GenPlan: Generative Sequence Models as Adaptive Planners

Akash Karthikeyan, Yash Vardhan Pant

University of Waterloo, Canada
{a9karthi, yash.pant}@uwaterloo.ca

Abstract

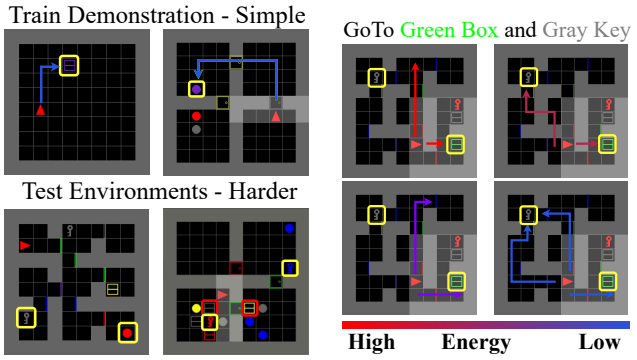
Sequence models have demonstrated remarkable success in behavioral planning by leveraging previously collected demonstrations. However, solving multi-task missions remains a significant challenge, particularly when the planner must adapt to unseen constraints and tasks, such as discovering goals and unlocking doors. Such behavioral planning problems are challenging to solve due to: a) agents failing to adapt beyond the single task learned through their reward function, and b) inability to generalize to new environments, e.g., those with walls and locked doors, when trained only in planar environments. Consequently, state-of-the-art decision-making methods are limited to missions where the required tasks are well-represented in the training demonstrations and can be solved within a short (temporal) planning horizon. To address this, we propose GenPlan: a stochastic and adaptive planner that leverages discrete-flow models for generative sequence modeling, enabling sample-efficient exploration and exploitation. This framework relies on an iterative denoising procedure to generate a sequence of goals and actions. This approach captures multi-modal action distributions and facilitates goal and task discovery, thereby generalizing to out-of-distribution tasks and environments, i.e., missions not part of the training data. We demonstrate the effectiveness of our method through multiple simulation environments. Notably, GenPlan outperforms state-of-the-art methods by over 10% on adaptive planning tasks, where the agent adapts to multi-task missions while leveraging demonstrations from single-goal-reaching tasks.

Code — https://github.com/CL2-UWaterloo/GenPlan

1 Introduction

An intelligent autonomous agent must be adaptable to new tasks at runtime beyond those encountered during training. 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 1A. 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.*



(a) *Adaptation to harder tasks.* (b) *Iterative refinement.*

Figure 1: *Overview.* GenPlan is a generative, multi-step planner that optimizes energy landscape to adapt to complex tasks and iteratively refine long-horizon missions. Goals are highlighted in yellow, and distractors are marked in red.

Learning such behaviors from demonstrations is challenging and often requires a 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) (Emmons et al. 2022) simplifies traditional Reinforcement Learning (RL) by using a behaviour cloning (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 (Lambert, Pister, and Calandra 2022).

Thus, a categorical distribution is better suited for capturing abstracted sequence planners (Lee et al. 2024), as it enables clustering of different modes and provides interpretable predictions of subtask orders. More recently, generative modeling-based works allow conditional sequence-

level predictions (Janner et al. 2022; Chi et al. 2023). 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 (discrete) see Karthikeyan and Pant, Appendix C. They often get stuck in local optima when handling unseen cases, as they lack distributional fidelity and struggle to adapt to new tasks.

Contributions of this work. To address these challenges, we propose GenPlan, a discrete flow-based framework for sequence modeling that utilizes a Continuous Time Markov Chain (CTMC)-based sampling method. Our key contributions are as follows:

- GenPlan frames planning as the iterative denoising of trajectories using discrete flow models. This approach enables adaptability by allowing goal and task discovery.
- Learn an energy function to guide denoising and minimize it to generate action sequences.
- GenPlan uses an entropy-based lower bound over action probabilities to encourage adaptability and generalization to previously unseen tasks and environments.

Thus, instead of greedily selecting the next action, GenPlan jointly learns goal and action distributions with bidirectional context. This approach prevents the agent from stalling and getting stuck in local regions. Additionally, framing planning as generative modeling, GenPlan allows for generalization to novel environments, as long as the objective function remains consistent (i.e., assigning minimal energy to successful trajectories). Through extensive simulations, we empirically demonstrate that GenPlan outperforms state-of-the-art models by over 10% in adaptive planning, particularly in challenging environments with multiple sub-goals, while only leveraging demonstrations from single-goal-reaching tasks.

