

Generative Models for Planning and Decision-Making

by

Akash Karthikeyan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2025

© Akash Karthikeyan 2025

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Akash Karthikeyan is the sole author of this thesis, which was written under the supervision of Professor Yash Vardhan Pant. The core chapters of this thesis draw upon papers that are either published or currently under review.

1. Chapter 3 originates from work presented at the *AAAI Conference on Artificial Intelligence* and the *Conference on Robots and Vision 2025*, respectively:
 - Akash Karthikeyan and Yash Vardhan Pant. GenPlan: Generative Sequence Models as Adaptive Planners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(17):17797–17804, Apr. 2025
 - Akash Karthikeyan and Yash Vardhan Pant. Adaptformer: Sequence Models as Adaptive Iterative Planners. *Proceedings of the Conference on Robots and Vision*, May 2025
2. Chapter 4 elaborates on research detailed in a paper currently under peer review:
 - Akash Karthikeyan and Yash Vardhan Pant. DiffFP: Learning Behaviors from Scratch via Diffusion-based Fictitious Play. *Initial results presented at the IJ-CAI 2025 Workshop on User-Aligned Assessment of Adaptive AI Systems. Full version under review.*

Abstract

Generative models have achieved remarkable progress across domains such as vision and language. However, their application to sequential decision-making and planning remains challenging. In reinforcement learning and robotics, agents must handle task hierarchies, long-horizon dependencies, adapt to harder unseen tasks and environments, and, especially in multi-agent settings, respond to adversarial or evolving opponents. Despite progress in behavioral cloning and offline policy learning, existing approaches often struggle to generalize beyond the train distribution or to learn robust, interactive behaviors in competitive games. These limitations restrict current systems to narrow tasks with short temporal horizons, or deterministic settings. For instance, behavioral planners trained on single-goal environments struggle scaling to multi-task missions requiring subgoal discovery and adaptive reasoning, as there is no straightforward mechanism for iterative test-time adaptation to these unseen tasks. Similarly, in multi-agent reinforcement learning, standard policy optimization often yields unimodal, brittle strategies that overfit to specific opponents and fail to converge to a Nash equilibrium in continuous state-action games. This thesis explores challenges and opportunities in using generative models for planning and decision-making tasks, specifically energy-based and diffusion-based models which serve as both representations and solvers for planning and policy learning. In the single-agent setting, we introduce GenPlan, a discrete-flow planner that reframes planning as iterative denoising over trajectories using an energy-guided diffusion process. This formulation enables task and goal discovery, and adaptation to unseen environments. In the multi-agent setting, we propose DiffFP, a diffusion policy gradient method within the fictitious play framework. By approximating best responses through diffusion models, DiffFP captures multimodal strategies, improves sample efficiency, and remains robust to evolving opponents in dynamic, continuous state-action games. Our empirical studies show that GenPlan outperforms baselines by over 10% on adaptive planning tasks, generalizing from single-task demonstrations to complex, compositional multi-task missions. Likewise, DiffFP achieves up to $3\times$ *faster convergence* and $30\times$ *higher success rates* compared to other baseline reinforcement learning algorithms in multi-agent benchmarks. These results demonstrate the potential of generative modeling not only for representation learning, but as a unified substrate for planning, learning, and decision-making across settings.

Acknowledgements

I am luckier than I deserve.

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Yash Vardhan Pant, for his constant support, insightful feedback, and for giving me the opportunity to explore research in a direction that I have grown to deeply care about. His mentorship has played a pivotal role in shaping my research thinking and this thesis.

I chose to pursue a graduate degree largely due to the overwhelmingly positive research experiences I had as an undergraduate at TCE and as a visiting researcher at the University of Toronto. I am grateful to Professor Igor Gilitschenski and Professor Saravana Perumaal Subramanian for those early opportunities and for introducing me to the joy of research, and the value of persistence.

I am especially thankful to the members of our research lab for their support, and the many thoughtful discussions (academic and otherwise) that helped keep me grounded and motivated throughout this journey.

A heartfelt thanks to my friends for the memories, the laughter, and most of all, for making any place feel at home.

I am grateful to the Vector Institute for the Vector Scholarship in Artificial Intelligence, and to Mitacs for their financial support through the Mitacs Globalink Graduate Fellowship.

Dedication

To my parents, for inspiring me and for their unwavering support.

Table of Contents

Author’s Declaration	ii
Statement of Contributions	iii
Abstract	iv
Acknowledgements	v
Dedication	vi
List of Figures	x
List of Tables	xiii
List of Abbreviations	xvi
1 Introduction	1
1.1 Learning for Planning	3
1.2 Learning for Multi-Agent Systems	4
1.3 Thesis Contributions	5
1.4 Thesis Outline	6

2	Literature Review	7
2.1	Learning for Planning	7
2.2	Learning in Multi-Agent Games	9
3	Planning with Discrete Diffusion	11
3.1	Overview	11
3.2	Preliminaries and Problem Statement	13
3.3	GenPlan: Method	16
3.3.1	Energy-Guided Denoising Model	16
3.3.2	Joint Denoising Model	16
3.3.3	Training Objective	17
3.3.4	Planning	18
3.4	Simulation Studies	19
3.4.1	BabyAI	19
3.4.2	Manipulation Tasks	26
4	Learning for Multi-Agent Games	30
4.1	Overview	30
4.2	Preliminaries and Problem Statement	31
4.3	DiffFP: Method	33
4.3.1	Fictitious Play	34
4.3.2	Approximating Best Response via Reinforcement Learning	34
4.4	Simulation Studies	38
4.4.1	Baselines	38
4.4.2	Racing Environment	39
4.4.3	Multi-Agent Particle Environment	41
4.4.4	Discussion - Why Diffusion Policies?	44
5	Conclusion	46

Open Source	49
References	50
APPENDICES	59
A Planning with Discrete Diffusion - Details	60
A.1 BabyAI Implementation Details	60
A.1.1 Input and Networks	60
A.1.2 Environments	61
A.1.3 Baseline Models	63
A.2 Additional Results and Discussion	64
A.3 Continuous Tasks Implementation Details	68
A.4 MetaWorld Implementation Details.	68
A.5 Rate Matrices and Sampling	70

List of Figures

1.1	Multi-Task Mission Adaptation. Adaptive planners should be capable of leveraging simple task demonstrations and scaling to: 1) recognizing and executing implicit subtasks ($1 \rightarrow 4$) in long-horizon missions, 2) generalizing to tasks involving multiple goals, and 3) adaptively learning new skills (e.g., unblocking pathways). Goals are highlighted in yellow, while distractors are marked in red.	2
3.1	Overview. GenPlan is a generative, multi-step planner that leverages simple demonstrations to adapt to complex tasks by learning an energy landscape and iteratively refining trajectories with respect to this energy. Goals are highlighted in yellow, and distractors are marked in red.	12
3.2	Method Overview. GenPlan, trained on offline data (A), learns to jointly model action, goal, and state distributions. In (B), the joint denoising model (see Section 3.3.2) takes in a corrupted trajectory τ^t and predicts the clean trajectory τ^1 . (C) Demonstrates the joint inference of goals and actions by simulating the reverse Continuous-time Markov chain (CTMC), as detailed in Algorithm 2.	15
3.3	Energy Landscape. GenPlan, when conditioned on sub-goals, implicitly assigns minimal energy to necessary sub-goals (e.g., picking up keys, opening doors) for task completion. States closer to the white region are more likely to be transitioned into. LEAP, in contrast, does not prioritize these sub-tasks.	25

3.4	GenPlan Analysis. (A) Success rate as a function of the number of denoising iterations, reported using 3 different training seeds evaluated on 250 environments, specifically for <i>GoToObjMazeS4N3G1 (Adaptive Planning (AP))</i> and <i>GoToObjMazeS4G1 (Trajectory Planning (TP))</i> . (B) Energy landscape learned by GenPlan during training, illustrating that GenPlan effectively captures the true action distribution by assigning low energy to noise-free sequences.	27
3.5	State Coverage (AP). State visit frequency is evaluated across 10 unseen maze layouts with varying goal positions (Rooms 1-9), starting from a fixed agent position (Room 1).	27
3.6	State Coverage (TP). Coverage is evaluated in larger environments. Results are reported across 50 map variations from a fixed start position (Room 1).	27
3.7	Policy Deployment. We deploy the policy on an AGILEX LimoBot. See the project webpage for more details.	28
4.1	(A–B) Failure Modes of Baselines. Baseline agents exhibit suboptimal behaviors such as stalling or inefficient path planning. (C–E) Training Progression of DiffFP. Agent trajectories sampled from successive FP iterations illustrate the emergence of strategic behaviors. Over time, agents learn to navigate more efficiently, reducing their time-to-goal from 80 to 45 steps under identical initial conditions.	39
4.2	Exploitability. We report exploitability (see Def. 3) computed over 100 evaluation runs. Lower values indicate better robustness, with the proposed DiffFP achieving the lowest exploitability.	42
4.3	A. Exploitability on the Racing Task. Mean and standard deviation of exploitability over 10 episodes per Fictitious Play (FP) iteration. B. Normalized Episodic Rewards. This metric reflects training performance and stability.	42
4.4	A. MPE-Tag Env., B. MPE-Adversary Env. Dashed lines indicate possible agent trajectories. C. Q-value Map. At convergence to a near Nash equilibrium, ego agents exhibit coordinated behaviors with diverse roles e.g., one agent advances toward the goal while the other acts as a decoy to distract or mislead the adversaries.	43

A.1	Trajectory Planning (TP). The agent is randomly initialized and must navigate through a maze-like environment to reach the goal(s). In each evaluation, the map layout, the agent’s initial position, and goal positions are varied. Note that in the case of GenPlan, the agent does not have access to the ground truth goal positions, whereas other baselines do.	62
A.2	Instruction Completion (IC). These tasks are more complex than TP, requiring the agent to perform additional actions such as <i>pickup</i> , <i>drop</i> , and <i>open</i> in addition to navigation. For example, in <i>DoorsOrder</i> , the agent must open the specified doors in the correct order without accidentally entering other rooms.	62
A.3	Adaptive Planning (AP). The model is trained on simpler tasks and tested for adaptability by progressively increasing task difficulty. In <i>GotoObjMazeClose</i> and <i>GoToSeqUn</i> , we evaluate whether the model can demonstrate novel skills, such as unblocking obstacles and opening doors.	62
A.4	GenPlan I/O. During planning, the model takes in a corrupted trajectory along with the current state and past observations (if available), and iteratively recovers the clean trajectory.	65
A.5	GenPlan Configurations. (A) Training progression of GenPlan, showing that GenPlan-M converges faster than GenPlan-U. (B) Success rate as a function of entropy.	66

List of Tables

3.1	BabyAI quantitative performance. Success rates of the models across different environments are presented. The abbreviations SW , NX , RY , and GZ in the environment names represent the size (W) of a room in the map, the number of obstacles (X), the number of rows (Y), and the number of goals (Z) during testing, respectively. The term “ <i>Close</i> ” indicates that the agent requires door-opening actions. See Appendix A.1.2 for more details. .	21
3.2	BabyAI quantitative performance. Success rates of the models across different environments are presented. The abbreviations SW , NX , RY , and GZ in the environment names represent the size (W) of a room in the map, the number of obstacles (X), the number of rows (Y), and the number of goals (Z) during testing, respectively.	22
3.3	Ablation. The values represent the mean success rates as detailed in Section 3.4.1, with reductions shown relative to the results in Tables 3.1 and 3.2.	24
3.4	Comparison of Energy-Based and DDPM Baselines on MiniGrid Tasks. We compare various sampling techniques and generative objectives on discrete planning tasks. GenPlan demonstrates significant improvements in success rates, particularly on more complex tasks, by combining energy objectives with DFM sampling.	26
3.5	Quantitative results. Comparison of baselines on unconditional continuous-space tasks.	29
4.1	Head-to-Head Trials. Gap denotes the mean relative distance gained by the attacker. Successes refer to the number of episodes where the attacker successfully closed the initial gap or overtook the target. Attacking agent is initialized behind the race leader.	41

4.2	Robustness to Unseen Opponents. Evaluation of crash rates (lower is better) and mean rewards (higher is better) when the agent is tested against up to five previously unseen adversaries.	41
4.3	Robustness Against Unseen Opponents. Head-to-head evaluation in the Multi-Agent Particle Environment (MPE)-Adversary environment. Colored numbers indicate relative gains (↑) or losses (↓) when ego and adversary roles are swapped.	44
4.4	Robustness Against Unseen Opponents. Head-to-head evaluation in the MPE-Tag environment. Colored numbers indicate relative gains (↑) or losses (↓) when ego and adversary roles are swapped.	45
A.1	Offline data. All of these components are mapped to a latent embedding, as shown in Figure 3.2B.	61
A.2	Hyperparameters of GenPlan for BabyAI experiments.	63
A.3	Training Configurations. For most tasks, planning is based solely on the current state observation and aims to plan for H steps. However, for <i>KeyCorridor</i> and <i>BlockUnlock</i> , which involve complex sub-tasks and require memory to track completed tasks, context is used.	64
A.4	BabyAI environment setting details.	65
A.5	Stochastic Environments. Success rates of models in task planning and adaptive environments using single-step rollouts to account for stochasticity.	66
A.6	Time Considerations. Comparison of training and evaluation times across different environments and methods. Training is performed with a batch size of 64 over 500 epochs, while evaluation is conducted across 250 environments.	67
A.7	Instruction Prompt Ablation. We ablate the instruction embeddings from the input to assess their significance, as evidenced by the reduction in success rates across environments.	67
A.8	Performance with Suboptimal Trajectories. We introduce 25% random actions into the training data to simulate suboptimal trajectories. Despite this, GenPlan-M demonstrates robust generalization, outperforming DT across all environments.	68
A.9	Quantitative Performance. Comparison between MDDT and GenPlan. Success rates (SR) and mean rewards (MR) are reported across three unique seeds.	69

A.10 Dataset and Performance Scores.	The agent is trained only on datasets from the 5 in-distribution environments and evaluated on all nine. Datasets from adaptive environments are not used.	70
A.11 Masking Interpolant.	The sample is initialized with the mask state (represented by 6) and iteratively denoised to obtain the clean sequence (in green).	71
A.12 Uniform Interpolant.	The sample is initialized with uniform noise (in red) and iteratively denoised to obtain the clean trajectory (in green). . . .	71

List of Abbreviations

AP Adaptive Planning [xi](#), [xii](#), [20](#), [23–25](#), [27](#), [61](#), [62](#), [64–66](#)

BC Behavior Cloning [1](#), [3](#), [11](#), [13](#), [15](#), [23](#), [24](#)

CEM Cross-Entropy Method [23](#), [24](#)

CTMC Continuous-time Markov chain [x](#), [12–15](#), [17](#), [18](#), [60](#), [61](#), [66](#), [70](#)

DDPM Denoising Diffusion Probabilistic Models [35](#)

DFM Discrete Flow Model [12](#), [16](#), [17](#), [67](#)

DP Diffusion Policy [37](#), [38](#)

DT Decision Transformer [13](#), [22](#), [26](#), [63](#), [65–67](#)

FP Fictitious Play [xi](#), [4](#), [9](#), [10](#), [30](#), [31](#), [33–38](#), [40](#), [42](#), [46](#)

FSP Fictitious Self-Play [4](#), [10](#)

IC Instruction Completion [20](#), [23](#), [24](#), [61](#), [64](#), [65](#)

MARL Multi-Agent Reinforcement Learning [4](#), [9](#)

MCMC Markov chain Monte Carlo [8](#), [23](#), [24](#)

MDP Markov Decision Processes [13](#)

MLM Masked Language Model [8](#), [20](#), [66](#)

MPE Multi-Agent Particle Environment [xiv](#), [41](#), [43–45](#)

POMG Partially Observable Markov Game [31](#), [33](#)

RL Reinforcement Learning [1–4](#), [8](#), [9](#), [13](#), [16](#), [21](#), [31](#), [34](#), [38](#)

RvS Reinforcement learning via Supervised learning [3](#)

SAC Soft Actor Critic [69](#)

TP Trajectory Planning [xi](#), [xii](#), [19](#), [23–27](#), [62](#), [64–66](#)

Chapter 1

Introduction

Generative models have emerged as powerful tools across vision and language domains, achieving state-of-the-art performance in several tasks. *Image and video generation* has seen significant advances through diffusion and transformer-based models that can synthesize realistic scenes and coherent dynamics from text or visual prompts [38, 16, 9]. In *world modeling*, generative models enable the simulation and control of comprehensive 3D environments with dynamics and interaction, supporting tasks such as future prediction, scene completion, and agent-based simulation [21, 9]. For *text generation and composition*, large language models demonstrate strong zero-shot and in-context learning capabilities across a range of natural language tasks [27]. With sufficient data and modeling capacity, these generative systems produce coherent, high-fidelity outputs that rival human-level performance on many benchmarks. However, achieving similar success in robotics and sequential decision-making is considerably more challenging.

- **Task Specification:** Unlike language or vision, task goals in [Reinforcement Learning \(RL\)](#) and robotics are often implicit, multimodal, or must be inferred from sparse, delayed feedback, making direct conditioning difficult.
- **Long-Horizon Consistency:** Planning over extended time horizons requires coherence across multi-step sequences and the ability to reason and re-plan in response to harder tasks and failures. Existing methods typically rely on an oracle to provide subgoal sequences, task-hierarchy decompositions, and demonstration data.
- **Generalization:** A fundamental challenge in [Behavior Cloning \(BC\)](#) [66, 41] is the covariate shift between training and deployment. Since the model learns to mimic

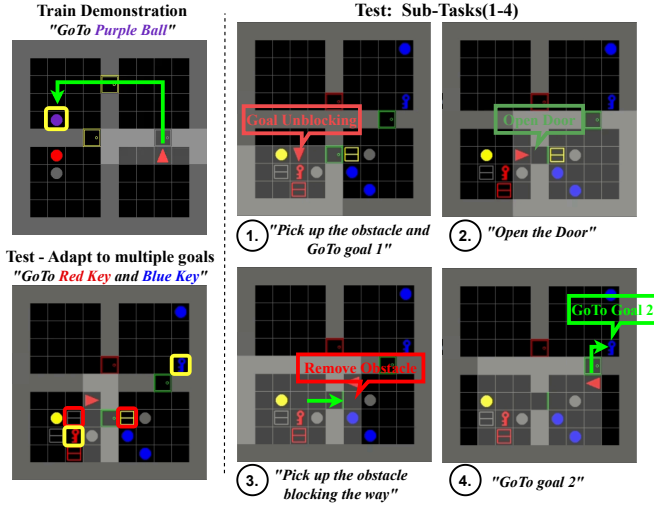


Figure 1.1: **Multi-Task Mission Adaptation.** Adaptive planners should be capable of leveraging simple task demonstrations and scaling to: 1) recognizing and executing implicit subtasks (1 $\rightarrow$ 4) in long-horizon missions, 2) generalizing to tasks involving multiple goals, and 3) adaptively learning new skills (e.g., unblocking pathways). Goals are highlighted in yellow, while distractors are marked in red.

expert actions only at states seen in the demonstrations, it struggles to recover from off-distribution errors, often compounding small mistakes into failure.

- **Procedural Reasoning:** In multi-agent or competitive settings, agents must reason about and respond to diverse, possibly adversarial opponents. Success requires modeling not just task completion, but interactive and adaptive behavior.

