EDGI: EQUIVARIANT DIFFUSION FOR PLANNING WITH EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Embodied agents operate in a structured world, often solving tasks with spatial, temporal, and permutation symmetries. Most algorithms for planning and model-based reinforcement learning (MBRL) do not take this rich geometric structure into account, leading to sample inefficiency and poor generalization. We introduce the Equivariant Diffuser for Generating Interactions (EDGI), an algorithm for MBRL and planning that is equivariant with respect to the product of the spatial symmetry group $\mathrm{SE}(3)$, the discrete-time translation group $\mathbb{Z}$, and the object permutation group $\mathbb{S}_n$. EDGI follows the Diffuser framework Janner et al. (2022) in treating both learning a world model and planning in it as a conditional generative modeling problem, training a diffusion model on an offline trajectory dataset. We introduce a new $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$-equivariant diffusion model that supports multiple representations. We integrate this model in a planning loop, where conditioning and classifier-based guidance allow us to softly break the symmetry for specific tasks as needed. On navigation and object manipulation tasks, EDGI improves sample efficiency and generalization.

1 INTRODUCTION

Our world is awash with symmetries. The laws of physics are the same everywhere in space and time—they are symmetric under translations and rotations of spatial coordinates as well as under time shifts.[1] In addition, whenever multiple identical or equivalent objects are labeled with numbers, the system is symmetric with respect to a permutation of the labels. Embodied agents are exposed to this structure, and many common robotic tasks exhibit spatial, temporal, or permutation symmetries. The gaits of a quadruped are independent of whether it is moving East or North, and a robotic gripper would interact with multiple identical objects independently of their labeling. However, most reinforcement learning (RL) and planning algorithms do not take this rich structure into account. While they have achieved remarkable success on well-defined problems after sufficient training, they are often sample-inefficient (Holland et al., 2018) and lack robustness to changes in the environment.

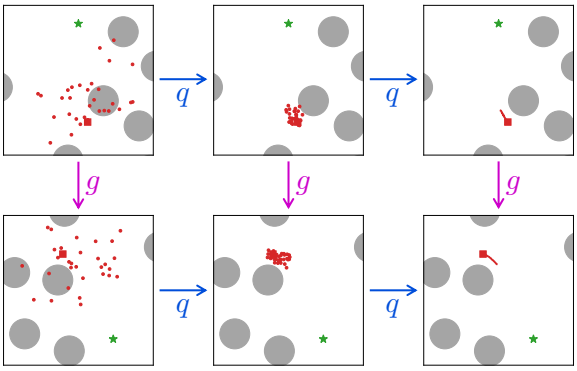

Figure 1: Schematic of EDGI in a navigation task, where the agent (red square) plans the next actions (red dots) to reach the goal (green star) without touching obstacles (grey circles). **Top**: planning as conditional sampling from a diffusion model. **Bottom**: effect of a group action. Equivariance requires the diagram to commute.

To improve the sample efficiency and robustness of RL algorithms, we believe it is paramount to develop them with an awareness of their symmetries. Such algorithms should satisfy two key desiderata. First, policy and world models should be equivariant with respect to the relevant symmetry

[1]This is true in the approximately flat spacetime on Earth, as long as all velocities are much smaller than the speed of light. A machine learning researcher who finds herself close to a black hole may disagree.

group. Often, for embodied agents this will be a subgroup of the product group of the spatial symmetry group $\mathrm{SE}(3)$, the group of discrete time shifts $\mathbb{Z}$, and one or multiple object permutation groups $\mathbb{S}_n$. Second, it should be possible to softly break (parts of) the symmetry group to solve concrete tasks. For example, a robotic gripper might be tasked with moving an object to a specific point in space, which breaks the symmetry group $\mathrm{SE}(3)$. First works on equivariant RL have demonstrated the potential benefits of this approach (van der Pol et al., 2020; Walters et al., 2020; Mondal et al., 2021; Muglich et al., 2022; Wang and Walters, 2022; Wang et al., 2022; Cetin et al., 2022; Rezaei-Shoshtari et al., 2022; Deac et al., 2023). However, these works generally only consider small finite symmetry groups such as C_n and do not usually allow for soft symmetry breaking at test time.

In this paper, we introduce the *Equivariant Diffuser for Generating Interactions* (EDGI), an equivariant algorithm for model-based reinforcement learning and planning. EDGI consists of a base component that is equivariant with respect to the full product group $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$ and supports the multiple different representations of this group we expect to encounter in embodied environments. Moreover, EDGI allows for a flexible soft breaking of the symmetry at test time depending on the task.

