MEANMAC: ONE-STEP MEANFLOW POLICIES WITH TEAM Q-LEARNING FOR MULTI-AGENT COORDINATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline multi-agent reinforcement learning (MARL) requires policies that can represent diverse cooperative behaviors from fixed datasets while remaining efficient enough for multi-agent decision making. Although diffusion- and flow-based policies provide substantially greater expressivity than conventional unimodal policies, existing generative MARL methods either rely on iterative sampling at execution time or adopt a two-stage pipeline that first learns a multi-step joint generator and subsequently distills it into one-step executable policies, thereby introducing additional inference or training overhead and separating generative modeling from the final policy. To address this challenge, we propose **MeanMAC**: **One-Step MeanFlow Policies with Team Q-Learning for Multi-Agent Coordination**, which directly learns a shared one-step MeanFlow policy for multi-agent action generation and couples policy improvement through a centralized team-value objective. The shared actor maps each agent's local information and noise directly to an action in a single forward pass, while the resulting joint action is optimized through team-level Q-learning, allowing generative distribution modeling and cooperative policy improvement within a unified training procedure. Consequently, MeanMAC avoids iterative denoising or ODE sampling and separate teacher–student distillation while retaining efficient factorized action generation and team-level coordination. Extensive experiments across three offline MARL benchmark suites, covering **10 environments and 33 task–dataset configurations** on MA-MuJoCo, MPE, and SMACv1, demonstrate strong performance across both continuous and discrete control.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) provides a general framework for cooperative decision making and has been applied to embodied control, multi-player games, transportation systems, and socio-economic simulation (Peng et al., 2021; Samvelyan et al., 2019; Ma et al., 2024; Zheng et al., 2022). However, conventional MARL often relies on extensive online interaction, which can be costly or risky in real-world systems. Offline MARL instead learns cooperative policies from fixed datasets without additional environment interaction, but inherits the distribution-shift challenge of offline reinforcement learning, where policy improvement must remain close to the support of the collected data (Levine et al., 2020; Kumar et al., 2020). This problem is more difficult in multi-agent systems because coordination depends on the joint action distribution rather than individual actions alone, while offline datasets may contain multiple cooperative strategies and heterogeneous demonstrations, resulting in highly multi-modal joint behavior distributions.

Generative policies provide a natural way to model such multi-modal behavior. Recent generative MARL methods use diffusion- or flow-based policies to represent complex cooperative distributions beyond conventional Gaussian or deterministic policies (Zhu et al., 2024; Li et al., 2025a; Lee et al., 2026), but their expressivity is commonly accompanied by iterative denoising or numerical ODE integration and therefore repeated network evaluations at every decision step. In contrast, recent single-agent advances show that generative policies can be reduced to one-step generation: MeanFlow-style methods replace iterative velocity integration with averaged-velocity or residual noise-to-action mappings (Geng et al., 2026; Wang et al., 2026), while reformulated one-step generators can be jointly optimized with Q-learning for both distribution modeling and value-based policy improvement (Wang

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.

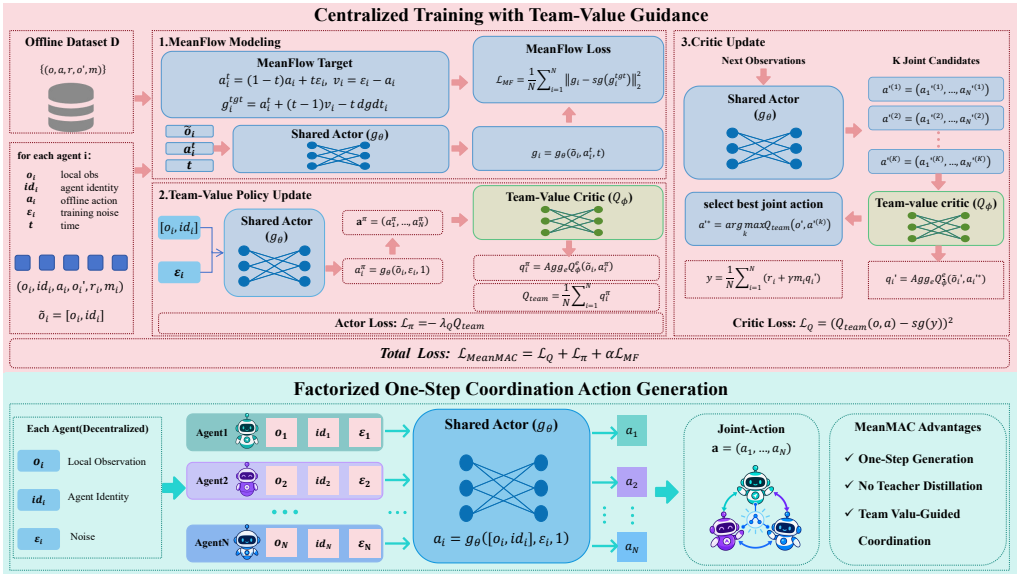


Figure 1: **Overview of MeanMAC.** During centralized training, a shared one-step MeanFlow actor is jointly optimized by offline action modeling and team-value Q-learning, while value-guided selection over complete joint-action candidates is used for Bellman learning. The learned actor retains factorized one-step action generation, allowing each agent to directly map its local observation, identity, and noise to an action without iterative generative sampling or teacher–student distillation.

et al., 2026). These developments provide an efficient route for generative decision making, but extending them from single-agent RL to multi-agent coordination is not straightforward.

In multi-agent systems, the joint action space grows rapidly with the number of agents and the policy must preserve dependencies across different cooperative modes (Liu et al., 2025; Barde et al., 2023). Existing offline MARL methods address this challenge through centralized value learning and value decomposition (Yang et al., 2021; Pan et al., 2022; Son et al., 2019), behavior regularization or learned dynamics (Barde et al., 2023), and more recently generative joint-policy modeling (Zhu et al., 2024; Li et al., 2025a; Lee et al., 2026). Although generative approaches improve expressivity, iterative sampling introduces substantial computational cost. MAC-Flow alleviates this issue by first learning a multi-step joint flow policy and then distilling it into decentralized one-step policies (Lee et al., 2026), but this two-stage pipeline separates joint generative modeling from optimization of the final executable policy. Simply replacing it with independent one-step generators is also insufficient, since individually plausible actions do not necessarily form a high-value coordinated joint action.

To address these challenges, we propose **MeanMAC**, a MeanFlow-based one-step Q-learning framework that directly learns a shared one-step MeanFlow policy for multi-agent decision making. Each agent produces an action from local information and noise through a single forward pass, while the resulting joint action is optimized through a centralized team-value objective. The key idea is to use Q-learning as a coordination signal that shapes locally generated one-step actions at the joint-action level, while the MeanFlow objective preserves generative modeling of the offline action distribution. MeanMAC therefore optimizes the shared one-step policy within a unified learning procedure, avoiding iterative denoising or ODE sampling and a separate teacher–student distillation stage.

Our contributions are summarized as follows.

- We formulate one-step generative offline MARL as a coordination problem, where efficient local generation must be coupled with team-level joint-action optimization.
- We propose MeanMAC, which directly combines factorized one-step MeanFlow policies with centralized team-value Q-learning, avoiding iterative sampling and teacher–student distillation.
- We evaluate MeanMAC across three offline MARL benchmark suites and 33 task–dataset configurations, together with analyses of rollout coordination, scalability, dataset-supported policy modes, offline-to-online adaptation, and component ablations.

2 Related Work

Offline multi-agent reinforcement learning. Offline reinforcement learning learns decision-making policies entirely from fixed datasets without further environment interaction (Levine et al., 2020; Kumar et al., 2020; Wu et al., 2019). In cooperative multi-agent reinforcement learning (MARL), the challenge is amplified because policy distribution shift is coupled across agents: even a small deviation by one agent may move the resulting joint behavior away from the coordinated state–action distribution supported by the offline dataset. Existing offline MARL methods address this issue through value regularization, policy constraints, model-based optimization, and value decomposition. Representative approaches include MA-ICQ (Yang et al., 2021), MA-CQL and OMAR (Pan et al., 2022), OMIGA (Wang et al., 2023), and related offline multi-agent value-based or policy-regularized methods (Fujimoto & Gu, 2021; Barde et al., 2023; Liu et al., 2025). While these approaches provide effective and scalable mechanisms for offline policy optimization, their policy parameterizations are typically Gaussian, deterministic, or autoregressive, which can limit their ability to represent the complex and potentially multi-modal action distributions arising from heterogeneous demonstrations and diverse cooperative behaviors.

Generative policies and one-step policy learning. Diffusion and flow-based generative models provide substantially more expressive policy classes for offline RL by modeling complex action or trajectory distributions (Janner et al., 2022; Ajay et al., 2022; Chi et al., 2025; Wang et al., 2022; Park et al., 2025; Zhang et al., 2025; Frans et al., 2025; Ding et al., 2024). Diffusion policies generate samples through iterative denoising (Ho et al., 2020; Song & Ermon, 2019; Song et al., 2020b; Nichol & Dhariwal, 2021), whereas flow-matching policies learn continuous velocity fields that transport a simple noise distribution toward the data distribution (Lipman et al., 2022; Liu et al., 2022; Lipman et al., 2024; Alles et al., 2025). Although these generative policies can better capture multi-modal behavior than conventional unimodal policies, their iterative sampling procedures introduce considerable inference cost, and coupling them directly with Q-learning may require optimization through long denoising or ODE trajectories. Recent one-step and few-step methods reduce this cost through distillation (Salimans & Ho, 2022; Song et al., 2023; 2020a; Meng et al., 2023; Luhman & Luhman, 2021), shortcut or averaged-velocity parameterizations (Geng et al., 2026; Frans et al., 2025), and Q-learning-compatible flow objectives (Park et al., 2025; Wang et al., 2026). In particular, MeanFlow-based Q-learning reformulates averaged-velocity generation into a direct residual noise-to-action mapping, enabling one-step action generation together with value-based policy optimization; however, these advances are primarily developed for single-agent decision making and do not directly resolve the coordination problem arising from multiple simultaneously optimized agents.

Generative models for offline MARL. Recent work has introduced expressive generative policies into offline MARL to better capture multi-modal cooperative behaviors. MADiff (Zhu et al., 2024) models multi-agent trajectories through iterative diffusion, providing strong representational capacity at the cost of repeated denoising. DoF (Li et al., 2025a) improves scalability by decomposing centralized diffusion generation into decentralized policies, while MADiTS (Yuan et al., 2025) exploits reusable high-quality coordination patterns from offline trajectories to construct cooperative behaviors more efficiently. Moving toward one-step generation, OM2P (Li et al., 2025b) employs averaged velocity modeling at the individual-agent level to reduce iterative sampling. MAC-Flow (Lee et al., 2026) further learns an expressive joint flow policy and distills it into one-step decentralized policies for efficient execution. Despite these advances, existing methods either retain iterative generation, model local one-step actions without explicit joint-value coupling, or rely on a two-stage teacher–student pipeline that separates joint behavior modeling from the final executable policy. To address these limitations, we propose MeanMAC, which directly learns a shared one-step MeanFlow policy and uses team-level Q-learning to jointly shape the locally generated actions according to the value of the complete joint action. This enables direct one-step multi-agent generation without iterative sampling or separate distillation while preserving coordinated policy improvement.

3 Background

Offline cooperative MARL. We consider offline cooperative multi-agent reinforcement learning formulated as a decentralized partially observable Markov decision process (Dec-POMDP) (Bernstein et al., 2002), defined by $\mathcal{M} = (\mathcal{I}, \mathcal{S}, \{\mathcal{O}_i\}_{i=1}^N, \{\mathcal{A}_i\}_{i=1}^N, \mathcal{T}, \{\Omega_i\}_{i=1}^N, r, \gamma)$, where $\mathcal{I} = \{1, \dots, N\}$ denotes the set of agents. At each timestep, agent i receives a local observation $o_i \in \mathcal{O}_i$ and selects an action $a_i \in \mathcal{A}_i$, forming the joint observation $\mathbf{o} = (o_1, \dots, o_N)$ and joint action $\mathbf{a} = (a_1, \dots, a_N)$.

Learning is performed solely from a fixed dataset $\mathcal{D}$ collected by unknown behavior policies, without further interaction with the environment. The objective is to learn decentralized policies $\Pi = \{\pi_i(a_i \mid o_i)\}_{i=1}^N$ that maximize the expected discounted return

$$J(\Pi) = \mathbb{E}_{\boldsymbol{\tau}\sim p^{\Pi}}\left[\sum_{t=0}^{H}\gamma^t r(s_t, \mathbf{a}_t)\right], \tag{1}$$

while avoiding excessive deviation from behaviors supported by the offline dataset.

