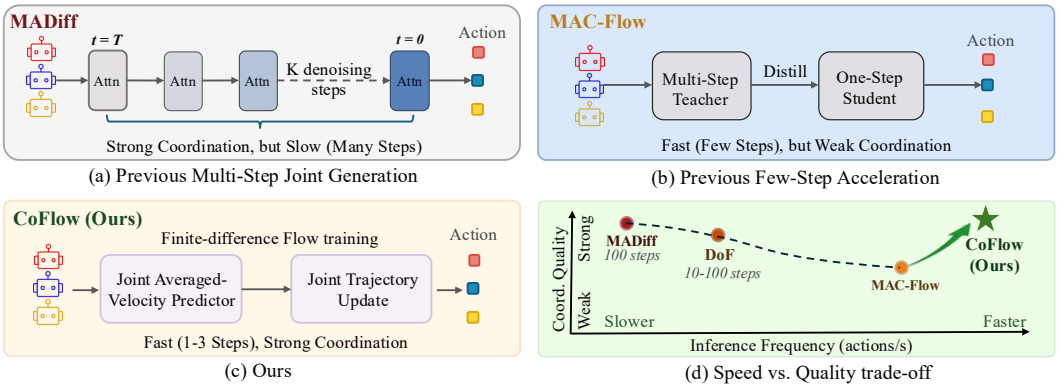
CoFlow: Coordinated Few-Step Flow for Offline Multi-Agent Decision Making

Anonymous authors
Paper under double-blind review

Abstract

How to achieve both fast joint trajectory generation and effective coordination in generative offline multi-agent reinforcement learning? Multi-agent diffusion models generate joint trajectories through iterative sampling but incur high inference latency. Distillation reduces sampling steps, but it may cause the student model to lose information about the cross-agent dependencies learned by the teacher, leading to a lower coordination success rate. A directly trained joint generator can model coordination among agents during one-step or few-step generation. Nonetheless, insufficient modeling of cross-agent dependencies may still impair coordination. Furthermore, its consistency training process introduces heavy computational and memory costs. In this paper, we propose **Co**ordinated few-step **Flow** (CoFlow), which directly learns a joint averaged-velocity field for distillation-free one-step and few-step multi-agent trajectory generation. CoFlow uses Coordinated Velocity Attention (CVA) to incorporate teammate trajectory information for more accurate prediction of each agent's trajectory. To reduce the cost of consistency training for this joint model, CoFlow approximates the correction term using finite differences. Across 48 configurations on MPE and SMAC, CoFlow supports centralized and decentralized execution. Under centralized execution, our CoFlow outperforms our reproduced baseline by 11.2%, averaging relative gains equally across both benchmark suites, with a 12.93× speedup in model sampling. Its CVA improves normalized scores by 165.1% on average over the same model with cross-agent attention disabled. Compared with the implementation using exact derivatives, finite-difference training achieves a 1.78× speedup and reduces peak GPU memory by 41.1%. **Code is provided in the supplementary material.**



(a) Previous Multi-Step Joint Generation (b) Previous Few-Step Acceleration

(c) Ours (d) Speed vs. Quality trade-off

Figure 1: The dilemma between quality and efficiency in offline multi-agent trajectory generation. **(a)** MADiff-style multi-step joint generation preserves coordination through many denoising steps. **(b)** MAC-Flow distills a joint teacher into one-step students, requiring preservation of information about cross-agent dependencies during compression. **(c)** Our CoFlow combines joint averaged-velocity learning, CVA coupling, and finite-difference training. **(d)** CoFlow shifts the trade-off between speed and quality against representative diffusion and acceleration baselines.

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.

1 INTRODUCTION

Generative policies have become an important paradigm in offline multi-agent reinforcement learning (MARL), modeling cooperative behavior through joint generation of agents' trajectories (Zhu et al., 2024; Yuan et al., 2025; Li et al., 2025; Lu et al., 2025). Existing multi-agent diffusion (Ho et al., 2020; Song et al., 2021b) and flow-matching policies (Lipman et al., 2023; Liu et al., 2023) typically use multi-step iterative sampling, whose effectiveness has been demonstrated (Zhu et al., 2024; Li et al., 2025; Lee et al., 2026). As illustrated in Figure 1(a), the model repeatedly updates the joint trajectory to model cross-agent dependencies during generation, but repeated network evaluations incur substantial inference costs (Janner et al., 2022; Wang et al., 2023b; Chi et al., 2023). Therefore, reducing the number of generation steps while maintaining effective coordination is a central challenge for generative multi-agent decision making.

To reduce multi-step sampling costs (Salimans & Ho, 2022; Song et al., 2023), existing approaches distill a joint teacher into per-agent students, enabling generation in fewer steps (Lee et al., 2026), as illustrated in Figure 1(b). However, this compression must preserve individual trajectory quality and information about the cross-agent dependencies learned by the teacher (Lee et al., 2026). If information about these dependencies is lost, trajectories generated independently by multiple agents may fail to work together effectively, impairing cooperative task performance (Barde et al., 2024; Hernandez-Leal et al., 2019).

To avoid losing information about cross-agent dependencies during distillation, one route is to extend existing direct few-step generation methods (Frans et al., 2025; Zou et al., 2026b) to joint multi-agent trajectory modeling. These methods learn to generate through one or a few updates without training a separate teacher (Geng et al., 2025; 2026; Guo et al., 2025). Direct joint learning requires teammate information to coordinate trajectories within a limited number of updates. Training the joint averaged-velocity model also incurs computational and memory costs through its consistency objective (Geng et al., 2025; 2026). Few-step multi-agent generation must therefore address both the modeling of cooperative dependencies and the efficiency of training.

To address these challenges, we propose **Co**ordinated few-step **Flow** (**CoFlow**), a distillation-free framework for offline multi-agent decision making, with its core design illustrated in Figure 1(c). CoFlow applies averaged-velocity generation to joint multi-agent trajectory learning, enabling the model to generate all agents' trajectories through one or a few updates. Within this joint generation framework, we address cooperative modeling and training efficiency. First, *Coordinated Velocity Attention* (CVA) allows each agent to use teammate trajectory information when generating its own trajectory, modeling cross-agent dependencies within every update. Second, we approximate the consistency correction to reduce the training computation and memory required by this joint averaged-velocity model. CoFlow supports both centralized and decentralized execution. Figure 1(d) compares generation speed and task performance across methods.

Our contributions are threefold. **(1) Joint few-step generation framework**: we propose CoFlow for offline multi-agent decision making, directly learning a joint trajectory generation model for one-step or few-step generation without teacher distillation. **(2) Cross-agent coordination modeling**: we design Coordinated Velocity Attention to model cross-agent dependencies using teammate trajectory information at each generation update. A gated residual mechanism regulates the influence of this information on each agent's prediction. **(3) Efficient consistency training**: we use finite differences to approximate the consistency correction, reducing the per-update computation and memory costs of training the joint averaged-velocity model.

2 RELATED WORK

Offline multi-agent RL. Offline MARL aims to learn cooperative policies from fixed logged datasets (Levine et al., 2020). Dataset and baseline resources support research in this setting (Formanek et al., 2023), while standardised baselines and evaluation protocols are important for reliable comparisons (Formanek et al., 2024). Value-based methods regularize Q-values to mitigate distribution shift, including MA-ICQ (Yang et al., 2021), MA-CQL and OMAR (Pan et al., 2022), and OMIGA (Wang et al., 2023a). Recent work scales these to larger teams via low-rank interaction structure, in-sample sequential updates, and per-agent score decomposition (Zhan et al., 2025; Liu et al., 2025; Qiao et al., 2026). Sequence-modeling baselines such as the multi-agent decision transformer (Meng et al.,

2023) and policy-regularization methods (Fujimoto & Gu, 2021; Kostrikov et al., 2022) complete the comparison set. These methods usually parameterize per-agent Gaussian or autoregressive policies, motivating generative models for multimodal joint-action distributions in coordinated behavior.

Single-agent few-step / one-step generation. Generative trajectory models have become the leading paradigm for single-agent offline RL, covering planning, policy learning, energy guidance, and visuomotor control (Janner et al., 2022; Wang et al., 2023b; Hansen-Estruch et al., 2023; Chen et al., 2023; Ren et al., 2025; Zou et al., 2026a; Lu et al., 2023; Chi et al., 2023), all rooted in denoising diffusion probabilistic models (DDPM) and score-based generative modeling (Ho et al., 2020; Song et al., 2021b). Flow matching (Lipman et al., 2023; Liu et al., 2023) reaches parity via Flow Q-learning, energy-weighted flow matching, ReinFlow, and Flow Policy Optimization (FPO) (Park et al., 2025; Zhang et al., 2025a;b; McAllister et al., 2026). Reducing the ∼20 sampling steps these methods require is an active subfield. *Distillation* compresses a multi-step teacher into a one-step student via Consistency Models, Progressive Distillation, and denoising diffusion implicit models (DDIM) (Song et al., 2023; Salimans & Ho, 2022; Song et al., 2021a). *Direct one-step parameterizations* integrate the trajectory in a single forward pass via averaged velocity fields, shortcut reparameterization, dispersive regularization, and latent / discrete variants (Geng et al., 2025; Frans et al., 2025; Zou et al., 2026b; Dao et al., 2024; Gat et al., 2024; Rombach et al., 2022). These methods now match multi-step quality at 1–3 steps in single-agent settings, motivating a multi-agent counterpart with an explicit inter-agent mechanism.

Multi-agent few-step / one-step generation. Existing multi-agent generative methods sit at one of two extremes. The *diffusion family* preserves or factorizes joint generation but inherits multi-step inference cost. MADiff (Zhu et al., 2024) embeds cross-agent attention into diffusion-based planning, MADiTS (Yuan et al., 2025) stitches high-quality coordination segments, and DoF (Li et al., 2025) factorizes a centralized diffusion via independence-conditioned decomposition. Lu et al. (2025) find that unconditional sampling with selection often beats guided sampling. The *few-step flow family* reduces sampling cost: MAC-Flow (Lee et al., 2026) distills a joint flow teacher into per-agent students. CoFlow learns a joint averaged-velocity field directly, combining cross-agent coupling within each generation step with finite-difference consistency training.

3 METHODOLOGY

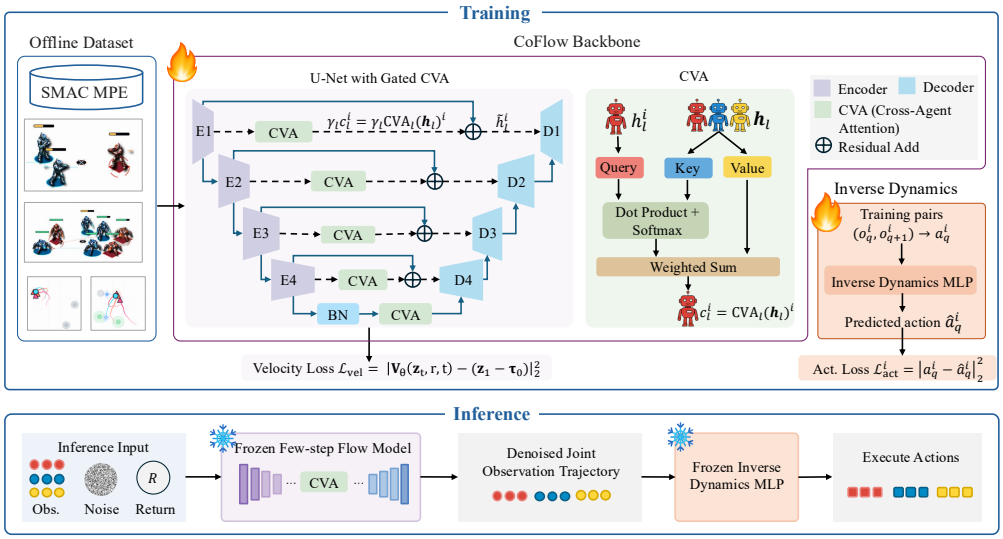


Figure 2: CoFlow training and inference overview. During training, the shared U-Net backbone injects gated CVA features into the skip paths to learn a joint velocity field, while a separate inverse-dynamics multilayer perceptron (MLP) is trained from offline observation pairs. During inference, the frozen few-step flow model predicts a denoised joint observation trajectory, and the frozen inverse-dynamics MLP maps the first predicted next observation to executable actions. The loss panels show unnormalized per-example squared errors for illustration. The full objectives in Equations (8) and (9) include averaging and use legal-action cross-entropy for discrete control.

CoFlow learns a joint averaged-velocity field for few-step multi-agent trajectory generation. CVA models cross-agent dependencies, while finite-difference consistency training reduces the cost of training this joint field. Training matches the network prediction plus a detached correction to the conditional velocity. Inference uses the averaged prediction alone. Figure 2 shows the architecture, with background and derivations in Sections B.1 and B.3.

3.1 Joint Averaged-Velocity Generation

From fixed offline trajectories, conditioned on observation history and target return, we construct the joint training pair (conditioning arguments are suppressed):

$$\mathbf{z}_t = (1-t)\boldsymbol{\tau}_0 + t\mathbf{z}_1, \qquad \mathbf{v}_c = \mathbf{z}_1 - \boldsymbol{\tau}_0, \tag{1}$$

where $\boldsymbol{\tau}_0$ is a clean joint observation trajectory, $\mathbf{z}_1$ is independent standard Gaussian noise, and $\mathbf{v}_c$ is the sample-conditional velocity. Bold symbols stack N agent blocks of dimension d. Superscript i selects block i, e.g., $z_t^i = [\mathbf{z}_t]^i$. Flow time runs from data ($t = 0$) to noise ($t = 1$).

