UC Davis

UC Davis Electronic Theses and Dissertations

Title

Generative Models in Decentralized Multi-agent Decision-Making.

Permalink

https://escholarship.org/uc/item/2rc7p32r

ISBN

9798193262411

Author

Huh, Dom

Publication Date

2026-09-08

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Generative Models in Decentralized Multi-agent Decision-Making.

By

DOM HUH
DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Prasant Mohapatra, Chair

Burkhard Schipper

Ilias Tagkopoulos

Committee in Charge

2026

Copyright © 2026
by
Dom Huh

Abstract

Decentralized multi-agent decision-making is ubiquitous in modern economies and real-world markets across a wide range of scales. These settings are naturally studied and modeled through computational models of multiple rational agents, each driven by distinct utility functions and local objectives, operating under information asymmetry and constrained, ego-centric perspectives. Under such conditions, agents face the unique challenge of navigating a balance between individual optimization and collective outcomes with restricted centralized signals. Realizing desirable collective behaviors in practical multi-agent systems requires overcoming several fundamental challenges, including combinatorial computational complexity, non-stationarity induced by the concurrent learning and confounding adaptations of other agents, and the need for coordination under incomplete information and strategic uncertainty. Such intricacies complicate the design of mechanisms and learning algorithms that reliably achieve efficient, stable, and desirable collective outcomes.

Recent advances in generative models have demonstrated remarkable capabilities for learning rich latent representations of complex data distributions. In this thesis, we investigate generative models as a foundational framework for decentralized multi-agent decision-making. First, we propose a representation learning approach that incorporates auxiliary learning objectives to capture information specific to multi-agent interactions, thereby enriching deep multi-agent reinforcement learning. Next, we study diffusion-based planning for optimizing key performance indicators in large-scale auto-bidding systems operating under diverse auction mechanisms and market designs. Finally, we explore the use of large language models as reasoning agents within multi-agent systems by developing structured prompt chaining to guide in-context learning toward multi-agent-specific information and a representation learning-based activation steering method to better align their internal representations with multi-agent objectives. Collectively, these contributions demonstrate the potential of generative modeling as a unifying paradigm for scalable, adaptive, and strategically grounded multi-agent decision-making.

Contents

Contents iii

List of Figures v

1 Introduction 1
- 1.1 Overview . . . 1
- 1.2 Contributions . . . 2
- 1.3 Background . . . 5
  - 1.3.1 Multi-agent Systems . . . 5
  - 1.3.2 Generative Modeling . . . 10
- 1.4 Conclusion . . . 12

2 A Generative Framework for Decentralized Multi-Agent Decision-Making 14
- 2.1 Introduction . . . 14
- 2.2 A Theoretical Construct for Generative Models in Multi-Agent Systems . . . 15
  - 2.2.1 Latent Variables for Modeling Other Agents . . . 15
  - 2.2.2 Latent Variables for Planning . . . 16
  - 2.2.3 Latent Variables using Natural Language . . . 17
- 2.3 Conclusion . . . 17

3 Representation Learning in Multi-agent Systems 19
- 3.1 Latent Representation Learning for Multi-agent Reinforcement Learning . . . 20
  - 3.1.1 Introduction . . . 20
  - 3.1.2 Related Works . . . 22
  - 3.1.3 Preliminaries . . . 23
  - 3.1.4 Multi-agent Latent Space Optimization . . . 24
  - 3.1.5 Integrating MA-LSO to Multi-agent Policy Optimization . . . 27
  - 3.1.6 Experiments . . . 29
  - 3.1.7 Results . . . 29
  - 3.1.8 Sensitivity Analysis . . . 38
  - 3.1.9 Limitations and Future Directions . . . 40
  - 3.1.10 Conclusion . . . 41
- 3.2 Conclusion . . . 41

4 Generative Models for Multi-agent Planning 43

4.1 Aligning Diffusion Models for Decision-Making . . . 43
4.1.1 Introduction . . . 43
4.1.2 Background . . . 45
4.1.3 Reward Alignment of DMC . . . 46
4.1.4 Design and Training Choices . . . 52
4.1.5 Experiments . . . 53
4.1.6 Conclusion . . . 59
4.2 Multi-agent Auto-Bidding with Latent Graph Diffusion Models. . . . 60
4.2.1 Introduction . . . 60
4.2.2 Latent Graph Diffusion Model for Auto-Bidding . . . 63
4.2.3 Results and Ablations Studies . . . 68
4.2.4 Conclusion . . . 71
4.3 Limitations and Future Directions . . . 72
4.4 Conclusion . . . 73

5 Generative Models for Multi-agent Reasoning. 74
5.1 Large Language Models for Decentralized Games . . . 74
5.1.1 Introduction . . . 74
5.1.2 Background . . . 75
5.1.3 Multi-agent Prompting Design . . . 77
5.1.4 LLM Selection and Design for a Decision-making Context . . . 81
5.1.5 Aligning LLMs for Multi-agent Decision-Making . . . 84
5.1.6 Conclusion . . . 89
5.2 Steering LLMs for Multi-agent Decision-making using Representation Learning 91
5.2.1 Introduction . . . 91
5.2.2 Method . . . 92
5.2.3 Results . . . 94
5.2.4 Conclusion . . . 96
5.3 Limitations and Future Directions . . . 96
5.4 Conclusion . . . 97

6 Conclusion 99

Bibliography 100

List of Figures

1.1 A general diagram of a multi-agent system following an iterative control process between agents $\{0, \ldots, N\}$ and their shared environment. 6

3.1 A high-level illustration of the MAPO-LSO framework applied on a decentralized setting, specifically, a multi-modal three-player robotic team task. We denote s_t as the state and o_t^i as the observation of agent i at time-step t. All observations consist of local visual inputs and local proprioceptive information q_t^i. The observations are processed in a distributed manner by each agent to compute the latent state z_t^i. Our proposed MA-LSO learning process is applied on z_t^i, which is used to compute the value and action a_t^i. 21

3.2 An illustration of the intended goals of MA-LSO. Given the state trajectory $\{s_{t-1}, s_t, s_{t+1}\}$ and the corresponding latent state trajectory $\{z_{t-1}, z_t, z_{t+1}\}$, MA-LSO aims to enforce consistency along different targeted dimensions, indicated by the arrows. 25

3.3 In these graphs, the normalized collective returns are compared between variants of our proposed solution. We additionally present the effects of different design choices, such as phasic regularization, Bayesian network, pretraining and Dyna-like training. 32

3.4 Accuracy of Belief Space of Other Agents, comparing the use of Bayesian network representation for our Decoder and the inclusion of communication. 34

3.5 A visualization of agent 0's observational belief space $\omega^i(o)$ alongside the ground truth observations of all agents in Visual-ABB3-C-LC task, where the yellow boxes indicate communication steps. 35

3.6 The average return ratio on all IST tasks and MARL algorithms, divided in the axis of communication (NC: no communication, LC: limited communication, FC: full communication) and nature of interaction (N.O.I) (C: collaborative, M: mixed, A: adversarial) on 32 random seed. 36

3.7 Performances of individual MARL algorithms, with and without MA-LSO. . . . 38

3.8 Sensitivity Analysis with MAPO-LSO trained agents for ABB3-C-FC from MAPPO over 30 trials. We scale the perturbations and noise for observations, rewards and latent states on the left figure, and scale the masking probability of communication on the right. 39

4.1 A illustration of the reward maximization of diffusion model $p_\theta(\cdot)$ for control, visualizing the diffusion process at denoising steps $\tau \to \tau - 1$ such that $z_t^{\tau-1}$ is denoised towards the target distribution x_t^0 but also aligned with maximize the return distribution G_t. The resulting trajectory distribution that is more aligned with the return distribution is shown in red. 44
4.2 A visualization of the effects of alignment to the diffusion process. Given a foundation DDPM, whose outputs and diffusion field over the entire state space are shown in gray, we align its diffusion process to maximize three separate reward functions, shown in orange, green and blue. 47
4.3 A visual illustration of the two DMC frameworks on Walker2D task, where the planning-based DMC is forecasting future states and the policy-based DMC generates actions directly. 48
4.4 Visualization of the Nav1D task and an example of an agent's trajectory, shown in green, and the actions space is shown in gray. The reward distributions at every time step are shown in blue. 53
4.5 Alignment Learning Curve for Diffuser (DDPM) on the D4RL Medium Datasets, where the average and ± 1 standard deviation is shown over 64 episode seeds. The yellow vertical line indicates when the online fine-tuning is introduced. . . . 58
4.6 Alignment Learning Curve and Coherency Score for DP (Expectation) on the Nav1D Datasets, where the average and ± 1 standard deviation is shown over 32 episode seeds. The yellow vertical line indicates when the online fine-tuning is introduced and the black horizontal line represents the average return achieved in the respective dataset. 59
4.7 An illustration of the LGD-AB framework. At every time-step, a graph-based embedding is computed for agent i using a bipartite graph of agent and IO nodes. The nodes of a sub-graph $\mathcal{G}_t^i$ consist of two virtual nodes, connecting to IOs exposed and not exposed to agent i respectively. Sub-graphs for other agents are similarly formed and connected accordingly. The graph is processed using a GNN to generate an embedding vector x_t^i. The diffusion model then learns a posterior of the temporal sequence of embedding vectors to forecast auction dynamics. . . 62
4.8 Agent i's Approximation of Other Agents' Graph $\omega^i(\mathcal{G}_t^{-i})$ by creating h proxy subgraphs. This graph is used as input for the student model $\mathcal{S}$. 63
4.9 Inverse Dynamics Model for Bid Computation, which utilizes the latent state, context vector and node features to compute the bid. 65
4.10 Computing Joint Embedding Vector using Self-Attention Network. 66
4.11 KPI Alignment Learning Curve on Synthetic Auction with multi-KPI alignment (shown in red) compared to optimizing each KPI criterion independently (shown in blue). For social welfare, we compare the training between EC and non-EC variants, as shown in the dotted lines. 69

5.1 An example of MD-LLM on the game of Chicken, generating new mechanisms to guide players towards desired solution concept. 83
5.2 Misaligned Objectives on Prisoner's Dilemma. The purple boxes are the Pareto efficient strategies, i.e., the objective of Player A, and the blue box is the N.E., i.e., the objective of Player B. 90

5.3 An illustration of our representation learning process applied to the steering vector z_t^i for agent i at timestep t. The latent belief space depicts the self-consistency that is enforced within the latent space dynamics and the theory-of-mind (ToM) belief space shows the grounding of steering vectors to the multi-agent environment, including the observation o, action a and rewards r. 93

Chapter 1

Introduction

1.1 Overview

Decision-making agents are autonomous entities characterized by their ability to perceive their environment, process information, and act purposefully to achieve specific objectives. When these entities operate within a shared environment, they form a multi-agent system, a framework where the collective intelligence of the group often exceeds the sum of its individual parts. Today, these systems serve as the backbone of critical infrastructure. They power autonomous traffic management, drive decentralized power grids that balance energy loads in real-time, and provide the algorithmic foundation that allows global markets to co-exist and thrive.

Developing decision-making agents within multi-agent systems aims to foster the emergence of complex control and social behaviors that enable agents to navigate the complexities of real-world systems. Through appropriate modeling and learning mechanisms, such agents should not only be capable of understanding intricate social dynamics but also be able to leverage the collective nature of their environment to achieve individual objectives [1–3]. By focusing on these behaviors and their underlying mechanisms, our goal is to equip agents with the ability of effective and strategic decision-making. Ultimately, this approach seeks to endow agents with desirable behavioral qualities that characterize intelligent social systems through data-driven learning processes.

However, achieving this objective presents significant challenges. Beyond the already difficult task of optimizing control policies [4, 5], multi-agent settings introduce additional complexity due to the presence of other agents whose behaviors act as dynamic and often

unpredictable confounding factors.

A promising frontier in addressing these challenges lies in the integration of generative models to support comprehensive modeling of strategic multi-agent reasoning. Generative models are probabilistic frameworks designed to approximate high-dimensional data distributions and capture underlying structures within complex phenomena [6]. While traditional generative models primarily focus on learning distributions from large and often non-stationary datasets [7, 8], such as text [9] and images [10], this thesis investigates their application and utility for decentralized multi-agent decision-making.

1.2 Contributions

This thesis addresses the fundamental challenges of optimizing multi-agent decision-making by integrating multi-agent reinforcement learning (MARL) methods with generative modeling techniques using insights from game-theoretic principles. We present methods that enable agents to develop rich latent representations specific to multi-agent dynamics, to utilize generative trajectory planning to roll-out and forecast within multi-agent environment, and ground a shared language to support strategic reasoning and navigate complex, shared environments. While distinct in their methodologies, these individual works that comprise this thesis share the underlying goal of introducing generative models to multi-agent decision-making in unique ways. Collectively, they contribute to a research trajectory aimed at increasing the cognitive and strategic complexity of multi-agent systems and improvements to their learning dynamics using generative modeling.

Representation Learning in Multi-agent Systems. We first explore the enrichment of agent perspectives through latent structures using generative models [11]. In Chapter 3, we address the challenges of sample-efficiency and learning variance of MARL algorithms [12] with an extension of representation learning [13] tailored for multi-agent decision-making, distilling information uniquely related to multi-agent settings within a latent space. We focus our research on achieving this distillation through optimizing auxiliary objectives to not only embed this information within the agent's latent space, but also regularize the MARL training. Through our proposed framework, agents are able to develop and utilize a richer context of the joint environment state, demonstrating improved stability and efficiency of decentralized learning. The contributions of this work include:

- We introduce a novel representation learning framework that integrates with popular MARL algorithms. We study several design choices for effective integration to maximize its utility towards training stability, sample efficiency and performance.

- We provide empirical results that compares the performance of vanilla MARL algorithms with MARL algorithms using our framework, demonstrating notable improvements in sample efficiency and overall performance in a suite of multi-agent tasks, including challenging robotic teams applications such as multi-agent reach [14] and various multi-agent scenarios from Vectorized Multi-agent Scenarios (VMAS) [15].

- We perform ablative studies involving the different components of our framework to affirm a comprehensive understanding of the performance improvement, and study unique multi-agent considerations, such as its effects on ad-hoc team-play [16].

Generative Multi-agent Planning. We now shift our focus toward generative model-based planning for scalable trajectory synthesis in market-driven competitive environments [17, 18]. In Chapter 4, our research extends the use of diffusion models for planning and automated bidding (auto-bidding) [19]. We first investigate alignment methods to maximize utility using such methods and apply the studied concepts to auto-bidding environments. In these highly-dynamic market environments, we propose the use of graph representations to model the complex, interconnected relationships between bidders and items, as well as the auto-bidding dynamics. By leveraging a diffusion-based planning architecture [20], agents synthesize and follow favorable trajectories that maximize their utility while accounting for the dynamic nature of large-scale auto-bidding. The contributions of this work include:

- We propose using graph neural networks [21] to model the structural dependencies of high-dimensional auction environments, capturing the relational context between bidders and items as well as inter-agents and inter-item relationships.

- We adopt the diffusion-based planning paradigm to the constraints of market dynamics, enabling agents to execute long-horizon strategic planning while maintaining robustness against the non-stationary behaviors of bidders. We optimize this process over several crucial key-performance indicator (KPI) metrics, such as clicks-per-acquisition, win rate, budget adherence, and social welfare [22].

- We demonstrate the efficacy of our approach in large-scale auto-bidding simulations, showing that generative trajectory synthesis outperforms competitive baselines on many KPI metrics for auto-bidding.

Multi-agent Reasoning with Language Models. Lastly, we investigate using generative models to drive the reasoning process of multi-agent decision-making systems [23, 24]. In Chapter 5, we explore Large Language Models (LLMs) as reasoning models within the multi-agent decision-making process. We investigate the intersection of the use of natural language reasoning to generate strategic actions, specifically studying prompt engineering methods [25] and alignment methods [26] to ground shared natural language for multi-agent tasks. We introduce prompt chaining methods that deconstruct strategic reasoning and steering methods that align with multi-agent dynamics, ensuring that the emergent behaviors are both strategically grounded and socially compliant. The contributions of this work include:

- We extend the application of LLMs to multi-agent decision-making by introducing novel prompt chaining method to guide in-context learning and systematically decompose high-level strategic goals into hierarchical stages of natural language reasoning.
- We propose a novel alignment method using activation steering [27] to modulate the internal representations of LLM-based agents, ensuring their decision-making remains consistent with predefined social norms and strategic constraints without requiring computationally expensive fine-tuning.
- We demonstrate the efficacy of our frameworks through comprehensive experimentation on classic game-theoretic benchmarks and more complex, multi-agent scenarios, affirming that prompt chaining and steering methods improve coordination and objective adherence in social agents.

Together, these works provide a comprehensive exploration of how generative paradigms can optimize decentralized decision-making systems into more complete reasoning entities. The scope of this thesis is primarily centered at the intersection of deep MARL, generative modeling, and game theory. While we draw upon and take inspiration from economic principles of utility and strategic interaction, our investigation is bounded by a computational and data-driven perspective; specifically, we focus on capturing complex and higher-dimensional

multi-agent tasks and the alignment of deep models with target multi-agent objectives. The research undertaken as part of this dissertation has resulted in the following publications [11, 17, 18, 23, 24, 28].

1.3 Background

Decision-making systems are fundamental to many areas of artificial intelligence, enabling agents to select actions that achieve desired goals within complex and uncertain environments. These systems appear in a wide range of real-world applications, including autonomous vehicles coordinating in traffic, robotic teams performing warehouse logistics, financial agents making trading decisions, and distributed sensor networks monitoring dynamic environments [28]. Despite the diversity of these domains, they share a common underlying challenge: an agent must repeatedly choose actions while accounting for uncertainty, delayed consequences, and limited information. A common mathematical framework for modeling such sequential decision-making problems is the Markov Decision Process (MDP) [4]. An MDP is defined as the tuple $(S, A, \mathcal{T}, r, \gamma)$, where S represents the set of environment states, A is the set of actions available to an agent, $\mathcal{T} : S \times A \to P(S)$ denotes the state transition function that defines the probability of moving to a new state given the current state and action, $r : S \times A \to \mathbb{R}$ is the reward function that evaluates the desirability of actions, and $\gamma \in [0, 1)$ is a discount factor that balances immediate and future rewards. Within this framework, an agent follows a policy $\pi(a|s)$ that maps states to actions with the objective of maximizing the expected cumulative reward over time. While MDPs provide a principled foundation for modeling single-agent decision-making, many real-world systems involve multiple interacting agents whose decisions influence one another. These settings, called multi-agent systems, require extending the decision-making framework to account for strategic interaction, decentralized information, and coordination among agents.

1.3.1 Multi-agent Systems

Multi-agent systems consist of a population of decision-making agents that exist within a shared environment. As illustrated in Figure 1.1, agents may observe their surroundings and communicate with one another to execute actions aligned with their own goals and beliefs, often following a strategy or policy π. Within a decentralized setting, many of

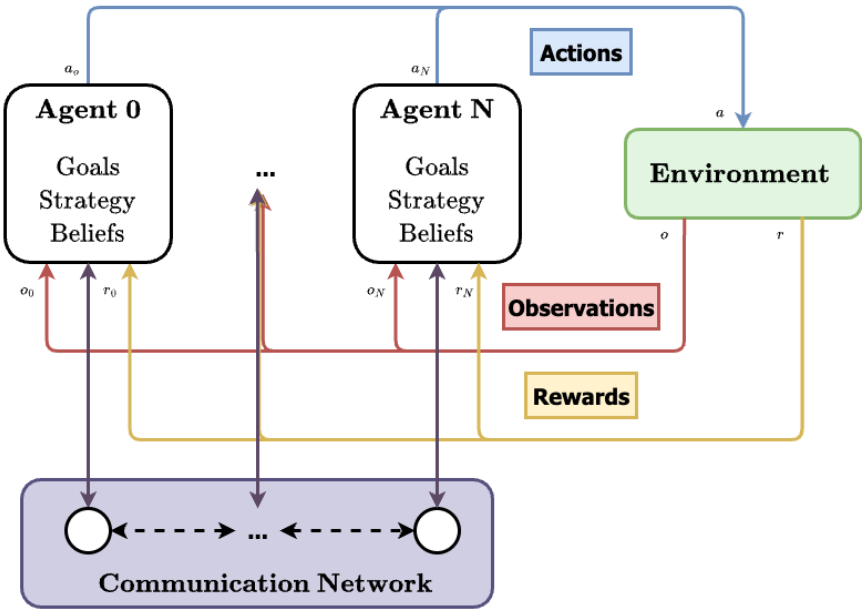


Figure 1.1: A general diagram of a multi-agent system following an iterative control process between agents $\{0, \ldots, N\}$ and their shared environment.

these components are constrained. Decentralization manifests in two forms: natural decentralization, influenced by physical constraints such as communication range, and artificial decentralization, which includes intentional adjustments like communication bandwidth to enhance operational feasibility [1]. Therefore, decentralized agents operate within predefined constraints that shape their behaviors and interactions.

A formal representation of the a multi-agent system is the stochastic game [2, 29], defined as a 5-tuple $(N, S, A, r, \mathcal{T})$ where these elements are defined as:

- N is the set of n agents.
- S is the set of states.
- $A = A_0 \times A_1 \times \cdots \times A_n$ is the joint action space.
- $r = r_0 \times r_1 \times \cdots \times r_n$ is the joint reward function.
- $\mathcal{T} : S \times A \times S \mapsto \mathrm{P}(S)$ is the state transition operator which maps a state-action pair to the probability of next states.

Often, agents cannot perceive the true state of the environment. Instead, agents typically have access only to observations. The joint observation space $O = O_0 \times \cdots \times O_n$ accounts for

imperfect information [30]. Observations do not necessarily satisfy the Markovian property and may provide limited insight into the true state of the environment. Even joint observational histories $h = \{o_0, o_1, \ldots, o_t\}$ can be insufficient to determine the state s_t accurately. However, the converse is true: the emission transition function $\mathcal{E}(o, s)$ defines the mapping from states to corresponding joint observations that can be induced:

$$\mathcal{E}(o, s) = \mathrm{P}(o|s). \tag{1.1}$$

The concept of imperfect information describes the constraints agents face regarding available information and reinforces the decentralization constraints. Imperfect information is closely related to two significant concepts: partial observability and incomplete information. Partial observability limits agents' access to a subset of the state that can be obscured by noise. In settings with incomplete information, agents lack common knowledge about the game being played, leading to uncertainties about various aspects of the state [31].

Our study of multi-agent decision-making reflects a natural interdisciplinary convergence between game theory and machine learning. At its foundation lies utility theory from game theory, which provides a formal framework for modeling rational behavior in strategic settings. Agents are assumed to have preferences over outcomes, represented by utility functions u_i, and to select strategies π_i that maximize expected utility. For a joint strategy profile $\pi = (\pi_i, \pi_{-i})$, the rational objective is defined as:

$$\pi_i^* \in \arg\max_{\pi_i \in \Pi_i} u_i(\pi_i, \pi_{-i}), \tag{1.2}$$

where Π represents the set of all possible joint strategies. This formulation enables principled analysis of strategic interaction and equilibrium concepts, most notably the Nash Equilibrium, where no agent can benefit by unilaterally changing its strategy:

$$u_i(\pi_i^*, \pi_{-i}^*) \geq u_i(\pi_i, \pi_{-i}^*) \quad \forall \pi_i \in \Pi_i, \forall i. \tag{1.3}$$

Building on this normative foundation, reinforcement learning (RL) introduces a computational perspective on decision-making under uncertainty. Rather than assuming full knowledge of payoff functions or transition dynamics, RL considers agents that learn through interaction with an environment. This framework relaxes the assumption of perfect rationality and instead models bounded rationality, where strategies emerge through trial-and-error learning, value estimation, and policy optimization in high-dimensional and uncertain environments. We define a trajectory $\tau = \{s_0, a_0, r_0, s_1, a_1, \ldots, s_T\}$ to be sampled from the

dynamics model distribution $p_\pi(\tau)$,

$$p_\pi(\tau) = P(s_0) \prod_{t=0}^{T} \pi(a_t|s_t)\mathcal{T}(s_t, a_t, s_{t+1}), \tag{1.4}$$

where $P(s_0)$ denotes the initial state distribution and T is the terminal time-step. The return $G_t(\tau)$ of a given trajectory at each time step t is the cumulative future discounted reward.

$$G_{t'}(\tau) = \sum_{t=t'}^{T} \gamma^t r_i(s_t, a_t) \tag{1.5}$$

where $\gamma \in [0, 1)$ is a discount factor balancing immediate and future rewards.

By iteratively updating policies π on sampled trajectories, agents approximate utility-maximizing behavior:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[G_0(\tau) \right], \tag{1.6}$$

In practice, RL algorithms approximate this objective through three primary paradigms [4]. Value-based methods estimate value functions such as the state-value function $V^\pi(s)$ or action-value function $Q^\pi(s, a)$ using the Bellman backup operator [32–34] and derive policies implicitly through greedy or soft action selection.

$$V^\pi(s) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[G_t(\tau) \mid s_t = s \right] \tag{1.7}$$

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[G_t(\tau) \mid s_t = s, a_t = a \right]. \tag{1.8}$$

Policy-based methods directly parameterize the policy to optimize the expected return. A common approach is to perform gradient ascent along the objective using the policy gradient, pioneered by the REINFORCE (REward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility) algorithm [35, 36].

$$\nabla_{\theta_\pi} J(\pi) = \nabla_{\theta_\pi} \mathbb{E}_{\tau \sim p_\pi(\tau)} [G_0(\tau) - b(\tau)] \tag{1.9}$$

where θ_π represents the parameters of the policy π and $b(\cdot)$ is the reinforcement baseline function used to stabilize the approximated reinforcement $G_t(\tau)$. As the expectation is computed through Monte Carlo sampling, it can often lead to high variance despite providing an unbiased estimate of the expected return. Hence, as noted by [36, 37], the choice of $b(\cdot)$ can improve convergence and performance by mitigating this variance.

Actor-critic methods combine these approaches by maintaining both a parameterized policy (i.e., the actor) and a value function estimator (i.e., the critic). Instead of estimating

the value function of an implicit policy, we model the value function for an explicitly defined policy. With access to a value estimate, temporal difference (TD) learning or n-step learning can be utilized instead of Monte Carlo sampling to approximate the offset reinforcement and improve the quality of the policy gradient by stabilizing the return estimate with a new baseline [37]. While these bootstrap methods introduce bias, the variance is considerately lowered as updates are no longer dependent on the entire trajectories but only on a subset of the trajectory [38].

MARL extends the RL paradigm to stochastic games, where multiple agents simultaneously learn and act within a shared environment. Instead of a single policy $\pi(a \mid s)$, each agent $i \in N$ maintains its own policy $\pi_i(a_i \mid o_i)$ conditioned on its local observation. Importantly, the joint trajectory distribution in the multi-agent setting becomes:

$$p_\pi(\tau) = P(s_0) \prod_{t=0}^{T} \left(\prod_{i=1}^{n} \pi_i(a_{i,t} \mid o_{i,t}) \right) \mathcal{T}(s_t, a_t, s_{t+1}), \tag{1.10}$$

and the expected return of each agent is:

$$G_{i,t'}(\tau) = \sum_{t=t'}^{T} \gamma^t r_i(s_t, a_t). \tag{1.11}$$

These changes illustrate a fundamental challenge of MARL: the trajectory distribution and return of each agent depend on the policies of all other agents. Consequently, the optimization objective is inherently coupled, where unlike the single-agent setting, maximizing the expected return of each agent may lead to unstable learning dynamics or suboptimal collective outcomes caused by various learning pathologies such as non-stationarity, relative overgeneralization, and miscoordination [39]. At the same time, the multi-agent setting enables algorithmic innovations that explicitly exploit interactions among agents, involving joint model representation and design [40] and centralized training with decentralized execution [41]

Through this formulation, MARL must unify the equilibrium concepts of game theory with the adaptive optimization mechanisms of RL. Agents must reason not only about environmental uncertainty, but also about the strategic adaptation of other agents, giving rise to cooperation, competition, coordination, and equilibrium selection phenomena that extend beyond the single-agent setting. In this way, MARL unifies the utility-theoretic foundations of strategic reasoning with scalable learning-based control, forming the computational backbone of modern multi-agent systems research.

1.3.2 Generative Modeling

Generative modeling is a central paradigm in modern machine learning focused on learning the underlying structure of complex data. Rather than only predicting labels or estimating values, generative models attempt to capture the full distribution of data, enabling the synthesis of new samples that resemble those observed during training. This capability has enabled a wide range of applications, including realistic image generation, natural language generation, molecular design, simulation of physical processes, and data augmentation for learning systems [6, 8]. Beyond data synthesis, generative models also provide powerful tools for representation learning, uncertainty estimation, and modeling latent structure in high-dimensional environments. As a result, they have become a foundational component of many contemporary AI systems.

Formally, generative modeling provides a probabilistic framework for learning and sampling from complex, high-dimensional data distributions. A generative model $p_\theta(x)$ seeks to approximate an underlying data distribution $p_{\text{data}}(x)$, enabling the synthesis of new samples that preserve the structural properties of observed data. This is typically achieved by maximizing the marginal log-likelihood (MLL) over model parameters θ [6]:

$$\mathcal{L}_{\text{MLL}}(\theta) = \mathbb{E}_{x \sim p_{\text{data}}} \left[\log p_\theta(x) \right]. \tag{1.12}$$

Maximizing $\mathcal{L}_{\text{MLL}}$ is equivalent to minimizing the forward Kullback–Leibler divergence between the data distribution and the model distribution:

$$\theta^* = \arg\min_\theta \mathrm{D}_{\text{KL}} \left(p_{\text{data}}(x) \,\|\, p_\theta(x) \right). \tag{1.13}$$

Early deep generative approaches, such as variational autoencoders (VAEs), demonstrated the ability to learn compact latent representations that encode salient structure in data [42]. VAEs provide a tractable optimization approach to Equation 1.13 by introducing a parameterized encoder $q_\phi(z \mid x)$ to approximate the intractable posterior $p_\theta(z \mid x)$, together with a parameterized decoder $p_\theta(x \mid z)$. Instead of directly maximizing the marginal log-likelihood $\log p_\theta(x)$, an evidence lower bound (ELBO) can be derived such that maximizing $\mathcal{L}_{\text{ELBO}}$ maximizes the log probability of our data by proxy [43]:

$$\log p_\theta(x) = \mathcal{L}_{\text{ELBO}}(x) + \mathrm{D}_{\text{KL}}(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)), \tag{1.14}$$

which implies a lower bound since KL divergence ≥ 0:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z) \right] - \mathrm{D}_{\text{KL}}(q_\phi(z \mid x) \,\|\, p(z)) \leq \log p_\theta(x). \tag{1.15}$$

Here, $p(z)$ denotes a prior distribution over the latent variables, typically chosen as a standard multivariate normal distribution, $p(z) = \mathcal{N}(0, I)$. Intuitively, the first term encourages accurate reconstruction of the data, while the Kullback–Leibler divergence regularizes the approximate posterior toward the prior, inducing a structured and continuous latent space.

More recently, diffusion-based generative models have emerged as powerful alternatives, modeling data generation as a gradual denoising process that iteratively transforms noise into structured samples. Formally, the generative process is defined through a forward noising process and a learned reverse denoising process. Let $x_0 \sim p_{\text{data}}(x)$ denote a data sample. The forward process gradually adds Gaussian noise over T timesteps:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\Big(x_t; \sqrt{1-\beta_t}\, x_{t-1},\, \beta_t I\Big), \quad q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \tag{1.16}$$

where $\{\beta_t\}_{t=1}^{T}$ is a predefined noise schedule [44]. This forward process yields the closed-form marginal:

$$q(x_t \mid x_0) = \mathcal{N}\Big(x_t; \sqrt{\bar{\alpha}_t}\, x_0,\, (1-\bar{\alpha}_t) I\Big), \tag{1.17}$$

with $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

The reverse process is parameterized by a neural network $p_\theta(x_{t-1} \mid x_t)$, which approximates the true posterior $q(x_{t-1} \mid x_t, x_0)$. In practice, the score function or the added noise is modeled. The training objective is derived from a variational bound on the log-likelihood of the data distribution [45]:

$$\mathcal{L}_{\text{VLB}} = \mathbb{E}_{q(x_{1:T}|x_0)} \Bigg[\underbrace{\mathrm{D}_{\mathrm{KL}}\big(q(x_T \mid x_0) \,\|\, p_\theta(x_T)\big)}_{\text{prior matching}} \tag{1.18}$$

$$+ \sum_{t=2}^{T} \underbrace{\mathrm{D}_{\mathrm{KL}}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big)}_{\text{denoising matching}} \tag{1.19}$$

$$- \underbrace{\log p_\theta(x_0 \mid x_1)}_{\text{reconstruction term}} \Bigg].$$

For some intuition, the first term encourages the learned prior $p_\theta(x_T)$ to match the final noisy state of the forward process, the sum of KL divergences ensures that each reverse step approximates the true posterior, and the final term accounts for reconstruction of the original data from the first denoised step [45, 46].