2 Related Works

Offline RL. Offline RL focuses on learning policies from collected demonstrations, as illustrated in figure 2A, without further interaction with the environment (Levine et al. 2020). A key challenge in this approach is the *distribution shift* between the training demonstrations and the runtime distribution. Several regularization strategies have been proposed to address this, such as reducing the discrepancy between the learned and behavioral policies (Fujimoto, Meger, and Precup 2019; Kumar et al. 2019). 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 (Chen et al. 2021; Janner, Li, and Levine 2021; Furuta, Matsuo, and Gu 2022) adopt an autoregressive modeling objective, 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, they fail in scenarios like Example 1 due to deadlocks. Moreover, they struggle in unconditional rollouts, lacking both guidance and the ability to generalize to novel goals.

Planning with Sequence Models. Planning Transformer (Sun et al. 2022) introduces procedural planning with transformers by framing planning as a model-based RL problem. It employs a joint action and state representation model to mitigate compounding errors inherent in transformer-based architectures. LEAP (Chen et al. 2023) 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 prevents error accumulation and demonstrates 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 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 (Haarnoja et al. 2017; Eysenbach et al. 2022) and diffusion models (Janner et al. 2022; Chi et al. 2023) have been applied to offline RL tasks. While diffusion-based models often require well-represented datasets, they fail to generalize to unseen harder tasks, as shown in figure 1A. In contrast, energy-based models can learn an energy landscape that helps generalize to unseen tasks; however, they are challenging to sample from, requiring cross-entropy or MCMC sampling (Chen et al. 2023), and are often unstable to train (Chi et al. 2023). Recently, energy-diffusion(Du, Mao, and Tenenbaum 2024) proposes guided 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.

3 Preliminaries and Problem Statement

Notations. We model discrete data as a sequence $(x_1, \ldots, x_H)$, where H is the horizon. This sequence is represented by $\boldsymbol{x}$, 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 $X = \{1, \ldots, x_{|X|}\}$, 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 $G = \{g_1, \ldots, g_n\}$ within the state space S. Formally, we consider learning in a MDP $\mathcal{M} = \langle S, A, P, R \rangle$. The MDP tuple comprises S and A to denote the state and action spaces, respectively, with discrete states $s_k \in S$ and actions $a_k \in A$. The transition function $P(\cdot|s_k, a_k)$ returns a probability distribution over the next states given a state-action pair. The reward function 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.

RL as sequence modeling. Decision Transformer (DT) (Chen et al. 2021) frames RL as a sequence modeling problem, allowing training through upside-down RL (Schmidhuber 2020) 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 $\tau_i = (s_1, a_1, g_1, \ldots, s_H, a_H, g_H)$ has a length H, DT aims to train an agent π_θ by minimizing the cross-entropy loss $\mathcal{L}_{\text{CE}}(\hat{\boldsymbol{a}}, \boldsymbol{a})$ between the predicted actions $\hat{\boldsymbol{a}}$ and the true actions $\boldsymbol{a}$ in the demonstrations. Although this BC-based approach has been successful in causal inference tasks (e.g., robotics), 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 (DFM). For simplicity, we assume $H = 1$. However, this can be extended to multidimensional data by employing factorization assumptions, as demonstrated in (Campbell et al. 2024). 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 $\mathrm{d}t$:

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

$$= \delta\{x^t, j\} + R_t(x^t, j)\mathrm{d}t \tag{2}$$

where $\delta\{i, j\}$ equals 1 when $i = j$ and 0 otherwise. We define a forward corruption process (data-to-noise interpolation) as $p_t(x^t) = \mathbb{E}_{p_{\text{data}}(x^1)}\left[p_{t|1}(x^t|x^1)\right]$, 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 ($\mathcal{C}$):

$$p^{\text{mask}}_{t|1}(x^t \mid x^1) = \mathcal{C}\left(t\delta\{x^1, x^t\} + (1-t)\delta\{[\mathbb{M}], x^t\}\right) \tag{3a}$$

$$p^{\text{unif}}_{t|1}(x^t \mid x^1) = \mathcal{C}\left(t\delta\{x^1, x^t\} + (1-t)\frac{1}{|X|}\right) \tag{3b}$$

Here, $[\mathbb{M}]$ represents a mask state (Devlin et al. 2018) 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 $\partial_t p_t(\cdot) = R_t(\cdot|\cdot)^\top p_t$. Campbell et al. (2024) proposes the use of a conditional rate matrix, $R_t(x^t|j) = \mathbb{E}_{p_{1|t}}[R_t(x^t, j|x^1)]$, which enables the simulation of the conditional flow, by sampling along CTMC.

