}^{H-1}$, the sampler accumulates the log-density contributions from the initial Gaussian latent and the Gaussian perturbations injected at each flow step. After accounting for the change of variables induced by the final $\tanh$ transformation, the resulting scalar is

$$\widetilde{\ell}^{\,i} = \sum_{d=1}^{D} \log \mathcal{N}(a_{0,d}^i; 0, 1) + \sum_{h=0}^{H-1} \sum_{d=1}^{D} \log \mathcal{N}(\epsilon_{h,d}^i; 0, 1) - \sum_{d=1}^{D} \log\left(1 - (u_d^i)^2 + 10^{-6}\right),$$

where u^i denotes the latent variable immediately before the final $\tanh$ transformation. The first term accounts for the initial Gaussian latent, the second accumulates the log-density of the Gaussian perturbations along the flow trajectory, and the last term is the standard change-of-variables correction associated with the $\tanh$ transformation.

We give a theorem to establish that the gap between the approximation and the real integration is bounded.

Theorem 4 (Approximation error of the path score)**.** *Let $\ell_{\theta_i}(a^i, \tilde{a}^i \mid s)$ denote the exact log-density of a sampled flow trajectory, including the Gaussian base density, all Euler–Maruyama transition densities, and the* $\tanh$ *change-of-variables term. Let $\widetilde{\ell}^{\,i}$ be the practical approximation in equation 59. Assume that the noise scales satisfy*

$$0 < \underline{\sigma} \le \sigma_{\tau_h} \sqrt{\Delta \tau_h} \le \overline{\sigma}, \qquad h = 0, \ldots, H-1, \tag{59}$$

and that the pre-activation satisfies

$$1 - (u_d^i)^2 \geq \kappa > 0, \qquad d = 1, \ldots, D. \tag{60}$$

Then

$$\ell_{\theta_i}(a^i, \tilde{a}^i \mid s) - \widetilde{\ell}^{\,i} = -D \sum_{h=0}^{H-1} \log\Big(\sigma_{\tau_h} \sqrt{\Delta\tau_h}\Big) + \sum_{d=1}^{D} \log\left(1 + \frac{10^{-6}}{1 - (u_d^i)^2}\right), \tag{61}$$

and hence

$$\left|\ell_{\theta_i}(a^i, \tilde{a}^i \mid s) - \widetilde{\ell}^{\,i}\right| \leq DH \max\{|\log \underline{\sigma}|, |\log \overline{\sigma}|\} + D \log\left(1 + \frac{10^{-6}}{\kappa}\right). \tag{62}$$

In particular, when the noise schedule and discretization are fixed, the first term is a parameter-independent constant. The remaining approximation error is introduced only by the numerical stabilization term 10^{-6} and is negligible when κ is not too small.

Proof. From the Euler–Maruyama update in equation 30, we can write

$$a^i_{\tau_{h+1}} = \mu_{\theta_i}(a^i_{\tau_h}, s, \tau_h) + \sigma_{\tau_h} \sqrt{\Delta\tau_h}\, \epsilon^i_h, \tag{63}$$

where $\epsilon^i_h \sim \mathcal{N}(0, I)$. Therefore, the exact transition density is

$$p_{\theta_i}\left(a^i_{\tau_{h+1}} \mid a^i_{\tau_h}, s\right) = \mathcal{N}\left(a^i_{\tau_{h+1}}; \mu_{\theta_i}, \sigma^2_{\tau_h} \Delta\tau_h I\right). \tag{64}$$

Using the reparameterization above,

$$\log p_{\theta_i}\left(a^i_{\tau_{h+1}} \mid a^i_{\tau_h}, s\right) = \sum_{d=1}^{D} \log \mathcal{N}(\epsilon^i_{h,d}; 0, 1) - D \log\left(\sigma_{\tau_h} \sqrt{\Delta\tau_h}\right). \tag{65}$$

Summing over all H transitions and adding the Gaussian base density gives

$$\begin{aligned} \ell_{\theta_i} = &\sum_{d=1}^{D} \log \mathcal{N}(a^i_{0,d}; 0, 1) + \sum_{h=0}^{H-1} \sum_{d=1}^{D} \log \mathcal{N}(\epsilon^i_{h,d}; 0, 1) \\ &- D \sum_{h=0}^{H-1} \log\left(\sigma_{\tau_h} \sqrt{\Delta\tau_h}\right) - \sum_{d=1}^{D} \log\left(1 - (u^i_d)^2\right). \end{aligned} \tag{66}$$

Comparing this expression with equation 59, we obtain

$$\ell_{\theta_i} - \widetilde{\ell}^{\,i} = -D \sum_{h=0}^{H-1} \log\left(\sigma_{\tau_h} \sqrt{\Delta\tau_h}\right) + \sum_{d=1}^{D} \log\left(\frac{1 - (u^i_d)^2 + 10^{-6}}{1 - (u^i_d)^2}\right), \tag{67}$$

which is equivalent to the stated equality.

Since

$$1 - (u^i_d)^2 \geq \kappa, \tag{68}$$

we have

$$0 \leq \log\left(1 + \frac{10^{-6}}{1 - (u^i_d)^2}\right) \leq \log\left(1 + \frac{10^{-6}}{\kappa}\right). \tag{69}$$

Moreover,

$$\left|\log\left(\sigma_{\tau_h} \sqrt{\Delta\tau_h}\right)\right| \leq \max\{|\log \underline{\sigma}|, |\log \overline{\sigma}|\}. \tag{70}$$

Applying the triangle inequality yields the desired bound. □

The path score $\widetilde{\ell}^i$ is computed jointly with the action during sampling and serves as a tractable surrogate for the marginal log-policy $\log \pi_{\theta_i}(a^i \mid s)$ in the actor objective. Since it is accumulated along the sampling trajectory, no additional likelihood evaluation is required after action generation.