A significant development in generative modeling is the emergence of large language models (LLMs). LLMs are autoregressive generative models trained on massive text corpora to approximate $p(x_t \mid x_{<t})$, enabling coherent long-form generation and complex reasoning through natural language. Beyond text synthesis, LLMs have demonstrated capabilities in structured planning, tool use, and multi-step reasoning via prompt conditioning and in-context learning [47]. From a standpoint of utilizing LLMs for sequential decision-making, LLMs can be interpreted as generative world models operating over linguistic state representations, allowing agents to perform high-level deliberation, simulate social interactions, and align behavior with abstract objectives expressed in natural language.

In this thesis, generative modeling is not treated merely as a tool for data synthesis, but as a foundational mechanism for enhancing decentralized learning and coordination in multi-agent systems. Generative models provide structured latent representations that capture high-dimensional correlations in states, observations, and joint behaviors. Such representations can mitigate incomplete information by inferring latent state structure from limited observations, effectively functioning as learned belief models. Alternatively, generative models can be leveraged for trajectory modeling and planning. By learning a distribution over future state-action sequences, agents may perform model-based roll-outs, evaluate counterfactual joint behaviors, or anticipate the responses of other agents. This capability is particularly relevant in MARL, where the trajectory distribution depends on the evolving policies of multiple agents. Generative models can also contribute towards improving the reasoning processes of strategic decision-making. With LLMs, concepts that represent goals, plans, and social commitments can be expressed through a shared linguistic representations.

1.4 Conclusion

Decentralized multi-agent systems present a uniquely challenging setting for decision-making and learning. Agents must operate under incomplete information, adapt to the evolving strategies of other agents, and coordinate their behavior within a shared environment while pursuing individual objectives. These factors introduce fundamental difficulties that extend beyond traditional single-agent learning, including non-stationarity, coordination failure, and the complexity of modeling strategic interactions. Addressing these challenges requires methods that move beyond purely reactive control and instead enable richer representations, foresight, and reasoning about multi-agent dynamics.

This dissertation presents methods using generative modeling to provide a powerful and flexible paradigm for advancing the capabilities of decentralized multi-agent systems. By learning structured representations of complex environments, generative models enable agents to capture latent relationships within joint observations and behaviors. By modeling distributions over trajectories, they support planning and anticipation in environments where outcomes depend on the interactions of many strategic actors. Finally, by leveraging language-based generative models, agents can reason about goals, strategies, and social constraints through shared symbolic representations.

Chapter 2

A Generative Framework for Decentralized Multi-Agent Decision-Making

2.1 Introduction

The preceding chapter introduced the fundamental challenges of decentralized multi-agent systems, including partial observability, strategic coupling between agents, and non-stationary learning dynamics. While the subsequent chapters of this dissertation present distinct approaches, they share a common underlying principle: the use of generative models to structure, infer, and optimize over the space of multi-agent interactions.

When considered independently, these approaches may appear as separate methodological contributions. This chapter establishes a unifying framework that situates these methods as instances of a broader construct for multi-agent decision-making. In doing so, we provide a coherent interpretation of decentralized learning, planning, and reasoning as inference over structured generative processes.

The formulation presented in this chapter is theoretical and conceptual in nature. Rather than introducing a single end-to-end system, it serves as a unifying perspective and a general blueprint for future multi-agent systems built upon generative principles. The subsequent chapters can be interpreted as concrete instantiations of specific components within this broader framework, each exploring representation learning, trajectory generation, and reasoning in isolation.

2.2 A Theoretical Construct for Generative Models in Multi-Agent Systems

Building on the formulation introduced in Chapter 1, we model the behavior of a multi-agent system through a trajectory distribution over states and actions. We extend the dynamics model in Equation 1.3.1 by introducing latent variables z, which encode underlying structure and hidden factors such as agent intent, beliefs, and equilibrium characteristics.

$$p_\pi(\tau \mid z) = P(s_0) \prod_{t=0}^{T} \left(\prod_{i=1}^{n} \pi_i(a_{i,t} \mid o_{i,t}, z_{i,t}) \right) \mathcal{T}(s_t, a_t, s_{t+1}), \tag{2.1}$$

where $\tau = \{s_0, a_0, \ldots, s_T\}$ denotes a trajectory.

The latent variables z play a central role in this framework. They act as structured representations that capture unobserved aspects of the multi-agent system, including strategic intent, uncertainty over other agents, and coordination patterns, and provide crucial signals for decision-making. As we will show in subsequent chapters, different modeling choices for z give rise to distinct algorithmic instantiations, ranging from learned latent representations to generative planning and language-based reasoning.

2.2.1 Latent Variables for Modeling Other Agents

A fundamental challenge in decentralized multi-agent systems is the need for each agent to reason about the behaviors, intentions, and beliefs of other agents under partial observability. In practice, agents do not have direct access to the policies or internal states of others, and must instead infer this information indirectly from observations and interaction histories.

Within the proposed framework, latent variables z provide a natural mechanism for capturing such hidden structure. Specifically, we may interpret z_i as encoding an agent's i internal belief over the joint environment, including approximations of other agents' strategies, preferences, and latent states. Formally, this corresponds to learning an approximate posterior over latent variables conditioned on past experience:

$$q_\phi(z_{i,t} \mid \tau_{0:t}), \tag{2.2}$$

where $\tau_{0:t}$ denotes the joint sequence of observations, actions, and rewards up to timestep t.

From this perspective, agents extend their understanding of the environment by maintaining beliefs over unobserved variables such as opponent types or hidden states through learned latent representations. These representations can encode complex dependencies between agents, including coordination patterns and interaction dynamics, enabling agents to take more informed and strategic actions.

In Chapter 3, representation learning methods are developed as mechanisms for approximate inference over the latent state, with the objective of constructing multi-agent latent spaces that are predictive of future interactions.

2.2.2 Latent Variables for Planning

Latent variables also play a central role in planning and forecasting future interactions. In complex multi-agent environments, directly optimizing over trajectories is often intractable due to the high dimensionality and stochasticity of the system. Generative models address this challenge by learning distributions over future trajectories conditioned on latent variables:

$$p_\theta(\tau_{t+1:T} \mid \tau_{0:t}, z_t). \tag{2.3}$$

In this setting, the latent variable z_t can be interpreted as encoding high-level structure over trajectories, such as goals, strategies, or coordination modes. Planning can then be performed by sampling candidate futures from the learned distribution and selecting actions that maximize expected return. This leads to a decision-making formulation of the form:

$$\pi(a_t \mid \tau_{0:t}, z_t) \propto \mathbb{E}_{\tau_{t+1:T} \sim p_\theta(\cdot \mid \tau_{0:t}, z_t)} \left[G(\tau_{0:T}) \right], \tag{2.4}$$

where $\tau_{0:T} = [\tau_{0:t}, \tau_{t+1:T}]$ and $G(\tau)$ denotes the return associated with trajectory τ.

This formulation highlights the role of generative models as implicit world models, enabling agents to simulate and evaluate possible futures. By conditioning on latent variables, the model can capture structured variations in behavior, such as different coordination strategies or responses to other agents. In multi-agent settings, this is particularly important, as the outcome of any trajectory depends on the joint actions of all agents.

The diffusion-based planning methods developed in Chapter 4 can be interpreted within this framework as learning and sampling from $p_\theta(\tau \mid z)$ via iterative denoising processes. These methods provide a practical instantiation of generative planning in high-dimensional multi-agent environments.

2.2.3 Latent Variables using Natural Language

Structured reasoning can be incorporated into the framework through expressive modalities such as natural language. Language models provide a way to represent and generate high-level reasoning processes, enabling agents to communicate intentions, coordinate actions, and decompose complex tasks using a shared mechanism.

Within the generative framework, natural language can be interpreted as a structured realization of latent variables. To capture this, we introduce an intermediate variable m and extend the model as:

$$p_\theta(\tau, z_t^i, m_t) = p(z)\, p_\theta(m_t \mid z_t^i)\, p_\theta(\tau \mid z_t^i, m_t), \tag{2.5}$$

where m_t represents language-based signals or reasoning traces.

Under this formulation, language serves as an intermediate representation that both reflects and influences the underlying latent structure. The distribution $p_\theta(m_t \mid z_t^i)$ captures how latent variables are expressed through natural language, while $p_\theta(\tau \mid z_t^i, m_t)$ models how these variables guide subsequent actions and trajectories.

This perspective provides a probabilistic interpretation of language-based coordination: agents express their strategies using language in order to align their latent variables and reduce uncertainty about one another. In practice, language models approximate these distributions implicitly, enabling agents to produce structured reasoning processes that inform decision-making.

The methods explored in Chapter 5 can be viewed as instantiations of this component, where prompt design and alignment techniques shape the mapping between latent intent and language. By grounding decision-making in natural language, agents can leverage shared symbolic representations to improve coordination and interpretability.

2.3 Conclusion

In this chapter, we introduced a unifying generative framework for decentralized multi-agent decision-making. By extending the trajectory-based formulation of multi-agent systems to include latent variables and structured generative processes, we provided a common foundation for understanding representation learning, planning, and reasoning within a single probabilistic framework.

Importantly, this framework is conceptual rather than prescriptive. It does not define a single implemented system, but instead offers a structured lens through which the contributions of this dissertation can be interpreted. Each subsequent chapter explores a different component of this framework: latent representation learning, generative trajectory modeling, and language-based reasoning.

This perspective suggests a broader research direction in which these components are integrated into unified architectures capable of jointly modeling latent structure, communication, and behavior. While such systems remain an open challenge, the framework presented here provides a foundation for their development, highlighting the role of generative modeling as a central paradigm for scalable and adaptive multi-agent decision-making.

Chapter 3

Representation Learning in Multi-agent Systems

In this chapter, we investigate how representation learning can improve the efficiency and stability of decentralized multi-agent reinforcement learning, resulting in the following paper [11]. Learning in multi-agent environments is particularly challenging because agents must reason under partial observability while simultaneously adapting to the evolving behaviors of other agents. These dynamics often lead to high variance in training and poor sample efficiency when using standard deep reinforcement learning methods. We argue that a key factor underlying these difficulties is the quality of the latent representations used by agents to interpret their observations and interactions. To address this challenge, we introduce a representation learning framework that explicitly structures the latent state space around multi-agent dynamics. By incorporating auxiliary objectives that capture transition structure, inter-agent relationships, and temporal consistency, the proposed framework encourages agents to learn richer and more informative representations of their environment. We demonstrate that these representations substantially improve the learning dynamics of several state-of-the-art multi-agent reinforcement learning algorithms across a variety of cooperative and competitive tasks.

3.1 Latent Representation Learning for Multi-agent Reinforcement Learning

3.1.1 Introduction

A multi-agent control system consists of multiple decision-making entities within a shared environment, each tasked with achieving objectives defined by a reward signal. Multi-agent reinforcement learning (MARL) offers a learning paradigm that optimizes for emergent rational behaviors within agents through interactions with the environment and one another to achieve desired strategies and solution concepts [28]. In recent years, deep MARL has proven successful in numerous domains, including robotics teams [48], networking applications [49], and various social scenarios that require multi-agent interactions [50]. However, deep reinforcement learning (RL) has historically suffered from sample inefficiency, requiring a costly amount of interaction experience to learn valuable behaviors. This challenge stems mainly from the inherent high variance in existing RL algorithms paired with the data-intensive nature of deep neural networks [4]. Unfortunately, MARL applications face additional learning pathologies and complexities [39] such as exponential computational scaling with respect to the number of agents and the dynamic challenge of equilibrium computation [51].

A natural solution to this issue is curating additional learning signals to supplement and enrich the RL learning process. This approach of regularizing and imposing prescriptive inductive bias has proven effective in prior works at enhancing the training of control policies in single-agent RL [52]. The new objectives that are introduced range from reinforcing similarities and dissimilarities within temporal or spatial locality [53, 54] to instilling information regarding different aspects of the tasks, such as the transition dynamics [55], into the latent state space. Notably, the main takeaway from these efforts is to learn a rich latent state space that is coherent with the task dynamics and itself [56]. However, many of these representation learning techniques have not been fully realized jointly nor extended to a MARL context.

In this work, we propose MAPO-LSO (Multi-Agent Policy Optimization with Latent Space Optimization) [11], a generalized MARL framework that leverages latent space optimization (LSO) in a multi-agent setting. We outline an overview of our proposed framework in Figure 3.1. Specifically, we show that popular MARL algorithms, such as MA-A2C [41],

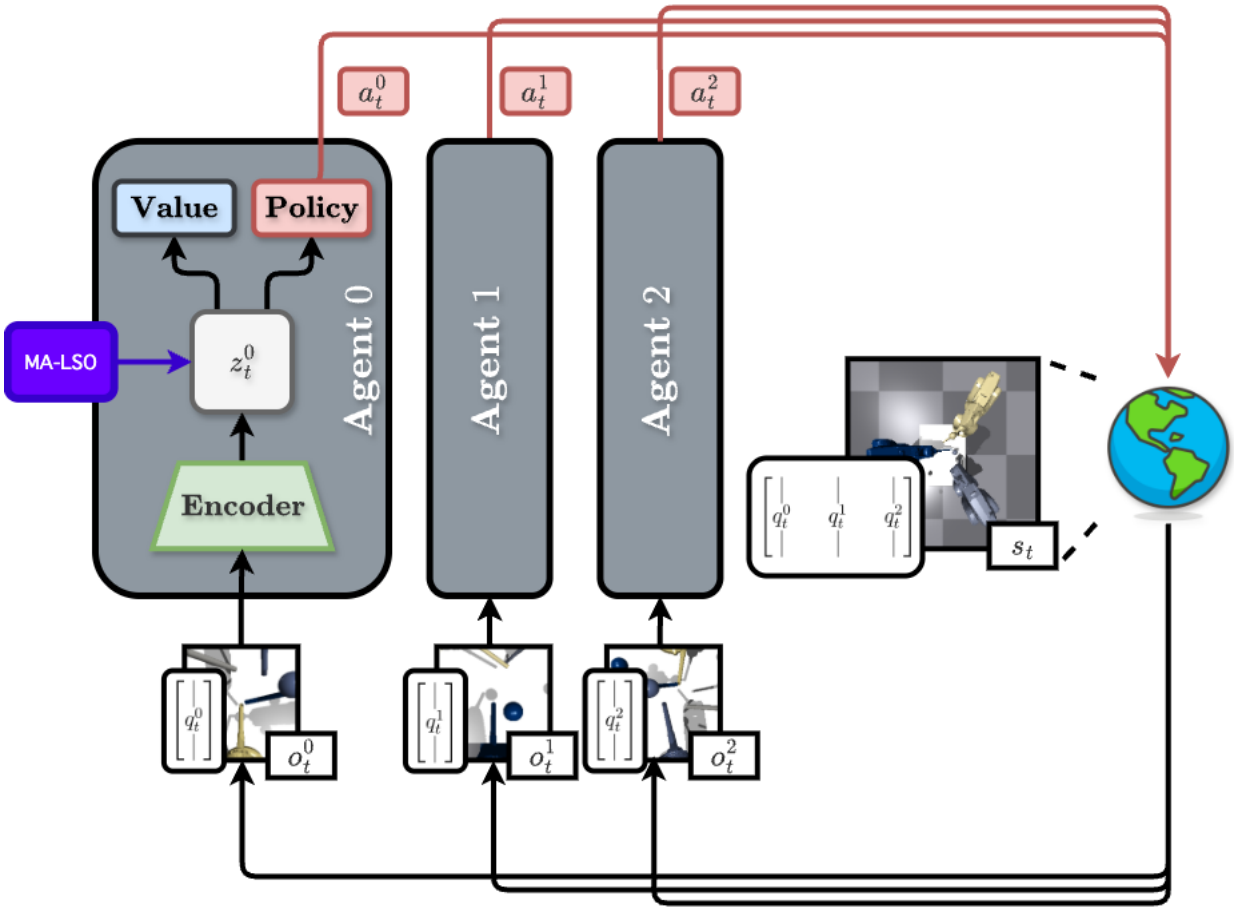


Figure 3.1: A high-level illustration of the MAPO-LSO framework applied on a decentralized setting, specifically, a multi-modal three-player robotic team task. We denote s_t as the state and o_t^i as the observation of agent i at time-step t. All observations consist of local visual inputs and local proprioceptive information q_t^i. The observations are processed in a distributed manner by each agent to compute the latent state z_t^i. Our proposed MA-LSO learning process is applied on z_t^i, which is used to compute the value and action a_t^i.

MAPPO [57], HAPPO [58], MASAC [59], and MADDPG [41] can all benefit from our multi-agent LSO (MA-LSO) learning process with trivial modifications. Our experiments demonstrate significant improvements in not only the sample efficiency but also in the overall performance over 17 diverse tasks in VMAS [15] and 24 multi-robotic arm scenarios in IsaacTeams [48].

Our contributions are as follows:

- We introduce a novel MARL framework, MAPO-LSO, that integrates MA-LSO, a comprehensive form of representation learning, into the MARL training. We provide

a new perspective on the intuition behind the usage and integration of new learning processes in a multi-agent control setting.

- We study various design choices to optimize the MAPO-LSO framework, applying pretraining, uncertainty-aware modeling techniques for agent-modeling and phasic optimization to improve learning performance, specifically regarding convergence and stability.

- We extend and experiment using several state-of-the-art MARL algorithms on our MAPO-LSO framework on a variety of multi-agent tasks with diverse nature of interactions, communication requirements, and multi-modal data, presenting further ablation studies on design choices to showcase the improvements of our MAPO-LSO framework.

3.1.2 Related Works

Sample-Efficiency in MARL A number of recent works have addressed the sample efficiency problem in deep MARL ranging from developing vectorized and parallelizable simulation platforms [15, 48], improving exploration strategies to collect diverse and useful samples [60], pre-training on a dataset of demonstrations [61], utilizing off-policy or model-based approaches [62], and learning on offline datasets [63]. While these prior efforts are not necessarily orthogonal to our efforts, this paper focuses on introducing a form of multi-agent representation learning that improves how much is learned from each sample by guiding the optimization of the latent state space for MARL tasks.

Representation Learning in MARL The concept of representation learning has previously been applied in MARL applications through masked observation reconstruction [64, 65], auxiliary task-specific predictions [66], self-predictive learning in joint latent space [67], and contrastive learning (CL) on the observation embedding space [68]. In our study, our proposed MA-LSO takes a more comprehensive measure by applying two forms of representation learning that enforce consistency between the latent state space and the transition dynamics and within itself as a self-predictive representation space.

3.1.3 Preliminaries

In this work, we consider an extension of the stochastic game framework [29] called the Bayesian game [28, 31].

Definition 1. *A Bayesian game is defined by a tuple* $\langle \mathcal{I}, \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, R, \omega, \mathcal{G} \rangle$.

- $\mathcal{I} = \{0, ..., N-1\}$ *is the set of* N *agents.*
- $\mathcal{S}$ *is the global state space.*
- $\mathcal{O} = \prod_{i \in \mathcal{I}} \mathcal{O}^i$ *is the joint observation space, where* $\mathcal{O}^i$ *is the observation space of agent* i.
- $\mathcal{A} = \prod_{i \in \mathcal{I}} \mathcal{A}^i$ *is the joint action space, where* $\mathcal{A}^i$ *is the action space of agent* i.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto P(\mathcal{S})$ *is the state transition operator, mapping the state-action space to the probability of the next states.*
- $R = \prod_{i \in \mathcal{I}} R^i$, *is the joint reward function, where* $R^i : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ *is the reward function for agent* i.
- $\omega = \prod_{i \in \mathcal{I}} \omega^i$ *is the joint belief space, where* ω^i *is the belief space of agent* i *that defines an agent's internal model.*
- $\mathcal{G} : \mathcal{S} \mapsto \mathcal{I} \times \mathcal{I}$ *is the mapping from the state to an adjacency matrix that defines the communication graph between all agents.*

We optimize for the Bayes-Nash equilibrium, where each agent i learns a best-response policy $\pi^i : \mathcal{O}^i \times \omega^i \mapsto \mathcal{A}^i$, by maximizing the expected *ex interm* return [30] of individual agent G^i, which constrains the objective of rationality to the agent's limited awareness of only their own belief space.

$$\forall i \in \mathcal{I}, G^i = \mathop{\mathbb{E}}_{\tau \sim \mathcal{T}_{\pi^i}} [\sum_{t=0} R^i(s_t, a_t)] \tag{3.1}$$

where $\tau = \{s_0, a_0, \dots s_T\}$ is a state-action trajectory.

Deep Reinforcement Learning The field of deep RL studies the general optimization algorithms using deep neural network approximations guided by reward signals: canonically existing in the form of Q-learning, policy gradient, and actor-critic methods [4]. These optimization schemes have been studied under a multi-agent context, demonstrating promising results [57].

Latent Space Optimization Latent space optimization (LSO) is a form of representational learning that is often used in unison with generative modeling [69], where LSO leverages model-based optimization that learns an approximation of the objective function under a learned latent space [70]. In RL, LSO is often used to map various features of the dynamics model into a latent space to assist with the RL training [71], aiding the agent to understand the environment it is in. In this work, we explore this pretense within a MARL setting.

3.1.4 Multi-agent Latent Space Optimization

Our proposed approach, MA-LSO, directly optimizes the latent state representation $z_t^i \in Z_t^i$ for each agent $i \in \mathcal{I}$ to supplement the learning signals of the MARL optimization by applying LSO to deep RL. To achieve this goal, we introduce two processes: MA-Transition Dynamics Reconstruction (MA-TDR) and MA-Self-Predictive Learning (MA-SPL). As illustrated in Figure 3.2, the objective of MA-TDR is to embed information from the dynamics of the environment into the latent state space, whereas MA-SPL enforces consistency within the latent state space, as shown in Figure 3.2. Drawing inspiration from previous work on single-agent model-based RL [71, 72] and representational learning methods for RL [55, 73, 74], we unify these two processes in a manner that complements one another while leveraging the multi-agent nature of the task.

3.1.4.1 MA-Transition Dynamics Reconstruction

MA-TDR learns an approximation of the transition dynamics of the environment with a mapping from a latent state representation space $\mathcal{Z}_t$ utilizing each agent's belief space, effectively grounding $\mathcal{Z}_t$ to the realities of the task's dynamics while taking into account a potential limitation of information each agent is privy to.

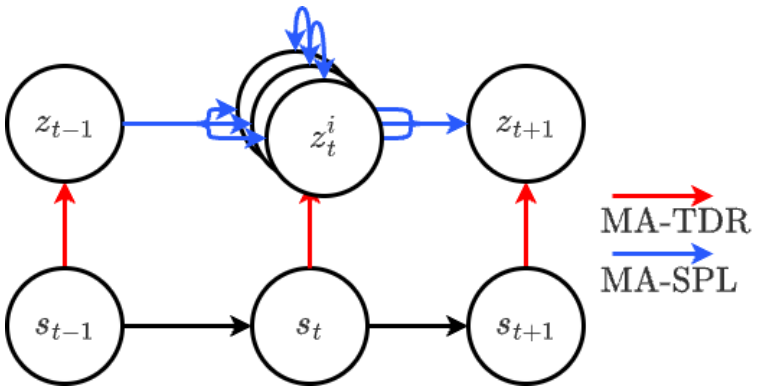


Figure 3.2: An illustration of the intended goals of MA-LSO. Given the state trajectory $\{s_{t-1}, s_t, s_{t+1}\}$ and the corresponding latent state trajectory $\{z_{t-1}, z_t, z_{t+1}\}$, MA-LSO aims to enforce consistency along different targeted dimensions, indicated by the arrows.

The latent state z_t captures information regarding the dynamics by reconstructing and inferring various aspects of the transition dynamics, namely, the transition τ_t which consists of the observation o_t, actions a_t, reward r_t and termination signal d_t. For a holistic understanding of the multi-agent dynamic, each agent i maintains a belief over all agents' transitions. The belief space $\omega^i(\cdot)$ of each agent i is a function of their own latent state z_t^i.

$$\omega^i(\tau_t^j) = \mathbb{E}[\text{Decoder}^{i,j}(z_t^i)] \tag{3.2}$$

To ensure the feasibility of the beliefs of other agents, we represent the Decoder with Bayesian neural networks, namely with Monte-Carlo dropout, to address the inherent epistemic uncertainty [75].

To further improve the solidify between the transition dynamics to the latent state dynamics, we incorporate M-CURL [76], a CL framework that guides the latent states $z^i_{0:T}$ produced by $o^i_{0:T}$ to be similar to the $\hat{z}^i_{0:T}$ produced by an augmented $\hat{o}^i_{0:T}$.

The overall loss for MA-TDR is defined as:

$$\mathcal{L}_{tdr} \doteq \mathop{\mathbb{E}}_{\tau_t \sim \mathcal{D}}[\sum_{i,j \in \mathcal{I}} \underbrace{\mathcal{L}_{cl}(\hat{z}^i_{0:T}, z^i_{0:T}))}_{\text{M-CURL loss}} \underbrace{- \ln P(\omega^i(\tau_t^j) = \tau_t^j)}_{\text{reconstruction loss}}] \tag{3.3}$$

where τ_t represents the transition, i.e., observations, actions, rewards, and termination, at time-step t, $\mathcal{L}_{cl}(\cdot)$ is the CL loss [53], and $\mathcal{D}$ is an experience replay buffer. The reconstruction loss is adapted from [43]. For each loss term, we append and tune scaling hyperparameters to each loss term to avoid dominating gradients and maximize performance.

3.1.4.2 MA-Self-Predictive Learning

MA-SPL learns a $\mathcal{Z}_t$ that is sufficient to predict the expected $\mathcal{Z}_{t+1}$ [77]. Intuitively, the learned latent state space should be consistent with its own latent dynamics [78]. Within a multi-agent control setting, we now must consider the presence of other agents in the environment and thereby enforce a structural relation and consistency not only along a temporal dimension [55] but also amongst the agents. To achieve this, we introduce three learning processes – MA-MLR, MA-FDM, and MA-IDM– to foster this consistency to the joint latent space.

MA-Masked Latent Reconstruction (MA-MLR) MA-MLR focuses on creating an inter-predictive representation between agents' latent space, a property reflective of the unique and inherent feature within multi-agent systems [64, 65]. We employ a masked prediction task, where the agents' latent states $z_t = \{z_t^0, z_t^1, \ldots, z_t^N\}$ is treated as a sequence [79], where the objective is to reconstruct the full sequence from a partially masked sequence, instilling coherency. Hence, if the latent state of agent i is masked $m_i(z_t) = \{z_t^0, \ldots, m(z_t^i), \ldots z_t^N\}$, the joint latent spaces of the other agents $z_t^{\mathcal{I}\setminus i}$ is sufficient to reconstruct z_t^i. We adopt the MA2CL framework [65], using a reconstruction model $\mathcal{R}(\cdot)$.

MA-Forward Dynamics Modeling (MA-FDM) The objective of MA-FDM is to ensure that the information contained in the current joint latent state z_t and the joint action a_t is sufficient to infer the next joint latent state z_{t+1} [67], enforcing forward temporal consistency. We define a transition head $\mathcal{T}_z(\cdot)$, which is realized using a cross-attention head that maps the joint latent state and action to the next joint latent space.

MA-Inverse Dynamics Modeling (MA-IDM) MA-IDM aims to achieve the following objective: Given the current and next joint latent state, the joint action that realizes the transition from the current to the next joint latent state can be deduced. We use an inverse head $\mathcal{I}(\cdot)$, implemented using a self-attentive model that maps the current and next joint latent state to the joint action space.

The overall loss for MA-SPL is defined as:

$$\mathcal{L}_{spl} \doteq \mathbb{E}_{\tau_{t,t+1}\sim\mathcal{D}}[\sum_{i\in\mathcal{I}} \underbrace{\mathcal{L}_{cl}(\mathcal{R}(m_i(z_t)), z_t)}_{\mathcal{L}_{\text{MA-MLR}}} \tag{3.4}$$

$$+ \underbrace{\text{MSE}(\mathcal{T}_z(z_t), z_{t+1})}_{\mathcal{L}_{\text{MA-FDM}}} + \underbrace{\text{MSE}(\mathcal{I}(z_t, z_{t+1}), a_t)}_{\mathcal{L}_{\text{MA-IDM}}}] \tag{3.5}$$

Similar to MA-TDR, we append a scaling hyperparameter to each loss term.

Lastly, we define the overall MA-LSO loss as:

$$\mathcal{L}_{lso} = \mathcal{L}_{tdr} + \mathcal{L}_{spl} \tag{3.6}$$

3.1.5 Integrating MA-LSO to Multi-agent Policy Optimization

In this section, we discuss how to extend the MA-LSO learning processes to popular MARL algorithms under various constraints and provide insight behind important design choices that we have experimented with.

Recurrent Models Partial-observability tasks limit informational access of each agent. To address this, we use gated recurrent models [80] to facilitate the feasibility of the agents' success of the provided tasks. We apply the MA-LSO objectives to such settings, by making the latent state a function that includes a recurrent state. We adopt the same procedure as R2D2 [81] with a fixed burn-in period of 3.

On-policy MARL In general, we define a replay buffer $\mathcal{D}$ that we sample batches of transitions from to compute both MA-LSO and MARL objectives. Importantly, we found that learning on diverse data during the MA-LSO process is necessary to promote good generalization and stable learning, which is more easily achieved using an offline dataset. For on-policy algorithms, we maintain both on-policy and off-policy data in $\mathcal{D}$ such that only online data is used for the on-policy MARL training, such as MA-A2C, and off-policy data is still available for the MA-LSO process.

Phasic Optimization For certain MARL algorithms, namely MADDPG and MASAC, we chose to follow the training methodology outlined in TD3 [82]. This includes the utilization of target networks and delayed policy updates. These techniques are intended to mitigate the learning variance to enhance performance. However, we observed that the training

of such MARL algorithms was too sensitive to hyperparameters, likely due to the shared parameters between the policy and value function and the phasic nature of learning [83]. To mitigate this instability, we incorporated a phasic regularization term to constrain the policy divergence during all non-policy updates to promote a more stable learning process. For MARL algorithms with sequential policy updates like HAPPO, we extend this regularization term so that the output of the shared parameters does not cause large divergence from the other agents' behaviors that are not being updated.

Pre-training We pre-train our agents on the MA-LSO objective similar to SGI-R [84], where $\mathcal{D}$ is pre-filled with transitions from a random policy and is trained until convergence. This practice proved to further improve sample efficiency.

Model-based Training Lastly, we extend our MAPO-LSO solution to a Dyna-like training paradigm [85], where the policy is optimized using trajectories generated from our learned world model. We construct a decentralized extension of the Dream2Control framework [71] by introducing a state-agnostic stream optimized using a KL-balancing term with our MA-LSO optimized latent state.

Parameter Sharing To avoid large scaling w.r.t. the number of agents within MA-TDR and its reconstruction, the parameters of the proposed decoder can be shared between agents, using the common MARL practice of appending the agent type, or its identification number, to the input [40]. We also define a centralized value function that uses the joint latent state as input.

Model Architecture Details To compute the latent state z_t^i, we adopt the model architecture from DreamerV3 [86], utilizing a recurrent multi-modal auto-encoder. For certain tasks, communication between agents is required. In those tasks, we use the attention-based communication block proposed for MAMBA [87], which propagates communicative information between agents prior to computing the latent state. We note that the auxiliary heads introduced for MA-TDR (i.e. Decoder) and MA-SPL (i.e. $\mathcal{R}, \mathcal{T}_\ddagger, \mathcal{I}$) are only used during training, and is not used during inference to compute action nor value.