MeanFlow and reformulated MeanFlow. Flow matching learns a transport field between a simple prior distribution and a target data distribution (Lipman et al., 2022; Liu et al., 2022). Given a dataset action a, noise $e \sim p_0$, and the interpolation $a_t = (1-t)a + te$, the corresponding instantaneous velocity is $v(a_t, t) = e - a$. Mean Flow (Geng et al., 2026) replaces repeated integration of the instantaneous velocity with an average velocity over a finite interval,

$$u(a_t, b, t) = \frac{1}{t-b}\int_b^t v(a_\tau, \tau)\, d\tau, \qquad 0 \le b < t \le 1, \tag{2}$$

which enables one-step inference by first estimating the average velocity and then transforming noise into an action,

$$\hat{a} = e - u_\theta(e, 0, 1), \qquad e \sim \mathcal{N}(0, I). \tag{3}$$

Reformulated MeanFlow (Wang et al., 2026) further converts this velocity-based inference into a direct residual noise-to-action mapping,

$$g_\theta(o, a_t, b, t) = a_t - u_\theta(o, a_t, b, t), \tag{4}$$

so that one-step inference becomes $\hat{a} = g_\theta(o, e, 0, 1)$. The residual generator is trained to satisfy the MeanFlow identity through

$$\mathcal{L}_{\mathrm{MFI}}(\theta) = \mathbb{E}\left[\left\|g_\theta(o, a_t, b, t) - \mathrm{sg}(g_{\mathrm{tgt}})\right\|_2^2\right], \tag{5}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient and g_{tgt} is the regression target induced by the MeanFlow identity. This formulation directly learns the complete noise-to-action transformation while retaining one-step generative inference.

Individual-global-max principle. For cooperative value learning, the individual-global-max (IGM) principle establishes a consistency relation between agent-wise utility functions and the global team action-value function (Son et al., 2019; Rashid et al., 2020). Specifically, the maximizer represented by the collection of individual utilities is required to agree with a maximizer of the global action-value function:

$$\arg\max_{\mathbf{a}} Q_{\mathrm{tot}}(\mathbf{o}, \mathbf{a}) = \left(\arg\max_{a_1} Q_1(o_1, a_1), \ldots, \arg\max_{a_N} Q_N(o_N, a_N)\right). \tag{6}$$

This consistency provides a factorized representation of cooperative action preferences while preserving agreement with the team-level value objective. Representative IGM-compatible realizations include additive value decomposition as in VDN (Sunehag et al., 2017) and monotonic value mixing as in QMIX (Rashid et al., 2020).

4 METHOD

As illustrated in Figure 1, MEANMAC directly learns a shared one-step MeanFlow actor for multi-agent action generation under centralized team-value guidance. Its training consists of three coupled components: MeanFlow modeling of offline per-agent actions, team-value-guided policy improvement of the shared one-step actor, and Bellman critic learning with value-guided joint-action selection. Each agent generates actions from its local observation, identity, and noise through the shared actor, while shared local critics are aggregated into a team value that couples policy optimization across agents. Coordination is therefore induced through the value of complete joint actions rather than through a centralized joint actor or a separate teacher–student distillation process.

4.1 MEANMAC: ONE-STEP GENERATIVE MULTI-AGENT COORDINATION

Policy class and one-step generation. For each agent i, we augment its local observation with a one-hot agent identity, $\tilde{o}_i = [o_i, \mathrm{id}_i]$, and share a reformulated MeanFlow actor g_θ across agents. Given local noise $\epsilon_i \sim p_0$, each agent generates

$$a_i^\pi = g_\theta(\tilde{o}_i, \epsilon_i, 1), \qquad \mathbf{a}^\pi = (a_1^\pi, \dots, a_N^\pi) \tag{7}$$

where p_0 is the base noise distribution. The induced joint policy is factorized as

$$\pi_\theta^{\mathrm{base}}(\mathbf{a} \mid \mathbf{o}) = \prod_{i=1}^{N} \pi_\theta(a_i \mid o_i, \mathrm{id}_i) \tag{8}$$

so each local action requires only one forward evaluation of the shared actor.

MeanFlow modeling. To model the multi-modal actions contained in the offline data, given a dataset action a_i, noise $\epsilon_i \sim p_0$, and interpolation time $t \in (0, 1]$, we construct

$$a_i^t = (1-t)a_i + t\epsilon_i, \qquad v_i = \epsilon_i - a_i \tag{9}$$

where a_i^t is the interpolated action and v_i is its instantaneous velocity. The actor predicts $g_i = g_\theta(\tilde{o}_i, a_i^t, t)$, and the reformulated MeanFlow identity gives the regression target

$$g_i^{\mathrm{tgt}} = a_i^t + (t-1)v_i - t\frac{dg_i}{dt} \tag{10}$$

where dg_i/dt denotes the total derivative along the interpolation path. The MeanFlow objective is

$$\mathcal{L}_{\mathrm{MF}}(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} w_i \left\| g_i - \mathrm{sg}(g_i^{\mathrm{tgt}}) \right\|_2^2\right] \tag{11}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient and w_i is the adaptive regression weight.

Team-value coordination. Factorized action generation alone does not ensure that locally plausible actions form a coordinated joint action. MeanMAC therefore evaluates generated actions using shared local critic ensembles. For an agent-action pair, we define

$$q_i = \mathrm{Agg}_e\, Q_\phi^e(\tilde{o}_i, a_i), \qquad Q_{\mathrm{team}}(\mathbf{o}, \mathbf{a}) = \frac{1}{N}\sum_{i=1}^{N} q_i \tag{12}$$

where Agg_e aggregates the critic ensemble and Q_{team} evaluates the complete joint action. This team value provides a common coordination signal that jointly shapes the locally generated actions across agents.

Value-guided joint-action selection. For Bellman learning, stochastic one-step generation produces multiple complete joint-action candidates at the next observation:

$$\mathbf{a}'^{(k)} = \left(g_\theta(\tilde{o}'_1, \epsilon_1^{(k)}, 1), \dots, g_\theta(\tilde{o}'_N, \epsilon_N^{(k)}, 1)\right), \qquad k = 1, \dots, K \tag{13}$$

The target team critic selects one complete candidate according to

$$k^\star = \arg\max_k Q_{\bar{\phi},\mathrm{team}}\left(\mathbf{o}', \mathbf{a}'^{(k)}\right), \qquad \mathbf{a}'^\star = \mathbf{a}'^{(k^\star)} \tag{14}$$

where $\bar{\phi}$ denotes the target critic parameters. The maximization is performed over complete joint actions, rather than independently selecting a local action for each agent.

4.2 LEARNING OBJECTIVES

Actor loss. Fresh noise samples are passed through the shared one-step actor to obtain $\mathbf{a}^\pi$. Policy improvement is driven by the team value of the resulting joint action:

$$\mathcal{L}_\pi(\theta) = \underbrace{-\lambda_Q \mathbb{E}_{\mathbf{o}\sim\mathcal{D}, \mathbf{a}^\pi\sim\pi_\theta}\left[Q_{\phi,\mathrm{team}}(\mathbf{o}, \mathbf{a}^\pi)\right]}_{\text{Team-Value Q Loss}} \tag{15}$$

where λ_Q controls the scale of value-guided policy improvement.

Critic loss. For a transition $(\mathbf{o}, \mathbf{a}, \mathbf{r}, \mathbf{o}', \mathbf{m}) \sim \mathcal{D}$, let $\mathbf{a}'^\star$ be the selected next joint action. The target utility and Bellman target are

$$q'_i = \mathrm{Agg}_e\, Q_{\bar{\phi}}^e(\tilde{o}'_i, a_i'^\star), \qquad y = \mathrm{sg}\left[\frac{1}{N}\sum_{i=1}^{N}(r_i + \gamma m_i q'_i)\right] \tag{16}$$

where m_i is the bootstrap mask and γ is the discount factor. For critic member e, the current team value is

$$Q^e_{\phi,\text{team}}(\mathbf{o}, \mathbf{a}) = \frac{1}{N} \sum_{i=1}^{N} Q^e_\phi(\tilde{o}_i, a_i) \tag{17}$$

and the critic objective is

$$\mathcal{L}_Q(\phi) = \underbrace{\mathbb{E}_\mathcal{D} \left[\frac{1}{E} \sum_{e=1}^{E} \left(Q^e_{\phi,\text{team}}(\mathbf{o}, \mathbf{a}) - y \right)^2 \right]}_{\text{Bellman Q Loss}} \tag{18}$$

Overall objective. MeanMAC jointly optimizes Bellman value learning, team-value-guided policy improvement, and MeanFlow modeling:

$$\mathcal{L}_\text{MeanMAC} = \underbrace{\mathcal{L}_Q(\phi)}_{\text{Critic Loss}} + \underbrace{\mathcal{L}_\pi(\theta)}_{\text{Team-Value Q Loss}} + \underbrace{\alpha \mathcal{L}_\text{MF}(\theta)}_{\text{MeanFlow Loss}} \tag{19}$$

where α balances offline generative modeling against value-guided policy improvement. Thus, MeanMAC directly learns a shared one-step MeanFlow actor for local action generation, while team-level evaluation of complete joint actions provides the coordination signal.

5 Experiments

We evaluate MeanMAC from several complementary perspectives. First, we examine whether direct one-step generative Q-learning can achieve strong offline MARL performance across continuous- and discrete-control domains without iterative generative sampling or a separate distillation stage. Second, we investigate whether factorized one-step action generation induces coordinated joint behavior during rollout and whether such coordination remains structured as the number of agents increases. Third, we study whether the learned policy remains aligned with behavior modes supported by the offline dataset instead of drifting toward unsupported regions. Finally, we evaluate whether MeanMAC can continue improving when additional online experience becomes available after offline training.

5.1 Experimental Setup

We evaluate MeanMAC on three offline MARL benchmark suites: MA-MuJoCo (Peng et al., 2021), Multi-Agent Particle Environments (MPE) (Lowe et al., 2017), and SMACv1 (Samvelyan et al., 2019), covering continuous locomotion, continuous particle-world coordination, and discrete StarCraft II micromanagement, respectively. MA-MuJoCo includes `HalfCheetah`, `Ant`, and `Hopper`; MPE includes `Spread`, `Tag`, and `World`; and SMACv1 includes `3m`, `8m`, `2s3z`, and `2c_vs_64zg`. We compare against imitation and conventional offline RL baselines, including BC (Wu et al., 2019), Diffusion BC (Wang et al., 2022), MA-TD3+BC (Fujimoto & Gu, 2021), MABCQ (Jiang & Lu, 2023), and MACQL (Kumar et al., 2020); representative offline MARL methods MAICQ (Yang et al., 2021), OMAR (Pan et al., 2022), and OMIGA (Wang et al., 2023); and recent generative MARL approaches MADiff (Zhu et al., 2024), DoF and DoF-P (Li et al., 2025a), and MAC-Flow (Lee et al., 2026). Results are reported as mean $\pm$ standard deviation over five random seeds, and detailed benchmark, dataset, baseline, and hyperparameter settings are provided in Appendix C.

5.2 Main Results

MA-MuJoCo. As shown in Table 1, MeanMAC achieves the highest average return of 2853.0, outperforming MAC-Flow (2672.8) and MADiff (2391.7) on continuous-control coordination tasks. **MPE.** As reported in Table 2, MeanMAC obtains the highest average normalized score of 79.3, compared with 68.6 for DoF and 60.8 for MADiff, demonstrating effective team-value-guided generation across diverse coordination tasks and dataset qualities. **SMACv1.** Table 3 shows that MeanMAC achieves the highest average return of 15.99, slightly exceeding DoF (15.96) and MADiff (14.55). Overall, MeanMAC achieves strong performance across both continuous and discrete offline MARL benchmarks without iterative generative sampling or separate distillation.

5.3 Rollout-Level Coordination

Table 1: **Performance on MA-MuJoCo.** Results are reported as mean ± standard deviation over five seeds. M-E and M-R denote Medium-Expert and Medium-Replay, respectively. The top three results in each row are highlighted in red, orange, and green.

Task	Dataset	MAICQ	OMIGA	MATD3BC	OMAR	MACQL	MADiff	MAC-Flow	MeanMAC (ours)
Hopper	Medium	501.8±14.0	1189.3±544.3	870.4±156.7	21.3±24.9	901.3±199.9	1436.8±449.5	1023.5±253.0	2428.9±648.5
	M-R	195.4±103.6	774.2±494.3	269.7±41.8	3.3±3.2	31.4±15.2	936.1±574.0	1166.3±451.9	1229.5±161.2
	M-E	355.4±373.9	709.0±595.7	2904.3±477.4	1.4±0.9	2751.8±123.3	2810.4±723.2	2988.3±480.2	2374.9±186.1
HalfCheetah	Expert	2955.9±459.2	3383.6±552.7	4401.6±169.1	−206.7±161.1	4589.5±98.5	4711.4±213.6	4650.0±271.6	5095.6±83.4
	Medium	2549.3±96.3	3608.1±237.4	2620.8±69.9	−265.7±147.0	3189.4±306.9	2650.0±365.4	4358.5±369.2	3975.4±14.6
	M-R	1922.4±612.9	2504.7±83.5	3528.9±120.9	−235.4±154.9	3500.7±293.9	2830.5±292.8	3030.2±436.8	4010.8±188.3
	M-E	2834.0±420.3	2948.5±518.9	3518.1±381.0	−253.8±63.9	4738.2±181.1	4410.9±836.8	5139.9±84.1	5146.6±24.1
Ant	Expert	2050.0±11.9	2055.5±1.6	2046.9±17.1	312.5±297.5	2082.4±21.7	2060.0±10.3	2060.2±20.0	2077.2±5.0
	Medium	1412.4±10.9	1418.4±5.4	1422.6±21.1	−1710.0±1589.0	1033.9±66.4	1428.4±14.7	1432.4±17.8	1480.9±15.1
	M-R	1016.7±53.5	1105.1±88.9	995.2±52.8	−2014.2±844.7	434.6±108.3	1294.5±360.2	1498.4±20.3	1474.8±17.1
	M-E	1590.2±85.6	1720.3±110.6	1636.1±96.0	−2992.8±7.0	1800.2±21.5	1740.2±158.9	2053.3±20.4	2088.8±4.4
Average rewards		1580.3	1947.0	2201.3	−667.3	2277.6	2391.7	2672.8	2853.0

Table 2: **Normalized performance on MPE.** Results are reported as scores with mean ± standard deviation over five seeds. The top three results in each row are highlighted in red, orange, and green.