In the one-step setting, the flow trajectory consists of a single transition from the initial latent a_1^i to the final latent a_0^i. The resulting path score contains the Gaussian base density, the stochastic transition density, and the change-of-variables correction induced by the final $\tanh$ transformation. Thus, even with one-step generation, $\widetilde{\ell}^{\,i}$ provides a tractable scalar likelihood surrogate that can be directly incorporated into entropy-regularized actor optimization.

Importantly, Theorem 4 shows that the practical approximation $\widetilde{\ell}^i$ differs from the exact path score ℓ_{θ_i} only by a parameter-independent normalization term and a bounded numerical stabilization error. Therefore, using $\widetilde{\ell}^i$ preserves the relevant optimization behavior while avoiding the intractable marginalization over intermediate flow states.

C.4 Relationship to Baseline Algorithms

OMAF differs from existing Gaussian policies and diffusion policies in both policy parameterization and likelihood estimation. Compared with Gaussian policies, such as HARL (Liu et al., 2024; Zhong et al., 2024), OMAF parameterizes the policy through a flow velocity field rather than directly specifying a fixed-form Gaussian distribution. Since the marginal likelihood of a flow policy is generally intractable, we use the path score surrogate introduced in Section 5.1 to provide a tractable alternative for entropy-regularized policy optimization. More importantly, even in the one-step setting, the resulting action distribution is fundamentally different from a Gaussian policy. Specifically, the velocity network takes a Gaussian latent variable as input and transforms it through a nonlinear neural network, allowing the induced action distribution to exhibit substantially richer and more complex distributional structures than those obtained by directly predicting a Gaussian mean and standard deviation. Thus, the expressiveness of OMAF arises not from explicitly increasing the number of distributional parameters, but from the nonlinear transformation of Gaussian noise through the learned velocity field.

Compared with diffusion policies, e.g., OMAD (Li et al., 2026d), OMAF differs primarily in the quantity learned to parameterize the generative process. Diffusion policies learn a score function, typically characterized by $\nabla_{a_\tau} \log p_\tau(a_\tau, s, \tau)$, whereas OMAF directly learns the velocity field governing the transport of samples along the flow trajectory. Under appropriate transport formulations, the velocity and score can be closely related, and both approaches generally require a trajectory-based surrogate rather than directly evaluating the marginal likelihood of the generated action. Consequently, their generation processes share a similar structure: both transform an initial noisy sample into a terminal action through a learned continuous-time dynamics. The key distinction is that the learned velocity field in OMAF enables effective few-step, and in our setting one-step, action generation, whereas diffusion policies typically rely on iterative denoising across multiple steps. OMAF therefore retains the flexible generative modeling of noise-to-action transport while substantially reducing the computational cost of action generation.

Existing flow-based reinforcement learning methods, such as SAC-Flow (Zhang et al., 2026b) and FlowRL (Lyu et al., 2026), primarily focus on single-agent decision making, where a flow policy models the action distribution conditioned on the observation of an individual agent. In contrast, OMAF is designed specifically for multi-agent reinforcement learning and adopts a centralized training with decentralized execution (CTDE) framework to model coordinated behaviors among multiple agents.

This CTDE formulation is particularly important for flow-based policies in multi-agent settings. Although each individual flow policy generates actions locally, the centralized training process provides a global learning signal that allows the policies to account for the effects of other agents' actions. Consequently, OMAF can learn coordinated behaviors while preserving decentralized execution and avoiding the need for global information at deployment time.

Another key distinction is that OMAF is designed around one-step flow-based action generation. Rather than directly extending a single-agent flow policy to multiple agents, OMAF jointly optimizes the decentralized flow policies with the centralized critic under the multi-agent CTDE framework. This design enables expressive continuous action modeling while substantially reducing the

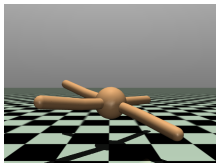
sampling overhead associated with iterative generative policies. These differences make OMAF particularly suited to online multi-agent coordination, where both expressive action distributions and efficient interaction with the environment are essential.

D Experimental Details

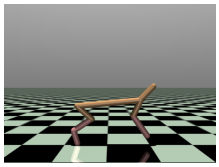
D.1 Experimental Environments



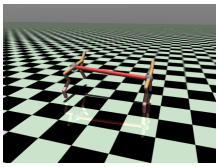
(a) Cooperative Navigation



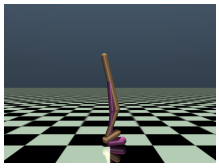
(b) Ant



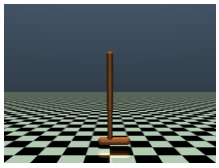
(c) HalfCheetah



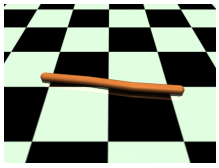
(d) CoupledHalfCheetah



(e) Walker2d



(f) Hopper



(g) Swimmer

Figure 6: Multi-agent particle environments (MPE) and Multi-agent MuJoCo (MAMuJoCo).

We conduct experiments on two widely used multi-agent continuous-control benchmarks, Multi-Agent Particle Environments (MPE) (Lowe et al., 2017)[1] and Multi-Agent MuJoCo (MAMuJoCo) (Peng et al., 2021)[2]. In MPE, we consider one representative task, *Cooperative Navigation*. Cooperative Navigation requires multiple agents to jointly cover a set of landmarks while avoiding collisions (Figure 6a).