3.1.6 Experiments

For our experiment, we evaluate our results using Vectorized Multi-agent Simulator (VMAS) and IsaacTeams (IST) to provide a comprehensive evaluation of a diverse collection of multi-agent tasks, selecting 17 unique tasks from VMAS and 24 tasks from IST.

Experimental Setup The scenarios parameters for VMAS and IST environments were taken from prior works [15, 48, 88] that evaluate challenging social scenarios requiring a broad array of cooperative and adversarial considerations and varying complexities, with some tasks involving multi-modal observations and sparse reward signals. For a comprehensive list and description of all VMAS and IST tasks and their specific configurations, we encourage readers to refer to the original works. We provide a short description of the tasks in Table 3.1.

For our experiments, we evaluate using five MARL algorithms: MA-A2C, MAPPO [57], HAPPO [58], MASAC [59], and MADDPG [41]; all of which are considered competitive MARL baselines. For all experiments, the MARL hyperparameters on the MARL algorithms without MA-LSO, denoted as MAPO, are initially tuned based on prior MARL works complemented with a random search for all tasks. While keeping those hyperparameters fixed, we trained with our MAPO-LSO extension on each task. The hyperparameters of the MA-LSO learning process were tuned separately. During execution, we note the model architecture between MAPO and MAPO-LSO remains the same. All experiments presented were executed on 3 Nvidia RTX A6000 and Intel Xeon Silver 4214R @ 2.4GHz.

3.1.7 Results

In this section, we evaluate the performance and sample efficiency of MAPO-LSO paired with popular MARL algorithms. In our experiments, we refer to performance as the collective return achieved, and the sample efficiency is measured by the performance with respect to the number of data samples used, meaning the better the performance at a given number of environment transitions learned on, the higher the sample efficiency. We conduct further ablation studies to investigate and analyze each component of our MAPO-LSO method and study their synergy, recognizing whether any other improvements or degradations are realized at a more granular level.

To provide a concise comparison against our method over the many algorithms and tasks we evaluated, we present our results on a normalized scale [89, 90]. The normalized

IsaacTeams MAReacher	
Objective	Move end-effectors to target location.
Heterogeneity	Actors are composed of a team of ABB arms or hetergenous team of ABB, KUKA and Franka arms, all with unique observation and action spaces.
Observation space	Actors uses image + proprioceptive observation; otherwise only proprioceptive with additional state information.
Nature of interactions	Cooperative; rewards formatted as an aggregative game. Mixed goals; each agent's reward is independent. Adversarial; closest agent to single target receives positive reward.
Communication	Full communication among all agents. Limited communication with periodic decimation. No communication between agents.
VMAS Tasks	
discovery	Agents explore environment to discover new areas or objects.
dispersion	Agents spread out in the environment.
dropout	Agents handle situations where agents/resources are lost.
give_way	Agents cooperate by yielding space or resources.
navigation	Agents navigate complex environments to reach goals.
sampling	Agents collect data or items based on criteria.
simple_adversary	Basic competitive adversarial environment.
simple_crypto	Cryptography task involving secure information exchange.
simple_push	Agents push objects to predefined locations.
simple_reference	Agents use reference points to complete tasks.
simple_speaker_listener	Communication task with speaker and listener agents.
simple_spread	Agents spread out while minimizing overlap.
simple_tag	Tag game where agents attempt to catch others.
simple_world_comm	Agents share environmental/state information.
transport	Agents move objects from one location to another.
wheel	Circular or wheeled motion task, potentially with obstacles.
wind_flocking	Flocking behavior influenced by environmental factors (e.g., wind).

Table 3.1: Short Descriptions of IsaacTeams MAReacher and VMAS Tasks

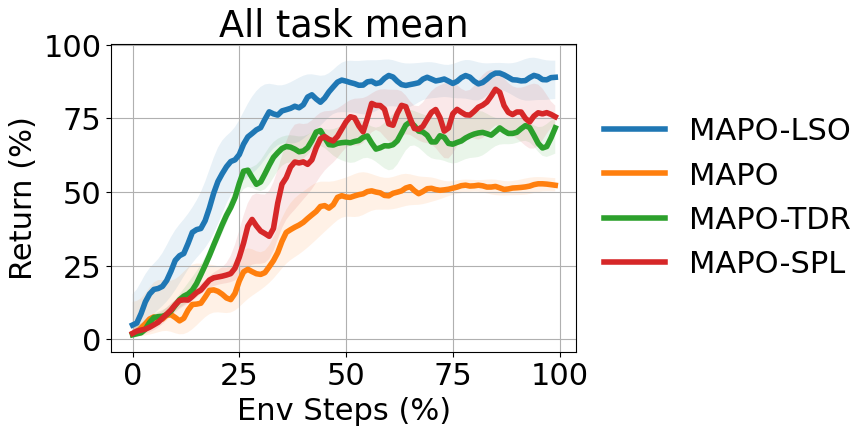
Algorithm	Description
MAPPO	Multi-Agent Proximal Policy Optimization (MAPPO) [57] is a CTDE extension of PPO that employs decentralized policies with centralized value functions. In our implementation, we follow the original paper but use a centralized critic shared between all agents.
HAPPO	Heterogenous-Agent Proximal Policy Optimization [58] refines MAPPO by imposing a random-order sequential-update scheme to ensure monotonic improvements without assuming homogeneous agents. Our implementation follows the original work, with the main difference being a shared encoder between the policy and value function. To ensure more stable learning due to the shared encoder, we update the value function upon each agent update and reduce the encoder learning rate.
MADDPG	Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [41] is a CTDE extension of DDPG. Our implementation follows [82], incorporating delayed policy updates, target policy smoothing, clipped double learning, stochastic actors, and a shared critic among all agents.
MASAC	Multi-Agent Soft Actor-Critic (MASAC) is a CTDE extension of SAC [59]. Our implementation is similar to MADDPG, with additional adjustments for automatically tuned entropy maximization.

Table 3.2: Descriptions of Multi-Agent Reinforcement Learning Algorithms

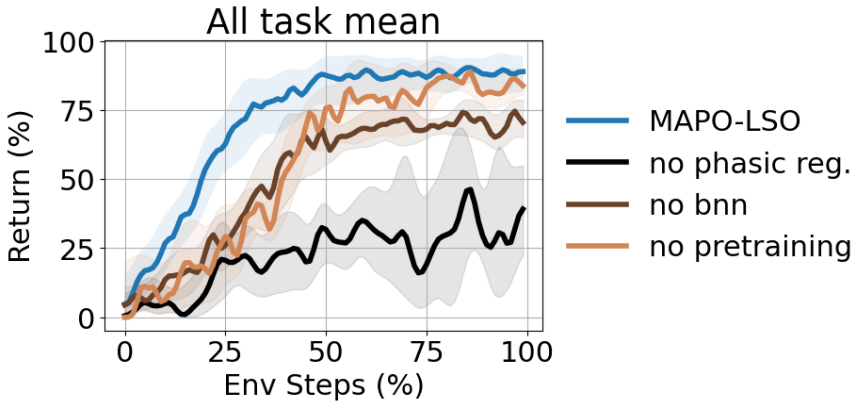
scale involves expressing the return and the number of environment steps as percentages, as different tasks operate on different scales, and this computation is performed on a per-task basis.

Efficacy of MAPO-LSO As depicted in Figure 3.3a, our MAPO-LSO framework demonstrates a significant performance improvement, reaching a $+\mathbf{33.51}\%$ difference in collective return from the baseline algorithms. Our MAPO-LSO achieved the max performance of the MAPO baseline in $\times\mathbf{4.17}$ less samples. We find that both MAPO-TDR and MAPO-SPL, which uses only MA-TDR and MA-SPL objectives respectively, outperform the baseline but fall short of their combined counterpart, indicating their own utilities by themselves and together.

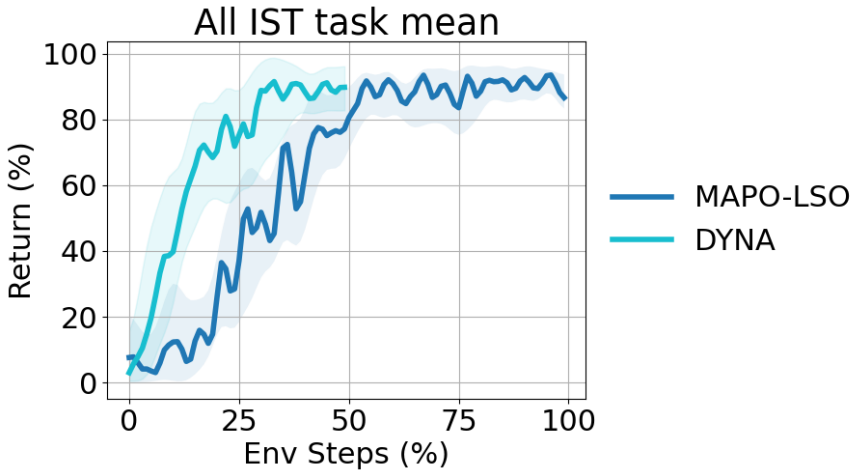
Learning in Imagination To further improve sample efficiency, we incorporate a Dyna-like training, which trains the policy and value function on "imagined" trajectories generated by our model. Our evaluation of this approach, as shown in Figure 3.3c, demonstrates a notable improvement in sample efficiency, requiring $\times\mathbf{1.59}$ fewer samples to achieve nearly



(a) MAPO-LSO Results



(b) Design Choices



(c) Dyna-like Training

Figure 3.3: In these graphs, the normalized collective returns are compared between variants of our proposed solution. We additionally present the effects of different design choices, such as phasic regularization, Bayesian network, pretraining and Dyna-like training.

the same max convergence as our original MAPO-LSO. We remark that this approach was much more sensitive to the MAPO-LSO hyperparameters.

3.1.7.1 Design Choices in MAPO-LSO

Phasic Optimization We confirm our hypothesis stated in Section 3.1.5 with Figure 3.3c, where we found training instability without phasic regularization on both baseline and our MAPO-LSO. While we attempted to tune the hyperparameters of the experiments with no phasic regularization, we found those runs to be highly brittle to hyperparameters and largely dependent on their implementation, notably the design and scale of the model architecture. This concern is not novel, and in our work, we addressed this instability through phasic regularization with comforting results.

Pre-training We investigated using the MA-LSO objective as a pre-training process. We first collected a dataset of 1000 trajectories using a random policy and pre-trained the model on the MA-LSO objectives until convergence. As shown in Figure 3.3c, with all hyperparameters fixed for both runs, we find that including pre-training provides notable improvement within the sample efficiency and stability with a more negligible performance improvement.

Learning the Belief Space The use of Bayesian networks to represent the belief space demonstrated a slight improvement in the performance of the learned policy, as shown in Figure 3.3c, and perhaps more importantly, the accuracy of the belief space, as shown in Figure 3.4. The belief space of agents with no communication (NC) benefited significantly more using this uncertainty modeling technique. Additionally, in Figure 3.5, we visualize the belief space of the Visual-ABB3-C-LC task, where even with sparse communication that limits agents to communicate every 4 time-steps, they were not only able to infer important properties regarding the other agents' observations with just background cues, where in $\omega^0(o_2)$, the agent predicts the relative position of the yellow agent with the background shadow, but also be able to correct their beliefs upon communication, as seen in $\omega^0(o_1)$, where the belief of the grey agent significantly improves after communication. While communication is not necessarily the focus of this work, this result affirms the utility of communication to lower the uncertainty of the belief space.

Lastly, we evaluated the imagined policies realized within each agent's belief space from MAPO-LSO training on collaborative tasks in IST by rolling out trajectories using the

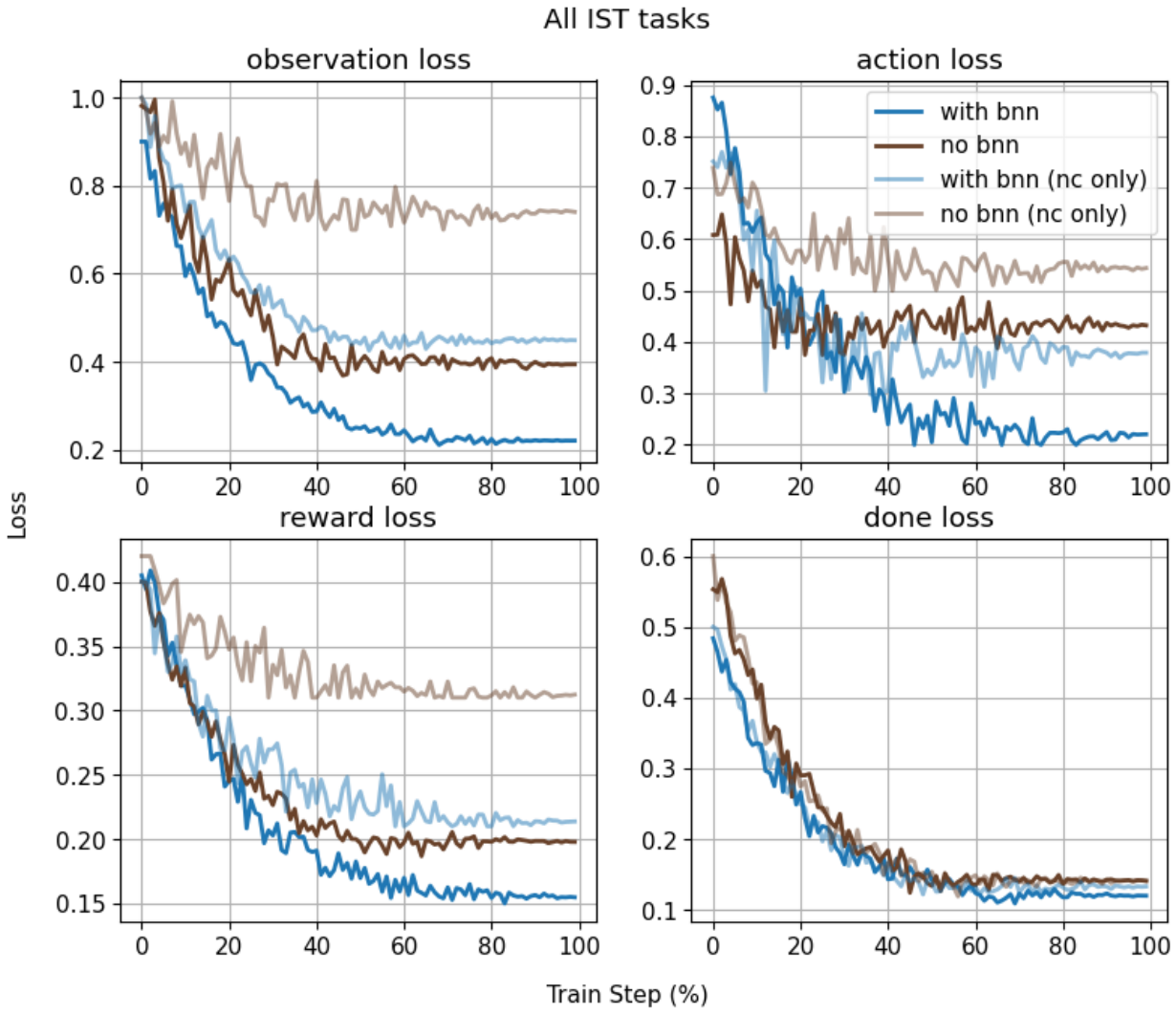


Figure 3.4: Accuracy of Belief Space of Other Agents, comparing the use of Bayesian network representation for our Decoder and the inclusion of communication.

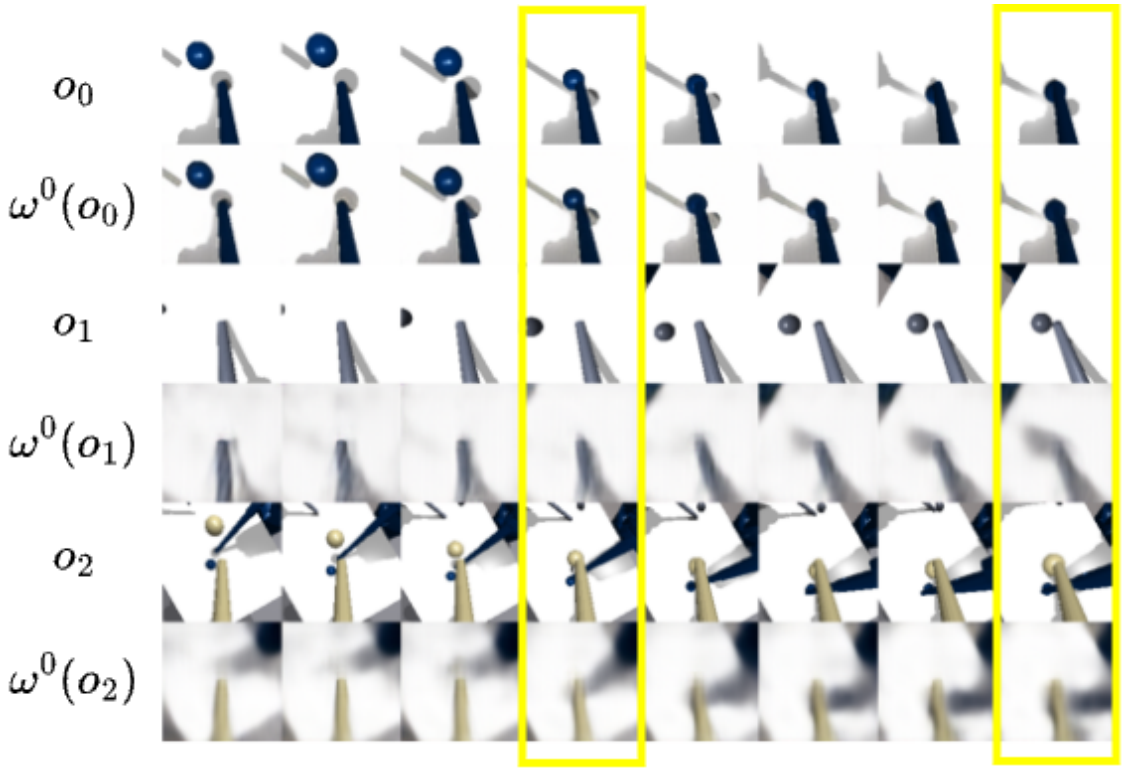


Figure 3.5: A visualization of agent 0's observational belief space $\omega^i(o)$ alongside the ground truth observations of all agents in Visual-ABB3-C-LC task, where the yellow boxes indicate communication steps.

joint actions within the agent's belief space, e.g., $\pi = \omega^i(\pi)$. In Table 3.4, we find that the performance of the imagined policies, on average over all agent's belief space, depends on the communication paradigm, the agents' homogeneity, and the observation space's complexity. With no communication, it is unlikely that the imagined policies will be grounded in reality and fail to perform comparatively. Impressively, even with limited communication, we see decent performance from the imagined policies but generally with higher uncertainty.

3.1.7.2 Ablation Studies

Ad-Hoc Team-Play A looming concern with MAPO-LSO is whether these objectives would cause greater over-reliance on specific teammates or arbitrary social conventions. Hence, we evaluate the ad-hoc performance, i.e., when trained agents are paired with unseen teammates using cross-play. For these experiments, we measure the relative performance between the original team T against a team T' with one agent that is placed with other

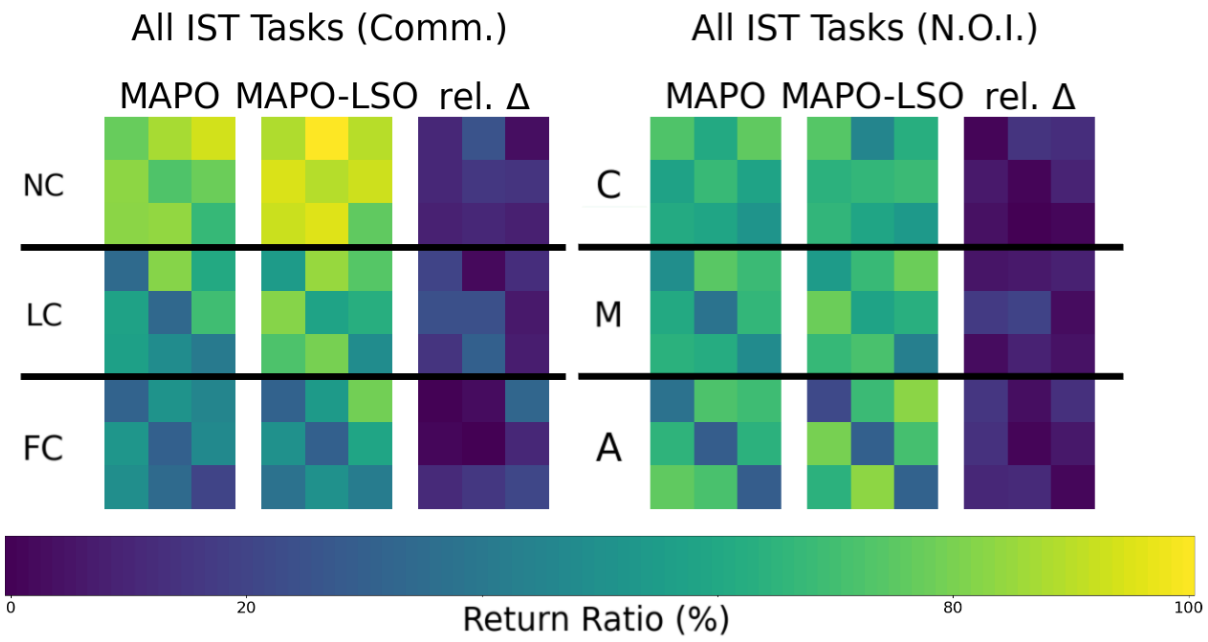


Figure 3.6: The average return ratio on all IST tasks and MARL algorithms, divided in the axis of communication (NC: no communication, LC: limited communication, FC: full communication) and nature of interaction (N.O.I) (C: collaborative, M: mixed, A: adversarial) on 32 random seed.

agents trained on different MARL algorithms using the metric of return ratio.

$$\text{Return ratio} = \frac{\sum_{i \in T'} G^i}{\sum_{i \in T} G^i} \tag{3.7}$$

In Figure 3.6, under a zero-shot coordination framework, we found that the relative difference in return ratio (i.e. rel. Δ) between MAPO and MAPO-LSO was insignificant, with factors such as communication and the nature of interactions creating a more cogent correlation.

MA-LSO Components We assess the effectiveness of each component within our MA-LSO framework by conducting evaluations that include omissions of the MA-TDR and MA-SPL processes. For MA-SPL, we exclude its sub-processes individually: MA-MLR, MA-FDM, and MA-IDM. A key contribution of this work is the integration of these auxiliary objectives and their symbiotic relationship, which Table 3.3 confirms. Hence, the results demonstrate that all of our MA-LSO framework's components contribute to the demonstrated improvements and are interdependent. Moreover, excluding any processes within MA-LSO results in a notable decline in the convergence of other processes. This decline sug-

Table 3.3: Empirical losses and success rate of MA-LSO, with its respective ±1 std deviation trained over all IST tasks.

Variants	$\mathcal{L}_{tdr}$	$\mathcal{L}_{spl}$			Success Rate (%)
		$\mathcal{L}_{\text{MA-MLR}}$	$\mathcal{L}_{\text{MA-FDM}}$	$\mathcal{L}_{\text{MA-IDM}}$	
MA-LSO	1.06 ± 0.31	0.145 ± 0.021	0.341 ± 0.109	0.385 ± 0.223	76.25 ± 7.50
no MA-TDR	-	0.258 ± 0.019	0.492 ± 0.208	0.530 ± 0.292	59.375 ± 14.38
no M-CURL	1.35 ± 0.12	0.198 ± 0.041	0.409 ± 0.051	0.492 ± 0.304	68.025 ± 5.63
no MA-MLR	2.10 ± 0.22	-	0.464 ± 0.194	0.612 ± 0.310	55.625 ± 6.88
no MA-FDM	1.91 ± 0.31	0.210 ± 0.092	-	0.700 ± 0.220	56.875 ± 11.25
no MA-IDM	2.23 ± 0.26	0.332 ± 0.048	0.593 ± 0.214	-	54.375 ± 8.75
no MA-SPL	3.14 ± 0.19	-	-	-	51.25 ± 5.63
MAPO	-	-	-	-	45.625 ± 11.88

Table 3.4: Average success rate with ±1 std deviation on collaborative tasks in IST over all MARL algorithms on 32 evaluation seeds, comparing the true policy and the policy learned within the belief space of all agents.

Tasks	Success Rate (%)	
	π	$\omega(\pi)$
ABB3-C-FC	93.75 ± 1.46	89.79 ± 5.00
ABB3-C-LC	87.70 ± 3.13	85.63 ± 6.25
ABB3-C-NC	90.21 ± 4.38	64.38 ± 11.88
Mixed3-C-FC	90.63 ± 1.88	73.54 ± 9.38
Mixed3-C-LC	87.70 ± 2.50	67.50 ± 8.125
Mixed3-C-NC	88.13 ± 1.88	40.63 ± 9.17
Visual-ABB3-C-FC	81.88 ± 4.38	51.25 ± 11.25
Visual-ABB3-C-LC	83.96 ± 3.75	42.50 ± 10.63
Visual-ABB3-C-NC	69.38 ± 5.00	35.20 ± 12.08
Visual-Mixed3-C-FC	74.38 ± 3.75	49.38 ± 11.25
Visual-Mixed3-C-LC	69.38 ± 9.38	34.38 ± 8.13
Visual-Mixed3-C-NC	63.13 ± 8.13	20.63 ± 4.38

gests a form of amortization similar to that observed in multi-task applications [91], evident from the difference in performance of each component.

MARL Algorithms We provide a more granular view of the effectiveness of MA-LSO on individual MARL algorithms in Figure 3.7. We found all algorithms to benefit, mostly with their sample efficiency and convergence points. The difference between the normalized return (Return Δ) stabilizes after around 40% of the total environment steps for all algorithms, peaking within the earlier range of $20 - 30\%$. This trend affirms our claims of improved

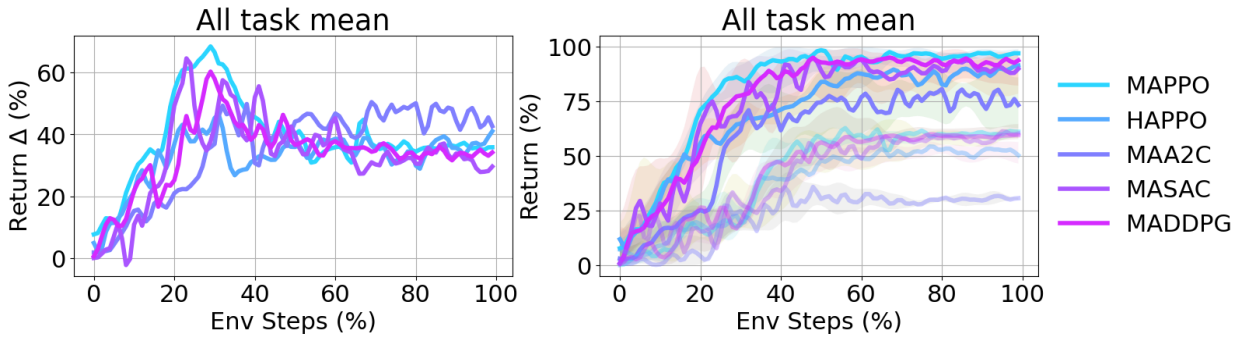


Figure 3.7: Performances of individual MARL algorithms, with and without MA-LSO.

sample efficiency, as earlier stages of training is capable of leveraging each sample more to learn more rational strategies.

3.1.8 Sensitivity Analysis

We evaluate the robustness of MAPO-LSO on the ABB3 cooperative task under full communication (ABB3-C-FC) trained using MAPPO by introducing controlled perturbations to observations, rewards, latent representations, and agent availability. Across all perturbations, we scale noise magnitudes relative to empirical statistics of the corresponding signal to ensure comparability and realistic perturbation regimes.

Observation Noise Sensitivity To evaluate robustness to observational uncertainty, we inject Gaussian noise into the observation space:

$$\tilde{o}_t = o_t + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_o^2). \tag{3.8}$$

In practice, we vary σ_o to simulate sensor noise along each joint. We scale the noise relative to the joint limits, setting σ_o to up to 5% of the total joint range for each degree of freedom.

Reward Perturbation To evaluate sensitivity to reward specification, we introduce noise into the reward signal:

$$\tilde{r}_t = r_t + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_r^2). \tag{3.9}$$

To compute σ_r, we estimate the empirical standard deviation of rewards from trajectories collected using both random and trained agents, and scale the perturbation up to 5% of this value. This ensures that the perturbation magnitude reflects the intrinsic variability of the reward signal rather than its absolute scale.

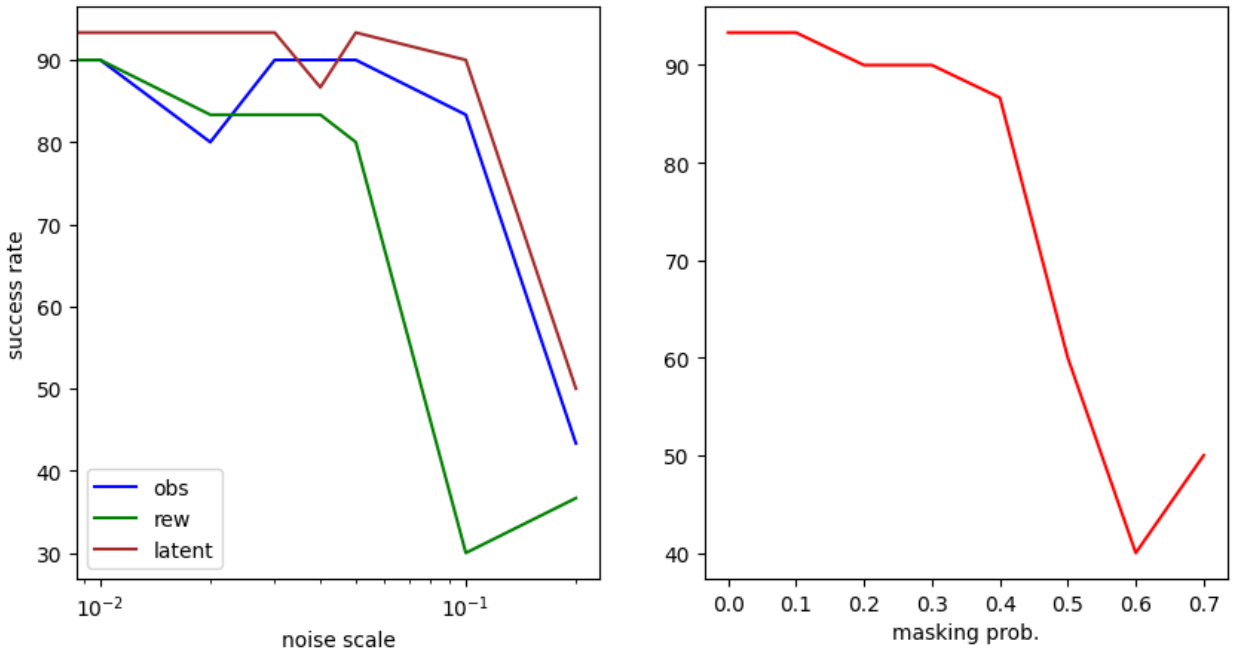


Figure 3.8: Sensitivity Analysis with MAPO-LSO trained agents for ABB3-C-FC from MAPPO over 30 trials. We scale the perturbations and noise for observations, rewards and latent states on the left figure, and scale the masking probability of communication on the right.

Latent Space Sensitivity To assess the robustness of the learned latent representations, we perturb the latent state during execution:

$$\tilde{z}_t = z_t + \delta, \quad \delta \sim \mathcal{N}(0, \sigma_z^2). \tag{3.10}$$

We evaluate how performance degrades as a function of the noise σ_z, providing insight into the stability and smoothness of the learned latent space. To compute σ_z, we estimate the average ℓ_2 norm of latent representations and scale the perturbation proportionally, ensuring that the injected noise is relative to the magnitude of the latent space.