Task	Dataset	OMAR	MACQL	MAICQ	MA-TD3+BC	MADiff	DoF-P	DoF	MeanMAC (ours)
Spread	Medium	45.0±18.8	35.3±10.3	29.3±5.5	32.3±3.8	67.5±8.5	60.5±8.5	75.6±8.7	71.1±2.9
	Md-Replay	35.3±14.0	19.2±6.4	13.7±5.6	14.4±5.8	48.4±3.4	48.1±3.6	57.4±6.8	35.8±9.9
	Random	30.4±8.2	20.5±5.8	5.3±3.4	8.8±4.4	20.6±7.6	34.5±5.4	35.9±6.8	42.6±5.7
Tag	Expert	115.8±13.6	84.3±10.2	95.2±10.1	113.1±11.6	120.8±11.3	120.1±6.3	125.6±8.6	130.1±1.5
	Medium	55.3±16.7	62.3±27.8	58.3±18.0	63.3±25.6	78.6±12.3	83.9±9.6	86.3±10.6	94.4±1.1
	Md-Replay	52.4±18.3	23.9±16.2	29.5±21.8	25.7±20.1	57.4±13.4	51.7±10.1	65.4±12.5	67.4±3.1
	Random	10.9±3.8	2.7±4.4	2.2±2.6	3.7±3.5	13.3±3.4	14.8±3.2	16.5±6.3	52.3±1.4
World	Expert	113.4±23.1	65.4±20.2	98.5±21.8	95.3±18.3	124.7±20.1	138.4±20.1	135.2±19.1	136.6±1.6
	Medium	69.2±21.5	56.4±6.4	69.9±20.1	72.4±9.3	80.1±13.4	86.4±10.6	85.2±11.2	109.9±4.1
	Md-Replay	42.6±28.2	21.3±10.3	12.0±9.1	15.4±8.1	51.6±12.1	58.1±11.5	58.6±10.4	71.3±9.0
	Random	9.2±3.6	2.4±3.2	1.0±2.2	2.8±3.5	6.1±2.2	15.1±3.0	13.1±2.1	60.4±2.1
Average rewards		52.7	35.8	37.7	40.7	60.8	64.7	68.6	79.3

To examine whether MeanMAC learns coordinated behavior beyond aggregate returns, we visualize generated rollouts on the MPE `Spread` task under the `Random` and `Medium` datasets in Figure 2. At the initial step, agents exhibit inconsistent assignments and uncoordinated directions. After several environment steps, the trajectories become visibly more structured: agents move toward different landmarks and form more consistent assignment patterns. This qualitative result suggests that MeanMAC does not merely generate independent per-agent actions. Instead, team-value-guided optimization encourages the factorized one-step generators to produce locally generated actions that are compatible at the joint-behavior level.

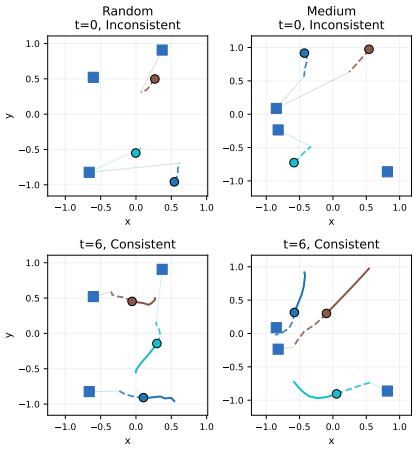


Figure 2: Rollout-level coordination on MPE `Spread`. MeanMAC rollouts are shown on the `Random` and `Medium` datasets. Squares denote landmarks and circles denote cooperative agents. From $t = 0$ to $t = 6$, initially inconsistent assignments evolve into coordinated motion patterns.

5.4 Scaling in the Landmark Covering Game

We next study whether the coordination structure learned by MeanMAC is preserved as the number of agents and landmarks increases. Figure 3 compares trajectories from the offline dataset with trajectories generated by MeanMAC for $N = 3, 5, 10, 20$. The offline trajectories exhibit coordinated agent–landmark assignments, with different agents moving toward different landmarks, and MeanMAC preserves this coordination pattern across all tested team sizes. In particular, the generated trajectories remain distributed across distinct agent–landmark directions rather than collapsing to only a few common behaviors, providing qualitative evidence that the one-step generative policy can preserve diverse coordination patterns as the coordination problem becomes larger.

Table 3: **Performance on SMACv1.** Results are reported as mean ± standard deviation over five seeds. The top three ranks in each row are highlighted in red, orange, and green.

Task	Dataset	BC	MABCQ	Diffusion BC	MAICQ	MACQL	MADiff	DoF	MeanMAC (ours)
3m	Good	16.0±1.0	3.7±1.1	19.5±0.5	18.7±0.7	19.1±0.1	19.3±0.5	19.8±0.2	18.4±0.2
	Medium	8.2±0.8	4.0±1.0	13.3±0.7	13.9±0.8	13.7±0.3	16.4±2.6	18.6±1.2	17.3±0.9
	Poor	4.4±0.1	3.4±1.0	4.2±0.2	8.4±2.6	4.2±0.1	10.3±6.1	10.9±1.1	11.1±0.3
8m	Good	16.7±0.4	4.8±0.6	19.4±0.5	19.6±0.2	18.9±0.9	18.9±1.1	19.6±0.3	19.6±0.2
	Medium	10.7±0.5	5.6±0.6	18.6±0.6	17.9±0.3	15.5±1.5	16.8±1.6	18.6±0.8	18.5±0.1
2s3z	Good	18.2±0.4	7.7±0.9	18.0±1.0	18.3±0.2	17.4±0.3	15.9±1.2	18.5±0.8	19.4±0.2
	Medium	12.3±0.7	7.6±0.7	13.4±1.4	17.0±0.1	15.6±0.4	15.6±0.3	18.1±0.9	16.9±0.2
	Poor	6.7±0.3	6.6±0.2	6.2±1.2	9.9±0.6	8.4±0.8	8.5±1.3	10.0±1.1	10.5±0.1
2c_vs_64zg	Good	17.5±0.4	10.1±0.2	17.8±1.3	14.2±0.3	12.9±0.2	14.7±2.2	16.1±0.8	18.0±0.2
	Medium	12.5±0.3	9.9±0.2	10.5±1.1	12.0±0.1	11.6±0.1	12.8±1.2	13.9±0.9	15.0±0.0
	Poor	9.7±0.2	9.0±0.2	10.2±2.3	9.8±0.3	10.2±0.1	10.8±1.1	11.5±1.1	11.2±0.1
Average rewards		12.08	6.58	13.74	14.52	13.41	14.55	15.96	15.99

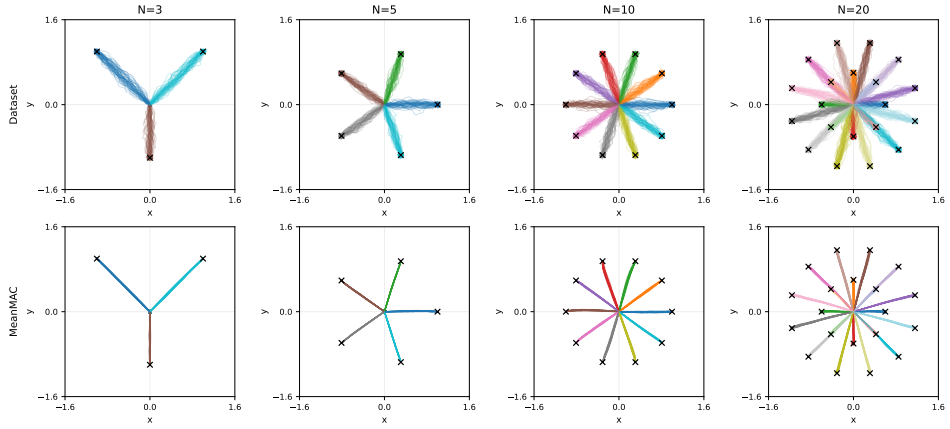


Figure 3: **Coordination scaling in the landmark covering game.** Offline dataset trajectories (top) and MEANMAC trajectories (bottom) are shown for $N = 3, 5, 10, 20$ agents and landmarks. MeanMAC maintains coordinated one-to-one landmark coverage as the team size increases.

5.5 DATASET-SUPPORTED POLICY MODES

We further examine how the policy induced by MEANMAC relates to the behavioral distribution contained in the offline dataset. Figure 4 visualizes state–action samples from the original MPE `Tag` datasets together with samples generated by trained MEANMAC policies using t-SNE. Across `Random`, `Medium`, `Medium-Replay`, and `Expert`, policy samples substantially overlap with regions occupied by the corresponding offline data while concentrating on comparatively compact subsets of the embedding. This pattern is particularly visible on the broader `Random` and `Medium-Replay` datasets, suggesting that value-guided one-step policy optimization can favor structured behavior modes represented in the data instead of simply reproducing the entire heterogeneous dataset distribution.

5.6 OFFLINE-TO-ONLINE ADAPTATION

We further evaluate whether MEANMAC can exploit newly collected online experience after offline training. As shown in Figure 5(a–b), online fine-tuning begins at normalized step 0.5 on MA-MuJoCo `HalfCheetah` Medium and Medium-Replay. In both settings, MEANMAC maintains a clear advantage over MAC-Flow and FlowBC and further improves after online interaction begins, although the learning curves exhibit moderate fluctuations. These results indicate that the same one-step generative Q-learning framework can directly incorporate newly collected experience and continue improving beyond the fixed offline dataset.



Figure 4: **t-SNE visualization of policy and dataset samples on MPE `Tag`.** Blue points denote samples from the offline dataset and orange points denote samples generated by trained MEANMAC policies under `Random`, `Medium`, `Medium-Replay`, and `Expert` datasets. The learned policy largely overlaps with dataset-supported regions while concentrating on more compact behavior modes in the embedding.

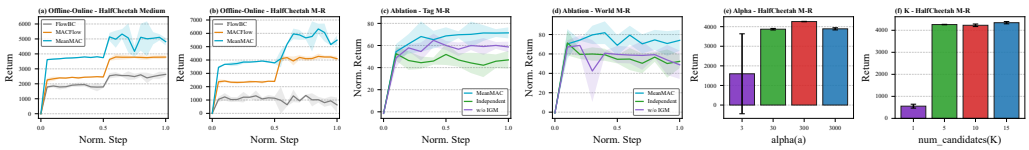


Figure 5: **Additional analyses of MeanMAC.** (a–b) Offline-to-online adaptation on MA-MuJoCo `HalfCheetah` Medium and Medium-Replay, where online fine-tuning starts at normalized step 0.5. (c–d) Coordination ablations on MPE `Tag` and `World` Medium-Replay. (e) Sensitivity to the MeanFlow weight α and (f) sensitivity to the number of joint-action candidates K on `HalfCheetah` Medium-Replay. Results report mean $\pm$ standard deviation over four random seeds.

5.7 ABLATION STUDY

We investigate the role of team-level coordination by comparing the full MEANMAC with the `w/o IGM` and `Independent` variants on MPE `Tag` and `World` under the Medium-Replay setting. The `w/o IGM` variant removes the IGM-compatible value factorization while retaining centralized joint-action value guidance, whereas the `Independent` variant removes team-level coupling and optimizes each agent only with its local value objective. As shown in Figure 5(c–d), the full model generally achieves higher returns than both variants, indicating that one-step generative modeling alone is insufficient and that structured team-value coupling is important for coordinated policy improvement.

5.8 SENSITIVITY ANALYSIS

We further analyze the MeanFlow weight α and the number of joint-action candidates K on MA-MuJoCo `HalfCheetah` Medium-Replay. As shown in Figure 5(e), performance improves substantially as α increases from 3 to 300, while a further increase to 3000 reduces the return, suggesting that an appropriate balance between generative behavior modeling and Q-guided policy improvement is important. Figure 5(f) shows that using a single candidate performs substantially worse, whereas $K \in \{5, 10, 15\}$ yields consistently strong performance, with $K = 15$ achieving the highest mean return among the tested settings. This confirms the benefit of evaluating multiple complete joint-action candidates while also showing that a moderate candidate budget is sufficient.