For MAMuJoCo, we consider these locomotion environments: *Ant* (Figure 6b), *HalfCheetah* (Figure 6c), *CoupledHalfCheetah* (Figure 6d), *Walker2d* (Figure 6e), and *Swimmer* (Figure 6g). MAMuJoCo converts conventional single-agent MuJoCo control tasks into multi-agent settings by assigning different subsets of the robot's actuators to different agents. The resulting agents jointly control the same physical system, making coordinated continuous control essential for successful locomotion. The configurations used in our experiments are summarized below.

Ant (2×4, 2×4d, 4×2). Ant is a four-legged robot with coupled three-dimensional dynamics. The different agents control disjoint subsets of the robot's actuators, requiring coordinated limb movements to maintain stability while producing forward locomotion.

HalfCheetah (2×3, 6×1). HalfCheetah is a planar articulated robot whose actuators are distributed among multiple agents. Effective locomotion depends on coordinating the different body segments to generate consistent forward movement.

CoupledHalfCheetah. CoupledHalfCheetah consists of two HalfCheetah robots connected by an elastic tendon, requiring agents to coordinate their independently controlled robots to achieve efficient forward locomotion.

Hopper (3×1). Hopper is a two-dimensional one-legged robot controlled through three joint torques, with the goal of maintaining balance while hopping forward. In the multi-agent setting, the three joints are partitioned among three agents, with each agent controlling one joint.

Walker2d (2×3). Walker2d is a two-legged robot with an upright-body constraint. Since control is distributed across agents, successful locomotion requires jointly maintaining body stability and generating forward motion.

[1] https://pettingzoo.farama.org/environments/mpe/

[2] https://robotics.farama.org/envs/MaMuJoCo/

Swimmer (2×1). Swimmer is an articulated robot operating in a fluid-like environment. Its forward movement relies on coordinated motion across multiple connected body segments, making temporal coordination between agents important for effective locomotion.

Compared with the conventional single-agent MuJoCo formulation, MAMuJoCo introduces decentralized control by distributing the robot's actuators across multiple agents. Although the agents interact through a shared physical system, each agent controls only a subset of the action dimensions. As a result, the effect of an individual action depends on the states and actions of the other agents, creating strong physical coupling and coordination requirements. These characteristics make MAMuJoCo a useful benchmark for evaluating multi-agent policy learning in high-dimensional continuous control.

Table 1: Hyperparameters used across all tasks in the MPE and MAMuJoCo environments.

Name of the Hyperparameter	**Value**
Timesteps to Start Learning L_{init}	50000
Critic Soft Update Coefficient ρ	0.0
Discount Factor γ	0.99
Policy Delay d_l	3
Batch Size	512
Replay Buffer Size $\lvert\mathcal{D}\rvert$	1000000
Initial Entropy Coefficient α	0.2
Target Entropy $\mathcal{H}_{\text{target}}$	0
Number of Critics	2
Number of Samples for Softmax Q-Value Estimation K	10
Number of Flow Steps H	1
Transformer Hidden Dimension	96
Number of Transformer Layers	2
Number of Attention Heads	4
Batch Normalization Momentum	0.99
Optimizer	Adam
Adam β_1	0.5
Adam β_2	0.99

D.2 HYPERPARAMETERS

For the critic, we employ a centralized Multi-Layer Perceptron (MLP) to estimate a scalar Q-value from the joint state and joint action. The state-action input is concatenated and normalized using Batch Normalization before being fed into the network. The critic consists of two hidden layers with 1024 units per layer and ReLU activations, followed by a single scalar output. We maintain two critics to reduce estimation bias and improve the stability of value learning.

For the actor, we employ the flow-based policy described in Appendix C. The flow velocity is parameterized by an observation-conditioned Transformer, which takes the current latent action, flow time, and local observation as inputs. We use a one-step flow transformation ($H = 1$), while the Transformer architecture and the corresponding policy optimization details are provided in Appendix C.

To stabilize critic training, we maintain a target policy network for generating next-step actions. The target policy is updated according to an exponential moving average with coefficient ρ. In our experiments, $\rho = 0$, corresponding to a hard-update scheme in which the target policy is updated directly rather than through a soft moving average. The critic uses a discount factor of $\gamma = 0.99$ for computing discounted target values.

Table 1 summarizes the hyperparameters shared across all tasks, including training settings, network architectures, flow configuration, normalization, and optimizer parameters. Table 2 reports task-specific optimization hyperparameters, including the policy and critic learning rates, gradient clipping thresholds, and the temperature β used for softmax Q-value estimation. Unless otherwise specified, all experiments are conducted with 3 independent random seeds.

Table 2: Task-specific optimization hyperparameters for the MPE and MAMuJoCo environments.

