LEARNING COORDINATED REPRESENTATIONS FOR FLOW-BASED MULTI-AGENT IMITATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Cooperative multi-agent systems require decentralized agents to coordinate seamlessly to accomplish shared tasks. Learning from multi-modal joint demonstrations that exhibit diverse strategies makes multi-agent imitation learning particularly challenging: independently executed policies may select individually plausible actions that accidentally mix distinct strategies, leading to inconsistent joint behaviors. We address this challenge by introducing the latent-conditioned factorization gap, which measures the divergence between a centralized joint policy and a product of latent-conditioned decentralized policies. We show that, with sufficiently expressive local policies, this gap decomposes into conditional total correlation among agents' actions and conditional mutual information terms capturing dependence on unobserved global information. To efficiently optimize this objective with flow-matching policies, we derive a practical, density-free surrogate based on joint-to-local vector field alignment. Extensive experiments on Multi-Agent MuJoCo and cooperative multi-quadruped benchmarks demonstrate that our approach significantly outperforms existing baselines in preserving coordinated multi-modal behavior.

1 INTRODUCTION

Cooperative multi-agent systems (Wang et al., 2022; Guestrin et al., 2002) require multiple agents to coordinate their decisions under decentralized execution. Offline multi-agent imitation learning (MAIL) typically trains decentralized policies by maximizing the log-likelihood of individual actions from pre-collected joint expert demonstrations. Standard approaches factorize the joint policy independently across agents under the implicit assumption of unimodal optimal joint behavior (Bui et al., 2024; 2025). However, expert demonstrations often exhibit substantial multi-modality, as the same task can be completed through distinct yet equally valid joint strategies (e.g., assigning different roles to agents). In such settings, successful imitation requires modeling crucial inter-agent dependencies; agents must coordinate on actions consistent with a single, coherent joint strategy. Without such coordination, decentralized execution can accidentally mix actions from different demonstrated strategies, producing joint behaviors that are individually plausible but collectively inconsistent. Recovering these inter-agent dependencies under decentralized execution remains a central challenge in multi-agent learning (Le et al., 2017).

To handle multi-modal demonstrations in single-agent settings, prior works have leveraged latent-conditioned models (Li et al., 2017; Hausman et al., 2017) and expressive generative models such as diffusion models (Chi et al., 2025; Choi et al., 2025) and flow-matching models (Rouxel et al., 2024). These methods demonstrate that expressive policy classes can avoid mode-averaging and recover multiple modes from such demonstrations. Similarly, multi-agent frameworks have extended these generative paradigms to decentralized policies to achieve a similar goal (Zhu et al., 2024; Dong et al., 2025; Lee et al., 2026). One common approach is to introduce an episode-level latent variable shared across agents and learn latent-conditioned decentralized policies (Le et al., 2017). Specifically, this shared latent variable specifies a joint strategy, enabling agents to coordinate their actions without requiring per-step communication. Conventionally, this latent representation is trained by encoding joint behavior into z and decoding it through individual policies, primarily optimizing a reconstruction objective. Additionally, mutual information objectives (Li et al., 2017; Hausman et al., 2017) can be incorporated to encourage the latent variable to distinguish between different joint trajectories. However, neither objective explicitly requires the latent representation to capture the inter-agent dependencies necessary for joint behaviors to be executable as a product of decentralized policies.

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.

This motivates us to ask: rather than assessing whether the latent variable merely reconstructs or distinguishes joint behaviors, *does conditioning on the latent variable render the demonstrated joint behavior factorizable into decentralized policies?* To answer this, we propose the latent-conditioned factorization gap, measuring the divergence between a latent-conditioned centralized joint policy and a product of latent-conditioned decentralized policies. Minimizing this discrepancy with respect to the encoder enables the latent representation to capture precisely the inter-agent dependencies that otherwise impede decentralized execution. However, evaluating this factorization gap with modern expressive policies, such as flow-matching models, introduces severe computational challenges due to the high cost of density evaluation. To bypass explicit density estimation, we relate this divergence to the discrepancy in their vector fields along a shared Gaussian interpolation schedule. Crucially, a zero factorization gap corresponds to exact alignment between the centralized joint vector field and the concatenated decentralized vector fields. This result yields a practical, density-free surrogate that optimizes the latent representation by measuring the vector field discrepancy between each agent's decentralized policy and its corresponding component in the joint vector field.

We evaluate our proposed approach on two diverse multi-robot coordination suites, including MAMuJoCo (Peng et al., 2021) and MQE (Xiong et al., 2024), using multi-modal demonstration datasets. In particular, MQE features cooperative multi-quadruped tasks that exhibit diverse strategies, such as agent ordering, role assignment, and target selection. Across these settings, we evaluate both task performance and demonstration recovery against a broad set of baselines: Gaussian behavior cloning, latent-conditioned and mutual information methods, flow-based behavior cloning, and joint-flow distillation. Our results demonstrate that learning latent representations explicitly tailored for decentralized factorization is essential, whereas high policy expressiveness or mode-separation objectives alone are insufficient.

Our contributions are summarized as follows:

1. **Factorization view of multi-modal multi-agent imitation** We formulate latent representation learning around the requirement that demonstrated joint behaviors become conditionally representable by a product of decentralized policies, rather than merely requiring the latent variable to distinguish different demonstration modes.
2. **Information-theoretic characterization of decentralized coordination** We show that the latent-conditioned factorization gap decomposes into conditional total correlation among the agents' actions and conditional mutual-information terms measuring dependence on observations unavailable to each local policy.
3. **Density-free factorization objective for flow policies** Using the relationship between flow vector fields and scores, we connect the centralized-to-decentralized distributional discrepancy to vector-field discrepancies and derive a practical joint-to-local vector field alignment objective for learning the latent representation and decentralized policies.
4. **Evaluation under diverse coordination modes** We evaluate the approach on MAMuJoCo and MQE cooperative tasks involving different movement directions, role assignments, and execution orders.

2 Method

2.1 Background: Flow-matching Models with Gaussian Interpolants

Flow-matching learns a time-dependent vector field $v_{\theta,\lambda}$ that transports samples from a simple source distribution p_0 to a target data distribution p_1 (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023; Ma et al., 2024). Given flow time $\lambda \in [0, 1]$, the generative process is governed by the ordinary differential equation (ODE):

$$\frac{dx_\lambda}{d\lambda} = v_{\theta,\lambda}(x_\lambda), \qquad x_0 \sim p_0.$$

To avoid backpropagating through the ODE solver during training, flow-matching uses a continuous interpolation path between $x_0 \sim p_0$ and $x_1 \sim p_1$. In particular, we consider the affine Gaussian interpolation path f_λ:

$$x_\lambda = f_\lambda(x_0, x_1) = \alpha_\lambda x_1 + \sigma_\lambda x_0, \qquad x_0 \sim \mathcal{N}(0, I), \tag{1}$$

where α_λ and σ_λ are differentiable λ-dependent coefficients satisfying boundary conditions $(\alpha_0, \sigma_0) = (0, 1)$ and $(\alpha_1, \sigma_1) = (1, 0)$. The vector field $v_{\theta,\lambda}$ is regressed against the conditional

velocity $u_\lambda(x_0, x_1) = \frac{\partial}{\partial\lambda} f_\lambda(x_0, x_1)$:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{x_0,x_1,\lambda}\left[\|v_{\theta,\lambda}(x_\lambda) - u_\lambda(x_0, x_1)\|^2\right]. \tag{2}$$

A common choice of f_λ is the linear path (Liu et al., 2023), where $\alpha_\lambda = \lambda$ and $\sigma_\lambda = 1 - \lambda$, yielding $x_\lambda = \lambda x_1 + (1-\lambda)x_0$ and $u_\lambda(x_0, x_1) = x_1 - x_0$.

Relation between velocity field and score Let $p_\lambda(x_\lambda)$ denote the marginal density of x_λ in Eq. 1, and $s_\lambda(x_\lambda) := \nabla_{x_\lambda} \log p_\lambda(x_\lambda)$ be its score function. For affine Gaussian interpolants, the marginal vector field and score are equivalent parameterizations of the continuous probability path (Albergo & Vanden-Eijnden, 2023; Ma et al., 2024).

Lemma 2.1 (Vector–Score relation). *Consider the affine Gaussian interpolation in Eq. 1 with $\alpha_\lambda > 0$ and $\sigma_\lambda > 0$ for $\lambda \in (0, 1)$. The population-optimal vector field $v_{\star,\lambda}$, the minimizer of Eq. 2, satisfies*

$$v_{\star,\lambda}(x_\lambda) = \frac{\dot\alpha_\lambda}{\alpha_\lambda} x_\lambda + \kappa_\lambda \nabla_{x_\lambda} \log p_\lambda(x_\lambda), \tag{3}$$

where $\dot\alpha_\lambda = \frac{d\alpha_\lambda}{d\lambda}$, $\dot\sigma_\lambda = \frac{d\sigma_\lambda}{d\lambda}$, and $\kappa_\lambda := \frac{\dot\alpha_\lambda \sigma_\lambda^2}{\alpha_\lambda} - \dot\sigma_\lambda \sigma_\lambda$. See Appendix B.1 for the detailed proof.

2.2 Problem Setup: Cooperative Multi-Agent Imitation Learning

Cooperative multi-agent systems aim to accomplish a shared task through decentralized execution under partial observability. Formally, agent $i \in \{1, \dots, N\}$ receives an observation $o^i \in \mathcal{O}^i$, and executes an action $a^i \in \mathcal{A}^i$ via its policy $\pi^i : \mathcal{O}^i \to \mathcal{A}^i$. We define the joint observation $o = (o^1, \dots, o^N) \in \mathcal{O}$ and joint action $a = (a^1, \dots, a^N) \in \mathcal{A}$ over the joint spaces $\mathcal{O} = \prod_{i=1}^N \mathcal{O}^i$ and $\mathcal{A} = \prod_{i=1}^N \mathcal{A}^i$. Offline MAIL seeks to learn a set of decentralized policies $\{\pi^i\}_{i=1}^N$ from a pre-collected dataset $\mathcal{D} = \{\tau\}$ without environment interactions. Here, $\tau = \{o^{(t)}, a^{(t)}\}_{t=0}^H$ denotes an H-step joint trajectory, where $o^{(t)} = \{o^{i,(t)}\}_{i=1}^N$ and $a^{(t)} = \{a^{i,(t)}\}_{i=1}^N$ represent the joint observation $o \in \mathcal{O}$ and joint action $a \in \mathcal{A}$ at time step t. In this setting, the standard objective is to maximize the joint log-likelihood over the dataset (i.e., behavior cloning or BC (Bain & Sammut, 1995; Zhan et al., 2018)) under the assumption of independence among agents (Bui et al., 2024; 2025): $\mathbb{E}_{\mathcal{D}}\left[\sum_{i=1}^N \log \pi^i(a^i \mid o^i)\right]$.

In real-world scenarios, expert demonstrations often exhibit multi-modality due to diverse yet equally valid strategies for completing the task. Under such multi-modal demonstrations, fitting Gaussian policies leads to mode-averaging. To model these complex distributions without mode-averaging, we can adopt expressive flow-matching models for our decentralized policies. However, simply enhancing the expressiveness of individual policies does not prevent coordination failure under decentralized execution, as shown in Figure 1. This is because the independence assumption forces agents to sample their actions independently across different behavioral modes. Consequently, agents fail to coordinate on a single consistent strategy, leading to severe performance degradation.

Latent-conditioned policies To address this, we utilize decentralized policies conditioned on a shared latent variable $\pi^i(a^i|o^i, z)$ (Le et al., 2017), where $z \sim p(z)$ is a discrete latent variable. At the beginning of each episode, z is sampled once, shared among all agents, and kept fixed throughout the episode. By structuring z to represent distinct joint behavior modes present in the demonstrations, agents can synchronize their execution strategies across modes without explicit online communication. By introducing an encoder $q_\phi(z|\tau)$, we minimize the negative evidence lower bound (ELBO) on the joint log-likelihood, which consists of q_ϕ and $\{\pi_\theta^i\}_{i=1}^N$. Parameterizing π_θ^i via latent-conditioned vector fields $v_{\theta,\lambda}^i$, the flow-matching loss becomes

$$\mathcal{L}_{\text{FM}}^{\text{ELBO}}(\phi, \theta) = \mathbb{E}_{\substack{\tau\sim\mathcal{D},\lambda,\\ z\sim q_\phi(z|\tau)}}\left[\sum_{t=0}^{H}\sum_{i=1}^{N}\left\|v_{\theta,\lambda}^i(a_\lambda^{i,(t)}, o^{i,(t)}, z) - u_\lambda(\epsilon^{i,(t)}, a^{i,(t)})\right\|^2\right] + D_{\text{KL}}(q_\phi(\cdot|\tau)\,\|\,p(\cdot)). \tag{4}$$

Here, analogous to the flow-matching formulation in Section 2.1, $a^{i,(t)}$ denotes the action from the dataset for agent i at time step t (corresponding to the data x_1), $\epsilon^{i,(t)} \sim \mathcal{N}(0, I)$ is the Gaussian noise (corresponding to x_0), and $a_\lambda^{i,(t)} = \alpha_\lambda a^{i,(t)} + \sigma_\lambda \epsilon^{i,(t)}$ is the interpolated action at flow time λ. The first term represents the flow-matching reconstruction loss weighted by $q(z|\tau)$, while the second serves as a KL regularization term. The detailed derivation is in Appendix B.2.