Despite these challenges, generative modeling provides a compelling foundation for addressing them. As flexible parametric models, generative models are well-suited to capture complex, multimodal distributions over trajectories, goals, or strategies. When combined with RL, sampling, and optimization-based inference, they enable scalable decision-making under uncertainty. Moreover, in self-play settings, generative models can learn adaptive behaviors through interaction, offering a recipe for robustness and generalization in both single-agent and multi-agent environments. This thesis explores the role of generative modeling in decision-making, presenting two distinct contributions unified under the theme of planning and decision-making through generative models. These contributions are formalized as two core problems: planning via iterative denoising (problem 1) and multi-agent decision-making via minmax objective in dynamic games (problem 2). We build upon these problem formulations in Chapters 3 and 4, respectively.

1.1 Learning for Planning

To enable generalizable planning, we require agents that can adapt to novel tasks and environments beyond those encountered during training, as seen in Figure 1.1. This is crucial for operating in complex environments that may introduce distractors (i.e., objects the agent has not seen before) and have multiple novel goals.

Example 1 *Consider a task where an agent is initially trained on a simple single-goal-reaching task in a planar environment. During evaluation, the agent must adapt to complex environments with walls, locked doors, and multiple goals, as illustrated in Figure 1.1. The planner must generate multi-step trajectories, exploring to locate goals while avoiding getting stuck, unblocking paths, and navigating around walls to complete the mission.*

Learning adaptive behaviors from demonstrations is challenging and often requires a well-labelled and diverse dataset that accurately represents the necessary tasks. This is much harder for long-horizon tasks, where the correct order of subtasks is critical (e.g., collecting a key before unlocking a door).

Reinforcement learning via Supervised learning (RvS) [19] simplifies traditional RL by using a BC objective and framing RL as sequence modeling, where actions are sequentially predicted based on next best action. However, this approach often struggles in order-critical multi-task missions, as generated plans can suffer from locally optimal decisions, resulting in global dead-ends, or accumulate errors without accounting for mistakes in previous actions [51].

Thus, we explore the use of abstracted sequence planners [53], which model trajectories as categorical distributions. This formulation is better suited for capturing multimodal task structures, as it enables clustering of distinct behavioral modes and yields interpretable predictions over subtask orderings. More recently, generative modeling-based works allow conditional sequence-level predictions [41, 13]. However, based on our results evaluating energy models and diffusion methods with various objectives (e.g., BC) and sampling techniques, such as the cross-entropy method, Gibbs sampling, and energy gradient-based guidance, these approaches fail on multi-step planning (see Section 3.4.1). They often get stuck in local optima when handling unseen cases, as they lack distributional fidelity and struggle to adapt to new tasks. To address these challenges and to develop a framework for procedural reasoning, we reframe this setting as an iterative denoising problem, which serves as the basis for our planner GenPlan (see problem 1).

1.2 Learning for Multi-Agent Systems

Within game-theoretic frameworks, [Multi-Agent Reinforcement Learning \(MARL\)](#) has achieved remarkable success in domains such as Go, Atari, Chess, Shogi, and StarCraft II [73, 72, 59, 80], demonstrating the transformative impact of [MARL](#) in complex and dynamic environments. In these systems, agents learn strategies through policy optimization techniques, parameterized with deep neural networks.

A common approach to learning robust policies in interactive competitive settings is self-play [68], where agents iteratively improve by playing against copies or past versions of themselves. Similarly, [Fictitious Play \(FP\)](#) [5] is a game-theoretic algorithm for approximating Nash equilibria in imperfect-information games: at each iteration, every agent computes its best response to the empirical average of its opponents’ past actions. In two-player zero-sum games, the resulting sequence of joint strategies converges to a Nash equilibrium. Recent advances have extended [Fictitious Self-Play \(FSP\)](#) to more complex settings, including extensive-form and Markov games [36, 26]. At each iteration, every agent computes a best response to the empirical distribution over past strategies, and then updates its strategy accordingly. Under certain conditions, this procedure converges to a Nash equilibrium.

In practice, computing an exact best response in continuous and high-dimensional environments is challenging [87]. [RL](#)-based methods are typically employed to approximate the best responses; however, standard [RL](#) policies tend to be unimodal [86]. This unimodality limits their ability to capture the inherent multimodality of complex tasks (e.g., left vs. right-side overtaking in racing), often resulting in policies that overfit to recent opponent behaviors while being easily exploitable by previously encountered opponents.

Recent work has demonstrated the power of diffusion models in capturing diverse data distributions, with notable success in behavioral cloning, imitation learning, and planning [38, 13, 41]. However, their application in online [RL](#) remains relatively under-explored. While some approaches leverage diffusion models in offline settings [84, 1, 43], integrating them into the online [RL](#) framework is challenging due to the dynamic nature of value estimation as policies evolve and the diffusion objective cannot directly capture these changes [86, 67, 17].

These challenges motivate the need for expressive, multimodal best-response policies that can adapt online to evolving opponents. In this thesis, we explore how diffusion-based generative models can serve as powerful policy classes for this purpose. To address these challenges we formalize multi-agent decision-making as a minmax optimization problem over joint continuous-space policies (see problem 2).

1.3 Thesis Contributions

To address the challenges outlined above, this thesis presents the following contributions unified under the theme of planning and decision-making through generative models.

- **From Prediction to Planning.** A central question in model-based RL is: what prediction problem should the generative model solve to enable effective planning? Standard approaches rely on short-horizon, step-wise prediction of transitions, but these often fail in long-horizon or sparse-reward settings where the correct behavior spans many steps and involves implicit subgoal structure. To address this, we adopt a sequence-level generative modeling perspective and reframe planning as an iterative energy minimization problem. This formulation enables procedural reasoning and facilitates task and goal discovery over extended horizons.
- **Generative Planning via Discrete Diffusion.** In the single-agent setting, we develop GenPlan, a discrete diffusion-based planning framework that treats planning as iterative denoising in trajectory space. Instead of embedding a generative model within a traditional planner, GenPlan incorporates the planning mechanism directly into the generative model. This approach allows both the representation and the planning process to improve with data. By learning an energy-guided denoising process over trajectories, GenPlan enables task and goal discovery, generalization to novel environments, and robust recovery from suboptimal or partial plans. These capabilities are particularly important in long-horizon, compositional tasks.
- **Diffusion as Best Response in Multi-Agent Games.** In competitive multi-agent settings, policy learning becomes a moving-target problem, where each agent must adapt to the evolving strategies of others. Classical self-play algorithms iteratively improve by computing best responses, but in practice these instantiations struggle in continuous state-action spaces due to the unimodality and poor sample efficiency of standard policy classes. We propose DiffFP, a diffusion policy gradient method that computes best responses via generative modeling. By training a diffusion model to represent the best response in continuous action space, and updating the opponent model online, DiffFP achieves fast convergence to approximate Nash equilibria, multimodal strategy discovery, and robustness to opponent shifts.

Outlook. Our results suggest that integrating planning directly into generative models, rather than using them merely as components can unlock improved adaptability, expressivity, and convergence in decision-making. This viewpoint opens several exciting directions:

planning in abstract state and time representations, combining learned dynamics with structured optimizers, amortized planners that allows test-time flexibility, and eventually, foundation models for control that generalize across tasks and domains.

1.4 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 reviews related literature on learning-based planning through sequence models and multi-agent frameworks designed to address decision-making in dynamic environments. Chapter 3 formulates the long-horizon planning problem as an iterative denoising objective and introduces GenPlan, a discrete generative planner. We present the modeling framework and corresponding simulation studies across various planning and manipulation tasks. Chapter 4 introduces the problem of multi-agent decision-making under competitive and semi-competitive (general-sum) settings. We present DiffFP, a generative, game-theoretic framework for continuous and dynamic games, along with empirical evaluations in multi-agent simulation environments. Finally, Chapter 5 summarizes the key insights from both lines of work and outlines promising directions for future research at the intersection of generative modeling and sequential decision-making.

Chapter 2

Literature Review

We review relevant prior work across two domains: (i) planning and reasoning with sequence models from demonstrations, and (ii) robust decision-making in multi-agent settings through fictitious play and generative modeling. While both domains aim to solve long-horizon, adaptive decision-making problems, they differ fundamentally in learning objectives. In the planning setting, the goal is to reason and generalize from demonstrations using offline data, often under distribution shift. In contrast, the multi-agent setup involves optimizing minmax formulation and learning from scratch without access to expert trajectories, requiring agents to discover robust, strategically sound behaviors through repeated fictitious play.

2.1 Learning for Planning

Offline Reinforcement Learning. Recent successes in deep sequence modeling have sparked interest in applying such models to decision-making from expert demonstrations. Offline reinforcement learning aims to learn policies solely from previously collected trajectories, without any additional environment interaction [55]. This setting is particularly appealing in scenarios where real-world data collection is costly or unsafe. A key challenge in this approach is the *distribution shift* between the training demonstrations and the distribution at inference. Several regularization strategies have been proposed to address this, such as reducing the discrepancy between the learned and behavioral policies [24, 49].

However, these approaches often fail to adapt to unseen tasks and are typically limited to single-step planning. We are interested in optimizing sequence-level plans.

More recent works [11, 42, 25] adopt an autoregressive modeling objective [29, 78], leveraging the self-attention mechanism of sequence models. By conditioning on desired returns or goal states, these methods guide the generation of future actions, provided such states are encountered during training. While effective for behavior cloning tasks, these methods fail to adapt to harder or unseen tasks, often resulting in deadlocks and compounding errors. Moreover, they struggle in unconditional rollouts, lacking both guidance and the ability to generalize to novel goals. In contrast, our approach optimizes sequence-level action plans that leverage simple training demonstrations (e.g., single-goal tasks) to scale to and reason about complex objectives (e.g., multi-task missions).

Planning with Sequence Models. Planning Transformer [76] introduces procedural planning with transformers by framing planning as a model-based Reinforcement Learning (RL) problem. It employs a joint action and state representation model to mitigate compounding errors inherent in transformer-based architectures. LEAP [10] addresses planning as an iterative energy minimization problem, utilizing a Masked Language Model (MLM) to learn trajectory-level energy functions. This approach employs Gibbs sampling to mitigate error accumulation and demonstrates improved generalization to novel test scenarios. However, LEAP’s reliance on an oracle for goal positions limits its effectiveness, particularly in larger mazes, as shown in our simulation studies (see Table 3.1). Without conditional guidance, performance declines significantly. This limitation can cause the agent to enter loops, leading to *stalling actions*, where it fails to make progress.

Generative Models in Planning. Deep generative models have recently demonstrated success in offline RL. Broadly, energy-based models [31, 20] and diffusion models [41, 13] have been applied to offline RL tasks. While diffusion-based models often require well-represented datasets, they fail to generalize to unseen harder tasks. In contrast, energy-based models can learn an energy landscape that helps generalize to unseen tasks; however, they are hard to sample from, requiring long cycles of cross-entropy or Markov chain Monte Carlo (MCMC) sampling [10], and are often unstable to train [13]. Recent work on Energy-Diffusion [18] attempts to unify both paradigms by guiding diffusion sampling with energy gradients. However, in planning tasks, we observed that the low-energy trajectories can be gamed by repetitive frequent actions from the demonstrations (e.g., forward actions), causing the model to get trapped in local minima.

To address these challenges, we propose an energy-diffusion framework (GenPlan) capable of iteratively refining plans for unseen environments by learning annealed energy landscapes and employing diffusion-based sampling to enable goal and task discovery.

2.2 Learning in Multi-Agent Games

In [Multi-Agent Reinforcement Learning \(MARL\)](#), we focus on zero-sum and general-sum settings, where agents are often grouped into ego and adversarial teams. These environments naturally lend themselves to fictitious play training regimes, which are commonly employed to discover robust and strategically sound policies through repeated interactions with evolving opponents.

Adversarial Robustness. Maximum Entropy [RL](#) [32] encourages robustness by maximizing expected reward while maintaining high entropy in the policy distribution. Though this has proven effective in standard continuous control tasks [77], it struggles in dynamic multi-agent settings where adversarial behaviors lead to non-stationary learning targets. To address these challenges, population-based training frameworks inspired by empirical game theory [85] have been proposed. These maintain a pool of evolving strategies and compute best responses, as in Policy-Space Response Oracles (PSRO) [52].

In continuous control tasks, [2] propose opponent sampling, where agents train against past versions of opponents using various sampling strategies similar to [Fictitious Play \(FP\)](#), but do not provide exploitability studies. Similarly, Adversarially Robust Control (ARC) [50] tackles robustness in semi-competitive autonomous driving settings by training agents against an ensemble of adversaries, while regularizing towards imitation-learned behaviors to prevent overfitting.

More recent approaches, such as population-based diversity shaping [14], explicitly target diversity and robustness in continuous control via population-based training. However, these methods often incur additional computational cost and remain sensitive to adversary tuning, with diminishing returns observed as population size increases [79]. In general, when computing best responses under a finite budget, approximate best-response operators must be truncated at each iteration, resulting in under-trained policies populating the population. Furthermore, relearning policies from scratch at each iteration becomes intractable and hinders scalability to larger action spaces [57]

A range of multi-agent RL approaches aim to produce policies that are less exploitable than those obtained via standard self-play. One prominent example is Counterfactual

Regret Minimization (CFR) [90], which has achieved superhuman performance in games like poker [6]. However, extending CFR to continuous-action games with high-dimensional state and action spaces remains an open challenge. In contrast, self-play continues to be a dominant paradigm due to its scalability and empirical success across a wide range of complex domains [74, 80].

Self-Play for Zero-Sum Games. Self-play can, in principle, converge to a Nash equilibrium if reinforcement learning produces a best response at each iteration [36]. In practice, convergence is often elusive due to challenges such as *non-transitivity*, as in rock-paper-scissors, where learning dynamics may enter persistent cycles [64]. Even in transitive games, progress can stall because of local minima or the limited expressivity of the policy architecture.

Neural Fictitious Self-Play [37] provides a general framework for multi-agent systems through end-to-end deep reinforcement learning and has recently been extended to continuous-action settings, where best responses are approximated using deep reinforcement learning [26]. Such extensions are particularly relevant in domains such as robotics and autonomous driving, which involve continuous action spaces, partial observability, and imperfect information.

Inspired by Fictitious Self-Play (FSP) and its neural parametrization, we extend FP to high-dimensional continuous-control tasks and learn robust best-response policies *from scratch* via diffusion-based policy representations. These representations are capable of capturing multimodal strategies, avoiding cyclic forgetting, and improving sample efficiency. Our formulation accommodates both symmetric and asymmetric games, enabling scalable learning with improved sample efficiency and generalization.

Chapter 3

Planning with Discrete Diffusion

3.1 Overview

Planning in complex, long-horizon environments demands more than just [Behavior Cloning \(BC\)](#), it requires procedural reasoning, goal discovery, and adaptability to new scenarios.

Contributions. In this work, we propose GenPlan [47], a generative planner that models planning as trajectory-level sequence generation, framed through the lens of energy minimization and discrete diffusion.

- GenPlan formulates planning as iterative denoising of action trajectories using a discrete diffusion model, enabling subgoal discovery and adaptation.
- We learn an energy function over trajectories and guide the denoising process by minimizing this energy, improving planning quality and robustness.
- To promote generalization, GenPlan incorporates an entropy-based lower bound over action distributions, which encourages diversity and adaptability in novel tasks.

Unlike traditional approaches that greedily select the next action, GenPlan jointly models the distribution over states and actions with bidirectional context. This joint modeling helps the agent avoid error accumulation, prevent local deadlocks, and generate globally consistent plans. Furthermore, by embedding the planning process into the generative model itself, GenPlan ensures that both representation and decision-making quality improve with more data and training.

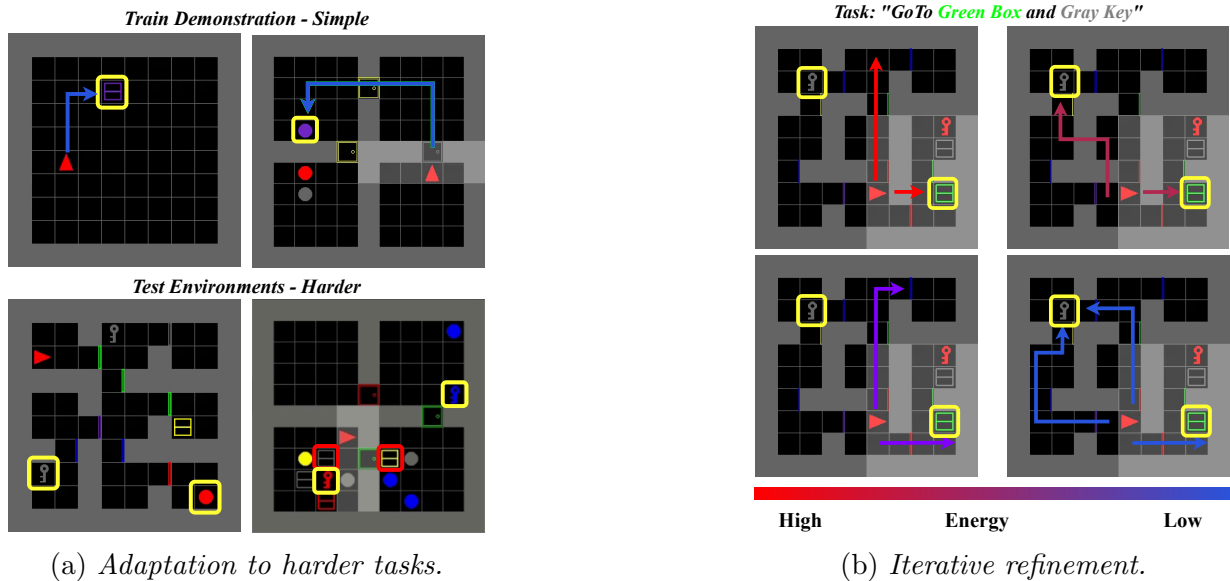


Figure 3.1: **Overview.** GenPlan is a generative, multi-step planner that leverages simple demonstrations to adapt to complex tasks by learning an energy landscape and iteratively refining trajectories with respect to this energy. Goals are highlighted in yellow, and distractors are marked in red.

System Components. The denoising planner integrates [Discrete Flow Model \(DFM\)](#) with a sequence model and is composed of two main components:

1. The denoising network $p_{1|t}^\theta$, which is trained to reconstruct clean trajectories from noisy samples (see [Algorithm 1](#)).
2. The transition rate matrix $R_t(\mathbf{x}^t, \mathbf{x}'^t | \mathbf{x}^1)$, which defines a [Continuous-time Markov chain \(CTMC\)](#) governing the iterative sampling process.

Together, these components support flexible, guided planning through the sampling procedure outlined in [Algorithm 2](#). We also introduce several architectural and objective design choices that enhance planner performance, discussed in subsequent sections. For full implementation and rate matrix details, see [Appendix A.5](#).

3.2 Preliminaries and Problem Statement

Notations. We model discrete data as a sequence $(x_1, \dots, x_H)$, where H is the horizon. This sequence is represented by $\mathbf{x}$, with each x_k denoting a step in the sequence. The superscript x^t indicates the time step in the CTMC, with $t \in [0, 1]$. Each $x_k^t \in X$ takes a discrete value from the set $X = \{1, \dots, x_{|X|}\}$, with $|X|$ denoting the cardinality of this set. This notation applies to any discrete variable, such as a state s or an action a . We denote the dataset by $\mathcal{D}$, representing a collection of demonstrations. We use $\mathcal{C}$ and $\mathcal{U}$ to denote categorical and uniform distributions, respectively. Additionally, we utilize $\delta\{i, j\}$ to represent the Kronecker delta, which equals 1 when $i = j$ and 0 otherwise.