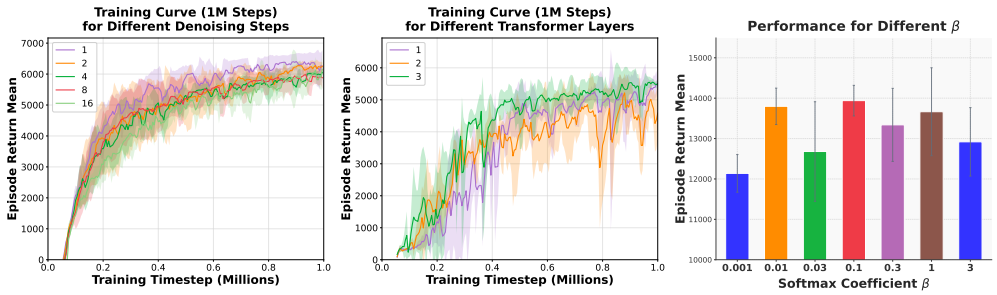
Task	Policy Learning Rate	Critic Learning Rate	Gradient Clip Norm	Softmax Temperature β
Cooperative Navigation 5 Agents	3×10^{-6}	1×10^{-3}	10000	1×10^{-1}
Ant 2×4	3×10^{-4}	3×10^{-3}	10000	1×10^{-2}
Ant 2×4d	1×10^{-4}	3×10^{-3}	10000	1×10^{-2}
Ant 4×2	3×10^{-5}	3×10^{-3}	10000	1×10^{-2}
HalfCheetah 2×3	3×10^{-4}	3×10^{-3}	10	1×10^{-1}
HalfCheetah 6×1	3×10^{-4}	1×10^{-3}	1	1×10^{-2}
Walker2d 2×3	3×10^{-5}	3×10^{-3}	10000	1×10^{-3}
Hopper 3×1	3×10^{-4}	1×10^{-3}	10000	1×10^{-2}
CoupledHalfCheetah	3×10^{-4}	1×10^{-3}	10000	3×10^{-1}
Swimmer 2×1	1×10^{-3}	3×10^{-4}	0.1	1×10^{-2}

D.3 Additional Experimental Results

D.3.1 Additional Ablation Studies

In this section, we perform ablation studies to investigate the impact of three key design choices in OMAF: the denoising steps of the flow policy, the Transformer depth of the velocity network, and the softmax temperature for Q-value estimation. We systematically vary these components to understand their individual effects on policy performance and training efficiency.

Effect of Denoising Steps. Figure 7a compares OMAF with different numbers of denoising steps on the Ant2x4 environment over 1M training steps. All variants are able to learn effective policies and achieve more than 6,500 episodic returns, demonstrating the robustness of the proposed flow policy across different denoising configurations. Notably, the one-step variant achieves the best performance, while requiring only a single function evaluation to generate an action. This result highlights the efficiency of one-step denoising for online policy optimization. In contrast, increasing the number of denoising steps does not lead to consistent performance improvements. A potential explanation is that our policy is updated through Q-value guidance, where the policy gradient is backpropagated through the generated action. With multiple denoising steps, the action depends on a longer sequence of flow transformations, which may introduce additional optimization effects and make the Q-guided policy update less direct. Therefore, one-step flow generation provides a particularly favorable trade-off between expressive policy modeling and efficient gradient-based policy optimization.



(a) Ablation on denoising steps. (b) Ablation on denoising steps. (c) Ablation on softmax β.

Figure 7: Analysis of OMAF design choices on Ant 2×4. (a) Performance with different numbers of denoising steps over 1 million training steps, demonstrating the effectiveness of one-step denoising for online policy optimization. (b) Performance with different numbers of Transformer depth over 1 million training steps. (c) Performance under different softmax temperature settings.

Impact of Transformer Depth. To evaluate how network architecture depth influences policy learning, we compare OMAF across different numbers of Transformer layers (1, 2, and 3). As shown in the training curves, a shallower network (depth = 1) yields a simpler policy, but its limited representation capacity slows down early learning and bounds the overall return. Increasing the depth enhances policy expressiveness, with depth = 3 achieving faster convergence and superior final performance. However, deeper architectures also introduce higher training difficulty and potential optimization instability, as reflected in the variance observed with intermediate settings. These results emphasize the need to strike a balance between policy expressiveness and optimization stability when configuring the network depth.

Effect of Softmax Temperature. We further investigate the effect of the softmax temperature β on the HalfCheetah 2×3 environment with $\beta \in \{0.001, 0.01, 0.03, 0.1, 0.3, 1, 3\}$. The experimental results are presented in the bar plot in Fig. 7c. The results indicate that the temperature needs to be appropriately selected to achieve effective softmax-based target value correction. In particular, moderate temperature values provide a better balance between smoothing the Q-values and preserving their relative differences, which facilitates more effective Q-guided policy optimization.

D.3.2 Exploration Details

To examine the exploration benefits of our expressive flow policies, we analyze the state distributions collected in the replay buffers after the first 250K training steps on Ant 2×4 (Li et al., 2026d). We discretize the 2D state space (dimensions 23 and 13) over $[-4, 8.5] \times [-17.5, 17.5]$ with a grid interval of 0.5, yielding 1750 bins. OMAF visits 1,162 distinct bins, achieving the broadest state-space coverage, compared with 958 for OMAD, 942 for HATD3, 861 for MAFlowRL, 826 for HASAC, and 819 for MAMFPO. The uniquely explored regions further show that OMAF's expressive entropy-regularized flow policy enables broader exploration and helps avoid getting trapped in local optima.

D.3.3 Scalability

As the number of agents grows, multi-agent policy learning becomes increasingly challenging due to the growing dimensionality of joint observations and actions, as well as the stronger coordination dependencies among agents. These challenges are particularly pronounced for expressive generative policies, where action generation may introduce additional computational overhead. In conventional diffusion-based policies, multiple denoising iterations can further increase the cost of generating joint actions, making efficient scaling an important consideration for large-agent systems. Moreover, directly increasing the number of agents in physics-based environments such as MAMuJoCo changes the underlying system dynamics, making controlled scalability studies difficult. We therefore evaluate OMAF on Cooperative Navigation, where the number of agents can be varied while preserving the same underlying task structure.