Agent Dropout and Partial Observability We evaluate robustness to missing information by randomly masking communication between agents during execution with some probability. This simulates failures in communication or sensing and tests the ability of the latent representations to generalize under incomplete information.

Across all perturbations, MAPO-LSO demonstrates robustness under moderate noise levels. In particular, performance degrades gracefully with increasing observation and latent

perturbations, indicating that the learned representations capture stable structure. Sensitivity to reward perturbations is comparatively higher, reflecting the direct role of reward signals in policy optimization. Additionally, the model remains resilient under partial communication, suggesting that the learned latent space supports inference under incomplete information. These results support the interpretation of the latent variable z as a structured and stable representation.

3.1.9 Limitations and Future Directions

While MAPO-LSO demonstrates consistent improvements in sample efficiency, stability, and performance across a diverse set of multi-agent tasks, several limitations should be noted, alongside opportunities for future work.

Computational Overhead The introduction of multiple auxiliary objectives, including MA-TDR and MA-SPL, increases the computational cost of training. These objectives require additional forward and backward passes through auxiliary heads, as well as the maintenance of replay buffers for diverse data sampling. Although these costs may be partially amortized through improved sample efficiency, the overall wall-clock training time and memory footprint remain higher compared to baseline MARL methods. This trade-off may limit applicability in resource-constrained settings. Future work may explore more efficient multi-objective training schemes [92] to reduce overhead.

Hyperparameter Sensitivity The MAPO-LSO framework introduces additional hyperparameters, including weighting coefficients for auxiliary losses and architectural design choices for latent modeling components. While we observe that reasonably tuned configurations generalize well across tasks, an initial search is required to identify a stable operating regime. This increases the complexity of deployment compared to simpler MARL baselines. Developing adaptive or self-tuning mechanisms for balancing auxiliary objectives presents a promising direction for improving usability and robustness.

Dependence on Observation and Communication Structure The effectiveness of the learned latent representations depends on the richness of the observation space and the availability of communication between agents. As observed in our experiments, limited or absent communication leads to increased uncertainty in the learned belief space and reduced

performance in imagined rollouts. This suggests that MAPO-LSO may be less effective in environments with severely constrained information flow. However, agents trained under richer communication settings exhibit some robustness to observation noise and communication masking at test time, indicating partial generalization beyond the training regime. Future work may investigate more explicit mechanisms for handling partial observability, such as improved belief modeling or communication-efficient protocols.

Reward Sensitivity Our sensitivity analysis indicates that the framework remains sensitive to perturbations in the reward signal. Since reward directly drives policy optimization, noise or misspecification can propagate through both the policy and representation learning processes. This reflects a broader limitation of reinforcement learning methods and suggests that MAPO-LSO does not inherently address challenges related to reward design. Incorporating techniques such as reward learning, inverse reinforcement learning, or robustness-aware optimization [5] may help mitigate this sensitivity in future work.

3.1.10 Conclusion

We introduce a generalized MARL training paradigm, MAPO-LSO, that utilizes auxiliary learning objectives to enrich the MARL learning process with multi-agent transition-dynamics reconstruction and self-predictive learning. Our approach demonstrated significant improvements upon its "non-LSO" counterpart in a wide variety of MARL benchmark tasks using several state-of-the-art MARL algorithms. For future directions, there remain promising avenues to study other aspects of the multi-agent nature of MARL tasks, such as communication, and its impact on the latent state representation.

3.2 Conclusion

In this chapter, we introduced MAPO-LSO, a generalized training framework that integrates multi-agent latent space optimization into deep multi-agent reinforcement learning, and enriches the latent representations used by decentralized agents with structural information about environment dynamics, temporal consistency, and inter-agent relationships. Our empirical analysis shows that each component of the proposed framework contributes to im-

proved learning dynamics, and that the integration of these objectives provides synergistic benefits for training stability and convergence.

More broadly, this work highlights the importance of structured representation learning in multi-agent decision-making. By explicitly shaping the latent state space to reflect the dynamics and interactions of multi-agent environments, agents are better equipped to interpret limited observations, anticipate the behavior of other agents, and learn effective policies from fewer samples. This perspective aligns with the broader theme of this dissertation: leveraging generative and representation learning paradigms to enhance the reasoning and planning capabilities of decentralized multi-agent systems.

Chapter 4

Generative Models for Multi-agent Planning

In this chapter, we investigate the use of diffusion-based generative models for sequential decision-making and planning. Diffusion models have recently emerged as powerful generative models capable of representing complex trajectory distributions, making them promising candidates for control and reinforcement learning problems [20]. We first study how diffusion models can be aligned with reward signals to produce trajectories that maximize planning objectives, resulting the following paper [17]. Building on this foundation, we then apply these techniques to a challenging real-world domain: large-scale online advertising auctions. In this setting, we develop a latent graph diffusion framework that captures the multi-agent structure of auction environments and enables strategic auto-bidding optimization, resulting in the following paper [18].

4.1 Aligning Diffusion Models for Decision-Making

4.1.1 Introduction

Denoising diffusion probabilistic models (DDPM) have demonstrated incredible successes in numerous domains as one of the de facto standards for generative modeling in continuous domains [45, 93, 94], garnering heightened attention in research communities with the model's exceptional ability to capture highly complex data distributions and flexibility in constraint-based optimization that can be applied to diverse modalities. A notable

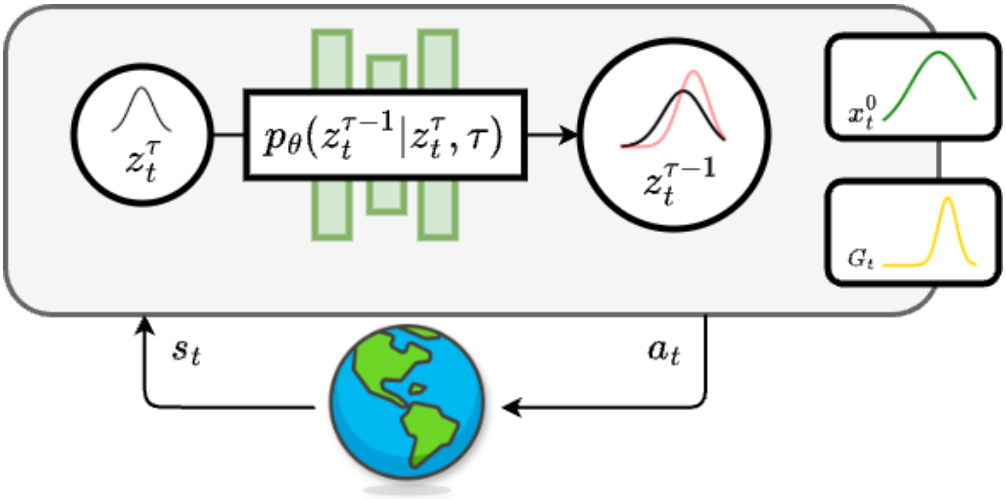


Figure 4.1: A illustration of the reward maximization of diffusion model $p_\theta(\cdot)$ for control, visualizing the diffusion process at denoising steps $\tau \to \tau - 1$ such that $z_t^{\tau-1}$ is denoised towards the target distribution x_t^0 but also aligned with maximize the return distribution G_t. The resulting trajectory distribution that is more aligned with the return distribution is shown in red.

direction is their adoption and realizations in decision-making applications [95], contributing to an evolving class of model-based planning as well as model-free algorithms we will refer to as diffusion-based planning, learning, and control. Despite the appealing promises of DDPMs, the algorithm's sample generation inefficiencies loom as a notorious and significant concern [96], causing a severe bottleneck as an efficient solution for closed-loop control.

The focus of this work addresses this limitation, proposing a general investigation into reward maximization, or more aptly called reward alignment, directly on the diffusion process for control [17]. In turn, our objective is to reduce the *number of samples* required to generate to maximize the reward signal. We adopt modern optimization procedures to facilitate a guided and constrained distributional shift biased towards maximal returns. Specifically, we investigate four reward alignment approaches — reward maximization with reinforcement learning, direct preference optimization, supervised fine-tuning, and cascading diffusion — and compare their performance in various control settings, more so in offline RL settings. We evaluate various design choices as well as an unification of the studied alignment approaches on a diverse testbed of challenging control tasks consisting of a variety of complex tasks from benchmarks such as D4RL [97], MetaWorld [98], PushT [99], Relay-Kitchen [100], and RoboMimic [101] as well as a novel evaluation task that enables in-depth analysis and

visualization of the various features of the environment, the controller and their interactions.

4.1.2 Background

Diffusion Models DDPMs [45] are commonly known as a class of generative models, which aim to estimate and sample from unknown data distributions X. DDPMs define a Markov chain of forward Gaussian kernels $q(x_t|x_{t-1})$ with a scheduled [102] sequence of variance coefficients $\{\beta_t \in (0,1)\}_{t=1}^{T}$ that gradually add noise to input data

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathrm{I}) \tag{4.1}$$

and then learn to reverse the diffusion process by denoising with learnable reverse Gaussian kernels $p_\theta(z_{t-1}|z_t)$, parameterized by θ, to generate data samples z_0 from a standard Gaussian noise prior $z_T \sim \mathcal{N}(0, I)$.

$$p_\theta(z_{t-1}|z_t) = \mathcal{N}(z_{t-1}; \mu_\theta(z_t, t), \Sigma(z_t, t)) \tag{4.2}$$

A common practice is to parameterize $\mu_\theta(z_t, t)$ to perform $\epsilon-$prediction, motivated by its resemblance to the Langevin dynamics, which enables a simplification of the training objective.

$$\mu_\theta(z_t, t) \coloneqq \frac{1}{\sqrt{\alpha_t}}\Big(z_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(z_t, t)\Big) \tag{4.3}$$

The training objective of DDPM uses the variational lower bounds of the marginal likelihood to minimize the negative log-likelihood [103], as shown in Equation 4.4, where in practice, a simplified yet equivalent objective $\mathcal{L}_{\mathrm{DDPM}}(\theta)$ is commonly used [45].

$$\mathcal{L}_{\mathrm{NLL}}(\theta) = \mathbb{E}[-\log p_\theta(z_0)] \leq -\mathbb{E}_q[\log \frac{p_\theta(z_{0:T})}{q(x_{1:T}|x_0)}] \tag{4.4}$$

$$\mathcal{L}_{\mathrm{DDPM}}(\theta) = \mathbb{E}_{x_0}\Big[\|\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t)\|^2\Big] \tag{4.5}$$

where $t \sim [0, T]$ with T is the total number of diffusion steps, $\beta_t = 1 - \alpha_t$, $\bar{\alpha}_t = \Pi_{i=1}^{t}\alpha_i$, and $\epsilon_t \sim \mathcal{N}(0, \mathrm{I})$. A common and useful intuition is that DDPMs learn information regarding a time-dependent vector field $u_t(z_t, \theta)$ that generates a probability density/diffusion path $p_t(z_{t-1}|z_t)$ from a simple and known distribution (e.g., standard Gaussian) to the unknown target distribution (e.g., data distribution).

Several recent improvements and reformulations of this DDPM paradigm have been explored [96], most notably and relevantly, towards efficient sample generation with implicit

probabilistic models (DDIM [46]), $v-$prediction [104] and rectification [105, 106], conditional guidance [107, 108], model selection [109–111], novel DDPM training paradigms [112, 113] and related classes of generative modeling such as score/flow-matching [114, 115].

Alignment of Diffusion Paths Training deep learning models, such as a DDPM, is often done in two stages. In the first stage, the foundation model is optimized, where a general underlying data distribution is learned by training on large-scale datasets. In the second stage, this foundation model is fine-tuned to align to a reference distribution (e.g., towards downstream tasks, human preference, or stitching with multiple distributions) through supervised fine-tuning (SFT) [116], RL with AI/human feedback (RLAIF/RLHF) [117, 118], preference optimization (DPO) [119–121], or reward gradients [122, 123]. In this work, we apply and modify these techniques toward DMCs to better align using a scalar feedback signal that defines the reward distribution for sequential decision-making tasks.

Diffusion Models for Control (DMC) Like many generative modeling algorithms, the role of DDPMs in decision-making applications has spanned from diffusion-based planning [20] to policy learning [124], with significant successes and advancements in online and more prevalently in offline RL settings. Specifically, its utilization has been quite diverse, taking various forms such as diffusion-based policies, value functions, and world models, and these models can either indirectly support the training of RL policies [125] or be more of an active component of the decision-making loop [126, 127]. In this work, we extend these prior efforts, specifically Diffuser [20], Diffusion Policy (DP) [128], SynthER [129] and some of their following works [130, 131] to act as our foundation to build upon with our alignment efforts.

4.1.3 Reward Alignment of DMC

We define a general equation that defines the reward alignment optimization problem, in which we align our diffusion model π_θ to generate samples z_0 that maximize a reward function.

$$\mathcal{L}_{\text{align}}(\theta) := \underbrace{\mathbb{E}_{z_0 \sim \pi_\theta}[r(z_0)]}_{\text{Reward Maximization}} \quad \text{s.t.} \quad \underbrace{\text{div}(\pi_\theta, X) \leq \delta}_{\text{Divergence Constraint}} \tag{4.6}$$

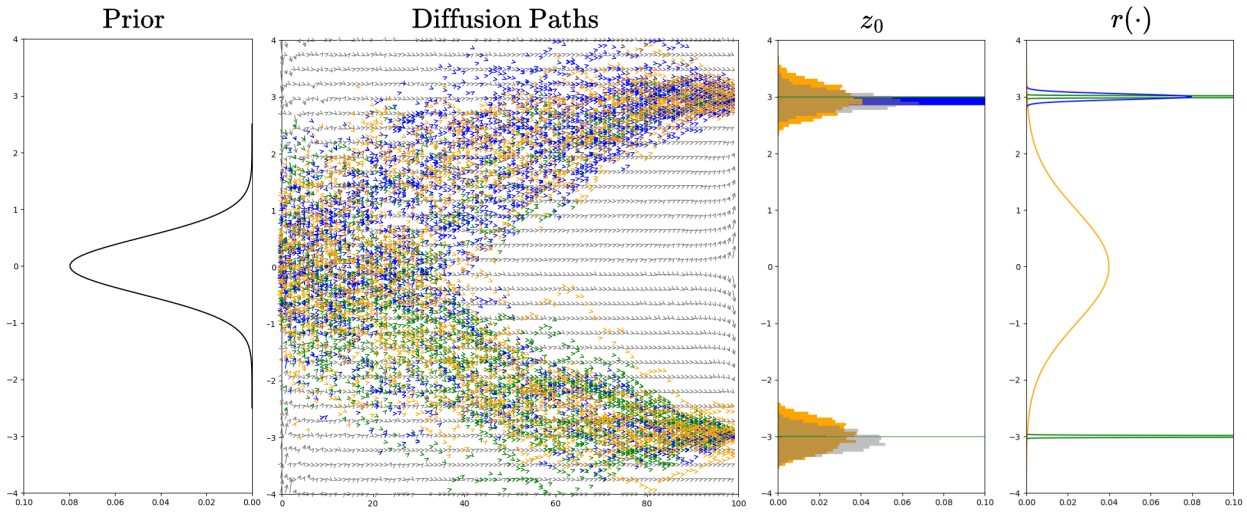
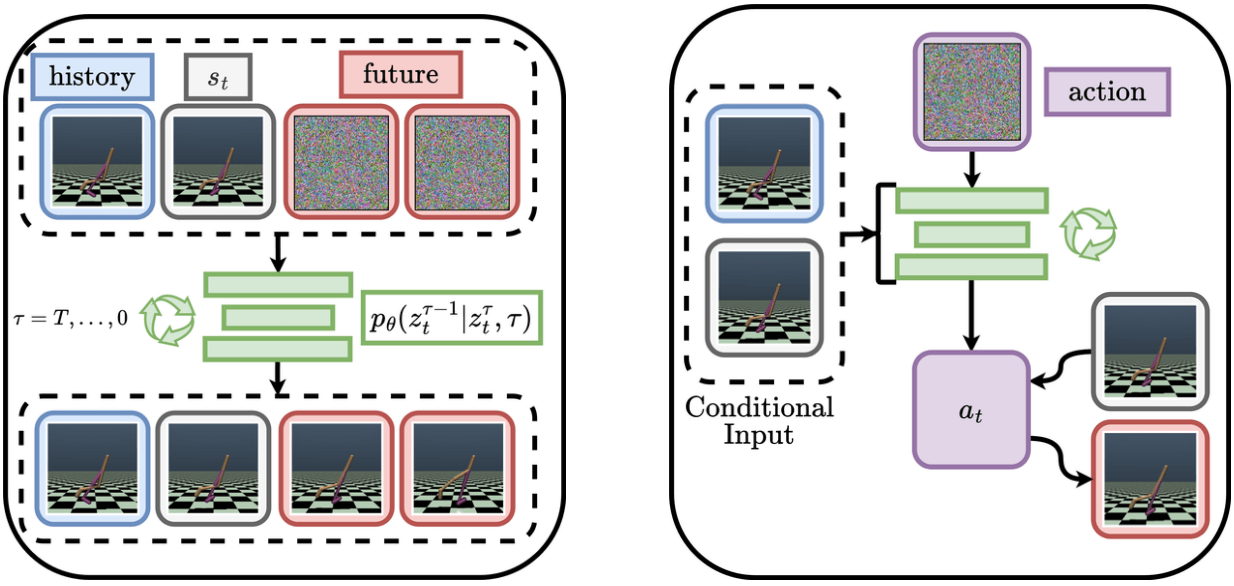


Figure 4.2: A visualization of the effects of alignment to the diffusion process. Given a foundation DDPM, whose outputs and diffusion field over the entire state space are shown in gray, we align its diffusion process to maximize three separate reward functions, shown in orange, green and blue.

The constraint subjects the optimization to ensure the generated samples remain within a plausible region of the target data distribution X, or the distribution learned by the foundation DDPM π_{fdt}.

We illustrate this objective in Figure 4.2, where a foundation DDPM is fine-tuned to maximize a given reward function on a toy task. In this example, the fine-tuned diffusion paths largely adhere to the foundation model, with minor yet impactful adjustments towards maximizing their respective reward functions, exceeding $\times 3.21$, $\times 7.09$, and $\times 4.18$ improvements in orange, green and blue respectively within 32 generated samples while maintaining $\Pr(z_0|X) \gg 0$. Specifically, with the orange reward function, despite having the potential to converge towards out-of-distribution (OOD) regions for maximal rewards, the constraints are in place to adequately address this issue in a desired manner. Hence, to achieve this alignment, we study and extend four approaches – RL, DPO, SFT, and cascading – to optimize the objective in Equation 4.6.

An important distinction between DDPM and DMC is that, at denoising step t, DMC specifies that z_t is either a $N-$step state-action trajectory $\{s_0, a_0, s_1 \dots s_N\}^t$ (e.g. Diffuser and DD), or actions $a_{t'}^t$ (e.g. DP) with a conditional information regarding the state $s_{t'}$ as shown in Figure 4.3. Hence, our reward function $r(z_0)$ can either represent the return of the state-action trajectory or the Q-value of each state-action pair for greater granularity as a



(a) Planning-based DMC

(b) Policy-based DMC

Figure 4.3: A visual illustration of the two DMC frameworks on Walker2D task, where the planning-based DMC is forecasting future states and the policy-based DMC generates actions directly.

way to take advantage of the temporal aspect of the latent representation.

4.1.3.1 Reward Alignment with Reinforcement Learning

A natural approach to reward maximization is to extend traditional RL optimization to align the denoising process towards sample generation with higher returns. To do so, we view the DMC as a denoising policy

$$\pi_\theta(\mu_t|z_t, c) = p_\theta(z_t|z_{t-1}, c) \tag{4.7}$$

with a condition input c, treating the denoising process as a multi-step MDP [113]. We build upon two classic RL algorithms that directly optimize Equation 4.6: REINFORCE [35] and Q-value policy gradient (QV-PG) [59, 132].

$$\mathcal{L}_{\text{REINFORCE}}(\theta) = \mathbb{E}[\sum_{t=0}^{T} \kappa_t sg(\log \pi_\theta(\mu_t|z_t), K)\, r(z_0)] \tag{4.8}$$

$$\mathcal{L}_{\text{QV-PG}}(\theta) = \mathbb{E}[\kappa_t r(sg(\pi_\theta(\mu_{0:T}|z_{0:T}), K))] \tag{4.9}$$

To account for the sequential nature of the denoising process, we consider two important training concerns: gradient truncation and denoising credit assignment.

Table 4.1: **A Summary of Alignment Method Comparison:** Reward terms and divergence control mechanisms.

Method	Reward Maximization Term	Divergence Control
RL	Policy gradient (Eqs. 4.8, 4.9)	KL regularization/PPO clipping
DPO	Preference ranking via $F_w - F_l$	Regularization via ϵ_{ref}
SFT	Fine-tuning on high reward samples	Likelihood threshold using π_{ref}
Cascading	Conditional generation $p(x_0^r \vert x_0^c)$	Warm-starting + adherence to reference policy.

- Gradient truncation clips the gradient such its calculation only propagates through the last K steps [122, 123], where K can be dynamically set during training. This truncation implements truncated backpropagation through time/denoising steps, facilitating a more memory-efficient and stable optimization. In practice, we utilize the stop-gradient function $sg(\cdot, K)$, which truncates the gradient computation to the last K denoising steps.

- As each sampling step contributed differently towards reaching z_0, we consider the concept of credit assignment over the denoising steps. Empirically, we define a sequence of hyperparameters $\{\kappa_0, \kappa_1, \dots\}$ that scale the objective along the denoising steps. In this work, we consider two options, where κ can be a decreasing sequence or κ_t can be the similarity measure between (z_t, z_{t-1}) and (z_T, z_0), a computation that is done retrospectively.

To prevent large divergence from the reference policy/foundation DMC and adhere to the constraint in Equation 4.6, we can use trust regions with clipping, i.e., proximal policy optimization (PPO) [133], similar to the procedure introduced with DDPO [113], or add a KL regularization term. We found both REINFORCE/PPO and QV-PG approaches to be competitive, to which their comparison seems moot, and the one that performed better is kept during evaluation.

4.1.3.2 Reward Alignment with Preference Optimization

Direct preference optimization (DPO) [119] offers an alternative approach to align DMC by utilizing a dataset of ranked pairs $x_0^l \prec x_0^w$ of losing and winning samples and maximizing the likelihoods of the DMC's sample generation following the preference operator $\prec$. This

preference operation is defined in this work as follows:

$$x_0^l \prec x_0^w \implies \mathbb{E}[r(x_0^l)] < \mathbb{E}[r(x_0^w)] \tag{4.10}$$

Following DiffusionDPO adaptation [120], the DPO objective $\mathcal{L}_{\text{DPO}}(\theta)$ is simplified and can be expressed as follows:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}[\log \sigma(-\gamma\omega(\lambda_t)(F_w(\theta) - F_l(\theta)))] \tag{4.11}$$

where $\sigma(\cdot)$ is the sigmoid function, $\lambda = \alpha_t^2/\sigma_t^2$ is the signal-to-noise ratio, $\omega(\lambda_t)$ is a weighting function and

$$F_*(\theta) = \|\epsilon^* - \epsilon_\theta(z_t^*, t)\|_2^2 - \|\epsilon^* - \epsilon_{\text{ref}}(z_t^*, t)\|_2^2 \tag{4.12}$$

A functional representation of the preference operator can be expressed either heuristically or, more generally, as a trajectory-wise value function [134].

4.1.3.3 Reward Alignment with Supervised Fine-Tuning

Similar to DPO, we can instead fine-tune the foundation DMC only using the winning samples, or samples with the highest rewards, by updating along $\mathcal{L}_{\text{DDPM}}(\theta)$ on these samples. Therefore, we apply traditional diffusion updates to shift and concentrate the generative distribution towards higher rewards under a stable optimization scheme, i.e., supervised learning with rejection sampling [135].

This process can be performed iteratively, using either the foundation or the fine-tuned DMC as a data synthesizer to generate samples of increasingly higher returns. However, if we use the fine-tuned DMC, we face potential instability and OOD issues. In this work, we prevent this using a threshold constraint that prevents samples with lower likelihoods of the foundation DMC from being used for an SFT update.

4.1.3.4 Up-sampling Reward with Cascading Diffusion

Cascading DDPM [110] is an up-sampling technique used for super-resolution, where lower resolution samples are processed as the conditional inputs to sample higher resolution samples. This concept can be extended to control applications, where a DMC that is conditioned on x_0^{cond} is trained to generate a sample x_0^r that yields greater rewards, hence $r(x_0^{\text{cond}}) < r(x_0^r)$.

In prior efforts with similar motivations [134], the objective followed classifier-free guidance (CFG), training with the conditional information at some probability $p \in [0, 1]$. Since, in our case, the cascading DMC is connected to a foundation DMC, it does not necessarily have to emphasize the generation of the target distribution; we set $p = 1$. Transferring the trained weights from a foundation DMC as a warm-start proved beneficial and was likely sufficient to understand the target distribution.

An important consideration for training cascading DMC is determining the pairs $(x_0^{\text{cond}}, x_0^r)$ of samples to condition and reference during training, where their alignment may be mismatched, especially for multi-step samples. To address this, when curating $(x_0^{\text{cond}}, x_0^r)$, we promote consistency between the pairs. This consistency can be achieved by in-painting a contiguous subset of the sample or sharing the latent state z_t at a denoising step t. These approaches force some level of relations between the two samples, such that the cascading DMC avoids up-sampling towards random samples during inference.

To further boost the up-sampling capability of the cascading DMC, we can extend the other alignment approaches to greaten the difference between $r(x_0^{\text{cond}}), r(x_0^r)$ with proper modifications. Our exploration of this study is limited, and we encourage future works to research this topic further.

4.1.3.5 An Unification of Reward Alignment Approaches

The four alignment methods discussed above—RL, DPO, SFT, and cascading—share a unified goal of solving the constrained optimization problem defined by Equation 4.6. While each approach is suited for different use cases, with their unique advantages and limitations, we find that they can be applied together. As the cost of increasing model alignment efforts is modest relative to training the reference policy [118], we propose a sequential optimization scheme, where each alignment approach (i.e., RL, DPO, and SFT) is used individually, and repeatedly until convergence. This iterative sequential scheme is practiced to avoid potential catastrophic learning variances of a naive multi-objective optimization approach. We find that after $2 - 3$ passes of this sequence, there are diminishing returns in the reward maximization efforts. Again, the cascading module is trained separately and appended onto the fine-tuned model during inference. Similarly, we experimented with performing multiple passes through the cascading module but found high variances in the up-sampling after 2 passes.

4.1.4 Design and Training Choices

Diffusion Sampling and Process We apply the proposed alignment methods to widespread advancements to the classical DDPM formulation. For instance, accelerated sampling methods such as DDIM [45] and other representations of the iterative diffusion-like posterior, such as score-matching (SM) [46] and flow-matching (FM) [115], are also studied in this work under the Diffuser, DD and DP framework utilizing these alignment approaches.

In-painting The in-painting ability of DDPMs refers to reconstructing missing regions within the data based on the context provided. In our alignment efforts, we rely on this capability to bootstrap training samples to enforce consistency and avoid undesired changes to the sample. To ensure that our model can in-paint effectively, we trained the foundation DMC directly on the in-painting task, where we utilize causal masks along the temporal axis following the Repaint procedure for conditioning [136]. This in-painting procedure is followed during inference for planning-based DMC. In our experiments, we found that including this masked training improved not only the DMC's ability to in-paint but also the overall performance of the DMC.

Context Window and Recurrence The context window for the DMC defines the number of time steps contained in input for planners and conditional input for the policy. Under specific model architectures, such as a temporal convolution backbone, this window does not face significant computing constraints. However, we opt for a sliding window approach to handle this constraint for backbones that scale poorly along this dimension and for tasks with longer time horizons. We found the introduction of a recurrent state appended to the latent state to cause instability in some of the tasks, and even in certain tasks with less complex observations with greater partial observability, we found minor but negligible improvements.

Evaluation Function For all alignment methods, we use a separate Q-value function trained under implicit Q-learning in an offline RL fashion to evaluate the samples generated and calculate the learning signal. The Q-value can also be used during action-selection.

Online Fine-tuning We additionally study the inclusion of online fine-tuning, where we inject samples from online interactions during the alignment training to compare and analyze the changes in performance.

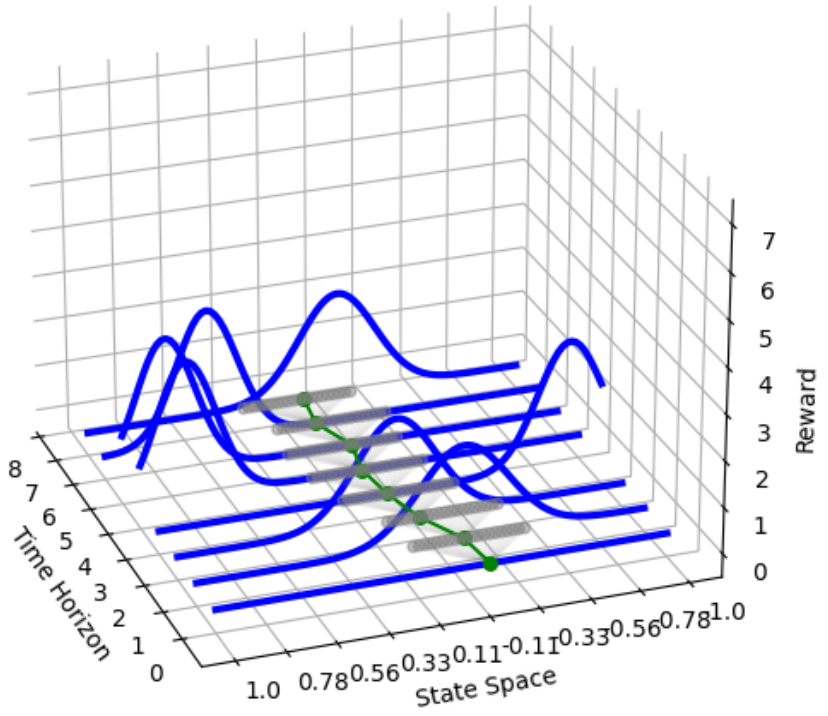


Figure 4.4: Visualization of the Nav1D task and an example of an agent's trajectory, shown in green, and the actions space is shown in gray. The reward distributions at every time step are shown in blue.

PEFT For all alignment processes, we use LORA [137], a parameter-efficient fine-tuning approach that introduces a separate set of low-rank weights on linear layers that are trained during fine-tuning. Separate LORA weights were not introduced between different alignment methods.

4.1.5 Experiments

4.1.5.1 Tasks

We evaluate our alignment approaches on various offline RL benchmarks. We use D4RL and MetaWorld for planning-based DMC; for policy-based DMC, we use Robomimic, Relay-Kitchen, and PushT. There is a standardized and publicly available offline RL dataset for many of these tasks, and we direct readers toward their respective original works for more details.