CoFlow learns the joint interval-averaged velocity

$$\mathbf{u}(\mathbf{z}, r, t) = \frac{\mathbf{z} - \Phi_{r\leftarrow t}(\mathbf{z})}{t - r}, \qquad r < t, \tag{2}$$

where r is the reference time and $\Phi_{r\leftarrow t}$ transports the joint state from t to r. The field $\mathbf{u}$ and its diagonal limit, the instantaneous marginal velocity $\mathbf{v}$, are theoretical quantities. Their network approximations carry subscript θ.

We learn this interval displacement through conditional-velocity supervision and the MeanFlow consistency relation (Geng et al., 2025).

3.2 Gated Coordinated Velocity Attention

We use one shared temporal U-Net for the agent streams. CVA attends across agents and updates each encoder skip before decoder fusion:

$$c_l^i = \mathrm{CVA}_l(\boldsymbol{h}_l)^i, \qquad \tilde{h}_l^i = h_l^i + \gamma_l c_l^i, \tag{3}$$

where $h_l^i, \tilde{h}_l^i$ are agent i's layer-l skips before/after CVA, c_l^i is the message, and γ_l is a zero-initialized scalar gate.

Shared multi-head attention incorporates the U-Net conditioning embeddings. The decoder fuses the enhanced skips with its current features, and the bottleneck uses the same gated operation. Zero initialization gives an independent backbone. Learned gates activate teammate information within each evaluation. Thus $u_\theta^i = [\mathbf{u}_\theta]^i$ can depend on the full joint input. See Section B.6 for the decomposition.

3.3 Finite-Difference Consistency Training

Here, consistency training enforces the relationship between averaged and instantaneous velocities: the averaged velocity plus its interval-scaled derivative along the flow trajectory should match the instantaneous velocity. We use finite differences to approximate this derivative when constructing the training correction.

Along a marginal flow trajectory, $(t-r)\mathbf{u}(\mathbf{x}_t, r, t) = \mathbf{x}_t - \mathbf{x}_r$. Differentiating at fixed r yields

$$\frac{d}{dt}\left[(t-r)\mathbf{u}(\mathbf{x}_t, r, t)\right] = \frac{d}{dt}(\mathbf{x}_t - \mathbf{x}_r) = \mathbf{v}(\mathbf{x}_t, t), \tag{4}$$

where $\mathbf{x}_t$ follows the marginal ordinary differential equation (ODE) driven by $\mathbf{v}$. The reference state $\mathbf{x}_r$ is fixed.

The product and chain rules give the joint MeanFlow identity:

$$\begin{aligned} \mathbf{u} + (t-r)\frac{d\mathbf{u}}{dt} &= \mathbf{v}, \\ \frac{d\mathbf{u}}{dt} &= (\nabla_{\mathbf{x}}\mathbf{u})\mathbf{v} + \partial_t \mathbf{u}, \end{aligned} \tag{5}$$

where $\nabla_{\mathbf{x}}\mathbf{u}$ is the full joint Jacobian, including cross-agent blocks. All fields are evaluated along $\mathbf{x}_t$. The partial derivative ∂_t is taken with r and the state held fixed.

Using $\mathbf{v}_c$ as the joint training tangent, we approximate the derivative using a finite difference (FD):

$$\boldsymbol{\delta}_{\mathrm{FD}} \coloneqq \mathrm{stopgrad}\left[\frac{\mathbf{u}_\theta(\mathbf{z}_t + \varepsilon\mathbf{v}_c, r, t+\varepsilon) - \mathbf{u}_\theta(\mathbf{z}_t, r, t)}{\varepsilon}\right], \tag{6}$$

where $\mathbf{u}_\theta$ predicts the joint averaged velocity, θ includes the gates, and the signed step satisfies $\varepsilon \neq 0$ and $r \leq t + \varepsilon \leq 1$. The data-derived tangent $\mathbf{v}_c$ is held fixed. Shifting state and time together follows the direction $(\mathbf{v}_c, 1)$ in Equation (5). Since $\mathbf{v}(\mathbf{z}_t, t) = \mathbb{E}[\mathbf{v}_c \mid \mathbf{z}_t, t]$, sampled tangents yield a stochastic surrogate.

We therefore match the averaged prediction plus its interval-scaled correction to the instantaneous-velocity target:

$$\mathbf{V}_\theta \coloneqq \mathbf{u}_\theta(\mathbf{z}_t, r, t) + (t-r)\boldsymbol{\delta}_{\mathrm{FD}}, \tag{7}$$

$$\mathcal{L}_{\mathrm{vel}}(\theta) = \frac{1}{Nd}\mathbb{E}\,\|\mathbf{V}_\theta - \mathbf{v}_c\|_2^2, \tag{8}$$

where uppercase $\mathbf{V}_\theta$ is the corrected training prediction and lowercase $\mathbf{v}_c$ is the data-derived target. The expectation averages trajectories, noise, and time pairs, with coordinate normalization $1/(Nd)$.

With stopgrad, only the base prediction receives gradients. Its value is reused in the quotient, and the shifted query needs no backward graph. At $r = t$, the correction vanishes and the shifted query is skipped. The finite-difference correction is evaluated on the CVA-coupled joint field, so it incorporates cross-agent interactions within the same trajectory update. The Jacobian-vector product (JVP) relation and Taylor bound are in Sections B.3 and B.5.

3.4 Action Decoding and Inference

Because the flow generates observation trajectories, a separate three-layer inverse-dynamics multilayer perceptron (MLP) maps consecutive observations to executable actions. Its loss and the joint objective are

$$\mathcal{L}_{\mathrm{act}}(\phi) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[\ell_{\mathrm{act}}\left(a_q^i, I_\phi^i(o_q^i, o_{q+1}^i); \mathcal{U}_q^i\right)\right], \tag{9}$$

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{\mathrm{vel}}(\theta) + \lambda\mathcal{L}_{\mathrm{act}}(\phi), \qquad \lambda = 1, \tag{10}$$

where I_ϕ^i is the inverse-dynamics predictor with parameters ϕ, o_q^i, a_q^i denote observations and actions at environment step q, and λ weights action learning. The loss ℓ_{act} is continuous-action mean squared error (MSE) or cross-entropy over legal actions $\mathcal{U}_q^i$.

The interval length converts averaged velocity to displacement. Subtraction moves from noise toward data. Starting at $\mathbf{z}_{t_K} = \mathbf{z}_1$, CoFlow runs K joint updates, then decodes each agent's action from its output block:

$$\begin{aligned} \mathbf{z}_{t_{k-1}} &= \mathbf{z}_{t_k} - (t_k - t_{k-1})\hat{\mathbf{u}}_\theta(\mathbf{z}_{t_k}, t_{k-1}, t_k), \\ a_q^i &= \mathrm{Dec}_i\left(I_\phi^i(o_q^i, \hat{o}_{q+1}^i); \mathcal{U}_q^i\right), \end{aligned} \tag{11}$$

where $1 = t_K > \cdots > t_0 = 0$, $k = K, \ldots, 1$, and $\hat{\mathbf{u}}_\theta$ combines return-conditioned and return-dropped predictions for guidance. The next observation $\hat{o}_{q+1}^i$ is read from block i of the generated trajectory. The decoder returns a continuous action or the highest-logit legal action.

For $K = 1$, the generated trajectory is $\hat{\boldsymbol{\tau}}_0 = \mathbf{z}_1 - \hat{\mathbf{u}}_\theta(\mathbf{z}_1, 0, 1)$. The correction is used only in training. CoFlow-C (centralized execution) uses joint observations to generate a joint trajectory and decode all agents' actions. CoFlow-D (decentralized execution) performs joint trajectory prediction conditioned on each agent's local observations, with unavailable teammate context masked. Cross-agent attention models teammate behavior within each local prediction, and each agent executes only its own decoded action. Guidance and algorithms are in Sections B.3 and B.4. Proofs are in Sections B.5 to B.7.

4 Experiments

We organize the evaluation around three linked research questions (RQ):

RQ1. How well does CoFlow perform on continuous and discrete MARL benchmarks?

Table 1: Main results across MPE and SMAC. CoFlow entries report mean ± standard deviation across three random seeds using 1–3 sampling steps (Section 4.1). MPE uses OMAR-normalized scores. SMAC uses episodic returns. *Dataset* gives the offline-data mean. Bold and underline mark the best and second-best scores before display rounding.

(a) MPE, OMAR-normalized score (↑)

	Dataset		*Gaussian / Value-based*					*Diffusion*	*Flow (Ours)*	
Task	Quality	Data	BC	MA-ICQ	MA-TD3+BC	MA-CQL	OMAR	MADiff-C	CoFlow-C	CoFlow-D
Spread	Expert	96.2	35.0±2.6	104.0±3.4	108.3±3.9	98.2±5.2	114.9±2.6	95.0±5.3	**114.9±3.5**	100.0±10.5
	Md-Replay	9.3	10.0±3.8	13.6±5.7	15.4±5.6	31.4±7.2	37.9±6.1	30.3±2.5	**42.9±5.3**	24.0±3.2
	Medium	27.0	31.6±4.8	29.3±5.5	39.4±3.6	34.1±7.2	47.9±18.9	**64.9±7.7**	57.6±1.2	48.9±2.9
	Random	0.1	−0.5±3.2	6.3±3.5	9.8±4.9	24.0±9.8	34.4±5.3	6.9±3.1	**45.5±8.5**	10.8±5.0
Tag	Expert	89.5	40.0±9.6	113.0±14.4	115.2±12.8	119.3±14.0	123.9±10.5	**168.8±13.1**	137.2±8.7	124.8±2.4
	Md-Replay	8.6	0.9±1.4	34.5±27.8	28.7±20.9	41.7±15.3	47.1±15.3	**98.8±11.3**	66.8±1.7	75.0±12.0
	Medium	43.8	22.5±1.8	63.3±20.0	65.1±29.5	61.7±23.1	66.7±23.2	**133.5±20.2**	110.3±6.5	86.0±10.5
	Random	0.0	1.2±0.5	2.2±1.3	6.0±2.1	11.1±2.8	11.1±2.8	10.6±3.7	**30.6±6.0**	7.0±1.9
World	Expert	93.8	33.0±9.9	109.5±22.8	110.3±21.3	119.8±28.1	110.4±25.7	109.3±15.4	**156.6±9.5**	128.1±10.0
	Md-Replay	11.2	2.3±1.5	12.0±9.1	17.4±8.1	19.3±18.3	42.9±19.5	19.8±6.2	**43.6±10.4**	42.6±4.6
	Medium	54.5	25.3±2.0	71.9±20.0	73.4±9.3	58.6±11.2	74.6±11.5	84.7±12.3	**128.8±12.8**	115.3±7.2
	Random	0.0	−2.4±0.5	1.0±3.2	2.8±5.5	0.6±2.0	5.9±5.2	6.1±2.4	1.3±3.1	**23.5±12.7**

(b) SMAC, episode return (↑)

	Dataset		*Gaussian / Value-based*			*Transformer*	*Diffusion*	*Flow*		*Flow (Ours)*	
Task	Quality	Mean	BC	MA-ICQ	MA-CQL	MADT	MADiff-C	DoF	MAC-Flow	CoFlow-C	CoFlow-D
3m	Good	16.5	16.0±1.0	18.8±0.6	19.0±0.3	19.6±0.7	19.3±0.5	**19.8±0.2**	**19.8±0.2**	18.8±4.2	17.3±0.8
	Medium	10.0	8.2±0.8	18.1±0.7	18.9±0.7	17.2±0.7	16.4±2.6	18.6±1.2	18.0±3.2	**20.0±4.2**	12.8±0.9
	Poor	4.7	4.4±0.1	**14.4±1.2**	5.8±0.4	8.9±0.3	10.3±6.1	10.9±1.1	10.6±2.2	8.1±1.9	10.4±3.1
2s3z	Good	18.3	18.2±0.4	19.6±0.3	19.1±0.8	19.4±0.1	15.9±1.2	18.5±0.8	19.5±0.5	**20.0±1.3**	18.3±3.5
	Medium	12.6	14.3±0.7	17.2±0.8	14.3±2.0	17.4±0.3	15.6±0.3	18.1±0.9	17.6±0.6	18.7±1.8	**18.8±0.9**
	Poor	6.9	6.7±0.3	**12.1±0.4**	10.1±0.7	9.9±0.2	8.5±1.3	10.0±1.1	8.5±0.6	9.8±0.6	10.1±0.8
5m_vs_6m	Good	16.6	16.6±0.6	16.3±0.9	13.8±3.1	18.0±1.0	16.5±2.8	17.7±1.1	**18.6±3.5**	18.3±0.7	13.9±1.2
	Medium	12.6	14.2±0.5	17.2±0.4	16.8±3.1	17.5±0.4	15.2±2.6	16.2±0.9	15.6±1.3	**17.8±0.9**	15.2±2.8
	Poor	7.5	7.5±0.2	9.4±0.4	10.4±1.0	8.9±0.3	8.9±1.3	10.8±0.3	9.8±2.1	**12.5±1.2**	10.9±0.9
8m	Good	16.9	16.7±0.4	19.6±0.3	13.1±6.1	19.2±0.1	18.9±1.1	19.6±0.3	19.7±0.3	**20.0±1.2**	**20.0±0.5**
	Medium	10.1	10.7±0.5	18.6±0.5	16.3±3.1	18.0±0.5	16.8±1.6	18.6±0.8	**19.4±0.6**	17.5±0.8	17.4±0.9
	Poor	5.3	5.3±0.1	10.8±0.8	4.6±2.4	5.1±0.1	9.8±0.9	**12.0±1.2**	11.5±0.8	6.8±0.3	6.0±0.3

RQ2. How much reward gain is routed through CVA versus per-agent capacity?