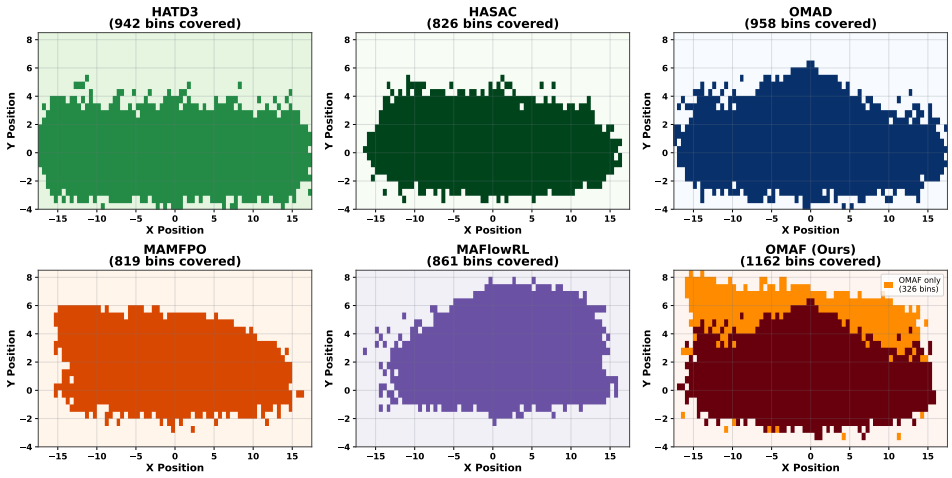


Figure 8: State coverage comparison on representative dimensions (23 and 13) at 250K steps. We visualize the state occupancy within the replay buffers for HATD3, HASAC, diffusion-based OMAD, flow-based MAFlowRL and MAMFPO, and OMAF. Colored regions indicate visited states. OMAF achieves the broadest coverage, where orange regions are uniquely explored by OMAF compared with the diffusion-based OMAD, demonstrating superior exploration.

Table 3 reports the performance with 10 agents and 20 agents. OMAF outperforms the considered baselines, including conventional centralized-critic methods and existing flow- or diffusion-based approaches. The relatively small variance also indicates stable coordination among a larger number of agents. More importantly, OMAF generates actions using a single flow step, avoiding the repeated denoising procedure required by multi-step generative policies. This substantially reduces the computation associated with joint action generation as the number of agents increases. Overall, the results demonstrate that OMAF can maintain effective multi-agent coordination in a substantially larger system while retaining an efficient action-generation mechanism, providing a practical basis for scaling flow-based MARL to larger agent populations.

Table 3: Performance comparison on Cooperative Navigation with 10 and 20 agents.

Task	HATD3	HASAC	MAMFPO	MAFlowRL	OMAD	**OMAF (Ours)**
Cooperative Navigation $N = 10$	-464.3 ± 16.8	-471.3 ± 14.0	-478.4 ± 8.3	-467.2 ± 23.4	-445.1 ± 3.3	$\mathbf{-429.4 \pm 8.3}$
Cooperative Navigation $N = 20$	-1445.1 ± 41.3	-1452.1 ± 18.3	-1680.7 ± 45.8	-1540.6 ± 56.1	-1801.4 ± 36.8	$\mathbf{-1433.4 \pm 14.7}$

D.4 Discussion

D.4.1 Training and Evaluation Protocol

For a fair and consistent comparison, we evaluate all algorithms using three independent random seeds, uniformly fixed to 0, 42, and 3407 across all methods. All other experimental settings are kept consistent with the respective algorithm implementations to ensure that the comparison is not affected by differences in random initialization. At each evaluation point, we roll out the current policy for 10 episodes and use the average episodic return as the evaluation performance; we then report the mean and standard deviation across the three random seeds.

To evaluate performance under a limited interaction budget, we compare OMAF with all baseline methods within 2M training steps. This evaluation protocol emphasizes the early-stage learning performance of each method and allows us to assess how effectively the algorithms utilize limited environment interactions. Under this setting, OMAF achieves strong performance within substantially fewer training steps, demonstrating its advantage in the early stages of online learning.

We further examine the learning curves over longer training horizons. While several baseline methods continue to improve with additional environment interactions, OMAF reaches its converged performance earlier. This indicates that the baselines can benefit from prolonged training, but require more interaction steps to realize their full performance. In contrast, OMAF is built upon an efficient generative policy that enables effective policy learning with fewer environment interactions, highlighting its interaction efficiency for online multi-agent reinforcement learning.

D.4.2 Discussion about the Performance Metric in the Radar Plot

Performance The Performance dimension in the radar plot is computed from the peak evaluation returns across ten benchmark environments: Ant 2×4, Ant 4×2, Ant 2×4d, Walker2d 2×3, HalfCheetah 2×3, HalfCheetah 6×1, Coupled HalfCheetah, Hopper 3×1, Swimmer 2×1, and Cooperative Navigation. For each environment and algorithm, we first identify the peak evaluation return and average it across the three random seeds (0, 42, and 3407).

To account for the different reward scales across environments, we independently normalize the mean peak returns of all algorithms within each environment using min-max normalization:

$$\frac{R_{e,m} - \min_{m'} R_{e,m'}}{\max_{m'} R_{e,m'} - \min_{m'} R_{e,m'}} \times 100, \tag{71}$$

where $R_{e,m}$ denotes the mean peak return of method m on environment e. A higher return always corresponds to a higher normalized score. The Performance score is then obtained by averaging the normalized scores over the ten environments.