Reinforcement Learning

We extend the standard Markov Decision Processes (MDP) framework by incorporating a sequence of goals $G = \{g_1, \dots, g_n\}$ within the state space $\mathcal{S}$. Formally, we consider learning in a MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. The MDP tuple comprises $\mathcal{S}$ and $\mathcal{A}$ to denote the state and action spaces, respectively, with discrete states $s_k \in \mathcal{S}$ and actions $a_k \in \mathcal{A}$. The transition function $\mathcal{P}(\cdot | s_k, a_k)$ returns a probability distribution over the next states given a state-action pair. The reward function $\mathcal{R}$ provides a binary reward $r_k(s_k, a_k, s_{k+1}) = \mathbb{I}\{s_{k+1} \in G\}$. The aim is to learn a policy $\pi_\theta(a_k | s_k)$ that maximizes cumulative rewards. While this approach allows for policy optimization, it is often restricted to single-step rollouts. In contrast, we focus on a multi-step planning framework that enables planning over a longer horizon, avoiding locally optimal decisions that may lead to global dead-ends.

Reinforcement Learning as Sequence Modeling

Decision Transformer (DT) [11] frames Reinforcement Learning (RL) as a sequence modeling problem, allowing training through upside-down RL [70] in a supervised manner. Given a dataset $\mathcal{D} = \{\tau_i \mid 1 \leq i \leq N\}$ of near-optimal trajectories collected through demonstrations, where each trajectory $\tau_i = (s_1, a_1, g_1, \dots, s_H, a_H, g_H)$ has a length H , DT aims to train an agent π_θ by minimizing the cross-entropy loss $\mathcal{L}_{\text{CE}}(\hat{\mathbf{a}}, \mathbf{a})$ between the predicted actions $\hat{\mathbf{a}}$ and the true actions $\mathbf{a}$ in the demonstrations. Although this BC-based approach has been successful in various robotic tasks, it lacks the ability for sequential refinement and is prone to error accumulation. Our approach addresses these limitations by integrating sequence models with an iterative denoising method for more efficient multi-step planning.

Discrete Flow Model

For simplicity, we assume $H = 1$. However, this can be extended to multidimensional data by employing factorization assumptions, as demonstrated in [8]. Consider the probability flow p_t as marginal distribution of x^t (samples in the CTMC). The objective of generative flow models is to transform source (noise) samples $p_0(x^0) = p_{\text{noise}}(x^0)$ to target (data) samples $p_1(x^1) = p_{\text{data}}(x^1)$. The probability that x^t will jump to other states j is determined by a rate matrix $R_t \in \mathbb{R}^{|X| \times |X|}$. Thus, we can represent the transition probabilities for an infinitesimal time dt :

$$p_{t+dt|t}(j|x^t) = \begin{cases} R_t(x^t, j)dt & \text{for } j \neq x^t \\ 1 + R_t(x^t, x^t)dt & \text{for } j = x^t \end{cases} \quad (3.1)$$

$$= \delta\{x^t, j\} + R_t(x^t, j)dt \quad (3.2)$$

where $\delta\{i, j\}$ equals 1 when $i = j$ and 0 otherwise. We define a forward corruption process (data-to-noise interpolation) as $p_t(x^t) = \mathbb{E}_{p_{\text{data}}(x^1)} [p_{t|1}(x^t|x^1)]$, with p_t represented by the conditional flow $p_{t|1}(\cdot|x^1)$. We consider two types of interpolants: masking-based and uniform noise, both modeled as categorical distributions ($\mathcal{C}$):

$$p_{t|1}^{\text{mask}}(x^t | x^1) = \mathcal{C} (t\delta\{x^1, x^t\} + (1-t)\delta\{[\mathbf{M}], x^t\}) \quad (3.3a)$$

$$p_{t|1}^{\text{unif}}(x^t | x^1) = \mathcal{C} \left(t\delta\{x^1, x^t\} + (1-t)\frac{1}{|X|} \right) \quad (3.3b)$$

Here, $[\mathbf{M}]$ represents a mask state [15] and $\frac{1}{|X|}$ denotes uniform prior. To simulate the reverse flow (interpolation from noise to data), we need to determine the rate matrix R that generates these marginals. This is expressed as $\partial_t p_t(\cdot) = R_t(\cdot|\cdot)^\top p_t$. Following [8], we employ a conditional rate matrix defined as $R_t(x^t|j) = \mathbb{E}_{p_{1|t}}[R_t(x^t, j|x^1)]$. This formulation enables simulation of the conditional flow by sampling along a Continuous-time Markov chain.

Learning a denoising model. The $p_{1|t}$ term is analytically intractable, but it can be approximated using a neural network with parameters θ , and this model is commonly referred to as a denoising model ($p_{1|t}^\theta$) in literature.

Challenges. Given an offline dataset, we wish to learn a planner that allows for sampling multi-step plans for a given task and environment. Primarily, we wish to address the following challenges: (1) Unconditional trajectory generation (goal position not available to the planner). (2) Even when goals are specified, goal-conditioned trajectories may provide insufficient guidance for long-horizon tasks, especially when the goal position is far from

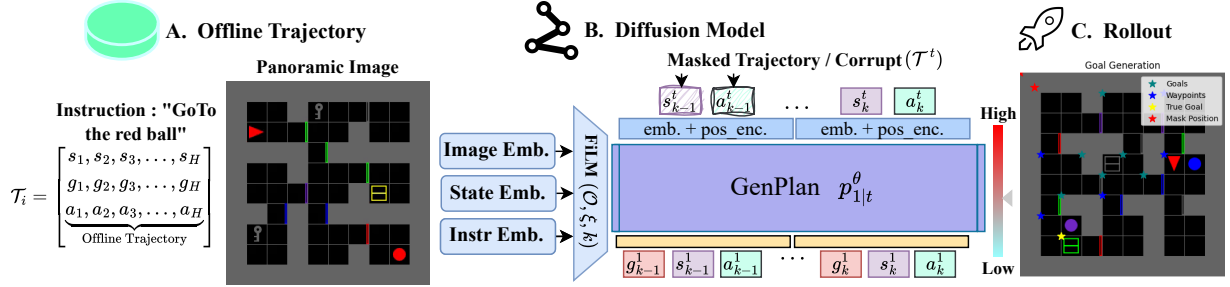


Figure 3.2: **Method Overview.** GenPlan, trained on offline data (A), learns to jointly model action, goal, and state distributions. In (B), the joint denoising model (see Section 3.3.2) takes in a corrupted trajectory τ^t and predicts the clean trajectory τ^1 . (C) Demonstrates the joint inference of goals and actions by simulating the reverse CTMC, as detailed in Algorithm 2.

the agent’s current position. (3) The agent may fail to adapt to multi-goal settings (harder) at test time, as BC-based objectives result in memorization and are prone to accumulating errors over a long planning horizon. (4) Need for stochasticity in the planner, as we may encounter tasks previously unencountered (e.g., door-unlocking tasks) or unfamiliar environments (e.g., mazes).

Problem 1 To address these challenges, we propose GenPlan (Figure 3.2), which frames planning as an iterative denoising process. Formally, the objective is as follows:

$$\mathbf{a}^* = \arg \min_{\mathbf{a}^1} \mathbb{E}_{\mathbf{a}^0 \sim p^0, \mathbf{o} \sim \mathcal{D}} [\mathcal{E}(p_{1|t}^\theta(\mathbf{a}^1 | \mathbf{a}^0, \mathbf{o}))] \quad (3.4)$$

Given a corrupted action sequence $\mathbf{a}^0$ sampled from a noise distribution, and a task description or environment observations $\mathbf{o} = (o_1, \dots, o_{ctx})$ (e.g., a panoramic image), where ctx denotes the context length, our objective is to jointly denoise the corrupted sequence to obtain samples that closely approximate the true sequence $\mathbf{a}^1$. Here, $\mathcal{E}$ represents the energy function (see Section 3.3.1).

3.3 GenPlan: Method

3.3.1 Energy-Guided Denoising Model

Given demonstrations (see Section 3.2), our aim is to learn an implicit energy function for a sequence, denoted as $\mathcal{E}(\mathbf{a}^t)$. This energy function is designed to assign lower energy to optimal action sequences. We define the energy as the sum of negative pseudo-likelihood over the horizon, formulated as $\mathcal{E}(\mathbf{a}^1) = \mathbb{E}_{(\mathbf{a}^1) \sim \mathcal{D}} \sum_H [-\log p_{1|t}^\theta(\mathbf{a}^1 | \mathbf{a}^0, \mathbf{o})]$ adapted from [28].

Our approach leverages the DFM objective [8, 7] to learn a locally normalized energy score. This score allows us to evaluate generated rollouts and frame planning as an iterative denoising process. Energy guidance ensures that the optimization process stays at an energy minimum at each step [28, 10]. The denoiser $p_{1|t}^\theta$ is optimized under the constraint that its entropy $\mathcal{H}$ remains above a lower bound β . This constraint encourages stochasticity, enhancing the generative model’s adaptability to new tasks and environments.

$$\min_{\theta} \mathbb{E}_{\mathbf{a}_{1:H}^0 \sim p_0, (\mathbf{o}_{1:H}, \mathbf{a}_{1:H}^1) \sim \mathcal{D}} \left[\sum_{k=1}^H -\log p_{1|t}^\theta(a_k^1 | a_k^0, o_k) \right], \quad (3.5a)$$

$$\text{s.t. } \mathbb{E}_{\mathbf{a}_{1:H}^0 \sim p_0, (\mathbf{o}_{1:H}, \mathbf{a}_{1:H}^1) \sim \mathcal{D}} \left[\sum_{k=1}^H \mathcal{H}(p_{1|t}^\theta(a_k | a_k^0, o_k)) \right] \geq \beta \quad (3.5b)$$

By optimizing at the sequence level, the model avoids getting trapped in local minima and can naturally learn a multimodal distribution, where $\mathcal{E}(\mathbf{a}) \simeq \mathcal{E}(\mathbf{a}')$, finding multiple viable solutions. This framework enables the flexible formulation of RL and planning tasks. In Section 3.4.1, we demonstrate the use of a joint model (Section 3.3.2) for generating unconditional rollouts.

3.3.2 Joint Denoising Model

Goal Generation. GenPlan addresses complex planning tasks by using instruction prompts or the environment’s observations to define objectives. We achieve this by learning a goal distribution conditioned on observations $\mathbf{o}$, which guides the action denoising process. Unlike existing methods [11, 10] that rely on simulators for runtime goal positions and task proposals, GenPlan jointly learns the goal distribution, extracting a goal sequence $\mathbf{g}$ jointly with actions $\mathbf{a}$ during planning. This dynamic goal proposal promotes exploration and improves performance in long-horizon tasks by reducing error accumulation.

State Sequence. In addition to goal generation, we also learn a state denoising model. We found that goal-conditioning becomes ambiguous when the proposed goal is far from the agent’s position. To prevent the agent from stalling or getting stuck in local regions, we learn the state sequence, which further aids in learning the action distribution.

These auxiliary modules are trained in the same manner as the action sequence (denoising-based [DFM](#)) and can generalize to out-of-distribution tasks, provided the energy function itself generalizes—assigning low energy to successful trajectories. Next, we briefly describe the processes involved in training and sampling from [DFM](#) [8, 7].

Forward Diffusion. We begin by corrupting samples drawn from the dataset $\mathcal{D}$ using the noise schedules (eq. 3.3a, 3.3b). Specifically, we apply $\mathbb{E}_{p_{\text{data}}(\tau^1), t \sim \mathcal{U}[0,1]} p_{t|1}^{\text{mask}}(\tau^t | \tau^1)$, where t is the [CTMC](#) timestep that controls the amount of corruption. The corruption process is applied to $\mathbf{s}$, $\mathbf{a}$, and $\mathbf{g}$, with the jump rate controlled by t . This training is independent of the rate matrix, offering greater flexibility during inference.

Backward Diffusion. The denoising objective is applied at the trajectory level, making the transition from noise to data distribution challenging, particularly for longer horizons. This requires an iterative process. Using the corrupted tokens as input, we train the joint denoising model ($p_{1|t}^\theta$) to approximate the true data distribution. We employ a negative-log-likelihood-based loss (eq. 3.6) to predict $\mathbf{s}^1$, $\mathbf{g}^1$, and $\mathbf{a}^1$. As noted in [8], the denoiser $p_{1|t}^\theta(\tau^1 | \tau^t)$ is independent of the choice of rate matrix used for sampling, $R_t(\mathbf{x}^t, \mathbf{x}^t | \mathbf{x}^1)$. This offers flexibility during sampling and simplifies the training. Following Algorithm 2, we can sample the reverse [CTMC](#) to generate rollouts.

Observation Conditioning. We employ a transformer-style architecture [11], using bi-directional masks as in [15] to allow future actions to influence preceding ones. Observations $\mathbf{o}$ are encoded using FiLM (Feature-wise Linear Modulation) [63] to process images I , instruction prompts ξ , and agent positions. The context length is typically set to 1 but can be extended to increase the agent’s memory (see Figure 3.2B).

3.3.3 Training Objective

$$\mathcal{L}_{\mathbf{x}} = \left[- \sum_{k=1}^H \delta \{x_k^t, [\mathbf{M}]\} \log p_{1|t}^\theta(\mathbf{x}^1 | \mathbf{x}^t, \mathbf{o}) \right] \quad (3.6)$$

Algorithm 1 GenPlan Training

```
1: init denoiser  $p_{1|t}^\theta$ ,  $\tau^0 \sim p_{\text{noise}}$ ,  $\beta = 0.5$ ,  $\text{maxiters} = 5k$ 
2: for  $i$  in  $\text{maxiters}$  do
3:    $t \sim \mathcal{U}[0, 1]$ ,  $\tau^1 \sim \mathcal{D}$ ,  $\mathbf{o} \sim \mathcal{D}$ 
4:    $p_{t|1}^{\text{mask}}(\tau^t | \tau^1)$  // see eq. 3.3a
5:    $\mathcal{L}_{\text{NLL}} \leftarrow \mathcal{L}_{\mathbf{a}} + \mathcal{L}_{\mathbf{s}} + \mathcal{L}_{\mathbf{g}}$  // see eq. 3.6
6:    $\mathcal{L}_{\text{ent}} \leftarrow \mathbb{E}_{\mathbf{a} \sim \mathcal{D}} \left[ \mathcal{H} \left( p_{1|t}^\theta(\mathbf{a} | \tau^0, \mathbf{o}) \right) \right]$  // see eq. 3.5b
7:    $\mathcal{L}_\pi(\theta) = \mathcal{L}_{\text{NLL}} - \lambda \mathcal{L}_{\text{ent}}$ 
8:    $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}_\pi(\theta)$ 
9:    $\lambda \leftarrow \lambda - \left( \mathcal{H} \left[ p_{1|t}^\theta(\cdot | \tau^0, \mathbf{o}) \right] - \beta \right)$ 
10: end for
```

The training objectives described in eq. 3.5a, 3.5b are used to learn the model parameters (θ, λ) . Here, θ represents the parameters of the denoiser $p_{1|t}^\theta$, which is parameterized as a transformer, as discussed in Section 3.3.2. The parameter $\lambda \in [0, \infty]$ is the Lagrangian multiplier associated with the constraints in eq. 3.5b. We alternate gradient descent steps between updating $p_{1|t}^\theta$ and λ . In practice, $\lambda \rightarrow 0$, ensuring that the lower bound β on entropy is satisfied [89]. To generalize the denoiser across different levels of corruption, we train the model to recover trajectories with varying noise levels. For each $\langle \mathbf{a}, \mathbf{s}, \mathbf{g} \rangle$, we apply the loss $\mathcal{L}_{\mathbf{x}}$ (see eq. 3.6). The Kronecker delta ensures that the loss is only calculated for the corrupted regions.

3.3.4 Planning

Once the denoising model has been trained, we can generate trajectories that approximate the data distribution by reversing the CTMC and interpolating noise back into the data through iterative denoising (I_{max} iterations). The following algorithm outlines this process, based on the approach in [8]. This procedure is applied to all components of the trajectories.

Algorithm 2 GenPlan Sampling

```
1: init  $\tau^0 \sim p_0$ , choice of  $R_t(\tau^t, \cdot | \tau^1)$ ,  $\Delta t = \frac{1}{I_{max}}$ , get  $\mathbf{o}$ 
2: for  $t \in \{0, \Delta t, 2\Delta t, \dots, 1\}$  do
3:    $R_t^\theta(\tau^t, \cdot) \leftarrow \mathbb{E}_{p_{1|t}^\theta(\tau^1 | \tau^t, \mathbf{o})} [R_t(\tau^t, \cdot | \tau^1)]$ 
4:    $\tau^{t+\Delta t} \sim \mathcal{C}(\delta\{\tau^t, \tau^{t+\Delta t}\} + R_t^\theta(\tau^t, \tau^{t+\Delta t})\Delta t)$ 
5:    $t \leftarrow t + \Delta t$ 
6: end for
7: return  $\mathbf{a}, \mathbf{s}, \mathbf{g}$  // extract from  $\tau^1$ 
```

Remark 1 *Extending rate matrices to sequences: For multidimensional input, line 4 of Algorithm 2 factorizes $\mathcal{C}(\cdot)$ as shown below. This is due to the following reasons: (1) The corruption process is independent across the horizon, meaning R_t^k depends only on x_k^t, j_k , and x_k^1 . (2) Once the process reaches x_k^1 , it remains there, i.e., $R_t^k(x_k^t = x_k^1, j_k | x_k^1) = 0$.*

$$\delta\{\mathbf{x}^t, \mathbf{x}'^t\} + \sum_{k=1}^H \delta\{\mathbf{x}_{\setminus k}^t, \mathbf{x}_{\setminus k}'^t\} \mathbb{E}_{p_{1|t}^\theta(x_k^1 | \mathbf{x}^t)} \left[R_t^k(x_k^t, j_k | x_k^1) \right] \Delta t$$

3.4 Simulation Studies

We evaluate the performance of GenPlan in BabyAI [12] and continuous manipulation tasks, focusing on the agent’s adaptive and generalization capabilities. We implemented GenPlan using Python 3.8 and trained it on a 12-core CPU alongside an RTX A6000 GPU.

3.4.1 BabyAI

BabyAI offers a diverse set of tasks and environments (discrete) focused on planning and task completion.

Simulation Setup

We conduct simulations in a modified BabyAI suite following three paradigms.

1. **Trajectory Planning (TP)**. The agent navigates to one or more goals in a maze world, with map layouts, agent initialization, and goal positions varied in each run, evaluating GenPlan’s generalization.

2. **Instruction Completion (IC)**. The agent operates in a multi-objective environment requiring complex decision-making, including *exploration*, *object manipulation*, *key collection*, and *sequential goal completion*. Task order and navigation are critical.
3. **Adaptive Planning (AP)**. The model, trained on simple goal-reaching tasks, is evaluated for zero-shot adaptation across harder environments without additional fine-tuning. This includes testing the agent in increasingly complex environments, such as mazes with multiple goals and closed doors, where the agent must demonstrate door-opening and navigation skills. Additionally, we assess the model’s ability to adapt to environments in which obstacles must be unblocked to succeed.

To navigate and interact with the environment, the agent can choose from six actions: *left*, *right*, *forward*, *open*, *drop*, or *pick up*. Success rates in reaching goals and completing tasks are reported across 250 novel environments. The map layout, goals, obstacles, and agent positions are randomized in each run. As described in Section 3.3.2, the planner uses an image observation and instruction to denoise the action sequences. Further details on the environment, baseline configurations, and implementation are provided in Appendix A.1.

