STATE COMBINATORIAL GENERALIZATION IN DECISION MAKING WITH CONDITIONAL DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many real-world decision-making problems are combinatorial in nature, where states (e.g., surrounding traffic of a self-driving car) can be seen as a combination of basic elements (e.g., pedestrians, trees, and other cars). Due to combinatorial complexity, observing all combinations of basic elements in the training set is infeasible, which leads to an essential yet understudied problem of *zero-shot generalization to states that are unseen combinations of previously seen elements.* In this work, we first formalize this problem and then demonstrate how existing value-based reinforcement learning (RL) algorithms struggle due to unreliable value predictions in unseen states. We argue that this problem cannot be addressed with exploration alone, but requires more expressive and generalizable models. We demonstrate that behavior cloning with a conditioned diffusion model trained on expert trajectory generalizes better to states formed by new combinations of seen elements than traditional RL methods. Through experiments in maze, driving, and multiagent environments, we show that conditioned diffusion models outperform traditional RL techniques and highlight the broad applicability of our problem formulation.

1 INTRODUCTION

In many real-world decision-making tasks, environments can be broken down into combinations of fundamental elements. For instance, in self-driving tasks, the surrounding environment consists of elements like bicycles, pedestrians, and cars. Due to the exponential growth of possible element combinations, it is impractical to encounter and learn from every possible configuration during training. Rather than learning how to act in each unique combination, humans instead learn to interact with individual elements – such as following a car or avoiding pedestrians – and then extrapolate this knowledge to unseen combinations of elements. Therefore, it is important to study the *generalization to unseen combinations of known elements*, hereafter referred to as the out-of-combination (OOC) generalization, and to develop algorithms that can effectively handle these unseen scenarios.

Despite the success of reinforcement learning (RL) in decision-making tasks, many existing RL algorithms, particularly in offline settings, struggle to perform adeptly under state distribution shifts between training and testing, which typically occur when the learned policy visits states that differ from the data collection policy at test time (Levine et al., 2020; Kakade & Langford, 2002; Lyu et al., 2022; Schulman, 2015). While there have been works studying this problem, most of them either (1) focus on distribution shifts where the training and testing sets share the same support but different probability densities, without accounting for the presence of entirely new and unseen element combinations (Finn et al., 2017; Ghosh et al., 2022), or (2) allow unseen elements in test combinations, which makes the problem ill-posed without introducing other potentially unrealistic assumptions (Song et al., 2024; Zhao et al., 2022). As a result, these works have failed to recognize and address the critical challenge of generalization to unseen combinations of seen elements and therefore fail to capture and compose existing knowledge for these fundamental elements.

In this work, we directly tackle the problem of state combinatorial generalization in decision-making tasks, where testing states consist of unseen combinations of elements encountered during training. As illustrated in Figure 1, our task differs conceptually from traditional distribution shift problems. Unlike simple distribution shifts, where the testing set remains within the support of the training

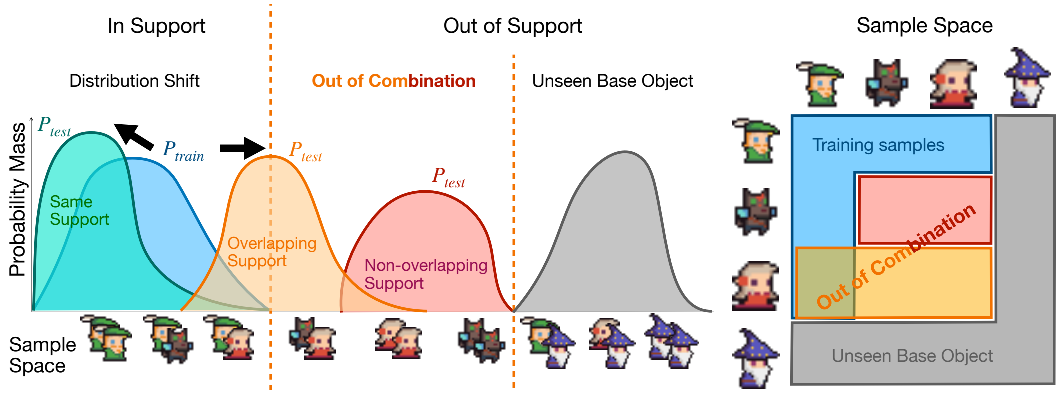


Figure 1: Different forms of out-of-distribution states. are seen base elements and is unseen base element. Their combination forms the sample space. Classic distribution shift assumes states to have the same support but different probability density. We study generalization for out-of-combination states in this work, where test time state distribution has different and possibly non-overlapping support compared to training states.

set, our proposed task requires algorithms to handle out-of-support states that are never seen during training. This makes our problem both more challenging and more representative of real-world scenarios. At the same time, our OOC setting is better defined than the unconstrained out-of-support (OOS) setting, where testing states may include completely arbitrary unseen elements and therefore is inadequately formulated and intractable without other potentially impractical assumptions such as the existence of state distance metrics (Song et al., 2024) or isomorphic Markov decision processes (MDPs) (Zhao et al., 2022). By focusing on new combinations of known elements, our setting strikes a balance between real-world applicability and tractability, making it more suitable for standardized evaluation and formal analysis.

To facilitate this study, we first provide formal definitions of state combination and OOC generalization. We then demonstrate the challenge of this task by showing how traditional RL algorithms struggle to generalize in this setting due to unreliable value prediction, and the need for a more expressive policy. On the hunt for a suitable solution, we draw inspiration from the linear manifold hypothesis in diffusion models (Chung et al., 2023; He et al., 2024b) and recent advances in combinatorial image generation (Okawa et al., 2024), and present diffusion models as a promising direction by showing how they can naturally account for the combinatorial structure of states into the diffusion process, enabling better generalization in OOC settings.

Experimentally, we evaluate the models on three distinct different RL environments: maze, driving, and multiagent games. All three settings are easily adaptable to the OOC generalization problem using existing RL frameworks, demonstrating the broad applicability of the combinatorial state setup. We demonstrate behavior cloning (BC) with a conditioned diffusion model outperforms not only vanilla BC and offline RL methods like CQL (Kumar et al., 2020) but also online RL methods like PPO (Schulman et al., 2017) in zero-shot OOC generalization. To explore factors contributing to its generalization, we visualize the states predicted by the conditioned diffusion model. Our results demonstrate that the model effectively captures the core attributes of each base element and accurately composes future states by integrating these fundamental attributes. We demonstrate that, while exploration is commonly used to enhance model generalization, OOC generalization relies instead on the use of a more expressive policy.

2 Related Work

2.1 Generalization in RL

Zero-shot domain transfer The problem of zero-shot domain transfer assumes that the model is trained and tested on different domains that might have some similarities but are sampled from different underlying distributions (Kirk et al., 2023). One widely used technique is domain randomization, approaching this problem by producing a wide range of contexts in simulation (Kirk et al., 2023; Mehta et al., 2020). Although the focus is also unsupported state space, it commonly assumes that information about the testing environment is not accessible (Mehta et al., 2020) and focuses more on sim2real problems (Kirk et al., 2023). Whereas we assume test time information is given through conditioning but restricting the training set to have narrow coverage.

Subtask and Hierarchical RL These two settings focus on learning reusable skills that can be sequenced to complete long horizon tasks (Parr & Russell, 1997; Lin et al., 2022; Dietterich, 2000; Nachum et al., 2018; Jothimurugan et al., 2023; Bakirtzis et al., 2024). The concept of compositionally is also a key component in subtask learning, where different sub-trajectories or intermediate goals are composed together to better perform a long horizon task (Jothimurugan et al., 2023; Lin et al., 2022; Bakirtzis et al., 2024; Mendez et al., 2022). We would like to note the difference between compositionally in trajectory stitching and our definition of state composition, where *subtasks in trajectory stitching are often data supported* as shown in Figure 2.

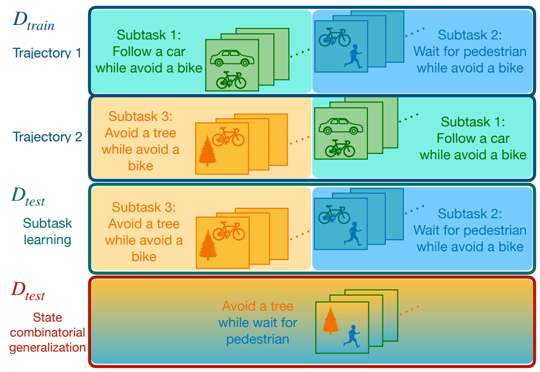


Figure 2: Visualization of states in trajectories for training, subtask learning, and state combinatorial generalization. Subtask learning involves stitching together subtask 3 in the training trajectory 2 with subtask 2 in trajectory 1. Combinatorial generalization involves simultaneously avoiding a tree and waiting for a pedestrian. Each of those two elements appeared in the training states but had never been combined.

2.2 Combinatorial Generalization

In Computer Vision The closest line of work to ours is combinatorial generalization for image generation where the model needs to learn new combinations of a discrete set of basic concepts like color and shapes and generalize to unseen combinations (Wiedemer et al., 2024; Okawa et al., 2024; Schott et al., 2021; Hwang et al., 2023). This problem is often approached with disentangled representation learning (Liu et al., 2023; Schott et al., 2021) with models like VAE but little evidence shows they can fully exhibit generalization ability (Schott et al., 2021; Montero et al., 2020). Okawa et al. (2024) studied the capabilities of conditioned diffusion models on a synthetic shape generation task and showed that their composition ability emerges with enough training, first to closer concepts, then to farther ones.

In RL Song et al. (2024) addresses the problem of generalization to unsupported states by decomposing it into the closest state in the training set and their difference, which requires the existence of a distance function to map the unseen state back to data supported region to ensure conservatism. However, we do not assume there exists a distance function between states and we do not explicitly encourage the model to be conservative. Zhao et al. (2022) uses an object oriented environment to study compositional generalization by learning the world model under the assumption that different combinations have isomorphic MDPs and objects are replaceable with each other. However, we do not assume our MDPs to be isomorphic, as each object in our setup possesses unique attributes that are non-transferable, leading to the emergence of complex underlying modalities. To the best of our knowledge, we are the first to investigate the problem of generalization to unsupported states with novel combinations of basic elements, without relying on mapping unseen states back to data-supported regions.