Our work builds on the Diffuser method by Janner et al. (2022), who approach both the learning of a dynamics model and planning within it as a generative modeling problem. The key idea in Diffuser is to train a diffusion model on an offline dataset of state-action trajectories. To plan with this model, one samples from it conditionally on the current state, using classifier guidance to maximize the reward.

Our main contribution is a new diffusion model that is equivariant with respect to the product group $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$ of spatial, temporal, and permutation symmetries and supports data consisting of multiple representations. We introduce a new way of embedding multiple input representations into a single internal representation, as well as novel temporal, object, and permutation layers that act on the individual symmetries. When integrated into a planning algorithm, our approach allows for a soft breaking of the symmetry group through test-time task specifications both through conditioning and classifier guidance. We demonstrate EDGI in 3D navigation and robotic object manipulation environments. Compared to non-equivariant baselines, we find a performance improvement in the low-data regime as well as increased robustness to symmetry transformations of the environment.

2 Background and Preliminaries

Equivariant deep learning. Equivariant networks directly encode the symmetries described by a group G in their architecture. For the purposes of this paper, we are interested in the symmetries of 3D space, which include translations and rotations and are described by the special Euclidean group $\mathrm{SE}(3)$, discrete-time translations $\mathbb{Z}$, and object permutations, which are defined using the symmetric group of n elements $\mathbb{S}_n$. We now recall the main definition of equivariance and invariance.

Definition 1. A function $f : \mathcal{X} \to \mathcal{Y}$ is equivariant with respect to a group G if it respects the action, i.e. $g \cdot f(x) = f(g \cdot x), \forall g \in G,$ and $x \in \mathcal{X}$. A map $h : \mathcal{X} \to \mathcal{Y}$ is called G-invariant, if $h(x) = h(g \cdot x), \forall g \in G$ and $x \in \mathcal{X}$.

Here $\cdot$ is the binary operation of the group and $\mathcal{X}$ and $\mathcal{Y}$ are input and output spaces that carry a G-action. In machine learning contexts, equivariance plays an important role in modeling whenever we have a group action on the input space and we want to "preserve" this symmetry through the network and in the output. A linear group action is given by a group representation $\rho : G \to \mathrm{GL}(\mathbb{R}^n)$, which itself is a group homomorphism and satisfies $\rho(g_1 g_2) = \rho(g_1)\rho(g_2)$ as $\mathrm{GL}(\mathbb{R}^n)$ is the group of $n \times n$ invertible matrices with group operation being ordinary matrix multiplication.

For generative modeling, we seek to model G-invariant densities. As proven in (Köhler et al., 2020; Bose and Kobyzev, 2021; Papamakarios et al., 2021), given a G-invariant prior density it is sufficient to construct a G-equivariant map to reach the desired G-invariant target density. In Sec. 3, we design G-equivariant diffusion architectures to model a distribution of trajectories that are known to be symmetric with respect to the product group $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$.

Diffusion models. Diffusion models (Sohl-Dickstein et al., 2015) are latent variable models that generate data by iteratively inverting a diffusion process. This diffusion process starts from a clean data sample $x \sim q(x_0)$ and progressively injects noise for $i \in [T]$ steps until the distribution is pure noise. The reverse, generative process takes a sample from a noise distribution and denoises it by progressively adding back structure, until we return to a sample that resembles being drawn from the empirical data distribution $p(x)$. In diffusion models, it is customary to choose a parameter-free

diffusion process (e. g. Gaussian noise with fixed variance). Specifically, define $q(x_t|x_{t-1})$ as the forward diffusion distribution modeled as a Gaussian centered around the sample at timestep x_{t-1}:

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

where β_t is a known variance schedule. The reverse generative process is learnable and can be parametrized using another distribution $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I)$, and the constraint that the terminal marginal at time T is a standard Gaussian—i.e. $p(x_T) = \mathcal{N}(0, I)$. The generative process can be learned by maximizing a variational lower bound on the marginal likelihood,

$$\log p(x_0) \geq \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log\left(p(x_T)\prod_{t=1}^{T}\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right)\right]\,. \tag{2}$$

In practice, instead of predicting the mean of the noisy data, it is convenient to predict the noise level ϵ_t directly Ho et al. (2020). Furthermore, to perform low-temperature sampling in diffusion models it is possible to leverage a pretrained classifier to guide the generation process (Dhariwal and Nichol, 2021). To do so we can modify the diffusion score by including the gradient of the log likelihood of the classifier $\bar{\epsilon}_\theta(x_t, t, c) = \epsilon_\theta(x_t, t) - \lambda\sigma_t\nabla_{x_t}\log p(c|x_t)$, where λ is the guidance weight and c is the label.