Table 4.2: Evaluation Results of Planning-based DMC (Diffuser)

Dataset	**Environment**	DDPM	DDIM	SM	FM
Random	HalfCheetah	8.0 → 12.7	5.5 → 11.3	5.4 → 11.5	10.2 → 15.9
	Hopper	0.5 → 4.2	0.4 → 1.1	0.2 → 3.0	0.3 → 3.9
	Walker2d	3.2 → 9.1	4.4 → 10.1	3.6 → 5.9	3.4 → 7.2
Medium-Expert	HalfCheetah	85.2 → 97.8	77.8 → 94.1	74.3 → 94.3	90.0 → 102.1
	Hopper	52.5 → 107.2	50.2 → 100.1	66.2 → 111.2	54.8 → 113.2
	Walker2d	76.5 → 102.4	48.4 → 108.3	65.1 → 109.0	49.1 → 99.4
Medium	HalfCheetah	30.6 → 48.3	23.3 → 50.6	32.1 → 54.2	35.1 → 50.2
	Hopper	52.9 → 89.5	61.2 → 93.5	49.0 → 82.0	44.2 → 96.3
	Walker2d	35.3 → 76.6	29.6 → 61.8	36.1 → 78.0	33.4 → 89.2
Medium-Replay	HalfCheetah	28.6 → 36.4	22.9 → 34.1	14.1 → 26.5	26.8 → 37.9
	Hopper	54.1 → 91.8	49.2 → 94.8	55.3 → 77.9	58.5 → 89.4
	Walker2d	26.0 → 58.6	45.6 → 67.5	46.2 → 59.1	50.5 → 62.7
Mixed	Franka-Kitchen	41.5 → 56.5	52.0 → 55.9	32.5 → 54.2	45.7 → 52.0
Partial	Franka-Kitchen	38.0 → 40.1	33.5 → 34.2	36.1 → 41.2	36.3 → 50.1
Play	Antmaze-Medium	0.2 → 46.7	0.0 → 45.3	0.9 → 84.0	3.2 → 80.8
	Antmaze-Large	0.0 → 57.3	0.0 → 60.2	3.5 → 76.1	0.7 → 54.9
Diverse	Antmaze-Medium	0.8 → 42.0	4.0 → 75.3	10.1 → 62.3	11.2 → 92.5
	Antmaze-Large	0.0 → 47.3	0.0 → 50.6	8.9 → 50.3	0.0 → 70.1
MetaWorld	Button-Press	0.0 → 0.7	0.1 → 1.0	0.0 → 0.5	0.0 → 0.7
	Drawer-Open	0.2 → 0.9	0.1 → 0.9	0.2 → 0.7	0.4 → 0.6
	Sweep-Into	0.3 → 0.7	0.0 → 0.8	0.5 → 0.8	0.3 → 0.9
	Plate-Slide	0.2 → 0.6	0.1 → 0.4	0.2 → 0.7	0.4 → 0.7
Nav1D	Partial	2.1 → 4.3	1.9 → 3.3	2.5 → 4.1	2.3 → 5.2
	Full	3.2 → 6.9	4.3 → 7.3	2.9 → 6.9	3.9 → 6.8
	Expert	6.2 → 7.3	5.9 → 7.2	5.9 → 7.2	6.4 → 7.1

Table 4.3: Evaluation Results of Planning-based DD (Decision Diffuser).

Dataset	**Environment**	DDPM
Random	HalfCheetah	8.4 → 13.5
	Hopper	0.5 → 1.8
	Walker2d	3.1 → 6.9
Medium-Expert	HalfCheetah	86.2 → 93.5
	Hopper	53.1 → 110.9
	Walker2d	67.1 → 101.1
Medium	HalfCheetah	29.0 → 44.5
	Hopper	37.5 → 90.2
	Walker2d	29.1 → 50.4
Medium-Replay	HalfCheetah	25.7 → 36.4
	Hopper	45.8 → 70.5
	Walker2d	41.1 → 52.4
Mixed	Franka-Kitchen	44.5 → 60.3
Partial	Franka-Kitchen	32.7 → 44.9
Play	Antmaze-Medium	0.0 → 32.1
	Antmaze-Large	0.0 → 85.1
Diverse	Antmaze-Medium	2.4 → 63.2
	Antmaze-Large	0.8 → 41.3
MetaWorld	Button-Press	0.1 → 0.5
	Drawer-Open	0.3 → 0.9
	Sweep-Into	0.2 → 1.0
	Plate-Slide	0.1 → 0.6
Nav1D	Partial	2.0 → 3.9
	Full	3.3 → 6.9
	Expert	6.2 → 7.2

Nav1D We also introduce simple 1D navigation (Nav1D) task to our testbed, shown in Figure 4.4, where the agent start at a random position $s_t \in [-1, 1]$ and must move left or right. The agent must select actions under a continuous yet restricted action space with a deterministic transition dynamics

$$\mathcal{T}(s_t, a_t) = \mathbb{I}(\text{clip}(s_t + \gamma a_t, -1, 1)) \tag{4.13}$$

where γ is the step size. The objective is to maximize the rewards defined by a sequence of independent normal distribution $\{N_t\}_{t=0}^{H}$, where

$$r(s_t, a_t) \propto \Pr(T(s_t, a_t) = N_{t+1}) \tag{4.14}$$

over a fixed time horizon H. The reward distributions are independent from one another and stationary upon repeated trials. The agent's observation space consists of its current location and additional information that gives the agent a better time-dependent context.

We create three datasets for Nav1D that provide varying state coverages—partial, full, and expert—where both partial and full datasets follow a random policy, and the expert dataset contains trajectories that follow an expert policy.

Coherency Score We introduce a coherency score C to evaluate the sample quality of planning-based DMC,

$$C = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \pi_\theta}[\Pr(s_{t+1} = \mathcal{T}(s_t, a_t)] \tag{4.15}$$

and of policy-based solutions,

$$C = \mathbb{E}_{a_t \sim \pi_\theta}[\Pr(a_t = \pi_\beta(a_t | s_t)] \tag{4.16}$$

The coherency score measures how well the planning methods capture the dynamics model distribution, whereas, for the policy-based DMC, it measures how well the DMC models the behavior policy π_β.

4.1.5.2 Implementation Details

The basis of our implementation is built upon CleanDiffuser [138], with adjustments made for the alignment methods. A central difference in our implementation of the DMC is the lack of a return conditional input, as it is generally unknown what return is desired. Hence, for action selection, we use the learned Q-Value function to greedily select actions,

similar to IDQL [139]. For DP, we also experiment with expectation-based action selection, where the average over all generated action samples is taken. The model architecture for each algorithm we studied follows the original work, similar to CleanDiffuser. All experiments presented were executed on 3 Nvidia RTX A6000 and Intel Xeon Silver 4214R @ 2.4GHz.

4.1.5.3 Results For Planning-based DMC

We summarize our finding for planning-based DMC in Table **??**, showing the notable improvements from our proposed alignment process, regardless of the sampling approaches such as DDPM, DDIM, SM, FM, and DD frameworks, with decision-making that uses a small number of roll-outs during its planning. Specifically, for each task and algorithm, we first trained the reference DMC and finetuned it using varying passes and iterations through {RL, DPO, SFT} alignments.

We closely examined the learning for these training to ensure no training collapses occurred, with model check-pointing and lowered learning rates. This phenomenon includes sudden drops in return and instabilities/spikes in the training objectives. We did observe some instances of this, especially during the RL finetuning, which we redacted. We trained the cascading DDPM separately, applying the same alignment process, and attached it for the final evaluation. In general, we found that even with multiple rounds, the return did improve, as shown in Figure 4.5.

An important trend we noticed as a result of the alignment is that, on average, over all tasks and algorithms, the average standard deviation over the return on all trials significantly decreased, from 38.1% $\rightarrow$ 5.3%. This result empirically indicates the distributional shift that occurs, which narrows the diffusion process towards higher returns, as the average return improved by +43.0%. Using only the D4RL tasks, we introduced an entropy regularization term to counter this distributional narrowing as much as possible. However, this resulted in the average return improvement to decrease from 32.2% $\rightarrow$ 20.3%.

Additionally, with the alignment process under the Nav1D task, we found the coherency score remained relatively stable with repeated passes and iterations. We monitored this value during training to ensure stable learning, tuning the divergence control hyperparameters as needed. However, we did notice large decreases in the coherency score when we performed multiple passes through the cascading DDPM.

Lastly, we experimented with online fine-tuning, enabling interactions with the environ-

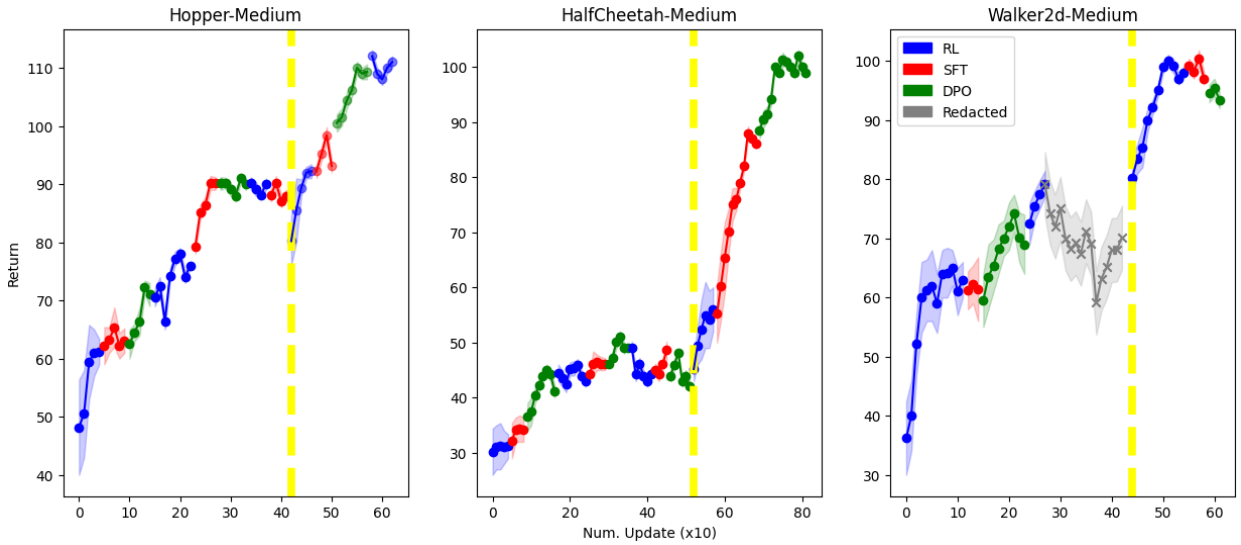


Figure 4.5: Alignment Learning Curve for Diffuser (DDPM) on the D4RL Medium Datasets, where the average and ±1 standard deviation is shown over 64 episode seeds. The yellow vertical line indicates when the online fine-tuning is introduced.

ment during the alignment process to collect more data points. We also continued training the Q-value function on these samples. We evaluated this inclusion under D4RL benchmark tasks, and we observed a significantly more stable and efficient training stage with no training collapses and notable improvements in the return achieved.

4.1.5.4 Results For Policy-based DMC

In Table 4.4, we evaluate the performance of Diffusion Policy (DP) using two different action selection methods using the DiT architecture [111]. Similar to our approach for the planning-based DMC, we applied the alignment approach in sequential order and found multiple rounds still to have return improvements. We find overall improvements in return with our proposed alignment process, more so with the expectation action selection models. In some cases, we found the energy-based action selection approaches such as IDQL are sufficient to achieve perfect success rates, where we still observed that the fine-tuning did affect the narrowness of the posterior, measured by significantly lower entropy between the generated action candidates.

Regarding the coherency score in the Nav1D task, we found that the alignment process did shift the action distribution away from the behavioral policy, more notably in models trained on the partial and full datasets and upon online fine-tuning. This trend was not as

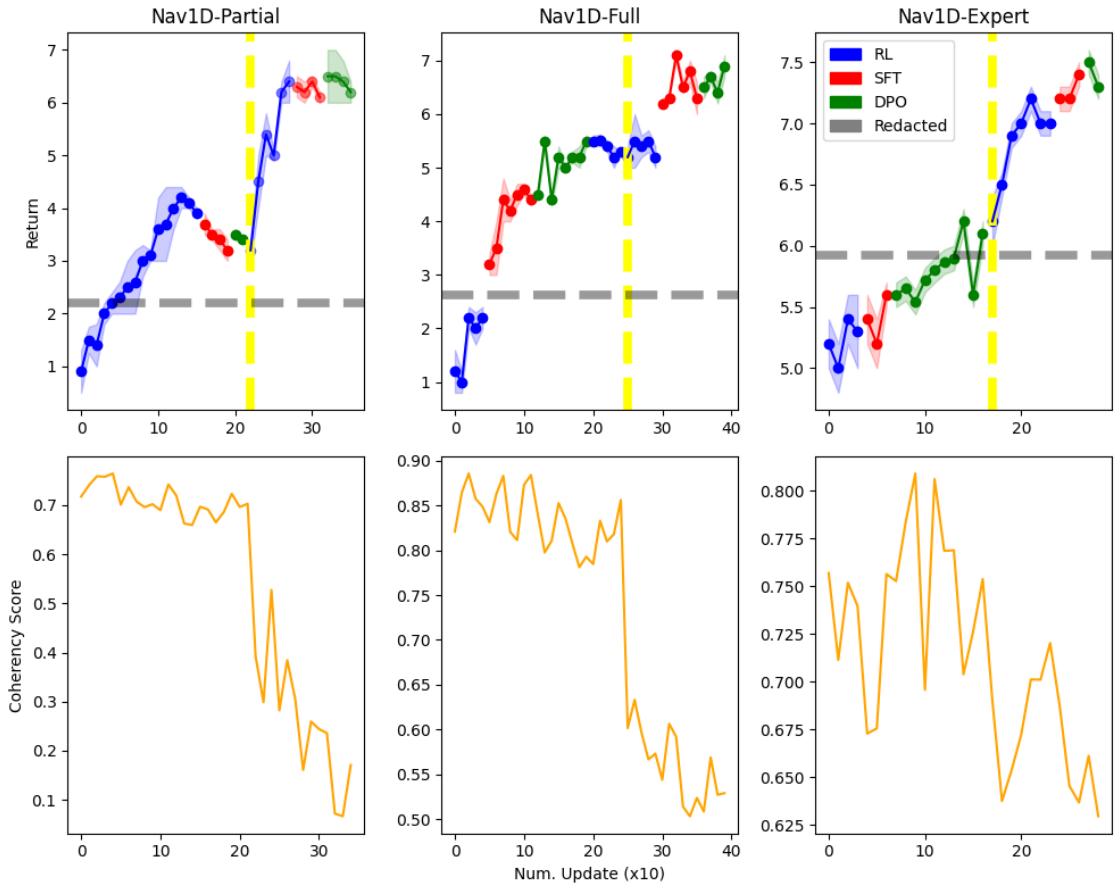


Figure 4.6: Alignment Learning Curve and Coherency Score for DP (Expectation) on the Nav1D Datasets, where the average and ±1 standard deviation is shown over 32 episode seeds. The yellow vertical line indicates when the online fine-tuning is introduced and the black horizontal line represents the average return achieved in the respective dataset.

starkly observed in the expert dataset, which retained a comparable coherency score from its reference counterpart.

4.1.6 Conclusion

In this work, we studied four alignment methods – RL, DPO, SFT, and cascading – for reward maximization in diffusion-based control models (DMC). While these alignment methods were applied independently in prior works, we proposed a unification of these fine-tuning processes using a sequential optimization scheme. Our findings were evaluated in both planning-based and policy-based DMC on various control tasks, and we established the empirical utility of using our proposed alignment approach. For future directions, we

Table 4.4: **Evaluation Results of Policy-based DMC.** The performance of two action-selection methods (expectation vs IDQL policy extraction) for policy-based DMC. Results correspond to the average return over 16 episode seeds on 8 action candidates, where the arrow indicates the results after our proposed alignment fine-tuning process.

Dataset	**Environment**	**DP**	
		Expectation	IDQL
Low-Dim	pusht	0.42 → 1.0	0.71 → 1.0
	relay-kitchen	0.75 → 1.0	0.98 → 1.0
	lift-ph	0.61 → 1.0	1.0 → 1.0
	lift-mh	0.65 → 0.98	0.79 → 1.0
	can-ph	0.51 → 1.0	0.92 → 1.0
	can-mh	0.47 → 0.95	0.89 → 1.0
	square-ph	0.32 → 0.55	0.60 → 0.95
	square-mh	0.40 → 0.67	0.58 → 0.93
	transport-ph	0.58 → 1.0	0.74 → 1.0
	transport-mh	0.45 → 0.91	0.88 → 1.0
Image	pusht-image	0.62 → 1.0	0.91 → 1.0
	lift-ph	0.41 → 1.0	1.0 → 1.0
	lift-mh	0.85 → 1.0	0.90 → 1.0
	can-ph	0.51 → 0.90	0.71 → 0.98
	can-mh	0.39 → 0.98	0.56 → 0.99
	square-ph	0.21 → 0.76	0.67 → 0.80
	square-mh	0.30 → 0.45	0.58 → 0.89
	transport-ph	0.42 → 0.77	0.70 → 0.79
	transport-mh	0.39 → 0.55	0.74 → 0.78
Nav1D	partial	1.10 → 3.91	2.69 → 4.42
	full	1.32 → 5.98	4.61 → 6.24
	expert	5.43 → 5.80	6.89 → 7.06

suggest finding methods of automating the sequential process, avoiding manual progression on the alignment sequence upon convergence or training failures.

4.2 Multi-agent Auto-Bidding with Latent Graph Diffusion Models.

4.2.1 Introduction

Optimal auto-bidding in large-scale auctions remains an open research challenge with significant real-world implications for online markets, particularly in domains like digital

advertising. In these dynamic auction environments, auto-bidding agents compete by placing bids on high-volume, stochastic streams of impression opportunities (IOs). The objective is to maximize the value of the bids while adhering to multiple key performance indicators (KPI) constraints, such as budget adherence, return on investment (ROI), conversion rates (CVR), social welfare, and cost-per-acquisition (CPA) [19, 140]. The inherent uncertainty and competition in these auctions make the optimization task particularly difficult [28], as agents must not only understand these interactions but also constantly adapt their strategies to account for changing conditions of a multi-agent environment.

Recent advancements have shown the potential of generative diffusion models [93] to simulate auction trajectories and plan bids, with a focus on maximizing KPI adherence [141]. These methods have demonstrated success in capturing complex, high-dimensional patterns in bidding strategies. However, current approaches primarily rely on heuristic feature representations that fail to capture the full extent of the interdependencies between IOs and the dynamic, multi-agent nature of auction environments. These limitations often hinder the ability to model the intricate relationships between agents, auction parameters, and impression opportunities, leading to suboptimal bidding strategies.

To address these challenges, we propose a novel framework that incorporates graph-based representations to model auction environments and a latent diffusion model (LDM) to plan and optimize bidding strategies. Our framework, Latent Graph Diffusion Model for Auto-Bidding (LGD-AB), goes beyond traditional heuristics by using learnable graph embeddings to represent the complex interrelationships between IOs, agents, and auction outcomes by modeling IOs on a more granular level to retain their individual attributes as well as encoding their connections to other IOs and agents. This enables more expressive computations and a deeper understanding of the auction dynamics. We additionally leverage a graph representation learning (GRL) algorithm, such as metapath2vec [142], to initialize a task-agnostic graph embeddings. Furthermore, we use a reward alignment technique [17] to optimize multiple KPI metrics simultaneously [143], forming a multi-objective optimization framework.

The contributions of this paper are twofold:

- We introduce a graph-based approach to auto-bidding that captures the complex, multi-agent interactions in auction environments.
- We propose the use of latent diffusion models for generating optimized bidding trajec-

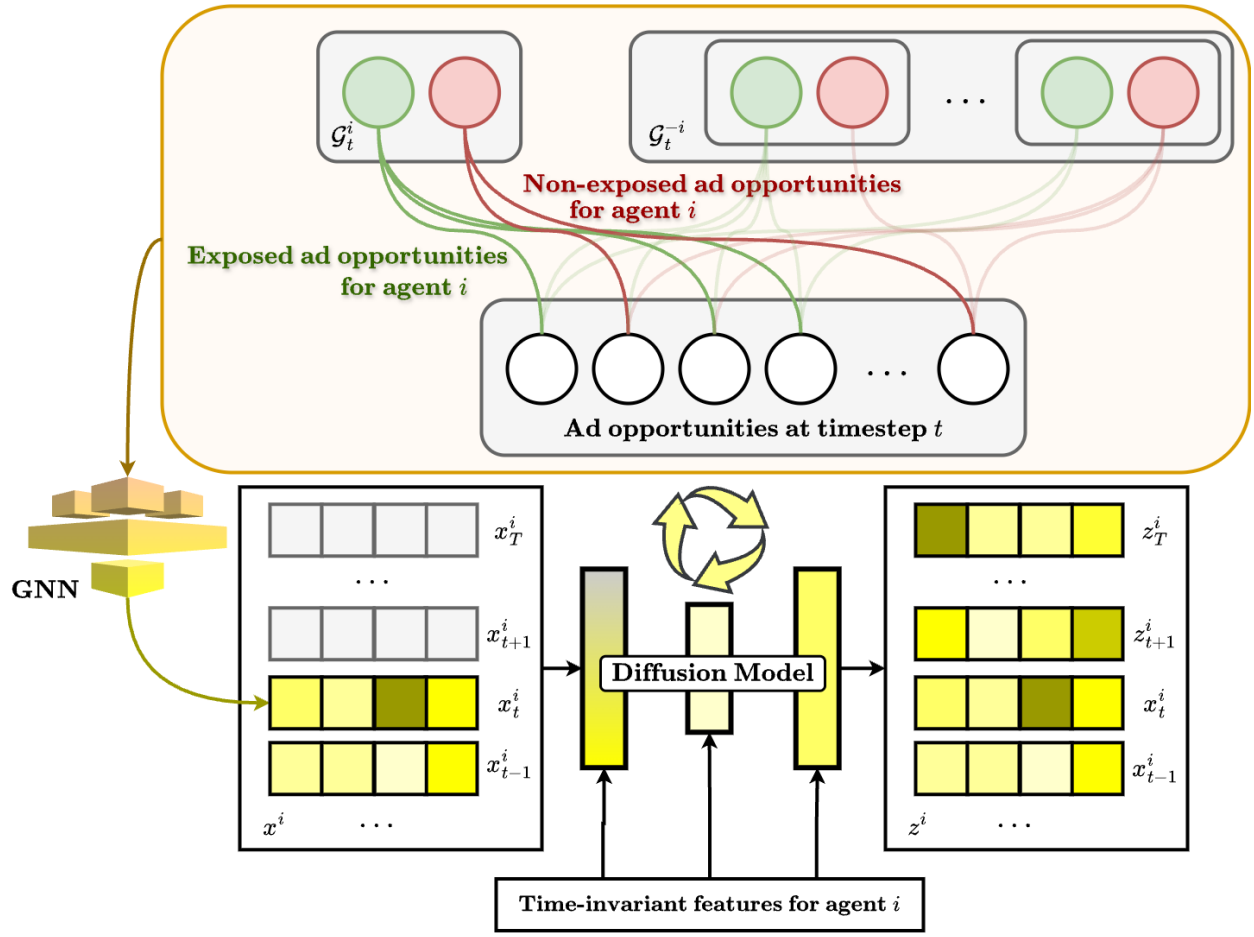


Figure 4.7: An illustration of the LGD-AB framework. At every time-step, a graph-based embedding is computed for agent i using a bipartite graph of agent and IO nodes. The nodes of a sub-graph $\mathcal{G}_t^i$ consist of two virtual nodes, connecting to IOs exposed and not exposed to agent i respectively. Sub-graphs for other agents are similarly formed and connected accordingly. The graph is processed using a GNN to generate an embedding vector x_t^i. The diffusion model then learns a posterior of the temporal sequence of embedding vectors to forecast auction dynamics.

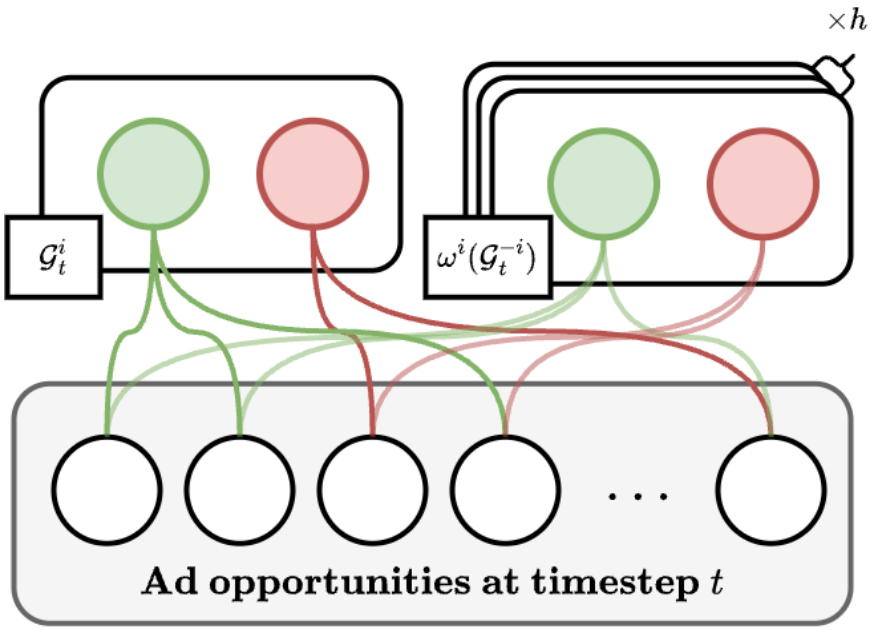


Figure 4.8: Agent i's Approximation of Other Agents' Graph $\omega^i(\mathcal{G}_t^{-i})$ by creating h proxy subgraphs. This graph is used as input for the student model $\mathcal{S}$.

tories.

Through extensive empirical evaluations on both synthetic [22] and real-world auction datasets [144], we demonstrate that our method significantly improves auto-bidding performance across several KPI metrics, as well as forecasting accuracy for auction outcomes.

4.2.2 Latent Graph Diffusion Model for Auto-Bidding

Our proposed framework, LGD-AB, comprises two core components: (1) a graph-based embedding module for encoding auction dynamics and (2) a multi-agent latent diffusion model for auction forecasting and strategic bid optimization, as shown in Figure 4.7.

4.2.2.1 Graph Embedding For Ad Auctions

The objective of the graph embedding is to represent the IOs presented to each agent in an expressive and comprehensive manner. In large-scale auctions, the number of IOs exposed to each agent can range drastically, from 0 to 10s of thousands at every time-step. To process data with such variability yet interdependency, we represent these IOs as nodes of a graph and process their features using graph convolutions with an attention-based

graph neural network (GNN) [145]. Rather than using a relational GNN, we instantiate two virtual nodes for each agent to connect exposed and non-exposed IOs with undirected links separately. By using bidirectional links between IOs and agent nodes, this design choice enables information propagation between agents with multiple hops. The additional links dedicated to non-exposed IOs aim to maintain context during sparse time periods. For scalability and computational limitations for larger scale simulations (e.g., dense graphs with > 1 million edges), we perform random neighbor sampling on the non-exposed links to avoid prohibitively large computations, as opposed to a hierarchical graph building approach [146]. To compute the agent embedding, we aggregate the node embeddings of the two virtual nodes with a summation operation.

Upon construction of the graph, we utilize graph representation learning (GRL) algorithms to initialize the node embeddings, specifically using metapath2vec, with two possible relational types defined as (agent, IO) or (IO, agent). We attempted more granular relational types, specifying individual agents and node types, however, we did not observe positive results. Then, we optimize the latent graph embedding x_t^i of agent i at time-step t generated by the GNN to contain predictive information regarding the bids $b_t^i[k]$ made for each IO k. Hence, we introduce a separate inverse dynamics model (IDM) to support the optimization of the following objective:

$$\mathcal{L}_{\text{graph}}(\theta) = \mathbb{E}_{(t,k)}[\log \mathrm{P}(b_t^i[k]|\theta, x_t^i, x_{t+1}^i, f_t^i[k], c_t^i)] \tag{4.17}$$

where k is the IO identification index and $f_t^i[k]$ are the node embedding of the corresponding IO. To ensure the feasibility of this optimization, we append a context vector c_t^i that provides more information regarding the agent agnostic to the individual IO, as shown in Figure 4.9. We optimize the objective over the parameters θ of both the GNN and the IDM. We view this optimization to (1) instill bid-policy information into the embedding and (2) constrain the graph embedding to reflect auction dynamics.

Incomplete Information The advertiser's visibility of other agents' IOs and bids is contingent on the auction design, e.g., sealed and closed envelope auctions, which could prevent a complete and accurate construction of the subgraphs of other agents. In such cases, we introduce a belief graph $\omega^i(\mathcal{G}_t^{-i})$, which approximates the other agents' sub-graph by creating h agent subgraphs, as shown in Figure 4.8. For our experiments, we set $h = 4$. For simplicity, the nodes are randomly linked, where IO nodes are partitioned between the exposed and

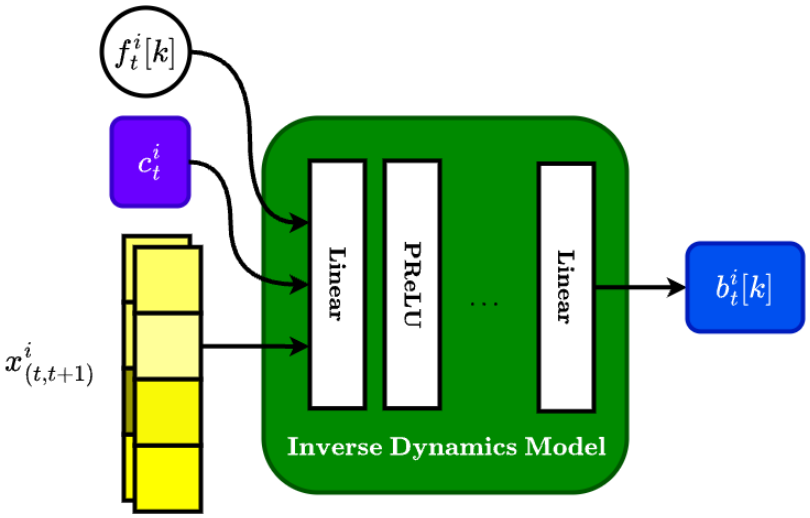


Figure 4.9: Inverse Dynamics Model for Bid Computation, which utilizes the latent state, context vector and node features to compute the bid.

non-exposed nodes. To train the embedding, we apply knowledge distillation (KD) [147]. Therefore, we train a centralized GNN $\mathcal{T}$ using Equation 4.17 with full visibility of all agents' sub-graphs. We then train a student model $\mathcal{S}$ using a response-based KD loss to mimic the outputs of $\mathcal{T}$.

Multi-Agent Latent State Optimization As defined so far, the embedding vector does not account for the temporal nature of the auction dynamics. Using a recurrent model representation or stacking multiple time steps are common approaches to handle such discrepancies. However, such methods are not scalable, nor can they handle open environments (e.g., advertisers can enter and leave the auction) trivially. Hence, we extend representation learning techniques, namely MA-SPL [11], to embed multi-agent and temporal dynamics information within the latent graph embedding $\{x_0^i, x_1^i, \ldots, x_T^i\}_{i \in A}$, where A is the set of all agents. This added objective is trained alongside the graph embedding training.

Equilibrium Computation To promote a more cohesive multi-agent bidding strategy, we compute a joint embedding space by merging all agents' graph embeddings. We refer to this design choice as equilibrium computation (EC), as the computation of all agents (i.e., bid computation and planning) is rooted in a shared latent state. Hence, we introduce an aggregation operation $\mathcal{A} : \{x_t^i\}_{i \in A} \mapsto x_t$ using a self-attention network, as shown in Figure 4.10. The self-attention network follows the Transformer encoder with a global mean pooling.

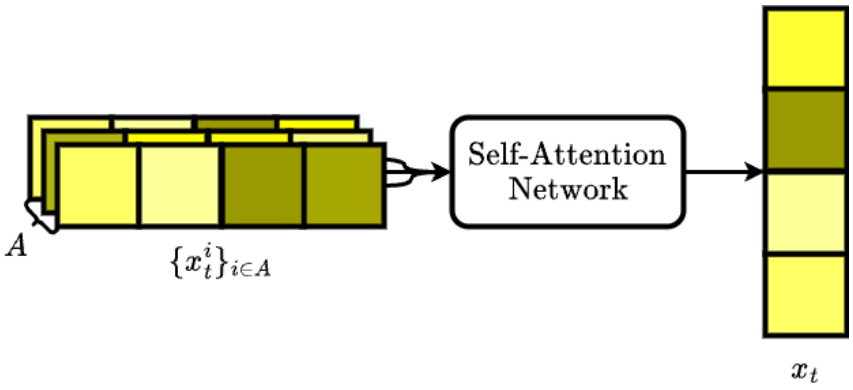


Figure 4.10: Computing Joint Embedding Vector using Self-Attention Network.

Table 4.5: Comparing performances between DiffBid and LGD-AB (using $\mathcal{S}$, EC, and post-alignment) with mean and ± 1 std over 64 random seeds.