2.3 Diffusion Model for Decision making

Diffusion models emerged as a popular architecture for decision-making tasks and demonstrated superior performance compared to traditional RL algorithms, especially on long-horizon planning tasks (Janner et al., 2022; Wang et al., 2022; Liang et al., 2023a; Mishra et al., 2023). Some following work further studied conditioned diffusion models (Chi et al., 2023; Ajay et al., 2022; Li et al., 2023) and demonstrated their ability to stitch trajectories with different skills or constraints together. Application in multi-task environment (He et al., 2024a; Liang et al., 2023b) and meta-learning setting (Ni et al., 2023; Zhang et al., 2024) further demonstrate their ability to capture multi-modality information in the offline dataset.

3 Problem Formulation

In this section, we formally define the problem of state combinatorial generalization by providing definitions for state combination and identify out-of-combination generalization as a problem for generalization to different supports in the same sample space.

3.1 States Formed by Element Combinations

Following Wiedemer et al. (2024), we first denote $e \in \mathbf{E}$ to be a *base element* for an environment. A base element is defined to be the most elementary and identifiable element that is relevant to the decision making task of interest. For example, in a traffic environment, the set $\mathbf{E}$ can be the set of vehicles that can occur in the environment such as {car, bike}; and in a 2D maze environment, the set $\mathbf{E}$ can be the set of possible locations labeled by the x, y-axis coordinate of the agent, i.e. $\mathbb{R}^2$. Suppose there are a finite number of n base elements in an environment. Since these elements are the fundamental components relevant to the decision making task, we can form a *latent vector* $\boldsymbol{z} = (z_1, z_2, ..., z_n) \in \mathbf{Z} \equiv \mathbf{E}^n$, where $z_i \in \mathbf{E}\ \forall i \in \{1, \ldots, n\}$ that represents the combination of all rudimentary components appearing in this environment related to the decision making task.

Each element can also be associated with a collection of *attributes* $\boldsymbol{r}$ such as the color of the vehicle and the velocity of the agent. Attributes are components that are necessary for rendering the states and the *rendering function* $f(\boldsymbol{z}, (\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_n))$ can then map the latent and the attributes to a state $\boldsymbol{s} \in \mathbf{S}$. In the traffic environment example, f is equivalent to reconstructing the cars and the bikes given their colors and positions, etc. All reconstructed base elements collectively determine a state $\boldsymbol{s}$. Concretely, we provide the following definition:

Definition 3.1 (States and latent vectors)**.** *For any state $\boldsymbol{s}$ with n base elements in state space $\boldsymbol{S}$ and rendering function f, we have $\boldsymbol{s} = f(\boldsymbol{z}, (\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_n))$ where the corresponding latent vector $\boldsymbol{z}$ in latent space $\boldsymbol{Z} \equiv \mathbf{E}^n$ for $\boldsymbol{s}$ is $\boldsymbol{z} = \big(z_1, z_2, ..., z_n\big)$ where $z_i \in \mathbf{E}$ for $i = 1, ..., n$.*

With our definition of base elements and states, *the combinatorial property of states naturally follows as the composition of different base elements in the latent space.*

Notice that in practice, for the same environment, one can define different base element sets depending on the desired granularity of the task. In addition, since we usually can only obtain observations of the states, in practice we can only extract the empirical latent vector $\tilde{\boldsymbol{z}}$ from the observation.

3.2 Generalization on Probability Space Support

Since we have identified the fundamental elements of the state in the target decision making task, we can formulate the distribution of states with the probability spaces of latent vectors. When our base element set is discrete, finite, and countable, the probability mass function (PMF) p can directly ascribe a probability to a sample in $\mathbf{Z}$. Then we can define the corresponding probability space as

Definition 3.2 (Probability space for discrete latents)**.** *Define the sample space $\mathbf{Z}$ as the set of all possible $\boldsymbol{z}$. σ-algebra $\Sigma = 2^{\mathbf{Z}}$ is the power set of $\mathbf{Z}$. $p : \mathbf{Z} \to [0, 1]$ such that $\sum_{\boldsymbol{z} \in \mathbf{Z}} p(\boldsymbol{z}) = 1$ is the PMF. Then the probability space over the latent vector $\mathbf{z}$ can be defined as $P = (\mathbf{Z}, \Sigma, p)$.*

When $\mathbf{Z}$ is a continuous space, we can also have the correspnding definitions.

Definition 3.3 (Probability space for continuous latents)**.** *Define the sample space $\mathbf{Z}$ as the set of all possible $\boldsymbol{z}$. σ-algebra $\Sigma = \mathcal{B}(\mathbf{Z})$ is the Borel set of $\mathbf{Z}$. $p : \mathbf{Z} \to [0, 1]$ such that $\int_{\boldsymbol{z} \in \mathbf{Z}} p(\boldsymbol{z}) d\boldsymbol{z} = 1$ is the probability dense function (PDF). Then the probability space over the latent vector $\mathbf{z}$ can be defined as $P = (\mathbf{Z}, \Sigma, p)$.*

The support of $P = (\mathbf{Z}, \Sigma, p)$ can then be defined as $\operatorname{supp} P := \{\boldsymbol{z} \in \mathbf{Z} : p(\boldsymbol{z}) > 0\}$.

State combinatorial generalization, or OOC generalization, is then defined as generalizing to latent probability space with a different support. Denote the latent probability space of training states as $P_{train} = (\mathbf{Z}, \Sigma_{train}, p_{train})$ and testing states as $P_{test} = (\mathbf{Z}, \Sigma_{test}, p_{test})$, then combinatorial generalization assumes $\operatorname{supp}\{P_{train}\} \neq \operatorname{supp}\{P_{test}\}$. That is to say, combinatorial generalization in state space requires generalizing to a distribution of latent vectors with different, and possibly non-overlapping support (Wiedemer et al., 2024). Whereas traditional distribution shift in RL normally assumes different PMF or PDF ($p_{train} \neq p_{test}$), as shown in Figure 1.

3.3 Constraint for OOC Generalization

One crucial assumption made by OOC generalization is that all base elements are seen at training time. Recall $\boldsymbol{z} = (z_1, z_2, ..., z_n)$ where $z_i \in \mathbf{E}$ for $i = 1, ..., n$. This indicates that the marginal distribution $p(z_i) > 0$ for all z_i at training time, or equivalently the training probability space has full support over the marginals. For discrete latent spaces, this also implies that every base element that appeared in the sample space would appear at least once in one latent feature $\boldsymbol{z}$. To ensure full support of base elements, the union of marginal supports at test time should be a subset of that at training time. Finally, to test generalizability, we assume $\text{supp}\{P_{train}\} \subsetneqq \mathbf{Z}$, i.e. the training probability space doesn't have full support on the entire latent space.

Constraint 3.4 (Combinatorial support)**.** *Given probability spaces $P = (\mathbf{Z}, \Sigma_P, p)$ and $Q = (\mathbf{Z}, \Sigma_Q, q)$ over latent vector $\boldsymbol{z} = (z_1, z_2, ..., z_n) \in \mathbf{Z}$ where $z_i \in \mathbf{E}$ for $i = 1, ..., n$, P has full combinatorial support for Q if: $\bigcup_{i=0}^{n}\{z_i \in \mathbf{E} : q(z_i) > 0\} \subseteq \bigcup_{i=0}^{n}\{z_i \in \mathbf{E} : p(z_i) > 0\}$.*

4 Why Traditional RL Fails

Most RL algorithms include estimating the expected cumulative reward of choosing a specific action given the current state (Schulman et al., 2017; Kumar et al., 2020; Haarnoja et al., 2018). We demonstrate the estimation of value functions is problematic given unsupported states and this can not be solved by more exploration or more training data in this section.

4.1 RL and expected reward estimaiton

Most deep RL algorithms rely on learning a Q or Value function, which takes in the current state as network input and predicts the expected future reward (Schulman et al., 2017; Haarnoja et al., 2018). Since states with unseen composition are unsupported and fall within the under-trained regions of the neural network, the value prediction is highly unreliable. This affects both value-based methods that directly choose the maximum action with erroneous Q value and policy-based methods that update the actor with an erroneous value prediction. We plot the expected Q-values learned by CQL alongside the actual return-to-go in both failed and success scenarios in Roundabout environment (Leurent, 2018) (Section 7.1) when presented with OOC states in Figure 3. The grey dashed line is the expected Q-values the model predicts for in-distribution states.

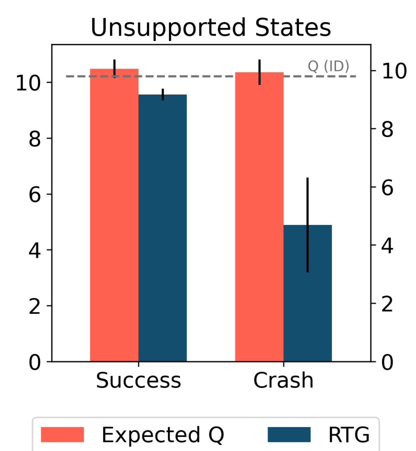


Figure 3: Expected Q value of CQL and actual return-to-go (RTG) in unsupported states in Roundabout environment.

One key observation can be made: *Q function shows signs of memorizing, which assigns similar Q values for both training and OOC states.*

Despite the problem of distribution shift being a central challenge for offline RL (Levine et al., 2020), online methods also suffer from unseen states when zero-shot generalizing to unsupported states. Traditionally distribution shifts are mitigated with a wider training state distribution under the assumption that test time states are sampled from a distribution with different probability density but same support. *However, since new states with different object combinations are out of support of the training environment, using a more exploratory online policy or collecting more training trajectories for offline RL will not fundamentally solve this issue*. We need a policy with better generalization to unsupported states to achieve zero-shot generalization in this problem.