Following this procedure, the Performance scores shown in the radar plot are 99.54 for OMAF, 81.60 for OMAD, 42.12 for MAMFPO, 39.13 for MAFlowRL, 11.67 for HASAC, and 6.24 for HATD3.

Training Time The Training Time dimension measures the wall-clock training efficiency on Ant 2×4 over 2M training steps, reported as seconds per 100K training steps. The measured times are 84.744, 97.602, 1427.891, 1701.290, 1807.912, and 2981.520 s/100K steps for HASAC, HATD3, OMAF, OMAD, MAMFPO, and MAFlowRL.

For visualization, the training times are mapped to a 0-100 scale such that a higher score indicates lower computational cost. This yields Training Time scores of 99.41, 99.32, 90.00, 88.09, 87.34, and 79.12, respectively. Notably, OMAF achieves the highest training efficiency among the evaluated generative-policy-based methods, requiring less wall-clock training time than OMAD, MAMFPO, and MAFlowRL under the same experimental setting.

Sample Efficiency The Sample Efficiency dimension measures the relative number of environment interactions required to reach a specified target return. For each environment, we define a target return threshold and record the number of training steps at which the mean learning curve across the three random seeds reaches this threshold for the first time. Methods that do not reach the target within the evaluation horizon are assigned a value of 1.0. The target return thresholds are 3510.4 for Ant 2x4, 2500 for Ant 4x2, 3620.8 for Ant 2x4d, 3370.7 for Walker2d 2x3, 5000 for HalfCheetah 2x3, 10000 for HalfCheetah 6x1, 1402.3 for Coupled HalfCheetah, 2800 for Hopper 3x1, 47.0 for Swimmer 2x1, and -120 for Cooperative Navigation 5.

For each environment, we compute the ratio based on the number of training steps required to reach the corresponding threshold and then average the ratios across the ten environments. This gives average ratios of 9.51 for OMAF, 5.61 for OMAD, 1.39 for HASAC, 2.32 for HATD3, 4.32 for MAMFPO, and 4.91 for MAFlowRL. For visualization, these values are normalized by the OMAF value of 9.51, resulting in Sample Efficiency scores of 100.00, 59.00, 14.62, 24.40, 45.43, and 51.63 for OMAF, OMAD, HASAC, HATD3, MAMFPO, and MAFlowRL, respectively. A higher score indicates that the target performance is reached with fewer environment interactions.

Exploration The Exploration dimension measures the coverage of the visited state space during the early stage of training. Specifically, we collect the states encountered by each algorithm during the first 250K training steps on the Ant 2×4 environment and construct a two-dimensional state occupancy map using the 15th and 25th state dimensions. We discretize the state space into a fixed grid with a bin width of 0.5, where the first dimension ranges from -17.5 to 17.5 with 70 bins and

the second dimension ranges from -4 to 8.5 with 25 bins, resulting in 1,750 bins in total. Samples falling outside these ranges are discarded rather than assigned to boundary bins. A bin is considered visited if it contains at least one collected state. The resulting numbers of occupied bins are 942 for HATD3, 826 for HASAC, 819 for MAMFPO, 861 for MAFlowRL, 958 for OMAD, and 1,162 for OMAF.

For visualization, we apply min-max normalization across the six algorithms, using 819 as the minimum and 1,162 as the maximum $S_m = \frac{N_m}{1162} \times 100$, where N_m denotes the number of occupied bins for algorithm m. This yields Exploration scores of 100.00, 82.44, 81.07, 74.10, 71.08, and 70.48 for OMAF, OMAD, HATD3, MAFlowRL, HASAC, and MAMFPO, respectively. A higher score indicates broader coverage of the evaluated state space during the first 250K training steps.

Scalability The Scalability dimension measures the performance of each algorithm under different numbers of agents. We use the Cooperative Navigation environment with $N = 10$ and $N = 20$ agents and compute the metric from the mean episodic returns at each scale. The reported standard deviations are not used in the calculation. For consistency with the naming of other baselines, IMFPO and IFlowRL are denoted as MAMFPO and MAFlowRL, respectively.

For each agent scale independently, we apply min-max normalization to the mean returns: $S_{N,m} = \frac{R_{N,m} - R_{N,\min}}{R_{N,\max} - R_{N,\min}} \times 100$, where $R_{N,m}$ is the mean return of algorithm m at agent scale N. Since higher returns indicate better performance, a larger normalized score corresponds to better performance at the given scale. At $N = 10$, the best and worst mean returns are -429.4 (OMAF) and -478.4 (MAMFPO), respectively. At $N = 20$, they are -1433.4 (OMAF) and -1801.4 (OMAD), respectively.

We then average the normalized scores across the two agent scales to obtain the final Scalability score. For example, HATD3 obtains a score of 28.78 at $N = 10$ and 96.82 at $N = 20$, resulting in a final score of 62.80. The resulting Scalability scores are 100.00 for OMAF, 62.80 for HATD3, 54.70 for HASAC, 46.86 for MAFlowRL, 33.98 for OMAD, and 16.40 for MAMFPO. A higher score indicates stronger performance across the evaluated agent scales.

GPU Hours The GPU Hours dimension measures the computational cost required for 100K training steps on the Ant 2×4 environment, where a higher score indicates lower computational cost. We use the measured training time in seconds per 100K steps and account for the total GPU compute capacity used by each algorithm. Evaluation time is not included in the calculation. IMFPO and IFlowRL are denoted as MAMFPO and MAFlowRL, respectively.