RQ3. When does one-step inference suffice, and which design choice drives it?

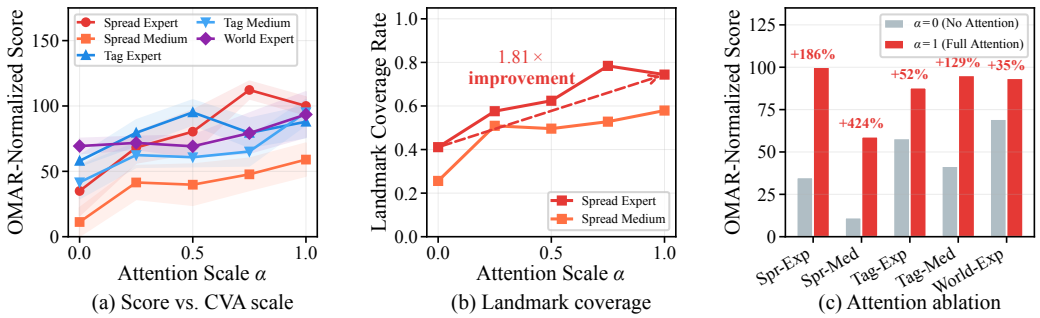
More detailed visualization experiments, ablation studies, and analyses are provided in Section A.

Benchmarks. We evaluate on two standard offline MARL suites: Multi-Agent Particle Environment (MPE), with Spread / Tag / World (Mordatch & Abbeel, 2018; Pan et al., 2022), and StarCraft Multi-Agent Challenge (SMAC), with 3m / 2s3z / 5m_vs_6m / 8m (Samvelyan et al., 2019; Formanek et al., 2023). We follow the dataset splits and evaluation protocol of MADiff (Zhu et al., 2024). Full environment details are in Section A.

Baselines. We compare against value-based and Gaussian-policy methods (BC, MA-ICQ, MA-CQL, MA-TD3+BC, OMAR) (Pomerleau, 1988; Yang et al., 2021; Kumar et al., 2020; Fujimoto & Gu, 2021; Pan et al., 2022), transformer policies (MADT) (Meng et al., 2023), diffusion models (MADiff, MADiTS, DoF) (Zhu et al., 2024; Yuan et al., 2025; Li et al., 2025), and flow methods (MAC-Flow) (Lee et al., 2026). MADiff-C denotes centralized execution. Its results are obtained by running the official MADiff implementation under our evaluation protocol. Results for the remaining baselines are taken from their respective papers.

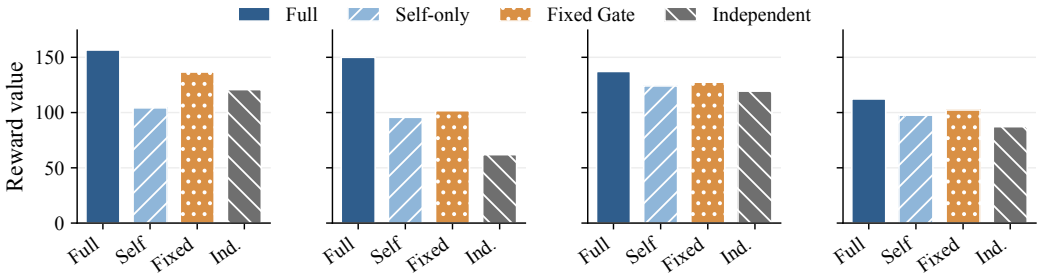
4.1 Setup

CoFlow variants and protocol. CoFlow and CoFlow-base share the same CVA architecture. CoFlow learns joint averaged velocities through finite-difference consistency training, whereas CoFlow-base uses ordinary conditional flow matching to learn instantaneous velocities. Both support one or multiple sampling steps. At one step, CoFlow-base directly applies a single Euler update across the full noise-to-data interval, without an averaged-velocity training objective. We evaluate both variants with the same execution suffixes: CoFlow-C and CoFlow-base-C use centralized execution, while



(a) Score vs. CVA scale (b) Landmark coverage (c) Attention ablation

Figure 3: Effect of CVA coupling on coordination quality in MPE. Panel **(a)** shows OMAR-normalized scores across CVA scales, panel **(b)** reports landmark coverage, and panel **(c)** summarizes the $\alpha = 0$ and $\alpha = 1$ endpoints from panel **(a)**.



(a) World Expert (b) World Medium (c) Tag Expert (d) Tag Medium

Figure 4: Effect of CVA design choices on MPE performance. Each panel compares full CVA, self-only attention, fixed gating, and independent-agent generation using the same OMAR-normalized score metric.

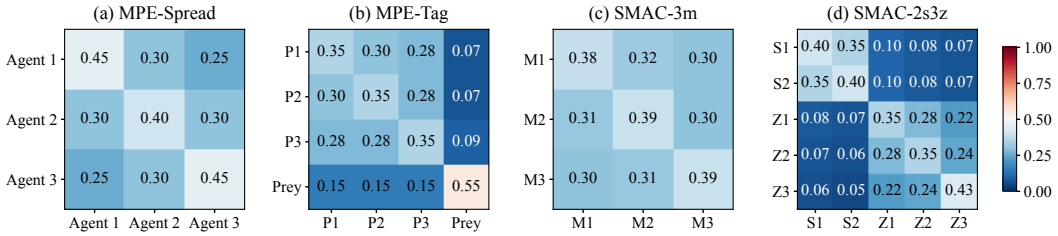
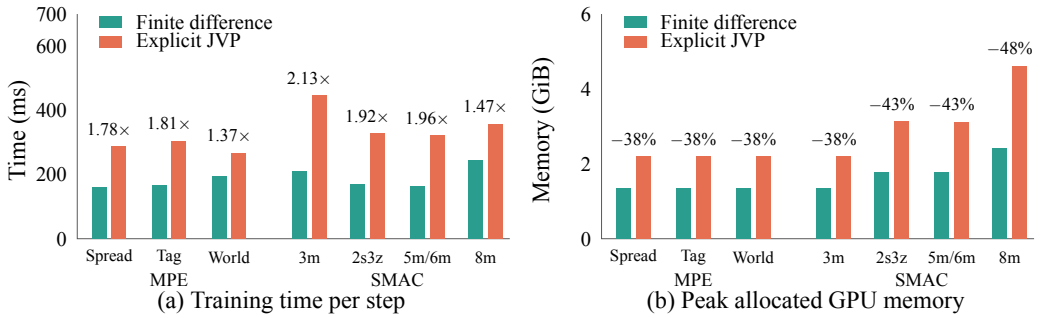


Figure 5: CVA attention in representative CoFlow-C runs under centralized execution: MPE-Spread, MPE-Tag, SMAC-3m, and SMAC-2s3z. Rows and columns index agents. The shared color scale indicates feature weights.

CoFlow-D and CoFlow-base-D use decentralized execution. MADiff-C scores and uncertainties come from our reproduction. Those of the remaining baselines follow the source papers. For CoFlow-C/D, Table 1 reports mean performance and standard deviation across three random seeds, using 1–3 sampling steps. Curves are shaded by ± 1 standard error. For MPE, we report OMAR-normalized score and apply the same transformation to the offline-data mean and CoFlow returns. The normalization anchors are given in Section A.

4.2 Performance (RQ1)

Table 1 reports the main comparison (best per row in **bold**, second best underlined). Compared with MADiff-C, CoFlow-C achieves an average relative performance improvement of 11.2%, with the two benchmark suites weighted equally. We first compare the mean scores within each suite and then average the two relative improvements. The aggregate gain summarizes performance across tasks and dataset qualities. Individual comparisons are reported in the table for both centralized and decentralized execution. We examine the contribution of cross-agent information sharing in §4.3.



(a) Training time per step (b) Peak allocated GPU memory

Figure 6: Per-update cost of finite difference and explicit JVP across all seven MPE/SMAC task shapes on an A100. We report mean CUDA-event time and peak allocated GPU memory (batch 32, horizon 24, 5 warm-up and 20 measured updates). Here 5m/6m denotes 5m_vs_6m.

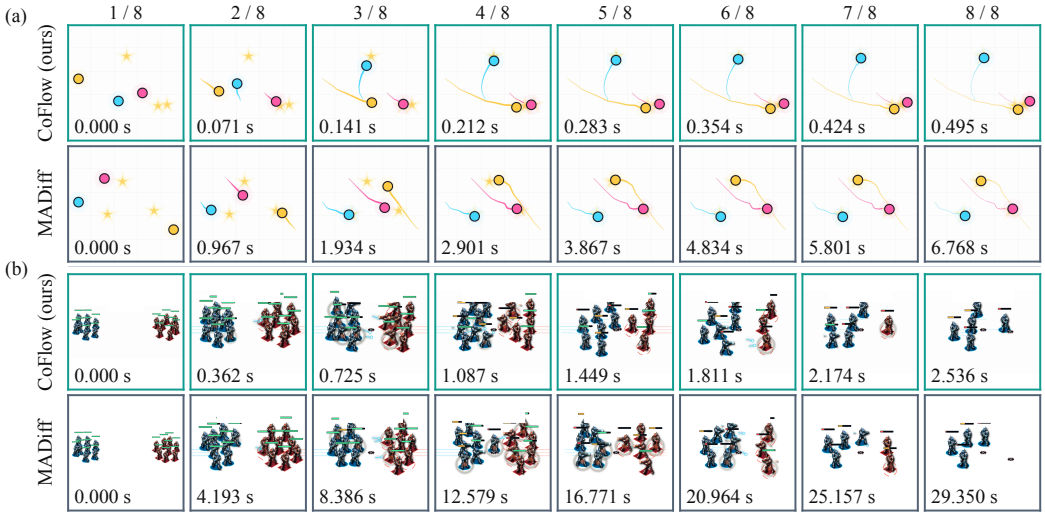


Figure 7: Policy rollouts with accumulated model sampling time: **(a)** MPE-Spread (landmark coverage) and **(b)** SMAC-8m (eight-versus-eight marine combat). Each row pair shows CoFlow and MADiff rollouts, respectively, with their corresponding model sampling times.

4.3 Coordination preservation (RQ2)

We evaluate CVA as the coordination mechanism by sweeping its contribution through a gating factor $\alpha \in [0, 1]$, with α=0 giving the attention-disabled setting and α=1 using the full module. Figure 3 reports three signals. Full attention (α=1) gives a higher OMAR-normalized score than the attention-disabled setting (α=0) on the five displayed MPE configurations, averaging a 165.1% relative score improvement and reaching 424% on Spread Medium (panels a and c). Landmark coverage, the fraction of landmarks covered by at least one agent at each timestep, increases from α=0 to α=1 on the two Spread configurations (panel b). The largest gains occur on Spread, where agents must cover different landmarks (panel c). At low α, agents cluster on the same landmark. The increase in coverage with α links teammate information sharing to more distributed landmark occupancy on these Spread configurations.

Figure 4 contrasts CVA design choices across four MPE configurations. Self-only restricts attention to each agent's own features. Fixed gating removes gate adaptation. Independent generation processes agents separately. Full CVA gives the highest reported World and Tag scores. The self-only and fixed-gating comparisons assess the effects of teammate information and learned gate scaling, respectively. All variants are compared under the same experimental configuration using OMAR-normalized scores.

The attention matrices in Figure 5 show task-dependent feature sharing with the same architecture and inputs without explicit role labels. MPE-Spread shows near-uniform mixing. MPE-Tag concentrates attention among predators and assigns weaker attention to the prey. SMAC-2s3z shows a block-

diagonal Stalker/Zealot grouping. Figure 10 reports learned gates and measured CVA residuals. Across the displayed task–layer pairs, the mean residual norm is 45.5%–67.3% of the input feature norm during one-step rollouts. This measures the total attention-branch correction, including self information.

4.4 Few-step sufficiency (RQ3)

The appendix heatmap in Figure 9 shows how performance varies with the sampling budget: CoFlow reaches near-peak scores with few updates on most splits.

One-step scores are close to the best scores measured over the tested sampling budgets on most configurations. On the training side, Figure 6 shows that the finite-difference implementation averages 1.78× faster training steps and 41.1% lower peak GPU memory than the tested explicit JVP implementation across the seven displayed task shapes. These measurements quantify the per-update savings from approximating the consistency correction of the joint model under the same training setup, using synthetic batches and weighting task shapes equally. On the inference side, Figure 7 compares accumulated model-only inference time on an NVIDIA A100. Each timestamp multiplies per-decision sampling latency by elapsed rollout steps, excluding simulator time. CoFlow uses one flow update per decision, and MADiff uses 15 DDIM denoising steps. The measured model sampling time is lower for CoFlow on the displayed rollouts. The complete timing table in Table 2 reports an average speedup of 12.93×.