Trajectory optimization with diffusion. We are interested in modeling systems that are governed by discrete-time dynamics of a state $s_{h+1} = f(s_h, a_h)$, given the state s_h and action a_h taken at timestep h. The goal in trajectory optimization is then to find a sequence of actions $\mathbf{a}^*_{0:H}$ that maximizes an objective (reward) $\mathcal{J}$ which factorizes over per-timestep rewards $r(s_h, a_h)$. Formally, this corresponds to the following optimization problem:

$$\mathbf{a}^*_{0:H} = \arg\max_{a_{0:H}} \mathcal{J}(s_0, a_{0:H}) = \arg\max_{a_{0:H}} \sum_{h=0}^{H} r(s_h, a_h), \tag{3}$$

where H is the planning horizon and $\tau = (s_0, a_0, \ldots, s_H, a_H)$ denotes the trajectory.

A practical method to solve the trajectory optimization problem in Eq. (3) is to unify the problem of learning a model of the state transition dynamics and the problem of planning with this model into a single generative modeling problem. Janner et al. (2022) propose to train a diffusion model on offline trajectory data consisting of state-action pairs, learning a density $p_\theta(\tau)$. Planning can then be phrased as a conditional sampling problem: finding the distribution

$$\tilde{p}_\theta(\tau) \propto p_\theta(\tau)c(\tau), \tag{4}$$

over trajectories τ where $c(\tau)$ encodes constraints on the trajectories and specifies the task for instance as a reward function. Diffusion models allow conditioning in a way similar to inpainting in generative image modeling, and reward maximization in analogy to classifier-based guidance.

3 Equivariant Diffuser for Generating Interactions (EDGI)

We now describe our EDGI method. We begin by discussing the symmetry group $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$ and common representations in robotic problems. In Sec. 3.2 we introduce our key novelty, an $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$-equivariant diffusion model for state-action trajectories τ. Finally, we show how a diffusion model trained on offline trajectory data can be used for planning in Sec. 3.3.

3.1 Symmetry and Representations

Symmetry group. We consider the symmetry group $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$, which is a product of three distinct groups: 1. the group of spatial translations and rotations $\mathrm{SE}(3)$, 2. the discrete time translation symmetry $\mathbb{Z}$, and 3. the permutation group over n objects $\mathbb{S}_n$. It is important to note, however, that this symmetry group may be softly broken in an environment. For instance, the direction of gravity usually breaks the spatial symmetry group $\mathrm{SE}(3)$ to the smaller group $\mathrm{SE}(2)$, and distinguishable objects in a scene may break permutation invariance. We follow the philosophy of modeling invariance with respect to the larger group and including any symmetry-breaking effects as inputs to the networks.

We require that spatial positions are always expressed relative to a reference point, for example, the robot base or center of mass. This guarantees equivariance with respect to spatial translations: to achieve $\mathrm{SE}(3)$ equivariance, we only need to design an $\mathrm{SO}(3)$-equivariant architecture.