Baselines

We evaluate both versions of GenPlan, named GenPlan-M (GP-M) and GenPlan-U (GP-U), corresponding to the masked and uniform noise variants, respectively. These are compared against several baselines:

1. **LEAP** [10]: A **Masked Language Model (MLM)** based multi-step planner, which has access to simulator-based goal-conditioning for sub-goals.
2. **LEAP $\ominus$ GC**: A variant of LEAP where simulator access is removed to evaluate its unconditional rollouts.
3. **LEAP $\oplus$ $\mathcal{H}$** . A variant of LEAP that incorporates a lower bound on policy entropy (see eq. 3.5b), enabling improved adaptation to novel tasks.
4. **DT** [11]: Utilizes a causal transformer for planning, also with simulator-based goal-conditioning.

While DT and LEAP could learn goal distributions, the MLE objective suffers from poor generalization, often predicting goal positions as corner cells. In contrast, the generative

objective combined with DFM sampling in GenPlan demonstrates superior generalization (Figure 3.2C). To improve competitiveness, baselines with simulator access to goal positions are categorized as conditional rollouts, while those relying on instructions and images to recover successful trajectories are categorized as unconditional rollouts. Comparative results are presented in Tables 3.1 and 3.2.

Remark 2 *LEAP outperforms popular baselines, including model-free [RL](#) algorithms like Batch-Constrained Deep Q-Learning [24] and Implicit Q-Learning [48], as well as model-based [RL](#) methods such as the Planning Transformer [76] and Model-based Offline Policy [88], in single-goal tasks within BabyAI environments, as shown in Table 1 of LEAP [10].*

Env.	Uncond. Rollouts			Cond. Rollouts	
	GenPlan-U	GenPlan-M	LEAP $\ominus$ GC	LEAP	DT
Traj. Planning (TP)					
GoToObjMazeS4G1	52.4%	62%	44%	49.2%	46.8%
GoToObjMazeS4G2	38.8%	39.6%	20%	37.6%	35.2%
GoToObjMazeS7G1	45.6%	44.8%	12%	33.2%	40%
GoToObjMazeS7G2	21.2%	19.6%	3.6%	4%	13.6%
TP Mean (7.6 $\uparrow$)	39.5%	41.5%	19.9%	31%	33.9%
Instr. Completion (IC)					
GoToObjMazeClose	42.8%	48.4%	18%	38.8%	40%
DoorsOrder	40.8%	35.2%	11%	36.4%	40.8%
BlockUnlock	13.2%	16%	0%	0.8%	0%
KeyCorridorS3R3	11.6%	17.6%	0%	0.4%	3.6%
IC Mean (8.2 $\uparrow$)	27.1%	29.3%	7.25%	19.1%	21.1%

Table 3.1: **BabyAI quantitative performance.** Success rates of the models across different environments are presented. The abbreviations **SW**, **NX**, **RY**, and **GZ** in the environment names represent the size (**W**) of a room in the map, the number of obstacles (**X**), the number of rows (**Y**), and the number of goals (**Z**) during testing, respectively. The term “*Close*” indicates that the agent requires door-opening actions. See Appendix A.1.2 for more details.

	Uncond. Rollouts				Cond. Rollouts	
Environment	GenPlan-U	GenPlan-M	LEAP $\ominus$ GC	LEAP $\oplus$ H	LEAP	DT
Adaptive Planning (AP)						
GoToLocalS10N10G2	82.4%	88%	76%	69.2%	78%	25.6%
GoToObjMazeS4N3G1	56%	62%	44.8%	52%	48%	24%
GoToObjMazeClose	31.2%	34.8%	10%	16.4%	10%	8.8%
GoToObjMazeS4G2	28.8%	34.8%	14%	21.2%	18.4%	3.6%
GoToSeqS5R2Un	35.6%	42%	29.2%	30.8%	38%	29.2%
AP Mean (13.84 $\uparrow$)	46.8%	52.32%	34.8%	37.92%	38.48%	18.24%

Table 3.2: **BabyAI quantitative performance.** Success rates of the models across different environments are presented. The abbreviations **SW**, **NX**, **RY**, and **GZ** in the environment names represent the size (**W**) of a room in the map, the number of obstacles (**X**), the number of rows (**Y**), and the number of goals (**Z**) during testing, respectively.

Results

GenPlan consistently achieves higher success rates, particularly in adaptive planning and long-horizon tasks, as shown in Table 3.1. Success rates decline as environments grow or involve multiple goals (e.g., *MazeS7G2*), reflecting the challenges of multi-step planning, where committing to incorrect plans results in deadlocks. GenPlan succeeds in complex, order-critical multi-task missions that require completing sub-tasks (e.g., key collection, obstacle unblocking) to achieve higher-level objectives, where baselines fail, particularly in *KeyCorS3R3* and *BlockUn*. In tasks requiring strict task order (e.g., *DoorsOrder*), [Decision Transformer \(DT\)](#) performs comparably due to simulator access for correct goal sequencing, while GenPlan relies solely on instruction embedding as seen in Section 3.3.2.

Next, we assess the model’s adaptation capabilities in novel environments (see Table 3.2). Initially trained in a simple, single-goal plane world task, the model is then evaluated on increasingly challenging tasks. Through the joint denoising process, GenPlan can eventually find a plan given enough timesteps as long as the energy function generalizes well. The goal-generation module allows dynamic updates of sub-goals, aiding exploration, while the entropy regularizer helps the model learn new skills, such as obstacle unblocking, even without explicit demonstrations in the dataset. We hypothesize that this is due to GenPlan’s more effective joint optimization process. In practice, LEAP’s sampling process may lead to misalignment between generated actions and goals, even with an oracle for the goal sequence (see Figure 3.3).

Ablation Studies

We assess the significance of various components of GenPlan, results are presented in Table 3.3.

Joint Prediction ($\mathcal{L}_s + \mathcal{L}_g$). This integration of goal generation ($\mathcal{L}_g$) with state sequence denoising ($\mathcal{L}_s$) enhances IC performance and prevents stalling in TP tasks. Without state denoising, the agent tends to exhibit stalling actions as the agent may get stuck in a local region. On the other hand, without $\mathcal{L}_g$, the agent lacks awareness of sub-tasks associated with the overall task. This is evident in the performance drops observed in the *KeyCorS3R3* and *DoorsOrder* experiments in Table 3.3, where the agent struggles to achieve sub-goals like key collection or correctly identifying the door sequence.

Entropy. Entropy acts as a regularizer while minimizing the trajectory’s energy. LEAP often gets stuck in local minima when recovering trajectories, leading to the hallucination of actions from demonstrations, such as constantly moving forward or performing in-place turns. The impact of entropy is particularly significant in AP tasks, where the agent navigates more complex environments. This is evident in the performance drops observed in the *MazeClose* environment. However, in IC tasks, where additional actions like “open” or “pickup” are required, entropy often hinders performance, making it less suitable for such scenarios.

Noise Schedule. In Tables 3.1 and 3.2, we compare the use of uniform interpolants versus masking. While masking interpolants generally outperform uniform interpolants in most tasks, the latter succeed in some larger environments. This success is attributed to the inherent randomness in the denoising process, as discussed in Section 3.3.4. With masking, the model more easily identifies jumps, simplifying the training. This is also reflected through faster convergence (training) of masked interpolants.

Comparisons with Generative Approaches

We evaluate different generative objectives (BC, Noise Contrastive Estimation) and sampling techniques (Random Shooting (RS), Cross-Entropy Method (CEM), Markov chain Monte Carlo (MCMC) sampling, and energy gradient-guided sampling). The results for the AP and IC tasks are reported in Table 3.4.

Attribute	GenPlan-M	Reduction	LEAP $\ominus$ GC
MazeS4G2 (TP)			
w/o JP	32%	↓ 7.6%	20%
w/o Entropy	34.4%	↓ 5.2%	
KeyCorS3R3 (IC)			
w/o JP	13.6%	↓ 4%	0%
w/o Entropy	7.6%	↓ 10%	
w/o History	0.4%	↓ 17.2%	0%
DoorsOrder (IC)			
w/o JP	15.6%	↓ 19.6%	11%
w/o Entropy	34.4%	↓ 0.8%	
SeqS5R2Un (AP)			
w/o JP	32.8%	↓ 9.2%	29.2%
w/o Entropy	34.8%	↓ 7.2%	

Table 3.3: **Ablation.** The values represent the mean success rates as detailed in Section 3.4.1, with reductions shown relative to the results in Tables 3.1 and 3.2.

(i) **Denoising Diffusion Probabilistic Models (DDPM)** [38, 13] rely on a BC objective thus struggle to generalize to harder tasks, as shown in Example 1. Models trained solely with BC objectives lack distributional fidelity, limiting their ability to adapt to novel scenarios.

(ii) **Energy models** are challenging to sample from due to their reliance on extensive CEM or MCMC cycles and their instability during training [13]. Additionally, guided sampling in DDPM (using energy gradients) often results in local minima, where low-energy trajectories are gamed through repeating actions frequently observed in the dataset (e.g., repeated forward movements).

We observe that GenPlan significantly outperforms both approaches, as demonstrated in Table 3.4.

Discussion

Energy Landscape. We investigate the energy assignment to trajectories with varying noise levels throughout training and observe that (a) GenPlan learns to reduce the energy for clean sequences and (b) increases the energy for corrupted trajectories (Figure 3.4B). Since this denoising is done jointly, the energy measure provides an implicit sense of the

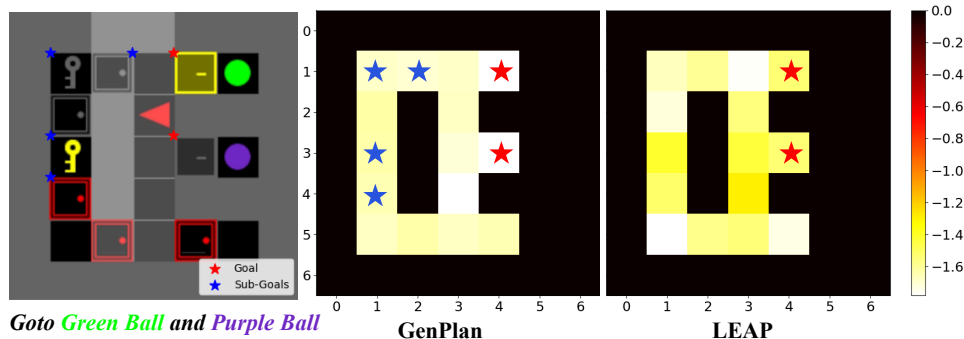


Figure 3.3: **Energy Landscape.** GenPlan, when conditioned on sub-goals, implicitly assigns minimal energy to necessary sub-goals (e.g., picking up keys, opening doors) for task completion. States closer to the white region are more likely to be transitioned into. LEAP, in contrast, does not prioritize these sub-tasks.

goodness of an action at a particular state (also see Figure 3.3). This helps verify that the objective in eq. 3.5a indeed facilitates denoising.

Effect of Iterative Denoising. At inference, the number of denoising steps for planning is determined independent of training (Algorithm 2), enabling post-hoc adaptations with various rate samplers. For harder problems, more iterations or better samplers can be used to obtain minimal energy trajectories. From Figure 3.4A, we observe that around $H/2$ iterations provide good results, allowing approximately two dimensions to transition simultaneously in a single step.

Coverage. We perform coverage experiments on both adaptive planning (AP) and trajectory planning (TP) tasks in novel environments. For both tasks, we evaluate models across unseen environments where the map layout (fixed) was not observed during training, and the goal position is randomized to different rooms. The results, shown in Figure 3.5, report the coverage and success rates for each model.

In AP tasks, where both the task and environment are novel, the agent is trained on a planar goal-conditioned task (without walls) and must adapt to a maze structure in a zero-shot setting. As illustrated in Figure 3.5, GenPlan significantly outperforms the baselines due to its goal generation module (Section 3.3.2), which effectively proposes sub-goals and goal regions, providing clear guidance. In contrast, baseline models struggle with ambiguous guidance (Figure 3.3), often leading to agent stalling, even with access

Sampling / Objective		Traj. Planning	Instruction Completion	
		GoToObjMazeS4G1	KeyCorridorS3R3	DoorsOrder
Energy Models	Random	29.2%	2.8%	26.0%
	CEM	38.9%	6.4%	28.4%
	MCMC	44.0%	0.0%	11.0%
DDPM	Diffusion BC	25.2%	0.8%	29.2%
	Energy (Gradient)	2.0%	0.0%	0.0%
GenPlan	Energy+DFM	62.0%	17.6%	35.2%

Table 3.4: **Comparison of Energy-Based and DDPM Baselines on MiniGrid Tasks.** We compare various sampling techniques and generative objectives on discrete planning tasks. GenPlan demonstrates significant improvements in success rates, particularly on more complex tasks, by combining energy objectives with DFM sampling.

to the simulator for ground truth. GenPlan demonstrates efficient exploration and strong generalizability in these settings.

For **TP** tasks, we evaluate the models’ trajectory planning capabilities in a maze-goal-reaching task with unseen maps (Figure 3.6). This setup excludes goal modules (learned or oracle) and focuses solely on trajectory planning. Given the map and instructions, GenPlan successfully plans trajectories, even when initializing goals in different rooms. In comparison, baseline models struggle, with **DT** performing competitively in generalizing to the same task but facing significant challenges in adaptation. GenPlan consistently outperforms baselines in both generalization and adaptation tasks.

Policy Deployment. We demonstrate policy rollout on the AGILEX LimoBot ¹ with a differential drive for the *GoToObjMazeS4G2* task. Using a scale of 0.3 m per grid cell, and employing onboard odometry and dead-reckoning, the supplementary video² showcases the implementation. This deployment highlights our method’s ability to bridge abstract plans to real-world execution via trajectory optimization and low-level tracking.

3.4.2 Manipulation Tasks

We evaluate the models on continuous manipulation tasks: (a) PushT [22] and (b) Franka Kitchen [30]. Both tasks use state-based variants in continuous manipulator environments.

¹Thanks to Praneeth KVK for assistance in setting up the real-world robot demonstration.

²<https://aku02.github.io/projects/adaptformer/>

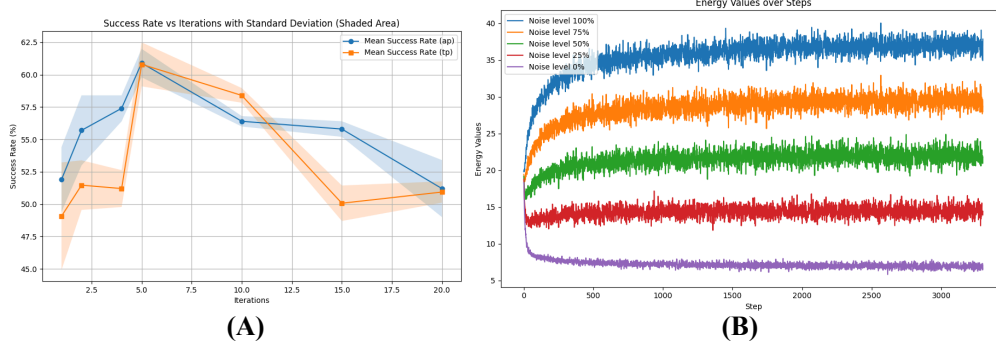


Figure 3.4: **GenPlan Analysis.** (A) Success rate as a function of the number of de-noising iterations, reported using 3 different training seeds evaluated on 250 environments, specifically for *GoToObjMazeS4N3G1* (AP) and *GoToObjMazeS4G1* (TP). (B) Energy landscape learned by GenPlan during training, illustrating that GenPlan effectively captures the true action distribution by assigning low energy to noise-free sequences.

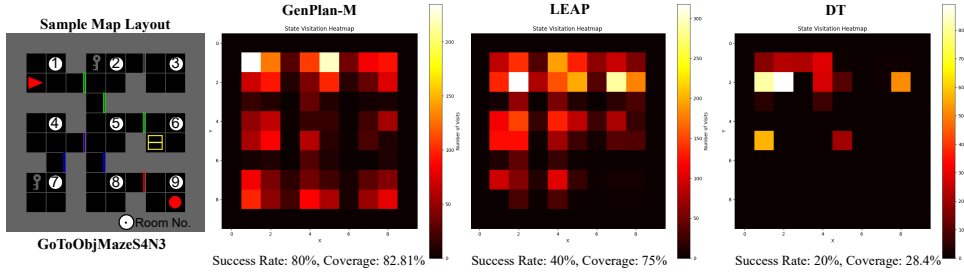


Figure 3.5: **State Coverage (AP).** State visit frequency is evaluated across 10 unseen maze layouts with varying goal positions (Rooms 1-9), starting from a fixed agent position (Room 1).

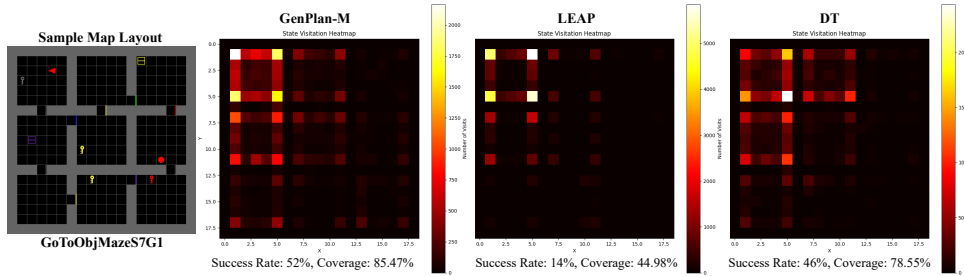


Figure 3.6: **State Coverage (TP).** Coverage is evaluated in larger environments. Results are reported across 50 map variations from a fixed start position (Room 1).

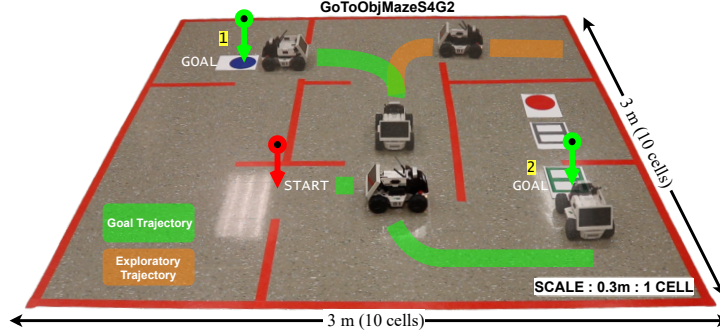


Figure 3.7: **Policy Deployment.** We deploy the policy on an AGILEX LimoBot. See the [project webpage](#) for more details.

The PushT task assesses task comprehension in unconditional rollouts, with performance measured by final coverage, defined as the IoU between the T-block and the target position. The Franka Kitchen task involves seven possible subtasks, where each demonstration trajectory completes a subset of four in some order. It evaluates long-horizon planning, multimodality, and task commitment, where certain baselines may fail due to mode-shifting or incomplete tasks. Both tasks are evaluated in an unconditional rollout setting. For details on the environment setup and implementation, see Appendix A.3.

Baselines

For our evaluations, we compare GenPlan-M with several baselines to highlight its performance in continuous tasks. GenPlan-M retrieves continuous actions by learning a categorical distribution over discrete latent codes, with corresponding offsets, similar to the Vector Quantized Behavior Transformer (VQ-BeT) [53].

Unlike VQ-BeT, which employs a behavior cloning objective, GenPlan-M uses an energy-based loss (see Section 3.3) and DFM sampling with causal masks, providing greater flexibility and improved generalization. Additionally, we benchmark against Diffusion Policy (DP), a state-of-the-art behavior cloning method, evaluating both its CNN-based (DP-C) and Transformer-based (DP-T) variants. For model and environment hyperparameters, we adopt the configurations from [53].