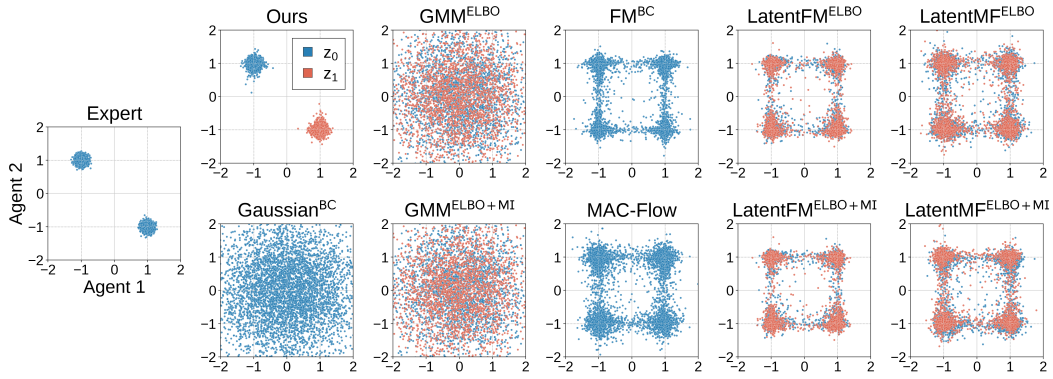


Figure 1: Joint action distribution on the 2D multi-agent toy task. Each agent's action space consist of dependent and independent coordinate actions; the plot visualize the joint dependent action space. The expert dataset contains two coordinated modes at (-1, 1) and (1, 1). Gaussian policies suffers from mode-averaging, while unconditioned flow baselines fail to maintain inter-agent dependencies, generating uncoordinated actions at (-1,-1) and (1,1). Latent baselines fail to disentangle joint modes across z. In contrast, our method structures the latent space effectively.

Difficulty of learning z to represent coordination Despite incorporating the latent variable z, optimizing the ELBO objective is insufficient for capturing complex agent coordination in practice. We conduct a toy experiment comparing Gaussian and flow-matching policies across BC and ELBO. As shown in Figure 1, fully factorized flow policies (FM^{BC}) produce uncoordinated actions, while ELBO-trained latent flow models ($\text{LatentFM}^{\text{ELBO}}$) fail to recover the joint behavior modes. Moreover, explicitly encouraging behavior separation by maximizing mutual information (MI) $I(z;\tau)$ ($\text{LatentFM}^{\text{ELBO+MI}}$) also fails to capture these coordinated modes. This stems from the fact that ELBO and MI objectives consider entire trajectory-level information without distinguishing between coordinated and independent factors. Because observations or independent action components contribute to the objectives, z is easily dominated by these components, failing to isolate the action subspace that exhibits inter-agent behavior dependencies. These results suggest that neither the ELBO nor the MI objective sufficiently forces z to encode the dependencies among agents. This limitation motivates a more rigorous representation learning objective for z. The detailed experimental setup is provided in Appendix C.1.

2.3 CoFac-FM: Coordinated Factorized Flow-Matching

Latent-conditioned factorization gap To overcome these limitations, we seek a latent representation z that enables the joint expert behavior to become conditionally independent, allowing the joint policy to factorize into decentralized policies. To this end, we define the latent-conditioned factorization gap as the discrepancy between a centralized policy $\pi_\eta^J(a|o,z)$ and the product of decentralized policies $\prod_i \pi_\theta^i(a^i|o^i,z)$:

$$\Delta_{\text{fac}}(\phi,\theta) = \mathbb{E}_{\mathcal{D},z\sim q_\phi(z|\tau)}\left[D_{\text{KL}}\left(\pi_\eta^J(\cdot \mid o,z) \parallel \prod_{i=1}^{N}\pi_\theta^i(\cdot \mid o^i,z)\right)\right] \tag{5}$$

where $\pi_\eta^J : \mathcal{O} \to \mathcal{A}$ is the centralized joint policy modeling joint expert demonstrations. Minimizing Eq. 5 with respect to the encoder encourages q_ϕ to organize the demonstrations such that, conditioned on z, the joint expert behavior factorizes into decentralized policies. In this way, z captures the inter-agent dependencies required for coordination, rendering individual agent actions conditionally independent given z. Thus, the latent variable effectively explains away coordination dependencies that would otherwise prevent successful decentralized factorization.

Information-theoretic interpretation The gap Δ_{fac} in Eq. 5 admits a decomposition that characterizes the sources of dependency that hinder decentralized factorization. Assuming sufficiently

expressive policies π^i_θ, the gap $\Delta_{\star,\text{fac}}(\phi) \equiv \min_\theta \Delta_{\text{fac}}(\phi, \theta)$ yields:

$$\Delta_{\star,\text{fac}}(\phi) = \text{TC}(a^1; \ldots; a^N \mid o, z) + \sum_{i=1}^{N} I(a^i; o^{-i} \mid o^i, z). \tag{6}$$

The first term, conditional Total Correlation (TC) (Watanabe, 1960), measures the residual dependence among the agents' actions given the full observation o and latent variable z, indicating coordination dependencies not captured by z. The second term measures the dependence of each agent's action a^i on the other agents' local observations o^{-i} given o^i and z, capturing information unavailable during decentralized execution. Thus, minimizing Δ_{fac} with respect to q_ϕ encourages z to account for both sources of dependence that hinder decentralized factorization. See Appendix B.3 for the details.

Our objective Although minimizing Δ_{fac} promotes conditional independence, Eq. 5 is computationally expensive when π^J_η and π^i_θ are modeled as flow-matching models. This is because evaluating their exact marginal densities requires integrating vector fields through an ODE solver. To avoid explicit density evaluation, we exploit the relationship between vector fields and score functions established in Lemma 2.1. Rather than directly approximating the KL divergence, we use this relationship to characterize its zero-gap condition as formalized below:

Theorem 2.2 (Flow factorization equivalence)**.** *For a fixed (o, z), let $\pi^J(a) := \pi^J(a \mid o, z)$ and $\pi^I(a) := \prod_{i=1}^N \pi^i(a^i)$, where $\pi^i(a^i) := \pi^i(a^i \mid o^i, z)$. Assume that: (i) the probability paths in Eq. 1 for π^J and each π^i are constructed using the same Gaussian interpolation schedule $(\alpha_\lambda, \sigma_\lambda)$; (ii) $v^J_{\star,\lambda}$ and $v^i_{\star,\lambda}$ for $i \in \{1, \ldots, N\}$ are their respective population-optimal vector fields; and (iii) the induced intermediate densities are strictly positive and sufficiently smooth. Then, we have:*

$$D_{\text{KL}}(\pi^J \parallel \pi^I) = \int_0^1 \frac{1}{\kappa_\lambda} \mathbb{E}_{a\sim\pi^J_\lambda} \left[\sum_{i=1}^{N} \left\| v^{J,i}_{\star,\lambda}(a) - v^i_{\star,\lambda}(a^i) \right\|^2 \right] d\lambda \tag{7}$$

where $v^{J,i}_{\star,\lambda}(a)$ denotes the i-th subvector of $v^J_{\star,\lambda}(a)$ corresponding to agent i. Consequently, $D_{\text{KL}}(\pi^J \parallel \pi^I) = 0$ if and only if $v^{J,i}_{\star,\lambda}(a) = v^i_{\star,\lambda}(a^i)$ for all $a \in \text{supp}(\pi^J_\lambda)$, for all $i \in \{1, \ldots, N\}$, and $\lambda \in (0, 1)$.

Motivated by Theorem 2.2, we propose a practical objective using trainable vector fields $v^J_{\eta,\lambda}$ and $v^i_{\theta,\lambda}$:

$$\Delta^{\text{fac}}_{\text{FM}}(\theta, \phi) = \mathbb{E}_{\mathcal{D},\lambda,q_\phi(z|\tau)} \left[\sum_{i=1}^{N} \left\| v^{J,i}_{\eta,\lambda}(a_\lambda, o, z) - v^i_{\theta,\lambda}(a^i_\lambda, o^i, z) \right\|^2 \right], \tag{8}$$

where $v^{J,i}_{\eta,\lambda}$ denotes the i-th agent's action component of the centralized joint vector field $v^J_{\eta,\lambda}$. Although Eq. 8 does not directly estimate Δ_{fac}, it targets the same zero-gap condition characterized by Theorem 2.2. Minimizing it with respect to θ aligns the decentralized vector fields with the corresponding components of the centralized policy, while minimizing it with respect to ϕ encourages the encoder to produce latent variables z under which the joint behavior becomes factorizable by decentralized policies.

Algorithm 1 CoFac-FM: Coordinated Factorized Flow-Matching

Input: Expert demonstrations $\mathcal{D}$, encoder q_ϕ, individual vector fields $v^i_{\theta,\lambda}$ for $i \in \{1, \ldots, N\}$, joint vector field $v^J_{\eta,\lambda}$.
for iterations **do**
  Sample trajectory batch $\tau \sim \mathcal{D}$ and compute the probabilities $q_\phi(z \mid \tau)$ for all $z \in \mathcal{Z}$.
  Update parameters ϕ, θ, and η using $\mathcal{L}^{\text{ELBO}}_{\text{FM}}$ in Eq. 4.
  Estimate the factorization gap $\Delta^{\text{fac}}_{\text{FM}}$ in Eq. 8 for all $z \in \mathcal{Z}$.
  Update ϕ and θ using the estimated gap $\Delta^{\text{fac}}_{\text{FM}}$ while keeping η fixed.
end for

Practical implementation The overall training procedure for our Coordinated factorized Flow-Matching (Cofac-FM) framework is outlined in Algorithm 1. We train the joint vector field $v^J_{\eta,\lambda}$ using a flow-matching loss analogous to Eq. 4, but defined directly over the joint observation o and joint action a rather than summing over individual agents. To infer z from τ, we use a subsequence of the trajectory than the full trajectory. We first fit both joint and individual vector fields over all discrete latent variables $z \in \mathcal{Z}$ via Eq. 4, and then perform representation learning for the encoder using the factorization gap $\Delta^{\text{fac}}_{\text{FM}}$ from Eq. 8.

3 EXPERIMENTS

We evaluate our method across a diverse range of robot control tasks, including Multi-Agent MuJoCo (MAMuJoCo) (Peng et al., 2021) and Multi-Agent Quadruped Environment (MQE) (Xiong et al., 2024). Built upon NVIDIA Isaac Gym (Makoviychuk et al., 2021), MQE provides a suite of cooperative and competitive multi-agent tasks. To evaluate our method in realistic settings, we extend MQE with several custom environments where multiple quadruped robots collaboratively transport cargo across complex terrains.

3.1 MAMUJOCO

Environmental setup MAMuJoCo is a multi-agent benchmark based on MuJoCo robotic control tasks, where a single robot is partitioned into multiple agents, with each agent controlling a distinct subset of its joints. We conduct experiments on four environments with different agent configurations: **(1) HalfCheetah (2x3, 6x1)** requires controlling the multi-agent system to move in desired directions, including running forward and backward. **(2) Ant (2x4, 4x2)** tasks the multi-agent system with moving in one of four directions, including forward, backward, left or right. **(3) AntGoal (2x4, 4x2)** requires the multi-agent system to navigate toward one of the goal positions evenly distributed along a circle. **(4) ManySegmentSwimmer (5x4, 10x2)** tasks the multi-agent system with swimming in one of four directions, including forward, backward, left, or right. We denote each configuration as Nxd, where N is the number of agents and d is the number of joints controlled by each agent.

For MAMuJoCo environments, we pre-train M expert policies using reinforcement learning, where each policy corresponds to one of M joint behavior modes. We then sample 10 trajectories from each expert policy to construct demonstration datasets.

Baselines We compare our method against the following baselines:

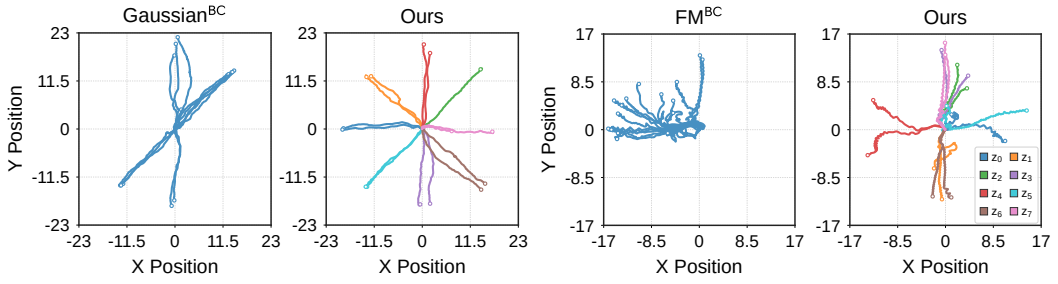
- **Gaussian^{BC}** (Bain & Sammut, 1995; Zhan et al., 2018) learns an independent Gaussian policy for each agent by maximizing the sum of per-agent action log-likelihoods.
- **GMM^{ELBO}** (Le et al., 2017) learns latent-conditioned Gaussian policies for individual agents by optimizing the ELBO objective.
- **GMM$^{\text{ELBO+MI}}$** extends GMM^{ELBO} by adding a mutual information term to encourage the latent variable to capture distinct behavior modes.
- **FM^{BC}** (Lipman et al., 2023; Zhang & Gienger, 2024) trains an independent flow-matching policy for each agent by minimizing the standard vector field loss.
- **LatentFM^{ELBO}** learns latent-conditioned flow-matching policies for individual agents using the ELBO objective defined in Eq. 4.
- **LatentFM$^{\text{ELBO+MI}}$** incorporates an MI objective into LatentFM^{ELBO} to encourage the latent variable z to represent distinct joint behavior modes.
- **MAC-Flow** (Lee et al., 2026) adapts the MAC-Flow architecture, originally designed for offline MARL, to offline MAIL setting by removing its RL components. It first trains a joint flow policy via supervised learning and then distills it into decentralized policies by minimizing the MSE between the i-th component of the joint vector field and the vector field of agent i.
- **LatentMF^{ELBO}** extends MAC-Flow with a latent-conditioned joint flow policy. The joint policy is trained using the ELBO objective and then distilled into latent-conditioned decentralized policies.
- **LatentMF$^{\text{ELBO+MI}}$** integrates an MI term into LatentMF^{ELBO} to encourage the latent variable to distinguish different joint behavior modes.