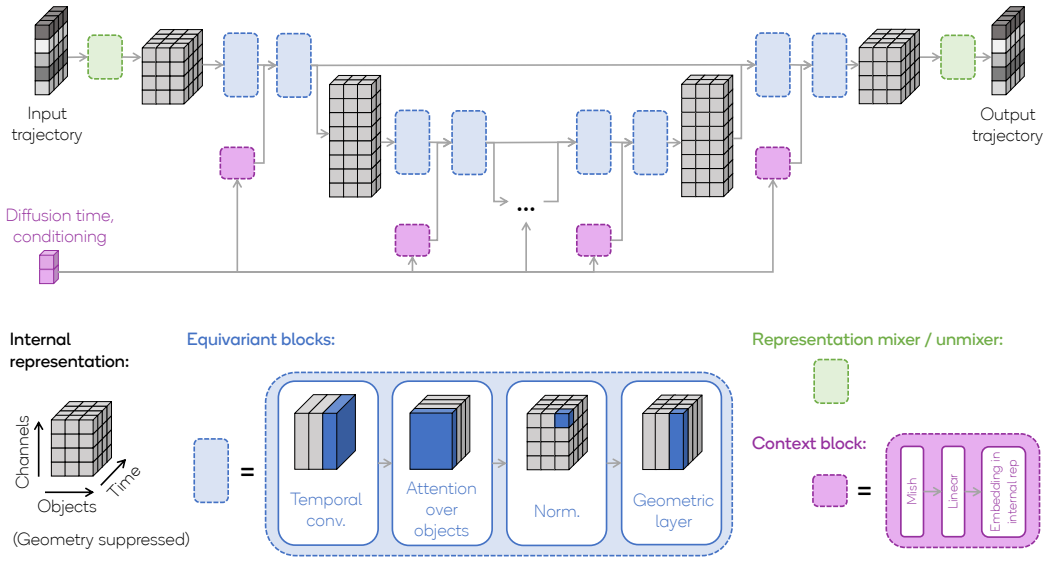

Figure 2: Architecture of our $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$-equivariant denoising network. Input trajectories (top left), which consist of features in different representations of the symmetry group, are first transformed into a single internal representation (green block). The data are then processed with equivariant blocks (blue), which consist of convolutional layers along the time dimension, attention over objects, normalization layers, and geometric layers, which mix scalar and vector components of the internal representations. These blocks are combined into a U-net architecture. For simplicity, we leave out many details, including residual connections, downsampling, and upsampling layers; see Appendix A.

Data representations. We consider 3D environments that contain an embodied agent as well as n other objects. We parameterize their degrees of freedom with two $\mathrm{SO}(3)$ representations, namely the scalar representation ρ_0 and the vector representation ρ_1. Any $\mathrm{SE}(3)$ pose can be transformed to these two representations, see Appendix A for details. We assume that all trajectories transform under the regular representation of the time translation group $\mathbb{Z}$ (similar to how images transform under spatial translations). Under $\mathbb{S}_n$, object properties permute, while robot properties or global properties of the state remain invariant. Each feature is thus either in the trivial or the standard representation of $\mathbb{S}_n$.

Overall, we thus expect that data in environments experienced by our embodied agent to be categorized into four representations of the symmetry group $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$: scalar object properties, vector object properties, scalar robotic degrees of freedom (or other global properties of the system), and vector robotic degrees of freedom (again including other global properties of the system).

3.2 Equivariant diffusion model

Our main contribution is a novel $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$-equivariant diffusion model which leads to an *invariant* distribution over trajectories. Specifically, given an invariant base density with respect to our chosen symmetry group—a Gaussian satisfies this property for $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$—and an equivariant denoising model with respect to the same group we arrive at a diffusion model that is $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$-invariant (Köhler et al., 2020; Papamakarios et al., 2021). Under mild assumptions, such an equivariant map that pushes forward the base density always exists (Bose and Kobyzev, 2021).

We design a novel equivariant architecture for the denoising model f. Implemented as a neural network, it maps noisy input trajectories τ and a diffusion time step i to an estimate $\hat{\epsilon}$ of the noise vector that generated the input. Our architecture does this in three steps. First, the input trajectory consisting of various representations is transformed into an internal representation of the symmetry group. Second, in this representation the data are processed with an equivariant network. Finally, the outputs are transformed from the internal representation into the original data representations present in the trajectory. We illustrate the architecture of our EDGI model in Fig. 2.

Step 1: Representation mixer. The input noisy trajectory consists of features in different represen-

tations of the symmetry group (see above). While it is possible to mirror these input representations for the hidden states of the neural network, the design of equivariant architectures is substantially simplified if all inputs and outputs transform under a single representation. Hence, we decouple the data representation from the representation used internally for the computation—in a similar fashion to graph neural networks that decouple the data and computation graphs.

Internal representation. We define a single internal representation that for each trajectory time step $t \in [H]$, each object $o \in [n]$, each channel $c \in [n_c]$ consists of an[2] $\mathrm{SO}(3)$ scalar s_{toc} and an $\mathrm{SO}(3)$ vector v_{toc}. We write $w_{toc} = (s_{toc}, v_{toc}) \in \mathbb{R}^4$. Under spatial rotations $g \in \mathrm{SO}(3)$, these features thus transform as the direct sum of the scalar and vector representations $\rho_0 \oplus \rho_1$:

$$w_{toc} = \begin{pmatrix} s_{toc} \\ v_{toc} \end{pmatrix} \to w'_{toc} = \begin{pmatrix} \rho_0(g) s_{toc} \\ \rho_1(g) v_{toc} \end{pmatrix} . \tag{5}$$

These internal features transform in the regular representation under time shift and in the standard representation under permutations $\mathbb{P}$ as $w_{toc} \to w_{to'c} = \sum_o \mathbb{P}_{o'o} \, w_{toc}$. There are thus no global (not object-specific) properties in our internal representations.