Env	Metric	GenPlan-M	VQ-BeT	DP-C	DP-T
PushT	Final Coverage	0.73	0.7	0.73	0.74
	Max Coverage	0.77	0.73	0.86	0.83
Kitchen	# Tasks	3.40	3.66	2.62	3.44

Table 3.5: **Quantitative results.** Comparison of baselines on unconditional continuous-space tasks.

Discussion

GenPlan-M demonstrates competitive performance against state-of-the-art behavior cloning approaches. While diffusion policies excel at low-level tasks such as PushT, GenPlan-M and VQ-BeT outperform the diffusion-based approach (DP-C) in the Kitchen environment, where diffusion lags behind by an entire subtask. *In particular, on PushT, GenPlan-M trails DP-T in max-coverage because its categorical, abstract parameterization trades fine-grained continuous-control dexterity for compositional planning; conversely, in Kitchen this abstraction improves long-horizon subtask sequencing and yields stronger performance.* These results highlight the advantages of modeling with categorical distributions (abstracted spaces) for multi-task missions and unconditional planning.

Unlike baselines that rely on an oracle for goal positions, often unavailable in real-world scenarios the goal generation module in GenPlan dynamically updates sub-goals, enabling improved state coverage (see Appendix A.1). To further illustrate these advantages, we present results on MetaWorld tasks, where we train a multi-task policy and evaluate the agent’s adaptive capabilities on unseen tasks (see Appendix A.4).

Chapter 4

Learning for Multi-Agent Games

4.1 Overview

Decision-making in multi-agent settings, particularly in dynamic and continuous control environments, poses unique challenges. Agents must reason about the evolving strategies of others, adapt to diverse opponent behaviors, and maintain stability under non-stationary training dynamics. In this work, we propose DiffFP, a generative, game-theoretic framework that leverages diffusion policies to compute best responses within a [Fictitious Play \(FP\)](#) setup.

Unlike standard reinforcement learning approaches that struggle to model multimodal strategies, or value-based methods that fail to represent stochastic policies effectively, DiffFP employs diffusion models to learn expressive best-response policies in continuous action spaces. By embedding a generative policy into the fictitious play loop, we achieve sample-efficient learning of robust and generalizable behaviors.

Contributions. To address the limitations of unimodal and unstable best-response policies in continuous state-action games, we introduce the following innovations:

- DiffFP [45] trains from scratch using iterative fictitious play, where each agent learns a diffusion-based best-response policy to the evolving empirical distribution of its opponents.
- The opponent’s average policy is modeled explicitly and updated online, improving stability and mitigating non-stationarity during training.

- Diffusion models enable expressive multimodal policies, allowing agents to represent mixed strategies and adapt to previously unseen opponents.

Our approach is general enough to accommodate both symmetric and asymmetric game structures, and we validate it in a range of continuous control benchmarks. Empirically, DiffFP achieves up to $3\times$ *faster convergence* and $30\times$ *higher success rates* than conventional [Reinforcement Learning \(RL\)](#) baselines, as measured by exploitability and win rate against held-out opponents.

System Components. DiffFP builds upon the [FP](#) framework with two core components:

1. The outer loop of [Fictitious Play](#), where agents maintain and update empirical average strategies (Algorithm 3).
2. A generative best-response policy implemented as a conditional diffusion model, trained using policy gradients to approximate optimal behavior against the average opponent (Algorithm 4).

These components interact to yield robust, convergent learning dynamics even in high-dimensional, continuous-action games. Subsequent sections describe the parameterization, training objectives, and sampling strategies for both the best-response policy and the average policy model.

4.2 Preliminaries and Problem Statement

We consider a [Partially Observable Markov Game \(POMG\)](#) [33] defined by the tuple $(\mathcal{I}, \mathcal{S}, b^0, \{\mathcal{A}_i\}, \{\mathcal{O}_i\}, \mathcal{T}, \Omega_i, \{R_i\})$, where:

- $\mathcal{I} = \{1, \dots, n\}$ is the finite set of agents;
- $\mathcal{S}_i \subseteq \mathbb{R}^{d_s}$ is the continuous state space for agent i , and $\mathcal{S} = \prod_{i \in \mathcal{I}} \mathcal{S}_i$ is the joint state space, with $\vec{s} = (s^1, \dots, s^n) \in \mathcal{S}$ denoting the joint state;
- $b^0 \in \mathcal{P}(\mathcal{S})$ denotes the initial state distribution;
- $\mathcal{A}_i \subseteq \mathbb{R}^{d_a}$ is the continuous action space for agent i , and $\mathcal{A} = \prod_{i \in \mathcal{I}} \mathcal{A}_i$ is the joint action space, with $\vec{a} = (a^1, \dots, a^n) \in \mathcal{A}$;

- $\mathcal{O}_i$ is the observation space for agent i , with observation function $\Omega_i : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{O}_i)$ mapping the global state to the agent's local observation (e.g., an occupancy grid);
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{S})$ is the (joint) transition function;
- $R_i : \mathcal{S} \times \mathcal{A} \rightarrow [r_{\min}, r_{\max}] \subset \mathbb{R}$ is the bounded reward function for agent i , where $r_{\min} \leq R_i(\vec{s}, \vec{a}) \leq r_{\max}$ for all $(\vec{s}, \vec{a}) \in \mathcal{S} \times \mathcal{A}$.

A (stochastic) policy for agent i is defined as a mapping $\pi^i : \mathcal{O}_i \rightarrow \mathcal{P}(\mathcal{A}_i)$, where $\mathcal{P}(\mathcal{A}_i)$ denotes the probability measures over the action space $\mathcal{A}_i$. The joint strategy of all agents is denoted as $\boldsymbol{\pi} := (\pi^1, \dots, \pi^n) = (\pi^i, \boldsymbol{\pi}^{-i})$, where $\boldsymbol{\pi}^{-i}$ represents the strategy of all agents except agent i .

Remark 3 (Two-player zero-sum game) *We consider a two-player zero-sum game, where the sum of agents' rewards satisfies $(\sum_i R_i = 0)$. Without loss of generality, this formulation extends naturally to multi-agent settings. The joint strategy is denoted as $\boldsymbol{\pi} = (\pi^{ego}, \pi^{opp})$, where π^{ego} and π^{opp} represent the behavioral strategies of the ego and opponent agents, respectively.*

Definition 1 (Best Response [5, 61]) *Given the opponent's strategy $\boldsymbol{\pi}^{-i}$, the best-response strategy π_{BR}^i for agent i is defined as*

$$\pi_{BR}^i \in \arg \max_{\pi^i \in \Pi^i} J_i(\pi^i, \boldsymbol{\pi}^{-i}), \quad (4.1)$$

where Π^i denotes the set of admissible policies for agent i , and $J_i : \Pi^1 \times \dots \times \Pi^n \rightarrow \mathbb{R}$ denotes the expected cumulative reward for agent i .

In zero-sum games [81, 56], the reward functions satisfy $R^{ego} = -R^{opp}$, resulting in a minmax formulation where one agent's gain is the other's loss. The goal is to compute joint strategies that are robust against adversarial opponents.

Definition 2 (ϵ -Nash Equilibrium [61]) *We define a joint strategy profile $\boldsymbol{\pi}^* = (\pi^{ego,*}, \pi^{opp,*})$ to be an ϵ -Nash equilibrium if, for all $i \in \{ego, opp\}$, the following holds:*

$$J_i(\pi^{i,*}, \boldsymbol{\pi}^{-i,*}) \geq J_i(\pi^i, \boldsymbol{\pi}^{-i,*}) - \epsilon, \quad \forall \pi^i \in \Pi^i. \quad (4.2)$$

Intuitively, this means that no agent can gain more than ϵ by unilaterally deviating from their current strategy. The deviation from an exact equilibrium can be quantified using the notion of *exploitability*, which measures how much an agent's return could be improved by switching to a best-response policy.

Definition 3 (Exploitability) We define the exploitability of a joint policy profile $\boldsymbol{\pi} = (\pi^i, \pi^{-i})$ for agent i as:

$$\varepsilon_i = J_i(\pi_{\text{BR}}^i, \pi^{-i}) - J_i(\pi^i, \pi^{-i}), \quad (4.3)$$

where π_{BR}^i is a best response to π^{-i} . The total exploitability is given by $\varepsilon = \sum_{i \in \mathcal{I}} \varepsilon_i$.

Problem 2 We consider a two-agent zero-sum *POMG* where agents interact in a continuous state and action space. Each agent $i \in \{\text{ego}, \text{opp}\}$ seeks to maximize its expected return ($J_i(\pi^i, \pi^{-i})$), under the assumption that its opponent behaves adversarially. This leads to a minmax optimization problem defined over joint policies.

$$J_i(\pi^i, \pi^{-i}) = \mathbb{E}_{\substack{\vec{a}_h \sim \pi(\cdot | \vec{o}_h), \\ \vec{s}_{h+1} \sim \mathcal{T}(\cdot | \vec{s}_h, \vec{a}_h)}} \left[\sum_{t=0}^H \gamma^t R_i(\vec{s}_h, \vec{a}_h) \right],$$

The goal of each agent is to solve the following minmax optimization:

$$\pi^{*,i} = \arg \max_{\pi^i \in \Pi^i} \min_{\pi^{-i} \in \Pi^{-i}} J_i(\pi^i, \pi^{-i}), \quad (4.4)$$

Without loss of generality, this formulation extends naturally to the multi-agent setting. In such cases, each agent maximizes its own return while accounting for the joint policies of all other agents, which together act adversarially in the zero-sum context.

4.3 DiffFP: Method

Notations. We use subscript h to denote timesteps in the *POMG*, and $t \in \{1, \dots, T\}$ for diffusion timesteps. Agent index is indicated by superscript i . The action taken by agent i at environment timestep h and diffusion step t is denoted as $a_{h,t}^i$. Under fictitious play, we denote the policy of agent i at iteration $k \in \{0, \dots, K\}$ as π_k^i . We use $\mathcal{U}\{1, \dots, T\}$ to denote a uniform distribution over the set $\{1, \dots, T\}$, and $\mathcal{N}$ to denote a Gaussian distribution. The symbol $\sim$ denotes sampling from a distribution; for example, $\epsilon \sim \mathcal{N}(0, I)$. For brevity, we drop the subscript h or t in equations where all terms refer to the same *POMG* timestep or diffusion step.

To solve the minmax problem in continuous zero-sum games described in Section 2, we adopt a *FP* style learning framework. Specifically, we integrate diffusion models as expressive policy classes to represent the agents' stochastic strategies in high-dimensional, continuous action spaces. We train a diffusion-based policy at each iteration to approximate the best response against a fixed opponent strategy distribution.

4.3.1 Fictitious Play

Fictitious Play [5] is an iterative algorithm in which agents update their strategies based on the empirical distribution of their opponents' past strategies. At each iteration k , each agent i computes a best response π_{BR}^i to the current average strategy π_k^{-i} of its opponents and updates its strategy

$$\pi_{k+1}^i = \frac{k}{k+1} \pi_k^i + \frac{1}{k+1} \pi_{\text{BR}}^i \quad (4.5)$$

One of the primary challenges in fictitious play lies in computing the best response policy, particularly in environments with continuous action spaces. To address this, we employ a generalized weakened fictitious play framework [54], which allows for approximate best responses and small perturbations. Formally, a sequence of mixed strategies π_k follows generalized weakened fictitious play if, for each agent i ,

$$\pi_{k+1}^i \in (1 - \alpha_{k+1}) \pi_k^i + \alpha_{k+1} \left(\text{BR}_{\epsilon_k}(\pi_k^{-i}) + M_{k+1}^i \right),$$

where the step size $\alpha_k \rightarrow 0$ and $\epsilon_k \rightarrow 0$, when $\sum_{k=1}^{\infty} \alpha_k = \infty$, and M_k^i is a perturbation term satisfying, for any constant $J > 0$,

$$\lim_{k \rightarrow \infty} \sup_j \left\{ \left\| \sum_{l=k}^{j-1} \alpha_{l+1} M_{l+1}^i \right\| \mid s.t. \sum_{l=k}^{j-1} \alpha_{l+1} \leq J \right\} = 0, \forall J \geq 0.$$

Algorithm 3 summarizes the **Fictitious Play** procedure in our framework. By iteratively refining the strategies through **FP** where RL is employed to approximate best responses we ensure that the overall exploitability decreases over time, thereby converging to an ϵ -Nash equilibrium.

4.3.2 Approximating Best Response via Reinforcement Learning

Finding an exact best response is often computationally intractable, especially in large or continuous spaces. Prior works [36, 54] have shown that instead of full-width backups, one can approximate the best response using **RL**, and similarly approximate the average strategy using supervised learning (SL) [36].

Policy Representation

To model complex and multimodal team dynamics, we propose leveraging a diffusion policy [86] to approximate the best response. While centralized modeling of joint team behavior offers computational benefits, it often falls short in capturing the inherent multimodality of optimal responses, particularly in competitive or dynamic environments. Diffusion models address this limitation by learning diverse behavior distributions, making them especially effective at representing the rich variability in agent interactions. At each iteration k of FP, we compute the best response against a fixed but unknown average opponent policy, denoted by π_k^{-i} . For brevity, when discussing diffusion-based best responses, we omit the agent index and FP index; the subscript t in the following denotes the diffusion timestep.

Denoising Diffusion Probabilistic Models (DDPM) [38, 60] form the backbone of this approach. DDPM defines a forward process that gradually perturbs the data, here an action $a_0 \sim q(a_0)$ drawn from the replay buffer is transformed into a tractable Gaussian prior $q(a_T)$ over T steps by iteratively applying Gaussian noise:

$$q(a_t|a_{t-1}) := \mathcal{N}(a_t; \sqrt{1 - \beta_t}a_{t-1}, \beta_t I), \quad \forall t \in \{1, \dots, T\} \quad (4.6)$$

where $\beta_t \in (0, 1)$ is the variance schedule. Through Markov chain property of the forward process, the distribution of any intermediate noisy action a_t can be expressed in closed form conditioned on the original action a_0 :

$$q(a_t|a_0) := \mathcal{N}(a_t; \sqrt{\bar{\alpha}_t}a_0, (1 - \bar{\alpha}_t)I), \quad \forall t \in \{1, \dots, T\} \quad (4.7)$$

with $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. The forward diffusion process (Equations 4.6 and 4.7) progressively transforms clean actions into an isotropic Gaussian distribution as $T \rightarrow \infty$.

Algorithm 3 Fictitious Play

- 1: **Initialize** π_0^i randomly for each agent $i \in \{\text{ego}, \text{opp}\}$.
 - 2: **for** $k = 0, \dots, K - 1$ **do**
 - 3: **for** each agent $i \in \{\text{ego}, \text{opp}\}$ **do**
 - 4: Compute best response: $\pi_{\text{BR}}^i \leftarrow \text{BR}(\pi_k^{-i})$.
 - 5: # Compute BR through Alg. 4
 - 6: Update strategy: $\pi_{k+1}^i \leftarrow \frac{k}{k+1} \pi_k^i + \frac{1}{k+1} \pi_{\text{BR}}^i$.
 - 7: **end for**
 - 8: **end for**
-

Algorithm 4 DiffFP as Best Response for Agent i

```

1: Initialize actor  $\pi_\theta^i$  and critic  $Q_\psi^i$  for all  $i \in \mathcal{I}$  with targets, buffer  $\mathcal{D} \leftarrow \emptyset$ , Set noise
   schedule  $\{\alpha_t\}_{t=1}^T$  with  $\bar{\alpha}_t = \prod_{j=1}^t \alpha_j$ , step size  $\eta$ , train steps  $M$  (for each FP iteration).
2: for iteration  $m = 1$  to  $M$  do
3:   # Experience Collection
4:   for timestep  $h = 0$  to  $H$  do
5:     # Sample actions following eq. 4.9
6:     Sample actions:  $a_h^i \sim \pi^i(\cdot|o_h^i)$ ,  $a_h^{-i} \sim \pi^{-i}(\cdot|o_h^{-i})$ 
7:     Construct joint action:  $\vec{a}_h = (a_h^i, a_h^{-i})$ 
8:     Transition:  $\vec{s}_{h+1} \sim \mathcal{T}(\cdot | \vec{s}_h, \vec{a}_h)$ ,  $o_{h+1}^i \sim \Omega_i(\cdot | \vec{s}_{h+1})$ 
9:     Reward:  $r_h^i = R_i(\vec{s}_h, \vec{a}_h)$ 
10:    Store  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_h^i, a_h^i, r_h^i, o_{h+1}^i)\}$  in buffer  $\mathcal{D}$ 
11:    # Train Q-function [35]
12:    Update parameters  $\psi$  following equation 4.10
13:    # Train Diffusion Policy
14:    Update parameters  $\theta$  using the loss in equation 4.8
15:    # Q-guided Action Refinement
16:    for  $n = 1$  to  $N_{\text{action-ascent}}$  do
17:       $\tilde{a}_h^{i,n+1} \leftarrow a_h^{i,n} + \eta \nabla_{a^i} \left( \min_{j=1,2} Q_{\psi_j}^i(o_h^i, a^i) \right) \big|_{a^i=a_h^{i,n}}$ 
18:    end for
19:    Add refined action to buffer:  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_h^i, \tilde{a}_h^{i,N})\}$ 
20:  end for
21: end for

```

The learned reverse process then inverts this trajectory to generate multimodal actions from the policy.

The reverse process corresponds to our policy $\pi_\theta(a_h|o_h) = p_\theta(a_{h,0:T}|o_h)$ begins from a sample $a_T \sim p(a_T) = \mathcal{N}(0, I)$ drawn from the prior and conditionally denoises via a sequence of parametrized transitions: $p_\theta(a_{h,t-1}|a_{h,t}, o_h) := \mathcal{N}(a_{h,t-1}; \mu_\theta(a_{h,t}, o_h, t), \sigma_t^2 I)$ where θ denotes the parameters of a neural network trained to approximate the true reverse dynamics $q(a_{h,t-1}|a_{h,t}, a_{h,0})$ and $\sigma_k^2 = \beta_t \frac{1-\bar{\alpha}_t-1}{1-\bar{\alpha}_t}$. The training objective is to maximize the Variational Lower Bound (VLB): $L_{\text{VLB}} := \mathbb{E}_{q(a_{0:T})} \left[\log \frac{p_\theta(a_{0:T})}{q(a_{1:T}|a_0)} \right]$. In practice, this objective simplifies to a conditional denoising score matching loss:

$$\mathbb{E}_{t \sim \mathcal{U}\{1, \dots, T\}, a_0 \sim \mathcal{D}, \epsilon \sim \mathcal{N}(0, I)} \left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} a_{0,h} + \sqrt{1 - \bar{\alpha}_t} \epsilon, o_h, t)\|^2 \right] \quad (4.8)$$

where $\epsilon_\theta(a_{h,t}, o_h, t) = \frac{\sqrt{1-\bar{\alpha}_t}}{\beta_t} (a_{h,t} - \sqrt{\bar{\alpha}_t} \mu_\theta(a_{h,t}, o_h, t))$ is a neural network trained to predict

the noise added during the forward process.