The detailed implementations are provided in Appendix D.

Table 1: Performance and distribution matching metrics across MAMuJoCo environments. Perf. (↑) denotes the normalized return, while MMD (↓) measures the discrepancy between generated and expert distributions. For each method, the first row shows the mean and the second row shows the std. across 4 random seeds, evaluated over 80 episodes per seed. Best results are shown in **bold**, and second-best results are underlined.

*HC: HalfCheetah, AG: AntGoal, MSS: ManySegmentSwimmer.

Method	HC (2x3)		HC (6x1)		Ant (2x4)		Ant (4x2)		AG (2x4)		AG (4x2)		MSS (5x4)		MSS (10x2)	
	Perf.	MMD	Perf.	MMD	Perf.	MMD	Perf.	MMD	Perf.	MMD	Perf.	MMD	Perf.	MMD	Perf.	MMD
$\text{Gaussian}^{\text{BC}}$	0.995	0.164	0.354	0.383	0.166	0.163	0.184	0.152	**0.761**	0.269	0.701	0.232	0.435	0.237	0.205	0.315
	(0.041)	(0.015)	(0.093)	(0.102)	(0.018)	(0.043)	(0.020)	(0.043)	(0.031)	(0.015)	(0.057)	(0.031)	(0.016)	(0.050)	(0.042)	(0.022)
GMM^{ELBO}	0.926	0.062	0.367	0.476	0.211	0.134	0.111	0.210	0.474	0.280	0.433	0.282	0.400	0.288	0.201	0.324
	(0.018)	(0.021)	(0.077)	(0.080)	(0.039)	(0.029)	(0.024)	(0.048)	(0.020)	(0.080)	(0.083)	(0.043)	(0.062)	(0.034)	(0.011)	(0.016)
$\text{GMM}^{\text{ELBO+MI}}$	0.943	0.068	0.364	0.434	0.158	0.151	0.084	0.212	0.465	0.302	0.457	0.277	0.370	0.338	0.289	0.309
	(0.028)	(0.007)	(0.065)	(0.084)	(0.036)	(0.019)	(0.011)	(0.043)	(0.127)	(0.058)	(0.075)	(0.059)	(0.055)	(0.044)	(0.028)	(0.019)
FM^{BC}	1.059	0.045	0.929	0.142	0.318	0.118	0.471	0.140	0.536	0.259	0.560	0.244	**0.670**	0.211	0.457	0.371
	(0.013)	(0.011)	(0.043)	(0.019)	(0.028)	(0.006)	(0.079)	(0.020)	(0.070)	(0.026)	(0.059)	(0.043)	(0.053)	(0.051)	(0.029)	(0.026)
$\text{LatentFM}^{\text{ELBO}}$	0.993	0.110	0.798	0.216	0.520	0.213	0.513	0.166	0.603	0.222	0.661	0.276	0.491	0.185	0.355	0.321
	(0.016)	(0.025)	(0.025)	(0.050)	(0.089)	(0.011)	(0.075)	(0.023)	(0.119)	(0.041)	(0.170)	(0.039)	(0.024)	(0.047)	(0.039)	(0.036)
$\text{LatentFM}^{\text{ELBO+MI}}$	1.020	0.085	0.790	0.155	0.558	0.170	0.400	0.157	0.629	0.260	0.704	0.221	0.508	0.166	0.395	0.316
	(0.021)	(0.042)	(0.046)	(0.052)	(0.062)	(0.023)	(0.049)	(0.058)	(0.109)	(0.056)	(0.050)	(0.037)	(0.077)	(0.026)	(0.067)	(0.017)
MAC-Flow	1.033	0.101	0.882	0.181	0.225	0.288	0.189	0.310	0.060	0.443	0.055	0.341	0.623	0.301	0.405	0.378
	(0.064)	(0.027)	(0.038)	(0.039)	(0.074)	(0.062)	(0.114)	(0.063)	(0.011)	(0.037)	(0.020)	(0.042)	(0.141)	(0.025)	(0.075)	(0.048)
$\text{LatentMF}^{\text{ELBO}}$	1.004	0.122	0.894	0.168	0.306	0.275	0.371	0.288	0.419	0.340	0.420	0.298	0.565	0.232	0.393	0.326
	(0.020)	(0.035)	(0.025)	(0.060)	(0.072)	(0.019)	(0.176)	(0.089)	(0.059)	(0.049)	(0.104)	(0.038)	(0.100)	(0.068)	(0.061)	(0.015)
$\text{LatetnMF}^{\text{ELBO+MI}}$	0.987	0.133	0.925	0.118	0.453	0.200	0.246	0.309	0.499	0.266	0.355	0.288	0.502	0.176	0.375	0.351
	(0.028)	(0.046)	(0.012)	(0.034)	(0.064)	(0.031)	(0.105)	(0.106)	(0.149)	(0.013)	(0.117)	(0.100)	(0.032)	(0.039)	(0.092)	(0.046)
Ours	**1.066**	**0.028**	**0.957**	**0.100**	**0.656**	**0.091**	**0.555**	**0.112**	0.721	**0.163**	**0.710**	**0.172**	0.535	**0.110**	**0.495**	**0.144**
	(0.006)	(0.005)	(0.036)	(0.041)	(0.029)	(0.013)	(0.038)	(0.016)	(0.085)	(0.030)	(0.059)	(0.042)	(0.106)	(0.035)	(0.049)	(0.016)



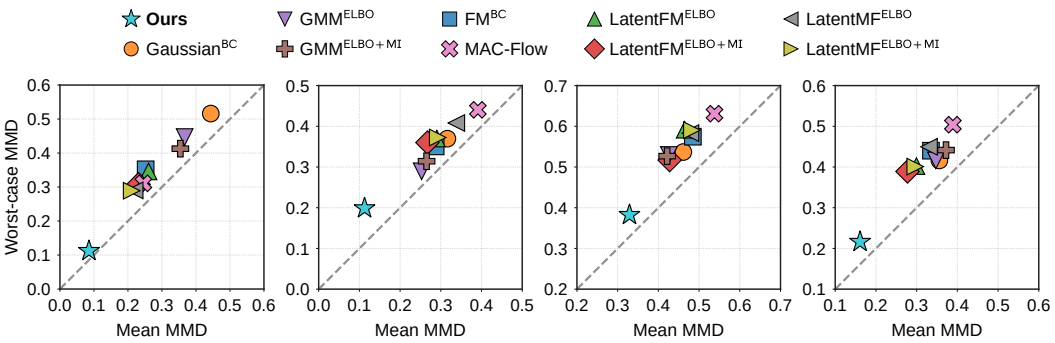
(a) AntGoal (2×4) (b) MSS (5×4)

Figure 2: Learned behaviors in AntGoal and ManySegmentSwimmer. Initial positions are randomly initialized around (0,0). Additional results for learned behaviors are provided in Appendix E.

Experimental results To evaluate performance against the baselines, we report the normalized return. We compute the return for each joint behavior mode, take the highest return across all modes, and normalize it relative to the corresponding expert return, following (Choi et al., 2025). This metric measures whether the learned multi-agent policies can successfully recover at least one expert joint behavior mode, and remains high even if only a single mode is learned. To evaluate how well the policies cover all behavior modes in the demonstration, we additionally report the Maximum Mean Discrepancy (MMD) between policy-generated samples and the demonstration dataset. A lower MMD indicates that the learned policies more closely recover the overall distribution of diverse expert behaviors. Table 1 summarizes the results.

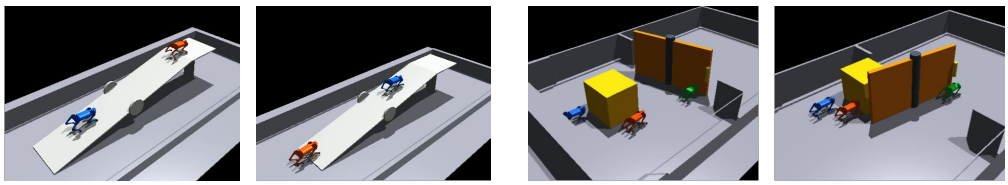
Overall, our algorithm obtains higher normalized returns and lower MMD scores across most tasks, demonstrating its ability to recover diverse expert behaviors more effectively than other baseline methods. Even in AG (2x4) and MSS (5x4) tasks where our approach does not yield the highest performance score, it still attains the lowest MMD score. This is because the baselines tend to only recover a limited subset of the joint behavior modes, whereas our method captures the full range of diverse behaviors present in the demonstrations, as illustrated in Figure 2.

Furthermore, we observe that increasing the number of agents within the same task (e.g., changing the HalfCheetah configuration from 2x3 to 6x1) leads to a performance drop across most algorithms.



(a) HalfCheetah (6×1) (b) Ant (2×4) (c) AntGoal (4×2) (d) MSS (5×4)

Figure 3: Average and worst-case MMD across MAMuJoCo environments (↓). Lower values indicate better distribution matching.



(a) (b) (c) (d)

Figure 4: Illustration of MQE. (a) Seesaw: orange robot as reacher. (b) Seesaw: blue robot as reacher. (c) Cargo Door: one agent opening the door. (d) Cargo Door: remaining agents moving cargo.

This decline occurs because the number of possible combinations of joint behavior mode grow exponentially (M^N), which exacerbates coordination challenges. Despite this difficulty, our method exhibits a smaller drop in performance. For example, in HalfCheetah (HC), our method incurs a minimal drop of 0.110 ($1.066 \rightarrow 0.956$), compared to a drop of 0.151 ($1.033 \rightarrow 0.882$) for MAC-Flow and a severe decline of 0.641 ($0.995 \rightarrow 0.354$) for Gaussian^{BC}.

To further analyze the behaviors learned across the M distinct demonstration modes, we evaluate two additional metrics. First, we compute the average minimum MMD: for each joint expert mode, we identify the latent variable that generates the closest sample distribution, and average these minimal MMD across all modes. This reflects the model's overall capacity to accurately recover the full diversity of expert strategies. Second, we compute the maximum minimum MMD values across all modes (i.e., the worst-case mode MMD), which explicitly evaluates the model's robustness against mode dropping. The results are summarized in Figure 3. In these plots, the bottom-left region represents accurate mode recovery. Furthermore, when a method's marker aligns closely with the $y = x$ diagonal, its worst-case MMD is comparable to its average MMD, indicating that it evenly recovers all modes without mode dropping. As shown in Figure 3, our method consistently occupies the bottom-left position, demonstrating accurate and balanced mode recovery.

3.2 MQE

Environmental setup MQE is a suite of multi-quadruped robotic environments that provides a variety of cooperative tasks. Based on this suite, we conduct experiments on four environments: **(1) Door:** Two quadruped agents must pass through a revolving door. The agents cannot pass through the door simultaneously, and therefore must coordinate the order in which they cross. We collect demonstrations corresponding to two distinct strategy modes, in which the two agents cross the door in different orders. **(2) Seesaw:** One of the two agents must reach a target position located beyond a seesaw, as shown in Figure 4. Since the target is located at the far end of the seesaw, one agent must hold the opposite end to stabilize it while the other agent climbs the seesaw and reaches the target. We collect demonstrations corresponding to two distinct coordination modes where the two agents take opposite roles of holder and reacher in each mode. **(3) Cargo Door:** Three agents must cooperate to transport cargo through a revolving door. One agent must open and hold the revolving door, while the other two agents transport the cargo through it, as shown in Figure 4. We collect

Table 2: Success rate and distribution matching metrics across MQE environments. Succ. (↑) denotes the success rate, while MMD (↓) measures the discrepancy between generated and expert distributions. The suffix -P indicates partial observable tasks. For each method, the first row shows the mean and the second row shows the std. across 4 random seeds, evaluated over 80 episodes per seed. Best results are shown in **bold**, and second-best results are underlined.