5 Why Diffusion Models Generalize Better

We first introduce diffusion model notations and then provide a proof sketch and experimental evidence of why diffusion models can generalize to OOC states.

5.1 Diffusion Models

Diffusion models are among the most popular methods for density estimation. Ho et al. (2020) proposed DDPM to model the data generation process with a forward and reverse process. In the forward process, noise is added to corrupt data $\boldsymbol{x}_t$ iteratively for T timesteps towards a standard Gaussian distribution. The target of diffusion modeling is to learn the reverse process $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu_\theta}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma_\theta}(\boldsymbol{x}_t, t))$. This way, we can sample from the data distribution by first obtaining a Gaussian noise $\boldsymbol{x}_t$ and then iteratively sampling from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. With reparametrization trick, we can train a model $\boldsymbol{\epsilon_\theta}$ to predict the noise $\boldsymbol{\epsilon}$ at each timestep t, and gradually denoise using update rule $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t-\sigma_t^2}}\epsilon_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon_t}$, where $\boldsymbol{\epsilon_t} \sim \mathcal{N}(0, I)$ with variance schedulers $\alpha_t, \bar{\alpha}_t$. Given the same pretained diffusion model, one can also perform DDIM sampling (Song et al., 2020a) $\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t, t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}}\epsilon_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon_t}$ to enable fast sampling.

Song et al. (2020b) formally established the connection between diffusion models and score-based stochastic differential equations (SDE). Interestingly, they discovered that each diffusion process has a corresponding probability flow ODE that shares the same intermediate marginal distributions $p(\boldsymbol{x}_t, t)$ for all t. The transformation between probability flow ODE and SDE can be easily achieved by adjusting the random noise hyperparameter σ in DDIM sampling.

5.2 OOC Generalization in Diffusion Models

We demonstrate that a well-trained diffusion model can naturally sample OOC states that satisfy combinatorial support constraint 3.4 with non-zero probability at test time. The key idea is that pseudo-random denoising trajectories can be constructed at inference time that yield OOC samples with non-zero probability.

Since our states are formed by combinations of base elements (Definition 3.1), with well constructed $\mathbf{Z}$ we can assume that the states lie on a lower dimensional manifold $\mathcal{M}$ (representing combinations of base elements) embedded in the high dimensional ambient state space. In some cases such as maze navigation where the latent space is a linear subspace, we can even assume that the underlying manifold $\mathcal{M}$ is a linear manifold whose tangent space is isomorphic to itself. With these assumptions, we present the following corollary whose proof is shown in the appendix B.

Corollary 5.1. *Suppose the states lie along a linear manifold $\mathcal{M}$ in the state space $\mathbf{S}$ and the latent space $\mathbf{Z}$ is well constructed so that $\mathbf{Z}$ is (affine) isomorphic to $\mathcal{M}$. Then a diffusion model p_θ that is well trained on P_{train} can sample an OOC state with non-zero probability.*

While the linear manifold assumption may not hold for more complex states, recent computer vision research provides evidence of the combinatorial generalization capabilities of diffusion models in more complicated data spaces: Okawa et al. (2024) showed that given different concepts like shape, color, and size in synthetic shape generation, conditional diffusion models demonstrate a multiplicative emergence of combinatorial abilities where it will first learn how to generalize to concepts closer to the training samples (i.e. only change one of color, shape, and size) and eventually adopt full compositional generalization ability with enough training. Aithal et al. (2024) identifies the phenomena where diffusion models generate samples out of the support of training distribution through interpolating different complex modes on a data manifold. Kadkhodaie et al. (2023) demonstrate generalization to unsupported data by showing two diffusion models trained on large enough non-overlapping data converge to the same denoising function. In the next sections, we discuss how to use diffusion models to handle this challenging problem and also provide empirical evidence in decision-making tasks.

6 Conditioned Planning with Diffusion

6.1 Diffusion for Imaginary Trajectory Prediction

We follow the same setup as in Janner et al. (2022) and learn a conditional diffusion model that denoises (predicts) the future state-action pairs given the current state and the latent vector. This formulation excels at OOC state generalization because the capability of the diffusion model to sample OOC states enables it to generate a better world model prediction. Since future states of an OOC state are almost always going to be another OOC state (same base element but different attributes), the diffusion model is capable of generating reasonable future predictions and thus facilitating planning. This advantage is visually demonstrated in Figure 6.

6.2 Experiment Setup

Since each state corresponds to a latent vector, we will use the latent vector, which is oftentimes the category of the base elements in the current state, as conditioning for diffusion models. This information is extracted from observations and is equally accessible to all models. We let the correspondence between observation, action, and latent vector (rendering function f) be implicitly learned by the model and incorporate a cross-attention layer after each residual block for Diffusion Unet (Janner et al., 2022) to facilitate learning. The expert offline data is collected for behavior cloning by fully trained PPO agents on each combination of the training environment and includes only successful rollouts. Since expert trajectories are used for training, we model actions together with states under the assumption that unseen states generated from new conditioning will correspond to statistically reasonable actions. The first action in the generated trajectory is then used to step the environment (Appendix D.5). Detailed architecture of the model can be found in Appendix D.3 and the planning process is described in Algorithm 1.

7 Experiments

The primary goal of our experiments is to answer the following questions: (1) (Wide applicability) Does the state-space of different existing RL environments exhibit a compositional nature? (2) (Advantages) What are some interesting features conditional diffusion models have that contribute to their performance when generalizing to OOC states? (3) (Conditioning) Does conditioning help with OOC generalization?

7.1 Single-agent Environment

Environment HighwayEnv (Leurent, 2018) is a self-driving environment where the agent needs to control a vehicle to navigate between traffic controlled by predefined rules. We specifically look at the Roundabout environment with two types of traffic: cars and bicycles (Visualization in Appendix D.7.1).

State in this environment is a composition of four environment vehicles that are either cars or bicycles and the ego agent, which is always a car. Environment observation contains observability, the locations and speed of the ego and surrounding agent, and *whether this agent is a car or a bike (Conditioning)*. During training time, the environment will only generate traffic of all cars or all bicycles with equal probability. During test time, environments will generate a mixture of cars and bicycles (detailed setup in Appendix D.8). Cars and bicycles have different sizes, max speeds, and accelerations, leading to different behavior patterns. This is an instance of generalizing to OOC states with non-overlapping support.

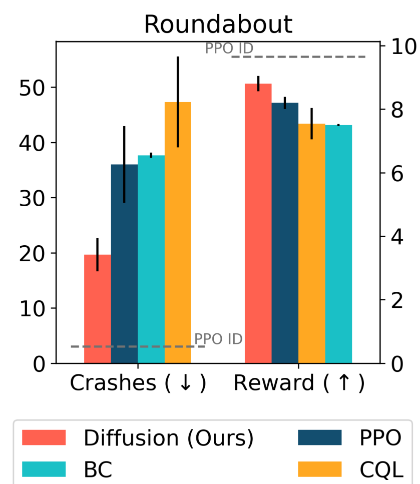


Figure 4: Total number of crashes and average reward for BC(MLP), PPO, CQL, and diffusion model in the testing environment.

Results The conditional diffusion model has almost half the number of crashes and higher reward when zero-shot generalizing to states with mixture traffic. Since we train the diffusion model exclusively on successful PPO trajectories, the training state distribution for diffusion is much narrower compared to that of other online methods. This is particularly interesting since it is widely acknowledged that online models have better generalization compared to offline models (Levine et al., 2020).

Takeaway 1: Conditional diffusion models, trained on an offline dataset with narrow state distribution with full combinatorial generalization support, have better zero-shot generalization performance to OOC states compared to online RL trained in the same environment.

7.2 Multi-agent Environment

Environment The StarCraft Multi-Agent Challenge (SMAC/SMACv2) (Samvelyan et al., 2019; Ellis et al., 2022) is a multi-agent collaborative game that takes several learning agents, each con-

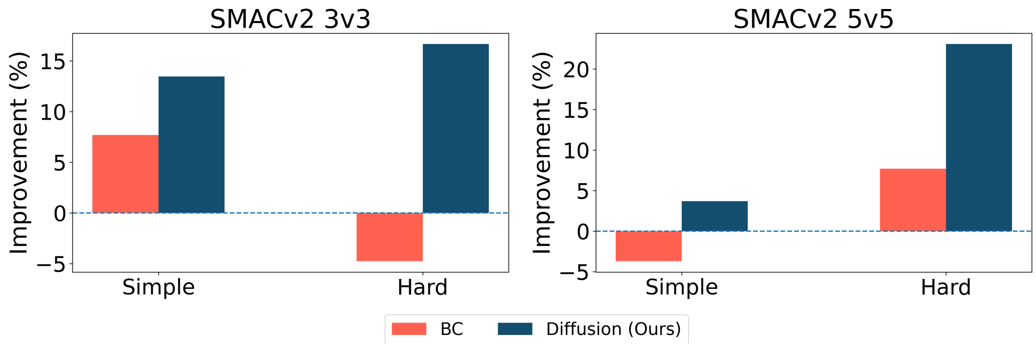


Figure 5: Relative improvement % compared to MAPPO on two SMACv2 scenarios: 3v3 and 5v5. Conditional Diffusion show large improvements over the MAPPO baseline, specially in the hard scenario, where we train on teams with the same unit type only but test on random team compositions.

trolling a single army unit, to defeat the enemy units controlled by the built-in heuristic AI. This benchmark is particularly challenging for its diverse army unit behaviors and complex team combinations, which enable diverse strategies like focus fire and kiting enemies to emerge (Ellis et al., 2024). Each agent's observation includes health, shield, position, *unit type (Conditioning)* of its own, visible teammates, and enemies.

We treat one agent as the ego agent, and consider its teammates and enemies as part of the environment. Then states can be naturally seen as compositions of the unit types in a particular playthrough. *We expect the ego agent to generate different policies when playing with or against different types of units, and we aim to test OOC generalization by changing the unit composition in the environment.* Since we use a MAPPO (Yu et al., 2022) for data collection, we report the performance gain/loss compared to MAPPO as shown in Figure 5. To treat the teammates and enemies of one particular agent as environment and change their combination, we control one unit with a conditional diffusion model and let MAPPO control the rest of its teammates.