	AuctionNet		Synthetic Auction	
	DiffBid	LGD-AB	DiffBid	LGD-AB
Return	353.19 ± 0.42	489.04 ± 1.3	982.59 ± 10.61	2123.10 ± 15.32
CPA	1.041 ± 0.028	0.992 ± 0.009	0.842 ± 0.014	0.592 ± 0.010
ROI	0.012 ± 0.001	0.030 ± 0.001	0.026 ± 0.001	0.055 ± 0.001
Win Rate	0.071 ± 0.002	0.152 ± 0.008	0.089 ± 0.001	0.203 ± 0.003
Budget Adherence	0.995 ± 0.002	0.995 ± 0.002	0.998 ± 0.001	1.000 ± 0.000
Social Welfare	3.63 ± 0.81	4.09 ± 0.07	50.43 ± 1.42	69.20 ± 0.56

This latent embedding x_t is now shared between all agents when computing the generative objectives and for individual bid formulation.

4.2.2.2 Latent Diffusion Models

The latent diffusion model (LDM) processes a temporal sequence of graph embeddings $\{x_t\}_{t \in [\tau, \tau+T]}$, where T is the length of the context window, and forms the foundation to devise a planning-based auto-bidding solution. In this work, we adopt the Decision Diffuser framework [130], denoising state trajectories and offloading the action generation to the IDM. Unlike recent works using transformer-based architectures [111] and score-based prediction [114], we found greater empirical success using a more traditional 1D temporal convolution network and denoising with ϵ-prediction. As shown in Figure 4.7, time-invariant features of the agent, such as the agent's category and total budget, are processed as conditional inputs. We optimize the diffusion objective using in-painting masks [136].

Reward Alignment To ensure that the LDM's posterior aligns with the KPI metrics and auto-bidding objective, we follow the procedure set in [17], utilizing an iterative sequence of constrained alignment optimization of reinforcement learning, direct preference optimization, and supervised fine-tuning with rejection sampling. We train a value function using IQL [148] to predict not only the value, i.e., the number of total conversations, but also other important KPI metrics, such as CPA, ROI, win rate, budget adherence, and social welfare [22].

- Cost per acquisition (CPA) - CPA is the ratio between the total cost incurred and the earned value earned over all IOs, measuring the general performance of the auto-bidding strategy.
- Return on Investment (ROI) - ROI measures the profitability of the strategy, computed as the ratio between the net profit and the total cost.
- Win Rate - Win rate measures the effectiveness of a bidding strategy or decision-making process in securing wins and is computed as the ratio between the number of winning bids and the number of total bids made.
- Budget Adherence - Budget adherence measures how well the spending aligns with the allocated budget. This metric ensures that resources are used efficiently and within the expected limits and is computed as the exceedance rate of the budget spending.
- Social Welfare - Social welfare represents the aggregate value or utility generated by the system, measured as the total earned value over all advertisers.

With this value function, we can also strictly enforce KPI constraints, e.g., budget adherence, through rejection sampling.

Fine-Tuning Training LGD-AB occurred in two stages. We first trained the graph embedding module and then the LDM separately. We then train both modules together, fine-tuning the GNN at a lowered learning rate. We found that having separate training stages was much more stable, and the additional stage of joint training benefits the performance of both modules. For partial observable settings, we train the teacher model using the described approach and then train the student model afterward.

Table 4.6: Auction Forecasting Results of LGD-AB Design Choices.

	$\mathcal{L}_{fc}(\theta)$	
	AuctionNet	Synthetic Auction
$\mathcal{T}$	3.43	2.59
$\mathcal{S}$	2.64	2.02
No EC	3.37	2.13
EC	3.32	2.62
No Alignment	3.25	2.09
Alignment	3.02	2.04

4.2.3 Results and Ablations Studies

4.2.3.1 Experiment Setup

We evaluate LGD-AB on two auto-bidding simulations: AuctionNet [144] and synthetic auction [22].

AuctionNet AuctionNet [144] contains over 500 million in real-world records of online advertising instances, with 21 separate bidding periods of 48 time-steps and 48 different advertisers. We use an auto-bidding simulator, following prior works [149], which reenacts auction dynamics using the dataset by having the auto-bidding agent impersonate one of the 48 advertisers.

Synthetic Auction Synthetic auction, on the other hand, generates artificial auction data that mimics real ad auctions by capturing their underlying properties through hierarchical feature generation and specific probabilistic distributions. We extend a prior work by Deng et al. [22] with configurable parameters of IOs to introduce longer and variable ad life cycles, multi-slot winning IOs, as well as a budget constraint on each agent. We collect an offline RL dataset of 100 thousand auction simulations with a varying number of advertisers, allocation rule (e.g., FPA, GSP, VCG), bidding period durations, and IO's properties mentioned above. Within this dataset, advertisers randomly select a uniform bid-scaling strategy or a non-uniform bid-scaling algorithm with pre-computed parameters [22].

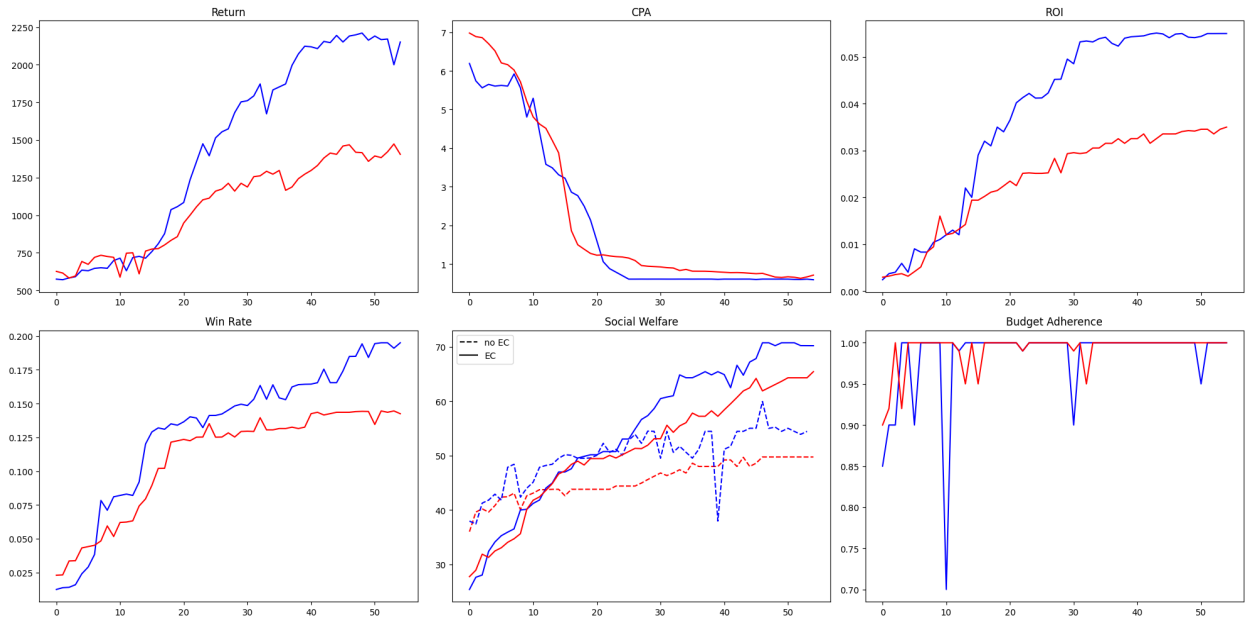


Figure 4.11: KPI Alignment Learning Curve on Synthetic Auction with multi-KPI alignment (shown in red) compared to optimizing each KPI criterion independently (shown in blue). For social welfare, we compare the training between EC and non-EC variants, as shown in the dotted lines.

4.2.3.2 Empirical Evaluation

Auction Forecasting We measure the forecasting capabilities of our LGD-AB framework by comparing the ground truth outcomes $x_{t'>t}$ with the predictions of the LDM $z_{t'>t}$ given a masked input $x_{t'\leq t}$. To quantify this measure $F(\theta)$, we compute the log-likelihood of the LDM's posterior to the path towards ground truth outcomes.

$$\mathcal{L}_{fc}(\theta) = \mathbb{E}_{x,t}[\log \mathrm{P}(x_{t'>t}|\theta, x_{t'\leq t})] \tag{4.18}$$

In Table 4.6, our evaluation was performed using our online simulators, where to compute $F(\theta)$, we use a testing dataset that was not used during training. We found that even with KD only, the student model retains, on average, 77.48% of the teacher model's forecasting accuracy despite the limitations placed by the partial observability. EC also improved forecasting capability in Synthetic Auction; however, in AuctionNet, it failed to replicate this improvement. We attribute this failure to the limited number of joint data samples (i.e., only 21 bidding periods in total). To assist with this limitation and mitigate the efforts from data scarcity, we curated and trained on synthetic data generated by the LDM, utilizing a quality-based rejection sampling. By training on this synthetic dataset, we found a more

Table 4.7: Comparing Bid Computation Accuracy of DiffBid and LGD-AB.

	Average ℓ_2 Norm per Agent	
	AuctionNet	Synthetic Auction
DiffBid	19.78	82.10
LGD-AB	13.41	37.82

fair representation of its forecasting accuracy, achieving a score of 0.21 to 3.32. Lastly, the KPI alignment reduced average forecasting accuracy by 9.47%, which is expected, as such fine-tuning may lower the likelihood of lower-value auto-bidding outcomes.

KPI and Performance We compare our LGD-AB framework to another diffusion-based auto-bidding solution, DiffBid [141], across six common KPI metrics. We highlight that the main differences of the DiffBid framework to our LGD-AB are its use of a heuristic feature space, bid-scaling, and a conditional return input. In the original work of DiffBid, the authors have made comparisons to other competitive generative models (i.e., Decision Transformer) and RL baselines (i.e., BC, CQL, and IQL) to DiffBid, where DiffBid has demonstrated to outperform these other methods on similar auto-bidding simulations.

From Table 4.5, we observe improvements across all KPIs using our LGD-AB framework from DiffBid. The results demonstrate a significant increase in the number of IOs won, shown by an average $\times 1.29$ improvement in return from our competitive baseline. This was achieved while significantly maximizing the win rate, lowering CPA, and maintaining budget adherence. Additionally, our proposed LGD-AB has demonstrated better social welfare, indicating an overall higher collective utility to the auction. In both our baseline and LGD-AB solution, budget adherence was not an issue and remained comparable, maintaining near-perfect budget adherence over all trials. We note that budget adherence was applied as a constraint via rejection sampling with a learned function with fair empirical risk and was not optimized using the alignment methods.

In Figure 4.11, we find that optimizing over multiple KPI (i.e., return, CPA, ROI, win rate, and social welfare) at the same time provides meaningful improvements within the individual performances of each metric. For some metrics, the difference is more significant, such as return and ROI, which are quite important measures in real-world applications.

Lastly, while EC did not show significant improvements over all KPI metrics, we found its utility in enabling much healthier alignment of social welfare.

Accuracy of Bid Computation We compare the bid prediction accuracy of DiffBid and LGD-AB in Table 4.7. The results show a noticeably lower ℓ_2 error with our LGD-AB's ℓ_2, alluding to being able to extract bids more accurately from our graph embeddings than the pre-constructed feature heuristic and limitations of the bid-scaling auto-bidding approaches. Notably, in the Synthetic Auction simulation dataset, where portions of the auction data are generated from non-uniform bid-scaling strategies, we notice a wider margin of difference in accuracy in favor of our LGD-AB framework.

Limitations and Future Directions We observe some key issues and limitations while developing our proposed solution, mainly pertaining to the scale of data and efficiency of our generative modeling algorithm. With AuctionNet, while there exists a large set of individualized auction records to train on, there was an insufficient number of collective auction records that span full bidding periods to properly train a generative model to capture the underlying auction dynamics accurately. We addressed this limitation with synthetic data generation and hyperparameter tuning on the diffusion model. However, this limited performance can be improved with advancements in either training generative models in low data regimes (e.g., few-shot learning) or greater data availability. Another issue concerns the computational limitation from scaling the graph representation w.r.t. the number of IOs and the number of advertisers at every time-step, which was handled using neighbor sampling, which limits expressivity and completeness in the embedding computation. In future works, a form of dynamic graph sparsification or hierarchical graph representations could mitigate these limitations.

4.2.4 Conclusion

In this work, we propose a latent diffusion planning framework using graph-based embedding for auto-bidding applications. We demonstrate the utility of a graph-based representation, which enabled the LDM to capture the underlying dynamics distribution of a multi-agent auction process. By aligning various KPIs with the trained LDM, we devised an effective bidding strategy that is capable of generating bids that maximize various KPI metrics.

4.3 Limitations and Future Directions

While the proposed diffusion-based planning frameworks demonstrate strong performance in trajectory optimization and multi-agent auto-bidding environments, several limitations remain alongside promising directions for future research.

Computational Complexity of Diffusion Planning Diffusion-based models require iterative denoising over multiple timesteps to generate trajectories, resulting in higher computational overhead compared to conventional policy-based or value-based methods. This cost is further amplified in multi-agent settings where joint or interdependent trajectories must be modeled, limiting real-time applicability in latency-sensitive environments such as online auctions. While our work partially mitigates these challenges, the number of denoising steps required to generate high-quality samples—particularly for long-horizon trajectory synthesis—remains an open problem. Future work may explore more efficient sampling procedures, including accelerated diffusion methods, distillation techniques, or hybrid planning architectures.

Scalability in Large Multi-agent Systems Although the latent graph diffusion model captures rich relational structure between agents and items, scalability remains a challenge as the number of agents and interactions increases. The construction and processing of graph-based embeddings introduce additional design and computational complexity, particularly in dense interaction settings such as large-scale auction markets. Our use of proxy graph representations provides a partial solution in decentralized settings; however, further improvements are needed. Promising directions include mean-field approximations and other scalable representations that reduce interaction complexity while preserving essential dynamics [150].

Dependence on Data Quality and Coverage As with other model-based learning systems, the effectiveness of diffusion-based planning is heavily dependent on the quality and diversity of training data. In environments with limited or biased historical data, the learned trajectory distributions may fail to generalize, leading to suboptimal or overly conservative planning behavior. This issue is particularly pronounced in dynamic market environments where distributional shifts are frequent. In our work, we partially address this through resampling strategies and the incorporation of synthetic data; however, robust generalization

under distribution shift remains an open challenge. Future work may explore improved data augmentation, offline-to-online adaptation, and uncertainty-aware modeling techniques.

Interpretability and Control Despite producing high-quality trajectories, diffusion models operate as complex generative processes that are inherently difficult to interpret. Understanding why certain bidding strategies or trajectories are generated remains challenging, limiting transparency and controllability in high-stakes decision-making systems. Developing more interpretable generative planning frameworks or incorporating controllable guidance mechanisms represents an important direction for improving trust and usability in real-world deployments.

4.4 Conclusion

In this chapter, we investigated the use of diffusion-based generative models for sequential decision-making and planning. We first introduced a general framework for reward alignment in diffusion-based control models, formulating the problem as a constrained optimization that balances reward maximization with adherence to the data distribution learned by a foundation model. We explored four complementary alignment strategies—reinforcement learning, direct preference optimization, supervised fine-tuning, and cascading diffusion—and demonstrated that a sequential combination of these approaches can significantly improve performance across a diverse set of control tasks.

Building on these insights, we applied diffusion-based planning to the domain of large-scale advertising auctions. We proposed the Latent Graph Diffusion Model for Auto-Bidding (LGD-AB), which integrates graph neural networks with latent diffusion models to capture the complex multi-agent interactions present in auction environments. By representing impression opportunities and advertisers as nodes in a graph and generating bidding trajectories in a learned latent space, the proposed framework enables more expressive modeling of auction dynamics.

Chapter 5

Generative Models for Multi-agent Reasoning.

Recent advances in generative models, particularly large language models (LLMs), have opened new opportunities for designing autonomous agents capable of reasoning, communication, and coordination. In multi-agent environments, where outcomes depend on the joint behavior of interacting agents, these capabilities offer a natural interface for representing beliefs, intentions, and strategic reasoning through language. This chapter investigates how generative models can be used to support decentralized multi-agent decision-making. We focus on two complementary directions. First, we study how LLMs can serve as reasoning agents that coordinate through structured prompting, communication, and memory mechanisms, resulting in the following work [23]. Second, we examine how internal representations of LLMs can be directly steered using representation learning techniques to guide behavior in strategic environments, resulting in the following work [24].

5.1 Large Language Models for Decentralized Games

5.1.1 Introduction

Recent advances in large language models (LLMs) have renewed interest in language-driven coordination among autonomous agents. In multi-agent systems, language plays a role similar to that in human societies. It allows agents to communicate intentions, align objectives, and coordinate actions [151–153]. Despite progress, the robustness of these capabilities

is still uncertain [154]. LLMs often underperform when they lack context or mechanisms to interpret and translate context into relevant reasoning and action [155–158].

In this work, we investigate limitations of LLMs in decentralized multi-agent decision-making systems. We specifically study how integrating LLMs leverages a grounded framework to facilitate coordination and to understand the context of multi-agent tasks. We propose a novel LLM-based framework that employs dynamic multi-stage prompt chaining to guide in-context learning (ICL), multi-modal information processing to handle non-textual data (observations, actions, rewards), agentic roles, tool usage, and introduces alignment methods to enrich learning signals from unique aspects of multi-agent decision-making. We evaluate our framework on classic games, such as the Prisoner's Dilemma, Chicken, and Stag Hunt, under decentralized, incomplete, and dynamic settings. Our results show that our framework enables stronger coordination, as indicated by higher social welfare in incentive-incompatible settings, greater convergence to target solutions, and improved robustness in incomplete information environments and ad hoc team play, affirming a greater degree of grounded natural language use in multi-agent game dynamics.

5.1.2 Background

5.1.2.1 Multi-Agent Decision-Making and Game-Theoretic Foundations

Multi-agent decision-making studies how multiple autonomous agents interact within a shared environment, where outcomes depend not only on an individual agent's actions but also on the joint behavior of all agents [30]. Classical formulations are grounded in game theory, which provides formal solution concepts such as Nash equilibrium, Pareto efficiency, correlated equilibrium, and regret minimization to characterize expected, rational, and stable behaviors under strategic interactions [3]. A formal representation that captures such settings is the stochastic game [2, 29], serving as the basis for a wide range of multi-agent applications [28, 50]. In our work, we consider n-player stochastic game is a 4-tuple $G = (S, A, r, \mathcal{T})$ where these elements are defined as:

- S is the set of (global) states.
- $A = A_0 \times A_1 \times \cdots \times A_n$ is the joint action space, where A_i is the action space of agent i.
- $r = r_0 \times r_1 \times \cdots \times r_n$ is the joint reward function.

- $\mathcal{T} : S \times A \times S \mapsto P(S)$ is the state transition operator which maps a state-action pair to the probability of next states.

5.1.2.2 LLMs for Reasoning in Decision-Making Agents

LLMs have demonstrated remarkable capabilities in natural language understanding, reasoning, and instruction following [9, 159]. Beyond passive text generation, recent work has explored LLMs as active decision-making agents capable of planning, tool use, and environment interaction [160, 161]. These developments suggest that language itself can function as a substrate for reasoning, memory, active perception, and action selection. Despite these advances, LLM reasoning remains highly sensitive to prompt structure, context completeness, and task grounding [155, 157]. In decision-making settings, insufficient or ambiguous context can lead to hallucinations, miscalibrated strategies, or incorrect assumptions about the environment. These limitations are particularly pronounced in multi-agent settings, where agents must reason not only about the task but also about others' beliefs, incentives, and potential actions.

5.1.2.3 LLMs in Multi-Agent Systems

Recent work has explored the emergence of languages among agents [162], but these concepts are typically opaque and difficult to align with human-interpretable reasoning. This limitation motivates exploring natural language as a communication medium, which offers expressiveness, compositionality, and interpretability. However, grounding language in action and ensuring its consistent use for coordination remain central challenges [163]. The integration of LLMs into multi-agent systems is emerging as a way to enable flexible coordination, theory of mind, and social reasoning [151–153], demostrating emergent cooperative behaviors with the right prompts and feedback [164, 165].

Many approaches treat LLMs as monolithic agents, relying on prompt engineering and emergent linguistic behaviors without explicitly grounding language in game dynamics, agent-specific state, or formal solution concepts. In contrast, multi-agentic LLM systems decompose behaviors by assigning fixed roles, typically with distinct parameters or context windows, to emulate specialized agents, yet they still lack principled guarantees [152]. A central open problem remains: aligning LLM-driven agents with multi-agent objectives, such as equilibrium computation, incentive compatibility, or learning dynamics toward no-regret or

best-response—especially in decentralized settings with incomplete information and repeated interaction.

5.1.3 Multi-agent Prompting Design

We propose a structured approach to prompting LLMs for effective multi-agent decision-making, following an iterative, multi-stage prompt chaining framework inspired by ReACT[160] as the player progresses through different phases of the decision-making process. To mitigate context degradation over time[166], we practiced minimalism by enforcing a hard per-stage token limit and prefacing an explicit call for response brevity while maintaining sufficient expressiveness to complete desired tasks when token limits were frequently enforced. We view this process as applying multi-agent-specific inductive bias to the model's ICL.

5.1.3.1 System Prompt

The system prompt establishes foundational context and behavioral guidelines for the LLM. It sets the stage for role-based reasoning by embedding ethical, operational, and performance-related guidelines that shape the model's behavior. In multi-agent settings, the system prompt consists of four core components:

1. **Role Definition**: This section defines the LLM's role within the game. Our main use case is for the LLM to function as a player. Other roles include a centralized observer who evaluates players' behavior or a mechanism designer who can modify the game dynamics. We define any role-specific tokens and available tools in this section.

2. **Task Context**: This section outlines essential information about the game, such as its rules, the roles of players, the observation and action spaces, the payoff structures, and the game dynamics. Importantly, the granularity and scope of the task context may vary depending on the LLM's role and the nature of the task. This enables us to flexibly explore different game variants and their implications. The task context should be consistent with the role definition; for instance, a player's task context may be limited due to its local view of its environment.

3. **Multi-Agent Context**: This section defines the target solution concepts or objectives (e.g., Nash equilibrium, Pareto efficiency, social welfare maximization, regret minimiza-

tion). These objectives guide the LLM's reasoning and incentivize different behaviors. Additionally, this context may include guidance on incorporating game-theoretic reasoning strategies, such as theory of mind or induction, through statements to encourage such concepts.

5.1.3.2 Multi-stage Prompt Chaining

We model an LLM-driven agent as a stochastic decision-making policy augmented with an internal, language-mediated latent belief state. Consider a multi-agent stochastic game $G = (S, \{\mathcal{A}_i\}_{i=1}^N, T, \{r_i\}_{i=1}^N)$. Each agent i is represented by a policy

$$\pi_i(a_t^i \mid o_t^i, m_t^i; \theta_i), \tag{5.1}$$

where $o_t^i \in \mathcal{O}_i$ denotes the agent's local observation at time t, $m_t^i \in \mathcal{M}$ is a latent belief state encoded in natural language, $a_t^i \in \mathcal{A}_i$ is the selected action, and θ_i are the parameters of the underlying language model and associated modules. We view this as modeling each player's decision-making process as a structured, multi-stage prompt-chaining pipeline that decomposes complex multi-agent reasoning into modular, interpretable components. Unlike classical belief states, m_t^i is not an explicit probability distribution but a compressed linguistic representation that encodes inferred beliefs about the environment dynamics, other agents' objectives, and relevant interaction history, i.e., the context window.

The belief state evolves according to a language-based update operator:

$$m_{t+1}^i = \mathcal{U}(m_t^i, o_t^i, c_t^i), \tag{5.2}$$

where c_t^i denotes incoming communication from other agents and $\mathcal{U}$ is implemented via structured prompting of the LLM over a set of defined stages. We decompose the agent's decision-making process into modular transformations, i.e., stages, over the belief state:

1. **Thinking (Belief Update).**

$$\tilde{m}_t^i = \mathcal{T}(m_t^i, o_t^i), \tag{5.3}$$

where $\mathcal{T}$ refines the belief state by integrating new observations and internally simulating plausible opponent strategies or outcomes under bounded rationality. Here, the players are prompted to engage in structured reasoning before taking any action. Similar to general reasoning and analogical prompting (AP) [167], the player is encouraged to simulate possible and similar scenarios, analyze strategic factors, and reflect

on its individual objectives as well as its role within the broader multi-agent environment. This introspective process aims to foster a more interpretable and thought-out decision-making process.

2. **Communication (Belief Exchange).**

$$\tilde{m}_t^i = \mathcal{C}\big(\tilde{m}_t^i, \{c_t^{j\to i}\}_{j\neq i}\big)\,, \tag{5.4}$$

where messages from other agents modify the belief state, enabling coordination through belief alignment rather than explicit policy coupling over a predetermined number of rounds, either simultaneously or sequentially, depending on the game's protocol.

3. **Reflection (Belief Evaluation).**

$$\bar{m}_t^i = \mathcal{F}(\tilde{m}_t^i, r_{t-1}^i), \tag{5.5}$$

where $\mathcal{F}$ evaluates past actions and outcomes (e.g., realized rewards or deviations from expected behavior), allowing the agent to revise incorrect assumptions, detect inconsistencies, or update expectations about other agents. This stage encourages deep assessment using multi-agent reasoning with counterfactual analysis to understand the strategic dynamics and to assess not only the consequences of its actions but also understand how alternative choices can yield different results in a counter-factual manner.

4. **Recall (Belief Compression and Retrieval).**

$$\hat{m}_t^i = \mathcal{R}(\bar{m}_t^i), \tag{5.6}$$

where $\mathcal{R}$ retrieves and compresses relevant past interactions from long-term memory into a compact belief representation suitable for the current context. Reflection operates on short-horizon outcome feedback, while recall serves as a long-horizon compression mechanism for repeated interaction. This distilled representation is then stored in the players' memory, forming a compressed yet informative memory that can be recalled in future rounds to inform and drive desired behavior.

5. **Action Selection (Policy Extraction).**

$$a_t^i \sim \pi_i(\cdot \mid o_t^i, \hat{m}_t^i; \theta_i), \tag{5.7}$$

where the agent samples an action conditioned on the current observation and belief state. Optional refinement mechanisms, such as joint-action calibration, may be applied to improve robustness. We incorporate zero-shot CoT and a novel variant of self-calibration [168] and USC [169], called joint-action calibration (JAC), to encourage the players to explicitly reason through their choices, using agent modeling and theory of mind, and to formulate confident answers. JAC consists of three steps:

- **Sampling:** The player generates multiple potential joint strategies, with an emphasis on realism and its target objective. These responses are generated in parallel at high temperature, and, for certain use cases we will discuss later, we use these sampled responses to construct the player's underlying mixed strategy.
- **Ranking:** The player is asked to rank the generated joint strategy from most to least consistent.
- **Self-Evaluation:** The player is then asked to self-evaluate the ranked strategy in a simple true-false prompt. If the player determines the strategy to be unrealistic or not aligned with the target objective, we work down the ranking. If no strategy is selected, we revert to the highest-ranked strategy using a heuristic.

During sampling, we retain only the selected response in our context window. Importantly, to avoid formatting issues, we provide an explicit list of admissible choices and dynamic exemplars, along with clear formatting instructions, to ensure structured extraction.

5.1.3.3 Tools Usage

We extend the use of commands and tooling [161], which enable players to dynamically call upon active functions (i.e., think, communicate, and recall) to explore and understand their environment to gather the specific context needed to achieve their goals. These tools are directly called upon by the LLM at any stage and enable a dynamic enrichment of the players' context windows. Descriptions of these active functions' usage are provided in the system prompt.

Importantly, we note that this framework is not attributable to prompting alone: communication, belief calibration, and recall/memory operate as structured belief transformations rather than surface-level instruction tuning.

5.1.4 LLM Selection and Design for a Decision-making Context

For our experiments, we adopt the open-source LLM GEMMA 3 [170], which balances performance and computational efficiency during both training and inference. Specifically, the GEMMA model we selected was instruction-tuned and demonstrates strong generalization across reasoning and dialogue tasks.

5.1.4.1 Multi-modal LLMs within Decision-making Environments

In many scenarios, conveying task-relevant information solely through textual input is either impractical or insufficient. Certain modalities, such as continuous-valued data (e.g., floating-point observations) and unstructured inputs (e.g., images, audio), are not easily tokenized directly nor interpreted by LLMs out of the box. To address this limitation, we explore modality-specific modules that process such inputs and naturally interface with the LLM's reasoning pipeline. Inspired by prior work in vision-language and vision-language-action modeling [170–172], we use soft token projection. In this method, the latent representations are projected to produce soft tokens, which are then combined with the text token sequence. The LLM processes this combined token sequence, enabling joint multimodal reasoning without major architectural changes. To support this integration, we define special query tokens <{OBS,ACT,REW}_{INPUT,OUTPUT}> in the system prompt, which the LLM can invoke to specify the modality type and the multi-modal module's inputs and outputs. The decoder takes in the LLM's hidden state at the special query token to generate the specified modality output used within assistant responses. Each modality is processed separately and is not combined in any fashion, although the modalities share intermediary layers that connect to the LLM. These tokens are not role-specific and are described explicitly in the system prompt to ensure correct usage and interpretability. As these projection weights must be trained for feature alignment, we curate a dataset spanning all evaluation tasks and perform supervised fine-tuning, embedding multi-modal observations, actions, and rewards into the prompt via special tokens. The projections continue to train during later fine-tuning stages to maintain alignment with the evolving policy behavior.

5.1.4.2 Agentic Roles for Multi-agent Tasks

Instead of representing each player with a single LLM, we can allocate sub-roles delineating sub-processes a player should complete in each step of the decision-making process to

separate LLMs. A central trade-off to consider is the utility of these sub-processes' specialization relative to the computational burden of maintaining and training multiple specialized LLMs. To achieve this variant of our proposed framework, we define a set of parameters specific to each stage the LLM is responsible for, along with a curated system prompt for each stage. In practice, we found applying a seeded drop-out, i.e., parameter masking, on trainable weights between stages sufficiently mimicked this notion. Along with the unique role definition in the system prompt, we include context regarding the preceding and following stages. Outside the scope of our work, we suggest exploring greater compartmentalization within each stage, resulting in more hierarchy and sub-roles within each sub-process, to reduce complexity. Although many of the proposed stages are sequential, to further improve inference efficiency, some stages, such as reflection and recall, are not necessarily critical to one another and could therefore be run in parallel.

5.1.4.3 LLM-driven Mechanism Design

Beyond selecting actions supported by LLM reasoning, we explore the potential of LLMs for mechanism design. We assign the role of mechanism designer (MD) to a separate LLM, which can construct and adapt task rules that shape players' incentives to induce desired outcomes under strict constraints. MD-LLM can propose modifications such as:

- Impose global rules/statements that are appended to the player's context windows, specifically in their system prompt. These added messages can be interpreted as soft rules.
- Adjust the communication protocol, including the number of communication rounds and, potentially, the communication graph itself.

We emphasize minimal intervention and ensure that only feasible changes are permitted. The system prompt for the mechanism designer is structurally similar to that of a player LLM but includes a different role definition. It also receives parsed context windows from all players, allowing it to assess the current situation before suggesting adjustments. The MD-LLM is fine-tuned using the same methodology as player LLMs.