Method	Door		Door-P		Seesaw		Seesaw-P		Cargo Door		Cargo Door-P		Cargo		Cargo-P	
	Succ.	MMD	Succ.	MMD	Succ.	MMD	Succ.	MMD	Succ.	MMD	Succ.	MMD	Succ.	MMD	Succ.	MMD
Gaussian^{BC}	0.688	0.096	0.100	0.196	0.312	**0.076**	0.150	0.155	0.312	0.219	0.100	0.196	0.463	0.174	0.388	**0.114**
	(0.065)	(0.023)	(0.094)	(0.013)	(0.041)	(0.016)	(0.061)	(0.039)	(0.041)	(0.048)	(0.094)	(0.013)	(0.074)	(0.031)	(0.160)	(0.022)
GMM^{ELBO}	0.625	0.160	0.081	0.325	0.075	0.339	0.019	0.463	0.081	0.303	0.081	0.325	0.050	0.372	0.025	0.243
	(0.136)	(0.020)	(0.078)	(0.020)	(0.018)	(0.043)	(0.011)	(0.072)	(0.054)	(0.025)	(0.078)	(0.020)	(0.000)	(0.037)	(0.025)	(0.041)
GMM$^{ELBO+MI}$	0.469	0.158	0.056	0.344	0.069	0.396	0.100	0.295	0.094	0.276	0.056	0.344	0.025	0.365	0.025	0.271
	(0.096)	(0.047)	(0.041)	(0.044)	(0.032)	(0.071)	(0.077)	(0.158)	(0.051)	(0.027)	(0.041)	(0.044)	(0.043)	(0.045)	(0.043)	(0.025)
FM^{BC}	0.800	0.067	0.100	0.226	**0.362**	0.087	0.200	0.166	0.150	0.218	0.100	0.226	**0.500**	**0.155**	0.312	0.118
	(0.000)	(0.012)	(0.035)	(0.057)	(0.074)	(0.016)	(0.071)	(0.015)	(0.050)	(0.020)	(0.035)	(0.057)	(0.184)	(0.040)	(0.156)	(0.008)
LatentFM^{ELBO}	0.300	0.284	0.144	0.347	0.062	0.332	0.069	0.319	0.131	0.397	0.144	0.347	0.150	0.358	0.050	0.307
	(0.113)	(0.115)	(0.099)	(0.059)	(0.028)	(0.106)	(0.027)	(0.068)	(0.157)	(0.114)	(0.099)	(0.059)	(0.050)	(0.017)	(0.035)	(0.045)
LatentFM$^{ELBO+MI}$	0.325	0.235	0.244	0.298	0.069	0.366	0.062	0.299	0.106	0.381	0.244	0.298	0.212	0.293	0.100	0.259
	(0.103)	(0.117)	(0.132)	(0.097)	(0.037)	(0.047)	(0.012)	(0.116)	(0.091)	(0.095)	(0.132)	(0.097)	(0.082)	(0.031)	(0.087)	(0.048)
MAC-Flow	0.812	0.080	0.138	0.249	0.163	0.145	0.062	0.221	0.100	0.231	0.138	0.249	0.188	0.174	0.100	0.143
	(0.054)	(0.021)	(0.065)	(0.015)	(0.054)	(0.019)	(0.022)	(0.027)	(0.146)	(0.025)	(0.065)	(0.015)	(0.054)	(0.032)	(0.094)	(0.013)
LatentMF^{ELBO}	0.512	0.152	0.144	0.313	0.006	0.281	0.006	0.278	0.212	0.332	0.144	0.313	0.000	0.253	0.000	0.202
	(0.133)	(0.048)	(0.089)	(0.038)	(0.011)	(0.036)	(0.011)	(0.094)	(0.195)	(0.078)	(0.089)	(0.038)	(0.000)	(0.021)	(0.000)	(0.029)
LatentMF$^{ELBO+MI}$	0.456	0.173	0.031	0.293	0.006	0.284	0.037	0.314	0.131	0.321	0.031	0.293	0.013	0.266	0.000	0.174
	(0.160)	(0.069)	(0.032)	(0.009)	(0.011)	(0.043)	(0.052)	(0.103)	(0.076)	(0.047)	(0.032)	(0.009)	(0.022)	(0.041)	(0.000)	(0.025)
Ours	**0.819**	**0.066**	**0.456**	**0.129**	0.356	0.090	**0.388**	**0.076**	**0.425**	**0.143**	**0.456**	**0.129**	0.487	0.210	**0.413**	0.146
	(0.021)	(0.011)	(0.111)	(0.016)	(0.072)	(0.027)	(0.041)	(0.023)	(0.165)	(0.056)	(0.111)	(0.016)	(0.156)	(0.039)	(0.108)	(0.020)

demonstrations corresponding to two distinct behavior modes, where a different agent takes the role of the door holder in each mode. **(4) Cargo:** Four agents must cooperate to transport cargo toward one of four corners. The agents must therefore coordinate both the target direction and their actions to move the cargo successfully. We collect demonstrations corresponding to four distinct behavior modes, with each transporting the cargo toward a different corner.

By default, MQE provides each agent with the positions of all other agents. To better reflect realistic scenarios, we additionally consider a partially observable setting by masking the positions of other agents when they are outside an agent's field of view. We denote this setting by appending `-P` to the environment name. The detailed environmental setups are provided in Appendix C.3.

Experimental results Table 2 summarizes the results. In these environments, our method achieves higher success rates and lower MMD scores across most tasks. We observe that introducing partial observability causes a substantial performance degradation across most methods. This is because under full observability, agents can readily determine the appropriate behavior mode from others' states, whereas incomplete information leads to an increased prevalence of multi-modal action choices. For instance, near the initial state, agents must choose among multiple valid behavior modes without access to global observations, making joint coordination significantly harder. While our method outperforms the baselines in these partially observable settings, it still experiences a performance drop despite promoting coordination via the latent variable z. This decline stems primarily from unobserved inter-agent dependencies. In particular, the conditional mutual information term $I(a^i; o^{-i} \mid o^i, z)$ in Eq. 6 highlights how an agent's action depends on the unobserved local observations of others (o^{-i}). This dependency cannot be fully resolved by the latent variable z alone due to the inherent inaccessibility of other agents' observations during execution.

4 Conclusion

We present CoFac-FM, a novel framework for offline MAIL from multi-modal demonstrations. To address the challenge of uncoordinated decentralized execution, we introduce a shared latent variable designed to capture complex inter-agent dependencies, rendering agents' actions conditionally independent. Additionally, we derive a practical, density-free flow-matching objective for latent representation learning. Extensive evaluations demonstrate the superiority of our framework across a variety of robotic control tasks.

AI USE STATEMENT

During the preparation of this work, we used generative AI tools to enhance the clarity and readability of the manuscript and to assist with code review and verification. All AI-assisted suggestions and code feedback were thoroughly reviewed, verified, and edited by us. We take full responsibility for the final content and integrity of this work.

REFERENCES

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

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, UK, July 1995]*, pp. 103–129. Oxford University Press, 1995.

The Viet Bui, Tien Anh Mai, and Thanh Hong Nguyen. Inverse factorized soft q-learning for cooperative multi-agent imitation learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=xrbgXJomJp.

The Viet Bui, Tien Anh Mai, and Thanh Hong Nguyen. MisoDICE: Multi-agent imitation from mixed-quality demonstrations. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=krG6UHAvYr.

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. *Int. J. Robotics Res.*, 2025.

Yunseon Choi, Minchan Jeong, Soobin Um, and Kee-Eung Kim. DPAIL: Training diffusion policy for adversarial imitation learning without policy optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=F2ILoE1eDj.

Dayi Dong, Maulik Bhatt, Seoyeon Choi, and Negar Mehr. MIMIC-D: multi-modal imitation for multi-agent coordination with decentralized diffusion policies. *CoRR*, 2025.

Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, 2002.

Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav S. Sukhatme, and Joseph J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 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*, Proceedings of Machine Learning Research. PMLR, 2022.

Hoang M. Le, Yisong Yue, Peter Carr, and Patrick Lucey. Coordinated multi-agent imitation learning. In *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2017.

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

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, 2017.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 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 *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXVII*. Springer, 2024.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=fgFBtYgJQX_.

Bei Peng, Tabish Rashid, Christian Schröder de Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Boehmer, and Shimon Whiteson. FACMAC: factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.

Quentin Rouxel, Andrea Ferrari, Serena Ivaldi, and Jean-Baptiste Mouret. Flow matching imitation learning for multi-support manipulation. In *23rd IEEE-RAS International Conference on Humanoid Robots, Humanoids 2024, Nancy, France, November 22-24, 2024*. IEEE, 2024.

Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, 2018.

Jianrui Wang, Yitian Hong, Jiali Wang, Jiapeng Xu, Yang Tang, Qing-Long Han, and Jürgen Kurths. Cooperative and competitive multi-agent systems: From optimization to games. *IEEE CAA J. Autom. Sinica*, 2022.

Michael Satosi Watanabe. Information theoretical analysis of multivariate correlation. *IBM J. Res. Dev.*, 1960.

Ziyan Xiong, Bo Chen, Shiyu Huang, Wei-Wei Tu, Zhaofeng He, and Yang Gao. MQE: unleashing the power of interaction with multi-agent quadruped environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2024, Abu Dhabi, United Arab Emirates, October 14-18, 2024*. IEEE, 2024.

Lantao Yu, Jiaming Song, and Stefano Ermon. Multi-agent adversarial inverse reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, Proceedings of Machine Learning Research. PMLR, 2019.

Eric Zhan, Stephan Zheng, Yisong Yue, Long Sha, and Patrick Lucey. Generative multi-agent behavioral cloning. *CoRR*, abs/1803.07612, 2018. URL http://arxiv.org/abs/1803.07612.

Fan Zhang and Michael Gienger. Affordance-based robot manipulation with flow matching. *CoRR*, 2024. URL https://doi.org/10.48550/arXiv.2409.01083.

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 37: Annual Conference on Neural Information Processing Systems*, 2024.

A Related works

Multi-agent imitation learning MAIL seeks to train decentralized policies from pre-collected joint demonstrations. Under the assumption of individual-global consistency, standard approaches (Bui et al., 2024; 2025) model the joint policy directly as a factorization of independent individual policies. Alternative frameworks (Le et al., 2017) introduce shared latent variables, conditioning individual policies to capture implicit coordination. While online adversarial methods (Song et al., 2018; Yu et al., 2019) extend MAIL to Markov games via occupancy matching, they require continuous environment interaction. In contrast, we focus on offline MAIL, learning expressive factorized policies strictly from a fixed dataset.

Multi-modal imitation and latent behavior representations Learning from multi-modal expert behavior has been extensively studied in single-agent imitation learning. To disentangle behavior modes, early works (Li et al., 2017; Hausman et al., 2017) incorporated mutual information regularization to recover interpretable latent factors underlying diverse demonstrations. More recently, behavioral cloning methods have adopted expressive generative models, such as diffusion policies (Janner et al., 2022; Chi et al., 2025; Choi et al., 2025) and flow-matching policies (Zhang & Gienger, 2024), to capture high-dimensional multi-modal trajectories without mode-averaging. While these works demonstrate the value of expressive generative models, multi-modality introduces an additional difficulty in multi-agent systems. Correctly modeling each agent's marginal distribution does not ensure that independently sampled actions correspond to a coherent joint strategy. Similarly, maximizing mutual information promotes behavior distinguishability, but fails to force the latent variable to eliminate the residual dependencies that impede decentralized execution. In contrast, our formulation redefines the operational role of z: it is learned explicitly to render the joint distribution conditionally factorizable. The resulting factorization gap directly optimizes the representation for decentralized execution, rather than merely expressing or separating behavioral modes.

Generative models for multi-agent coordination Diffusion (Ho et al., 2020) and flow-based models (Lipman et al., 2023; Liu et al., 2023) have recently been applied to multi-agent settings to model complex joint and individual action distributions. For instance, MADiff (Zhu et al., 2024) trains individual diffusion policies guided by a centralized critic, whereas MIMIC-D (Dong et al., 2025) and MAC-Flow (Lee et al., 2026) train centralized joint generative policies and distill them into decentralized policies. Although our method also leverages a centralized joint network during training, it departs from prior work by framing multi-agent coordination as a representation learning problem. Rather than distilling a joint policy into decentralized policies, we explicitly define the operational role of a coordination representation: conditioning on z must render the expert joint behavior factorizable into independent policies. In our framework, the centralized flow policy serves as a reference distribution to evaluate conditional factorizability, rather than merely a target for distillation. The resulting joint-to-local vector field discrepancy yields a unified learning signal that optimizes both the decentralized policies and the latent representation.