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 ($p^\theta_{1|t}$) 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 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).

4 GenPlan: Method

Overview. The denoising planner integrates the DFM with sequence models and is built on two key components: (a) the denoising model $p^\theta_{1|t}$, trained as described in algorithm 1, and (b) the rate matrix $R_t(\boldsymbol{x}^t, \boldsymbol{x}'^t|\boldsymbol{x}^1)$. Together, these components enable flexible planning using the algorithm 2. We discuss several design choices that improve the planner's performance. For further details on the implementation and the formulation of the rate matrix, refer to Karthikeyan and Pant, Appendix F.

Energy-Guided Denoising Model

Given demonstrations (see Section 3), our aim is to learn an implicit energy function for a sequence, denoted as $\mathcal{E}(\boldsymbol{a}^t)$. 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 $\mathcal{E}(\boldsymbol{a}^1) = \mathbb{E}_{(\boldsymbol{a}^1)\sim\mathcal{D}} \sum_H [-\log p^\theta_{1|t}(\boldsymbol{a}^1|\boldsymbol{a}^0, \boldsymbol{o})]$ adapted from (Goyal, Dyer, and Berg-Kirkpatrick 2021).

Our approach leverages the DFM objective (Campbell et al. 2024, 2022) 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 (Goyal, Dyer, and Berg-Kirkpatrick 2021; Chen et al. 2023). The denoiser $p^\theta_{1|t}$ is optimized under the constraint that its entropy $\mathcal{H}$ remains above a lower bound β. This constraint encourages stochasticity, enhancing the gen-

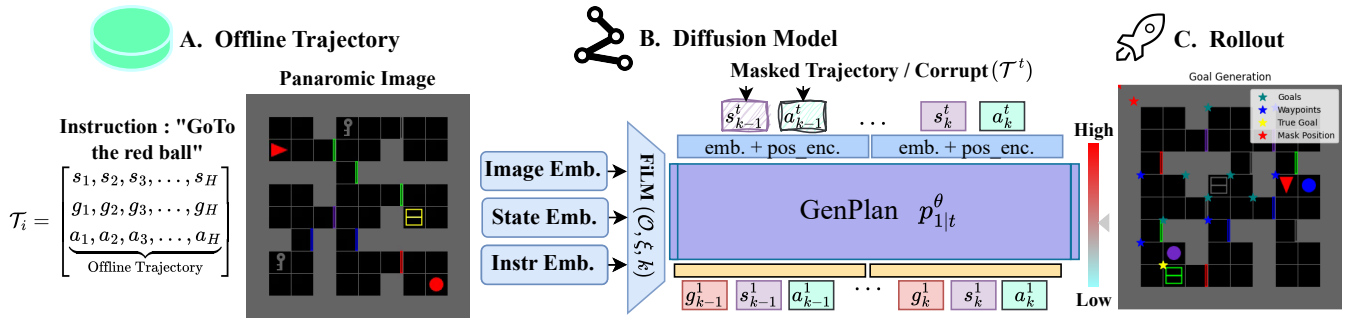


Figure 2: *Method Overview.* GenPlan, trained on offline data (*A*), learns to jointly model action, goal, and state distributions. In (*B*), the joint denoising model (see Section 4) takes in a corrupted trajectory τ^t and predicts the clean trajectory τ^1. *(C)* Demonstrates the joint inference of goals and actions by simulating the reverse CTMC, as detailed in Algorithm 2.

erative model's adaptability to new tasks and environments.

$$\min_{\theta} \mathbb{E}_{\boldsymbol{a}^0 \sim p_0, \boldsymbol{o} \sim \mathcal{D}} \left[\sum_{k=1}^{H} -\log p^{\theta}_{1|t}(\boldsymbol{a}^1 | \boldsymbol{a}^0, \boldsymbol{o}) \right], \tag{4a}$$

$$\text{s.t. } \mathbb{E}_{\boldsymbol{a}^0 \sim p_0, \boldsymbol{o} \sim \mathcal{D}} \left[\sum_{k=1}^{H} \mathcal{H}(p^{\theta}_{1|t}(\boldsymbol{a} | \boldsymbol{a}^0, \boldsymbol{o})) \right] \geq \beta \tag{4b}$$

By optimizing at the sequence level, the model avoids getting trapped in local minima and can naturally learn a multimodal distribution, where $\mathcal{E}(\boldsymbol{a}) \simeq \mathcal{E}(\boldsymbol{a}')$, finding multiple viable solutions. This framework enables the flexible formulation of RL and planning tasks. In Section 5, we demonstrate the use of a joint model (Section 4) for generating unconditional rollouts.