Remark 4 (CTDE for Multi-Agent Teams) *Modeling multi-agent teams under the centralized training with decentralized execution (CTDE) paradigm presents significant challenges [82]. Within the same team, agents must learn complementary roles and diverse behaviors, resulting in inherently multimodal action distributions. This coordination complexity makes it difficult for conventional methods to learn effective policies, while diffusion-based approaches are better suited to capture such diversity.*

Sampling

We iteratively recover the action following the reverse diffusion process. The actor is parameterized as a conditional diffusion model, defined as $\pi_\theta(a_h|o_h) := p_\theta(a_{h,0:T}|a_{h,T}, o_h)$. We retain only the denoised sample $a_{h,0}$, which serves as the sampled action.

$$\begin{aligned} a_{t-1} | a_t &= \frac{a_t}{\sqrt{\alpha_t}} - \frac{\beta_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \epsilon_\theta(a_t, o_h, t) + \sqrt{\beta_t} \epsilon, \\ \epsilon &\sim \mathcal{N}(0, I), \quad \text{for } t = T, \dots, 1. \end{aligned} \quad (4.9)$$

Policy Learning

Unlike a unimodal Gaussian policy, **Diffusion Policy (DP)** does not maintain an explicit likelihood function. To learn multimodal policies for each agent i , we adopt the framework from [86], which combines a behavior cloning loss (equation 4.8) with double Q-learning [35] to guide policy improvement. In practice, we use two critic networks Q_{ψ_1}, Q_{ψ_2} and their target networks $Q_{\psi'_1}, Q_{\psi'_2}$.

$$\mathbb{E}_{(o_h^i, a_h^i, o_{h+1}^i) \sim \mathcal{D}, a_{h+1,0}^i \sim \pi^i} \left[\left((r_h^i + \gamma \min_{j=1,2} Q_{\psi'_j}^i(o_{h+1}^i, a_{h+1,0}^i)) - Q_{\psi}^i(o_h^i, a_h^i) \right)^2 \right] \quad (4.10)$$

Policy improvement is achieved through action gradients that refine actions sampled from the replay buffer (line 17 of Algorithm 4). Additionally, we adopt reward-weighting [65] to prioritize higher-return samples and improve sample efficiency.

Implementation. We implement both the diffusion actor and critic networks using multilayer perceptrons (MLPs). The actor performs iterative denoising to generate actions, while the critic estimates the corresponding target value functions [35]. The overall best-response computation for an agent during each **FP** iteration is summarized in Algorithm 4.

To compute the best response, we fix the opponent’s average policy, which is stochastic and not explicitly known to the ego agent. Rather than approximating this average via supervised learning [36], we follow the exact averaging rule from equation 4.5 by sampling from the pool of historical best-response policies, weighted by their update coefficients as seen in line 5 of Algorithm 3.

4.4 Simulation Studies

We aim to answer the following questions through extensive simulation-based experiments:

- **Convergence.** Can DiffFP learn an approximately unexploitable policy profile? Specifically, in continuous and dynamic environments, can it converge to a joint ε -Nash equilibrium?
- **Efficiency.** What is the benefit of using a DP-based approach for modeling continuous-action zero-sum games, in terms of stability and sample efficiency?
- **Robustness.** Can our method learn diverse yet robust strategies that generalize to unseen opponents?

We evaluate DiffFP on two benchmark simulation environments: one featuring symmetric agents with dense rewards, and the other characterized by asymmetric roles and sparse rewards. However, both settings are imperfect information games. All experiments are implemented in Python 3.8 and executed on a 12-core CPU with an RTX A6000 GPU.

4.4.1 Baselines

We evaluate DiffFP against several reinforcement learning (RL) baselines, adapting them as (approximate) best-response within the Fictitious Play framework (Algorithm 3).

- **QSM** [67]: A generative (score-based) actor model trained by aligning the score function with the state-action gradient field, enabling policy learning via denoising objectives.
- **SAC** [32]: A maximum entropy RL algorithm that estimates a stochastic policy by maximizing a trade-off between expected return and entropy.

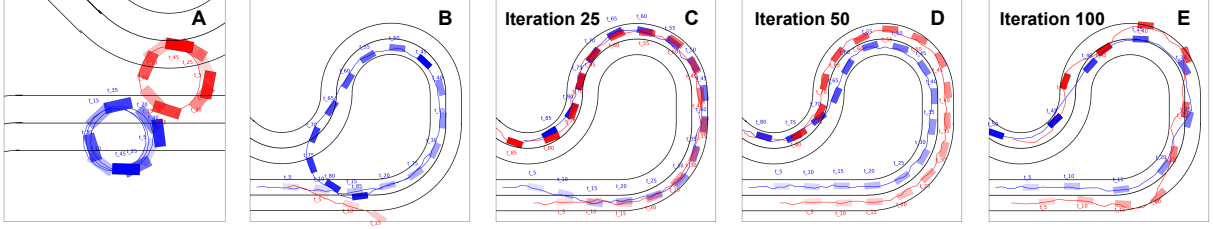


Figure 4.1: **(A–B) Failure Modes of Baselines.** Baseline agents exhibit suboptimal behaviors such as stalling or inefficient path planning. **(C–E) Training Progression of DiffFP.** Agent trajectories sampled from successive FP iterations illustrate the emergence of strategic behaviors. Over time, agents learn to navigate more efficiently, reducing their time-to-goal from 80 to 45 steps under identical initial conditions.

- **TD3** [23]: A deterministic actor-critic method that addresses overestimation bias in Q-learning through clipped double Q-learning and target policy smoothing.

Metrics. We evaluate best responses using the total exploitability of the policy, as defined in Equation 4.3. In practice, we use normalized episodic cumulative reward as the payoff measure. Additionally, we report normalized cumulative rewards for the racing task.

4.4.2 Racing Environment

Setup

We consider a multi-agent racing scenario in which agents must learn strategic behaviors such as overtaking, defending, and racing along a track. The interaction is modeled as a general-sum game [39], where each agent is rewarded for minimizing the arc-length gap to its opponent while being constrained by track limits and collision avoidance. This yields a dense reward function that provides continuous guidance throughout the episode. The introduction of the collision-avoidance penalty augments each agent’s cost to encode safety constraints, yet the fictitious play fixed point remains equivalent to the first-order optimality conditions of a Nash equilibrium in the original (unpenalized) game [75, 83]. We evaluate robustness by testing trained policies against unseen opponents across diverse track layouts.

Results

We report exploitability convergence in Figure 4.3. DiffFP consistently outperforms all baselines, exhibiting both faster and more stable convergence toward low exploitability (see Definition 3).

Limitations of Off-Policy RL. Although TD3 [23] and SAC [32] appear to converge in terms of exploitability metrics, qualitative evaluations reveal significant behavioral failures. As illustrated in Figure 4.1A and B, TD3 agents often fall into loop-like trajectories, circling without track progress, while SAC agents frequently violate track boundaries, failing to adhere to racing constraints. These behaviors highlight fundamental limitations of standard off-policy RL methods: they are typically Q-value mode-seeking and lack the expressiveness required to represent diverse and strategic behaviors [84]. Moreover, despite large adversarial populations, they tend to overfit to specific opponents, resulting in brittle policies that remain exploitable. While increasing the adversary population can help introduce more varied dynamics, it incurs diminishing returns due to reduced environment interaction per policy [79].

Instability of Q-Function. Generative approaches based on diffusion models [84] offer greater expressiveness and naturally support multimodal policy distributions, promoting behavioral diversity [86]. However, QSM suffers from high instability due to its strong reliance on accurate value function estimates. This is particularly problematic in multi-agent settings, where the opponent’s actions induce non-stationary dynamics, making Q-value estimation noisy and unreliable. As a result, exploitability convergence plots tend to be noisy and non-convergent.

Track Progress. Figure 4.3 reports the normalized track progress reward. DiffFP consistently achieves near-optimal performance, highlighting its robustness and stability in learning strategic racing behaviors.

Training Progression. Figure 4.1(C–E) illustrates agent trajectories across FP iterations. Initially (C), the blue agent trails behind. Over time (D), it learns to race competitively, overtaking on the inside and forcing the red agent to brake. Near convergence (E), both agents display aggressive and reactive behaviors: attacking, defending, and driving each other toward track boundaries, while completing the sector faster than in earlier iterations.

Attacking	Gap (m)	Successes	Crashes	Mean Reward
QSMFP	8.26	3 / 20	4 / 20	0.80
DiffFP	17.70	12 / 20	1 / 20	0.92

Table 4.1: **Head-to-Head Trials.** Gap denotes the mean relative distance gained by the attacker. Successes refer to the number of episodes where the attacker successfully closed the initial gap or overtook the target. Attacking agent is initialized behind the race leader.

Opponent	Crashes	Mean Reward
QSMFP	11 / 20	0.556
DiffFP	4 / 20	0.756

Table 4.2: **Robustness to Unseen Opponents.** Evaluation of crash rates (lower is better) and mean rewards (higher is better) when the agent is tested against up to five previously unseen adversaries.

Robustness to Unseen Opponents. Table 4.1 reports head-to-head results, where DiffFP demonstrates stronger robustness by achieving more successful overtakes and fewer collisions with the race leader compared to the QSM baseline, which exhibits higher crash rates. In scenarios with 3–5 additional agents (Table 4.2), DiffFP reliably avoids collisions and track violations through timely lane changes or braking, while baselines often exhibit unsafe behaviors.

4.4.3 Multi-Agent Particle Environment

To demonstrate the generalizability of our approach to broader multi-agent setups, we conduct experiments on the [Multi-Agent Particle Environment \(MPE\)](#) suite [58]. These tasks are particularly challenging due to their sparse reward structure and the existence of multimodal solutions. We focus on two asymmetric and imperfect information scenarios:

- **MPE-Adversary:** Ego agents must collaboratively cover landmarks to mislead a single adversary from locating the hidden target landmark (Figure 4.4B).
- **MPE-Tag:** Adversaries must coordinate to trap the ego agent, who must in turn evade capture (Figure 4.4A).

In both cases, agents operate under partial observability, receiving only local information about nearby entities. The environments are asymmetric in role and objective. We adopt

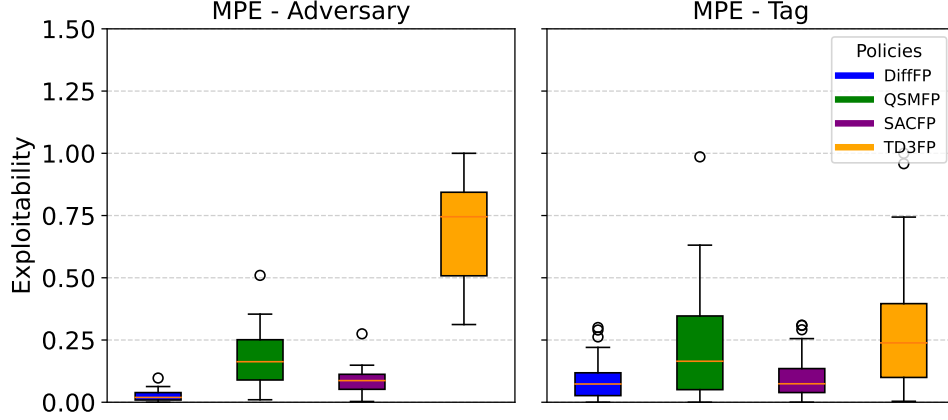


Figure 4.2: **Exploitability.** We report exploitability (see Def. 3) computed over 100 evaluation runs. Lower values indicate better robustness, with the proposed DiffFP achieving the lowest exploitability.

centralized training with decentralized execution (CTDE) framework. This setup is particularly challenging [82], as the ego agent is often required to adopt role-specific behaviors such as acting as a decoy which differ substantially from typical goal seeking strategies.

Results

We report the exploitability metric across 100 novel scenarios in Figure 4.2. DiffFP achieves the lowest exploitability with minimal variance, demonstrating strong generalization and stability. Qualitatively, only DiffFP and SACFP variants converge to near-optimal per-

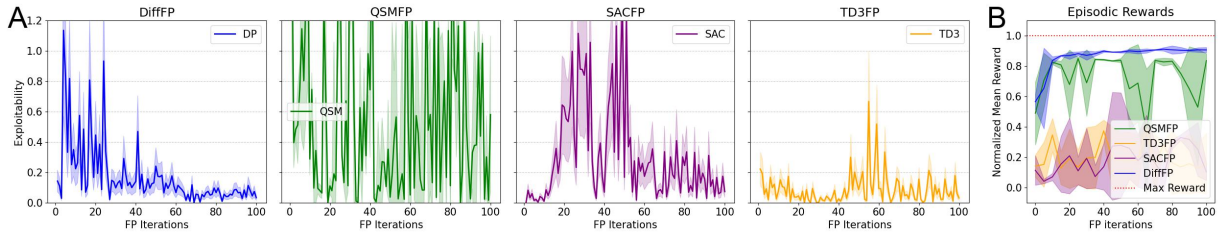


Figure 4.3: **A. Exploitability on the Racing Task.** Mean and standard deviation of exploitability over 10 episodes per FP iteration. **B. Normalized Episodic Rewards.** This metric reflects training performance and stability.

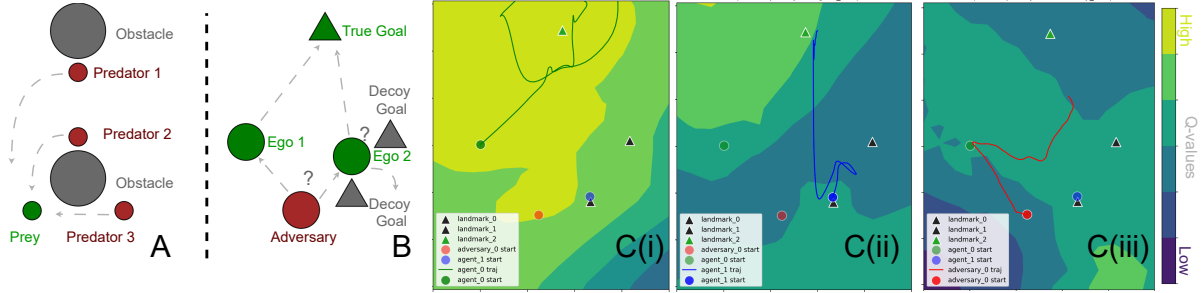


Figure 4.4: **A. MPE-Tag Env.**, **B. MPE-Adversary Env.** Dashed lines indicate possible agent trajectories. **C. Q-value Map.** At convergence to a near Nash equilibrium, ego agents exhibit coordinated behaviors with diverse roles e.g., one agent advances toward the goal while the other acts as a decoy to distract or mislead the adversaries.

formance, successfully learning cooperative behaviors. Figure 4.4 shows the Q-value map, where one ego agent acts as a decoy to distract the adversary while the other reaches the goal.

Robustness to Unseen Opponents. We evaluate robustness via cross-play matchups between all models (Tables 4.3 and 4.4). In the *MPE-Adversary* task, an ego win is recorded when the agent reaches the true goal without being caught; an adversary win occurs if it intercepts the goal-directed ego (catching a decoy still counts as failure), and a draw otherwise. In the *MPE-Tag* task, the ego wins by successfully evading all adversaries, otherwise the adversaries win. DiffFP consistently outperforms baselines when acting as the ego agent. When playing as the adversary, it improves baseline win rates and increases the number of draws, indicating strong generalization to unseen opponents.

Population-Based Hardening. In the *MPE-Adversary* task, we evaluate against SACPBT (SAC with Population-Based Training) [14], which introduces two additional adversaries preventing ego from overfitting to a single strategy. This not only results in stronger adversaries, achieving a 90% improvement over SACFP in the adversary role against DiffFP, but also increases training diversity. However, we observe that DiffFP, even without PBT, performs comparably against SACPBT, indicating its robustness to increasingly complex adversaries while avoiding the added computational cost.

Ego	Adv	Ego Wins	Adv Wins	Draws
Games where DiffFP is Ego				
DiffFP	SACFP	79	10	11
DiffFP	TD3FP	75	11	14
DiffFP	QSMFP	75	12	13
Games where DiffFP is Adversary				
SACFP	DiffFP	62 (17↓)	16 (6↑)	22 (11↑)
TD3FP	DiffFP	50 (25↓)	18 (7↑)	32 (18↑)
QSMFP	DiffFP	63 (12↓)	27 (15↑)	10 (3↓)

Table 4.3: **Robustness Against Unseen Opponents.** Head-to-head evaluation in the **MPE-Adversary** environment. Colored numbers indicate relative gains (↑) or losses (↓) when ego and adversary roles are swapped.

When Q-Value-Based Methods Excel. In the **MPE-Tag** setting, baseline performance is comparable to that of DiffFP, particularly in the case of adversaries. This is expected, as we employ a team-centralized training setup. Given the structure of the Tag game with a single ego agent and three adversaries the setting effectively resembles a population-based hardening variant of 1v1 predator-prey game, which encourages the emergence of more robust behaviors. Additionally, Q-mode-seeking models tend to perform well as adversaries; with more samples, the estimation error in the Q-value function is reduced, improving overall adversarial performance (as seen in QSMFP).

4.4.4 Discussion - Why Diffusion Policies?

In imperfect information games optimal strategies are often stochastic and multimodal. We highlight several key advantages of diffusion-based policies in this context:

- **Multimodal Best Responses:** Optimal behavior often involves stochastic mixtures over diverse actions. Traditional unimodal policies (e.g., Gaussian actors) tend to collapse to a single dominant mode, whereas diffusion policies naturally support multimodal action distributions, enabling richer and less predictable behaviors that are harder to exploit.
- **High-Entropy Stochasticity:** Diffusion models maintain high-entropy policies through sampling-based generation (equation 4.9). Unlike Gaussian policies, which

Ego	Adversary	Ego Wins	Adv Wins
DiffFP	SACFP	81	19
DiffFP	TD3FP	89	11
DiffFP	QSMFP	64	36
SACFP	DiffFP	75	25 (6↑)
TD3FP	DiffFP	66	34 (23↑)
QSMFP	DiffFP	68	32 (4↓)
QSMFP	SACFP	66	34
SACFP	QSMFP	42	58 (24↑)

Table 4.4: **Robustness Against Unseen Opponents.** Head-to-head evaluation in the **MPE-Tag** environment. Colored numbers indicate relative gains (↑) or losses (↓) when ego and adversary roles are swapped.

inject noise post hoc and often sacrifice action quality, diffusion models produce diverse yet high-quality actions directly via denoising steps.

- **Adversarial Robustness via Distributional Smoothing:** The iterative denoising process inherently regularizes against adversarial perturbations in the input space, noisy observations or adversarial behaviors leading to improved robustness under covariate shift and test-time opponent variability.

Chapter 5

Conclusion

This thesis explores the challenges and opportunities of using generative models not only to represent behaviors but also to perform *long-horizon planning* and *decision-making* in reinforcement learning domains. We focus on two settings:

- In the long-horizon tasks, we studied how to plan and reason from demonstrations using a generative model that jointly captures the structure of subgoals, actions, and task progression. This led to the development of GenPlan, a discrete energy-guided diffusion model that iteratively denoises trajectory sequences to solve long-horizon planning problems.
- In the multi-agent systems, we tackled the problem of learning policies in continuous zero-sum games from scratch. We introduced DiffFP, which frames best-response learning as a diffusion policy optimization problem within the [Fictitious Play \(FP\)](#), achieving multimodal strategy discovery and robust convergence toward equilibrium.

Across both settings, this thesis demonstrates that integrating expressive generative models into planning or fictitious play frameworks enables scalable and adaptable decision-making. Extensive evaluations validate our methods’ ability to generalize to novel tasks, stabilize training, and outperform competitive baselines in both single-agent and multi-agent domains.

This viewpoint opens several exciting research directions at the intersection of generative modeling and decision-making. One such direction is the development of amortized adaptive planners that enable test-time flexibility, allowing the planner to dynamically

adjust its optimization steps based on the current context or task difficulty. This could be especially powerful in scenarios with limited compute or time budgets, enabling planners to trade off between speed and precision on-the-fly.