B Proofs

B.1 Proof of Lemma 2.1 and Theorem 2.2

Proof of Lemma 2.1 (Vector–Score Relation). By definition, the population-optimal flow-matching vector field $v_{\star,\lambda}(x)$ that minimizes Eq. 2 is given by the conditional expectation of the conditional velocity $u_\lambda(x_0, x_1)$ given $x_\lambda = x$:

$$v_{\star,\lambda}(x) = \mathbb{E}_{x_0,x_1}\left[u_\lambda(x_0, x_1) \mid x_\lambda = x\right].$$

For the affine Gaussian interpolation path $x_\lambda = f_\lambda(x_0, x_1) = \alpha_\lambda x_1 + \sigma_\lambda x_0$ with $x_0 \sim \mathcal{N}(0, I)$, the conditional velocity is:

$$u_\lambda(x_0, x_1) = \frac{\partial}{\partial\lambda} f_\lambda(x_0, x_1) = \dot{\alpha}_\lambda x_1 + \dot{\sigma}_\lambda x_0.$$

Using $x_0 = \frac{x_\lambda - \alpha_\lambda x_1}{\sigma_\lambda}$, we can express $u_\lambda(x_0, x_1)$ in terms of x_λ and x_1:

$$u_\lambda(x_0, x_1) = \dot{\alpha}_\lambda x_1 + \dot{\sigma}_\lambda \left(\frac{x_\lambda - \alpha_\lambda x_1}{\sigma_\lambda}\right) = \frac{\dot{\sigma}_\lambda}{\sigma_\lambda} x_\lambda + \left(\dot{\alpha}_\lambda - \frac{\dot{\sigma}_\lambda \alpha_\lambda}{\sigma_\lambda}\right) x_1. \tag{9}$$

Taking the conditional expectation of Eq. 9 given $x_\lambda = x$ yields:

$$v_{\star,\lambda}(x) = \frac{\dot{\sigma}_\lambda}{\sigma_\lambda} x + \left(\dot{\alpha}_\lambda - \frac{\dot{\sigma}_\lambda \alpha_\lambda}{\sigma_\lambda} \right) \mathbb{E}[x_1 \mid x_\lambda = x]. \tag{10}$$

Next, we establish the relation between $\mathbb{E}[x_1 \mid x_\lambda = x]$ and the score function $\nabla_x \log p_\lambda(x)$. Given $x_0 \sim \mathcal{N}(0, I)$, the conditional distribution of x_λ given x_1 is Gaussian, $p(x_\lambda \mid x_1) = \mathcal{N}(x_\lambda; \alpha_\lambda x_1, \sigma_\lambda^2 I)$. The score of this conditional distribution is:

$$\nabla_{x_\lambda} \log p(x_\lambda \,|\, x_1) = -\frac{x_\lambda - \alpha_\lambda x_1}{\sigma_\lambda^2}.$$

By Bayes' rule, the marginal score function $s_\lambda(x) = \nabla_x \log p_\lambda(x)$ satisfies:

$$\begin{aligned} \nabla_x \log p_\lambda(x) &= \mathbb{E}_{x_1 | x_\lambda = x} \left[\nabla_x \log p(x \mid x_1) \right] \\ &= \mathbb{E}_{x_1 | x_\lambda = x} \left[-\frac{x - \alpha_\lambda x_1}{\sigma_\lambda^2} \right] \\ &= -\frac{x}{\sigma_\lambda^2} + \frac{\alpha_\lambda}{\sigma_\lambda^2} \mathbb{E}[x_1 \mid x_\lambda = x]. \end{aligned}$$

Rearranging this equation gives the posterior mean of x_1:

$$\mathbb{E}[x_1 \mid x_\lambda = x] = \frac{x}{\alpha_\lambda} + \frac{\sigma_\lambda^2}{\alpha_\lambda} \nabla_x \log p_\lambda(x). \tag{11}$$

Substituting Eq. 11 into Eq. 10:

$$\begin{aligned} v_{\star,\lambda}(x) &= \frac{\dot{\sigma}_\lambda}{\sigma_\lambda} x + \left(\dot{\alpha}_\lambda - \frac{\dot{\sigma}_\lambda \alpha_\lambda}{\sigma_\lambda} \right) \left(\frac{x}{\alpha_\lambda} + \frac{\sigma_\lambda^2}{\alpha_\lambda} \nabla_x \log p_\lambda(x) \right) \\ &= \left(\frac{\dot{\sigma}_\lambda}{\sigma_\lambda} + \frac{\dot{\alpha}_\lambda}{\alpha_\lambda} - \frac{\dot{\sigma}_\lambda}{\sigma_\lambda} \right) x + \left(\frac{\dot{\alpha}_\lambda \sigma_\lambda^2}{\alpha_\lambda} - \dot{\sigma}_\lambda \sigma_\lambda \right) \nabla_x \log p_\lambda(x) \\ &= \frac{\dot{\alpha}_\lambda}{\alpha_\lambda} x + \kappa_\lambda \nabla_x \log p_\lambda(x), \end{aligned}$$

where $\kappa_\lambda := \frac{\dot{\alpha}_\lambda \sigma_\lambda^2}{\alpha_\lambda} - \dot{\sigma}_\lambda \sigma_\lambda$. □

Proof of Theorem 2.2. (Flow factorization equivalence) By definition, the independent policy π^I is fully factorized as $\pi^I(a) = \prod_{i=1}^N \pi^i(a^i)$. Furthermore, constructing the intermediate a_λ^i for each agent i operates independently:

$$a_\lambda^i = \alpha_\lambda a^i + \sigma_\lambda \epsilon^i, \quad \epsilon^i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I).$$

Because all individual policies π^i share the same Gaussian interpolation schedule $(\alpha_\lambda, \sigma_\lambda)$, the forward process of π^I naturally mirrors this affine transformation structure via concatenation:

$$a_\lambda = \begin{bmatrix} a_\lambda^1 \\ \vdots \\ a_\lambda^N \end{bmatrix} = \alpha_\lambda \begin{bmatrix} a^1 \\ \vdots \\ a^N \end{bmatrix} + \sigma_\lambda \begin{bmatrix} \epsilon^1 \\ \vdots \\ \epsilon^N \end{bmatrix} = \alpha_\lambda a + \sigma_\lambda \epsilon.$$

Since the individual forward processes operate block-wise and originate from a factorized Gaussian normal distribution at $\lambda = 0$, the intermediate density of the independent flow preserves this factorization across all time steps $\lambda \in (0, 1)$:

$$\pi_\lambda^I(a_\lambda) = \prod_{i=1}^N \pi_\lambda^i(a_\lambda^i).$$

According to Lemma 2.1, the population-optimal vector field takes the form $v_{\star,\lambda}(a) = \frac{\dot{\alpha}_\lambda}{\alpha_\lambda} a + \kappa_\lambda \nabla_a \log \pi_\lambda(a)$. Because the intermediate density $\pi_\lambda^I(a_\lambda)$ is factorized, its log-density decomposes

into a sum of individual log-densities: $\log \pi^I_\lambda(a_\lambda) = \sum_{i=1}^N \log \pi^i_\lambda(a^i_\lambda)$. Consequently, its score function decomposes over the action dimensions:

$$\nabla_{a_\lambda} \log \pi^I_\lambda(a_\lambda) = \left[\nabla_{a^1_\lambda} \log \pi^1_\lambda(a^1_\lambda), \ldots, \nabla_{a^N_\lambda} \log \pi^N_\lambda(a^N_\lambda)\right]^\top.$$

Therefore, the population-optimal vector field for the factorized policy, $v^I_{\star,\lambda}(a)$, is the concatenation of the individual agents' vector fields:

$$v^I_{\star,\lambda}(a) = \left[v^1_{\star,\lambda}(a^1), \ldots, v^N_{\star,\lambda}(a^N)\right]^\top.$$

Now, consider the evolution of the KL divergence $D_{\mathrm{KL}}(\pi^J_\lambda \parallel \pi^I_\lambda)$ along the two probability paths. Although the joint and independent distributions are constructed using the same interpolation schedule $(\alpha_\lambda, \sigma_\lambda)$, they generally evolve along different probability paths with different population-optimal vector fields. Let $v^J_{\star,\lambda}$ and $v^I_{\star,\lambda}$ denote their respective vector fields. Their densities satisfy the continuity equations (Lipman et al., 2023)

$$\partial_\lambda \pi^J_\lambda(a) = -\nabla_a \cdot \left(\pi^J_\lambda(a) v^J_{\star,\lambda}(a)\right), \quad \partial_\lambda \pi^I_\lambda(a) = -\nabla_a \cdot \left(\pi^I_\lambda(a) v^I_{\star,\lambda}(a)\right).$$

Differentiating the KL divergence gives

$$\begin{aligned}\frac{d}{d\lambda} D_{\mathrm{KL}}(\pi^J_\lambda \parallel \pi^I_\lambda) &= \frac{d}{d\lambda} \int \pi^J_\lambda(a) \log \frac{\pi^J_\lambda(a)}{\pi^I_\lambda(a)}\, da \\ &= \int \partial_\lambda \pi^J_\lambda(a) \log \frac{\pi^J_\lambda(a)}{\pi^I_\lambda(a)}\, da - \int \pi^J_\lambda(a) \frac{\partial_\lambda \pi^I_\lambda(a)}{\pi^I_\lambda(a)}\, da,\end{aligned}$$

where we used $\int \partial_\lambda \pi^J_\lambda(a)\, da = 0$.

Substituting the continuity equations, the first term becomes

$$\begin{aligned}&-\int \nabla_a \cdot \left(\pi^J_\lambda(a) v^J_{\star,\lambda}(a)\right) \log \frac{\pi^J_\lambda(a)}{\pi^I_\lambda(a)}, da \\ &\quad = \mathbb{E}_{a\sim\pi^J_\lambda}\left[v^J_{\star,\lambda}(a)^\top \left(\nabla_a \log \pi^J_\lambda(a) - \nabla_a \log \pi^I_\lambda(a)\right)\right],\end{aligned}$$

where integration by parts is used and the boundary term is assumed to vanish.

For the second term,

$$\begin{aligned}-\int \pi^J_\lambda(a) \frac{\partial_\lambda \pi^I_\lambda(a)}{\pi^I_\lambda(a)}\, da &= \int \pi^J_\lambda(a) \frac{\nabla_a \cdot \left(\pi^I_\lambda(a) v^I_{\star,\lambda}(a)\right)}{\pi^I_\lambda(a)} da \\ &= \mathbb{E}_{a\sim\pi^J_\lambda}\left[\nabla_a \cdot v^I_{\star,\lambda}(a) + v^I_{\star,\lambda}(a)^\top \nabla_a \log \pi^I_\lambda(a)\right].\end{aligned}$$

Using integration by parts again,

$$\mathbb{E}_{a\sim\pi^J_\lambda}\left[\nabla_a \cdot v^I_{\star,\lambda}(a)\right] = -\mathbb{E}_{a\sim\pi^J_\lambda}\left[v^I_{\star,\lambda}(a)^\top \nabla_a \log \pi^J_\lambda(a)\right].$$

Therefore,

$$\frac{d}{d\lambda} D_{\mathrm{KL}}(\pi^J_\lambda \parallel \pi^I_\lambda) = \mathbb{E}_{a\sim\pi^J_\lambda}\left[\left(v^J_{\star,\lambda}(a) - v^I_{\star,\lambda}(a)\right)^\top \left(\nabla_a \log \pi^J_\lambda(a) - \nabla_a \log \pi^I_\lambda(a)\right)\right]. \tag{12}$$

According to Lemma 2.1, both probability paths use the same interpolation schedule and hence share the same coefficients:

$$v^J_{\star,\lambda}(a) = \frac{\dot\alpha_\lambda}{\alpha_\lambda} a + \kappa_\lambda \nabla_a \log \pi^J_\lambda(a), \quad v^I_{\star,\lambda}(a) = \frac{\dot\alpha_\lambda}{\alpha_\lambda} a + \kappa_\lambda \nabla_a \log \pi^I_\lambda(a).$$

Consequently, their difference satisfies

$$v^J_{\star,\lambda}(a) - v^I_{\star,\lambda}(a) = \kappa_\lambda \left(\nabla_a \log \pi^J_\lambda(a) - \nabla_a \log \pi^I_\lambda(a)\right) \tag{13}$$

Substituting Eq. 13 into Eq. 12 yields

$$\frac{d}{d\lambda} D_{\mathrm{KL}}\left(\pi_\lambda^J \parallel \pi_\lambda^I\right) = \frac{1}{\kappa_\lambda} \mathbb{E}_{a\sim\pi_\lambda^J}\left[\left\| v_{\star,\lambda}^J(a) - v_{\star,\lambda}^I(a)\right\|^2\right].$$

Since $v_{\star,\lambda}^I(a) = \left[v_{\star,\lambda}^1(a^1), \ldots, v_{\star,\lambda}^N(a^N)\right]^\top$,

$$\frac{d}{d\lambda} D_{\mathrm{KL}}\left(\pi_\lambda^J \parallel \pi_\lambda^I\right) = \frac{1}{\kappa_\lambda} \mathbb{E}_{a\sim\pi_\lambda^J}\left[\sum_{i=1}^N \left\| v_{\star,\lambda}^{J,i}(a) - v_{\star,\lambda}^i(a)\right\|^2\right]. \tag{14}$$

Integrating Eq. 14 from $\lambda = 0$ to $\lambda = 1$ therefore gives

$$D_{\mathrm{KL}}\left(\pi^J \parallel \pi^I\right) = \int_0^1 \frac{1}{\kappa_\lambda} \mathbb{E}_{a\sim\pi_\lambda^J}\left[\sum_{i=1}^N \left\| v_{\star,\lambda}^{J,i}(a) - v_{\star,\lambda}^i(a)\right\|^2\right] d\lambda$$

under the usual regularity conditions, with endpoint values interpreted by continuity when κ_λ vanishes or becomes singular at the boundary.