In our experiments, HASAC and HATD3 are trained on 8 NVIDIA 3090 GPUs. OMAF uses 8 NVIDIA 3090 GPUs and 4 NVIDIA A6000 GPUs, while MAMFPO and MAFlowRL use 4 NVIDIA 3090 GPUs and 8 NVIDIA A6000 GPUs. OMAD uses 600 NVIDIA V100 GPUs, 20 NVIDIA H20 GPUs, and 60 NVIDIA A100 GPUs. We estimate the computational cost as the aggregate FP32 peak throughput of the GPUs used by each algorithm multiplied by its wall-clock training time.

Importantly, OMAF achieves its strong performance with dramatically lower computational resource requirements than the diffusion-based OMAD, using only 12 GPUs compared with 680 GPUs in the reported training setup. This substantial reduction in GPU consumption highlights the exceptional training efficiency of OMAF, demonstrating that competitive or stronger policy performance can be achieved with only a small fraction of the computational resources required by iterative diffusion-based policy learning.

Because the resulting computational costs span a large range across methods, we first apply a logarithmic transformation and then linearly map the values to a 0–100 scale. The resulting GPU Hours scores are 100.00 for HASAC, 99.57 for HATD3, 90.00 for OMAF, 89.19 for MAMFPO, 87.66 for MAFlowRL, and 79.45 for OMAD. A higher score indicates lower estimated computational cost per 100K training steps.

D.4.3 Discussion about the Discrete Action Space

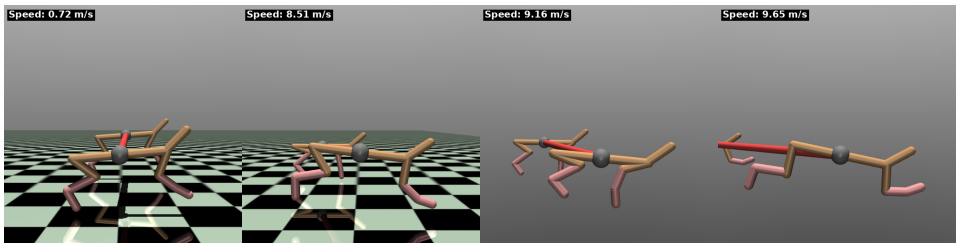
Our work focuses on efficient online multi-agent coordination in continuous control settings. Accordingly, we evaluate the proposed framework on representative continuous-control benchmarks,

including MPE and MAMuJoCo, where agents interact through continuous action spaces and exhibit challenging coordination dynamics arising from physical coupling. We do not consider discrete-action benchmarks such as SMAC in this work, as the proposed flow-based policy is specifically designed to model continuous action distributions. Extending the framework to discrete action spaces, for example through discrete flow formulations or categorical action representations, would require additional methodological development and is left for future work.

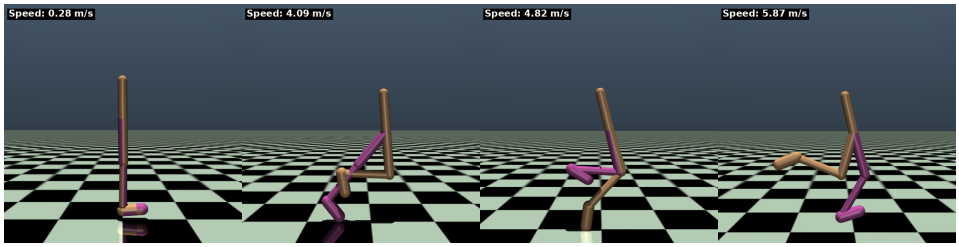
D.5 VISUALIZATION

To complement the quantitative evaluation, we visualize the behaviors learned by OMAF across four representative MAMuJoCo tasks in Figure 9. For each task, we show the agent configurations at timesteps $t \in 1, 100, 250, 500$ together with their instantaneous velocities. Across different environments, the agents progressively develop consistent locomotion patterns and maintain stable motion throughout the trajectories. In the CoupledHalfCheetah, Walker2d 2×3, and Ant 2×4 tasks, the agents exhibit coordinated movements despite their interactions being governed by the underlying coupled dynamics, while the Swimmer 2×1 agents maintain a coherent forward-moving pattern. The corresponding episode returns further indicate that these learned behaviors translate into effective control performance, with returns of 8320.8, 5971.7, 7495.1, and 138.9, respectively. These visualizations provide qualitative evidence that OMAF can learn stable and coordinated locomotion behaviors across diverse multi-agent continuous-control settings.

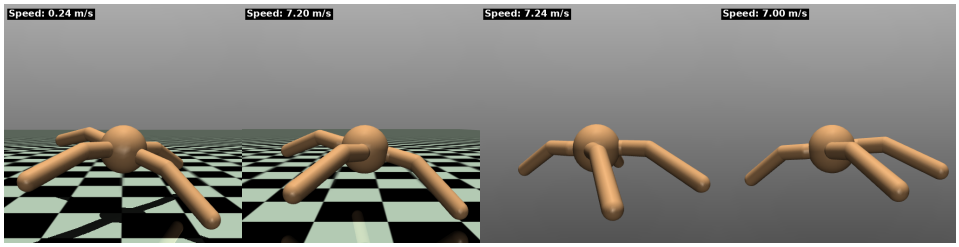
To further examine the coordination efficiency of OMAF under a limited training budget, we visualize the evaluation trajectories of all six algorithms on the Coupled HalfCheetah environment after a fixed number of training steps. Each trajectory corresponds to one evaluation episode generated by the learned policy. The visualization provides a qualitative comparison of the resulting coordination behaviors and shows that OMAF achieves effective coordination while reaching strong task performance within the given training budget.