Another promising avenue is planning in abstracted spaces, where generative models operate over high-level representations rather than raw state-action sequences. Such abstraction improves scalability and facilitates generalization across structurally similar but semantically distinct tasks. In robotics, for example, it enables reasoning over high-level predicates like “Will I reach a particular state?” without requiring precise temporal alignment or trajectory matching. This temporal abstraction is particularly well-suited for goal-conditioned reasoning as seen in GenPlan evaluations.

Moreover, generative models can learn structured latent dynamics that capture underlying regularities, making it possible to switch between modes of behavior or policies in environments with *hybrid dynamics* (e.g., contact-rich manipulation or mode-switching locomotion). These representations naturally support constraint-aware planning, where traditional optimization-based approaches often struggle due to non-differentiable objectives or high-dimensional discontinuities.

Together, these directions point toward a future where generative models serve not just as imitation engines, but as foundation models for decision-making—amortizing computation, enabling abstraction, and scaling to increasingly complex real-world domains [40, 18].

Limitations. While the proposed methods demonstrate strong empirical performance, there remains considerable scope for improvement. In GenPlan, the entropy constraint β , which governs action diversity, is manually tuned; a more principled, learnable alternative could improve adaptability across tasks. The framework also assumes access to near-optimal demonstrations during training. Although early results suggest some robustness to sub-optimal data, broader validation is necessary. In the multi-agent setting, DiffFP relies on short-term utility signals for reward, which may fail to capture long-horizon strategic value. Moreover, the learned diffusion policies lack explicit control over behavioral styles (e.g., aggressive vs. defensive), limiting both safety and interpretability in safety-critical or socially interactive scenarios.

Future Directions. To extend this line of work, we identify several promising directions:

- **Online Fine-Tuning.** Incorporating online fine-tuning methods, such as Hindsight Experience Replay [25], could enable GenPlan to adapt more effectively through real-world interactions. This approach bridges the gap between offline imitation and reinforcement learning while reducing sensitivity to hyperparameters.

- **Improved Learned Planning Algorithms.** Incorporating trajectory-level rewards and classical planning priors into DiffFP could enhance strategic depth, safety, and policy interpretability. While iterative denoising serves as one approach to implementing generative planning, future work could investigate tighter integration with system dynamics, structured constraint solvers, or tree-based sampling methods to improve coverage and ensure global consistency in the generated trajectories.
- **Amortized and Adaptive Planning.** Test-time planning remains computationally expensive. A promising direction is to amortize planning into a policy that executes efficiently under nominal conditions, while reverting to full planning only in novel or uncertain scenarios. Developing a robust mechanism to switch between these modes remains a key open challenge.
- **Generalization via Foundation Dynamics Models.** Traditional motion planning and control approaches often struggle to model hybrid dynamical and contact-rich systems due to non-differentiable objectives and high-dimensional discontinuities. Generative models, however, can capture combinatorial contact interactions and represent mode transitions through latent dynamics, enabling smooth and robust transitions across diverse and *a priori* unseen mode-specific behaviors.

In summary, this thesis makes a case for generative modeling not only as a representational tool, but as a substrate for decision-making itself unifying learning, reasoning, and acting within a single probabilistic framework.

Broader Impact. Iterative reasoning and self-play strategies extend naturally to other modalities, such as language and vision. For example, energy-based formulations can capture and compose semantic concepts (e.g., “smiling” + “male”) to drive image generation toward mode-specific, controllable outputs, while self-play paradigms can enhance robustness and generalization in language models by enabling them to critique and refine their own outputs.

Open Source

Open Source Implementations

Code to reproduce the results in this thesis is available at the following webpages:

- Chapter 3: <https://github.com/CL2-UWaterloo/GenPlan>
- Chapter 4: <https://github.com/CL2-UWaterloo/DiffFP>

Code References

The following libraries were used in this work: NumPy [34], PyTorch [62], JAX [3], einops [69], Gym [4], minGPT [44], and Diffusion Models in PyTorch [13, 86].

References

- [1] Suzan Ece Ada, Erhan Oztop, and Emre Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters*, 9(4):3116–3123, April 2024.
- [2] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *ICLR*, 2018.
- [3] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.
- [4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [5] George W. Brown. Iterative solution of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, New York, 1951.
- [6] Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. In *AAAI*, 2019.
- [7] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 2022.
- [8] Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *Forty-first International Conference on Machine Learning*, 2024.

- [9] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [10] Hongyi Chen, Yilun Du, Yiye Chen, Joshua B. Tenenbaum, and Patricio A. Vela. Planning with sequence models through iterative energy minimization. In *ICLR*, 2023.
- [11] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, pages 15084–15097, 2021.
- [12] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*, 2019.
- [13] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.
- [14] Pavel Czempin and Adam Gleave. Reducing exploitability with population based training. In *New Frontiers in Adversarial Machine Learning Workshop, ICML*, 2023.
- [15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.
- [16] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 34:8780–8794, 2021.
- [17] Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. In *NeurIPS*, 2024.
- [18] Yilun Du, Jiayuan Mao, and Joshua B. Tenenbaum. Learning iterative reasoning through energy diffusion. In *Forty-first International Conference on Machine Learning*, 2024.

- [19] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline RL via supervised learning? In *International Conference on Learning Representations*, 2022.
- [20] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- [21] Ruili Feng, Han Zhang, Zhantao Yang, Jie Xiao, Zhilei Shu, Zhiheng Liu, Andy Zheng, Yukun Huang, Yu Liu, and Hongyang Zhang. The matrix: Infinite-horizon world generation with real-time moving control, 2024.
- [22] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. *Conference on Robot Learning (CoRL)*, November 2021.
- [23] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *ICML*, pages 1587–1596, 2018.
- [24] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.
- [25] Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. In *International Conference on Learning Representations*, 2022.
- [26] Maxwell Goldstein and Noam Brown. Converging to unexploitable policies in continuous control adversarial games. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.
- [27] Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. Diffucoder: Understanding and improving masked diffusion models for code generation, 2025.
- [28] Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. Exposing the implicit energy networks behind masked language models via metropolis–hastings. In *International Conference on Learning Representations*, 2022.
- [29] Alex Graves. Generating sequences with recurrent neural networks, 2014.

- [30] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Proceedings of the Conference on Robot Learning*, Proceedings of Machine Learning Research, pages 1025–1037. PMLR, 30 Oct–01 Nov 2020.
- [31] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.
- [32] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.
- [33] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *National Conference on Artificial Intelligence*, 2004.
- [34] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.
- [35] Hado Hasselt. Double q-learning. *NeurIPS*, 23, 2010.
- [36] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *ICML*, pages 805–813. PMLR, 2015.
- [37] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games, 2016.
- [38] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020.
- [39] Junling Hu and Michael P. Wellman. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, page 1039–1069, December 2003.
- [40] Michael Janner. *Deep Generative Models for Decision-Making and Control*. Ph.d. dissertation, University of California, Berkeley, 2023.

- [41] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.
- [42] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in neural information processing systems*, volume 34, pages 1273–1286, 2021.
- [43] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *NeurIPS*, 36, 2024.
- [44] Andrej Karpathy. minGPT: A minimal pytorch re-implementation of the openai gpt training, 2020.
- [45] Akash Karthikeyan and Yash Vardhan Pant. DiffFP: Learning Behaviors from Scratch via Diffusion-based Fictitious Play. *Initial results presented at the IJCAI 2025 Workshop on User-Aligned Assessment of Adaptive AI Systems. Full version under review.*
- [46] Akash Karthikeyan and Yash Vardhan Pant. Adaptformer: Sequence Models as Adaptive Iterative Planners. *Proceedings of the Conference on Robots and Vision*, May 2025.
- [47] Akash Karthikeyan and Yash Vardhan Pant. GenPlan: Generative Sequence Models as Adaptive Planners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(17):17797–17804, Apr. 2025.
- [48] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.
- [49] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.
- [50] Sampo Kuutti, Saber Fallah, and Richard Bowden. Arc: Adversarially robust control policies for autonomous vehicles. In *ITSC*, pages 522–529, 2021.
- [51] Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating compounding prediction errors in learned dynamics models, 2022.
- [52] Marc Lanctot, Vinicius Zambaldi, Audrūnas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *NeurIPS*, page 4193–4206, 2017.

- [53] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 26991–27008. PMLR, 21–27 Jul 2024.
- [54] David S Leslie and Edmund J Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.
- [55] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.
- [56] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.
- [57] Siqi Liu, Luke Marris, Daniel Hennes, Josh Merel, Nicolas Heess, and Thore Graepel. NeuPL: Neural population learning. In *International Conference on Learning Representations*, 2022.
- [58] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *NeurIPS*, 2017.
- [59] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.
- [60] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.
- [61] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.
- [62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. 2019.

- [63] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.
- [64] Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, Mark Rowland, Pedro Ortega, Neil Burch, Thomas Anthony, David Balduzzi, Bart De Vylder, Georgios Piliouras, Marc Lanctot, and Karl Tuyls. From poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization. In *ICML*, 2021.
- [65] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, 2007.
- [66] Dean A. Pomerleau. Alvin: an autonomous land vehicle in a neural network. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, NIPS’88, page 305–313, Cambridge, MA, USA, 1988. MIT Press.
- [67] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *ICML*, 2024.
- [68] Julia Jean Robinson. An iterative method of solving a game. *Classics in Game Theory*, 1951.
- [69] Alex Rogozhnikov. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*, 2022.
- [70] Juergen Schmidhuber. Reinforcement learning upside down: Don’t predict rewards – just map them to actions, 2020.
- [71] Thomas Schmied, Markus Hofmarcher, Fabian Paischer, Razvan Pascanu, and Sepp Hochreiter. Learning to modulate pre-trained models in RL. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [72] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.
- [73] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharmashan Kumaran, Thore Graepel, et al.

- A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.
- [74] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.
 - [75] Riccardo Spica, Davide Falanga, Emiliano Cristofalo, Eduardo Montijano, Davide Scaramuzza, and Mac Schwager. A real-time game theoretic planner for autonomous two-player drone racing. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.
 - [76] Jiankai Sun, De-An Huang, Bo Lu, Yun-Hui Liu, Bolei Zhou, and Animesh Garg. Plate: Visually-grounded planning with transformers in procedural tasks. *IEEE Robotics and Automation Letters*, 7(2):4924–4930, 2022.
 - [77] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.
 - [78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
 - [79] Eugene Vinitzky, Yuqing Du, Kanaad Parvate, Kathy Jang, Pieter Abbeel, and Alexandre Bayen. Robust reinforcement learning using adversarial populations, 2020.
 - [80] Oriol Vinyals, Igor Babuschkin, Wojciech Marian Czarnecki, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019.
 - [81] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.
 - [82] Jiangxing Wang, Deheng Ye, and Zongqing Lu. More centralized training, still decentralized execution: Multi-agent conditional policy factorization. In *The Eleventh International Conference on Learning Representations*, 2023.
 - [83] Mingyu Wang, Zijian Wang, John Talbot, J. Christian Gerdes, and Mac Schwager. Game-theoretic planning for self-driving cars in multivehicle competitive scenarios. *IEEE Transactions on Robotics*, 37(4):1313–1325, 2021.

- [84] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *ICLR*, 2022.
- [85] Michael P. Wellman. Methods for empirical game-theoretic analysis. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI’06, page 1552–1555. AAAI Press, 2006.
- [86] Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning, 2023.
- [87] Shuo Yang, Hongrui Zheng, Cristian-Ioan Vasile, George Pappas, and Rahul Mangharam. Stlgame: Signal temporal logic games in adversarial multi-agent systems. In *Proceedings of the 7th Annual Learning for Dynamics & Control Conference*, pages 1102–1114, 2025.
- [88] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: model-based offline policy optimization. In *NeurIPS*, Red Hook, NY, USA, 2020. Curran Associates Inc.
- [89] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *International Conference on Machine Learning*. PMLR, 2022.
- [90] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *NeurIPS*, 2007.

APPENDICES

Appendix A

Planning with Discrete Diffusion - Details

Organization of Appendix. The Appendix is organized as follows: Appendix A.1 provides implementation details for the BabyAI environment. This is followed by additional results and discussions in Appendix A.2. Details of the continuous tasks are outlined in Appendix A.3, while Appendix A.4 includes further experiments on zero-shot adaptation to different tasks in MetaWorld. Finally, Appendix A.5 briefly discusses sampling and the choice of the rate matrix in Algorithm 2. This appendix expands upon the supplementary material from our work presented at the *AAAI Conference on Artificial Intelligence* [47].

A.1 BabyAI Implementation Details

A.1.1 Input and Networks

The various inputs and their corresponding dimensions are presented in Table A.1 for the BabyAI environments. We construct GenPlan following the transformer architecture (with bi-directional masks) described in [13]. The inputs to the model include token embeddings for observations, states, and actions, which are generated through specific mappings. Detailed illustrations of these mappings can be found in Figures 3.2 and A.4.

$$FiLM(I_{\text{ctx}}, s_{\text{ctx}}, \xi, t, k) \rightarrow \text{tok}_o \quad f_{\theta_s}(s_k^t) \rightarrow \text{tok}_s \quad f_{\theta_a}(a_k^t) \rightarrow \text{tok}_a$$

The observation token tok_o is obtained from the panoramic image I , state s_{ctx} , instruction ξ , and timestep t in the [Continuous-time Markov chain \(CTMC\)](#). The FiLM (Feature-wise

Linear Modulation) function [63] transforms these inputs into an observation embedding. Note that, in the case of a masked or corrupt state, these observation tokens are set to a synthetic mask state during training. We follow the same instruction encoder as used in the BabyAI agent model [12]. The context length, ctx (see Table A.3), acts as a memory for the agent and provides past trajectories of length ctx . This embedding, along with the corrupted state $\mathbf{s}_k^t$ and action $\mathbf{a}_k^t$ tokens generated by the embedding networks f_{θ_s} and f_{θ_a} , is fed into the transformer’s decoder stack. The transformer’s cross-attention mechanism is employed in this joint denoising process. The decoder then predicts the true labels $\mathbf{s}^1$, $\mathbf{g}^1$, and $\mathbf{a}^1$, with the observation tokens guiding the denoising process (Figure 3.2). Consequently, we have $H + 1$ tokens in the cross-attention module, where the $+1$ corresponds to the CTMC timestep-based embedding, and $2H$ tokens serve as input to the transformer’s decoder block. Further details on the input dimensions and the architecture are provided in Table A.2.

Input	Vector Space	Description
Image I	$\mathbb{R}^{S \times S \times 3}$	Symbolic representation of env. S (size of env.)
State $\mathbf{s}_k^t$	$\mathbb{R}^3$	Tuple corresponding to $\langle x, y, \text{dir} \rangle$
Goal $\mathbf{g}_k^t$	$\mathbb{R}^{\Pi \times 2}$	No. of goals (Π), each with x, y positions
Instructions ξ	—	Task-specific instructions
Actions $\mathbf{a}_k^t$	$\mathbb{R}^6$	Discrete set of actions available

Table A.1: **Offline data.** All of these components are mapped to a latent embedding, as shown in Figure 3.2B.

A.1.2 Environments

Detailed settings for the BabyAI environments are provided in Table A.4. We evaluate across three task paradigms as described in Section 3.4.1, with tasks in **Instruction Completion (IC)** and **Adaptive Planning (AP)** being particularly challenging. For instance, in the **BlockUnlock** task, the agent must first remove an obstacle by picking it up and repositioning it, then retrieve a key to unlock the door, and finally navigate to the goal, which is always located in the locked room. Due to the order-critical nature of these tasks, we observe that only GenPlan succeeds, while other baselines fail. Similarly, in **AP** tasks, the agent must generalize to more complex environments than those encountered during training. For further details, refer to Figures A.1, A.2, and A.3.

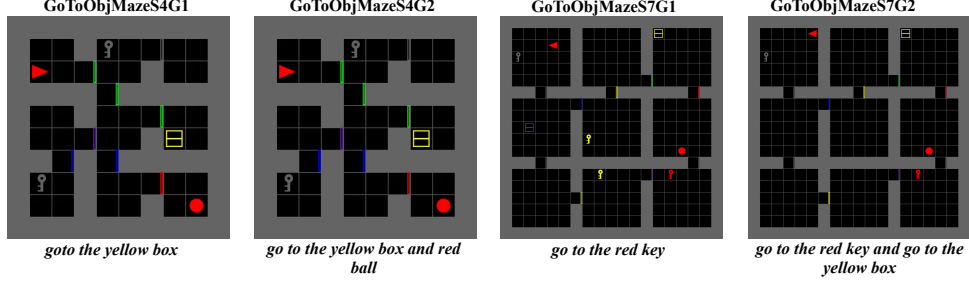


Figure A.1: **Trajectory Planning (TP)**. The agent is randomly initialized and must navigate through a maze-like environment to reach the goal(s). In each evaluation, the map layout, the agent’s initial position, and goal positions are varied. Note that in the case of GenPlan, the agent does not have access to the ground truth goal positions, whereas other baselines do.

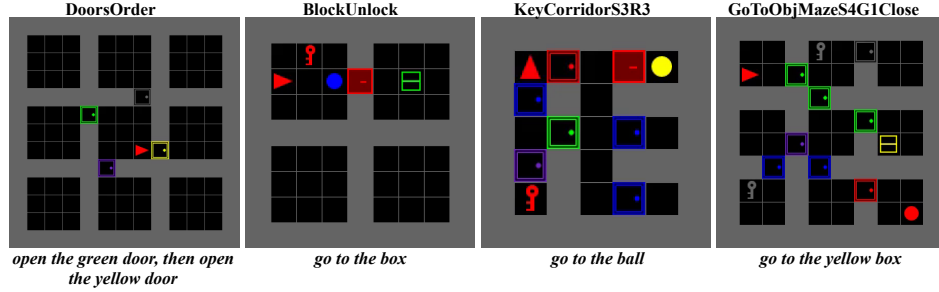


Figure A.2: **Instruction Completion (IC)**. These tasks are more complex than [Trajectory Planning \(TP\)](#), requiring the agent to perform additional actions such as *pickup*, *drop*, and *open* in addition to navigation. For example, in *DoorsOrder*, the agent must open the specified doors in the correct order without accidentally entering other rooms.

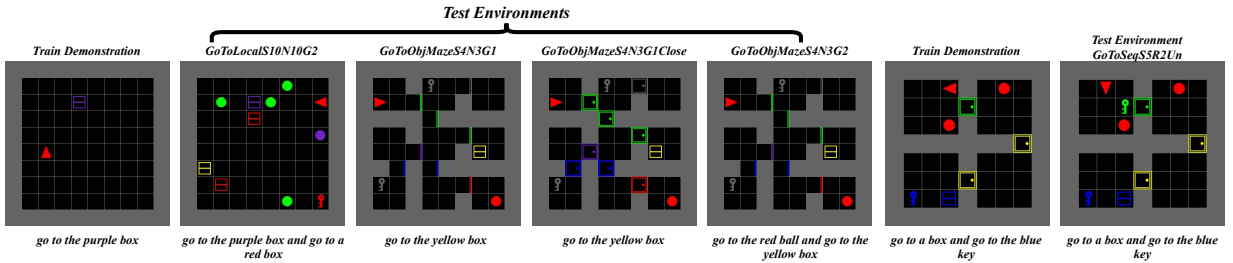


Figure A.3: **Adaptive Planning (AP)**. The model is trained on simpler tasks and tested for adaptability by progressively increasing task difficulty. In *GotoObjMazeClose* and *GoToSeqUn*, we evaluate whether the model can demonstrate novel skills, such as unblocking obstacles and opening doors.