□

B.2 Derivation of the evidence lower bound (ELBO)

The primary objective is to maximize the marginal log-likelihood of the demonstration dataset $\mathcal{D}$. For a given trajectory $\tau = \{o^{(t)}, a^{(t)}\}_{t=0}^H$, the marginal log-likelihood under the shared discrete latent variable $z \sim p(z)$ is defined as:

$$\log p(\tau) = \log \sum_{z\in\mathcal{Z}} p(z) \prod_{t=0}^H \prod_{i=1}^N \pi_\theta^i(a^{i,(t)} \,|\, o^{i,(t)}, z).$$

To make the optimization tractable and to infer the latent behavioral mode from the demonstrations, we introduce a variational posterior $q_\phi(z\,|\,\tau)$ conditioned on the joint trajectory τ. Multiplying and dividing the term inside the logarithm by $q_\phi(z\,|\,\tau)$ and applying Jensen's inequality yields:

$$\begin{aligned}
\log p(\tau) &= \log \sum_{z\in\mathcal{Z}} q_\phi(z \mid \tau) \frac{p(z)\prod_{t=0}^H \prod_{i=1}^N \pi_\theta^i(a^{i,(t)} \mid o^{i,(t)}, z)}{q_\phi(z \mid \tau)} \\
&= \log \mathbb{E}_{z\sim q_\phi(z|\tau)}\left[\frac{p(z)\prod_{t=0}^H \prod_{i=1}^N \pi_\theta^i(a^{i,(t)} \mid o^{i,(t)}, z)}{q_\phi(z \mid \tau)}\right] \\
&\geq \mathbb{E}_{z\sim q_\phi(z|\tau)}\left[\log\left(\frac{p(z)\prod_{t=0}^H \prod_{i=1}^N \pi_\theta^i(a^{i,(t)} \mid o^{i,(t)}, z)}{q_\phi(z \mid \tau)}\right)\right] \\
&= \mathbb{E}_{z\sim q_\phi(z|\tau)}\left[\log \prod_{t=0}^H \prod_{i=1}^N \pi_\theta^i(a^{i,(t)} \mid o^{i,(t)}, z) + \log \frac{p(z)}{q_\phi(z \mid \tau)}\right] \\
&= \mathbb{E}_{z\sim q_\phi(z|\tau)}\left[\sum_{t=0}^H \sum_{i=1}^N \log \pi_\theta^i(a^{i,(t)} \mid o^{i,(t)}, z)\right] - \mathbb{E}_{z\sim q_\phi(z|\tau)}\left[\log \frac{q_\phi(z \mid \tau)}{p(z)}\right] \\
&= \mathbb{E}_{z\sim q_\phi(z|\tau)}\left[\sum_{t=0}^H \sum_{i=1}^N \log \pi_\theta^i(a^{i,(t)} \mid o^{i,(t)}, z)\right] - D_{\mathrm{KL}}\left(q_\phi(\cdot \mid \tau) \parallel p(\cdot)\right).
\end{aligned}$$

Substituting flow-matching regression loss as a surrogate each negative log-likelihood term $-\log \pi_\theta^i(a^{i,(t)} \mid o^{i,(t)}, z)$ into the log-likelihood term, yields:

$$\mathbb{E}_{\substack{\tau\sim\mathcal{D},\lambda,\\ z\sim q_\phi(z|\tau)}}\left[\sum_{t=0}^H \sum_{i=1}^N \left\| v_{\theta,\lambda}^i(a_\lambda^{i,(t)}, o^{i,(t)}, z) - u_\lambda(\epsilon^{i,(t)}, a^{i,(t)})\right\|^2\right] + D_{\mathrm{KL}}\left(q_\phi(\cdot \mid \tau) \| p(\cdot)\right).$$

where $\epsilon^{i,(t)} \sim \mathcal{N}(0, I)$ and $\lambda \sim \mathcal{U}(0, 1)$.

B.3 Proof of Eq. 6

For a fixed encoder q_ϕ and centralized joint policy π_η^J, let $\rho_\phi(o, z)$ denote the distribution over (o, z) induced by $\tau \sim \mathcal{D}$ and $z \sim q_\phi(z \mid \tau)$. We define the corresponding joint distribution as

$$p_\phi(a, o, z) = \rho_\phi(o, z)\pi_\eta^J(a \mid o, z),$$

and let $\pi_\eta^{J,i}(a^i \mid o, z) = \int \pi_\eta^J(a \mid o, z)\, da^{-i}$ denote the i-th action marginal of $\pi_\eta^J(a \mid o, z)$ given full observation o and latent z.

By definition, the factorization gap can be written as

$$\begin{aligned}\Delta_{\text{fac}}(\phi, \theta) &= \mathbb{E}_{p_\phi}\left[\log \frac{\pi_\eta^J(a \mid o, z)}{\prod_{i=1}^N \pi_\theta^i(a^i \mid o^i, z)}\right] \\ &= \mathbb{E}_{p_\phi}\left[\log \frac{\pi_\eta^J(a \mid o, z)}{\prod_{i=1}^N \pi_\eta^{J,i}(a^i \mid o, z)}\right] + \sum_{i=1}^N \mathbb{E}_{p_\phi}\left[\log \frac{\pi_\eta^{J,i}(a^i \mid o, z)}{\pi_\theta^i(a^i \mid o^i, z)}\right].\end{aligned}$$

The first term is the conditional total correlation,

$$\text{TC}(a^1; \ldots; a^N \mid o, z) = \mathbb{E}_{p_\phi}\left[\log \frac{\pi_\eta^J(a \mid o, z)}{\prod_{i=1}^N \pi_\eta^{J,i}(a^i \mid o, z)}\right].$$

For each agent i, let $p_\phi(a^i \mid o^i, z)$ denote the local conditional distribution induced by p_ϕ. Introducing this distribution into the second term gives

$$\mathbb{E}_{p_\phi}\left[\log \frac{\pi_\eta^{J,i}(a^i \mid o, z)}{\pi_\theta^i(a^i \mid o^i, z)}\right] = \mathbb{E}_{p_\phi}\left[\log \frac{\pi_\eta^{J,i}(a^i \mid o, z)}{p_\phi(a^i \mid o^i, z)}\right] + \mathbb{E}_{p_\phi}\left[\log \frac{p_\phi(a^i \mid o^i, z)}{\pi_\theta^i(a^i \mid o^i, z)}\right]. \tag{15}$$

Since $o = (o^i, o^{-i})$ and $\pi_\eta^{J,i}(a^i \mid o, z) = p_\phi(a^i \mid o, z)$, the first term in Eq. 15 is

$$I(a^i; o^{-i} \mid o^i, z).$$

The second term can be expressed as

$$\mathbb{E}_{o^i, z}\left[D_{\text{KL}}\left(p_\phi(\cdot \mid o^i, z) \,\middle|\, \pi_\theta^i(\cdot \mid o^i, z)\right)\right].$$

Therefore,

$$\Delta_{\text{fac}}(\phi, \theta) = \text{TC}(a^1; \ldots; a^N \mid o, z) + \sum_{i=1}^N I(a^i; o^{-i} \mid o^i, z) + \sum_{i=1}^N \mathbb{E}_{o^i, z}\left[D_{\text{KL}}\left(p_\phi(\cdot \mid o^i, z) \,\|\, \pi_\theta^i(\cdot \mid o^i, z)\right)\right].$$

Under the assumption that the decentralized policies are sufficiently expressive, there exists θ such that

$$\pi_\theta^i(\cdot \mid o^i, z) = p_\phi(\cdot \mid o^i, z)$$

for every agent i almost surely. Hence, the final KL terms can attain zero, and minimizing over θ yields

$$\min_\theta \Delta_{\text{fac}}(\phi, \theta) \equiv \Delta_{\star,\text{fac}}(\theta) = \text{TC}(a^1; \ldots; a^N \mid o, z) + \sum_{i=1}^N I(a^i; o^{-i} \mid o^i, z),$$

which proves Eq. 6.

C EXPERIMENTAL SETUP

C.1 2D MULTI-AGENT TOY TASK

We consider a multi-agent bandit environment with two agents ($N = 2$). Each agent $i \in \{1, 2\}$ executes a two-dimensional action $a^i = [a^i_{\text{coord}}, a^i_{\text{indep}}]^\top \in \mathbb{R}^2$, consisting of a coordinated component a^i_{coord} and an independent component a^i_{indep}. The demonstration dataset is generated as follows. The joint coordinated action pair $(a^1_{\text{coord}}, a^2_{\text{coord}})$ is sampled from an equally weighted two-component Gaussian mixture distribution:

$$(a^1_{\text{coord}}, a^2_{\text{coord}}) \sim \frac{1}{2}\mathcal{N}\left(\begin{bmatrix}1\\-1\end{bmatrix}, 0.03^2 I\right) + \frac{1}{2}\mathcal{N}\left(\begin{bmatrix}-1\\1\end{bmatrix}, 0.03^2 I\right).$$

Concurrently, each agent's independent action component a^i_{indep} is sampled independently and uniformly from $\{-1, 1\}$ with additive Gaussian noise $\mathcal{N}(0, 0.03^2)$.

C.2 MAMUJOCO

MAMuJoCo provides a diverse set of multi-agent configurations for MuJoCo tasks. Using the `agent_conf` argument, we set the number of agents and the number of joints allocated to each agent. The `agent_obsk` argument specifies the number of nearest joints an agent can observe. We set `agent_obsk=1` across all tasks. We collect 10 trajectories per mode as expert demonstrations, with each trajectory consisting of 1,000 transitions.

HalfCheetah (2x3, 6x1) The objective in these tasks is to move the multi-agent system forward and backward along the x-axis as quickly as possible while maintaining balance. The local observation comprises joint angles and angular velocities. Each expert system is trained using a reward function based on directional velocity ($\pm x$-axis velocity).

Ant (2x4, 4x2) and AntGoal (2x4, 4x2) The goal is to control a multi-agent system to move forward, backward, left, or right as fast as possible while maintaining balance (Ant), or to navigate to one of eight target positions uniformly distributed around a circle of radius 20 (AntGoal). The local observation space includes joint angles and angular velocities. For Ant, each expert system is trained using a reward function based on directional velocity ($\pm x$- or $\pm y$-axis velocity). In AntGoal, (x, y) position coordinates are included in local observations, and each expert system is trained using a reward function based on the distance between the robot's current position and the goal.

ManySegmentSwimmer (5x4, 10x2) The goal is to control a multi-agent system to swim forward, backward, left, or right as fast as possible. The local observation comprises joint angles and angular velocities. Each expert system is trained using a reward function based on directional velocity ($\pm x$- or $\pm y$-axis velocity).

C.3 MQE

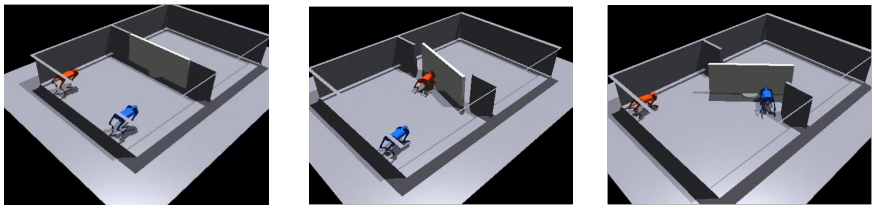


Figure 5: **Door** Two robots move to the next room via a revolving door by coordinating their crossing order.

Door In this task, two quadruped agents must navigate through a revolving door to reach the next room, as illustrated in Figure 5. Because the door cannot accommodate both agents simultaneously, they must coordinate the order in which they cross. The local observation includes the agent's own 3D position (x, y, z) and orientation (roll, pitch, yaw), the other agent's position and orientation, and the position and orientation of the door. We collect expert demonstrations across two distinct strategy modes, corresponding to the two possible crossing orders, with each strategy mode containing 10 episodes.

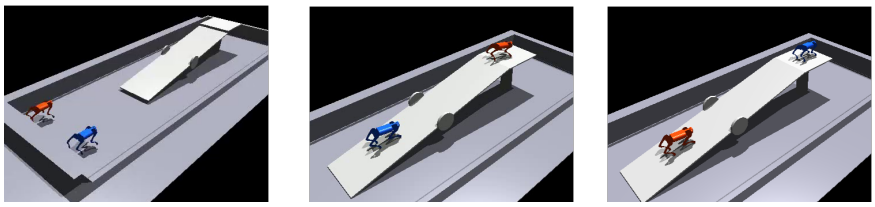


Figure 6: **Seesaw** One of two robots reach a target position located beyond a seesaw.

Seesaw In this task, one of two agents must reach a target position located beyond a seesaw, as illustrated in Figure 6. Because the target is at the far end, one agent must hold and stabilize the near end while the other climbs the seesaw to reach the goal. The local observation includes the agent's own 3D position (x, y, z) and orientation (roll, pitch, yaw), the other agent's position and orientation, and the seesaw's joint angle and angular velocity. We collect expert demonstrations across two distinct coordination modes, in which the two agents swap the roles of holder and reacher, with each strategy mode containing 10 episodes.

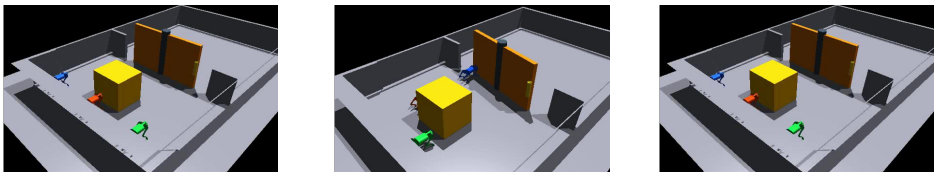


Figure 7: **Cargo Door** Three robots transport cargo to the adjacent room through a revolving door.

Cargo Door In this task, three agents must cooperate to transport cargo through a revolving door, as illustrated in Figure 7. One agent must open and hold the door while the other two transport the cargo through it. The local observation includes the agent's own 3D position (x, y, z) and orientation (roll, pitch, yaw), the other agents' positions and orientations, the cargo's position, and the door's position, angle, and angular velocity. We collect expert demonstrations across two distinct behavioral modes, where a different agent takes on the role of the door holder in each mode, with each strategy mode containing 10 episodes.