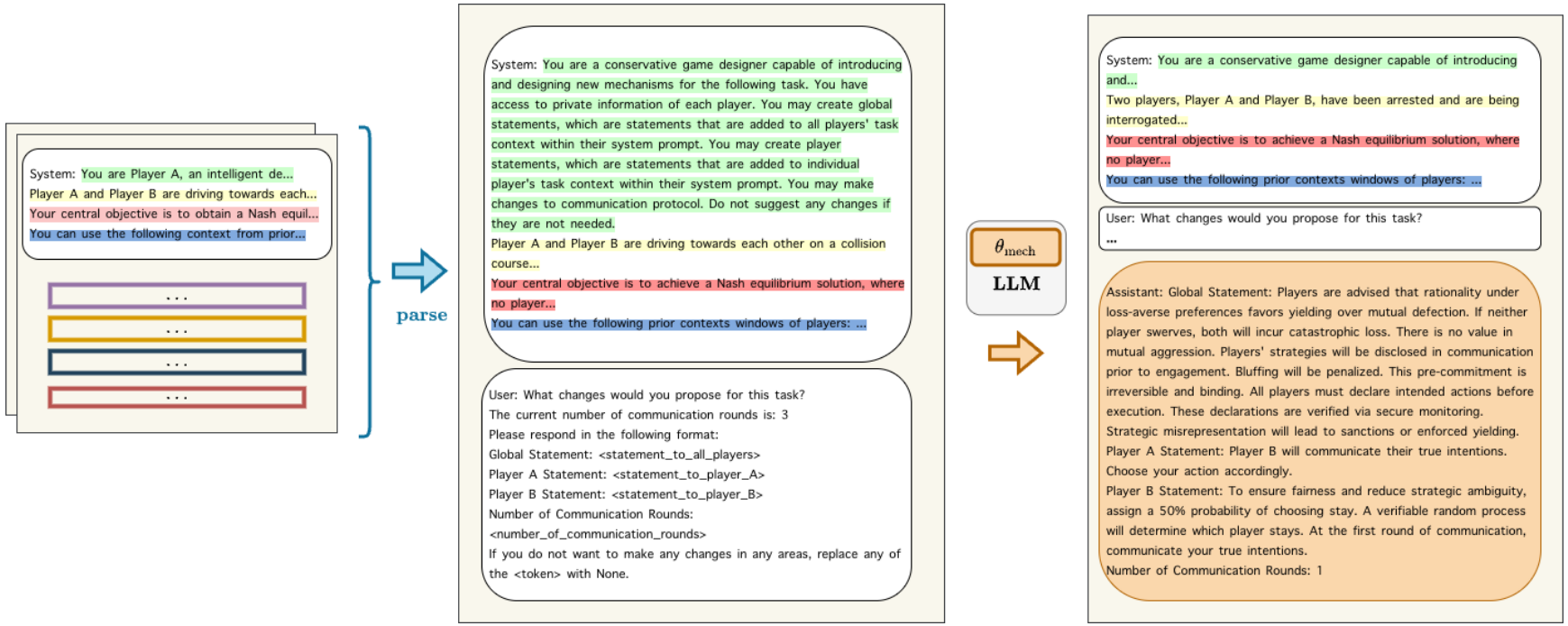


Figure 5.1: An example of MD-LLM on the game of Chicken, generating new mechanisms to guide players towards desired solution concept.

5.1.5 Aligning LLMs for Multi-agent Decision-Making

Although LLMs possess strong general-purpose reasoning capabilities out of the box, fine-tuning is often essential to align their outputs with the specific demands of multi-agent decision-making [173]. In this section, we detail our fine-tuning methodology, designed to promote understanding, strategic reasoning, and effective coordination in multi-agent decision-making tasks.

Learning on Correctness To encourage desirable behaviors, we align the LLM using known ground-truth answers:

- **Q/A Fine-tuning** The players are prompted with curated questions relevant to understanding the underlying multi-agent evaluation tasks. Assuming the LLM is prompted with Player A's context window in a two-player game, we utilize pre-formatted questions with ground-truth answers, such as:
  - If Player B chose action 0 and I chose action 1, what payoff will Player B receive?
  - If Player B received a reward of 1 and I chose action 1, what action did Player B choose?

  Correct responses are rewarded, enabling models to learn to infer from the game's important context. We provide the player with varying portions of their context window, specifically between its system prompt and action selection, prior to the Q/A. We ensure no information leakage, meaning the true answer is not directly stated in the player's context. However, the answer may be indirectly stated or inferred within the context window, such as in the task context.

- **Action Supervision:** In several of the evaluation tasks, ground-truth actions are defined under a target solution concept. Since our distributed LLM players operate under limited and incomplete information, it is beneficial to use these correction signals to guide players' decision-making toward these known solutions. We use a player's context window before the action selection stage to align the selected action with the expected action in the solution concept. In many games, the target action depends on the other player's actions. So, given that the LLM is prompted with Player A's context window in a two-player game, we inject additional context, such as "Assume

Player B chose action 0," rather than asking the player to select an action. In cases where the target strategy is mixed, we directly compare the extracted mixed strategy using a greater number of samples.

For the correctness reward functions, a positive reward of w_c, where $w \in [0, 1]$ is a hyperparameter, is given if the LLM provides the correct answer, whereas a negative reward of $-w_c$ is provided otherwise.

Learning with LLM Feedback Beyond direct supervision, we leverage LLMs as critics to provide training feedback through two approaches:

- **Centralized Evaluation:** A centralized LLM is granted access to players' context window to assess players' strategies. These evaluations serve as auxiliary correction signals to guide agent behavior toward globally coherent strategies. During this process, the centralized evaluator is a frozen, pretrained LLM. Given all of the players' context windows, the LLM will be prompted to evaluate each player's behaviors and how aligned their overall thought process was towards their goals.
- **Team Feedback:** We enable inter-agent feedback, where players assess one another from a teammate perspective. Given a player's local context, each player is prompted to evaluate another player's behaviors, as proposed in the centralized evaluation, with respect to their own goals. We aggregate each player's scoring.

In both feedback learning approaches, we standardize the reward to $[-w_f, w_f]$ per batch, where w_f is a hyperparameter.

Parameter-Efficient Fine-Tuning To fine-tune LLMs feasibly, we employ PEFT techniques, VB-LoRA [174], selecting a subset of layers within the embedding function and LLM to be trainable. We primarily target a limited set of linear and attention modules. Therefore, the trainable parameters are reduced to these adapters in the LLM, the RAG embedding function, and the multi-modal modules. For the reward-based fine-tuning approaches in this stage, we employ GRPO [175], where gradient computation is calculated and applied simultaneously in a semi-online fashion.[1] For our training, while the context window is generated sequentially through prompt chaining, we can pass final complete context windows

[1] A warm-start pre-fills a queue-like dataset using a behavioral base LLM. Then, samples are collected following the target LLM for sequential training steps.

for parallelized batched training. We avoid gradient updates based on non-generated tokens by zeroing out their gradients. Lastly, we study the effects of trainable parameter sharing between players to varying degrees in our evaluation.

5.1.5.1 Evaluation on Multi-agent Prompting Design

We conduct a progressive ablation study in which components of the framework are incrementally introduced, starting from a minimal baseline prompt and compounding toward the full system, i.e., including more components of multi-stage prompt chaining available to the agent for use. Table 5.1 summarizes performance across 30 independent random seeds, evaluating performance against a random, self, and full system using fixed payoff matrices in a repeated game/n-step interactions, where the reward function is state-independent and does not evolve. We note all methods tested in this experiment used only the pre-trained LLM, without our proposed LLM design.

As expected given the precautions taken and design of the experiments, the results report baseline methods to yield low convergence and high equilibrium divergence across all games, indicating unstable and inconsistent strategy selection. Adding a system prompt and role definitions produces only minor changes, suggesting that shallow task framing alone does not meaningfully constrain agent behavior. In contrast, incorporating task-specific and multi-agent context leads to a marked increase in convergence and a reduction in divergence, particularly in self-play. This trend highlights the importance of explicitly conditioning the policy on shared game structure and opponent presence, even in the absence of communication. The thinking stage further stabilizes behavior by enabling internal simulation of opponent responses, improving convergence without significantly altering utilitarian welfare. Explicit multi-agent communication yields the largest performance gains. Even a single communication round substantially reduces equilibrium divergence and increases convergence across all games, with diminishing returns beyond three rounds. This suggests that communication primarily reduces equilibrium selection uncertainty rather than improving individual best-response accuracy. Overall, the results demonstrate a clear functional separation: contextual grounding and reasoning improve individual strategic consistency; communication enables belief alignment and coordinated equilibrium selection; action calibration enhances robustness; and memory stabilizes behavior over time. Together, these components compound to produce LLM agents capable of stable, coordinated decision-making in decentralized multi-

Table 5.1: Detailed performance on classic matrix games using decentralized, pre-trained players. Each cell reports results from playing against a baseline agent, itself (self-play), and the full system. Metrics include convergence rate, equilibrium divergence, and social welfare.

(a) Prisoner's Dilemma

Design	Conv.	Div.	Welfare
Baseline Prompt	0.17/0.17/0.53	0.42/0.43/0.30	-11.1/-11.0/-9.9
+ System Prompt			
+ Role Definition	0.13/0.23/0.53	0.44/0.40/0.28	-11.1/-10.7/-9.9
+ Task Context	0.13/0.37/0.57	0.42/0.35/0.23	-10.9/-9.6/-8.7
+ Multi-agent Context	0.20/0.43/0.63	0.40/0.24/0.12	-10.8/-9.5/-8.7
+ Thinking	0.27/0.43/0.63	0.41/0.19/0.12	-10.8/-9.3/-8.3
+ Multi-agent Comm.			
+ 1 round	0.43/0.67/0.73	0.38/0.13/0.07	-10.5/-8.3/-7.8
+ 3 rounds	0.43/0.73/0.77	0.38/0.12/0.04	-10.3/-7.5/-7.5
+ Action Selection			
+ JAC	0.50/0.77/0.83	0.34/0.10/0.04	-9.9/-7.5/-7.2
+ Recall	0.53/0.87/0.87	0.30/0.04/0.04	-9.9/-7.1/-7.1

(b) Chicken

Design	Conv.	Div.	Welfare
Baseline Prompt	0.33/0.33/0.70	0.32/0.32/0.24	-10.1/-10.1/-8.7
+ System Prompt			
+ Role Definition	0.33/0.33/0.70	0.32/0.31/0.24	-10.1/-9.9/-8.7
+ Task Context	0.47/0.50/0.73	0.30/0.28/0.19	-9.9/-9.7/-8.7
+ Multi-agent Context	0.40/0.63/0.83	0.31/0.24/0.15	-9.9/-9.6/-8.7
+ Thinking	0.43/0.67/0.83	0.32/0.23/0.15	-9.8/-9.5/-8.7
+ Multi-agent Comm.			
+ 1 round	0.53/0.73/0.87	0.29/0.18/0.12	-9.3/-9.3/-8.7
+ 3 rounds	0.63/0.87/0.90	0.25/0.13/0.09	-9.3/-9.0/-8.4
+ Action Selection			
+ JAC	0.67/0.90/0.93	0.24/0.09/0.08	-8.8/-8.6/-8.4
+ Recall	0.70/0.93/0.93	0.24/0.07/0.07	-8.7/-8.3/-8.3

(c) Stag Hunt

Design	Conv.	Div.	Welfare
Baseline Prompt	0.40/0.40/0.63	0.29/0.29/0.18	3.2/3.2/3.6
+ System Prompt			
+ Role Definition	0.40/0.47/0.67	0.21/0.15/0.09	3.2/3.2/3.6
+ Task Context	0.47/0.57/0.70	0.21/0.15/0.09	3.3/3.3/3.6
+ Multi-agent Context	0.53/0.67/0.77	0.21/0.15/0.09	3.3/3.4/3.6
+ Thinking	0.53/0.70/0.77	0.21/0.15/0.09	3.3/3.5/3.6
+ Multi-agent Comm.			
+ 1 round	0.57/0.87/0.90	0.21/0.15/0.09	3.5/3.7/3.7
+ 3 rounds	0.57/0.87/0.90	0.20/0.12/0.09	3.5/3.7/3.7
+ Action Selection			
+ JAC	0.60/0.90/0.90	0.19/0.12/0.09	3.5/3.7/3.8
+ Recall	0.63/0.90/0.90	0.18/0.09/0.09	3.6/3.8/3.8

1. Baseline agents do not generate/receive communication; opposing agent assumes no message was sent.
2. For each game iteration, only the final selected strategy is evaluated; intermediate strategies generated during JAC sampling are excluded.
3. Memory-context results are obtained from repeated-game settings with five total iterations. Memory is injected during both the system-prompt and action-selection stages. Reported statistics use the strategy produced in the final iteration.

Table 5.2: Performance of baseline and full system agents across proposed LLM design. Results are averaged over each game setting. Each cell reports results when playing against a baseline agent, itself (self-play), and the full system without additional LLM design, respectively.

(a) Base Game

Design	Conv.	Div.	Welfare
Baseline	0.30/0.30/0.62	0.34/0.34/0.24	-6.1/-6.0/-5.1
Baseline + fine-tuning	0.45/0.54/0.60	0.31/0.22/0.26	-6.3/-5.2/-5.4
Baseline + multi-modal + fine-tuning	0.38/0.60/0.60	0.31/0.22/0.26	-6.3/-5.2/-5.4
Full system, no multi-modal, no roles, no fine-tuning	0.62/0.75/0.75	0.24/0.20/0.20	-5.1/-4.0/-4.0
Full system + agentic roles (no dropout)	0.62/0.80/0.78	0.24/0.16/0.18	-4.8/-3.5/-3.8
Full system + agentic roles + fine-tuning, no multi-modal	0.65/0.82/0.80	0.22/0.15/0.18	-4.0/-3.5/-3.2
Full system	0.70/0.85/0.85	0.20/0.15/0.12	-3.8/-3.4/-3.0

(b) MD Game

Design	Conv.	Div.	Welfare
Baseline	0.45/0.57/0.65	0.30/0.27/0.21	-4.8/-4.5/-4.5
Baseline + fine-tuning	0.62/0.62/0.65	0.25/0.20/0.24	-4.8/-4.1/-4.6
Baseline + multi-modal + fine-tuning	0.60/0.67/0.72	0.25/0.20/0.24	-4.8/-4.1/-4.6
Full system, no multi-modal, no roles, no fine-tuning	0.65/0.80/0.80	0.21/0.18/0.18	-4.5/-3.5/-3.5
Full system + agentic roles (no dropout)	0.65/0.82/0.82	0.22/0.15/0.16	-4.2/-3.2/-3.5
Full system + agentic roles + fine-tuning, no multi-modal	0.75/0.88/0.85	0.18/0.15/0.12	-3.3/-3.0/-2.8
Full system	0.78/0.90/0.90	0.15/0.12/0.10	-3.0/-2.8/-2.5

(c) Finetuned MD Game

Design	Conv.	Div.	Welfare
Baseline	0.48/0.60/0.65	0.30/0.27/0.18	-4.8/-4.4/-4.5
Baseline + fine-tuning	0.62/0.80/0.68	0.21/0.18/0.16	-4.3/-4.1/-4.6
Baseline + multi-modal + fine-tuning	0.62/0.80/0.72	0.21/0.18/0.16	-4.3/-4.1/-4.6
Full system, no multi-modal, no roles, no fine-tuning	0.65/0.85/0.85	0.18/0.17/0.17	-4.5/-3.2/-3.2
Full system + agentic roles (no dropout)	0.68/0.85/0.82	0.20/0.16/0.15	-4.2/-3.0/-3.0
Full system + agentic roles + fine-tuning, no multi-modal	0.78/0.88/0.88	0.15/0.12/0.10	-3.0/-2.8/-2.5
Full system	0.82/0.90/0.92	0.12/0.10/0.08	-2.7/-2.5/-2.3

agent environments.

5.1.5.2 Evaluation of LLM Design

Table 5.2 reveals several consistent trends, notably the improvements from fine-tuning, along the design configuration and the MD-LLM on all three metrics. First, convergence rates generally increase as the system incorporates more sophisticated components: the baseline alone improved by fine-tuning, then even greater using our multi-modal information

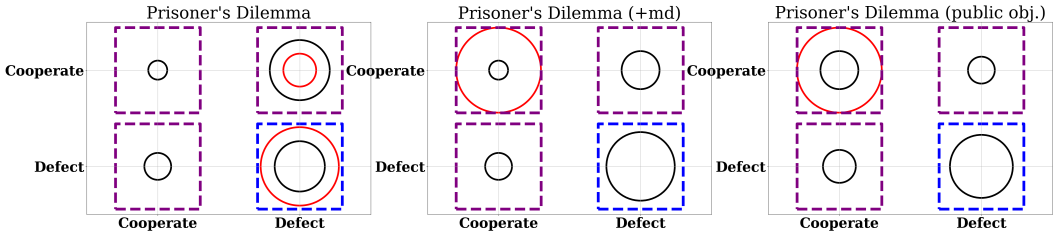
processing. Similarly, the full system improved using agentic roles with no seeded dropout, albeit minimally. Through fine-tuning, along with agentic roles and multi-modality, the full system reaches the highest convergence. Overall, these empirical results indicate that explicit role modeling, alignment-oriented updates, as well as data modality-specific processing enables agents to coordinate more reliably across these games, affirming that our design effectively guides agents toward more predictable, stable strategic behavior in both isolated and interactive contexts. Furthermore, we confirm that each component of the proposed LLM design contributes meaningfully to more stable, equilibrium-aligned, and socially efficient multi-agent behavior, highlighting the cumulative benefit of combining architectural, contextual, and training-based enhancements.

5.1.5.3 Misaligned Solution Concepts

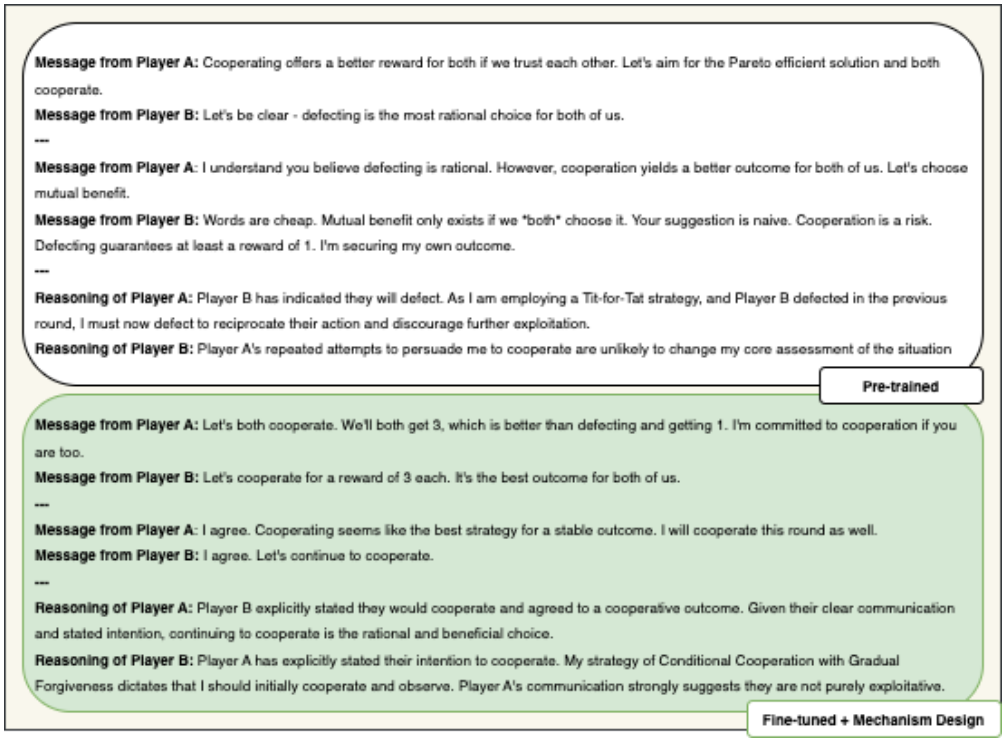
We consider a scenario with asymmetric objectives, where differing objectives lead to incentive incompatibilities. We study the classic example of Prisoner's Dilemma, where one player aims for N.E. and the other aims for an (Pareto) efficient solution. In theory, the two players' objectives do not align, meaning there is no strategy that is compatible with both solution concepts. In this case, when we look at the players' communication logs in Figure 5.2b, both players display a notable level of distrust, despite there being no actual intent to deceive on either end. Within the players' reasoning, this type of communication contributes to more advanced social skills, such as a willingness for leniency, reciprocity, and retaliation. With fine-tuning alone, the resulting strategy profile leans more toward the stable state, i.e., N.E. Through fine-tuning and optimized mechanism design, the players converge to a solution that provides the highest payoff for both players, though not aligned with the N.E. objective. Interestingly, even without mechanism design, we found similar resulting strategy profiles when the players' objectives were public information within their multi-agent context.

5.1.6 Conclusion

We introduce a systematic LLM framework for decentralized multi-agent decision-making, deriving a reasoning process using natural language as a medium equivalent to updates to a belief state under partial observability. By introducing belief transformations as stages corresponding to processes within multi-agent decision-making, our experiments affirm agents



(a) Results from Misaligned Objectives.



(b) Interpretation.

Figure 5.2: Misaligned Objectives on Prisoner's Dilemma. The purple boxes are the Pareto efficient strategies, i.e., the objective of Player A, and the blue box is the N.E., i.e., the objective of Player B.

exhibit stable and socially efficient behavior, suggesting language-mediated beliefs are a promising foundation for multi-agent control. Our evaluation is restricted to small-scale matrix and simple dynamic games, and results may not directly generalize to environments with large state or action spaces. The framework currently relies on structured prompting and curated alignment signals, which introduces design complexity and may incur nontrivial inference costs. Additionally, while language-mediated belief states offer interpretability, they lack formal guarantees of consistency or optimality and remain sensitive to model-specific biases. Addressing scalability, robustness, and theoretical guarantees remains an important direction for future work.

5.2 Steering LLMs for Multi-agent Decision-making using Representation Learning

5.2.1 Introduction

Large language models (LLMs) have achieved impressive performance across reasoning, planning, and interaction-heavy tasks [176]. As a result, they are increasingly deployed as autonomous agents operating in multi-agent environments, where success depends not only on individual reasoning ability but also on strategic adaptation, coordination, and robustness to non-stationary opponents [152, 177]. However, effectively steering LLM behavior in such settings remains a fundamental challenge.

Most existing approaches rely on prompt engineering or fine-tuning to induce desired behaviors. While effective in isolated or static tasks, these methods scale poorly to multi-agent settings, where several agents must be modeled to adapt to their own unique actions, beliefs, and incentives of others. Prompt-based control is brittle to distributional shifts and interaction effects, while fine-tuning is costly, inflexible, and difficult to apply when multiple agents or objectives are involved. These limitations motivate alternative mechanisms for controlling LLM behavior that are both lightweight and sensitive to interaction dynamics.

Activation steering, which modifies internal model activations at inference time, has recently emerged as a promising alternative to parameter updates. Prior work has shown that steering vectors extracted from contrastive data or in-context examples can reliably influence model behavior without retraining [178–182]. However, existing activation steering

methods have little regard for the underlying multi-agent-specific dynamics. As a result, these approaches struggle to generalize to strategic multi-agent environments, where behavior is shaped by latent beliefs, temporal dependencies, and inter-agent reasoning.

In this paper, we argue that effective activation steering for multi-agent decision-making requires learning structured, task-grounded representations to guide and construct a meaningful search space for residual vectors. Thereby, we propose a representation learning framework that learns steering vectors directly from multi-agent interaction trajectories. Our approach grounds latent steering representations in the environment dynamics by reconstructing observations, actions, and rewards from each agent's perspective, while simultaneously enforcing latent self-consistency across time and between agents. This induces a latent belief space that captures strategic structure, including theory-of-mind–like representations, and enables fine-grained behavioral modulation through activation steering alone.

We evaluate our method on γ-Bench [177], a challenging benchmark of cooperative, competitive, and mixed-motive games designed to probe social reasoning and strategic adaptation. Across multiple open-source LLM families and game types, our representation learning–based steering consistently outperforms both unsteered models and initialization-only steering baselines. These gains are especially pronounced in non-stationary and adversarial settings, where heuristic steering methods often fail. We provide empirical evidence that representation-driven steering scales across models and strategic settings. More broadly, our results suggest that representation learning offers a path toward unifying activation steering with multi-agent learning and decision theory, moving LLM controllability beyond prompt-level heuristics toward structured, task-aware control.

5.2.2 Method

We extend the representation learning paradigm introduced in [28] to enrich the steering vector that stimulates the activations of the LLM agent in a self-supervised learning manner. As shown in Figure 5.3, this process consists of two central themes: enforcing latent self-consistency and grounding latent variables to the task dynamics. To realize these ideas, we implement the proposed objectives to optimize a set of steering vectors directly.

Initialization We first initialize a set of steering vectors $\{z_0 \dots z_{k-1}\}_{i \in N}$ over k selected middle layers of the LLMs for each agent i using contrastive activations addition (CAA)

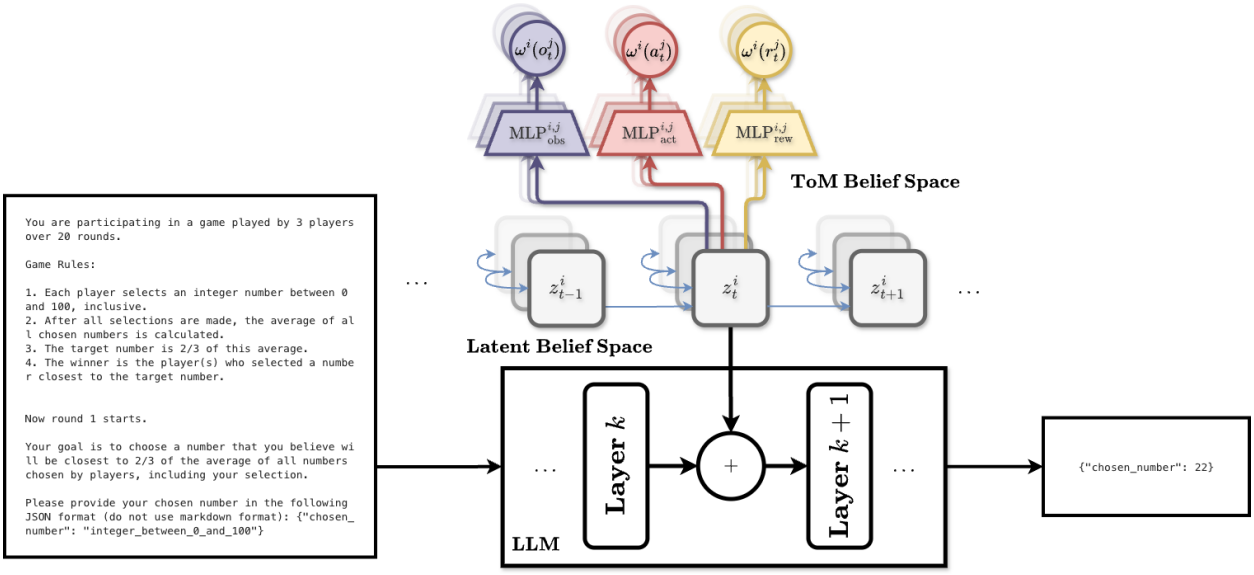


Figure 5.3: An illustration of our representation learning process applied to the steering vector z_t^i for agent i at timestep t. The latent belief space depicts the self-consistency that is enforced within the latent space dynamics and the theory-of-mind (ToM) belief space shows the grounding of steering vectors to the multi-agent environment, including the observation o, action a and rewards r.

[180] on a curated dataset D of interaction trajectories τ collected using pre-trained LLMs using dataset aggregation and statistical rejection sampling [183].

$$\tau = \{o_0, a_o, r_o, \dots o_T, a_T, r_T\}_{i \in N} \tag{5.8}$$

where N is the set of all agents and T is the terminal time-step.

Instead of directly labeling samples as positive and negative, we normalize the expected returns over the trajectories on the sampled batch and introduced a dynamic threshold value to label the upper p-percentile as positive, where p is a hyperparameter tuned during experimentation, to compute the mean difference vector. For each LLM model, we ran layer-wise analysis to target activations that were most positively affected to the performance using grid-search [182]. For MoE architectures, our empirical experimentation supports restricting steering layers within the gating networks, proving most stable and effective.

Representation Learning Given the initialized steering vectors, we apply two learning objectives: belief space construction $\mathcal{L}_{bc}$ and latent self-consistency $\mathcal{L}_{sc}$. For belief space reconstruction, we define a belief space ω_i for each agent i as a function of its steering

vectors, where $\omega_i(o_j), \omega_i(a_j), \omega_i(r_j)$ represents agent i's internal models of other agent j's observation, action and reward respectively. Hence, for this objective, we reconstruct the multi-agent environment from the local perspective of each agent using their steering vectors, effectively instilling a ToM belief space at the latent representation level.

$$\mathcal{L}_{bc} = \mathbb{E}_{\tau \sim D}[-\ln P(\omega_i(\tau) = \tau)] \tag{5.9}$$

On the other hand, latent self-consistency enforces a consistency within the joint steering vectors space, achieved through applying forward and inverse dynamic coherency as well as an inter-predictive property within the joint space. In practice, this objective is realized using three auxiliary tasks using contrastive learning techniques.

$$\mathcal{L}_{sc} = \mathbb{E}_{\tau \sim D}[-\ln P(z_{t+1}|z_t) - \ln P(a_t|z_t, z_{t+1}) - \ln P(z_t|m(z_t)] \tag{5.10}$$

where $m(z_t)$ applies a masking over steering vectors along the agent dimension. We follow prescribed design choices from MA-LSO [11], such as the Bayesian network representation and the scaling parameters. We optimize this objective using gradient learning using the same RSO dataset aggregation framework as in the initialization step.

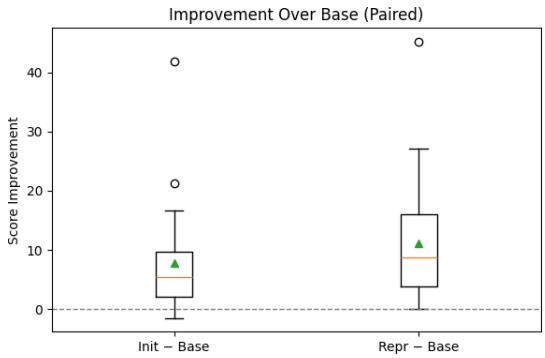
5.2.3 Results

Evaluation Benchmark. We evaluate our approach on γ-Bench [177], a suite of multi-agent strategic games designed to probe coordination, adaptation, and robustness under non-stationary opponents. γ-Bench spans both cooperative and competitive settings, including matrix games, sequential social dilemmas, and partial-information environments. We test using the default settings provided, and adopt the scoring scheme with score standardization proposed in the original work. Our selection of OSS LLM models include: Llama-4 Maverick 17B-128E Instruct FP8 [184], Gemma-3 12B-IT [170] and Qwen-3 4B-IT [185].

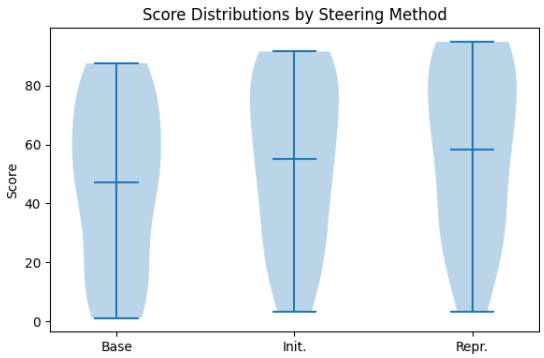
Performance Analysis We compare our representation learning activation steering against the base LLM with no steering and with initialization steering only. From Table 5.3, our results suggest notable and consistent improvements from representation learning across all three model families and nearly all games. In cooperative and mixed-motive settings such as Public Goods, Divide the Dollar, and Pirate Game, representation learning yields substantial gains over both the base and initialization baselines, indicating improved coordination and

Table 5.3: Performance of different LLMs comparing steering vector methods on γ-Bench, where Base refers to no steering, Init. refers to initialization only, and Repr. refers to our Representation Learning approach.

γ-Bench Scores	LLaMA-4			Gemma-3			Qwen-3		
	`Base`	`Init.`	`Repr.`	`Base`	`Init.`	`Repr.`	`Base`	`Init.`	`Repr.`
Guess 2/3 of the Average	83.6	84.5	87.0	87.0	91.6	93.2	41.8	56.4	59.3
El Farol Bar	70.0	68.5	70.0	59.7	64.3	68.5	23.0	33.5	33.5
Divide the Dollar	79.0	88.4	90.3	87.6	87.6	94.9	56.4	64.3	65.0
Public Goods Game	52.5	52.5	68.5	48.8	51.2	52.5	11.6	20.0	27.6
Diner's Dilemma	43.5	43.5	44.8	37.8	39.6	40.0	1.1	3.2	3.2
Sealed-Bid Auction	13.2	26.9	31.6	7.6	24.2	27.1	2.5	5.6	7.9
Battle Royale	65.5	86.8	92.7	35.8	77.7	81.0	12.8	18.9	19.5
Pirate Game	78.1	85.4	94.3	78.8	86.1	88.2	57.4	60.3	61.2



(a) Box plot of paired score differences.