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 $\boldsymbol{o}$, which guides the action denoising process. Unlike existing methods (Chen et al. 2021, 2023) that rely on simulators for runtime goal positions and task proposals, GenPlan jointly learns the goal distribution, extracting a goal sequence $\boldsymbol{g}$ jointly with actions $\boldsymbol{a}$ 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 similarly to the action sequence (denoising based DFM) and can generalize to out-of-distribution tasks, provided the energy function is generalizable—successful trajectories receive low energy. Next, we briefly describe the processes involved in training and sampling from DFM (Campbell et al. 2024, 2022).

Forward Diffusion. We begin by corrupting samples drawn from the dataset $\mathcal{D}$ using the noise schedules (eq. 3a,3b). Specifically, we apply $\mathbb{E}_{p_{\text{data}}(\tau^1), t \sim \mathcal{U}[0,1]} p^{\text{mask}}_{t|1}(\tau^t | \tau^1)$, where t is the CTMC timestep that controls the amount of corruption. The corruption process is applied to $\mathbf{s}$, $\boldsymbol{a}$, and $\boldsymbol{g}$, 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 ($p^{\theta}_{1|t}$) to approximate the true data distribution. We employ a negative-log-likelihood-based loss (eq. 5) to predict $\mathbf{s}^1$, $\boldsymbol{g}^1$, and $\boldsymbol{a}^1$. As noted in (Campbell et al. 2024), the denoiser $p^{\theta}_{1|t}(\tau^1 \mid \tau^t)$ is independent of the choice of rate matrix used for sampling, $R_t(\boldsymbol{x}^t, \boldsymbol{x}'^t \mid \boldsymbol{x}^1)$. 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 (Chen et al. 2021), using bi-directional masks as in (Devlin et al. 2018) to allow future actions to influence preceding ones. Observations $\boldsymbol{o}$ are encoded using FiLM (Feature-wise Linear Modulation) (Perez et al. 2018) 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 2B).

Training Objective

The training objectives described in eq. 4a,4b are used to learn the model parameters (θ, λ). Here, θ represents the parameters of the denoiser $p^{\theta}_{1|t}$, which is parameterized as a transformer, as discussed in Section 4. The parameter $\lambda \in [0, \infty]$ is the Lagrangian multiplier associated with the constraints in eq. 4b. We alternate gradient descent steps between updating $p^{\theta}_{1|t}$ and λ. In practice, $\lambda \to 0$, ensuring that the lower bound β on entropy is satisfied (Zheng, Zhang, and Grover 2022). To generalize the denoiser across differ-

Algorithm 1: GenPlan Training

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

ent levels of corruption, we train the model to recover trajectories with varying noise levels. For each $\langle \boldsymbol{a}, \mathbf{s}, \boldsymbol{g} \rangle$, we apply the loss $\mathcal{L}_{\boldsymbol{x}}$ (see eq. 5). The Kronecker delta ensures that the loss is only calculated for the corrupted regions.

$$\mathcal{L}_{\boldsymbol{x}} = \left[-\sum_{k=1}^{H} \delta\{x^t_k, [\mathtt{M}]\} \log p^{\theta}_{1|t}(\boldsymbol{x}^1 | \boldsymbol{x}^t, \boldsymbol{o}) \right] \tag{5}$$

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 (Campbell et al. 2024). This procedure is applied to all components of the trajectories.

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^k_t depends only on x^t_k, j_k, and x^1_k. (2) Once the process reaches x^1_k, it remains there, i.e., $R^k_t(x^t_k = x^1_k, j_k \mid x^1_k) = 0$.*

$$\delta\{\boldsymbol{x}^t, \boldsymbol{x}'^t\} + \sum_{k=1}^{H} \delta\{\boldsymbol{x}^t_{\setminus k}, \boldsymbol{x}'^t_{\setminus k}\} \mathbb{E}_{p^{\theta}_{1|t}(x^1_k|\boldsymbol{x}^t)} \left[R^k_t(x^t_k, j_k | x^1_k) \right] \Delta t$$

5 Simulation Studies

We evaluate the performance of GenPlan in BabyAI (Chevalier-Boisvert et al. 2019) 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.