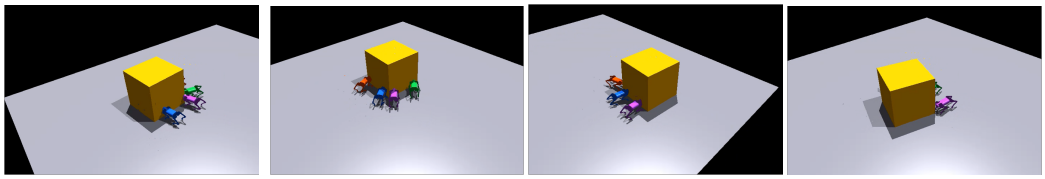


Figure 8: **Cargo** Four robots transport cargo to one of four corner positions.

Cargo Four agents must cooperate to transport cargo toward one of four corners, as illustrated in Figure 8. The agents must coordinate both their target direction and actions to transport the cargo successfully. The local observation includes the agent's own 3D position (x, y, z) and orientation (roll, pitch, yaw), the other agents' positions and orientations, and the cargo's position, orientation, linear velocity, and angular velocity. We collect expert demonstrations across four distinct behavioral modes, each corresponding to transporting the cargo toward a different corner, with each strategy mode containing 10 episodes.

In the partially observable environment variant denoted by the -P suffix, an agent observes another agent's position and orientation only when that agent falls within its 90° forward field of view. Otherwise, these state values are masked.

D Implementation Details

We use a uniform categorical distribution as the prior $p(z)$ during training. For evaluation, instead of sampling z uniformly, we sample from the dataset posterior $\hat{p}(z) \propto \int_{\mathcal{D}} p(\tau \mid z)p(z)d\tau$. For flow-matching policies, we adopt rectified flow models with $\alpha_\lambda = \lambda$ and $\sigma_\lambda = 1 - \lambda$. To sample from the flow-matching models, we use 16 flow time steps.

Gaussian baselines For Gaussian^{BC}, GMM^{ELBO}, and GMM$^{\text{ELBO+MI}}$, we employ standard Gaussian policies that output the mean and variance. These variants share the same core neural network architecture, with the GMM models additionally taking a latent conditioning vector as input. Specifically, the discrete latent variable $z \in \mathcal{Z}$ is mapped to a fixed-size embedding vector before being fed into the policy network. The trajectory encoder q_ϕ is implemented as a Recurrent Neural Network (RNN) that processes a subsequence of the trajectory. For GMM$^{\text{ELBO+MI}}$, the mutual information term $I(\tau; z)$ is optimized via its variational lower bound:

$$I(\tau; z) \geq \mathbb{E}_{\tau \sim \mathcal{D}, z \sim q_\phi(z|\tau)} \left[\log q_\phi(z \mid \tau)\right] + \mathcal{H}(z), \tag{16}$$

where $\mathcal{H}(z)$ denotes the entropy of the latent distribution z.

Flow-matching baselines For FM^{BC}, LatentFM^{ELBO}, and LatentFM$^{\text{ELBO+MI}}$, we construct individual vector fields that predict velocity vectors for flow-matching. Similar to the Gaussian baselines, the discrete latent variable is embedded as a fixed-size vector, and the trajectory encoder shares the same RNN architecture. For LatentFM$^{\text{ELBO+MI}}$, we use Eq. 16 to maximize the MI term.

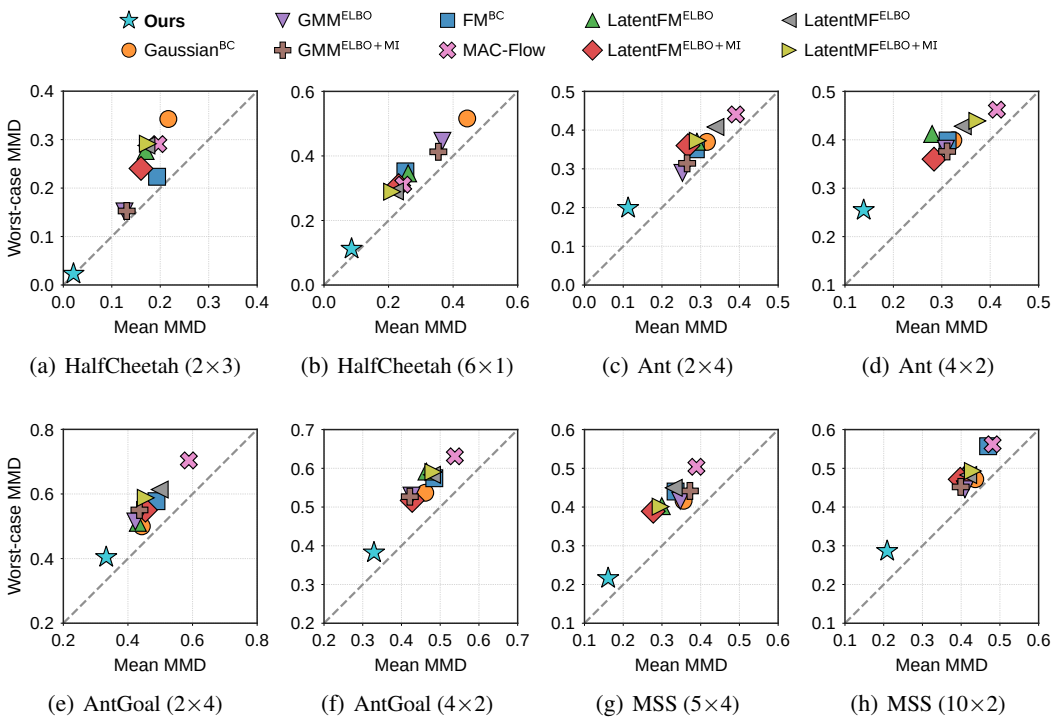
MAC-Flow and its variants MAC-Flow (MF) trains a joint flow-matching policy on demonstrations without latent variables via $\mathbb{E}_{\mathcal{D}}\left[||v^J_\eta(a_\lambda, o) - u_\lambda(\epsilon, a)||^2\right]$, which is subsequently distilled into individual flow-matching policies using a Mean Squared Error loss, $||v^{J,i}_{\eta,\lambda}(a_\lambda, o) - v^i_{\theta,\lambda}(a^i_\lambda, o^i)||^2$. In its latent variants, LatentMF^{ELBO} and LatentMF$^{\text{ELBO+MI}}$, the joint policy takes the latent variable z as input in the same manner as the Gaussian and flow-matching baselines, and is trained via $\mathbb{E}_{\mathcal{D}, q_\phi(z|\tau)}\left[||v^J_\eta(a_\lambda, o, z) - u_\lambda(\epsilon, a)||^2\right]$. The latent-conditioned individual policies are subsequently trained solely through distillation from the latent-conditioned joint policy via $||v^{J,i}_{\eta,\lambda}(a_\lambda, o, z) - v^i_{\theta,\lambda}(a^i_\lambda, o^i, z)||^2$. For LatentMF$^{\text{ELBO+MI}}$, we use Eq. 16 to train the encoder.

Ours Our proposed method consists of an encoder, a joint vector field, and individual vector fields. To ensure a fair comparison, all network components, including the encoder structure as well as the joint and individual vector field networks, use the exact same architectures across our method and all baseline algorithms. We train a joint flow-matching policy on demonstrations via $\mathbb{E}_{\mathcal{D}, q_\phi(z|\tau)}\left[||v^J_\eta(a_\lambda, o, z) - u_\lambda(\epsilon, a)||^2\right]$. We use the subsequence rather than the entire trajectory τ in Eq. 4. Eq. 5 precisely takes the form: $\mathbb{E}_{\mathcal{D}, z \sim q_\phi(z|\tau)}\left[\sum_t D_{\text{KL}}\left(\pi^J_\eta(\cdot \mid o^{(t)}, z) \parallel \prod_{i=1}^N \pi^i_\theta(\cdot \mid o^{i,(t)}, z)\right)\right]$. We accumulate this divergence in Eq. 5 over subsequence to train the encoder.

In MAMuJoCo, the number of latent variables is set to 4 for HalfCheetah, 8 for Ant, AntGoal, and ManySegmentSwimmer. In MQE, the number of latent variables is 4 for Door, Seesaw, and Cargo Door, and 8 for Cargo. The subsequence horizon is set to 64 for MAMuJoCo and 128 for MQE. These hyperparameters are shared with other baselines, particularly those that use the encoder.

E Additional Experimental Results

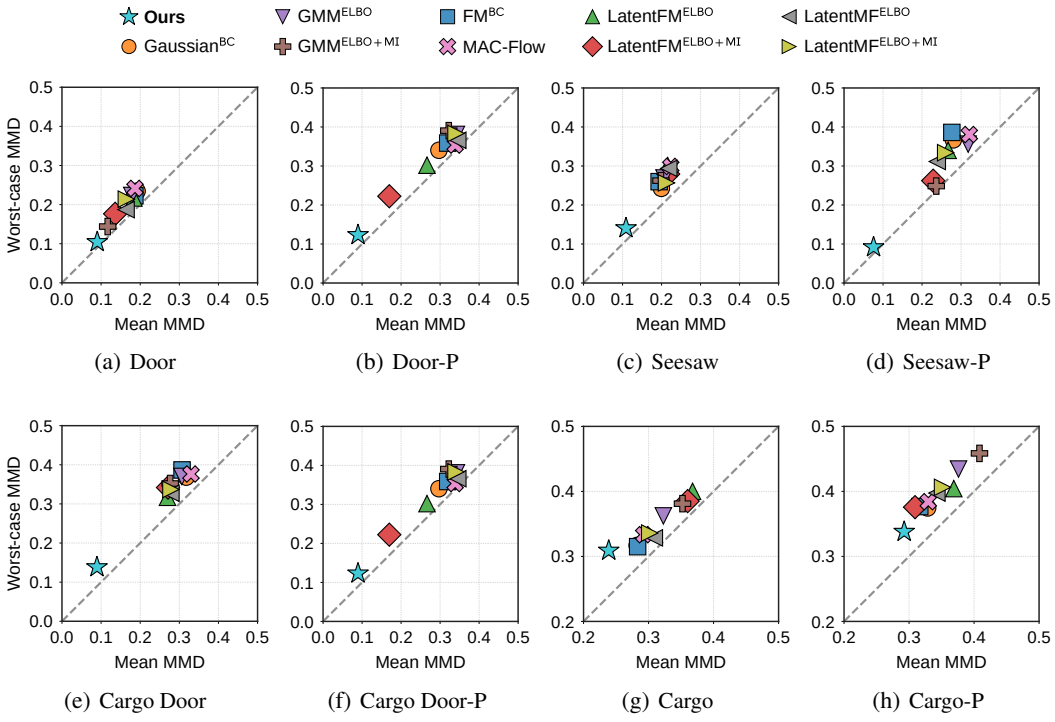
To compute MMD scores, we use a Gaussian kernel whose bandwidth is set to the median positive squared pairwise distance across expert transitions. Please refer to the attached supplementary material for videos illustrating the behaviors learned by our approach.



(a) HalfCheetah (2×3) (b) HalfCheetah (6×1) (c) Ant (2×4) (d) Ant (4×2)

(e) AntGoal (2×4) (f) AntGoal (4×2) (g) MSS (5×4) (h) MSS (10×2)

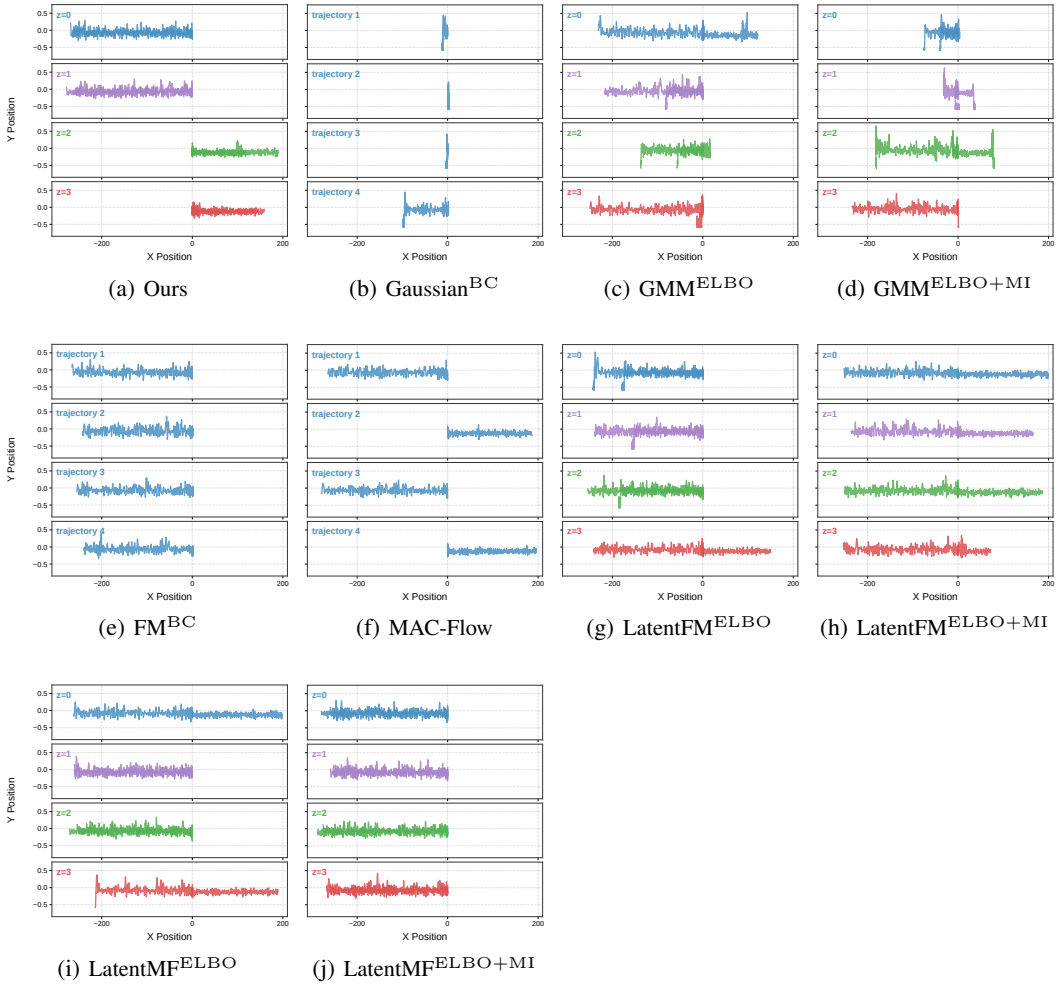
Figure 9: Average and worst-case MMD across MAMuJoCo environments (↓). Lower values indicate better distribution matching.