Setup The unit types in this experiment are Protoss.Stalker, Protoss.Zealot, and Protoss.Colossus, referred to as a, b, c respectively. We evaluate on two OOC scenarios: (1) *(Simple: Different but overlapping support)*: Train the model on randomly generated combinations (ABC) of all units and test it where all the units on the team have same type (AAA), (2) *(Hard: Non-overlapping support)*: the opposite scenario, where we train on teams with only one unit type (AAA), but during test-time we see any composition of these three units (ABC). More information about our setup could be found in Appendix D.9.

Results MAPPO performance drastically dropped in the hard OOC scenario by 55.2% for 5v5 and 50% for 3v3. If we substitute one agent generated by MAPPO with conditional diffusion, the success rate can be improved by 16.7% for 3v3 and 23.1% for 5v5 in hard OOC scenario as shown in Figure 5. Detailed success rates are shown in Table 10 and Table 11.

Takeaway 2: Multi-agent RL, viewed from the perspective of a single ego agent, naturally requires combinatorial generalization to collaborate/compete with different agent types. Compositional complexity can be found in a wide range of distinctly different real-world tasks like driving and multiagent decision-making

7.3 How Do Conditional Diffusion Models Generalize to OOC States?

To see how diffusion models generalize to OOC states, we render the states predicted by the diffusion models given different conditionings with the same current state, as shown below in Figure 6.

We can see that *conditionings determine the unit type of agent predicted by the diffusion model and also their behavior pattern. Whereas the current state determines other attributes like initial location and health.* Different conditionings will lead to different strategies. The circle unit has attack range 1 and the square unit has attack range 6. For units with short attack ranges, the optimal strategy is to approach their enemies before initiating an attack. Conversely, agents with large attack ranges are advised to attack their enemies from a distance to ensure their own safety. Figure 6 shows that if we condition on all circles, the diffusion model thinks players will form a cluster and if condition

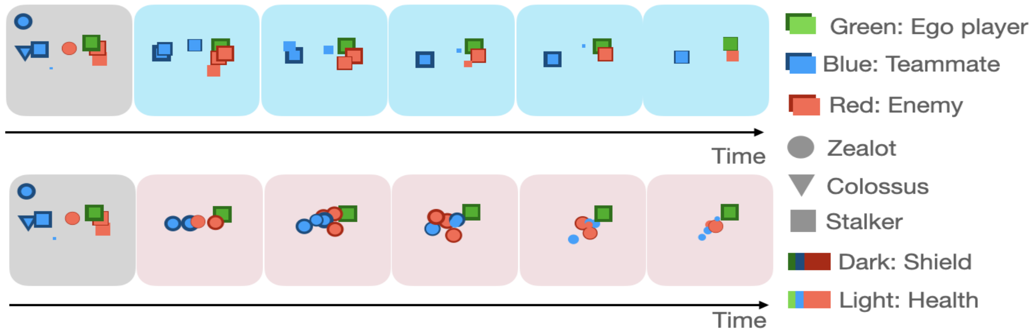


Figure 6: Rendering of future states predicted by the diffusion model given different conditionings. The grey box is the current state. Blue backgrounds are conditional on all Squares (long attack range) and pink backgrounds are conditioned on all circles (short attack range). Smaller sizes represent less shield and health. More examples shown in Appendix D.9.6.

on all squares, it will predict the players to attack each other from a distance, aligning well with the optimal policy. This demonstrates conditioned diffusion models' ability to *implicitly decompose states to learn underlying compositions* and *capture multimodality of different unit behavior* in the training data. It also demonstrates its ability to perform state stitching to accurately predict the world model.

Takeaway 3: Conditioned diffusion models show significant promise by effectively decomposing and capturing modes of individual base elements and performing state stitching, which helps them to accurately predict the world dynamics and generalize to OOC scenarios.

8 Ablations

In this section, we ablate over our design choices to (1) show the necessity of using the inductive bias of state latent vector as conditioning, (2) different model architectures to incorporate conditioning information

8.1 Necessity of Combinatorial Inductive Bias

We compare trajectories generated by the conditioned and unconditioned diffusion models in this section to demonstrate the importance of using combinatorial latent information as conditioning. In Maze2D (Fu et al., 2020), we formulate the navigation problem as a one-step generation process where the diffusion model learns how to generate an entire valid trajectory without rolling out current action and replan. Since there is only one planning step in this process, the generated trajectory can be seen as the "state" in this setting, where unseen trajectories correspond to unseen states instead of time-horizon trajectory stitching. The inductive bias we use is every training trajectory will pass through three waypoints that equally slice the trajectory. In this case, the set of all waypoints forms the base element set and their combination is the latent vector that determines the shape of a generated maze trajectory. During training, we extract three points that equally slice the trajectory and use them as conditioning. During test time, we specify a new combination of three waypoints we want the generated trajectory to pass.

We see that the unconditioned diffusion model successfully generated a trajectory if the start and end positions are in the training set (Figure 7b) but failed for unseen start and end points (Figure 7c). This demonstrates that unconditioned diffusion struggles to approximate unseen distributions. However, if conditioning on an unseen combination of the three waypoints, the conditioned diffusion model can generate unseen trajectories that still satisfy constraints supported by the training dataset (Figure 7d), demonstrating the conditioned diffusion model's ability to generalize to out-of-combination conditioning.

8.2 Model Architecture: Attention vs Concatenation

We also ablate over different model architectures: (1) concatenating the latent vector $\mathbf{z}$ with diffusion's time embedding, (2) performing cross attention between $\mathbf{z}$ and output of each Unet residual

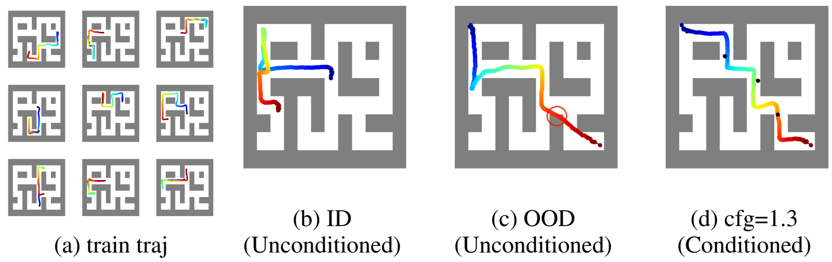


Figure 7: Trajectories generated in Maze2D for large maze. (a) Samples from the training set. (b) Trajectories directly generated by the unconditioned diffusion model given in distribution start and end positions. (c) Trajectories directly generated by the unconditioned diffusion model on unseen start and end positions. (d) Trajectories directly generated by a conditioned diffusion model using 3 waypoints (black dots) as conditioning with classifier-free guidance (cfg) weight 1.3. For results in medium maze please refer to Appendix D.6.1.

block (Architecture in Figure 10). Figure 8 shows our result: in general conditioned diffusion models outperform unconditioned ones and attention outperforms concatenation in 3 out of 4 cases.

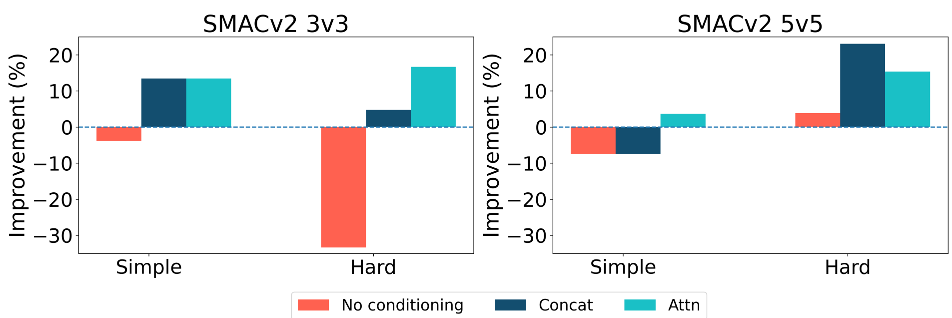


Figure 8: Improvement percentage over MAPPO for different types of conditioning in SMACv2.

Ablation Takeaway: Conditioned diffusion models, provided with information about the new composition of state, can generate better trajectories than unconditioned diffusion models. Also, cross-attention with the condition vector outperforms simply concatenating it with the time-embedding in most cases.

9 Conclusions

Despite the success of traditional RL models in decision-making tasks, they still struggle to generalize to unseen state inputs. Most existing work focuses on RL generalization under the assumption that generalization to a different probability density function with the same support or allows unseen base elements but also introduces other potentially unrealistic assumptions. However, we take it further and study the problem of generalization to out-of-support states, out of combination in particular, hoping the model can exploit the compositional nature of our world. We showed how this task is challenging for value-based RL and also how conditioned diffusion models can generalize to unsupported samples. We compare the models in different environments with detailed ablation and analysis, demonstrating how each of these classic environments can be formulated as a state combinatorial problem.

However, one limitation of our setup is we model combinatorial generalization in state space as a combination of base elements, which is valid for many real-world applications where complexity stems from exponentially many combinations but does not cover all cases. Oftentimes the distinction between different objects can be blurry (e.g. would a motorbike be a bike). Additionally, the model has difficulty with zero-shot generalization to unseen base objects. Another constraint is efficiency, as planning with diffusion models in stochastic environments requires denoising a trajectory at each planning step, which can be computationally intensive.

10 Ethics Statement

Developing data-driven decision-making models carries the risk of generating inappropriate or harmful actions. This work presents a conditioned model that can be manipulated through carefully forged conditioning, potentially leading to malicious actions.

11 Reproducibility

Please refer to Section D.3 for detailed model architecture. Pseudocode can be found in Section D.5. For Maze2D, hyperparameters for training conditioned diffusion model can be found in Section D.6.2. For the Roundabout environment, environment parameters (Section D.7.2), dataset detail (Section D.7.3), baseline details (Section D.8.1), hyperparameters for training conditioned diffusion model (Section D.8.1), and models sizes (Section D.8.2) can be found in appendix D.7. For SMACv2 environment, baseline and data collecting policy (Section D.9.1), hyperparameters for training conditioned diffusion model (Section D.9.1), dataset distribution (Section D.9.2), and detailed success rates (Section D.9.3) can be found in appendix (Section D.9). Model runtime and memory can be found in Section D.10.