Tasks shown in the keyframes. In Figure 7**(a)**, three agents (colored circles) must spread out to cover three landmarks (stars) while avoiding collisions. The trails show their movement toward different targets. In panel **(b)**, eight allied marines (blue) fight eight enemy marines (red). Each agent controls one allied unit, choosing movement and attack targets to defeat the opposing team. Health bars indicate remaining health. The eight snapshots in each row progress from left to right. These policy rollouts show the behavior of the corresponding methods and are annotated with accumulated model sampling times.

Additional appendix material. The appendix documents dataset return distributions in Figure 8, complete model-inference timing in Table 2, the full reward heatmap and training ablations in Figure 9 and Table 3, gate diagnostics in Figure 10, threshold-based few-step behavior in Figure 11, and rollout-level timing evidence in Figure 12.

5 Conclusion

We presented CoFlow, a distillation-free framework for coordinated few-step generation in offline multi-agent reinforcement learning. CoFlow learns a joint averaged-velocity field for few-step generation, uses CVA to model cross-agent dependencies, and employs finite-difference consistency training to reduce the training computation and memory costs of this joint model. On MPE and SMAC, CoFlow achieves an average relative score improvement of 11.2% over the reproduced centralized baseline, with the two suites weighted equally. The timing experiments measure a 12.93× model sampling speedup and lower per-update training computation and memory costs with finite differences.

Limitations. **(1)** For tasks requiring additional refinement, a budget-aware criterion could adapt the number of generation steps. **(2)** Our experiments cover teams of up to eight agents. Computational efficiency and coordination performance in larger teams remain to be evaluated. **(3)** Extending the framework to communication-, graph-, or value-decomposition-based coordination architectures remains future work. **(4)** The performance gap between centralized and decentralized execution varies across tasks. Future work could improve consistency among locally conditioned predictions.

AI use statement

We used AI tools only to polish the language of the manuscript. All technical content, experiments, analyses, claims, and final wording were reviewed and approved by the authors.

REFERENCES

Paul Barde, Jakob N. Foerster, Derek Nowrouzezahrai, and Amy Zhang. A model-based solution to the offline multi-agent reinforcement learning coordination problem. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 141–150, 2024. URL https://www.ifaamas.org/Proceedings/aamas2024/pdfs/p141.pdf.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. In *International Conference on Learning Representations (ICLR)*, 2024.

Claude Formanek, Asad Jeewa, Jonathan Shock, and Arnu Pretorius. Off-the-grid MARL: Datasets and baselines for offline multi-agent reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 2442–2444, 2023.

Claude Formanek, Callum R. Tilbury, Louise Beyers, Jonathan Shock, and Arnu Pretorius. Dispelling the mirage of progress in offline MARL through standardised baselines and evaluation. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations (ICLR)*, 2025.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 20132–20145, 2021.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J. Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30467–30476, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Geng_Improved_Mean_Flows_On_the_Challenges_of_Fastforward_Generative_Models_CVPR_2026_paper.html.

Yi Guo, Wei Wang, Zhihang Yuan, Rong Cao, Kuan Chen, Zhengyang Chen, Yuanyuan Huo, Yang Zhang, Yuping Wang, Shouda Liu, and Yuxuan Wang. SplitMeanFlow: Interval splitting consistency in few-step generative modeling. arXiv preprint arXiv:2507.16884, 2025.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-learning as an actor-critic method with diffusion policies. arXiv preprint arXiv:2304.10573, 2023.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6840–6851, 2020.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, pp. 9902–9915, 2022.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1179–1191, 2020.

Dongsu Lee, Daehee Lee, and Amy Zhang. Multi-agent coordination via flow matching. In *International Conference on Learning Representations (ICLR)*, 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 (ICLR)*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

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

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22825–22855, 2023. URL https://proceedings.mlr.press/v202/lu23d.html.

Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? In *International Conference on Learning Representations (ICLR)*, 2025. Spotlight.

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 (ICLR)*, 2026.

Linghui Meng, Muning Wen, Chenyang Le, Xiyun Li, Dengpeng Xing, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, Yaodong Yang, and Bo Xu. Offline pre-trained multi-agent decision transformer. *Machine Intelligence Research*, 20:233–248, 2023.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

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 (ICML)*, pp. 17221–17237, 2022.

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-learning. In *International Conference on Machine Learning (ICML)*, pp. 48104–48127, 2025.

Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1988.

Dan Qiao, Wenhao Li, Shanchao Yang, Hongyuan Zha, and Baoxiang Wang. OMSD: Offline multi-agent reinforcement learning via sequential score decomposition. In *International Conference on Machine Learning (ICML)*, 2026.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations (ICLR)*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 2186–2188, 2019.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, pp. 32211–32252, 2023.

Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023b.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 10299–10312, 2021.

Lei Yuan, Yuqi Bian, Lihe Li, Ziqian Zhang, Cong Guan, and Yang Yu. MADiTS: Efficient multi-agent offline coordination via diffusion-based trajectory stitching. In *International Conference on Learning Representations (ICLR)*, 2025.

Wenhao Zhan, Scott Fujimoto, Zheqing Zhu, Jason D. Lee, Daniel R. Jiang, and Yonathan Efroni. Exploiting structure in offline multi-agent RL: The benefits of low interaction rank. In *International Conference on Learning Representations (ICLR)*, 2025.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025a.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning flow matching policy with online reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4177–4206, 2024.

Guowei Zou, Weibing Li, Hejun Wu, Yukun Qian, Yuhang Wang, and Haitao Wang. D^2PPO: Diffusion policy policy optimization with dispersive loss. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 18891–18899, 2026a.

Guowei Zou, Haitao Wang, Hejun Wu, Yukun Qian, Yuhang Wang, and Weibing Li. One step is enough: Dispersive MeanFlow policy optimization. arXiv preprint arXiv:2601.20701, 2026b.

Appendix

APPENDIX CONTENTS

Section	Title
Appendix A	Experimental Details and Results
A.1	Experimental Setup
A.2	Benchmarks and Offline Datasets
A.3	Full Performance Summary
A.4	Ablations
A.5	Additional Coordination Analysis
Appendix B	Method Details and Theoretical Foundations
B.1	Preliminaries
B.2	Notation
B.3	Implementation Notes
B.4	Training and Inference Algorithms
B.5	Taylor Control of the Finite-Difference Surrogate
B.6	Joint Velocity Decomposition
B.7	One-Step Wasserstein Error Bound
B.8	Scaling with the Number of Agents

A EXPERIMENTAL DETAILS AND RESULTS

This section reports the experimental protocol, benchmark datasets, full results, ablations, few-step analyses, and coordination diagnostics.

A.1 EXPERIMENTAL SETUP

Network architecture. CoFlow uses a temporal U-Net with encoder-decoder skip connections, 1D convolutional blocks with kernel size 5, group normalization with 8 groups, Mish activations, channel multipliers [1, 4, 8], and base dimension 128. The multi-scale skip structure provides natural insertion points for gated CVA while keeping few-step trajectory generation memory-efficient.

Coordinated Velocity Attention (CVA). We apply 4-head CVA at the bottleneck and before each decoder skip fusion. Given skip features $h_l^i \in \mathbb{R}^{T_l \times F_l}$ for agent i at resolution l, CVA attends across the agent dimension and adds its output back through the zero-initialized residual gate in Equation (3). The decoder then concatenates the enhanced skip with its current state.

Time conditioning and action decoding. The averaged velocity model conditions on the flow endpoint time t and reference time r, equivalently on t and $t - r$, using sinusoidal encodings injected by feature-wise linear modulation (FiLM). For action extraction, a separate 3-layer inverse-dynamics MLP with hidden dimension 256 maps consecutive observation pairs (o_q, o_{q+1}) to a_q and stays outside the velocity U-Net.

A.2 BENCHMARKS AND OFFLINE DATASETS

Training cost benchmark. For Figure 6, both implementations use the same shared CVA network, synthetic Gaussian batches, AdamW with learning rate 2×10^{-4}, seed 100, guidance weight 1.2, and finite-difference step 10^{-3}. Each timed update includes the velocity loss, backward pass, and optimizer step. Data loading, inverse dynamics, and environment simulation are excluded. Time is averaged over 20 updates after 5 warm-up updates, and memory is the maximum allocated GPU memory over those updates. We average per-task JVP/FD time ratios and percentage memory reductions. The finite-difference implementation follows the velocity-model update in Algorithm 1, using the sample-conditional tangent $\mathbf{v}_c = \mathbf{z}_1 - \boldsymbol{\tau}_0$ and the two-time-conditioned joint field $\mathbf{u}_\theta(\mathbf{z}_t, r, t)$. The JVP implementation uses the same training setup and replaces the finite-difference correction in

Equation (6) with the explicit directional derivative in Equation (19). Both corrections are detached during optimization. Measurements were collected on a shared A100 server.

Environment protocol. We follow the released offline datasets, reward definitions, and benchmark configurations used by prior offline MARL baselines. MPE and SMAC use the original tasks with matched baseline results.

MPE score normalization. Following MADiff and other MPE baselines, we report OMAR-normalized score $100\,(R - R_{\text{random}})/(R_{\text{expert}} - R_{\text{random}})$, where R is the evaluated return and $R_{\text{random}}, R_{\text{expert}}$ are the fixed random-policy and expert-return anchors. We apply the same transformation to offline-data means and CoFlow returns.

Datasets. We evaluate on MPE and SMAC. MPE includes Spread, Tag, and World with expert, medium-replay, medium, and random datasets. SMAC covers symmetric 3m and 8m battles, heterogeneous 2s3z, and asymmetric 5m_vs_6m micromanagement tasks.

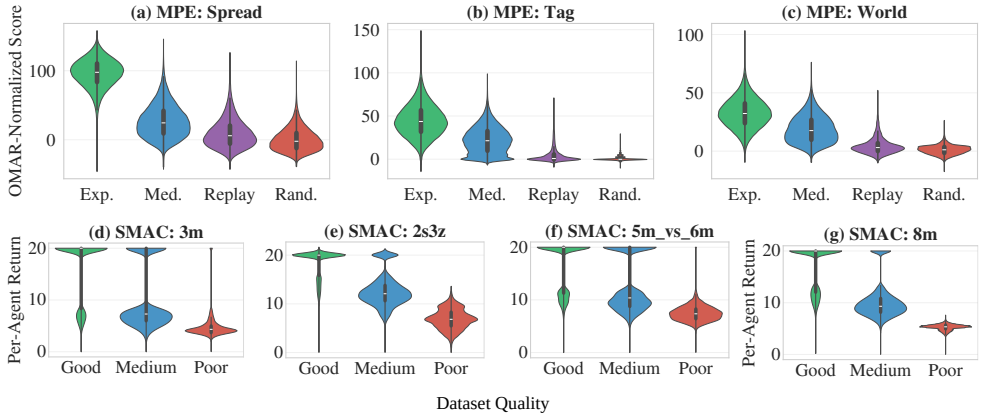


Figure 8: Offline dataset distributions. MPE values are OMAR-normalized scores computed from per-agent returns. SMAC values are per-agent returns.

Dataset return distributions. Figure 8 shows violin plots of the offline dataset returns. In each subplot, the horizontal axis indexes dataset quality and the vertical axis gives the corresponding return scale. MPE uses OMAR-normalized score, while SMAC uses per-agent returns. The violin width indicates the density of trajectories at each return level. The plots verify that the benchmark covers both high-return expert data and lower-return medium or random data, so the main comparison tests CoFlow across substantially different offline-data regimes.

Table 2: Relationship between sampling method and model-inference cost across benchmark task and dataset-quality pairs, measured on an NVIDIA A100. CoFlow uses one flow step and MADiff uses 15 DDIM denoising steps. CoF and MAD denote CoFlow and MADiff.

MPE								SMAC							
Task	Quality	Steps	Latency (ms)		Rollout (s)		Speedup	Task	Quality	Steps	Latency (ms)		Rollout (s)		Speedup
			CoF	MAD	CoF	MAD	MAD/CoF				CoF	MAD	CoF	MAD	MAD/CoF
Spread	Expert	25	19.8	270.7	0.495	6.768	13.68×	3m	Poor	60	20.2	275.1	1.215	16.509	13.59×
	Med.-Rep.	25	19.2	273.8	0.481	6.844	14.23×		Medium	58	26.5	275.1	1.540	15.958	10.37×
	Medium	25	20.2	264.2	0.506	6.606	13.06×		Good	35	19.7	277.9	0.689	9.726	14.11×
	Random	25	20.3	264.8	0.508	6.619	13.03×	2s3z	Poor	120	21.2	281.6	2.546	33.795	13.27×
Tag	Expert	25	19.4	262.9	0.486	6.573	13.53×		Medium	98	19.9	273.3	1.953	26.784	13.71×
	Med.-Rep.	25	19.6	259.1	0.491	6.478	13.19×		Good	116	21.1	270.1	2.449	31.328	12.79×
	Medium	25	19.4	262.3	0.485	6.558	13.51×	5m vs. 6m	Poor	70	22.7	284.7	1.588	19.926	12.55×
	Random	25	19.0	262.6	0.476	6.564	13.80×		Medium	39	24.5	282.0	0.956	10.999	11.50×
World	Expert	25	19.0	262.0	0.475	6.549	13.79×		Good	42	26.8	270.8	1.124	11.374	10.12×
	Med.-Rep.	25	19.2	260.8	0.481	6.519	13.56×	8m	Poor	120	24.5	301.4	2.939	36.170	12.31×
	Medium	25	18.9	262.0	0.472	6.550	13.88×		Medium	116	29.7	289.0	3.449	33.521	9.72×
	Random	25	19.7	274.8	0.492	6.869	13.95×		Good	114	22.2	290.6	2.536	33.128	13.07×

Note. MADiff timing uses randomly initialized same-shape networks because timing depends mainly on the sampling graph and tensor shapes. These measurements quantify model sampling cost.