6 CONCLUSION

We presented MEANMAC, a direct one-step generative approach for offline multi-agent coordination. MeanMAC formulates multi-agent coordination through factorized one-step MeanFlow policies: each agent generates its local action in a single forward pass, while team-level Q-learning jointly shapes the resulting actions according to the value of the complete joint action. This design directly learns the policies used for multi-agent action generation, avoiding iterative generative sampling and a separate teacher–student distillation process. Extensive experiments on MA-MuJoCo, MPE, and SMACv1 demonstrate strong performance across both continuous and discrete multi-agent control, together with effective coordination, scalability, and offline-to-online adaptation. These results show that coordinated multi-agent behavior can be learned through direct one-step generative policies when factorized action generation is optimized with a shared team-level value signal.

AI USE STATEMENT

During the preparation of this manuscript, the authors used ChatGPT to assist with language polishing, including grammar correction and phrasing improvements. ChatGPT was not used for research ideation. The authors reviewed all AI-assisted text and take full responsibility for the final content of this work.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Marvin Alles, Nutan Chen, Patrick van der Smagt, and Botond Cseke. Flowq: Energy-guided flow policies for offline reinforcement learning. *arXiv preprint arXiv:2505.14139*, 2025.

Paul Barde, Jakob Foerster, Derek Nowrouzezahrai, and Amy Zhang. A model-based solution to the offline multi-agent reinforcement learning coordination problem. *arXiv preprint arXiv:2305.17198*, 2023.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4): 819–840, 2002.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Zihan Ding, Amy Zhang, Yuandong Tian, and Qinqing Zheng. Diffusion world model: Future modeling beyond step-by-step rollout for offline reinforcement learning. *arXiv preprint arXiv:2402.03570*, 2024.

Claude Formanek, Asad Jeewa, Jonathan Shock, and Arnu Pretorius. Off-the-grid marl: Datasets with baselines for offline multi-agent reinforcement learning. *arXiv preprint arXiv:2302.00521*, 2023.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, volume 2025, pp. 34668–34684, 2025.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Jiechuan Jiang and Zongqing Lu. Offline decentralized multi-agent reinforcement learning. In *ECAI 2023: 26th European Conference on Artificial Intelligence, September 30–October 4, 2023, Kraków, Poland–Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, pp. 1148–1155. SAGE Publications 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2023.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Dongsu Lee, Daehee Lee, and Amy Zhang. Multi-agent coordination via flow matching. In *International Conference on Learning Representations*, volume 2026, pp. 99912–99943, 2026.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Chao Li, Ziwei Deng, Chenxing Lin, Wenqi Chen, Yongquan Fu, Weiquan Liu, Chenglu Wen, Cheng Wang, and Siqi Shen. Dof: A diffusion factorization framework for offline multi-agent reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 34830–34868, 2025a.

Zhuoran Li, Xun Wang, Hai Zhong, Qingxin Xia, Lihua Zhang, and Longbo Huang. Om2p: Offline multi-agent mean-flow policy. *arXiv preprint arXiv:2508.06269*, 2025b.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Zongkai Liu, Qian Lin, Chao Yu, Xiawei Wu, Yile Liang, Donghui Li, and Xuetao Ding. Offline multi-agent reinforcement learning via in-sample sequential policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19068–19076, 2025.

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.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Chengdong Ma, Aming Li, Yali Du, Hao Dong, and Yaodong Yang. Efficient and scalable reinforcement learning for large-scale network control. *Nature Machine Intelligence*, 6(9):1006–1020, 2024.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14297–14306. IEEE, 2023.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PmLR, 2021.

Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International conference on machine learning*, pp. 17221–17237. PMLR, 2022.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

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.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10199–10210, 2020.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

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.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

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. *arXiv preprint arXiv:1706.05296*, 2017.

Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. *Advances in Neural Information Processing Systems*, 36:52413–52429, 2023.

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. *arXiv preprint arXiv:2208.06193*, 2022.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yiqin Yang, Xiaoteng Ma, Chenghao Li, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:10299–10312, 2021.

Lei Yuan, Yuqi Bian, Lihe Li, Ziqian Zhang, Cong Guan, and Yang Yu. Efficient multi-agent offline coordination via diffusion-based trajectory stitching. In *International Conference on Learning Representations*, volume 2025, pp. 25018–25045, 2025.

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, 2025.

Stephan Zheng, Alexander Trott, Sunil Srinivasa, David C Parkes, and Richard Socher. The ai economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science advances*, 8(18):eabk2607, 2022.

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.

A Detailed Methodology

This section provides the mathematical details underlying the MeanFlow modeling, team-value coordination, value-guided joint-action selection, and Bellman learning introduced in the main text.

A.1 Reformulated MeanFlow Target and JVP

For agent i, given an offline action a_i, noise $\epsilon_i \sim p_0$, and interpolation time t, we define

$$a_i^t = (1-t)a_i + t\epsilon_i, \qquad v_i = \epsilon_i - a_i \tag{20}$$

where a_i^t denotes the interpolated action and $da_i^t/dt = v_i$.

The reformulated MeanFlow mapping is

$$g_\theta(\tilde{o}_i, a_i^t, t) = a_i^t - u_\theta(\tilde{o}_i, a_i^t, t) \tag{21}$$

where u_θ denotes the average velocity field. With the averaging interval starting from zero, the MeanFlow identity is

$$u = v_i - t\frac{du}{dt} \tag{22}$$

Since $g = a^t - u$, differentiation along the interpolation path gives

$$\frac{dg}{dt} = v_i - \frac{du}{dt} \tag{23}$$

and hence

$$u = (1-t)v_i + t\frac{dg}{dt} \tag{24}$$

Substituting this relation into $g = a^t - u$ gives the regression target

$$g_i^{\mathrm{tgt}} = a_i^t + (t-1)v_i - t\frac{dg_i}{dt} \tag{25}$$

where

$$g_i = g_\theta(\tilde{o}_i, a_i^t, t) \tag{26}$$

Because both a_i^t and the explicit time input vary with t, the total derivative is

$$\frac{dg_i}{dt} = \nabla_a g_\theta(\tilde{o}_i, a_i^t, t)\, v_i + \partial_t g_\theta(\tilde{o}_i, a_i^t, t) \tag{27}$$

where the first term is a Jacobian–vector product (JVP), allowing the directional derivative to be evaluated without explicitly constructing the full Jacobian.

To reduce the influence of samples with large regression residuals, we define

$$\delta_i = g_i - \mathrm{sg}(g_i^{\mathrm{tgt}}), \qquad d_i = \mathrm{mean}(\delta_i^2) \tag{28}$$

and use the stop-gradient adaptive weight

$$w_i = \mathrm{sg}\left[(d_i + c)^{\rho-1}\right] \tag{29}$$

where ρ controls the adaptive weighting and $c > 0$ is a numerical stabilizer. The MeanFlow objective is therefore

$$\mathcal{L}_{\mathrm{MF}}(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} w_i d_i\right] \tag{30}$$

A.2 Team-Value Factorization and IGM Compatibility

MeanMAC uses a shared ensemble of local critics. For agent i, the ensemble prediction is aggregated as

$$q_i(\tilde{o}_i, a_i) = \mathrm{Agg}_e\, Q_\phi^e(\tilde{o}_i, a_i) \tag{31}$$

and the resulting local utilities are combined into the team value

$$Q_{\mathrm{team}}(\mathbf{o}, \mathbf{a}) = \frac{1}{N}\sum_{i=1}^{N} q_i(\tilde{o}_i, a_i) \tag{32}$$

Property 1 (IGM compatibility). Assume the joint action space factorizes as $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$ and the local maxima exist. Since each utility q_i depends only on its corresponding local action,

$$\max_{\mathbf{a}} Q_{\text{team}}(\mathbf{o}, \mathbf{a}) = \frac{1}{N} \sum_{i=1}^{N} \max_{a_i} q_i(\tilde{o}_i, a_i) \tag{33}$$

and therefore

$$\arg\max_{\mathbf{a}} Q_{\text{team}}(\mathbf{o}, \mathbf{a}) = \left(\arg\max_{a_1} q_1, \ldots, \arg\max_{a_N} q_N \right) \tag{34}$$

which establishes the IGM compatibility of the additive team-value aggregation used by MeanMAC.

A.3 Value-Guided Joint-Action Selection

For a joint observation $\mathbf{o}$, the shared one-step actor is applied to each agent's local input using independently sampled noise:

$$a_i^{(k)} = g_\theta \left(\tilde{o}_i, \epsilon_i^{(k)}, 1 \right), \qquad \mathbf{a}^{(k)} = \left(a_1^{(k)}, \ldots, a_N^{(k)} \right) \tag{35}$$

for $k = 1, \ldots, K$. The candidate set is

$$\mathcal{C}_K(\mathbf{o}) = \left\{ \mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(K)} \right\} \tag{36}$$

and each complete joint action is scored by the target team critic:

$$S_k = Q_{\bar{\phi},\text{team}} \left(\mathbf{o}, \mathbf{a}^{(k)} \right) \tag{37}$$

The selected action is

$$k^\star = \arg\max_{1 \le k \le K} S_k, \qquad \mathbf{a}^\star = \mathbf{a}^{(k^\star)} \tag{38}$$

The same candidate index $k^\star$ is used for all agents. Thus, the selection preserves one complete sampled joint action instead of recombining independently selected local actions from different candidates. During Bellman learning, the same procedure is applied at the next joint observation $\mathbf{o}'$ to obtain $\mathbf{a}'^\star$.

Property 2 (candidate-set monotonicity). For a fixed target critic and nested candidate sets $\mathcal{C}_K \subseteq \mathcal{C}_{K+1}$,

$$\max_{\mathbf{a} \in \mathcal{C}_{K+1}} Q_{\bar{\phi},\text{team}}(\mathbf{o}, \mathbf{a}) \ge \max_{\mathbf{a} \in \mathcal{C}_K} Q_{\bar{\phi},\text{team}}(\mathbf{o}, \mathbf{a}) \tag{39}$$

For i.i.d. candidate sampling, the expected maximum critic score is likewise non-decreasing with the candidate budget:

$$\mathbb{E}\left[\max_{1 \le k \le K+1} S_k \right] \ge \mathbb{E}\left[\max_{1 \le k \le K} S_k \right] \tag{40}$$

This property concerns the learned critic score and does not imply monotonic improvement of the true environment return when the critic is imperfect.

When $K = 1$, the procedure reduces to the joint action induced by one set of locally sampled actions from the shared actor. For $K > 1$, local action generation remains factorized, whereas the final selected-action distribution is coupled through team-value ranking.

A.4 Critic Ensemble and Bellman Learning

MeanMAC maintains E shared local critics $\{Q_\phi^e\}_{e=1}^E$. For critic member e, its estimate of the team value is

$$Q_{\phi,\text{team}}^e(\mathbf{o}, \mathbf{a}) = \frac{1}{N} \sum_{i=1}^{N} Q_\phi^e(\tilde{o}_i, a_i) \tag{41}$$

Let $\mathbf{a}'^\star$ denote the value-guided next joint action. The target utility of agent i is

$$q_i' = \text{Agg}_e \, Q_{\bar{\phi}}^e(\tilde{o}_i', a_i'^\star) \tag{42}$$

and the corresponding team Bellman target is

$$y = \mathrm{sg}\left[\frac{1}{N}\sum_{i=1}^{N}(r_i + \gamma m_i q_i')\right] \tag{43}$$

where m_i is the bootstrap mask. All critic members are trained against the same team target:

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{\mathcal{D}}\left[\frac{1}{E}\sum_{e=1}^{E}\left(Q^e_{\phi,\mathrm{team}}(\mathbf{o},\mathbf{a}) - y\right)^2\right] \tag{44}$$

The target critic is updated by Polyak averaging:

$$\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi} \tag{45}$$

where $\tau \in (0, 1]$ controls the target-network update rate.

A.5 Actor Value Normalization and Overall Objective

For policy improvement, fresh noise samples produce the joint action $\mathbf{a}^\pi$ through the shared one-step actor. To reduce sensitivity to the numerical scale of the critic, MeanMAC uses

$$\lambda_Q = \mathrm{sg}\left[\frac{1}{\mathbb{E}\left[|Q_{\phi,\mathrm{team}}(\mathbf{o},\mathbf{a}^\pi)|\right]}\right] \tag{46}$$

The team-value policy objective is therefore

$$\mathcal{L}_\pi(\theta) = -\lambda_Q \mathbb{E}\left[Q_{\phi,\mathrm{team}}(\mathbf{o},\mathbf{a}^\pi)\right] \tag{47}$$

Consistent with the main-text objective, MeanMAC jointly optimizes

$$\mathcal{L}_{\mathrm{MeanMAC}} = \mathcal{L}_Q(\phi) + \mathcal{L}_\pi(\theta) + \alpha\mathcal{L}_{\mathrm{MF}}(\theta) \tag{48}$$

where α balances offline MeanFlow modeling against team-value-guided policy improvement. Thus, the shared one-step MeanFlow actor receives two complementary learning signals: distributional supervision from offline local actions through $\mathcal{L}_{\mathrm{MF}}$ and cooperative policy improvement through $\mathcal{L}_\pi$.