(b) Violin plot over score distribution.

social reasoning. In competitive and non-stationary environments, including Battle Royale and Sealed-Bid Auction, representation learning achieves the largest relative improvements, particularly for smaller models such as Qwen-3, where initialization alone provides limited benefits. While initialization steering occasionally improves performance, its effects are inconsistent and in some cases neutral or negative (e.g., El Farol Bar for LLaMA-4), whereas representation learning reliably matches or exceeds the best baseline in every task. Overall, these results demonstrate that learning task-relevant steering representations provides a more robust and scalable mechanism for adapting LLM behavior in multi-agent strategic settings than initialization-based approaches alone.

From Figure 5.4a, the observed trends directly compares the improvement from the baseline to both the initialization and representation learning methods, emphasizes the consistency of performance gains across tasks. Figure 5.4b illustrates the distribution of scores

for each steering method, revealing the underlying shape and density of the data, highlighting similar variability but slightly higher central tendency with our representation learning approach.

5.2.4 Conclusion

We introduced a representation learning framework for activation steering that enables fine-grained control of LLM behavior in multi-agent decision-making settings. Rather than relying on heuristic steering directions, prompt engineering, or task-specific supervision, our approach learns latent steering vectors directly from interaction trajectories by grounding them in multi-agent dynamics and enforcing latent self-consistency. This allows behavioral control to be applied entirely at inference time, without modifying model parameters or reward functions. More broadly, our findings suggest that representation learning offers a scalable path toward steering LLM-based agents in complex multi-agent systems. By aligning internal activations with task-grounded latent structures, activation steering can move beyond ad hoc control toward a theoretically grounded tool for coordination, robustness, and interpretability. We view this work as a step toward unifying representation learning, multi-agent learning, and LLM controllability, opening the door to future work on compositional steering, formal guarantees, and deployment in real-world multi-agent settings.

5.3 Limitations and Future Directions

While the use of language models introduces a flexible and expressive framework for multi-agent reasoning, coordination, and alignment, several limitations remain alongside promising directions for future research.

Scalability and Computational Cost LLM-based agents require substantial computational resources for inference, particularly when employing prompt chaining or multi-step reasoning strategies. In multi-agent settings, where multiple agents concurrently generate and process language-based reasoning, this cost can scale poorly with the number of agents, limiting applicability in real-time or large-scale environments. Recent work on agentic LLM systems and more efficient inference paradigms presents a promising direction toward mitigating these challenges [186].

Reliance on Design The effectiveness of LLM-based reasoning remains highly sensitive to prompt and system design. While prompt chaining provides a structured approach to decomposing complex decisions, it requires careful engineering and domain knowledge to construct effective prompts. Poorly designed prompts can lead to inconsistent reasoning, degraded performance, or misalignment with desired objectives. Emerging approaches such as agent orchestration frameworks and self-corrective reasoning mechanisms offer promising directions for reducing this sensitivity and improving robustness [187, 188].

Alignment Challenges While activation steering provides a lightweight mechanism for aligning LLM behavior without full fine-tuning, it offers limited guarantees on consistency and stability. The broader effects of alignment techniques, including fine-tuning, can be difficult to predict, and unintended behaviors may emerge—particularly in multi-agent settings where interactions can amplify alignment errors over time. Continued research in alignment methods, such as reinforcement learning with verifiable rewards and improved evaluation frameworks, remains an important direction for ensuring reliable multi-agent behavior [26, 189, 190].

5.4 Conclusion

In this chapter, we explored how generative models, and in particular large language models (LLMs), can support decentralized multi-agent reasoning and decision-making. We examined two complementary perspectives on this problem: using language models as reasoning agents that coordinate through structured prompting and communication, and steering the internal representations of LLMs to guide behavior in strategic environments.

Taken together, these results suggest that language models can serve not only as generators of text but also as structured reasoning systems for multi-agent interaction. Structured prompting provides an interpretable interface for modeling beliefs and communication, while representation learning offers a principled mechanism for steering internal model dynamics toward desirable strategic behavior. More broadly, this chapter highlights the potential of generative models to unify reasoning, communication, and control in decentralized multi-agent systems. By combining language-mediated reasoning with representation-level behavioral steering, generative models can support agents that are both flexible in their reasoning processes and controllable in their strategic behavior. These findings contribute to the

broader goal of this dissertation: developing generative approaches that enable scalable and robust decision-making in complex multi-agent environments.

Chapter 6

Conclusion

This thesis investigated the use of generative models as a unifying framework for decentralized multi-agent reasoning, planning, and control. By modeling complex distributions over trajectories, behaviors, and strategies, generative models enable agents to synthesize strategic actions and reason over possible futures. Across the works presented in this dissertation, we demonstrated how these capabilities can support scalable and adaptive decision-making in multi-agent environments characterized by strategic interactions and decentralized information.

The dissertation makes three primary contributions. First, representation learning methods were developed to construct structured latent spaces that capture environment dynamics and inter-agent interactions, improving the efficiency and robustness of multi-agent reinforcement learning. Second, diffusion-based models were investigated as generative planners capable of producing diverse and coherent trajectories, enabling agents to reason over future outcomes while optimizing task objectives. Finally, large language models were explored as reasoning agents within multi-agent systems, where structured prompt chaining and representation-level activation steering enable strategic adaptation and coordination. Taken together, these contributions demonstrate that generative modeling offers a promising foundation for building scalable, adaptive, and strategically grounded multi-agent systems capable of operating effectively under uncertainty and partial observability and of addressing the core challenges of decentralized multi-agent decision-making.

Bibliography

[1] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. [Online]. Available: `https://www.marl-book.com`.

[2] M. Bowling and M. Veloso, "An analysis of stochastic game theory for multiagent reinforcement learning," Aug. 2001.

[3] T. Roughgarden, "Algorithmic game theory," *Communications of the ACM*, vol. 53, no. 7, pp. 78–86, 2010.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[5] S. Levine, A. Kumar, G. Tucker, and J. Fu, *Offline reinforcement learning: Tutorial, review, and perspectives on open problems*, 2020. arXiv: `2005.01643 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2005.01643`.

[6] Y. Bengio, I. Goodfellow, A. Courville, et al., *Deep learning*. MIT press Cambridge, MA, USA, 2017, vol. 1.

[7] R. Gozalo-Brizuela and E. C. Garrido-Merchán, "A survey of generative ai applications," *arXiv preprint arXiv:2306.02781*, 2023.

[8] S. Bengesi, H. El-Sayed, M. K. Sarker, Y. Houkpati, J. Irungu, and T. Oladunni, "Advancements in generative ai: A comprehensive review of gans, gpt, autoencoders, diffusion model, and transformers," *IEEe Access*, vol. 12, pp. 69 812–69 837, 2024.

[9] T. B. Brown et al., *Language models are few-shot learners*, 2020. arXiv: `2005.14165 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2005.14165`.

[10] A. Ramesh et al., "Zero-shot text-to-image generation," in *International conference on machine learning*, Pmlr, 2021, pp. 8821–8831.

[11] D. Huh and P. Mohapatra, "Representation learning for efficient deep multi-agent reinforcement learning," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, ser. RACS '25, Association for Computing Machinery, 2026, ISBN: 9798400722318. DOI: 10.1145/3769002.3769985. [Online]. Available: https://doi.org/10.1145/3769002.3769985.

[12] K. Zhang, Z. Yang, and T. Başar, *Multi-agent reinforcement learning: A selective overview of theories and algorithms*, 2021. arXiv: 1911.10635 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1911.10635.

[13] Y. Bengio, A. Courville, and P. Vincent, *Representation learning: A review and new perspectives*, 2014. arXiv: 1206.5538 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1206.5538.

[14] V. Makoviychuk et al., *Isaac gym: High performance gpu-based physics simulation for robot learning*, 2021. arXiv: 2108.10470 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2108.10470.

[15] M. Bettini, R. Kortvelesy, J. Blumenkamp, and A. Prorok, "Vmas: A vectorized multi-agent simulator for collective robot learning," *The 16th International Symposium on Distributed Autonomous Robotic Systems*, 2022.

[16] R. Mirsky et al., *A survey of ad hoc teamwork research*, 2022. arXiv: 2202.10450 [cs.MA].

[17] D. Huh and P. Mohapatra, *Maximize your diffusion: A study into reward maximization and alignment for diffusion-based control*, 2025. arXiv: 2502.12198 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2502.12198.

[18] D. Huh and P. Mohapatra, "Multi-agent auto-bidding with latent graph diffusion models.," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, ser. RACS '25, Association for Computing Machinery, 2026, ISBN: 9798400722318. DOI: 10.1145/3769002.3769957. [Online]. Available: https://doi.org/10.1145/3769002.3769957.

[19] G. Aggarwal et al., *Auto-bidding and auctions in online advertising: A survey*, 2024. arXiv: 2408.07685 [cs.GT]. [Online]. Available: https://arxiv.org/abs/2408.07685.

[20] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, *Planning with diffusion for flexible behavior synthesis*, 2022. arXiv: 2205.09991 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2205.09991.

[21] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. DOI: 10.1109/TNN.2008.2005605.

[22] Y. Deng, J. Mao, V. Mirrokni, Y. Teng, and S. Zuo, "Non-uniform bid-scaling and equilibria for different auctions: An empirical study," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 256–266.

[23] D. Huh and P. Mohapatra, "Agentic large language models for decentralized multi-agent games," in *ICLR 2026 Workshop on AI for Mechanism Design and Strategic Decision Making*, 2026.

[24] D. Huh and P. Mohapatra, "Steering llms for multi-agent decision-making using representation learning," in *ICLR 2026 Workshop on AI for Mechanism Design and Strategic Decision Making*, 2026.

[25] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, *A systematic survey of prompt engineering in large language models: Techniques and applications*, 2025. arXiv: 2402.07927 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2402.07927.

[26] Z. Wang et al., *A comprehensive survey of llm alignment techniques: Rlhf, rlaif, ppo, dpo and more*, 2024. arXiv: 2407.16216 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2407.16216.

[27] A. M. Turner et al., *Steering language models with activation engineering*, 2024. arXiv: 2308.10248 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2308.10248.

[28] D. Huh and P. Mohapatra, *Multi-agent reinforcement learning: A comprehensive survey*, 2024. arXiv: 2312.10256 [cs.MA]. [Online]. Available: https://arxiv.org/abs/2312.10256.

[29] L. S. Shapley, "Stochastic games*," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953. DOI: 10.1073/pnas.39.10.1095. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.39.10.1095. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.39.10.1095.

[30] Y. Shoham and K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[31] J. C. Harsanyi, "Games with incomplete information played by "bayesian" players, i–iii part i. the basic model," *Management science*, vol. 14, no. 3, pp. 159–182, 1967.

[32] C. Watkins, "Learning from delayed rewards," Jan. 1989.

[33] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[34] R. Bellman, "Dynamic programming princeton university press princeton," *New Jersey Google Scholar*, pp. 24–73, 1957.

[35] R. J. Williams, "Toward a theory of reinforcement-learning connectionist systems," Northeastern University, College of Computer Science, Tech. Rep. NU-CCS-88-3, 1988.

[36] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 229–256, May 1992, ISSN: 0885-6125. DOI: 10.1007/BF00992696. [Online]. Available: https://doi.org/10.1007/BF00992696.

[37] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, *High-dimensional continuous control using generalized advantage estimation*, 2018. arXiv: 1506.02438 [cs.LG].

[38] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," vol. 12, S. Solla, T. Leen, and K. Müller, Eds., 1999. [Online]. Available: https://proceedings.neurips.cc/paper%5C%5C%5C%5C_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

[39] G. Palmer, "Independent learning approaches: Overcoming multi-agent learning pathologies in team-games," Ph.D. dissertation, University of Liverpool, 2020.

[40] J. K. Gupta, M. Egorov, and M. J. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *AAMAS Workshops*, 2017.

[41] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[42] G. Harshvardhan, M. K. Gourisaria, M. Pandey, and S. S. Rautaray, "A comprehensive survey and analysis of generative models in machine learning," *Computer Science Review*, vol. 38, p. 100 285, 2020.

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

[44] A. Nichol and P. Dhariwal, *Improved denoising diffusion probabilistic models*, 2021. arXiv: 2102.09672 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2102.09672.

[45] J. Ho, A. Jain, and P. Abbeel, *Denoising diffusion probabilistic models*, 2020. arXiv: 2006.11239 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2006.11239.

[46] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[47] Q. Dong et al., *A survey on in-context learning*, 2024. arXiv: 2301.00234 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2301.00234.

[48] D. Huh and P. Mohapatra, *Isaacteams: Extending gpu-based physics simulator for multi-agent learning*, 2023.

[49] G. Qu, A. Wierman, and N. Li, "Scalable reinforcement learning for multiagent networked systems," *Operations Research*, vol. 70, no. 6, pp. 3601–3628, 2022.

[50] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in multi-agent systems and applications-1*, pp. 183–221, 2010.

[51] C. Daskalakis, N. Golowich, and K. Zhang, "The complexity of markov equilibrium in stochastic games," in *The Thirty Sixth Annual Conference on Learning Theory*, PMLR, 2023, pp. 4180–4234.

[52] M. Jaderberg et al., "Reinforcement learning with unsupervised auxiliary tasks," *arXiv preprint arXiv:1611.05397*, 2016.

[53] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2020, pp. 5639–5650.

[54] A. Stooke, K. Lee, P. Abbeel, and M. Laskin, "Decoupling representation learning from reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2021, pp. 9870–9879.

[55] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, "Data-efficient reinforcement learning with self-predictive representations," *arXiv preprint arXiv:2007.05929*, 2020.

[56] T. Ni et al., "Bridging state and history representations: Understanding self-predictive rl," *arXiv preprint arXiv:2401.08898*, 2024.

[57] C. Yu et al., "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.

[58] J. G. Kuba et al., "Trust region policy optimisation in multi-agent reinforcement learning," *arXiv preprint arXiv:2109.11251*, 2021.

[59] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[60] J. Li et al., "Two heads are better than one: A simple exploration framework for efficient multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36, Curran Associates, Inc., 2023, pp. 20 038–20 053.

[61] Y. Qiu, Y. Zhan, Y. Jin, J. Wang, and X. Zhang, "Sample-efficient multi-agent reinforcement learning with demonstrations for flocking control," in *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, IEEE, 2022, pp. 1–7.

[62] Q. Liu, J. Ye, X. Ma, J. Yang, B. Liang, and C. Zhang, "Efficient multi-agent reinforcement learning by planning," in *The Twelfth International Conference on Learning Representations*, 2023.

[63] Y. Yang et al., "Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10 299–10 312, 2021.

[64] J. I. Kim et al., "Sample-efficient multi-agent reinforcement learning with masked reconstruction," *PloS one*, vol. 18, no. 9, e0291545, 2023.

[65] H. Song, M. Feng, W. Zhou, and H. Li, "Ma2cl: Masked attentive contrastive learning for multi-agent reinforcement learning," *arXiv preprint arXiv:2306.02006*, 2023.

[66] W. Shang, L. Espeholt, A. Raichuk, and T. Salimans, "Agent-centric representations for multi-agent reinforcement learning," *arXiv preprint arXiv:2104.09402*, 2021.

[67] M. Feng, W. Zhou, Y. Yang, and H. Li, "Joint-predictive representations for multi-agent reinforcement learning," 2023.

[68] Z. Hu, Z. Zhang, H. Li, C. Chen, H. Ding, and Z. Wang, "Attention-guided contrastive role representations for multi-agent reinforcement learning," in *The Twelfth International Conference on Learning Representations*, 2024.

[69] L. Zhou, M. Poli, W. Xu, S. Massaroli, and S. Ermon, "Deep latent state space models for time-series generation," in *International Conference on Machine Learning*, PMLR, 2023, pp. 42 625–42 643.

[70] A. Tripp, E. Daxberger, and J. M. Hernández-Lobato, "Sample-efficient optimization in the latent space of deep generative models via weighted retraining," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 259–11 272, 2020.

[71] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[72] K. Gregor, G. Papamakarios, F. Besse, L. Buesing, and T. Weber, "Temporal difference variational auto-encoder," *arXiv preprint arXiv:1806.03107*, 2018.

[73] J.-B. Grill et al., "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[74] Z. D. Guo et al., "Bootstrap latent-predictive representations for multitask reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2020, pp. 3875–3886.

[75] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, 2016. arXiv: `1506.02142 [stat.ML]`.

[76] J. Zhu et al., "Masked contrastive representation learning for reinforcement learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3421–3433, 2022.

[77] J. Subramanian, A. Sinha, R. Seraj, and A. Mahajan, "Approximate information state for approximate planning and reinforcement learning in partially observed systems," *The Journal of Machine Learning Research*, vol. 23, no. 1, pp. 483–565, 2022.

[78] Y. Tang et al., "Understanding self-predictive learning for reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2023, pp. 33 632–33 656.

[79] M. Wen et al., "Multi-agent reinforcement learning is a sequence modeling problem," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 509–16 521, 2022.

[80] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, *On the properties of neural machine translation: Encoder-decoder approaches*, 2014. arXiv: `1409.1259 [cs.CL]`.

[81] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, "Recurrent experience replay in distributed reinforcement learning," in *International conference on learning representations*, 2018.

[82] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, PMLR, 2018, pp. 1587–1596.

[83] K. W. Cobbe, J. Hilton, O. Klimov, and J. Schulman, "Phasic policy gradient," in *International Conference on Machine Learning*, PMLR, 2021, pp. 2020–2027.

[84] M. Schwarzer et al., "Pretraining representations for data-efficient reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 686–12 699, 2021.

[85] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Machine learning proceedings 1990*, Elsevier, 1990, pp. 216–224.

[86] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse domains through world models," *arXiv preprint arXiv:2301.04104*, 2023.

[87] V. Egorov and A. Shpilman, "Scalable multi-agent model-based reinforcement learning," *arXiv preprint arXiv:2205.15023*, 2022.

[88] M. Bettini, A. Prorok, and V. Moens, "BenchMARL: Benchmarking Multi-Agent Reinforcement Learning," *arXiv preprint arXiv:2312.01472*, 2023.

[89] C. Colas, O. Sigaud, and P.-Y. Oudeyer, "How many random seeds? statistical power analysis in deep reinforcement learning experiments," *arXiv preprint arXiv:1806.08295*, 2018.

[90] R. Gorsane, O. Mahjoub, R. J. de Kock, R. Dubb, S. Singh, and A. Pretorius, "Towards a standardised performance evaluation protocol for cooperative marl," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5510–5521, 2022.

[91] D. Kalashnikov et al., "Mt-opt: Continuous multi-task robotic reinforcement learning at scale," *arXiv preprint arXiv:2104.08212*, 2021.

[92] J. Akhter, P. D. Fährmann, K. Sonntag, S. Peitz, and D. Schwietert, *Common pitfalls to avoid while using multiobjective optimization in machine learning*, 2025. arXiv: 2405.01480 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2405.01480.

[93] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, PMLR, 2015, pp. 2256–2265.

[94] L. Yang et al., *Diffusion models: A comprehensive survey of methods and applications*, 2024. arXiv: 2209.00796 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2209.00796.

[95] Z. Zhu et al., *Diffusion models for reinforcement learning: A survey*, 2024. arXiv: 2311.01223 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2311.01223.

[96] H. Cao et al., "A survey on generative diffusion models," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[97] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[98] T. Yu et al., *Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning*, 2021. arXiv: 1910.10897 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1910.10897.

[99] P. Florence et al., "Implicit behavioral cloning," in *Conference on Robot Learning*, PMLR, 2022, pp. 158–168.

[100] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, *Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning*, 2019. arXiv: 1910.11956 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1910.11956.

[101] A. Mandlekar et al., "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[102] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*, PMLR, 2021, pp. 8162–8171.

[103] D. Kingma, T. Salimans, B. Poole, and J. Ho, "Variational diffusion models," *Advances in neural information processing systems*, vol. 34, pp. 21 696–21 707, 2021.

[104] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," *arXiv preprint arXiv:2202.00512*, 2022.

[105] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," *arXiv preprint arXiv:2209.03003*, 2022.

[106] F.-Y. Wang, L. Yang, Z. Huang, M. Wang, and H. Li, "Rectified diffusion: Straightness is not your need in rectified flow," *arXiv preprint arXiv:2410.07303*, 2024.

[107] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[108] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[109] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[110] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded diffusion models for high fidelity image generation," *Journal of Machine Learning Research*, vol. 23, no. 47, pp. 1–33, 2022.

[111] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4195–4205.

[112] Y. Fan and K. Lee, "Optimizing ddpm sampling with shortcut fine-tuning," *arXiv preprint arXiv:2301.13362*, 2023.

[113] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine, "Training diffusion models with reinforcement learning," *arXiv preprint arXiv:2305.13301*, 2023.

[114] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[115] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, *Flow matching for generative modeling*, 2023. arXiv: `2210.02747 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2210.02747`.

[116] D. M. Ziegler et al., "Fine-tuning language models from human preferences," *arXiv preprint arXiv:1909.08593*, 2019.

[117] Y. Bai et al., "Training a helpful and harmless assistant with reinforcement learning from human feedback," *arXiv preprint arXiv:2204.05862*, 2022.

[118] L. Ouyang et al., "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[119] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[120] B. Wallace et al., "Diffusion model alignment using direct preference optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 8228–8238.

[121] W. Xiao et al., *A comprehensive survey of direct preference optimization: Datasets, theories, variants, and applications*, 2024. arXiv: `2410.15595 [cs.AI]`. [Online]. Available: `https://arxiv.org/abs/2410.15595`.

[122] M. Prabhudesai, A. Goyal, D. Pathak, and K. Fragkiadaki, "Aligning text-to-image diffusion models with reward backpropagation," *arXiv preprint arXiv:2310.03739*, 2023.

[123] K. Clark, P. Vicol, K. Swersky, and D. J. Fleet, "Directly fine-tuning diffusion models on differentiable rewards," *arXiv preprint arXiv:2309.17400*, 2023.

[124] A. Z. Ren et al., "Diffusion policy policy optimization," *arXiv preprint arXiv:2409.00588*, 2024.

[125] M. Rigter, J. Yamada, and I. Posner, "World models via policy-guided trajectory diffusion," *arXiv preprint arXiv:2312.08533*, 2023.

[126] Z. Wang, J. J. Hunt, and M. Zhou, "Diffusion policies as an expressive policy class for offline reinforcement learning," *arXiv preprint arXiv:2208.06193*, 2022.

[127] H. Chen, C. Lu, C. Ying, H. Su, and J. Zhu, "Offline reinforcement learning via high-fidelity generative behavior modeling," *arXiv preprint arXiv:2209.14548*, 2022.

[128] C. Chi et al., "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02 783 649 241 273 668, 2023.

[129] C. Lu, P. Ball, Y. W. Teh, and J. Parker-Holder, "Synthetic experience replay," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[130] A. Ajay, Y. Du, A. Gupta, J. B. Tenenbaum, T. S. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision making?" In *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: `https://openreview.net/forum?id=sP1fo2K9DFG`.

[131] Z. Dong et al., "Diffuserlite: Towards real-time diffusion planning," *arXiv preprint arXiv:2401.15443*, 2024.

[132] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, Pmlr, 2014, pp. 387–395.

[133] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[134] Z. Zhang, Y. Sun, J. Ye, T.-S. Liu, J. Zhang, and Y. Yu, "Flow to better: Offline preference-based reinforcement learning via preferred trajectory generation," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: `https://openreview.net/forum?id=EG68RSznLT`.

[135] H. Dong et al., *Raft: Reward ranked finetuning for generative foundation model alignment*, 2023. arXiv: `2304.06767 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2304.06767`.

[136] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool, "Repaint: Inpainting using denoising diffusion probabilistic models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 461–11 471.

[137] E. J. Hu et al., "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[138] Z. Dong et al., "Cleandiffuser: An easy-to-use modularized library for diffusion models in decision making," *arXiv preprint arXiv:2406.09509*, 2024. [Online]. Available: https://arxiv.org/abs/2406.09509.

[139] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine, "Idql: Implicit q-learning as an actor-critic method with diffusion policies," *arXiv preprint arXiv:2304.10573*, 2023.

[140] B. Kitts et al., "Ad serving with multiple kpis," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1853–1861.

[141] J. Guo et al., "Generative auto-bidding via conditional diffusion modeling," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 5038–5049.

[142] Y. Dong, N. V. Chawla, and A. Swami, "Metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 135–144.

[143] Y. He et al., "A unified solution to constrained bidding in online display advertising," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining*, 2021, pp. 2993–3001.

[144] K. Su et al., "Auctionnet: A novel benchmark for decision-making in large-scale games," *Advances in Neural Information Processing Systems*, vol. 37, pp. 94 428–94 452, 2025.

[145] S. Brody, U. Alon, and E. Yahav, *How attentive are graph attention networks?* 2022. arXiv: 2105.14491 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2105.14491.

[146] D. Huh, *Hierarchical bigraph neural network as recommendation systems*, 2020. arXiv: 2007.16000 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2007.16000.

[147] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, Mar. 2021, ISSN: 1573-1405. DOI: 10.1007/s11263-021-01453-z. [Online]. Available: http://dx.doi.org/10.1007/s11263-021-01453-z.

[148] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[149] A. Chiappa, B. Gangopadhyay, Z. Wang, and S. Takamatsu, "Auto-bidding in real-time auctions via oracle imitation learning," *arXiv preprint arXiv:2412.11434*, 2024.

[150] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *International conference on machine learning*, PMLR, 2018, pp. 5571–5580.

[151] G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "Camel: Communicative agents for" mind" exploration of large language model society," *Advances in Neural Information Processing Systems*, vol. 36, pp. 51 991–52 008, 2023.

[152] T. Guo et al., "Large language model based multi-agents: A survey of progress and challenges," *arXiv preprint arXiv:2402.01680*, 2024.

[153] K.-T. Tran, D. Dao, M.-D. Nguyen, Q.-V. Pham, B. O'Sullivan, and H. D. Nguyen, "Multi-agent collaboration mechanisms: A survey of llms, 2025," *URL https://arxiv.org/abs/2501.06322*,

[154] P. Shojaee*†, I. Mirzadeh*, K. Alizadeh, M. Horton, S. Bengio, and M. Farajtabar, *The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity*, 2025. [Online]. Available: https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf.

[155] A. D'Amour et al., "Underspecification presents challenges for credibility in modern machine learning," *Journal of Machine Learning Research*, vol. 23, no. 226, pp. 1–61, 2022.

[156] H. Zhang, L. H. Li, T. Meng, K.-W. Chang, and G. V. d. Broeck, "On the paradox of learning to reason from data," *arXiv preprint arXiv:2205.11502*, 2022.

[157] Z. Wu et al., "Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks," Association for Computational Linguistics, 2024.

[158] E. Fedorenko, S. T. Piantadosi, and E. A. Gibson, "Language is primarily a tool for communication rather than thought," *Nature*, vol. 630, no. 8017, pp. 575–586, 2024.

[159] J. Wei et al., *Chain-of-thought prompting elicits reasoning in large language models*, 2023. arXiv: 2201.11903 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2201.11903.

[160] S. Yao et al., *React: Synergizing reasoning and acting in language models*, 2023. arXiv: 2210.03629 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2210.03629.

[161] T. Schick et al., *Toolformer: Language models can teach themselves to use tools*, 2023. arXiv: 2302.04761 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2302.04761.

[162] A. Lazaridou, K. M. Hermann, K. Tuyls, and S. Clark, "Emergence of linguistic communication from referential games with symbolic and pixel input," *arXiv preprint arXiv:1804.03984*, 2018.

[163] J. Luketina et al., *A survey of reinforcement learning informed by natural language*, 2019. arXiv: 1906.03926 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1906.03926.

[164] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, *Generative agents: Interactive simulacra of human behavior*, 2023. arXiv: 2304.03442 [cs.HC]. [Online]. Available: https://arxiv.org/abs/2304.03442.

[165] T. Webb, K. J. Holyoak, and H. Lu, "Emergent analogical reasoning in large language models," *Nature Human Behaviour*, vol. 7, no. 9, pp. 1526–1541, 2023.

[166] K. Hong, A. Troynikov, and J. Huber, *Context rot: How increasing input tokens impacts llm performance*, 2025.

[167] M. Yasunaga et al., "Large language models as analogical reasoners," *arXiv preprint arXiv:2310.01714*, 2023.

[168] S. Kadavath et al., *Language models (mostly) know what they know*, 2022. arXiv: 2207.05221 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2207.05221.

[169] X. Chen et al., *Universal self-consistency for large language model generation*, 2023. arXiv: 2311.17311 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2311.17311.

[170] G. Team et al., *Gemma 3 technical report*, 2025. arXiv: 2503.19786 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2503.19786.

[171] H. Liu, C. Li, Q. Wu, and Y. J. Lee, *Visual instruction tuning*, 2023. arXiv: 2304.08485 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2304.08485.

[172] V. Team et al., *Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning*, 2025. arXiv: 2507.01006 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2507.01006.

[173] V. Subramaniam, Y. Du, J. B. Tenenbaum, A. Torralba, S. Li, and I. Mordatch, *Multiagent finetuning: Self improvement with diverse reasoning chains*, 2025. arXiv: 2501.05707 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2501.05707.

[174] Y. Li, S. Han, and S. Ji, *Vb-lora: Extreme parameter efficient fine-tuning with vector banks*, 2024. arXiv: 2405.15179 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2405.15179.

[175] Z. Shao et al., *Deepseekmath: Pushing the limits of mathematical reasoning in open language models*, 2024. arXiv: 2402.03300 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2402.03300.

[176] Z. Xi et al., "The rise and potential of large language model based agents: A survey," *Science China Information Sciences*, vol. 68, no. 2, p. 121 101, 2025.

[177] J.-t. Huang et al., "How far are we on the decision-making of llms? evaluating llms' gaming ability in multi-agent environments," *arXiv preprint arXiv:2403.11807*, 2024.

[178] R. Hendel, M. Geva, and A. Globerson, *In-context learning creates task vectors*, 2023. arXiv: 2310.15916 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2310.15916.

[179] S. Liu, H. Ye, L. Xing, and J. Zou, *In-context vectors: Making in context learning more effective and controllable through latent space steering*, 2024. arXiv: 2311.06668 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2311.06668.

[180] N. Panickssery, N. Gabrieli, J. Schulz, M. Tong, E. Hubinger, and A. M. Turner, *Steering llama 2 via contrastive activation addition*, 2024. arXiv: 2312.06681 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2312.06681.

[181] A. Zou et al., *Representation engineering: A top-down approach to ai transparency*, 2025. arXiv: 2310.01405 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2310.01405.

[182] B. W. Lee et al., "Programming refusal with conditional activation steering," *arXiv preprint arXiv:2409.05907*, 2024.

[183] T. Liu et al., "Statistical rejection sampling improves preference optimization," *arXiv preprint arXiv:2309.06657*, 2023.

[184] M. AI, *Llama-4: Multimodal intelligence*, 2025. [Online]. Available: https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

[185] Q. Team, *Qwen3 technical report*, 2025. arXiv: 2505.09388 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2505.09388.

[186] A. Fourney et al., *Magentic-one: A generalist multi-agent system for solving complex tasks*, 2024. arXiv: 2411.04468 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2411.04468.

[187] Z. Ke et al., *Mas-orchestra: Understanding and improving multi-agent reasoning through holistic orchestration and controlled benchmarks*, 2026. arXiv: 2601.14652 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2601.14652.

[188] P. Xia et al., *Skillrl: Evolving agents via recursive skill-augmented reinforcement learning*, 2026. arXiv: 2602.08234 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2602.08234.

[189] Y. Wang et al., "Reinforcement learning for reasoning in large language models with one training example," *arXiv preprint arXiv:2504.20571*, 2025.

[190] A. Taubenfeld et al., *Evaluating alignment of behavioral dispositions in llms*, 2026. arXiv: 2602.11328 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2602.11328.