Complete 24-task model-inference timing. Table 2 reports the corresponding model-only timing comparison for all 24 benchmark task and dataset-quality pairs used in Table 1. Rows enumerate suite, task, and dataset quality. Columns report rollout length, per-call latency, full-rollout model time, and MADiff/CoFlow speedup. The table uses the same full-rollout convention as Figure 12: per-decision model sampling latency multiplied by the actual rollout length. CoFlow uses one-step sampling, while MADiff uses 15-step DDIM sampling.

A.3 Full Performance Summary

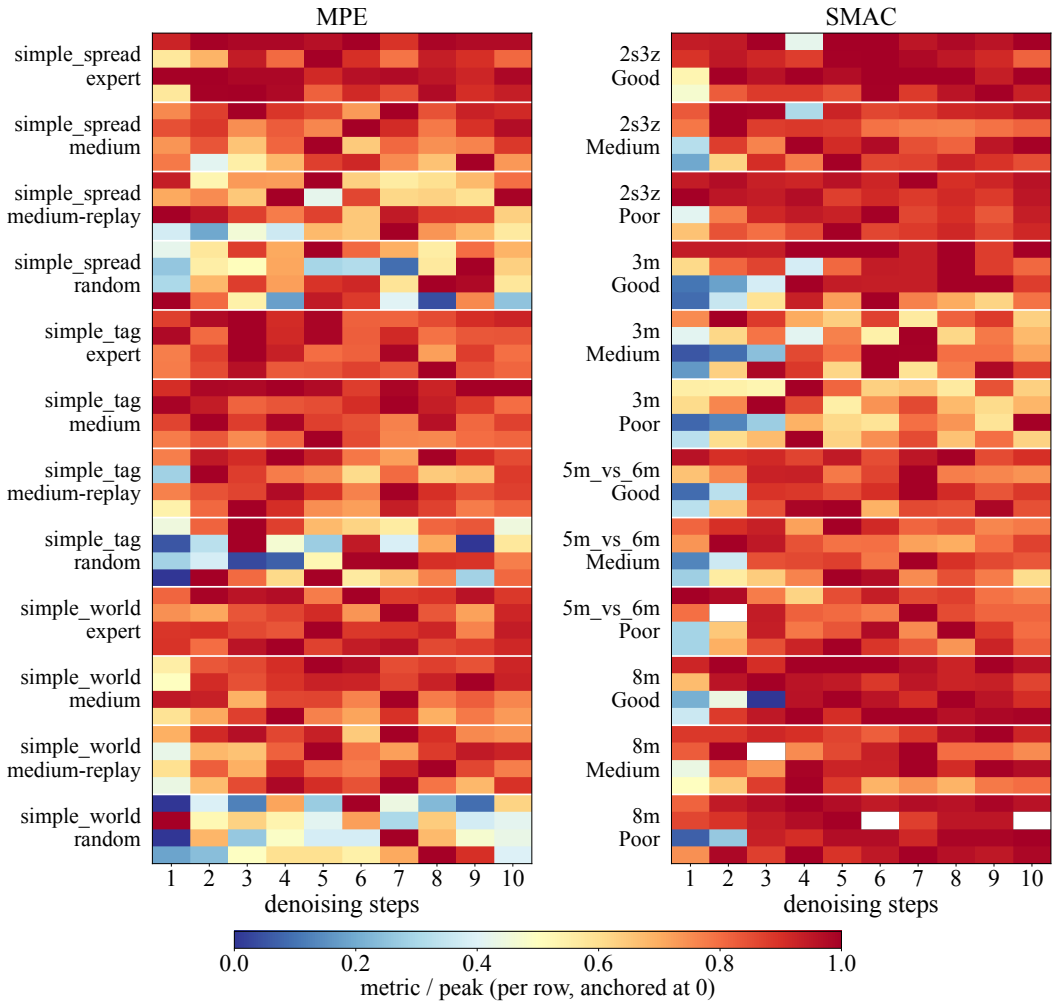


Figure 9: Environment-wise views of the denoising-step heatmap. Within each block for a task and dataset quality, rows follow CoFlow-C, CoFlow-D, CoFlow-base-C, and CoFlow-base-D.

Environment-wise summary of all 96 configurations. Figure 9 compresses every performance value from the main experiments into two environment-wise heatmaps. The horizontal axis is the denoising-step budget, and each block for a task and dataset quality stacks CoFlow-C, CoFlow-D, CoFlow-base-C, and CoFlow-base-D. Cell color encodes performance as a fraction of the row's peak, so red indicates near-best performance for that configuration. CoFlow rows are usually near their row peak from step one, while CoFlow-base rows more often improve with additional denoising. This pattern shows that CoFlow achieves strong performance with few generation steps, with additional refinement benefiting some low-coverage Poor splits and asymmetric SMAC cases.

A.4 Ablations

Effect of guidance weight. The classifier-free guidance weight ω controls the strength of return conditioning during inference. Table 3 presents the effect of varying ω on MPE-Spread with expert data. Setting $\omega = 1.0$ corresponds to standard conditional generation, while $\omega = 1.2$ gives the best performance and raises the OMAR-normalized score from 97.2 to 114.6 (18%). Larger values of 1.5 or 2.0 lower the score. Based on this MPE-Spread Expert sweep, we use $\omega = 1.2$ as the default across all experiments.

Table 3: Effect of guidance weight on MPE-Spread Expert.

Guidance Weight ω	OMAR-Normalized Score
1.0	97.2 ± 4.1
1.2	$\mathbf{114.6 \pm 0.5}$
1.5	111.3 ± 2.3
2.0	104.8 ± 3.3

The best result is in **bold**. Results are reported as mean $\pm$ standard deviation across three random seeds.

A.5 Additional Coordination Analysis

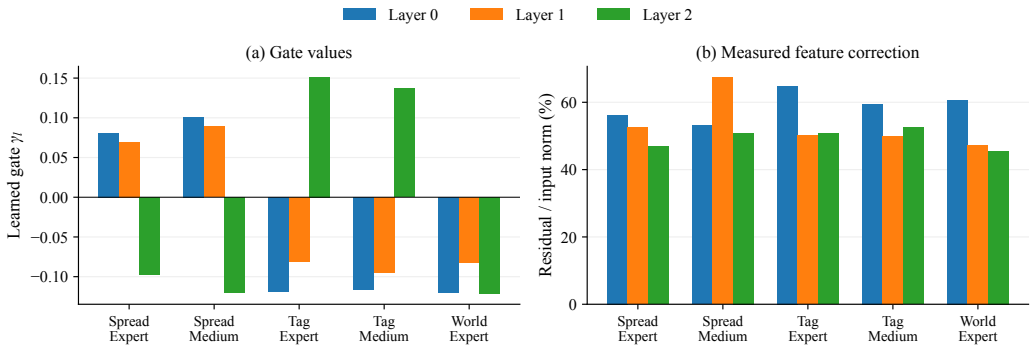


Figure 10: CVA gates and measured feature corrections in five MPE configurations. (a) Learned gates. (b) Mean residual-to-input norm ratio over 10 one-step rollout episodes. Layers 0–2 are the bottleneck and two decoder attention modules.

Measured CVA corrections. We load the five EMA checkpoints used for the gate analysis and record $\|\gamma_l \mathrm{CVA}_l(h_l)\|_2 / \|h_l\|_2$ during centralized one-step rollouts. Each norm spans the channels and trajectory positions of one agent. We average across agents, active environment steps, and both guidance queries within each episode, then across 10 episodes with evaluation seed 100. The checkpoints share training seed 100. Layers 0–2 are executed during sampling. The unused fourth gate is omitted. The ratio measures the total attention residual, including self information. Performance and coverage ablations assess its effect on coordination.

Steps to reach the offline dataset mean. Figure 11 provides an alternative threshold-based view of few-step behavior. Each panel fixes a dataset-mean percentage threshold η. The horizontal axis is the first denoising step at which a configuration reaches that threshold, and the vertical axis counts how many configurations reach it at that step. Bars are grouped by the four CoFlow variants. The concentration of mass at the first few steps shows that most configurations reach performance at or above the offline dataset mean quickly, while the small right tails identify the settings where additional denoising remains useful.

Tasks and visual symbols. In Figure 12**(a)**, three predators (triangles) cooperate to catch a faster prey (cyan circle) while navigating around two fixed obstacles (gray disks). The team is rewarded for contacting the prey, so the converging trails illustrate pursuit. Panel **(b)** uses the same three-predator, one-prey objective in World, with one obstacle, two food locations (blue diamonds), and two forest regions (green disks). The prey can earn additional reward by reaching food, while the predators coordinate their pursuit. In panel **(c)**, five allied marines (blue) face six enemy marines (red). Agents must coordinate movement and target selection to defeat a numerically larger team. Health bars and unit disappearances indicate damage and eliminations.

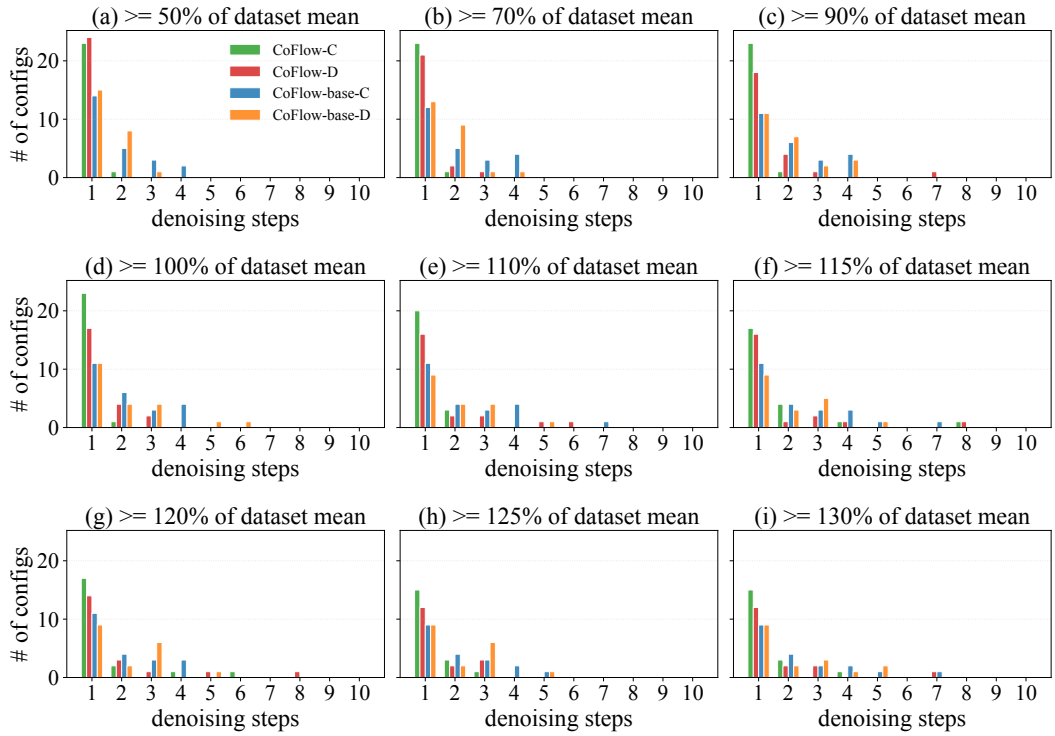


Figure 11: Grouped view of denoising steps needed to reach target performance thresholds. Each subplot compares the four CoFlow variants at every denoising-step budget.



Figure 12: Additional policy rollouts with accumulated model sampling time: **(a)** MPE-Tag, **(b)** MPE-World, and **(c)** SMAC-5m_vs_6m. Each adjacent row pair shows CoFlow and MADiff rollouts, respectively, with their corresponding model sampling times.

Rollout frames with model sampling times. Each task in Figure 12 appears as a pair of adjacent rows for CoFlow and MADiff. To keep the MPE shapes and SMAC unit sprites legible at paper scale, each row shows eight uniformly spaced frames from a rollout of the corresponding policy. The visual sequences show policy behavior in the environment. The overlaid timestamps report the corresponding method's accumulated measured model-inference time. These totals multiply the measured per-decision sampling latency by the actual rollout length, separating model speed from simulator and rendering overhead.

B Method Details and Theoretical Foundations

This appendix gives background and notation, training and inference algorithms, and proofs for the finite-difference approximation, joint velocity decomposition, and one-step error bounds.

Part I: Background and Notation

B.1 Preliminaries

We distinguish the conditional paths used to construct training pairs from the marginal flow whose interval displacement defines the averaged velocity. Ordinary flow-matching regression is background and defines the CoFlow-base ablation. The full CoFlow objective is Equation (8).