Algorithm 2: GenPlan Sampling

init $\tau^0 \sim p_0$, choice of $R_t(\tau^t, \cdot|\tau^1)$, $\Delta t = \frac{1}{I_{max}}$, get $\boldsymbol{o}$
for $t \in \{0, \Delta t, 2\Delta t, \ldots, 1\}$ **do**
  $R^{\theta}_t(\tau^t, \cdot) \leftarrow \mathbb{E}_{p^{\theta}_{1|t}(\tau^1|\tau^t, \boldsymbol{o})}\left[R_t(\tau^t, \cdot|\tau^1)\right]$
  $\tau^{t+\Delta t} \sim \mathcal{C}\left(\delta\{\tau^t, \tau^{t+\Delta t}\} + R^{\theta}_t(\tau^t, \tau^{t+\Delta t})\Delta t\right)$
  $t \leftarrow t + \Delta t$
end for
return $\boldsymbol{a}, \mathbf{s}, \boldsymbol{g}$ `// extract from` τ^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 4, 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 Karthikeyan and Pant, Appendix A.

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** (Chen et al. 2023): A 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. **DT** (Chen et al. 2021): 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 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 tra-

Env.	Uncond. Rollouts			Cond. Rollouts	
	GP-U	GP-M	LEAP⊖GC	LEAP	DT
Traj. Planning (TP)					
MazeS4G1	52.4%	**62%**	44%	49.2%	46.8%
MazeS4G2	38.8%	**39.6%**	20%	37.6%	35.2%
MazeS7G1	**45.6%**	44.8%	12%	33.2%	40%
MazeS7G2	**21.2%**	19.6%	3.6%	4%	13.6%
TP (7.6 ↑)	39.5%	**41.5%**	19.9%	31%	33.9%
Instr. Completion (IC)					
MazeClose	42.8%	**48.4%**	18%	38.8%	40%
DoorsOrder	**40.8%**	35.2%	11.2%	36.4%	**40.8%**
BlockUn	13.2%	**16%**	0%	0.8%	0%
KeyCorS3R3	11.6%	**17.6%**	0%	0.4%	3.6%
IC (8.2 ↑)	27.1%	**29.3%**	7.25%	19.1%	21.1%

Table 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.

jectories are categorized as unconditional rollouts. Comparative results are presented in Tables 1 and 2.

Remark 2 *LEAP outperforms popular baselines, including model-free RL algorithms like Batch-Constrained Deep Q-Learning (Fujimoto, Meger, and Precup 2019) and Implicit Q-Learning (Kostrikov, Nair, and Levine 2021), as well as model-based RL methods such as the Planning Transformer (Sun et al. 2022) and Model-based Offline Policy (Yu et al. 2020), in single-goal tasks within BabyAI environments, as shown in Table 1 of LEAP (Chen et al. 2023).*

Results. GenPlan consistently achieves higher success rates, particularly in adaptive planning and long-horizon tasks, as shown in Table 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***), DT performs comparably due to simulator access for correct goal sequencing, while GenPlan relies solely on instruction embedding as seen in Section 4.

Next, we assess the model's adaptation capabilities in novel environments (see Table 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).

Environment	Uncond. Rollouts			Cond. Rollouts	
	GP-U	GP-M	LEAP⊖GC	LEAP	DT
Adaptive Planning (AP)					
LocS10N10G2	82.4%	**88%**	76%	78%	25.6%
MazeS4N3G1	56%	**62%**	44.8%	48%	24%
MazeClose	31.2%	**34.8%**	10%	10%	8.8%
MazeS4G2	28.8%	**34.8%**	14%	18.4%	3.6%
SeqS5R2Un	35.6%	**42%**	29.2%	38%	29.2%
AP (13.84 ↑)	46.8%	**52.3%**	34.8%	38.4%	18.2%

Table 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.

Ablation Studies. We assess the significance of various components of GenPlan, results are presented in Table 3.