B Complexity analysis

We analyze the inference complexity of generating one joint action at a single environment step. Let N denote the number of agents and K denote the number of iterative sampling steps required by diffusion-based generative policies. For each agent, the local observation and action dimensions are treated as constants with respect to N. The analysis focuses on how the inference cost scales with the number of agents and iterative sampling steps.

The per-agent input column describes how the input size of one policy forward pass or one denoising step scales with the number of agents. The total complexity accounts for generating actions for the complete multi-agent system.

MeanMAC. At deployment, MeanMAC uses a factorized one-step actor. Each agent takes its local observation and identity embedding, samples a noise vector, and generates an action through a single forward pass:

$$a_i = g_\theta(\tilde{o}_i, \epsilon_i, 1), \qquad i = 1, \ldots, N. \tag{49}$$

No iterative denoising or ODE integration is required. Consequently, the inference cost of each agent is independent of N, while constructing the full joint action scales linearly with the number of agents, resulting in $\mathcal{O}(N)$ total complexity.

Comparison with iterative generative policies. Diffusion-based MARL methods require K successive denoising steps. DoF factorizes the diffusion process across agents, keeping the per-agent input size independent of N, but generating actions for the complete team still requires $\mathcal{O}(NK)$ computation. MADiff-D performs decentralized execution while modeling joint or teammate-conditioned information for each agent, causing the input size of each denoising step to scale with N and yielding $\mathcal{O}(N^2K)$ total complexity. MADiff-C performs joint generation using a centralized model and avoids repeated joint inference across agents, but iterative denoising still leads to $\mathcal{O}(NK)$ complexity.

Table 4: Inference complexity for generating one joint action.

Method	Execution	Per-agent input	Steps	Total complexity
Gaussian/value-based	Decentralized	$\mathcal{O}(1)$	1	$\mathcal{O}(N)$
DoF	Decentralized	$\mathcal{O}(1)$	K	$\mathcal{O}(NK)$
MADiff-D	Decentralized	$\mathcal{O}(N)$	K	$\mathcal{O}(N^2K)$
MADiff-C	Centralized	$\mathcal{O}(N)$	K	$\mathcal{O}(NK)$
MAC-Flow	Decentralized	$\mathcal{O}(1)$	1	$\mathcal{O}(N)$
MEANMAC	Decentralized	$\mathcal{O}(1)$	1	$\mathcal{O}(N)$

Summary. MEANMAC scales linearly with the number of agents and removes the iterative sampling factor K required by diffusion-based generative policies. Its asymptotic inference complexity is therefore comparable to standard one-step policies and MAC-Flow. Unlike MAC-Flow, which obtains its one-step policy through a separate distillation stage, MEANMAC directly learns a one-step MeanFlow policy.

C IMPLEMENTATION DETAILS

C.1 ADDITIONAL INFORMATION ON OFFLINE DATASETS

We evaluate MEANMAC on three offline multi-agent benchmark suites: Multi-Agent Particle Environments (MPE), Multi-Agent MuJoCo (MA-MuJoCo), and SMACv1. These benchmarks cover continuous particle-world coordination, high-dimensional multi-agent locomotion, and discrete cooperative micromanagement, respectively. For visualizations of the episode-return distributions in the offline datasets used by MEANMAC, we provide violin plots in Figure 6.

Dataset return distributions Dataset quality labels provide only a coarse description of the offline data. To further characterize the datasets used by MEANMAC, we compute the empirical episode-return distribution for each task–dataset pair. For a trajectory τ with N agents, we report the per-agent average episode return:

$$\bar{R}(\tau) = \sum_{t=0}^{T_\tau - 1} \frac{1}{N} \sum_{i=1}^{N} r_{i,t}, \tag{50}$$

where T_τ is the trajectory length. If a dataset stores a shared team reward, the shared reward is counted only once; if the same team reward is duplicated across agents, the agent-wise average gives the same normalization.

Figure 6 shows the return distributions of all offline datasets used in our experiments. The plots reveal that dataset quality labels generally correspond to different return regions, but the distributions are not always narrow or fully separated. In MPE, Expert data usually occupy higher-return regions, whereas Medium-Replay and Random contain broader and lower-return behaviors. In MA-MuJoCo, the return distributions show substantial within-dataset variation, especially on Hopper and HalfCheetah, indicating mixed locomotion behaviors. In SMACv1, Good datasets concentrate near high-return outcomes, while Poor datasets shift toward lower-return regions. These heterogeneous and overlapping distributions motivate the use of an expressive generative policy, together with team-value guidance, to select useful coordination modes from offline data.

MPE MPE (Lowe et al., 2017) is a lightweight continuous-control benchmark for studying multi-agent coordination in two-dimensional particle worlds. We evaluate on three cooperative scenarios: `Spread`, `Tag`, and `World`. In `Spread`, three agents must cover three landmarks while avoiding collisions, requiring implicit assignment between agents and landmarks. `Tag` is a predator–prey scenario, where three predators need to cooperate to capture a faster pre-trained prey. `World` extends the particle-world setting with more complex interaction patterns and environmental structures. Each scenario contains four dataset qualities: `Expert`, `Medium`, `Medium-Replay`, and `Random`. Expert datasets are collected from well-trained policies, Medium datasets from partially trained policies, Medium-Replay datasets from replay buffers during training, and Random datasets from random behaviors. The return distributions of the MPE datasets are visualized in Figure 6.

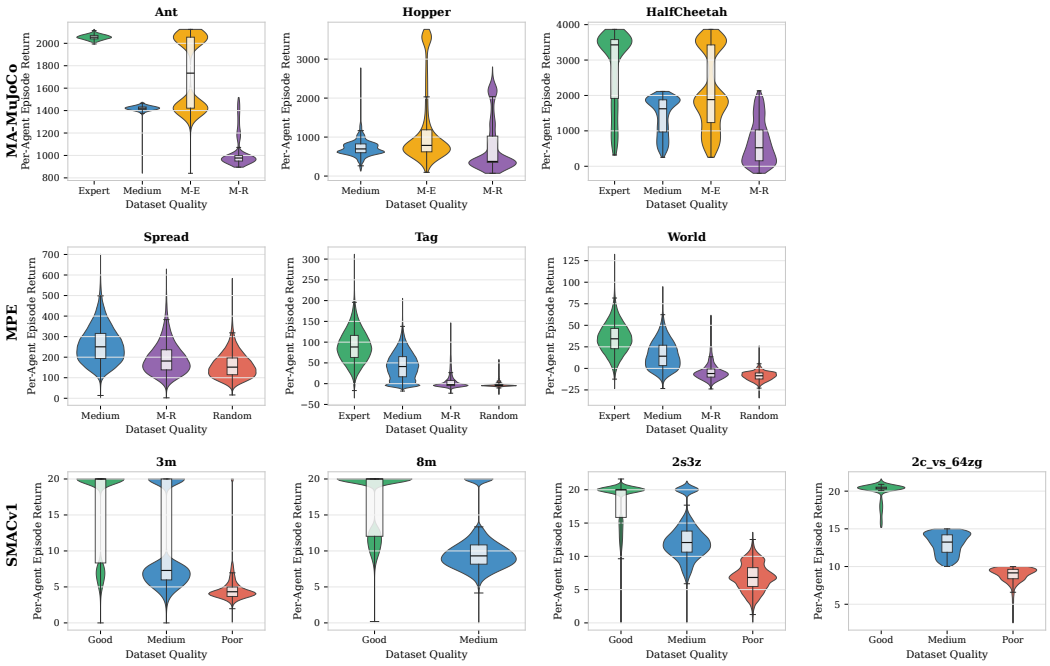


Figure 6: Per-agent average episode-return distributions of the offline datasets used by MEANMAC. Each violin corresponds to one task–dataset pair. The distributions show systematic differences across dataset qualities as well as substantial intra-dataset variation.

MPE normalized score. Following prior offline MARL evaluation protocols (Pan et al., 2022), we report normalized scores for MPE in Table 2. Let S denote the original episodic return. The normalized score is computed as

$$S_{\text{norm}} = 100 \times \frac{S - S_{\text{random}}}{S_{\text{expert}} - S_{\text{random}}}. \tag{51}$$

The expert and random reference scores for `Spread`, `Tag`, and `World` are $(516.8, 159.8)$, $(185.6, -4.1)$, and $(79.5, -6.8)$, respectively. Therefore, the MPE results in Table 2 should be interpreted as normalized returns rather than raw episodic returns.

Mamujoco MA-MuJoCo (Peng et al., 2021) decomposes a single MuJoCo robot into multiple agents, each controlling a subset of joints. Successful locomotion therefore requires agents to synchronize local controls to achieve stable global movement. We evaluate on `Ant`, `Hopper`, and `HalfCheetah`, corresponding to the implementation identifiers `2ant`, `3hopper`, and `6halfcheetah`, respectively. The offline datasets include four quality levels: `Expert`, `Medium`, `Medium-Expert`, and `Medium-Replay`. These datasets contain both high-quality locomotion trajectories and broader mixed-quality behaviors, making them suitable for testing whether a generative policy can model heterogeneous continuous-control coordination patterns. The return distributions of the MA-MuJoCo datasets are visualized in Figure 6.

SMACv1 SMACv1 (Samvelyan et al., 2019) provides cooperative micromanagement tasks in StarCraft II under partial observability and discrete action spaces. Each agent observes only local information, such as nearby allied and enemy units, health, distances, and available actions, while coordination requires joint strategies such as focus fire, positioning, and target selection. We evaluate on four maps: `3m`, `8m`, `2s3z`, and `2c_vs_64zg`. The first two maps are homogeneous-unit battles, `2s3z` contains heterogeneous allied units, and `2c_vs_64zg` is an asymmetric combat scenario. We use offline datasets from the off-the-grid benchmark (Formanek et al., 2023), with three quality levels: `Good`, `Medium`, and `Poor`. The return distributions of the SMACv1 datasets are visualized in Figure 6.

C.2 HYPERPARAMETERS

We report the hyperparameter settings used for MEANMAC below. Table 5 summarizes the shared training, MeanFlow-policy, actor, critic, and action-selection configurations. Tables 6–8 provide the task-specific settings for MPE, MA-MuJoCo, and SMACv1, respectively. Most optimization and network configurations are shared across domains. Only a small subset of hyperparameters, primarily the MeanFlow loss weight α and critic-ensemble aggregation rule, is adjusted for individual task–dataset pairs.

Table 5: **Hyperparameters for MeanMAC.**

Parameter	Value	Description
General Training Settings		
sequence_length	20	Replay trajectory length
sample_period	1	Replay sampling stride
buffer_size	2000000	Replay buffer capacity
batch_size	32	Training batch size
offline steps	500000	Number of gradient updates
discount γ	0.99	Discount factor
tau	0.005	Target-network update rate
optimizer	Adam	Optimizer
gradient clipping	1.0	Maximum gradient norm
MeanFlow Policy		
sigma	1.0	Gaussian noise scale
noise_type	gaussian	Noise distribution
adaptive_gamma	0.8	Adaptive L2 exponent
adaptive_c	10^{-4}	Numerical stabilizer
consistency_alpha	0.0	Consistency-loss weight
bound_loss_weight	1.0	Action-boundary penalty
tanh_squash	False	Tanh output squashing
use_output_layernorm	False	Output normalization
Actor Network		
actor_hidden_dims	256	Actor hidden dimension
actor_depth	3	Number of transformer blocks
actor_num_heads	2	Number of attention heads
actor_layer_norm	False	Actor layer normalization
actor lr	3×10^{-4}	Actor learning rate
actor lr schedule	constant	Learning-rate schedule
lr_min_ratio	0.1	Minimum cosine LR ratio
actor_bc_weight	0.0	Auxiliary BC-loss weight
Critic and Team-Value Learning		
value_hidden_dims	$(512, 512, 512, 512)$	Critic hidden dimensions
num_ensembles	4	Number of critic ensembles
layer_norm	True	Critic layer normalization
critic lr	3×10^{-4}	Critic learning rate
q_loss_weight	1.0	Q-guidance weight
q_agg	Task-dependent	See task-specific tables below
normalize_q_loss	True	Normalize Q-guidance loss
mixer_type	vdn	Team-value aggregation
team_mask_agg	min	Team terminal-mask aggregation
Best-of-K Action Selection		
action_mode	best	Value-guided candidate selection
num_candidates	Task-dependent	See task-specific tables below
MeanFlow Loss Weighting		
alpha	Task-dependent	See task-specific tables below
use_dynamic_alpha	False	Dynamic weighting disabled