Ordinary conditional flow matching. Sample $\boldsymbol{\tau}_0$ from the offline trajectory law p_τ and independent noise $\mathbf{z}_1 \sim \mathcal{N}(0, \mathbf{I})$, fixing the observation/return context. The linear interpolation and its sample-conditional velocity are given in Equation (1). Ordinary flow matching trains an instantaneous-velocity predictor by

$$\mathcal{L}_{\text{flow}}(\theta) = \frac{1}{Nd}\mathbb{E}_{t,\boldsymbol{\tau}_0,\mathbf{z}_1} \left\| \mathbf{v}_\theta(\mathbf{z}_t, t) - \mathbf{v}_c \right\|_2^2, \tag{12}$$

where $\mathbf{v}_\theta$ predicts instantaneous velocity.

The population regression target is

$$\mathbf{v}(\mathbf{z}, t) = \mathbb{E}[\mathbf{v}_c \mid \mathbf{z}_t = \mathbf{z}]. \tag{13}$$

These conditional and marginal regression objectives have the same parameter-dependent part (Lipman et al., 2023; Liu et al., 2023). At the pure-noise endpoint, independence gives $\mathbf{v}(\mathbf{z}, 1) = \mathbf{z} - \mathbb{E}[\boldsymbol{\tau}_0]$. Consequently, an exact single Euler update using this instantaneous endpoint velocity collapses generation to the conditional data mean.

Marginal transport and averaged velocity. Let $\Phi_{s\leftarrow t}(\mathbf{z})$ solve the marginal ODE

$$\frac{d}{ds}\Phi_{s\leftarrow t}(\mathbf{z}) = \mathbf{v}(\Phi_{s\leftarrow t}(\mathbf{z}), s), \qquad \Phi_{t\leftarrow t}(\mathbf{z}) = \mathbf{z}. \tag{14}$$

Under the usual well-posedness and transport assumptions, $\Phi_{0\leftarrow 1}$ maps the noise law to p_τ. Individual conditional interpolation paths are straight, but the trajectories of this marginal ODE need not be straight. The ideal averaged velocity is therefore (Geng et al., 2025)

$$\mathbf{u}(\mathbf{z}, r, t) = \frac{1}{t-r}\int_r^t \mathbf{v}(\Phi_{s\leftarrow t}(\mathbf{z}), s)\, ds = \frac{\mathbf{z} - \Phi_{r\leftarrow t}(\mathbf{z})}{t-r}. \tag{15}$$

This is the integral form of Equation (2). The integral is evaluated along the marginal ODE trajectory. At $r = t$, $\mathbf{u}$ is defined by its continuous extension $\mathbf{v}(\mathbf{z}, t)$. In particular, $\mathbf{z}_1 - \mathbf{u}(\mathbf{z}_1, 0, 1) = \Phi_{0\leftarrow 1}(\mathbf{z}_1)$. The consistency identity below enables learning this interval displacement using conditional-velocity supervision, without computing the transport integral for each training example.

Wasserstein-2 distance. For two probability measures μ, ν on $\mathbb{R}^{Nd}$ with finite second moments,

$$W_2(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu,\nu)} \int \|x - y\|^2 \, d\pi(x, y) \right)^{1/2}, \tag{16}$$

where $\Pi(\mu, \nu)$ contains the couplings of μ and ν.

A shared-randomness coupling gives

$$W_2(\mu,\nu) \le \left(\mathbb{E}_\xi \|F(\xi) - G(\xi)\|^2\right)^{1/2}, \tag{17}$$

where $F(\xi) \sim \mu$ and $G(\xi) \sim \nu$ share the same random input ξ.

We use this shared-noise coupling in Section B.7 to relate averaged-velocity approximation to one-step sample error.

B.2 Notation

Table 4 collects the method and analysis notation. Vector and feature norms are Euclidean/Frobenius. Linear-map norms are spectral. Training losses use coordinate means, while transport bounds use joint norms.

Table 4: Unified notation for CoFlow.

Symbol	Meaning
Background and multi-agent state	
$N; i, j; d$	Number of agents; agent indices; flattened trajectory dimension per agent.
$q; r, t, s; h = t - r$	Environment decision index; flow times with $r \le t$; interval length.
$\mathcal{D}; p_\tau; p_X$	Offline dataset; clean observation-trajectory law; law of random variable X.
$\boldsymbol{\tau}_0 = (\tau_0^i)_{i=1}^N$	Clean joint observation trajectory, with $\tau_0^i \in \mathbb{R}^d$.
$\mathbf{o}_q; \mathbf{a}_q$	Joint observations and actions; o_q^i, a_q^i are agent blocks.
$\mathbf{z}_1; \mathbf{z}_t$	Joint Gaussian noise in $\mathbb{R}^{Nd}$; sampled linear interpolant.
$\mathbf{x}_t; \Phi_{s\leftarrow t}$	Marginal ODE trajectory; transport from flow time t to s.
$\mathbf{v}_c; \mathbf{v}$	Sample-conditional velocity $\mathbf{z}_1 - \boldsymbol{\tau}_0$; theoretical marginal instantaneous field.
$\mathbf{u}; \mathbf{u}_\theta; \mathbf{v}_\theta$	Theoretical averaged field; learned averaged field; background/base instantaneous predictor.
$u_\theta^i = [\mathbf{u}_\theta]^i$	Agent-i output block of the joint field, depending on the full team input; the same block convention applies to other joint vectors.
Architecture and conditioning	
$\theta; \phi$	All velocity-network parameters, including gates; inverse-dynamics parameters.
$L, l; T_l, F_l$	Number and index of CVA layers; temporal length and feature width at layer l.
$\boldsymbol{h}_l; c_l^i; \tilde{h}_l^i$	Joint skip features; CVA message; enhanced skip. Subscript l identifies the layer, and the tilde marks gated enhancement.
$\alpha \in [0, 1]$	CVA intervention multiplier in the experiments; distinct from attention weights α_{lm}^{ij}.
$\gamma_l; \boldsymbol{\gamma}$	Learnable scalar gate; collection of gates included in θ.
$\mathsf{c}, \mathsf{c}_q; \mathsf{c}_q^{(i)}; m_i$	Observation context; acting-agent-i local context; teammate-observation masking operator.
$\mathcal{H}_q; C$	Observation window $[\mathbf{o}_{q-C}, \ldots, \mathbf{o}_q]$; number of past decisions, giving $C + 1$ observations.
$R_{\text{random}}, R_{\text{expert}}$	Random-policy and expert-return anchors used for the MPE score normalization.
$R; \widetilde{R}; \varnothing$	Target return; sampled training return condition; dropped return only (observation context retained).
$p_{\text{drop}}; \omega; \hat{\mathbf{u}}_\theta$	Probability of dropping the return condition; guidance weight; guided inference field.
Training and sampling	
$\rho; \mu_{\text{time}}, \sigma_{\text{time}}$	Fraction of time pairs with $r = t$; Gaussian parameters before logistic time transformation.
$\varepsilon_0; \varepsilon$	Positive nominal finite-difference magnitude; valid nonzero signed shift when $r < t$.

Continued on next page

Table 4 (continued)

Symbol	Meaning
stopgrad	Detaches the finite-difference or JVP correction; the data-derived tangent $\mathbf{v}_c$ is fixed.
$\mathcal{D}_{\mathbf{v}_c}$; D_ε	Directional derivative along $(\mathbf{v}_c, 1)$ at fixed r; finite-difference quotient.
$\boldsymbol{\delta}_{\mathrm{FD}}$; $\mathbf{V}_\theta$	Detached joint derivative correction; corrected training prediction $\mathbf{u}_\theta + (t-r)\boldsymbol{\delta}_{\mathrm{FD}}$.
$\mathbf{V}_\theta^{\mathrm{JVP}}$	Training prediction corrected by the stopped exact directional derivative.
$\mathcal{L}_{\mathrm{vel}}, \mathcal{L}_{\mathrm{flow}}, \mathcal{L}_{\mathrm{JVP}}$	Velocity losses normalized by Nd; ordinary flow loss is the background/base objective.
I_ϕ^i; ℓ_{act}; $\mathcal{L}_{\mathrm{act}}$	Agent-i inverse-dynamics predictor; per-example action loss; agent-averaged action objective.
$\mathcal{U}_q^i$; d_a^i; y	Legal discrete actions; continuous-action dimension; decoder output vector/logits.
Dec_i; λ; $\mathcal{L}$	Action decoding operator; action-loss weight; combined training objective.
$\mathbf{b}, \mathbf{b}_+$; $\mathbf{w}_k$	Base and shifted training evaluations; guided field evaluated before a simultaneous sampling update.
η_{lr}; K, k, t_k	Learning rate; sampling-step count, index, and time schedule.
$\hat{\boldsymbol{\tau}}_0$; $\hat{o}_{q+1}^i$	Generated trajectory and predicted next observation (hat); q indexes environment decisions and 0 the data endpoint.
$\hat{\boldsymbol{\tau}}_0^{(i)}$; $\mathbf{z}_1^{(i)}$	Joint prediction and noise in the local planning run of agent i; parentheses index the local planning run.
Bounds and mechanism analysis	
$u_{\theta,0}^i$; $u_{\mathrm{ind}}^i, u_{\mathrm{coord}}^i$	Gate-off intervention with other weights fixed; independent term and full gate-off difference.
$\bar{\mathbf{u}}_\theta$; $\bar{u}_\theta^i$; $\boldsymbol{\Delta}_{\mathrm{attn}}$	Stacked self-only attention reference; its agent block; cross-agent residual.
M, m; α_{lm}^{ij}; B_{lm}	Attention-head count/index; row-normalized attention weights; effective value/output map in the idealized bound.
g_l; $f_l^i, \bar{f}_l^i$; b_l^i	Shared non-expansive map in the bound; full/self-only features; injected cross-agent message.
D_l; δ_l^i; E_l	Feature diversity; feature difference between full and self-only attention; joint perturbation norm.
$\bar{\sigma}_l$; $\bar{\sigma}$	Gate magnitude times the sum of attention-head projection norms; maximum over layers.
$\mathcal{I}_\varepsilon$; M_2^i, M_2	Signed query segment; per-agent and joint second-directional-derivative bounds.
$\mathbf{R}_\varepsilon$; $\mathbf{g}(a)$	Joint Taylor remainder (distinct from return R); shifted joint query path.
W_2; μ, ν; $\Pi(\mu, \nu)$	Wasserstein-2 distance; two laws; admissible couplings π of those laws.
$\mathcal{T} = \Phi_{0\leftarrow 1}$; $\boldsymbol{\tau}_{\mathrm{ref}}$	Endpoint transport map; ideal sample coupled through the same noise.
$\bar{\mathbf{u}} = (\bar{u}^i)_{i=1}^N$; $\boldsymbol{\Delta}$	Chosen oracle self-only terms and remaining joint residual.
$\epsilon_{\mathrm{vel}}, \epsilon_i, \epsilon_{\mathrm{coord}}$	Unnormalized endpoint root mean square (RMS) errors for the joint field, self-only blocks, and coordination residual.
$\bar{\epsilon}$; D_N; ϵ_{attn}	Uniform self-only error bound; initial diversity for an N-agent team; assumed residual-error factor.
η	Dataset-mean percentage threshold in Figure 11; distinct from learning rate.

PART II: TRAINING AND INFERENCE PROCEDURE

B.3 IMPLEMENTATION NOTES

MeanFlow identity and joint dependence. Writing out the arguments in Equation (5) gives

$$\mathbf{u}(\mathbf{x}_t, r, t) + (t-r)\left[\nabla_{\mathbf{x}}\mathbf{u}(\mathbf{x}_t, r, t)\mathbf{v}(\mathbf{x}_t, t) + \partial_t \mathbf{u}(\mathbf{x}_t, r, t)\right] = \mathbf{v}(\mathbf{x}_t, t). \tag{18}$$

For the fixed training tangent $\mathbf{v}_c$, the joint derivative and its agent-i block are

$$\begin{aligned} \mathcal{D}_{\mathbf{v}_c}\mathbf{u}_\theta &= (\nabla_{\mathbf{z}}\mathbf{u}_\theta)\mathbf{v}_c + \partial_t\mathbf{u}_\theta, \\ [\mathcal{D}_{\mathbf{v}_c}\mathbf{u}_\theta]^i &= \sum_{j=1}^{N} \frac{\partial u_\theta^i}{\partial z^j}\, v_c^j + \partial_t u_\theta^i, \end{aligned} \tag{19}$$

where $\mathcal{D}_{\mathbf{v}_c}$ differentiates along $(\mathbf{v}_c, 1)$ at fixed r and context. All network terms use $(\mathbf{z}_t, r, t)$. The sum exposes dependence on teammate blocks.

The derivative-based reference predictor and its loss are

$$\begin{aligned} \mathbf{V}_\theta^{\mathrm{JVP}} &= \mathbf{u}_\theta + (t-r)\mathrm{stopgrad}[\mathcal{D}_{\mathbf{v}_c}\mathbf{u}_\theta], \\ \mathcal{L}_{\mathrm{JVP}} &= \frac{1}{Nd}\mathbb{E}\|\mathbf{V}_\theta^{\mathrm{JVP}} - \mathbf{v}_c\|_2^2. \end{aligned} \tag{20}$$

This fits $\mathbf{u}_\theta$ to the stopped target $\mathbf{v}_c - (t-r)\mathcal{D}_{\mathbf{v}_c}\mathbf{u}_\theta$. The sampled tangent yields a stochastic training target whose connection to the theoretical field $\mathbf{u}$ holds in conditional expectation. For a fixed network, the conditional expectation of the un-stopped JVP residual recovers the marginal consistency residual because the derivative is linear in $\mathbf{v}_c$. Gradient stopping avoids differentiating the JVP again. Its construction still has implementation-dependent compute and memory cost. Equation (6) approximates this joint derivative, with error bounded in Theorem B.1.