References

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*, 2020.

Sumukh K Aithal, Pratyush Maini, Zachary C Lipton, and J Zico Kolter. Understanding hallucinations in diffusion models through mode interpolation. *arXiv preprint arXiv:2406.09358*, 2024.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Georgios Bakirtzis, Michail Savvas, Ruihan Zhao, Sandeep Chinchali, and Ufuk Topcu. Reduce, reuse, recycle: Categories for compositional reinforcement learning. *arXiv preprint arXiv:2408.13376*, 2024.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

Hyungjin Chung, Suhyeon Lee, and Jong Chul Ye. Fast diffusion sampler for inverse problems by geometric decomposition. *arXiv preprint arXiv:2303.05754*, 2023.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, pp. 1282–1289. PMLR, 2019.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2212.07489*, 2022.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Dibya Ghosh, Anurag Ajay, Pulkit Agrawal, and Sergey Levine. Offline rl policies should be trained to be adaptive. In *International Conference on Machine Learning*, pp. 7513–7530. PMLR, 2022.

Jake Grigsby and Yanjun Qi. Measuring visual generalization in continuous control from pixels. *arXiv preprint arXiv:2010.06740*, 2020.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

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*, pp. 1861–1870. PMLR, 2018.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. *Advances in neural information processing systems*, 34: 3680–3693, 2021.

Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36, 2024a.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, and Stefano Ermon. Manifold preserving guided diffusion. In *The Twelfth International Conference on Learning Representations*, 2024b. URL `https://openreview.net/forum?id=o3BxOLoxm1`.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Shiyu Huang, Wentse Chen, Yiwen Sun, Fuqing Bie, and Wei-Wei Tu. Openrl: A unified reinforcement learning framework. *arXiv preprint arXiv:2312.16189*, 2023.

Geonho Hwang, Jaewoong Choi, Hyunsoo Cho, and Myungjoo Kang. Maganet: Achieving combinatorial generalization by modeling a group action. In *International Conference on Machine Learning*, pp. 14237–14248. PMLR, 2023.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Kishor Jothimurugan, Steve Hsu, Osbert Bastani, and Rajeev Alur. Robust subtask learning for compositional generalization. In *International Conference on Machine Learning*, pp. 15371–15387. PMLR, 2023.

Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision, control, and planning. *arXiv preprint arXiv:1902.01378*, 2019.

Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representation. *arXiv preprint arXiv:2310.02557*, 2023.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 267–274, 2002.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.

Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

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.

Jiachen Li, Quan Vuong, Shuang Liu, Minghua Liu, Kamil Ciosek, Henrik Christensen, and Hao Su. Multi-task batch reinforcement learning with metric learning. *Advances in Neural Information Processing Systems*, 33:6197–6210, 2020.

Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, pp. 20035–20064. PMLR, 2023.

Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. *arXiv preprint arXiv:2302.01877*, 2023a.

Zhixuan Liang, Yao Mu, Hengbo Ma, Masayoshi Tomizuka, Mingyu Ding, and Ping Luo. Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution. *arXiv preprint arXiv:2312.11598*, 2023b.

Yixin Lin, Austin S Wang, Eric Undersander, and Akshara Rai. Efficient and interpretable robot manipulation with graph neural networks. *IEEE Robotics and Automation Letters*, 7(2):2740–2747, 2022.

Yuejiang Liu, Alexandre Alahi, Chris Russell, Max Horn, Dominik Zietlow, Bernhard Schölkopf, and Francesco Locatello. Causal triplet: An open challenge for intervention-centric causal representation learning. In *Conference on Causal Learning and Reasoning*, pp. 553–573. PMLR, 2023.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11461–11471, 2022.

Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1711–1724, 2022.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pp. 1162–1176. PMLR, 2020.

Jorge A Mendez, Harm van Seijen, and Eric Eaton. Modular lifelong reinforcement learning via neural composition. *arXiv preprint arXiv:2207.00429*, 2022.

Utkarsh Aashu Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pp. 2905–2925. PMLR, 2023.

Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. In *International Conference on Machine Learning*, pp. 7780–7791. PMLR, 2021.

Milton Llera Montero, Casimir JH Ludwig, Rui Ponte Costa, Gaurav Malhotra, and Jeffrey Bowers. The role of disentanglement in generalisation. In *International Conference on Learning Representations*, 2020.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Fei Ni, Jianye Hao, Yao Mu, Yifu Yuan, Yan Zheng, Bin Wang, and Zhixuan Liang. Metadiffuser: Diffusion model as conditional planner for offline meta-rl. In *International Conference on Machine Learning*, pp. 26087–26105. PMLR, 2023.

Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

Maya Okawa, Ekdeep S Lubana, Robert Dick, and Hidenori Tanaka. Compositional abilities emerge multiplicatively: Exploring diffusion models on a synthetic task. *Advances in Neural Information Processing Systems*, 36, 2024.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, 10, 1997.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. *Advances in neural information processing systems*, 30, 2017.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340. PMLR, 2019.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Lukas Schott, Julius Von Kügelgen, Frederik Träuble, Peter Gehler, Chris Russell, Matthias Bethge, Bernhard Schölkopf, Francesco Locatello, and Wieland Brendel. Visual representation learning does not generalize strongly within the same domain. *arXiv preprint arXiv:2107.08221*, 2021.

John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Yeda Song, Dongwook Lee, and Gunhee Kim. Compositional conservatism: A transductive approach in offline reinforcement learning. *arXiv preprint arXiv:2404.04682*, 2024.

Jan Stanczuk, Georgios Batzolis, Teo Deveney, and Carola-Bibiane Schönlieb. Your diffusion model secretly knows the dimension of the data manifold. *arXiv preprint arXiv:2212.12611*, 2022.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. Compositional generalization from first principles. *Advances in Neural Information Processing Systems*, 36, 2024.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

Baoquan Zhang, Chuyao Luo, Demin Yu, Xutao Li, Huiwei Lin, Yunming Ye, and Bowen Zhang. Metadiff: Meta-learning with conditional diffusion for few-shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16687–16695, 2024.

Linfeng Zhao, Lingzhi Kong, Robin Walters, and Lawson LS Wong. Toward compositional generalization in object-oriented world modeling. In *International Conference on Machine Learning*, pp. 26841–26864. PMLR, 2022.

A RELATED WORK

Meta RL Meta RL is often seen as the problem of "learning to learn", where agents are trained on several environments sampled from a task distribution during meta-training and tested on environments sampled from the same distribution during meta-testing (Yu et al., 2020; Finn et al., 2017). In the K-shot meta-RL setting, the model can interact with the testing environment K times during meta-testing time to update the model using reward (Finn et al., 2017; Mitchell et al., 2021; Li et al., 2020; Rakelly et al., 2019). Our setting is different from Meta-RL as the training and testing environments are sampled from different distributions and conditioning is provided while restricting K to zero.

Attached below is a review of RL environments for generalization.

A.1 GENERALIZATION IN RL

Environments Most RL environments that test model generalization can be grouped into different reward functions (Rajeswaran et al., 2017; Zhang et al., 2018; Rakelly et al., 2019; Finn et al., 2017) or transition functions (Dennis et al., 2020; Machado et al., 2018; Packer et al., 2018; Zhang et al., 2018), goals or tasks (Finn et al., 2017; Yu et al., 2020), states (Nichol et al., 2018; Cobbe et al., 2019; Juliani et al., 2019; Küttler et al., 2020; Grigsby & Qi, 2020; Hansen et al., 2021; Mees et al., 2022; Cobbe et al., 2020). For environments with different state distributions, randomization (Grigsby & Qi, 2020) and procedural generation (Nichol et al., 2018; Küttler et al., 2020; Cobbe et al., 2020) are widely used to generate new states. Some vision-based environments (Juliani et al., 2019; Hansen et al., 2021; Mees et al., 2022) also use different rendering themes or layouts to generate unseen observations, more targeting sim2real problems. For robotics benchmarks like Metaworld (Yu et al., 2020) and RLbench (James et al., 2020), how much structure is shared between tasks like open a door and open a drawer is ambiguous (Ahmed et al., 2020). Also, benchmarks like Franka Kitchen (Gupta et al., 2019) focus on composing tasks at time horizons, requiring the model to concatenate trajectories corresponding to different subtasks. However, despite the large volume of generalization benchmarks, there is no benchmark designed for state combinatorial generalization to our best knowledge.

B PROOF OF COROLLARY 5.1

Corollary B.1 (Corollary 5.1)**.** *Suppose the states lie along a linear manifold $\mathcal{M}$ in the state space $\mathbf{S}$ and the latent space $\mathbf{Z}$ is well constructed so that $\mathbf{Z}$ is (affine) isomorphic to $\mathcal{M}$. Let $\boldsymbol{s}$ be a state in the training set with corresponding latent vector $\boldsymbol{z}$ and $\boldsymbol{s}'$ be an OOC state with corresponding latent vector $\boldsymbol{z}'$. Then a diffusion model p_θ that is well trained on P_{train} can sample $\boldsymbol{s}'$ with non-zero probability.*

Proof. We prove Corollary 5.1 by construction. Suppose $\mathcal{M}$ is a d-dimensional linear manifold and $\mathbf{S} \subset \mathbb{R}^k$, then we note that both $\mathcal{M}$ and $\mathbf{Z}$ are affine isomorphic to $\mathbb{R}^d$. Therefore, with necessary shifting, we can have $\boldsymbol{z}' = \boldsymbol{z} + \boldsymbol{y}$ where $\boldsymbol{y} \in \mathbf{Z}$. Let $\boldsymbol{v}$ be the corresponding vector of $\boldsymbol{y}$ in $\mathcal{M}$.