(a) Door (b) Door-P (c) Seesaw (d) Seesaw-P

(e) Cargo Door (f) Cargo Door-P (g) Cargo (h) Cargo-P

Figure 10: Average and worst-case MMD across MQE environments (↓). The suffix -P indicates partial observability. Lower values indicate better distribution matching.

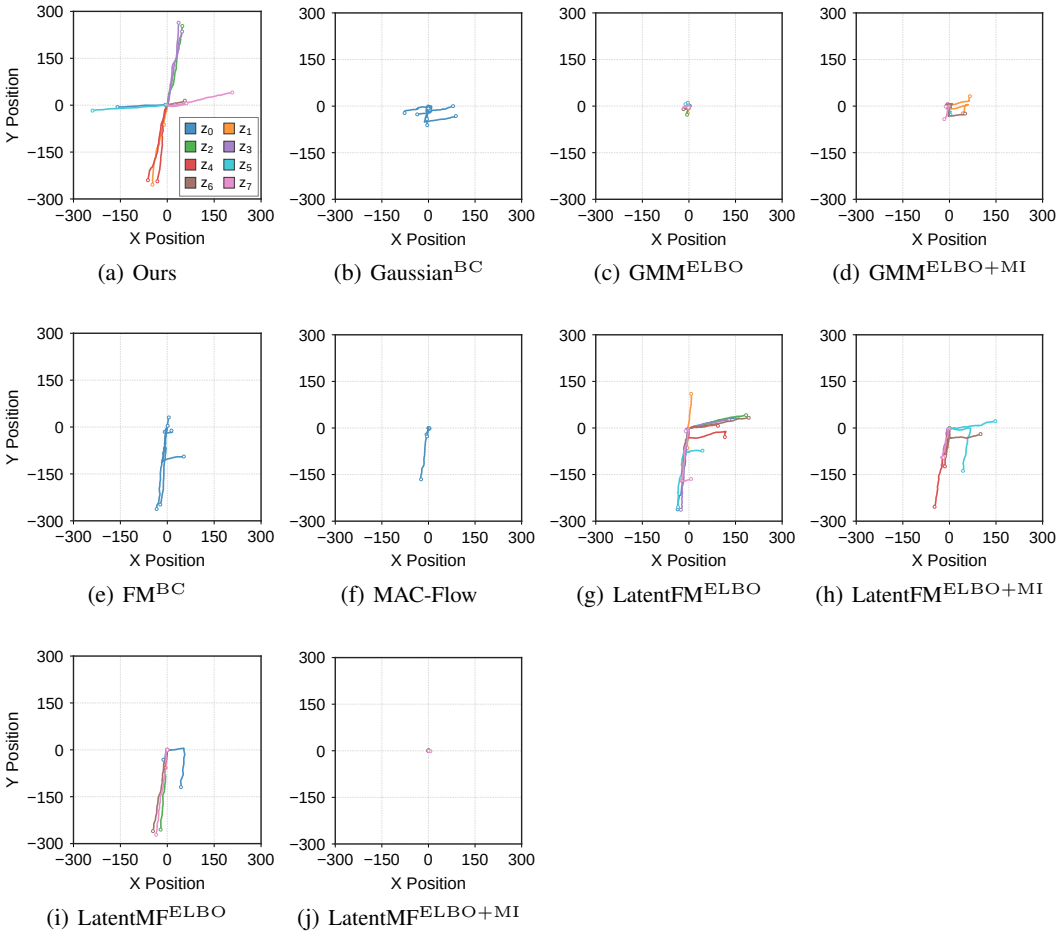


(a) Ours (b) Gaussian^{BC} (c) GMM^{ELBO} (d) GMM$^{\mathrm{ELBO+MI}}$

(e) FM^{BC} (f) MAC-Flow (g) LatentFM^{ELBO} (h) LatentFM$^{\mathrm{ELBO+MI}}$

(i) LatentMF^{ELBO} (j) LatentMF$^{\mathrm{ELBO+MI}}$

Figure 11: Learned behaviors in the HalfCheetah (6×1) across methods. Initial positions are randomly initialized around (0,0).

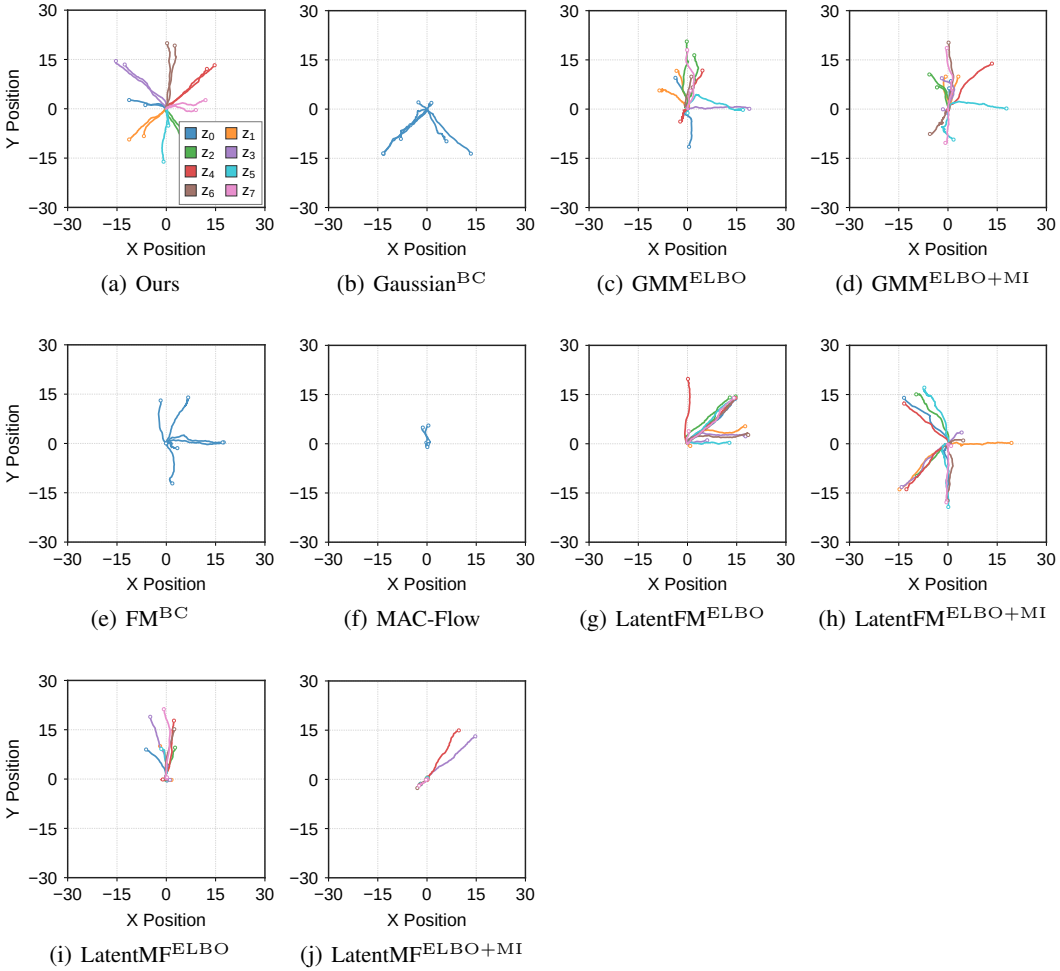


(a) Ours (b) Gaussian^{BC} (c) GMM^{ELBO} (d) GMM$^{\mathrm{ELBO+MI}}$

(e) FM^{BC} (f) MAC-Flow (g) LatentFM^{ELBO} (h) LatentFM$^{\mathrm{ELBO+MI}}$

(i) LatentMF^{ELBO} (j) LatentMF$^{\mathrm{ELBO+MI}}$

Figure 12: Learned behaviors in the Ant (4×2) across methods. Initial positions are randomly initialized around (0,0).

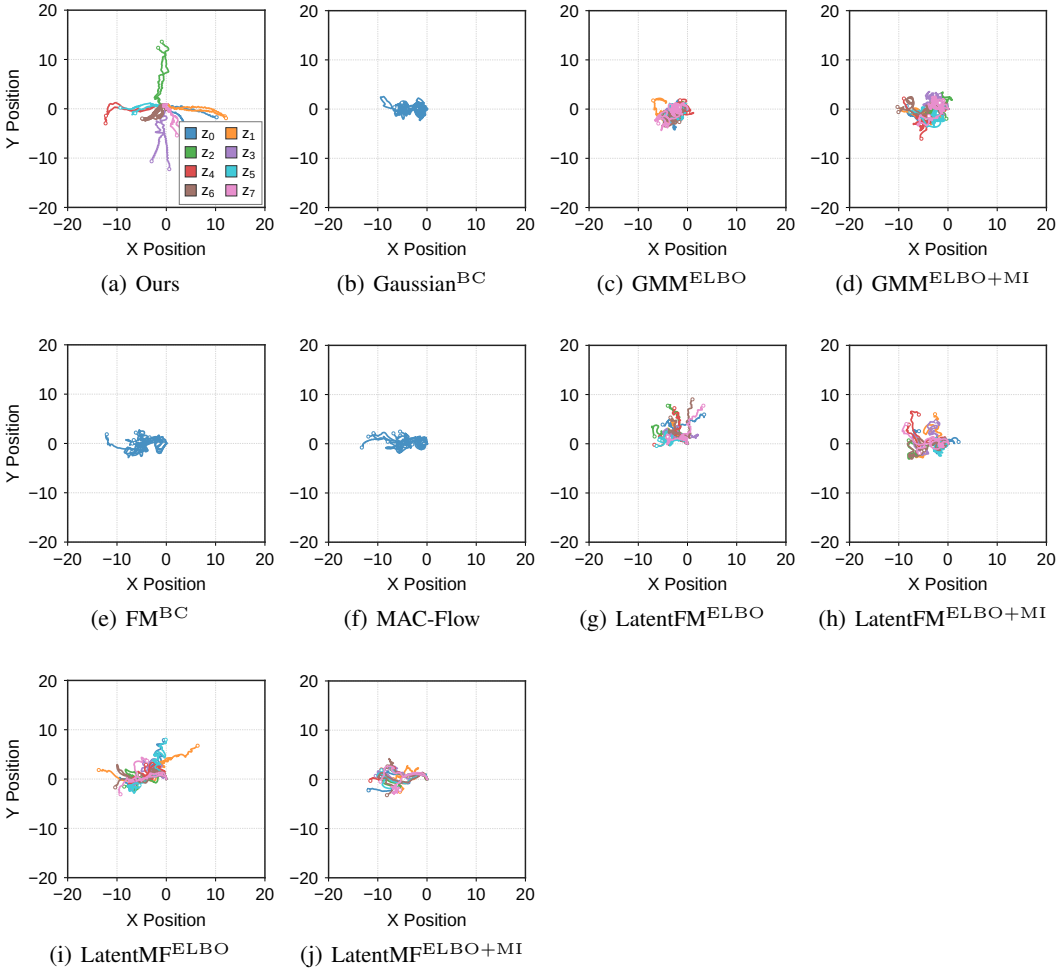


(a) Ours (b) Gaussian^{BC} (c) GMM^{ELBO} (d) GMM$^{\mathrm{ELBO+MI}}$

(e) FM^{BC} (f) MAC-Flow (g) LatentFM^{ELBO} (h) LatentFM$^{\mathrm{ELBO+MI}}$

(i) LatentMF^{ELBO} (j) LatentMF$^{\mathrm{ELBO+MI}}$

Figure 13: Learned behaviors in the AntGoal (4×2) across methods. Initial positions are randomly initialized around (0,0).



(a) Ours (b) Gaussian^{BC} (c) GMM^{ELBO} (d) GMM$^{\mathrm{ELBO+MI}}$

(e) FM^{BC} (f) MAC-Flow (g) LatentFM^{ELBO} (h) LatentFM$^{\mathrm{ELBO+MI}}$

(i) LatentMF^{ELBO} (j) LatentMF$^{\mathrm{ELBO+MI}}$

Figure 14: Learned behaviors in the ManySegmentSwimmer (10×2) across methods. Initial positions are randomly initialized around (0,0).