Table 6: **Task-specific hyperparameter settings for MeanMAC on MPE.**

Task	Dataset	α	Q agg	Candidates	Mode	Lr	Batch size	Q weight
simple_spread	Random	3000	mean	5	best	3×10^{-4}	32	1.0
simple_spread	Medium-Replay	3000	mean	5	best	3×10^{-4}	32	1.0
simple_spread	Medium	3000	min	5	best	3×10^{-4}	32	1.0
simple_tag	Random	3000	mean	5	best	3×10^{-4}	32	1.0
simple_tag	Medium-Replay	3000	mean	5	best	3×10^{-4}	32	1.0
simple_tag	Medium	3000	mean	5	best	3×10^{-4}	32	1.0
simple_tag	Expert	3000	min	5	best	3×10^{-4}	32	1.0
simple_world	Random	3000	mean	5	best	3×10^{-4}	32	1.0
simple_world	Medium-Replay	3000	mean	5	best	3×10^{-4}	32	1.0
simple_world	Medium	3000	mean	5	best	3×10^{-4}	32	1.0
simple_world	Expert	3000	min	5	best	3×10^{-4}	32	1.0

Table 7: **Task-specific hyperparameter settings for MeanMAC on MA-MuJoCo.**

Task	Dataset	α	Q agg	Candidates	Mode	Lr	Batch size	Q weight
2ant	Expert	300	mean	5	best	3×10^{-4}	32	1.0
2ant	Medium	300	mean	5	best	3×10^{-4}	32	1.0
2ant	Medium-Expert	300	mean	5	best	3×10^{-4}	32	1.0
2ant	Medium-Replay	300	mean	5	best	3×10^{-4}	32	1.0
3hopper	Medium	3000	mean	5	best	3×10^{-4}	32	1.0
3hopper	Medium-Expert	50	mean	5	best	3×10^{-4}	32	1.0
3hopper	Medium-Replay	3000	mean	5	best	3×10^{-4}	32	1.0
6halfcheetah	Expert	300	mean	5	best	3×10^{-4}	32	1.0
6halfcheetah	Medium	300	mean	5	best	3×10^{-4}	32	1.0
6halfcheetah	Medium-Expert	300	mean	5	best	3×10^{-4}	32	1.0
6halfcheetah	Medium-Replay	300	mean	5	best	3×10^{-4}	32	1.0

Table 8: **Task-specific hyperparameter settings for MeanMAC on SMACv1.**

Task	Dataset	α	Q agg	Candidates	Mode	Lr	Batch size	Q weight
3m	Good	3000	mean	5	best	3×10^{-4}	32	1.0
3m	Medium	3000	mean	5	best	3×10^{-4}	32	1.0
3m	Poor	3000	mean	5	best	3×10^{-4}	32	1.0
8m	Good	3000	mean	5	best	3×10^{-4}	32	1.0
8m	Medium	3000	mean	5	best	3×10^{-4}	32	1.0
2s3z	Good	3000	mean	5	best	3×10^{-4}	32	1.0
2s3z	Medium	3000	mean	5	best	3×10^{-4}	32	1.0
2s3z	Poor	3000	mean	5	best	3×10^{-4}	32	1.0
2c_vs_64zg	Good	3000	mean	5	best	3×10^{-4}	32	1.0
2c_vs_64zg	Medium	3000	mean	5	best	3×10^{-4}	32	1.0
2c_vs_64zg	Poor	3000	mean	5	best	3×10^{-4}	32	1.0

C.3 Detailed Network Architecture

This section summarizes the network parameterization used by MeanMAC. The architecture follows the factorized policy design described in the main text: the actor and critic parameters are shared across agents, while agent identities are explicitly provided as part of the local input.

Agent-conditioned input. For each agent $i \in \{1, \dots, N\}$, the local observation o_i is augmented with a one-hot identity vector,

$$\tilde{o}_i = [o_i, \mathrm{id}_i] \tag{52}$$

where $\mathrm{id}_i \in \mathbb{R}^N$ identifies agent i. The augmented observation is used by both the actor and critic, allowing a single shared parameterization to represent agent-dependent behaviors.

Shared one-step MeanFlow actor. MeanMAC uses a shared conditional MeanFlow generator

$$g_\theta(\tilde{o}_i, a_i^t, t) \in \mathcal{A}_i \tag{53}$$

where a_i^t denotes the noisy/interpolated action input and t is the MeanFlow time variable. The actor is instantiated using an MFDiT-style network with hidden dimension 256, 3 residual blocks, and 2 attention heads. The output dimension matches the action representation of each agent.

At one-step generation, noise is directly mapped to an action through

$$a_i = g_\theta(\tilde{o}_i, \epsilon_i, 1), \qquad \epsilon_i \sim p_0 \tag{54}$$

where p_0 denotes the base noise distribution. Thus, each local action requires only one actor forward pass and does not rely on iterative denoising or numerical ODE integration.

The underlying actor defines a factorized base policy

$$\pi_\theta^{\mathrm{base}}(\mathbf{a} \mid \mathbf{o}) = \prod_{i=1}^{N} \pi_\theta(a_i \mid o_i, \mathrm{id}_i) \tag{55}$$

where $\mathbf{o} = (o_1, \dots, o_N)$ and $\mathbf{a} = (a_1, \dots, a_N)$. No centralized joint actor or explicit cross-agent attention is introduced inside the policy network.

Shared critic ensemble. MeanMAC maintains an ensemble of $E = 4$ shared local critics,

$$Q_\phi^e(\tilde{o}_i, a_i), \qquad e = 1, \dots, E \tag{56}$$

where each critic receives the augmented local observation and the corresponding action. Each critic uses four hidden layers of width 512 with layer normalization. The ensemble predictions are aggregated at the agent level as

$$q_i = \mathrm{Agg}_e \, Q_\phi^e(\tilde{o}_i, a_i) \tag{57}$$

where Agg_e denotes the ensemble aggregation operator used for the corresponding task. A delayed target critic ensemble with the same architecture is maintained for Bellman learning and value-guided joint-action evaluation.

Team-value aggregation. The agent-wise utilities are combined using an additive team-value function

$$Q_{\mathrm{team}}(\mathbf{o}, \mathbf{a}) = \frac{1}{N} \sum_{i=1}^{N} q_i \tag{58}$$

where q_i denotes the aggregated utility of agent i. The positive additive aggregation is monotonic with respect to every local utility and therefore satisfies the IGM consistency property described in Section 3. This architecture separates factorized one-step action generation from centralized team-value evaluation: coordination is introduced through the learning objective and joint-action evaluation rather than through a centralized policy network.

C.4 Overall Training Procedure

Algorithm 1 summarizes the complete training procedure of MeanMAC. Each update jointly optimizes the one-step MeanFlow policy with team-value guidance and Bellman critic learning, while value-guided joint-action selection is used to construct coordinated next-action candidates.

C.5 BASELINE ALGORITHMS

We compare MEANMAC with representative offline MARL baselines from three families: Gaussian/value-based offline MARL methods, diffusion-based generative policies, and flow-based generative policies. Whenever an official implementation is available, we follow the released code or its reported configuration.

Imitation and single-agent offline RL extensions. **`BC`** is a behavior cloning baseline that directly imitates dataset actions. For continuous-control tasks, we use a Gaussian policy; for discrete-control tasks, we use a categorical policy over legal actions. This baseline measures how much performance can be obtained from pure supervised imitation without value-guided policy improvement.

`Diffusion BC` is a diffusion-based behavior cloning baseline. Instead of directly regressing dataset actions, it learns a conditional denoising process that maps noisy actions to clean actions under the current observation. At inference time, actions are generated by reverse denoising from Gaussian noise. This baseline evaluates whether iterative generative imitation alone can capture multimodal action distributions without explicit Q-guidance.

`MA-TD3+BC`[1] (Fujimoto & Gu, 2021) extends TD3+BC to the multi-agent setting. It combines TD3-style actor–critic learning with a behavior-cloning regularizer, so that policy improvement is encouraged to remain close to the offline dataset. This baseline represents a strong Gaussian actor–critic method for offline continuous-control MARL.

`MABCQ`[2] (Jiang & Lu, 2023) is a multi-agent adaptation of BCQ-style offline policy learning. It restricts policy improvement to behavior-supported actions, thereby reducing the risk of selecting out-of-distribution actions under a learned critic. In discrete domains, the policy is represented as a categorical distribution and action selection is restricted by behavior-policy support.

`MACQL`[3] (Kumar et al., 2020) follows the conservative Q-learning principle in multi-agent offline RL. It penalizes Q-values on out-of-distribution actions to mitigate value overestimation and extrapolation error. This baseline tests whether conservative value regularization is sufficient for coordinated multi-agent policy improvement without a generative one-step policy.

Offline MARL methods. **`MAICQ`**[4] (Yang et al., 2021) is an implicit-constraint offline MARL method. It reduces extrapolation error by relying on dataset-supported state–action pairs when estimating and optimizing value functions. This method is included as a representative value-based offline MARL baseline.

`OMAR`[5] (Pan et al., 2022) combines conservative value learning with zeroth-order actor rectification. It searches for high-value candidate actions under the learned critic and uses the rectified actions to improve the actor. This baseline is particularly relevant because it also uses value-guided action improvement, but it does not learn a one-step MeanFlow generator.

`OMIGA`[6] (Wang et al., 2023) introduces implicit global-to-local value regularization for offline MARL. It regularizes local value learning through global coordination information, aiming to improve cooperative policy learning under fixed datasets. We include it as a strong value-regularized offline MARL baseline.

Generative MARL methods. **`MADiff`**[7] (Zhu et al., 2024) is a diffusion-based offline MARL method that models multi-agent behaviors through iterative denoising. It can capture complex multi-agent distributions, but requires multiple denoising steps during action or trajectory generation. This baseline represents the iterative diffusion family of generative MARL methods.

[1] https://github.com/sfujim/TD3_BC
[2] https://github.com/sfujim/BCQ
[3] https://github.com/aviralkumar2907/CQL
[4] https://github.com/YiqinYang/ICQ
[5] https://github.com/ling-pan/OMAR
[6] https://github.com/ZhengYinan-AIR/OMIGA
[7] https://github.com/zbzhu99/madiff

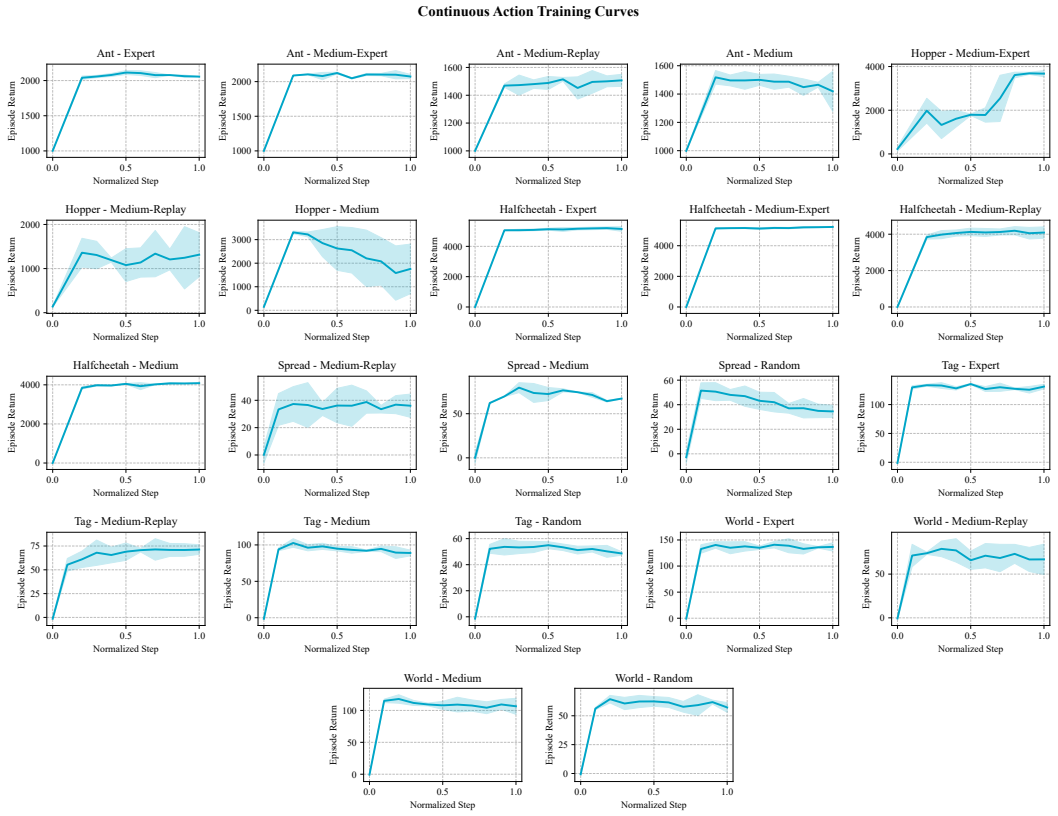


Figure 7: **Evaluation curves of MeanMAC on continuous-action tasks.** Episodic returns are evaluated periodically during offline training on MA-MuJoCo and MPE. The horizontal axis denotes normalized training progress.

DoF[8] (Li et al., 2025a) is a diffusion factorization framework for offline MARL. It factorizes centralized diffusion generation into decentralized multi-agent decision making, improving scalability over directly centralized diffusion policies. We compare against DoF to evaluate whether MeanMAC can obtain competitive coordination quality without iterative diffusion sampling.