Joint Prediction ($\mathcal{L}_\mathbf{s} + \mathcal{L}_g$). This integration of goal generation ($\mathcal{L}_g$) with state sequence denoising ($\mathcal{L}_\mathbf{s}$) 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, where the agent struggles to achieve subgoals 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 1 and 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 4. With masking, the model more easily identifies jumps, simplifying the training. This is also reflected through faster convergence (training) of masked interpolants.

Discussion. Unlike baselines that rely on an oracle for goal positions—often unavailable in real-world scenar-

Attribute	GenPlan-M	Reduction	LEAP⊖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: *Ablation.* The values represent the mean success rates as detailed in Section 5, with reductions shown relative to the results in Tables 1 and 2.

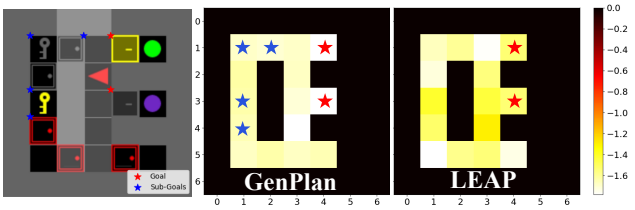


Figure 3: *Energy Landscape.* GenPlan, when conditioned on sub-goals, implicitly assigns minimal energy to necessary sub-goals (e.g., picking up keys, opening doors) for task completion. States closer to the white region are more likely to be transitioned into. LEAP, in contrast, does not prioritize these sub-tasks.

ios—the goal generation module in GenPlan dynamically updates sub-goals, enabling better state coverage (see Karthikeyan and Pant, Appendix B) and unconditional rollouts, where baselines fail to explore certain regions even with oracle guidance. To further illustrate these advantages, we present results on training progression through the learned energy landscape, mission success rates as a function of iterations, and the impact of environment stochasticity (see Karthikeyan and Pant, Appendix B).

Continuous Tasks

We evaluate the models on continuous manipulation tasks: (a) PushT (Florence et al. 2021) and (b) Franka Kitchen (Gupta et al. 2019). Both tasks use state-based variants in continuous manipulator environments. 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 Karthikeyan and Pant, Appendix D.

Env	Metric	GenPlan-M	DP-C	DP-T	VQ-BeT
PushT	IoU	0.73	0.73	**0.74**	0.68
Kitchen	# Tasks	3.40	2.62	3.44	**3.66**

Table 4: *Quantitative results.* Comparison of baselines on unconditional continuous-space tasks.

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) (Lee et al. 2024).

Unlike VQ-BeT, which employs a behavior cloning objective, GenPlan-M uses an energy-based loss (see Section 4) and DFM sampling with casual 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 (Lee et al. 2024).

Results. 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 outperforms the diffusion-based approach (DP-C) in the Kitchen environment, where diffusion falls behind by an entire subtask. These results highlight the advantages of modeling with categorical distributions for multi-task missions and unconditional planning.

6 Conclusion

We study the problem of learning to plan from demonstrations, particularly for unseen tasks and environments. We propose GenPlan, an energy-DFM-based planner that learns annealed energy landscapes and uses DFM sampling to iteratively denoise plans. Through simulation studies, we demonstrate how joint energy-based denoising improves performance in complex and long-horizon tasks.

Limitations. (1) The entropy lower bound β in eq. 4b is currently a hyper-parameter that must be manually specified. (2) We assume access to near-optimal demonstration trajectories for training (Section 5); however, this assumption may not hold in all settings. Initial results show that GenPlan performs well even when trained on datasets with a mixture of sub-optimal demonstrations (Karthikeyan and Pant, Appendix E). However, further studies are needed to assess its robustness to sub-optimality in demonstrations.

Future Work. To address the above limitations, we plan to extend GenPlan for online fine-tuning via hindsight experience replay (Zheng, Zhang, and Grover 2022; Furuta, Matsuo, and Gu 2022). Additionally, GenPlan offers a flexible and scalable framework that can be extended to multi-agent learning settings (Meng et al. 2022).

Acknowledgments

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Vector Scholarship in Artificial Intelligence, provided through the Vector Institute.

References

Campbell, A.; Benton, J.; De Bortoli, V.; Rainforth, T.; Deligiannidis, G.; and Doucet, A. 2022. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*.

Campbell, A.; Yim, J.; Barzilay, R.; Rainforth, T.; and Jaakkola, T. 2024. Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design. arXiv:2402.04997.

Chen, H.; Du, Y.; Chen, Y.; Tenenbaum, J. B.; and Vela, P. A. 2023. Planning with Sequence Models through Iterative Energy Minimization. In *ICLR*.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. In *Advances in Neural Information Processing Systems*, volume 34, 15084–15097.