Finite-difference boundary and gradient conventions. Use the same observation context, return condition, and stochastic masks for both queries. For a nonzero time interval, one valid rule for choosing the sign of the finite-difference step is

$$\varepsilon = \begin{cases} \varepsilon_0, & t + \varepsilon_0 \le 1, \\ -\min(\varepsilon_0, h/2), & t + \varepsilon_0 > 1, \end{cases} \tag{21}$$

where $h = t - r > 0$ and $\varepsilon_0 > 0$ is the nominal step magnitude.

This keeps the full segment inside $r \le t + \varepsilon \le 1$ and avoids a zero denominator. At $h = 0$, set the correction to zero without evaluating a quotient. The base query retains parameter gradients. Its detached value is reused in the quotient, while the shifted query is evaluated without a gradient graph. The Taylor bound applies to any nonzero signed step satisfying this domain constraint.

CVA and parameter convention. The N feature streams share the temporal U-Net. Every CVA skip and the bottleneck use the gated operation in Equation (3). The decoder concatenates the enhanced skip with its current state before its residual blocks. All gates belong to θ and start at zero. The learned combined scale of the gates and projection matrices is defined in Theorem B.2.

Action loss and decoding. The per-example loss and decoder are

$$\begin{aligned} \ell_{\mathrm{act}}(a, y; \mathcal{U}) &= \begin{cases} \|a - y\|_2^2 / d_a^i, & \text{continuous}, \\ -y_a + \log \sum_{b \in \mathcal{U}} e^{y_b}, & \text{discrete}, \end{cases} \\ \mathrm{Dec}_i(y; \mathcal{U}) &= \begin{cases} y, & \text{continuous}, \\ \arg\max_{b \in \mathcal{U}} y_b, & \text{discrete}, \end{cases} \end{aligned} \tag{22}$$

where $y = I_\phi^i(o_q^i, o_{q+1}^i)$, d_a^i is the continuous-action dimension, and $\mathcal{U} = \mathcal{U}_q^i$.

The legal-action set is ignored in the continuous case. Discrete training actions belong to the nonempty legal set, and ties at decoding use a fixed rule. The expectation in Equation (9) averages valid offline transitions, excluding padding or invalid transitions.

Observation context and execution modes. We use the joint history window

$$\mathcal{H}_q = [\mathbf{o}_{q-C}, \ldots, \mathbf{o}_q], \tag{23}$$

where C counts past decisions, giving $C + 1$ observations. The joint context is $\mathsf{c}_q = \mathcal{H}_q$.

The window reduces to the current observation when $C = 0$. For CoFlow-C, one joint generation uses c_q. CoFlow-D uses a separate locally conditioned prediction for each acting agent. In the one-step

case,

$$\begin{aligned} \hat{\boldsymbol{\tau}}_0 &= \mathbf{z}_1 - \hat{\mathbf{u}}_\theta(\mathbf{z}_1, 0, 1 \mid \mathsf{c}_q, R), && \text{CoFlow-C}, \\ \hat{\boldsymbol{\tau}}_0^{(i)} &= \mathbf{z}_1^{(i)} - \hat{\mathbf{u}}_\theta(\mathbf{z}_1^{(i)}, 0, 1 \mid \mathsf{c}_q^{(i)}, R), && \text{CoFlow-D}, \end{aligned} \tag{24}$$

where R is the target return and $\mathsf{c}_q^{(i)} = m_i(\mathcal{H}_q)$ masks unavailable teammate observations. Parenthesized (i) indexes the local joint prediction made by agent i.

Agent i decodes only its own output block. CoFlow-D shares CoFlow-C's trained weights and schedule. Within each locally conditioned joint prediction, cross-agent attention models teammate behavior by mixing generated trajectory features, without accessing unavailable teammate observations or hidden states.

Return guidance. For every interval $[r, t]$, define

$$\begin{aligned} \hat{\mathbf{u}}_\theta(\mathbf{z}, r, t \mid \mathsf{c}, R) = \mathbf{u}_\theta(\mathbf{z}, r, t \mid \mathsf{c}, \varnothing) \\ + \omega \left[\mathbf{u}_\theta(\mathbf{z}, r, t \mid \mathsf{c}, R) - \mathbf{u}_\theta(\mathbf{z}, r, t \mid \mathsf{c}, \varnothing)\right], \end{aligned} \tag{25}$$

where ω weights guidance and $\varnothing$ drops only the return condition, retaining observation context c.

At $\omega = 1$ this reduces to the conditional field. Training uses $\widetilde{R} = \varnothing$ with probability $p_{\text{drop}} = 0.25$, otherwise $\widetilde{R} = R$. Both queries share that choice. With two-query guidance, K sampling steps require $2K$ network evaluations.

B.4 Training and Inference Algorithms

The algorithms use the joint notation of the main text, with objective

$$\min_{\theta,\phi} \quad \mathcal{L}(\theta, \phi) = \frac{1}{Nd}\mathbb{E}\|\mathbf{V}_\theta - \mathbf{v}_c\|_2^2 + \frac{\lambda}{N}\sum_i \mathbb{E}\ell_{\text{act}}\left(a_q^i, I_\phi^i(o_q^i, o_{q+1}^i); \mathcal{U}_q^i\right). \tag{26}$$

Sample two independent logistic transforms of $\mathcal{N}(\mu_{\text{time}}, \sigma_{\text{time}}^2)$ with $\mu_{\text{time}} = -0.4$ and $\sigma_{\text{time}} = 1$, then sort them into $r \le t$. A fraction ρ is set to $r = t$. In the algorithms, $\mathcal{D}$ denotes offline data, η_{lr} the learning rate, and $\boldsymbol{\gamma}$ the gate collection. The temporary predictions $\mathbf{b}$, $\mathbf{b}_+$ and $\mathbf{w}_k$ denote the base, shifted, and guided queries.

Algorithm 1 CoFlow finite-difference training

Require: Offline data $\mathcal{D}$, velocity model $\mathbf{u}_\theta$, inverse dynamics I_ϕ, ρ, ε_0, p_{drop}, λ, learning rate η_{lr}

Initialize CVA gates $\boldsymbol{\gamma} = 0$ as part of θ
for each minibatch **do**
  Sample joint trajectories, valid action transitions, legal-action sets, context c, and return R from $\mathcal{D}$
  Draw joint noise $\mathbf{z}_1 \sim \mathcal{N}(0, \mathbf{I}_{Nd})$ independently of the clean trajectory
  Sample ordered time pairs $r \le t$ as above and set $r = t$ for a fraction ρ
  Set $\widetilde{R} = \varnothing$ with probability p_{drop}, and $\widetilde{R} = R$ otherwise
  Build $\mathbf{z}_t = (1 - t)\boldsymbol{\tau}_0 + t\mathbf{z}_1$ and $\mathbf{v}_c = \mathbf{z}_1 - \boldsymbol{\tau}_0$
  Evaluate the joint base prediction $\mathbf{b} = \mathbf{u}_\theta(\mathbf{z}_t, r, t \mid \mathsf{c}, \widetilde{R})$ with parameter gradients
  if $r < t$ (per example) **then**
    Choose ε by Equation (21)
    Evaluate $\mathbf{b}_+ = \mathbf{u}_\theta(\mathbf{z}_t + \varepsilon\mathbf{v}_c, r, t + \varepsilon \mid \mathsf{c}, \widetilde{R})$ without gradients, keeping the same stochastic masks
    Set $\boldsymbol{\delta}_{\text{FD}} = \text{stopgrad}[(\mathbf{b}_+ - \text{stopgrad}(\mathbf{b}))/\varepsilon]$
  else
    Set $\boldsymbol{\delta}_{\text{FD}} = \mathbf{0}$ and skip the shifted query
  end if
  Set $\mathbf{V}_\theta = \mathbf{b} + (t - r)\boldsymbol{\delta}_{\text{FD}}$
  Compute minibatch estimates of Equations (8) and (9) using the same normalization
  Update $(\theta, \phi) \leftarrow (\theta, \phi) - \eta_{\text{lr}}\nabla_{\theta,\phi}(\mathcal{L}_{\text{vel}} + \lambda\mathcal{L}_{\text{act}})$
end for
return Trained $\mathbf{u}_\theta$ and I_ϕ

Algorithm 2 CoFlow joint generation and action decoding

Require: Trained $\mathbf{u}_\theta$, I_ϕ, available context, return R, legal actions, schedule $1 = t_K > \cdots > t_0 = 0$, guidance ω, execution mode

1: CoFlow-C performs one run with context c_q and decodes all agents
2: CoFlow-D performs the following run locally for each acting agent i, with context $\mathsf{c}_q^{(i)}$, and decodes only block i
3: **for** each required planning run, with its context denoted by c **do**
4: Sample a joint noise tensor $\mathbf{z}_{t_K} \sim \mathcal{N}(0, \mathbf{I}_{Nd})$
5: **for** $k = K, K-1, \ldots, 1$ **do**
6: Evaluate the joint guided field $\mathbf{w}_k = \hat{\mathbf{u}}_\theta(\mathbf{z}_{t_k}, t_{k-1}, t_k \mid \mathsf{c}, R)$ using Equation (25)
7: Update the joint trajectory: $\mathbf{z}_{t_{k-1}} = \mathbf{z}_{t_k} - (t_k - t_{k-1})\mathbf{w}_k$
8: **end for**
9: Set $\hat{\boldsymbol{\tau}}_0 = \mathbf{z}_{t_0}$, and for each agent i executed by this run, read $\hat{o}_{q+1}^i$ from block i
10: Decode $a_q^i = \mathrm{Dec}_i(I_\phi^i(o_q^i, \hat{o}_{q+1}^i); \mathcal{U}_q^i)$ for the executed agents
11: **end for**
12: **return** Joint action for CoFlow-C, or each acting agent's own action for CoFlow-D

For $K = 1$, the loop gives $\hat{\boldsymbol{\tau}}_0 = \mathbf{z}_1 - \hat{\mathbf{u}}_\theta(\mathbf{z}_1, 0, 1 \mid \mathsf{c}, R)$, as in the main text. Sampling uses the averaged field alone, without the training correction in $\mathbf{V}_\theta$.

CoFlow-base variant. Removing the finite-difference target yields ordinary conditional-velocity regression, with predictor $\mathbf{v}_\theta(\mathbf{z}_t, t) \equiv \mathbf{u}_\theta(\mathbf{z}_t, 0, t)$ and loss $\mathcal{L}_{\text{flow}}$ in Equation (12). The shared CVA backbone and inverse-dynamics head are unchanged. This ablation learns an instantaneous field and uses Euler integration during sampling.

Approximation error. Under local smoothness, Theorem B.1 bounds the finite-difference target perturbation relative to the JVP target by $O((t-r)|\varepsilon|)$. This local bound quantifies the accuracy of the training correction at a fixed network state.

Part III: Theoretical Justification

B.5 Taylor control of the finite-difference surrogate

The finite difference shifts the full joint state along $(\mathbf{v}_c, 1)$ while holding r and the conditioning context fixed. This includes every cross-agent Jacobian block in Equation (19). The result below covers both signs of the step used in Equation (6).

Proposition B.1 (Taylor control for the FD surrogate)**.** *Fix $\mathbf{z}, \mathbf{v}_c \in \mathbb{R}^{Nd}$, $0 \le r < t \le 1$, and $h = t - r$. Let $\varepsilon \neq 0$ satisfy $r \le t + \varepsilon \le 1$, with $\mathcal{I}_\varepsilon = [\min(0, \varepsilon), \max(0, \varepsilon)]$. Assume $\mathbf{u}_\theta(\cdot, r, \cdot)$ is twice continuously differentiable along the query segment. Define*

$$D_\varepsilon \mathbf{u}_\theta(\mathbf{z}, r, t; \mathbf{v}_c) = \frac{\mathbf{u}_\theta(\mathbf{z} + \varepsilon \mathbf{v}_c, r, t + \varepsilon) - \mathbf{u}_\theta(\mathbf{z}, r, t)}{\varepsilon}, \tag{27}$$

where D_ε approximates $\mathcal{D}_{\mathbf{v}_c}$ by a finite difference.

The local curvature is bounded by

$$M_2^i = \sup_{a \in \mathcal{I}_\varepsilon} \left\| \frac{d^2}{da^2} u_\theta^i(\mathbf{z} + a\mathbf{v}_c, r, t + a) \right\|_2, \qquad M_2 = \left(\sum_i (M_2^i)^2 \right)^{1/2}, \tag{28}$$

where M_2^i bounds the local second directional derivative and M_2 is the joint bound.

The directional derivative in Equation (19) satisfies

$$D_\varepsilon \mathbf{u}_\theta = \mathcal{D}_{\mathbf{v}_c} \mathbf{u}_\theta + \mathbf{R}_\varepsilon, \qquad \|\mathbf{R}_\varepsilon\|_2 \le \tfrac{1}{2}|\varepsilon| M_2, \tag{29}$$

where $\mathbf{R}_\varepsilon$ is the joint Taylor remainder. Each block obeys $\|R_\varepsilon^i\|_2 \le \frac{1}{2}|\varepsilon| M_2^i$.