MAC-Flow[9] (Lee et al., 2026) is a flow-matching based offline MARL baseline. It first learns a joint flow policy to model coordinated behaviors and then extracts one-step decentralized policies through distillation and value guidance. This is the closest flow-based baseline to MeanMAC. In contrast, MeanMAC directly trains a one-step MeanFlow policy with team-value guidance, avoiding the two-stage distillation pipeline.

C.6 Evaluation Curves During Training

We additionally report the evaluation returns measured periodically during offline training. All runs use the shared hyperparameters in Table 5 and the corresponding task-specific settings in Tables 6, 7, and 8.

C.7 Computing Resources

All experiments are conducted on a GPU computing server equipped with four NVIDIA A100 GPUs, including three NVIDIA A100 PCIe GPUs with 40 GB memory each and one NVIDIA A100 SXM4 GPU with 80 GB memory, providing 200 GB of aggregate GPU memory. The server uses NVIDIA driver version 580.126.09, with CUDA 13.0 reported by the runtime environment. Independent training runs are assigned to available GPUs according to memory requirements. Unless

[8] https://github.com/xmu-rl-3dv/DoF

[9] https://github.com/DongsuLeeTech/mac-flow

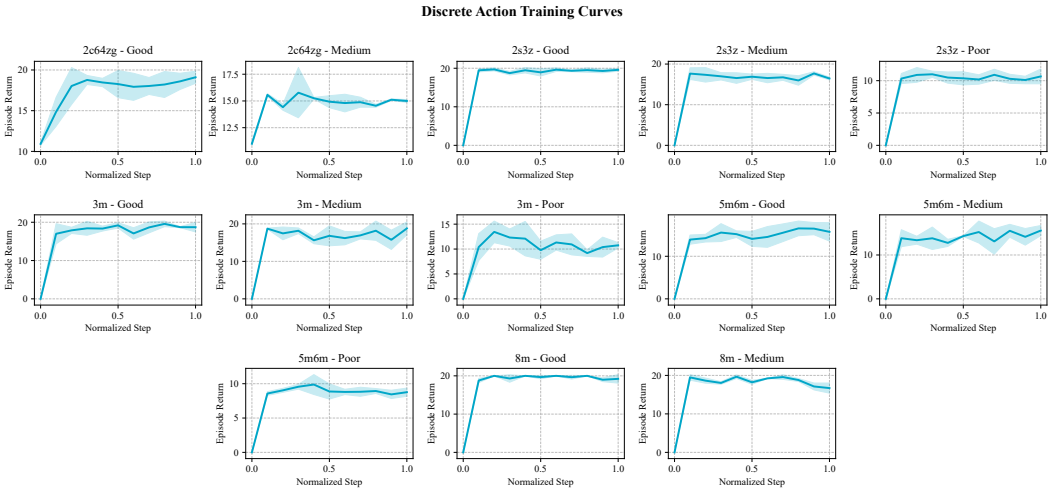


Figure 8: **Evaluation curves of MeanMAC on discrete-action tasks.** Episodic returns are evaluated periodically during offline training on SMACv1. The horizontal axis denotes normalized training progress.

otherwise specified, the main benchmark results are reported over five random seeds, while the ablation experiments are evaluated over four random seeds.

D Additional Results

D.1 Additional Offline-to-Online Results

We provide the complete offline-to-online evaluation across six task–dataset settings in Figure 9. The total training horizon is 1,000K steps, with online fine-tuning starting at normalized step 0.5, corresponding to 500K training steps. In addition to the `HalfCheetah` Medium and Medium-Replay results reported in the main text, we further evaluate MPE `Tag` Medium-Replay, MA-MuJoCo `Hopper` Medium-Replay, `HalfCheetah` Medium-Expert, and `HalfCheetah` Expert.

On `Tag` Medium-Replay, MeanMAC continues to improve after online interaction and maintains a large performance margin over MAC-Flow and FlowBC. On `Hopper` Medium-Replay, the return exhibits substantially larger fluctuations, yet MeanMAC generally remains above both baselines throughout the online phase. The additional `HalfCheetah` results show a similar trend: MeanMAC maintains a strong offline initialization on both Medium-Expert and Expert and continues to benefit from newly collected experience after online fine-tuning begins. Together with the main-text results, these experiments indicate that the same one-step generative policy can be continuously improved with online experience across different environments and offline dataset qualities.

D.2 Additional Coordination Ablations

We further provide the complete coordination ablation on MPE Medium-Replay in Figure 10. The full MeanMAC is compared with two variants: `w/o IGM`, which removes the IGM-compatible team-value factorization while retaining centralized joint-action value guidance, and `Independent`, which removes team-level coupling and optimizes each agent only through its local value objective. In addition to the `Tag` and `World` results discussed in the main text, the appendix includes `Spread` to examine whether the same effect persists across all three MPE tasks.

On `Spread`, the full model reaches a higher return and generally remains above both ablated variants throughout training. A similar separation is observed on `Tag`, where removing team-level coupling leads to the largest degradation, while `w/o IGM` retains intermediate performance. On `World`, the learning curves are more variable, but the full model still maintains stronger overall performance than the two variants. Across the three tasks, these results consistently support the role of team-level value coupling in shaping locally generated actions into coordinated joint behavior.

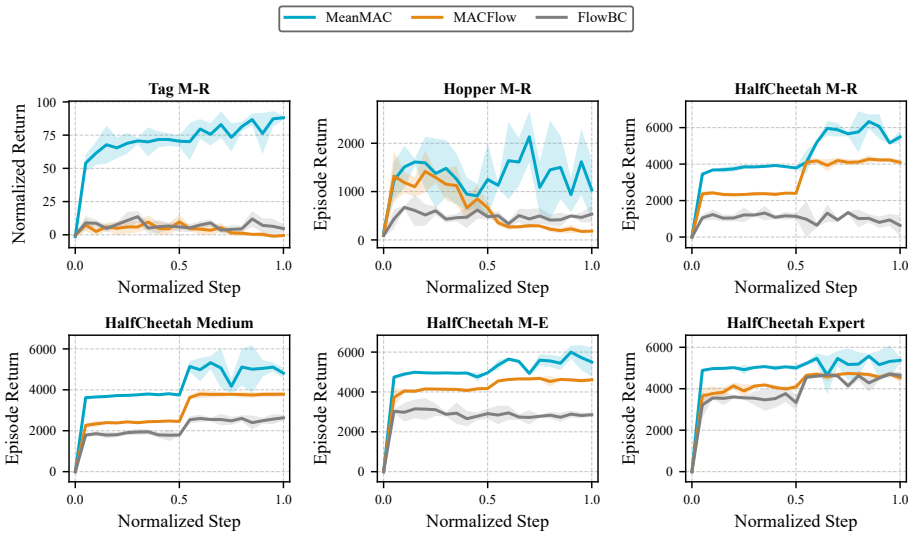


Figure 9: **Additional offline-to-online results.** Complete results on MPE `Tag` Medium-Replay and MA-MuJoCo `Hopper` Medium-Replay, `HalfCheetah` Medium-Replay, Medium, Medium-Expert, and Expert. Online fine-tuning starts at normalized step 0.5 (500K of 1,000K total training steps). Curves report mean ± standard deviation over four seeds.

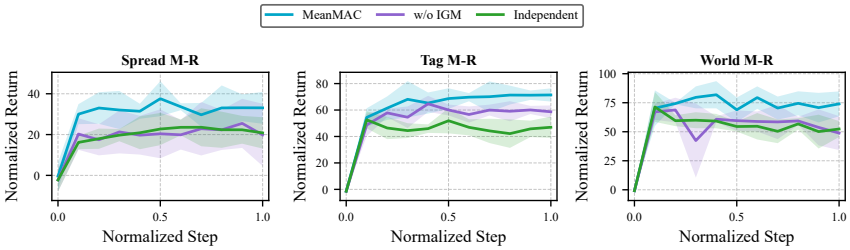


Figure 10: **Additional coordination ablations on MPE Medium-Replay.** We compare the full MEANMAC with `w/o IGM` and `Independent` on `Spread`, `Tag`, and `World`. Curves report mean ± standard deviation over four seeds.

D.3 SENSITIVITY TO THE MEANFLOW WEIGHT

We study the sensitivity of MEANMAC to the MeanFlow weight α on MPE `Tag` Medium-Replay, MPE `World` Medium-Replay, and MA-MuJoCo `HalfCheetah` Medium-Replay. As shown in Figure 11, the effect of α is task dependent. On `Tag`, relatively small values of α already yield strong performance, while larger values lead to a gradual reduction in return. `World` exhibits larger variation across different weights, with $\alpha = 3$ achieving the highest mean return but also substantially larger variance, whereas $\alpha = 300$ provides a more stable high-performing configuration. On `HalfCheetah`, increasing the MeanFlow weight substantially improves both performance and stability compared with $\alpha = 3$, with the best result obtained around $\alpha = 300$. Overall, these results indicate that the balance between team-value optimization and MeanFlow behavior modeling is important, while a single weight is not uniformly optimal across different domains.

D.4 SENSITIVITY TO THE NUMBER OF JOINT-ACTION CANDIDATES

We further analyze the number of joint-action candidates K used by value-guided action selection. As shown in Figure 12, using only one candidate consistently produces substantially weaker performance across all three tasks, highlighting the benefit of evaluating multiple complete joint actions with the team-value critic. Increasing K to 5 or 10 yields large improvements on `Tag`, `World`, and `HalfCheetah`. However, increasing the candidate set does not monotonically improve performance: `Tag` performs best around $K = 5$, `World` around $K = 10$, while `HalfCheetah` remains strong

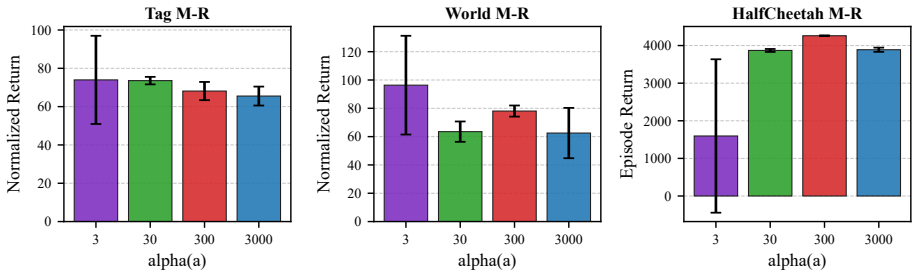


Figure 11: **Sensitivity to the MeanFlow weight α.** Results are reported on MPE `Tag` Medium-Replay, MPE `World` Medium-Replay, and MA-MuJoCo `HalfCheetah` Medium-Replay. Bars show mean $\pm$ standard deviation over four random seeds.

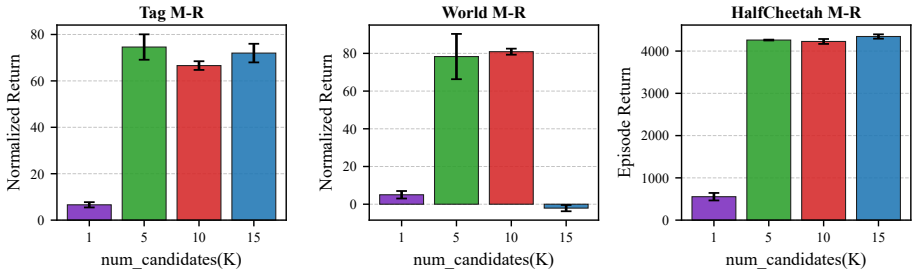


Figure 12: **Sensitivity to the number of joint-action candidates K.** We vary $K \in \{1, 5, 10, 15\}$ on MPE `Tag` Medium-Replay, MPE `World` Medium-Replay, and MA-MuJoCo `HalfCheetah` Medium-Replay. Bars show mean $\pm$ standard deviation over four random seeds.

for $K \in \{5, 10, 15\}$. In particular, the degradation on `World` at $K = 15$ suggests that an excessively large candidate set may make action selection more sensitive to critic estimation error. These results support using a moderate candidate set rather than maximizing K indiscriminately.

D.5 Training-Time Comparison

We compare the wall-clock training time of MeanMAC with MABCQ, DoF, and MAC-Flow on the SMACv1 `3m`, `2s3z`, and `8m` tasks. As shown in Figure 13, DoF requires substantially longer training across all three environments, whereas MeanMAC completes training within a much smaller time budget. MeanMAC incurs additional training cost relative to MABCQ and MAC-Flow due to one-step generative policy learning and team-value-guided joint-action evaluation, but the overhead remains moderate compared with the large computational cost of the diffusion-based DoF baseline. The training time of MeanMAC also increases only moderately across the three SMACv1 tasks, supporting the practical efficiency of direct one-step generative policy learning.

D.6 Policy Support Alignment

A central concern in offline generative Q-learning is that value maximization may push the learned policy toward unsupported state–action regions. To examine whether this happens in MeanMAC, we visualize samples from trained policy checkpoints and compare them with the original offline dataset using t-SNE. The visualization covers MPE `Tag`, MPE `World`, and MA-MuJoCo `6halfcheetah` across their corresponding dataset qualities.