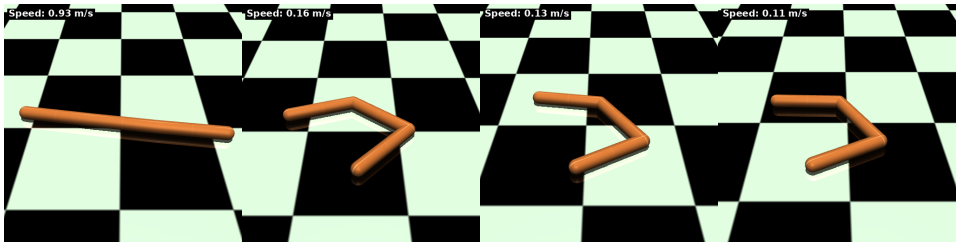
(a) CoupledHalfCheetah Return $= 8320.8$



(b) Walker2d 2×3 Return $= 5971.7$

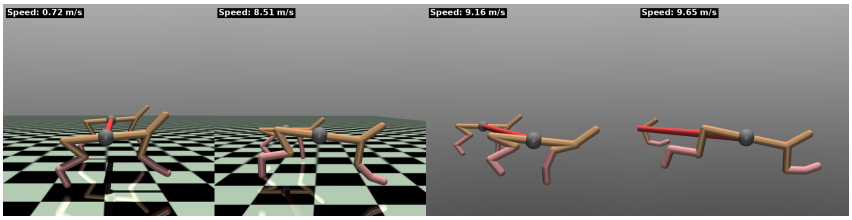


(c) Ant 2×4 Return $= 7495.1$

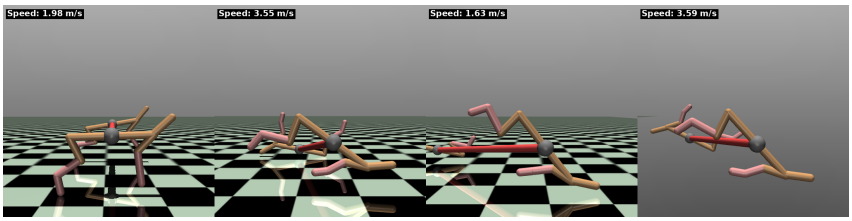


(d) Swimmer 2×1 Return $= 138.9$

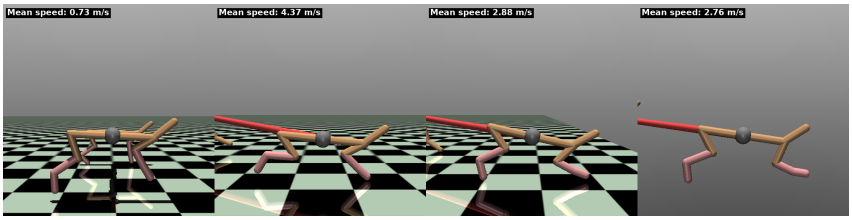
Figure 9: Visualization of learned diffusion policies across four distinct MAMuJoCo tasks. We display snapshots of the agents at timesteps $t \in \{1, 100, 250, 500\}$ with the instantaneous velocity, demonstrating the stable and coordinated behaviors achieved by our OMAF algorithm.



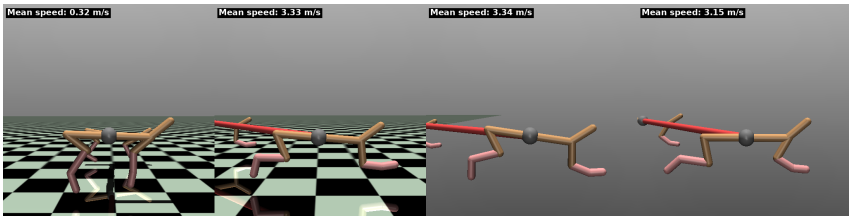
(a) OMAF Return = 8320.8



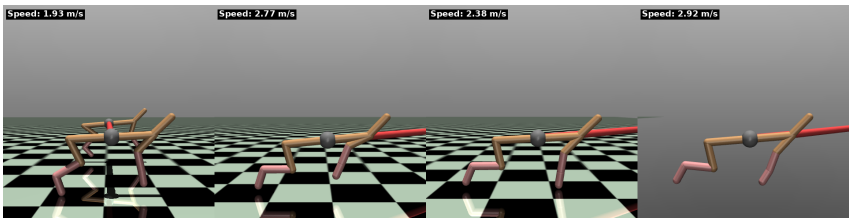
(b) OMAD Return = 1742.09



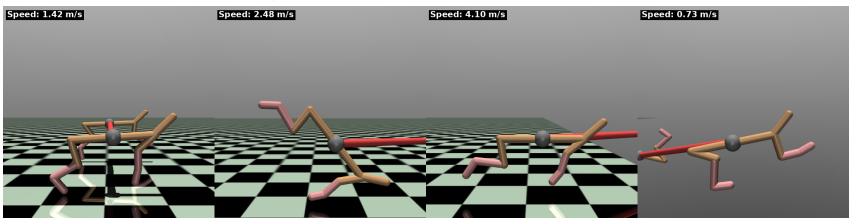
(c) HASAC Return = 2505.96



(d) HATD3 Return = 3356.11



(e) MAMFPO Return = 1841.64



(f) MAFlowRL Return = 2460.30

Figure 10: Visualization of OMAF and other baseline algorithms in MAMuJoCo task Coupled-HalfCheetah. We display snapshots of the agents at timesteps $t \in \{1, 100, 250, 500\}$ with the instantaneous velocity, demonstrating the stable and coordinated behaviors achieved by OMAF.