Now let's perform DDIM inversion on a training sample $\boldsymbol{s}$ to obtain the SDE trajectory $\{\boldsymbol{s}_t\}$. Let γ_t be the angle between $\epsilon_\theta(\boldsymbol{s}_t, t)$ and $\mathcal{M}$, there exist a set of vectors $\boldsymbol{v}_t$ such that $\boldsymbol{v} = \sum_t sin(\gamma_t)\sigma_t \boldsymbol{v}_t$ and $\boldsymbol{v}_t$ perpendicular to $\epsilon_\theta(\boldsymbol{s}_t, t)$. Then by construction and with necessary shifting, the trajectory $\boldsymbol{s}'_t = \boldsymbol{s}_t + \boldsymbol{v}_t$ is a valid diffusion denoising trajectory (with $\boldsymbol{v}_t$ acting as the "random" vector ϵ_t sampled at each time step). This trajectory will yield $\boldsymbol{s}'$ as the final state with non-zero probability because each intermediate Gaussian distribution $p_\theta(\boldsymbol{s}_{t-1}|\boldsymbol{s}_t) = \mathcal{N}(\boldsymbol{s}_{t-1}; \boldsymbol{\mu_\theta}(\boldsymbol{s}_t, t), \boldsymbol{\Sigma_\theta}(\boldsymbol{s}_t, t))$ is defined on the entire ambient space. By Stanczuk et al. (2022) we know that $\gamma_t \to \pi/2$ as $t \to 0$, therefore $\boldsymbol{v}$ can be a non-zero vector. Hence, there exists a $\boldsymbol{v} \in \mathcal{M}$ such that the sampling probability of $\boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{v}$ from diffusion model p_θ is non-zero. □

While we have proven non-zero probability above, one can easily spot that, the probability can become extremely close to zero if the OOC sample is very far away from all training examples due to the intermediate Gaussian distributions. This corresponds to generalizing to the out-of-distribution samples with unseen base elements (the gray area in Figure 1). One can mitigate this problem by increasing the coverage of the support of the training space, which is also a common method in traditional RL to mitigate the problem of generalization to unseen base elements. Applying other post-training sampling techniques like repainting (Lugmayr et al., 2022) can also allow extra Langevin steps when t is large.

C ADDITIONAL VISUALIZATION OF VALUE FUNCTION OF PPO

We include here the value prediction of PPO for in-distribution and OOC states to demonstrate that OOC states are also a problem for online methods.

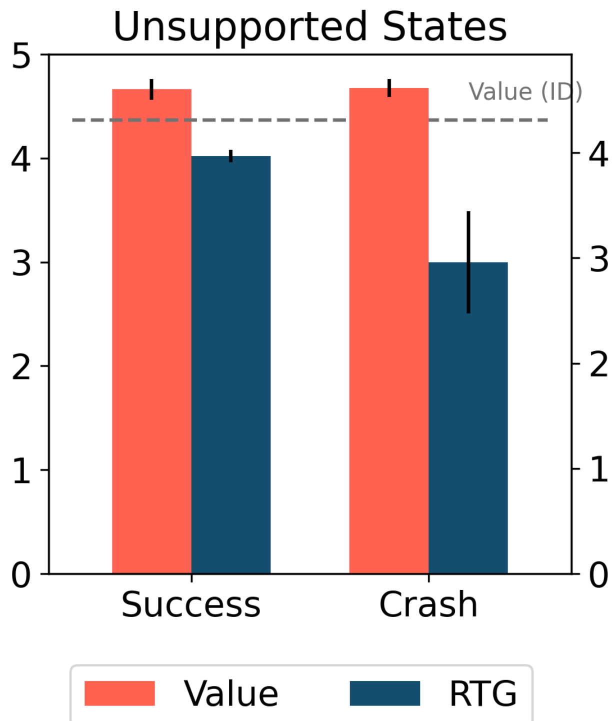


Figure 9: Value prediction of PPO and actual return-to-go (RTG) in unsupported states in Roundabout environment

D EXPERIMENT DETAILS

D.1 HARDWARE AND PLATFORM

Experiments are run on a single NVIDIA RTX A6000 GPUs, with all code implemented in PyTorch.

D.2 STATISTICS

All mean value is obtained by running with three different seeds and calculated with numpy.mean(). All error bar is obtained by numpy.std().

D.3 MODEL ARCHITECTURE

The backbone for Unet is based on Janner et al. (2022). We add cross-attention blocks after each residual block, except for the bottleneck layers. Inputs to the cross-attention blocks are the conditioning embedding and output of the residual block. To ensure local consistency of trajectory, we used 1D convolution along the horizon dimension. To keep the number of parameters for cross attention and the original Unet relatively balanced, we also used 1D convolution as the mapping from input to key, query, and value. Detailed model architecture is shown below in Figure 10.

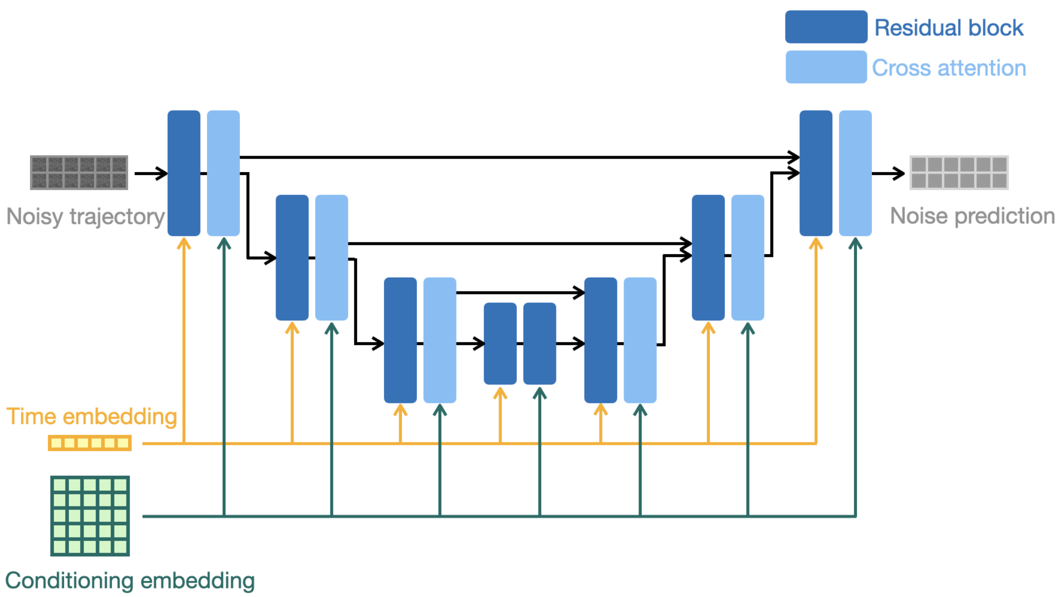


Figure 10: Model architecture.

The number of down-sampling/up-sampling and feature channel sizes is different for each experiment. Detailed parameters can be found in the section for each experiment.

D.4 Trajectory formulation

The trajectory $\boldsymbol{\tau} \in \mathbb{R}^d$ is represented by concatenating the state $\boldsymbol{s_u} \in \mathbb{R}^{d_S}$ and the action $\boldsymbol{a_u} \in \mathbb{R}^{d_A}$ at planning time step u and then horizontally stacking them for all time steps. For example, a trajectory with planning horizon h can be written as $\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{s_1} & \boldsymbol{s_2} \dots & \boldsymbol{s_h} \\ \boldsymbol{a_1} & \boldsymbol{a_2} \dots & \boldsymbol{a_h} \end{bmatrix}$.

D.5 Pseudo-code

Pseudo-code for planning with conditional diffusion model is shown below in Algorithm 1.

Algorithm 1 Planning with Attention-based Composition Conditioned Diffusion Model

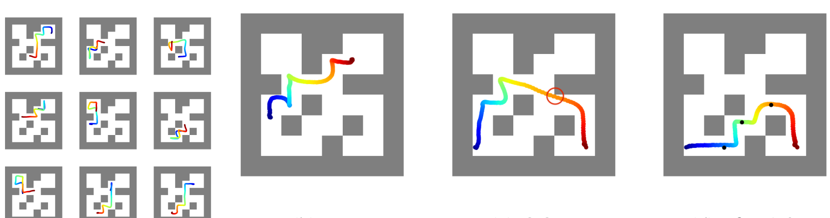
Input: Diffusion model $\boldsymbol{\epsilon_\theta}$, compositional elements extractor r, learnable embedding function h, classifier-free guidance scale λ, state dimentionality d_S, initial observation $\boldsymbol{o}$, environment simulator env

while not done **do**
- Initialize $\boldsymbol{\tau_t} \sim \mathcal{N}(0, I)$
- $\boldsymbol{c} \leftarrow r(\boldsymbol{o})$ ▷ Extract observed compositional information
- $\boldsymbol{z} \leftarrow h(\boldsymbol{c})$ ▷ Obtain element embedding
- **for** $t \leftarrow T, ...1$ **do**
  - $\boldsymbol{\tau}_t[: d_S, 0] \leftarrow \sqrt{\bar{\alpha}_t}\boldsymbol{o} + \sqrt{1-\bar{\alpha}_t}\epsilon$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ ▷ Replace the first observed state with noised $\boldsymbol{o}$
  - $\widetilde{\boldsymbol{\epsilon_t}} = (1+\lambda)\boldsymbol{\epsilon_\theta}(\boldsymbol{s}_t, \boldsymbol{z}, t) - \lambda\boldsymbol{\epsilon_\theta}(\boldsymbol{s}_t, t)$ ▷ Classifier free guidance
  - $\boldsymbol{\tau_{t-1}} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{\tau}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\widetilde{\boldsymbol{\epsilon_t}}\right) + \sigma_t\boldsymbol{\epsilon_t}$, where $\boldsymbol{\epsilon_t} \sim \mathcal{N}(0, I)$
- **end for**
- $\boldsymbol{a} \leftarrow \boldsymbol{\tau_0}[d_S :, 0]$ ▷ Extract action
- $\boldsymbol{o} \leftarrow env.step(\boldsymbol{a})$

end while

D.6 Maze2D

D.6.1 Extra Results



(a) train traj (b) ID (Unconditioned) (c) OOD (Unconditioned) (d) cfg=1.3 (Conditioned)

Figure 11: Trajectories generated in Maze2D for medium maze. (a) are samples from the training set. (b) are trajectories generated by the unconditioned diffusion model given in distribution start and end positions. (c) are generated by the unconditioned diffusion model on unseen start and end positions. (d) are generated by a conditioned diffusion model using 3 waypoints (black dots) as conditioning with classifier-free guidance (cfg) weight 1.3.