Hyperparameter	Value
Number of layers	4
Number of attention heads	4
Embedding dimension	128
Batch size	64
Image Encoder	nn.Conv2d
Image Encoder channels	128, 128
Image Encoder filter sizes	$2 \times 2, 3 \times 3$
Image Encoder maxpool strides	2, 2 (Image Encoder may vary a little depending on the environment size)
Instruction Encoder	nn.GRU
Instruction Encoder channels	128
State Encoder	nn.Linear
State Encoder channels	128, 128, 128
Max epochs	400
Dropout	0.1
Learning rate	$8 * 10^{-4}$
Adam betas	(0.9, 0.95)
Grad norm clip	1.0
Weight decay	0.1
Learning rate decay	Linear warmup and cosine decay (see code for details)

Table A.2: Hyperparameters of GenPlan for BabyAI experiments.

A.1.3 Baseline Models

We use the official implementations of LEAP [10] and [Decision Transformer \(DT\)](#) [11]. To enable these methods to handle multi-task missions which otherwise fail to generalize to novel goals (e.g., repeatedly predicting corner cells) we augment each baseline with simulator-based goal-conditioning. Specifically, the BabyAI simulator provides both the final goal and intermediate sub-goals for goal-conditioned planning. All models are then trained end-to-end.

Env	Context	Plan	GenPlan		LEAP
			Iteration	Entropy	Iteration
GoToObjMazeS4G1	1	10	5	0.3	10
GoToObjMazeS4G2	1	10	5	0.3	10
GoToObjMazeS7G1	1	20	10	0.3	50
GoToObjMazeS7G2	1	20	10	0.3	50
GoToObjMazeClose	1	10	5	0.3	10
DoorsOrder	1	10	5	0.3	10
BlockUnlock	10	30	15	0.3	50
KeyCorridorS3R3	10	30	15	0.3	50
GoToLocalS10N10G2	1	10	5	0.7	10
GoToObjMazeS4N3G1	1	10	5	0.7	10
GoToObjMazeClose	1	10	5	0.7	10
GoToObjMazeS4G2	1	10	5	0.7	10
GoToSeqS5R2Un	1	20	10	0.7	40

Table A.3: **Training Configurations.** For most tasks, planning is based solely on the current state observation and aims to plan for H steps. However, for **KeyCorridor** and **BlockUnlock**, which involve complex sub-tasks and require memory to track completed tasks, context is used.

A.2 Additional Results and Discussion

We present additional experiments to further evaluate GenPlan’s capabilities across diverse scenarios. These include analysis of convergence rates and entropy, as well as performance in stochastic environments and with sub-optimal data. Together, these studies offer deeper insights into GenPlan’s generalizability, robustness, and adaptability to complex tasks.

Rate of Convergence. With the masking interpolant, the model easily identifies which indices are corrupted, simplifying training and yielding faster convergence. We report the correct action rate (i.e., the proportion of the corrupted trajectory that the model can recover) and observe that masked denoising converges much faster (see Figure A.5A). Consequently, the masked tokens receive the lowest attention scores. In contrast, with the uniform interpolant it is harder for the model to determine which tokens are corrupted, leading to slower convergence.

Entropy. Continuing our discussion in Section 3.4.1, we report success rates as a function of the entropy lower bound (see Figure A.5B). For **TP** and **IC** tasks, entropy levels above 0.3 offer minimal benefit. In contrast, for **AP** tasks we set the entropy bound to 0.7, which promotes task discovery (i.e., performing novel actions absent from the dataset). Notably, GenPlan leverages this higher entropy to introduce novel actions at high-energy token indices, enhancing its planning capability.

Task	Env	Size	# Room	# Obs	Door	Unblocking	Max Steps
Task Planning (TP)	GoToObjMazeS4G1	10 × 10	9	4	Open	No	399
	GoToObjMazeS4G2	10 × 10	9	4	Open	No	499
	GoToObjMazeS7G1	19 × 19	9	7	Open	No	699
	GoToObjMazeS7G2	19 × 19	9	7	Open	No	999
Instruction Completion (IC)	GoToObjMazeClose	10 × 10	9	4	Closed	No	499
	DoorsOrder	13 × 13	9	0	Closed	No	144
	BlockUnlock	8 × 8	2	2	Closed	Yes	399
	KeyCorridorS3R3	7 × 7	6	2	Closed	No	499
Adaptive Planning (AP)	GoToLocalS10N10G2	10 × 10	10	4	Open	No	299
	GoToObjMazeS4N3G1	10 × 10	9	4	Open	No	499
	GoToObjMazeClose	10 × 10	9	4	Closed	No	499
	GoToSeqS5R2Un	9 × 9	4	5	Closed	Yes	499

Table A.4: BabyAI environment setting details.

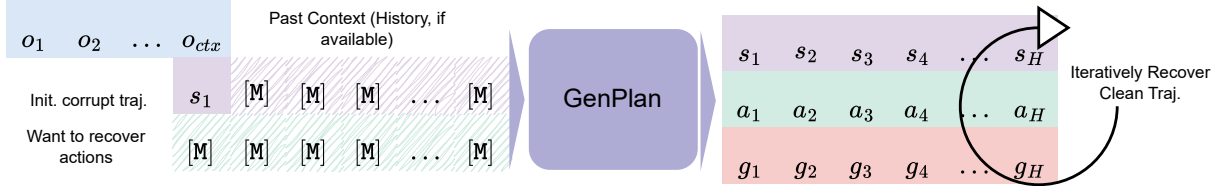


Figure A.4: **GenPlan I/O**. During planning, the model takes in a corrupted trajectory along with the current state and past observations (if available), and iteratively recovers the clean trajectory.

Stochasticity in Environments. In this setup, the agent operates in an environment where there is a 20% chance that its chosen action (e.g., “left” or “right”) will be randomly remapped to any other action with uniform probability. To handle this stochasticity, we replan after each step using single-step rollouts, unlike the multi-step rollouts used in Tables 3.1 and 3.2. We apply the same replanning strategy to all baseline models for a fair comparison.

The results in Table A.5 show that GenPlan performs on par with baseline methods in stochastic environments, where both use feedback at each step. However, in adaptive environments, even with feedback, the baseline model (DT) fails, while GenPlan and LEAP succeed.

Context Length. We use context length as a memory mechanism for tracking completed sub-goals, which is primarily beneficial in tasks like **KeyCorridorS3R3** and **BlockUnlock** (Table 3.1). While most TP tasks and multi-objective IC tasks operate under a Markovian assumption, these tasks involve complex sub-tasks requiring memory to track progress. Context-based planning is also enabled for baselines (see Figure A.4)

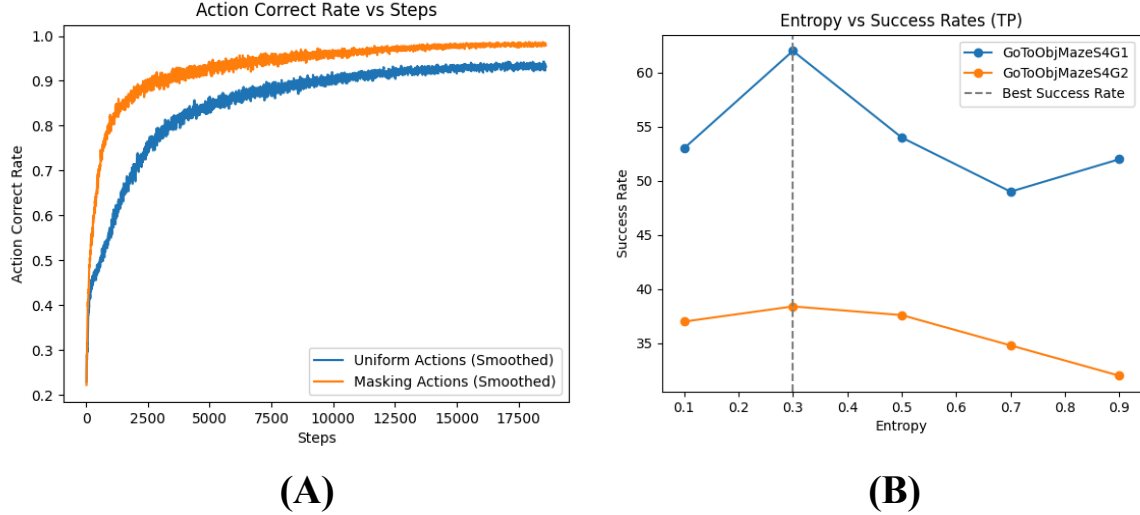


Figure A.5: **GenPlan Configurations.** (A) Training progression of GenPlan, showing that GenPlan-M converges faster than GenPlan-U. (B) Success rate as a function of entropy.

Env	GenPlan-M	LEAP	DT
GoToObjMazeS7G1 (TP)	48%	18%	52%
GoToObjMazeS4N3G1 (AP)	54.8%	52%	23.2%

Table A.5: **Stochastic Environments.** Success rates of models in task planning and adaptive environments using single-step rollouts to account for stochasticity.

Time considerations. We observe that due to the [Masked Language Model \(MLM\)](#) objective in LEAP, they must sequentially mask each action, which is time-consuming. Additionally, gradients are updated only after accumulating losses from all action tokens in the horizon, making the process memory-intensive. In contrast, GenPlan controls the number of masked tokens using the [CTMC](#) timestep t , eliminating the need for sequential masking and saving significant time. Although GenPlan is slightly slower than [DT](#) due to its iterative inference process and the bi-directional mask loss taking longer than the causal masks in [DT](#), it still performs within a reasonable time frame. The total training and inference times are reported in [Table A.6](#).

Mode	Env	GenPlan-M	LEAP	DT
Train	GoToObjMazeS4G2	34m	5hrs	30min
	GoToObjMazeS7G2	1hr 3min	6hrs	57min
Eval	GoToObjMazeS4	3mins	10min	2min
	GoToObjMazeS7	18min	36min	10min

Table A.6: **Time Considerations.** Comparison of training and evaluation times across different environments and methods. Training is performed with a batch size of 64 over 500 epochs, while evaluation is conducted across 250 environments.

Instructions Ablation. We observe that image inputs play a significant role in learning goal distributions. The training data comprises standalone examples of single goals or skills, while testing involves skill chaining based on prompts in a fully observable environment. GenPlan addresses these challenging tasks by learning energy landscapes and employing [Discrete Flow Model \(DFM\)](#) sampling.

Prompts are essential in instruction-completion tasks. Procedurally generated by MiniGrid environments, these prompts are available during both training and testing phases. To evaluate their significance, we ablate the instruction embeddings, which are key to instruction completion as seen in Table A.7. While the primary focus of this work is to adopt discrete representations for planning with sub-tasks, language grounding is left for future exploration.

Environment	Success Rate	Reduction
Doors Order (IC)	22%	13.2% ↓
GoToLocal10G2 (AP)	73%	15% ↓

Table A.7: **Instruction Prompt Ablation.** We ablate the instruction embeddings from the input to assess their significance, as evidenced by the reduction in success rates across environments.

Suboptimal Data. We evaluate our model’s performance when trained on suboptimal data and observe that GenPlan-M retains its generalization capability, whereas [DT](#) fails to adapt effectively. We hypothesize that GenPlan’s iterative energy-minimization objective enables it to compose suboptimal trajectories, thereby recovering longer-horizon behaviors from sub-optimal trajectories.

Environment	Optimal Data		Suboptimal Data (25%)	
	GenPlan-M	DT	GenPlan-M	DT
GoToObjMazeS4G1	62%	44%	55.2%	44%
GoToObjMazeS7G1	44.8%	40%	38.8%	15.2%
BlockUnlock	16%	0%	8%	0%

Table A.8: **Performance with Suboptimal Trajectories.** We introduce 25% random actions into the training data to simulate suboptimal trajectories. Despite this, GenPlan-M demonstrates robust generalization, outperforming DT across all environments.

A.3 Continuous Tasks Implementation Details

Simulation Setup

(i) **PushT** [22]: The task is to push a T-shaped block to a target pose (position and orientation) using an end-effector controlled by 2D velocity commands. The dataset comprises 206 human demonstrations. Evaluation metrics include final and maximum coverage, measured by the Intersection over Union (IoU) between the T-block and the target T position.

(ii) **Franka Panda Manipulation** [30]: A robotic manipulation task using a Franka Panda arm with a 7D action space, featuring 566 human demonstrations. The dataset comprises seven subtasks, and each trajectory executes four subtasks in a variable order. At test time, we measure performance by the number of subtasks completed during an unconditional rollout. Evaluations are performed in both joint-space and state-space variants.

We compare our approach with VQ-BeT [53], which employs a categorical transformer-based model to infer action codes, subsequently decoded using a pre-trained motion primitive model (VQ-VAE). In our experiments, we use the same pre-trained motion primitive but replace the auto-regressive transformer with GenPlan’s objective and sampling process.

A.4 MetaWorld Implementation Details.

We assess the adaptive capabilities of the agents in MetaWorld’s unseen tasks.

Category	Metric	MDDT	GenPlan-M
In-Dist.	SR	0.88 $\pm$ 0.032	0.786 $\pm$ 0.104
	MR	1566 $\pm$ 38.0	1465.66 $\pm$ 34.79
Out-Dist.	SR	0.3 $\pm$ 0.081	0.333 $\pm$ 0.047
	MR	617 $\pm$ 52.46	620.33 $\pm$ 48.58

Table A.9: **Quantitative Performance.** Comparison between MDDT and GenPlan. Success rates (SR) and mean rewards (MR) are reported across three unique seeds.

Simulation Setup

Our simulation setup closely follows that of [71], benchmarking GenPlan on MetaWorld tasks that encompass various robotic manipulation challenges. Both MDDT [71] and GenPlan are trained on five tasks (In-Dist.) and evaluated on nine tasks, including four new tasks (Out-Dist.) designed to test adaptation. Consistent with the BabyAI experiments, we directly evaluate zero-shot performance. Each episode consists of 200 timesteps per task, and we report success rates and mean rewards across Meta-World experiments.

To enable multi-domain learning and task adaptation, the action space is modeled as a categorical distribution with actions discretized into 64 bins using minmax tokenization, as in [71]. In MetaWorld, the state and action spaces are consistent across environments ($|S| = 39$, $|A| = 4$). The dataset comprises 10K trajectories per task (length 200) collected by task-specific **Soft Actor Critic (SAC)** agents, featuring behavior ranging from random to expert-level. Unlike the near-optimal demonstrations in BabyAI, this dataset offers broader variability. Sequential action prediction with a planning horizon of 1 is employed, which provides feedback at each step and improves adaptive performance. The environments are detailed in Table A.10.

Rollout. During evaluation, actions are sequentially predicted token by token to fill the 4-dimensional action space ($|A| = 4$). After completing the action, the corresponding observation is obtained (plan horizon of 1).

Baselines and Results

We follow the pre-processing steps from MDDT [71], with GenPlan utilizing a flow-based generative loss instead of the auto-regressive loss, as described in Section 3.3. During

Task	$\mathcal{S}$	$\mathcal{A}$	Dataset		Evaluation - SR	
			SR	Reward	GenPlan	MDDT
In Dist. Environments						
button-press-v2	39	4	1.0	1430.44	0.73±0.11	0.8±0.2
dial-turn-v2	39	4	0.8	1674.29	0.86±0.23	0.93±0.11
disassemble-v2	39	4	1.0	1396.55	0.66±0	0.8±0.2
plate-slide-v2	39	4	1.0	1667.35	1±0	1±0
reach-v2	39	4	1.0	1858.99	0.73±0.11	0.933±0.11
Adaptive Environments (Out Dist.)						
reach-wall-v2	39	4	1.0	1831.14	0.86±0.11	0.73 ±0.11
plate-slide-side-v2	39	4	1.0	1663.35	0±0	0±0
button-press-wall-v2	39	4	1.0	1508.16	0.46±0.11	0.4±0.2
coffee-button-v2	39	4	1.0	1499.17	0.13±0.067	0.2±0.2

Table A.10: **Dataset and Performance Scores.** The agent is trained only on datasets from the 5 in-distribution environments and evaluated on all nine. Datasets from adaptive environments are not used.

sampling, return-conditioned sampling [11] is used. While MDDT performs well on in-distribution tasks, GenPlan demonstrates superior generalization to closely related tasks (e.g., *reach* and *button* tasks) and adapts effectively to variations in orientation or axis without fine-tuning.

The masking variant of GenPlan benefits from a robust training process and multi-modality by predicting alternate actions at corrupt timesteps in the trajectory. As a result, GenPlan achieves slightly better scores in adaptive tasks. For details on datasets and environments, refer to Appendix A.4. Success rates over 3 seeds are summarized in Table A.10.

A.5 Rate Matrices and Sampling

Once the denoising model $p_{1|t}^\theta$ is trained, we can select a rate matrix to simulate the CTMC for generating samples (see Algorithm 2). [8] proposes the following rate matrix definition as an initial choice (for simplicity, consider the case where $H = 1$):

$$R_t^*(x_k^t, j_k | x_k^1) := \frac{\text{ReLU}(\partial_t p_{t|1}(j_k | x_k^1) - \partial_t p_{t|1}(x_k^t | x_k^1))}{\mathcal{Z}_{t|1}(x_k^t | x_k^1)},$$

t	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}
0.0	6	6	6	6	6	6	6	6	6	6
0.2	1	6	6	0	6	5	6	6	6	6
0.4	1	6	6	0	6	5	6	6	2	6
0.6	1	5	6	0	6	5	6	6	2	6
0.8	1	5	0	0	2	5	1	1	2	2
1.0	1	5	0	0	2	5	1	1	2	2

Table A.11: **Masking Interpolant.** The sample is initialized with the mask state (represented by 6) and iteratively denoised to obtain the clean sequence (in green).

t	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}
0.0	3	5	4	3	2	4	1	2	0	2
0.2	3	5	4	0	2	4	1	2	0	2
0.4	3	5	4	0	2	4	1	2	2	2
0.6	1	5	0	0	2	4	1	1	2	2
0.8	1	5	0	0	2	5	1	1	2	2
1.0	1	5	0	0	2	5	1	1	2	2

Table A.12: **Uniform Interpolant.** The sample is initialized with uniform noise (in red) and iteratively denoised to obtain the clean trajectory (in green).

where $\text{ReLU}(a) = \max(a, 0)$ and $\mathcal{Z}_t$ is the number of states with non-zero mass at t , defined as $\mathcal{Z}_t = |\{x_k^t : p_{t|1}(x_k^t | x_k^1) > 0\}|$. Note that $R_t^*(x_k^t, j_k | x_k^1) = 0$ when $p_{t|1}(x_k^t | x_k^1) = 0$ or $p_{t|1}(j_k | x_k^1) = 0$. Additionally, when $x_k^t = j_k$, we have $R_t^*(x_k^t, x_k^t | x_k^1) = -\sum_{j_k \neq x_k^t} R_t^*(x_k^t, j_k | x_k^1)$.

To ensure that the rate matrix R_t^* is well-defined and maintains the flow of probability mass between states, we assume that $p_{t|1}(x_k^t | x_k^1) > 0$. This assumption is crucial because it ensures that the state x_k^t is “alive” and carries some probability mass at time t . If $p_{t|1}(x_k^t | x_k^1) = 0$, then $R_t^*(x_k^t, j_k | x_k^1)$ would either be undefined or forced to be zero, as the state would be considered “dead,” unable to either receive or provide any mass. Without this assumption, a state could demand incoming mass without being able to reciprocate, violating the consistency of the conditional flow process and the Kolmogorov equation.