Chevalier-Boisvert, M.; Bahdanau, D.; Lahlou, S.; Willems, L.; Saharia, C.; Nguyen, T. H.; and Bengio, Y. 2019. BabyAI: First Steps Towards Grounded Language Learning With a Human In the Loop. In *International Conference on Learning Representations*.

Chi, C.; Feng, S.; Du, Y.; Xu, Z.; Cousineau, E.; Burchfiel, B.; and Song, S. 2023. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. ArXiv:2303.04137 [cs].

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Du, Y.; Mao, J.; and Tenenbaum, J. B. 2024. Learning Iterative Reasoning through Energy Diffusion. In *Forty-first International Conference on Machine Learning*.

Emmons, S.; Eysenbach, B.; Kostrikov, I.; and Levine, S. 2022. RvS: What is Essential for Offline RL via Supervised Learning? In *International Conference on Learning Representations*.

Eysenbach, B.; Zhang, T.; Levine, S.; and Salakhutdinov, R. 2022. Contrastive Learning as Goal-Conditioned Reinforcement Learning. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Florence, P.; Lynch, C.; Zeng, A.; Ramirez, O.; Wahid, A.; Downs, L.; Wong, A.; Lee, J.; Mordatch, I.; and Tompson, J. 2021. Implicit Behavioral Cloning. *Conference on Robot Learning (CoRL)*.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In *International Conference on Machine Learning*, 2052–2062.

Furuta, H.; Matsuo, Y.; and Gu, S. S. 2022. Generalized Decision Transformer for Offline Hindsight Information Matching. In *International Conference on Learning Representations*.

Goyal, K.; Dyer, C.; and Berg-Kirkpatrick, T. 2021. Exposing the Implicit Energy Networks behind Masked Language Models via Metropolis–Hastings. *arXiv preprint arXiv:2106.02736*.

Gupta, A.; Kumar, V.; Lynch, C.; Levine, S.; and Hausman, K. 2019. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*.

Haarnoja, T.; Tang, H.; Abbeel, P.; and Levine, S. 2017. Reinforcement Learning with Deep Energy-Based Policies. arXiv:1702.08165.

Janner, M.; Du, Y.; Tenenbaum, J. B.; and Levine, S. 2022. Planning with Diffusion for Flexible Behavior Synthesis. arXiv:2205.09991.

Janner, M.; Li, Q.; and Levine, S. 2021. Offline Reinforcement Learning as One Big Sequence Modeling Problem.

Karthikeyan, A.; and Pant, Y. V. 2024. GenPlan: Generative sequence models as adaptive planners. arXiv:2412.08565.

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

Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; and Levine, S. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32.

Lambert, N.; Pister, K.; and Calandra, R. 2022. Investigating Compounding Prediction Errors in Learned Dynamics Models. arXiv:2203.09637.

Lee, S.; Wang, Y.; Etukuru, H.; Kim, H. J.; Shafiullah, N. M. M.; and Pinto, L. 2024. Behavior Generation with Latent Actions. *arXiv preprint arXiv:2403.03181*.

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. arXiv:2005.01643.

Meng, L.; Wen, M.; Yang, Y.; Le, C.; Li, X.; Zhang, W.; Wen, Y.; Zhang, H.; Wang, J.; and Xu, B. 2022. Offline Pre-trained Multi-Agent Decision Transformer: One Big Sequence Model Tackles All SMAC Tasks. arXiv:2112.02845.

Perez, E.; Strub, F.; De Vries, H.; Dumoulin, V.; and Courville, A. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Schmidhuber, J. 2020. Reinforcement Learning Upside Down: Don't Predict Rewards – Just Map Them to Actions. arXiv:1912.02875.

Sun, J.; Huang, D.-A.; Lu, B.; Liu, Y.-H.; Zhou, B.; and Garg, A. 2022. PlaTe: Visually-Grounded Planning With Transformers in Procedural Tasks. *IEEE Robotics and Automation Letters*, 7(2): 4924–4930.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J.; Levine, S.; Finn, C.; and Ma, T. 2020. MOPO: Model-based Offline Policy Optimization. *arXiv preprint arXiv:2005.13239*.

Zheng, Q.; Zhang, A.; and Grover, A. 2022. Online decision transformer. In *International Conference on Machine Learning*. PMLR.