D.6.2 Experiment Details

We followed the setup used in Janner et al. (2022). The hyperparameters shared for large and medium mazes are shown below in Table 1. Large maze use a planning horizon of 384 and medium maze use a planning horizon of 256. Conditioning is passed through a positional embedding layer first to map each dimension of the waypoint (x, y, v_x, v_y) to a higher dimension of 21 and concatenate them to form a vector of size $(1, 21 * 4)$. Three waypoints are then stacked together to form a matrix of size $(3, 21 * 4)$ and passed into the cross-attention layer. In our experiment, directly using the waypoints as conditioning was unsuccessful.

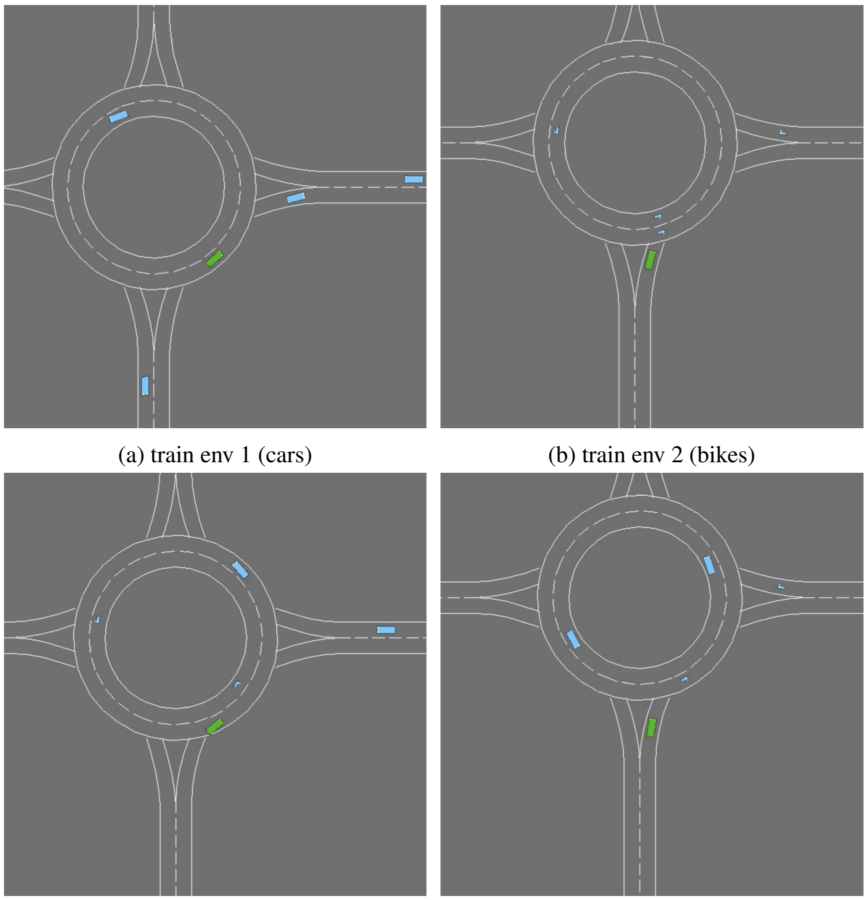
Parameter	**Value**
number of diffusion steps	256
action weight	1
dimension multipliers	(1, 4, 8)
classifier free guidance drop conditioning probability	0.1
steps per epoch	10000
loss type	l2
train steps	2e6
batch size	32
learning rate	2e-4
gradient accumulate every	2
ema decay	0.995

Table 1: Training parameter for diffusion model in Maze2D

D.7 Roundabout

D.7.1 Environment

The training environment consists of all cars or all bicycles and the testing environment is a mixture of traffic (Figure 12).



(a) train env 1 (cars)

(b) train env 2 (bikes)

(c) test env 1

(d) test env 2

Figure 12: Training and testing environments for Roundabout. The green vehicle is the ego agent and the blue ones are controlled by the environment. The large blue box represents a car and the small blue box represents a bicycle.

D.7.2 Environment Parameters

We changed the parameters to create a different type of traffic in the roundabout as shown below in Table 4. Also, since bicycles have slower speeds, we change the initialization position so that each environment vehicle can interact with the ego vehicle.

Parameter	**Car**	**Bicycle**
length	5.0	2.0
width	2.0	1.0
speed	[23, 25]	4
max acceleration	6.0	2.0
comfort max acceleration	3.0	1.0

Table 2: Parameters for car and bicycles in Roundabout environment

D.7.3 DATASET

In order to collect expert trajectories, we train two PPO models separately on the environment with all cars and all bicycles. We then collect 320000 successful trajectories in the training environment. All trajectories have a unified length of 12.

D.8 SETUP FOR ROUNDABOUT ENVIRONMENT

Here we describe how the Roundabout task in this paper conforms to our problem description. In this setting our base object set is $E = \{\text{car}, \text{bicycle}, \text{null}\}$ where null means an object is non-visible. Since the maximum number of objects in the roundabout is five and we fix the ego agent to be a car, support for the training observation is $\{(\text{car (ego agent)}, \text{car}, \text{car}, \text{car}, \text{car}), (\text{car (ego agent)}, \text{bicycle}, \text{bicycle}, \text{bicycle}, \text{bicycle})\}$ and for the testing observation is $\{(\text{car (ego agent)}, \text{bicycle}, \text{bicycle}, \text{car}, \text{car})\}$ assuming no ordering and when the state is fully observable. Since the supports for training and testing are non-overlapping under full observability, they will remain non-overlapping even when some traffic objects are out of sight, unless the ego agent is the only object present in the environment.

D.8.1 EXPERIMENT DETAILS

We use stable_baseline3 Raffin et al. (2021) as the implementation for PPO. The parameter is the default parameter used in the Highway environment Leurent (2018). We increased total timesteps because the environment now has two modalities (all cars and all bicycles) and we observed that PPO takes longer to converge. Detailed parameters for PPO and diffusion are shown below in Table 3 and Table 4.

Parameter	**Value**
policy	MlpPolicy
batch size	64
n_steps	768
n_epochs	10
learning rate	5e-4
gamma	0.8
total timesteps	2e5

Table 3: Training parameter for PPO

Parameter	**Value**
planning horizon	8
number of diffusion steps	80
action weight	10
dimension multipliers	(1, 4, 8)
conditioning embedding size	20
classifier free guidance drop conditioning probability	0.1
classifier free guidance weight	1.0
steps per epoch	10000
loss type	l2
train steps	1e4
batch size	32
learning rate	2e-4
gradient accumulate every	2
ema decay	0.995

Table 4: Training parameter for diffusion model in Roundabout

D.8.2 MODEL SIZE

We include the model size for different algorithms below in Table 5. To eliminate the concern for performance gain due to model size, we include the performance of a large BC model and PPO that has roughly the same number of parameters as the conditioned diffusion model.

	BC	**PPO**	**Diffusion**	**Large BC**	**Large PPO**
Model size	0.30 MB	0.60 MB	54.19 MB	55.65 MB	111.29MB (Policy:55.64+Value:55.64)
Number of parameters	75013	148998	13546370	13912325	27823622 (Policy:13911040 + Value:13911040)
OOD reward	7.50 (0.03)	8.19 (0.16)	8.81 (0.2)	7.71 (0.3)	8.43 (0.19)
OOD crashes	37.7 (0.5)	36.0 (5.7)	20.0 (2.5)	37.3 (4.0)	31.67 (1.89)

Table 5: Model size, number of parameters, and performance for different models.

D.8.3 RELIABLE CONDITIONING

We demonstrate the importance of having reliable information of base element composition, we compare the performance of the conditioned diffusion model given random and ground truth conditionings.

	Ground Truth	**Random**
Number of Crashes	19.67 (2.49)	24.33 (0.47)
Reward	8.81(0.2)	8.1 (0.09)

Table 6: Performance of conditioned diffusion model given ground truth and random conditionings.

D.9 StarCraft

D.9.1 Experiment Details

We use the codebase OpenRL Huang et al. (2023) for the implementation of MAPPO. Detailed parameters for MAPPO can be found in Table 7.

Parameter	Value
learning rate actor	5e-4
learning rate critic	1e-3
data chunk length	8
env num	8
episode length	400
PPO epoch	5
actor train interval step	1
use recurrent policy	True
use adv normalize	True
use value active masks	False
use linear LR decay	True

Table 7: MAPPO hyper-parameters used for SMACv2. We utilize the hyperparameters used in SMACv2 Ellis et al. (2022).

Detailed parameters for training a conditioned diffusion model for 5v5 are shown below in Table 8 and 3v3 in Table 9.