As shown in Figure 14, policy samples largely remain within or near the support of the dataset distribution. On `Tag` and `World`, the generated samples occupy compact subregions of the offline data manifold, suggesting that the policy does not simply reproduce all behavior modes, but instead selects structured modes within the data support. On `6halfcheetah`, the policy samples closely follow the curved trajectory manifold visible in the dataset, especially under `Expert`, `Medium-Expert`, and `Medium` data. These results provide qualitative evidence that MeanFlow regularization helps keep the one-step policy aligned with offline behavior support, while the team-value objective guides the actor toward useful modes inside that support. The analysis is qualitative: t-SNE projections do

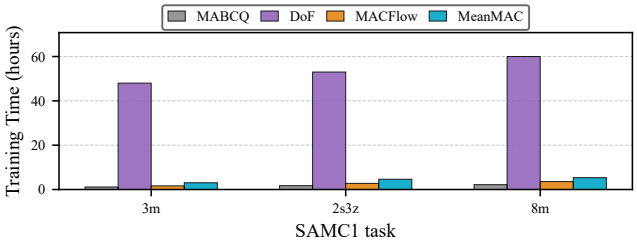


Figure 13: **Training-time comparison on SMACv1.** Wall-clock training time of MABCQ, DoF, MAC-Flow, and MEANMAC on `3m`, `2s3z`, and `8m`.

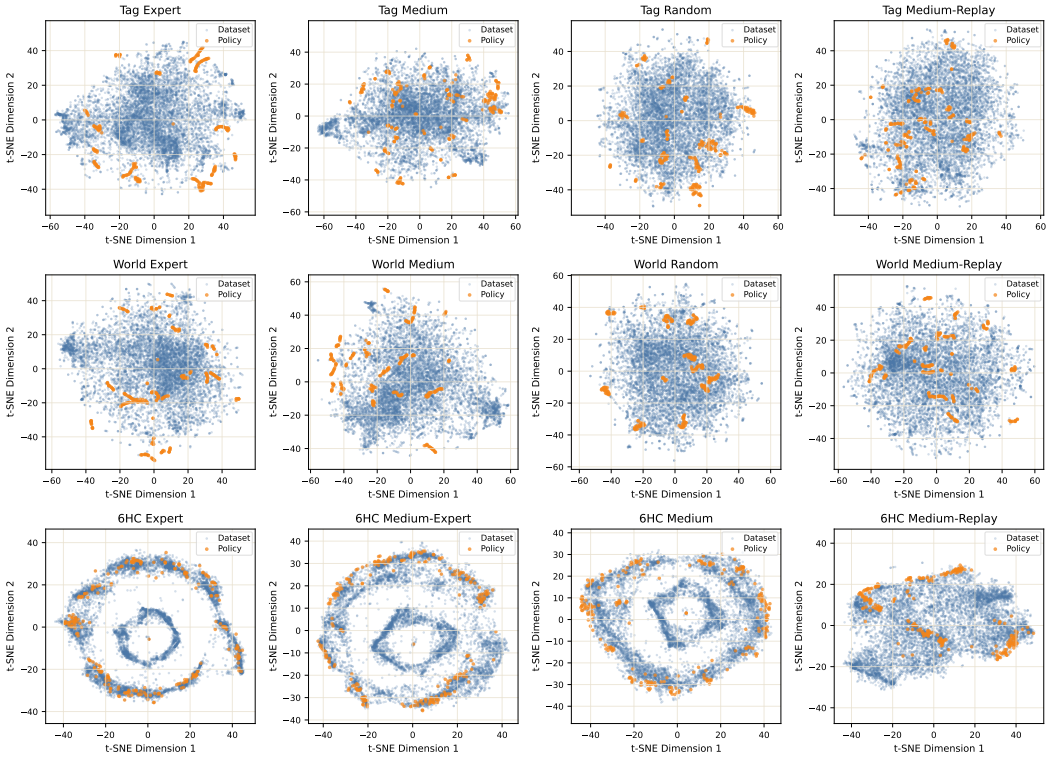


Figure 14: t-SNE visualization of dataset samples and MEANMAC policy samples from trained checkpoints. We compare the original offline data with generated policy samples on MPE `Tag`, MPE `World`, and MA-MuJoCo `6halfcheetah`. The policy samples mostly stay inside or near dataset-supported regions, suggesting value-guided in-distribution mode selection rather than unconstrained out-of-distribution extrapolation.

not prove strict support containment, but they help diagnose whether the learned policy visibly drifts away from the dataset manifold.

D.7 TEMPORAL COORDINATION PROGRESS ON MPE SPREAD

We further visualize the temporal evolution of coordination on MPE `Spread`. This analysis complements the main text by showing not only the initial and final states, but also intermediate coordination progress. Figure 15 shows snapshots at $t = 0$, $t = 3$, $t = 6$, and $t = 9$ under `Random`, `Medium`, and `Medium-Replay` datasets.

At $t = 0$, the assignments are marked as inconsistent: agents have not yet formed a clear one-to-one correspondence with landmarks. After a few execution steps, the behavior becomes consistent across all three dataset qualities. The agents move toward distinct landmarks and maintain more stable assignment patterns at later timesteps. This suggests that the learned one-step policies can produce

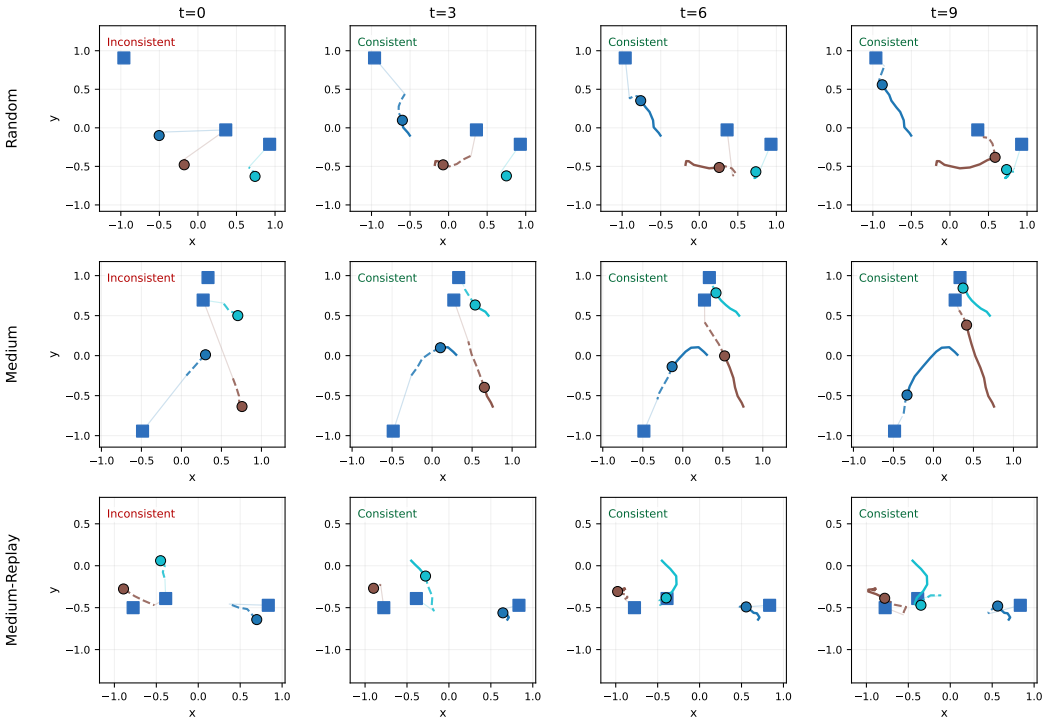


Figure 15: Temporal coordination progress on MPE `Spread`. We show policy rollouts at $t = 0$, $t = 3$, $t = 6$, and $t = 9$ under `Random`, `Medium`, and `Medium-Replay` datasets. The initial assignments are inconsistent, while later snapshots show consistent agent–landmark assignments.

temporally coherent multi-agent behavior, even though each agent executes locally. The result is consistent with the design of MEANMAC: the actor is factorized at the execution interface, but its training is shaped by a team-value objective that encourages compatible joint actions.

Algorithm 1: MEANMAC: One-Step Generative Multi-Agent Q-Learning

Data: Offline dataset $\mathcal{D}$, number of agents N, number of candidates K
while not converged **do**
  Sample $(\mathbf{o}, \mathbf{a}, \mathbf{r}, \mathbf{o}', \mathbf{m}) \sim \mathcal{D}$ `# Offline transition`
  for $i = 1, \dots, N$ **do**
    $\tilde{o}_i \leftarrow [o_i, \mathrm{id}_i]$, $\tilde{o}'_i \leftarrow [o'_i, \mathrm{id}_i]$ `# Agent-conditioned observations`

  `# ▷ MeanFlow Modeling`
  for $i = 1, \dots, N$ **do**
    $t \sim p(t)$, $\epsilon_i \sim p_0$ `# Sample time and noise`
    $a_i^t \leftarrow (1-t)a_i + t\epsilon_i$, $v_i \leftarrow \epsilon_i - a_i$ `# Noised action and velocity`
    $(g_i, dgdt_i) \leftarrow \mathrm{jvp}\big(g_\theta(\tilde{o}_i, \cdot, \cdot), (a_i^t, t), (v_i, 1)\big)$ `# MeanFlow derivative`
    $g_i^{\mathrm{tgt}} \leftarrow a_i^t + (t-1)v_i - t\, dgdt_i$ `# MeanFlow target`
    $err_i \leftarrow g_i - \mathrm{sg}(g_i^{\mathrm{tgt}})$ `# Regression error`

  $\mathcal{L}_{\mathrm{MF}} \leftarrow \dfrac{1}{N}\sum_{i=1}^{N} \mathrm{metric}(err_i)$ `# MeanFlow loss`

  `# ▷ Team-Value Policy Update`
  for $i = 1, \dots, N$ **do**
    $\epsilon_i^\pi \sim p_0$, $a_i^\pi \leftarrow g_\theta(\tilde{o}_i, \epsilon_i^\pi, 1)$ `# One-step local action`
    $q_i^\pi \leftarrow \mathrm{Agg}_e\, Q_\phi^e(\tilde{o}_i, a_i^\pi)$ `# Local utility`

  $Q_{\mathrm{team}}^\pi \leftarrow \dfrac{1}{N}\sum_{i=1}^{N} q_i^\pi$ `# Team value`
  $\mathcal{L}_\pi \leftarrow -\lambda_Q Q_{\mathrm{team}}^\pi$ `# Team-value Q loss`
  $\mathcal{L}_{\mathrm{Actor}} \leftarrow \mathcal{L}_\pi + \alpha\mathcal{L}_{\mathrm{MF}}$ `# Actor objective`

  `# ▷ Critic Update`
  $\mathbf{a}'^\star \leftarrow \mathtt{SelectJoint}(\mathbf{o}', K; \theta, \bar{\phi})$ `# Value-guided joint action`
  for $i = 1, \dots, N$ **do**
    $q'_i \leftarrow \mathrm{Agg}_e\, Q_{\bar{\phi}}^e(\tilde{o}'_i, a_i'^\star)$ `# Target local utility`

  $y \leftarrow \mathrm{sg}\left[\dfrac{1}{N}\sum_{i=1}^{N}\big(r_i + \gamma m_i q'_i\big)\right]$ `# Bellman target`
  for $e = 1, \dots, E$ **do**
    $Q_{\mathrm{team}}^e \leftarrow \dfrac{1}{N}\sum_{i=1}^{N} Q_\phi^e(\tilde{o}_i, a_i)$ `# Team value of critic` e

  $\mathcal{L}_Q \leftarrow \dfrac{1}{E}\sum_{e=1}^{E}\left(Q_{\mathrm{team}}^e - y\right)^2$ `# Bellman Q loss`

  $\mathcal{L}_{\mathrm{MeanMAC}} \leftarrow \mathcal{L}_Q + \mathcal{L}_{\mathrm{Actor}}$ `# Overall objective`
  Update θ, ϕ to minimize $\mathcal{L}_{\mathrm{MeanMAC}}$ `# Joint optimization`
  $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$ `# Target critic update`

Function `SelectJoint` $(\mathbf{o}, K; \theta, \bar{\phi})$ **:**
  for $k = 1, \dots, K$ **do**
    for $i = 1, \dots, N$ **do**
      $\tilde{o}_i \leftarrow [o_i, \mathrm{id}_i]$, $\epsilon_i^{(k)} \sim p_0$
      $a_i^{(k)} \leftarrow g_\theta(\tilde{o}_i, \epsilon_i^{(k)}, 1)$ `# One-step generation`
      $q_i^{(k)} \leftarrow \mathrm{Agg}_e\, Q_\phi^e(\tilde{o}_i, a_i^{(k)})$ `# Candidate utility`
    $\mathbf{a}^{(k)} \leftarrow (a_1^{(k)}, \dots, a_N^{(k)})$
    $S_k \leftarrow \dfrac{1}{N}\sum_{i=1}^{N} q_i^{(k)}$ `# Joint candidate value`
  $k^\star \leftarrow \arg\max_k S_k$ `# Select one complete candidate`
  return $\mathbf{a}^{(k^\star)}$