Consequently, for the FD and JVP corrected predictions at the same query,

$$\|\mathbf{V}_\theta - \mathbf{V}_\theta^{\text{JVP}}\|_2 \le \tfrac{1}{2} h |\varepsilon| M_2. \tag{30}$$

At $r = t$, both corrected predictions reduce to $\mathbf{u}_\theta$ and their difference is zero without any finite-difference evaluation.

Proof. Taylor's theorem for the joint query path gives, for either sign of ε,

$$\mathbf{g}(\varepsilon) = \mathbf{g}(0) + \varepsilon \mathbf{g}'(0) + \varepsilon^2 \int_0^1 (1-s)\mathbf{g}''(s\varepsilon)\, ds, \tag{31}$$

where $\mathbf{g}(a) = \mathbf{u}_\theta(\mathbf{z} + a\mathbf{v}_c, r, t + a)$. By the curvature definition, $\|\mathbf{g}''(a)\|_2 \le M_2$ along the segment. The chain rule gives $\mathbf{g}'(0) = \mathcal{D}_{\mathbf{v}_c}\mathbf{u}_\theta$. Dividing by ε identifies

$$\mathbf{R}_\varepsilon = \varepsilon \int_0^1 (1-s)\mathbf{g}''(s\varepsilon)\, ds, \qquad \|\mathbf{R}_\varepsilon\|_2 \le |\varepsilon| M_2 \int_0^1 (1-s)\, ds. \tag{32}$$

This proves Equation (29). Applying the same argument to block i gives its bound. Gradient stopping leaves numerical values unchanged, and multiplication by h gives Equation (30). □

This is a per-sample target bound. If $\mathbb{E}[h^2|\varepsilon|^2 M_2^2] < \infty$, its squared joint expectation is at most $\frac{1}{4}\mathbb{E}[h^2|\varepsilon|^2 M_2^2]$. Dividing by Nd expresses this target approximation bound as a mean over coordinates, as in the training loss.

B.6 Joint Velocity Decomposition

Gate-off decomposition used for interventions. Let $\boldsymbol{\gamma}$ collect all learned CVA gates. The full model has the functional decomposition

$$u_\theta^i(\mathbf{z}_t, r, t) = \underbrace{u_{\theta,0}^i(z_t^i, r, t)}_{u_{\text{ind}}^i} + \underbrace{\left[u_\theta^i(\mathbf{z}_t, r, t) - u_{\theta,0}^i(z_t^i, r, t)\right]}_{u_{\text{coord}}^i}, \tag{33}$$

where $u_{\theta,0}^i$ disables all gates while holding other weights fixed.

The theorem below uses a self-only reference that retains the gated transformation of the agent's own value features. Its cross-agent term $\boldsymbol{\Delta}_{\text{attn}}$ is therefore distinct from the entire gate-off difference u_{coord}. The bound applies to $\boldsymbol{\Delta}_{\text{attn}}$.

To bound the cross-agent residual, we analyze an idealized sequence of shared residual attention layers with non-expansive agent-local maps, holding observation conditions and flow times fixed.

Theorem B.2 (Joint velocity decomposition under non-expansive blocks)**.** *Consider L layers and M heads with the shared feature recursion*

$$f_l^i = g_l\left(f_{l-1}^i + \gamma_l \sum_{m=1}^{M}\sum_{j=1}^{N} \alpha_{lm}^{ij} B_{lm} f_{l-1}^j\right), \quad \alpha_{lm}^{ij} \ge 0, \quad \sum_j \alpha_{lm}^{ij} = 1, \tag{34}$$

where f_l^i are layer features, α_{lm}^{ij} is head-m attention from j to i, and B_{lm} combines its value/output maps. The shared maps g_l and the final output map are assumed non-expansive.

Input features are produced independently per agent. Define

$$\bar{\sigma}_l = |\gamma_l| \sum_m \|B_{lm}\|_{\text{op}}, \qquad \bar{\sigma} = \max_l \bar{\sigma}_l, \qquad D_l = \max_{i,j} \|f_l^i - f_l^j\|_2, \tag{35}$$

where $\bar{\sigma}_l$ is the combined scale of the gates and projection matrices and D_l the feature diversity, with input diversity D_0.

The network decomposes into self-only predictions and a cross-agent residual

$$\mathbf{u}_\theta(\mathbf{z}, r, t) = \bar{\mathbf{u}}_\theta(\mathbf{z}, r, t) + \boldsymbol{\Delta}_{\text{attn}}(\mathbf{z}, r, t), \tag{36}$$

where $\bar{\mathbf{u}}_\theta = (\bar{u}_\theta^i(z^i, r, t))_{i=1}^N$ stacks the self-only predictions and $\boldsymbol{\Delta}_{\text{attn}}$ is the cross-agent residual.

The self-only reference replaces each attention row by weight one on its own agent, retaining the gated self transformation. Then

$$\|\boldsymbol{\Delta}_{\text{attn}}\|_2 \le \frac{\sqrt{N}}{2}\left[(1+3\bar{\sigma})^L - (1+\bar{\sigma})^L\right] D_0. \tag{37}$$

When $L\bar{\sigma} \ll 1$, its leading-order expansion is $\sqrt{N} L \bar{\sigma} D_0$. The factor $\sqrt{N}$ disappears after per-agent normalization by $N^{-1/2}$.

Proof. Split the attention input into its self transformation and cross-agent message:

$$\begin{aligned} b_l^i &= \gamma_l \sum_m \sum_j \alpha_{lm}^{ij} B_{lm}(f_{l-1}^j - f_{l-1}^i), \\ f_l^i &= g_l\left[\left(I + \gamma_l \sum_m B_{lm}\right) f_{l-1}^i + b_l^i\right], \qquad \|b_l^i\|_2 \le \bar{\sigma}_l D_{l-1}, \end{aligned} \tag{38}$$

where b_l^i is the injected cross-agent message.

The bound on b_l^i follows from nonnegative row-normalized attention and the triangle inequality. Since the same g_l is non-expansive for all agents,

$$D_l \le (1 + \bar{\sigma}_l) D_{l-1} + 2\bar{\sigma}_l D_{l-1} \le (1 + 3\bar{\sigma})^l D_0. \tag{39}$$

Starting from $\bar{f}_0^i = f_0^i$, define the self-only reference recursively by

$$\bar{f}_l^i = g_l\left[\left(I + \gamma_l \sum_m B_{lm}\right) \bar{f}_{l-1}^i\right], \quad \delta_l^i = f_l^i - \bar{f}_l^i, \quad E_l = \left(\sum_i \|\delta_l^i\|_2^2\right)^{1/2}, \tag{40}$$

where $\bar{f}_l^i$ is the self-only feature and E_l the joint perturbation norm.

With $E_0 = 0$, non-expansiveness and the joint norm of the injected messages imply

$$E_l \le (1 + \bar{\sigma}_l) E_{l-1} + \sqrt{N}\bar{\sigma}_l D_{l-1}. \tag{41}$$

Iterating and applying the non-expansive output map gives

$$\begin{aligned} \|\mathbf{\Delta}_{\text{attn}}\|_2 &\le \sqrt{N}\bar{\sigma} \sum_{l=1}^{L} (1 + \bar{\sigma})^{L-l} (1 + 3\bar{\sigma})^{l-1} D_0 \\ &= \frac{\sqrt{N}}{2} \left[(1 + 3\bar{\sigma})^L - (1 + \bar{\sigma})^L\right] D_0. \end{aligned} \tag{42}$$

The zero-gate case follows by continuity. Expanding the bracket for $L\bar{\sigma} \ll 1$ yields $2L\bar{\sigma} + O(L^2\bar{\sigma}^2)$. If $D_0 = 0$, the cross-agent correction is zero under the stated shared-map assumptions. Extending this analysis to expansive or agent-dependent maps requires incorporating their Lipschitz factors and cross-agent differences into the recurrences. □

The expansion requires $L\bar{\sigma} \ll 1$, accounting for both gate magnitudes and projection norms.

B.7 One-Step Wasserstein Error Bound

We relate approximation of the averaged velocity in Equation (2) to the distribution of one-step samples. All observation and return conditions are fixed and suppressed. This result concerns the unguided averaged-velocity predictor. It also applies to an effective guided field if that field satisfies the same approximation assumption for the chosen target transport.

Assumption B.3 (Marginal transport and endpoint approximation)**.** The marginal flow admits a measurable endpoint map $\mathcal{T} = \Phi_{0\leftarrow 1}$ such that $\mathcal{T}(\mathbf{z}_1)$ has law p_τ for $\mathbf{z}_1 \sim \mathcal{N}(0, \mathbf{I})$. The relevant distributions have finite second moments, and the ideal endpoint averaged velocity is

$$\mathbf{u}(\mathbf{z}_1, 0, 1) = \mathbf{z}_1 - \mathcal{T}(\mathbf{z}_1). \tag{43}$$

Define the network's endpoint approximation error by

$$\epsilon_{\text{vel}}^2 := \mathbb{E}_{\mathbf{z}_1} \|\mathbf{u}_\theta(\mathbf{z}_1, 0, 1) - \mathbf{u}(\mathbf{z}_1, 0, 1)\|^2 < \infty, \tag{44}$$

where ϵ_{vel} is the unnormalized joint endpoint RMS error.

The oracle is defined by the displacement of the marginal transport.

Theorem B.4 (One-step Wasserstein error bound). *Under Theorem B.3, the one-step sample* $\hat{\boldsymbol{\tau}}_0 = \mathbf{z}_1 - \mathbf{u}_\theta(\mathbf{z}_1, 0, 1)$ *satisfies*

$$W_2(p_{\hat{\boldsymbol{\tau}}_0}, p_\tau) \le \epsilon_{\text{vel}}. \tag{45}$$

For any fixed decomposition of the oracle into per-agent terms and a joint residual,

$$\epsilon_{\text{vel}} \le \left(\sum_{i=1}^{N} \epsilon_i^2 \right)^{1/2} + \epsilon_{\text{coord}}, \tag{46}$$

where ϵ_i *and* ϵ_{coord} *are the RMS errors of the self-only terms and joint residual.*

Proof. Set $\boldsymbol{\tau}_{\text{ref}} = \mathcal{T}(\mathbf{z}_1) = \mathbf{z}_1 - \mathbf{u}(\mathbf{z}_1, 0, 1)$, whose law is p_τ by assumption. Coupling $\hat{\boldsymbol{\tau}}_0$ and $\boldsymbol{\tau}_{\text{ref}}$ through the same noise gives

$$\begin{aligned} W_2(p_{\hat{\boldsymbol{\tau}}_0}, p_\tau) &\le \left(\mathbb{E} \|\hat{\boldsymbol{\tau}}_0 - \boldsymbol{\tau}_{\text{ref}}\|^2 \right)^{1/2} \\ &= \left(\mathbb{E} \|\mathbf{u}_\theta - \mathbf{u}\|^2 \right)^{1/2} = \epsilon_{\text{vel}}, \end{aligned} \tag{47}$$

where omitted velocity arguments are $(\mathbf{z}_1, 0, 1)$.

For the second statement, write

$$\mathbf{u}_\theta - \mathbf{u} = (\bar{\mathbf{u}}_\theta - \bar{\mathbf{u}}) + (\boldsymbol{\Delta}_{\text{attn}} - \boldsymbol{\Delta}), \tag{48}$$

where $\bar{\mathbf{u}} = (\bar{u}^i)_{i=1}^N$ and $\boldsymbol{\Delta}$ are the oracle's chosen self-only terms and joint residual, evaluated at the same endpoint.

Define

$$\epsilon_i^2 = \mathbb{E} \|\bar{u}_\theta^i - \bar{u}^i\|^2, \qquad \epsilon_{\text{coord}}^2 = \mathbb{E} \|\boldsymbol{\Delta}_{\text{attn}} - \boldsymbol{\Delta}\|^2. \tag{49}$$

The triangle inequality in L^2, followed by orthogonality of the agent output blocks, yields

$$\begin{aligned} \epsilon_{\text{vel}} &\le \left(\mathbb{E} \|\bar{\mathbf{u}}_\theta - \bar{\mathbf{u}}\|^2 \right)^{1/2} + \epsilon_{\text{coord}} \\ &= \left(\sum_i \epsilon_i^2 \right)^{1/2} + \epsilon_{\text{coord}}. \end{aligned} \tag{50}$$

□

Relation to finite-difference training. Theorem B.1 bounds the local approximation error of the finite-difference training target, while Theorem B.4 bounds one-step sample error given endpoint averaged-velocity accuracy. Linking the training loss to endpoint accuracy requires control of the time distribution, approximation, and optimization.

B.8 Scaling with the number of agents

Assume $\epsilon_i \le \bar{\epsilon}$ and that ϵ_{coord} is bounded by the architectural envelope times ϵ_{attn}. Under the assumptions of Theorem B.2 and its expansion for $L\bar{\sigma} \ll 1$,

$$\epsilon_{\text{vel}} \lesssim \sqrt{N}\,\bar{\epsilon} + \sqrt{N}\,L\,\bar{\sigma}\,D_N\,\epsilon_{\text{attn}} \qquad (L\bar{\sigma} \ll 1), \tag{51}$$

where $D_N = \max_{i \ne j} \|f_0^i - f_0^j\|$ is the initial diversity for N agents.

The leading $\sqrt{N}$ factor is a consequence of measuring a joint vector in Euclidean norm. It disappears after per-agent normalization. Thus coordination scalability depends on both team size and the controlled growth of feature diversity D_N and learned gate scale $\bar{\sigma}$.