Parameter	Value
planning horizon	40
number of diffusion steps	256
action weight	1
dimension multipliers	(1, 4, 8)
conditioning embedding size	40
classifier free guidance drop conditioning probability	0.1
classifier free guidance weight	[0.7, 1.0, 1.3, 1.5]
steps per epoch	10000
loss type	l2
train steps	2e6
batch size	32
learning rate	2e-4
gradient accumulate every	2
ema decay	0.995

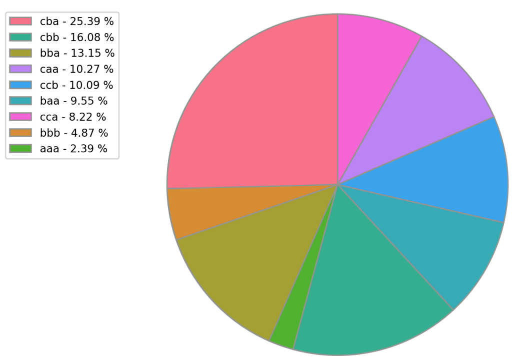
Table 8: Training parameter for diffusion model in StarCraft for 5v5

Parameter	Value
planning horizon	32
number of diffusion steps	256
action weight	1
dimension multipliers	(1, 4, 8)
conditioning embedding size	40
classifier free guidance drop conditioning probability	0.1
classifier free guidance weight	[0.7, 1.0, 1.3, 1.5]
steps per epoch	10000
loss type	l2
train steps	2e6
batch size	32
learning rate	2e-4
gradient accumulate every	2
ema decay	0.995

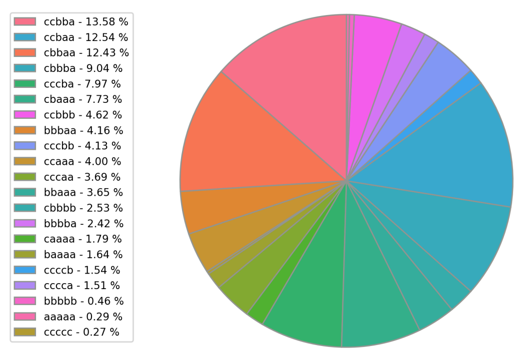
Table 9: Training parameter for diffusion model in StarCraft for 3v3

D.9.2 Dataset Initial State Distribution

The probability of generating each unit type in SMACv2 is imbalanced. Specifically, the probability for Stalker, Zealot, and Colossus is 0.45, 0.45, and 0.1 respectively. The initial state distribution of training trajectories collected by MAPPO for random combination is shown below in Figure 13a and 13b. Since we only keep the successful trajectories and use them as expert data, the distribution depends on the generation probability and MAPPO success rate for different team combinations. A total number of 240000 trajectories were used to train the diffusion model. Since diffusion is trained on local observations and actions of all MAPPO actors, the total number of training samples is 5*240000 for 5v5 and 3*240000 for 3v3.



(a) Distribution of initial state for 3v3 simple scenario



(b) Distribution of initial state for 5v5 simple scenario

D.9.3 DETAILED RESULTS ON SMACV2

Table 10 and 11 show the detailed performance of different algorithms in the 3v3 and 5v5 scenarios, respectively.

Env: 3v3	RL		Imitation Learning	
	2 PPO + 1 Rand	3 PPO	BC	2 PPO + 1 Diffusion
$ABC \to ABC$ (ID)	0.18 (0.01)	0.58 (0.02)	0.58 (0.07)	**0.59 (0.04)**
$ABC \to AAA$ (Simple)	0.07 (0.03)	0.52 (0.03)	0.56 (0.02))	**0.59 (0.02)**
$AAA \to AAA$ (ID)	0.09 (0.04)	**0.63 (0.02)**	0.6 (0.02)	0.61 (0.05)
$AAA \to ABC$ (Hard)	0.11 (0.02)	0.42 (0.02)	0.4 (0.06)	**0.49(0.02)**

Table 10: Success rate of each agent in 100 rounds. The first two rows correspond to the simple setting of generalization to states with different support and the last two rows correspond to non-overlapping support. Numbers in the parenthesis represent the standard error over 3 seeds. The best performing method is labeled bold. The 2 PPO + 1 Rand column shows the effect of replacing one PPO trained agent with a random agent as a baseline for comparison against the 2 PPO + 1 Diffusion case.

Env: 5v5	RL		Imitation Learning	
	4 PPO + 1 Rand	5 PPO	BC	4 PPO + 1 Diffusion
$ABC \to ABC$ (ID)	0.22 (0.04)	0.64 (0.05)	0.56 (0.05)	**0.66 (0.01)**
$ABC \to AAA$ (Simple)	0.11 (0.03)	0.54 (0.04)	0.52 (0.05)	**0.56 (0.02)**
$AAA \to AAA$ (ID)	0.14 (0.02)	**0.58 (0.04)**	0.54 (0.04)	0.55 (0.03)
$AAA \to ABC$ (Hard)	0.11 (0.02)	0.26 (0.05)	0.28 (0.04)	**0.32 (0.04)**

Table 11: Success rate of each agent in 100 rounds. The first two rows correspond to the simple setting of generalization to states with different support and the last two rows correspond to non-overlapping support. Numbers in the parenthesis represent the standard error over 3 seeds. The best performing method is labeled bold. The 4 PPO + 1 Rand column shows the effect of replacing one PPO trained agent with a random agent as a baseline for comparison against the 4 PPO + 1 Diffusion case.

D.9.4 DETAILED RESULTS FOR ABLATION

The ablation result for 3v3 and 5v5 scenarios are shown below in Table 12 and Table 13. The first column is the success rate without conditioning (No Cond). The second column represents concatenating the conditioning with time embedding (Concat). The last column represents passing conditioning as another input beside the trajectory to the cross-attention block (Attn).

Table 12: Ablation for Diffusion on 3v3

Env 3v3	2 PPO + 1 Diffusion		
	No Cond	Concat	Attn
$ABC \to ABC$ (ID)	0.55±0.03	0.59±0.04	0.59±0.05
$ABC \to AAA$ (Simple)	0.5±0.06	0.59±0.02	0.59±0.02
$AAA \to AAA$ (ID)	0.4±0.03	0.64±0.03	0.61±0.05
$AAA \to ABC$ (Hard)	0.28±0.03	0.44±0.05	0.49±0.02

D.9.5 2v2

The success rates for StarCraft 2v2 are shown below in Table 14. We can see that out-of-combination cases did not cause the performance to drop drastically for MAPPO. This is because the number of

Table 13: Ablation for Diffusion on 5v5

Env 5v5	4PPO + 1 Diffusion		
	No Cond	Concat	Attn
$ABC \rightarrow ABC$ (ID)	0.53±0.04	0.59±0.03	0.66±0.01
$ABC \rightarrow AAA$ (Simple)	0.50±0.03	0.50±0.01	0.56±0.02
$AAA \rightarrow AAA$ (ID)	0.47±0.08	0.55±0.03	0.58±0.04
$AAA \rightarrow ABC$ (Hard)	0.27±0.03	0.32±0.04	0.30±0.04

combinations in 2v2 is very limited (e.g. aa, bb, ab), and if one agent dies, MAPPO has encountered scenarios of playing with each unit type individually, therefore falling back to in distribution state again. This scenario also exists for 5v5 and 3v3 but only at the end of each game when only one agent is left.

Table 14: SMAC II success rate for 2v2

Env	BC	MAPPO		Diffusion		
	BC	1 PPO + 1 Rand	5 PPO	No Cond	Concat	Attn
$ABC \rightarrow ABC$ (ID)	0.54±0.02	0.06±0.02	**0.62±0.05**	0.43±0.04	0.56±0.03	0.57±0.067
$ABC \rightarrow AAA$ (Simple)	0.47±0.02	0.02±0.01	**0.57±0.02**	0.44±0.02	0.55±0.06	0.49±0.06
$AAA \rightarrow AAA$ (ID)	0.57±0.01	0.01±0.01	**0.64±0**	0.38±0.02	**0.63±0.04**	**0.63±0.02**
$AAA \rightarrow ABC$ (Hard)	0.4±0.03	0.04±0.02	**0.44±0.02**	0.29±0.02	**0.43±0.08**	0.41±0.06

D.9.6 More Rendering of States Predicted by the Diffusion model

More rendering of the future states predicted by the diffusion model is shown in Figure 14.

Figure 14: Rendering of future states predicted by the diffusion model given different conditionings. The grey box is the initial state. Yellow boxes are conditioned on the type of unit in the initial state. Green boxes are conditioned on all Triangles. Smaller sizes represent less shield or health.

D.10 MODEL RUNTIME AND GPU MEMORY

We include the training time and GPU memory used for the conditioned diffusion model below in Table 15 and 16.

Training Time	**Roundabout**	**SMACv2 2v2**	**SMACv2 3v3**	**SMACv2 5v5**
PPO	0.5h	9h	9h	9h
Diffusion	1h	48h	70h	98h

Table 15: Training time for PPO and conditioned diffusion model in different environments.

GPU Memory	**Roundabout**	**SMACv2 2v2**	**SMACv2 3v3**	**SMACv2 5v5**
Diffusion	542 MiB	1004 MiB	2892 MiB	4096 MiB

Table 16: GPU Memory for training conditioned diffusion model in different environments.

D.11 PARAMETER COMPARISON WITH CONCATENATION OR ATTENTION

We demonstrate the number of parameters in attention-based conditioning and concatenation-based conditioning to eliminate the concern regarding performance gain due to more parameters. Attention or concatenation has roughly the same number of parameters as the attention module is convolutional layers and concatenation increases the parameters of conditioning layer.

	Attention	**Concatenation**
Model size	617.06 MB	619.64 MB
Parameters	154264085	154911187

Table 17: Number of parameters in attention-based conditioning and concatenation-based conditioning.

D.12 SUBSTITUTING MORE MAPPO AGENTS WITH DIFFUSION AGENTS

We would like to ask the question of what about replacing more than one MAPPO agent with diffusion model. Figure 15 shows that the number of diffusion models does not have a positive correlation with the success rate. This is because MAPPO can learn a collaborative policy between actors and simply adding more ego-centric diffusion models will break the coordination between actions. Also, since the diffusion model is trained to play with all PPOs teammates, replacing other PPO actions with actions generated by diffusion models will cause a distribution shift that is hard to quantify.

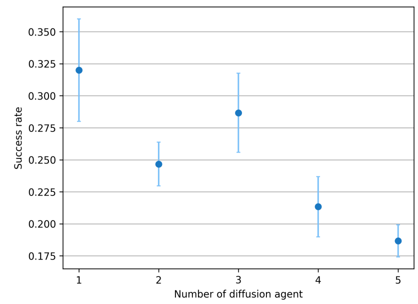


Figure 15: Success rate vs number of agents in SMACv2 5v5 hard scenario that are replaced with diffusion agents. Replacing more than one MAPPO agent with diffusion agents hurts performance.