Transforming input representations into internal representations. The first layer in our network transforms the input τ, which consists of features in different representations of $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$, into the internal representation. On the one hand, we pair up $\mathrm{SO}(3)$ scalars and $\mathrm{SO}(3)$ vectors into $\rho_0 \oplus \rho_1$ features. On the other hand, we get rid of global features – those unassigned to one of the n objects in the scene – by including them in the representation of each of the n objects.

Concretely, for each object $o \in [n]$, each trajectory step $t \in [H]$, and each channel $c = [n_c]$, we define the input in the internal representation as $w_{toc} \in \mathbb{R}^4$ as follows:

$$w_{toc} = \begin{pmatrix} \sum_{c'} \mathbf{W}^1_{occ'} s_{toc'} \\ \sum_{c'} \mathbf{W}^2_{occ'} v_{toc'} \end{pmatrix} + \begin{pmatrix} \sum_{c'} \mathbf{W}^3_{occ'} s_{t\emptyset c'} \\ \sum_{c'} \mathbf{W}^4_{occ'} v_{t\emptyset c'} \end{pmatrix} . \tag{6}$$

The matrices $\mathbf{W}^{1,2,3,4}$ are learnable and of dimension $n \times n_c \times n_s^{\text{object}}$, $n \times n_c \times n_v^{\text{object}}$, $n \times n_c \times n_s^{\text{global}}$, or $n \times n_c \times n_v^{\text{global}}$, respectively. Here n_s^{object} is the number of $\mathrm{SO}(3)$ scalar quantities associated with each object in the trajectory, n_v^{object} is the number of $\mathrm{SO}(3)$ vector quantities associated with each object, n_s^{global} is the number of scalar quantities associated with the robot or global properties of the system, and n_v^{global} is the number of vectors of that nature. The number of input channels n_c is a hyperparameter. We initialize the matrices $\mathbf{W}^i$ such that Eq. (6) corresponds to a concatenation of all object-specific and global features along the channel axis at the beginning of training.

Step 2: $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$-equivariant U-net. We then process the data with a $\mathrm{SO}(3) \times \mathbb{Z} \times \mathbb{S}_n$-equivariant denoising network. Its key components are three alternating types of layers. Each type acts on the representation dimension of one of the three symmetry groups while leaving the other two invariant—i. e. they do not mix internal representation types of the other two layers:

- *Temporal layers*: Time-translation-equivariant convolutions along the temporal direction (i. e. along trajectory steps), organized in a U-Net architecture.
- *Object layers*: Permutation-equivariant self-attention layers over the object dimension.
- *Geometric layers*: $\mathrm{SO}(3)$-equivariant interaction between the scalar and vector features.

In addition, we use residual connections, a new type of normalization layer that does not break equivariance, and context blocks that process conditioning information and embed it in the internal representation (see Appendix A for more details). These layers are combined into an equivariant block consisting of one instance of each layer, and the equivariant blocks are arranged in a U-net, as depicted in Fig. 2. Between the levels of the U-net, we downsample (upsample) along the trajectory time dimension by factors of two, increasing (decreasing) the number of channels correspondingly.

Temporal layers. Temporal layers consist of 1D convolutions along the trajectory time dimension. To preserve $\mathrm{SO}(3)$ equivariance, these convolutions do not add any bias and there is no mixing of features associated with different objects nor the four geometric features of the internal $\mathrm{SO}(3)$ representation.

Permutation layers. Permutation layers enable features associated with different objects to interact

[2]Pairing up just one scalar and one vector is a design choice; for systems in which scalar or vectorial quantities play a larger role, it may be beneficial to use multiple copies of either representation here.

via an equivariant multi-head self-attention layer. Here, there is no mixing between features associated with different time steps, nor between the four geometric features of the internal $\mathrm{SO}(3)$ representation. This is $\mathrm{SO}(3)$-equivariant, as the attention weights compute invariant $\mathrm{SO}(3)$ norms.

Geometric layers. Geometric layers enable mixing between the scalar and vector quantities that are combined in the internal representation but do not mix between different objects or across the time dimension. We construct an expressive equivariant map between scalar and vector inputs and outputs following Villar et al. (2021): We first separate the inputs into $\mathrm{SO}(3)$ scalar and vector components, $w_{toc} = (s_{toc}, v_{toc})^T$. We then construct a complete set of $\mathrm{SO}(3)$ invariants by combining the scalars and pairwise inner products between the vectors, $S_{to} = \{s_{toc}\}_c \cup \{v_{toc} \cdot v_{toc'}\}_{c,c'}$. These are then used as inputs to two MLPs ϕ and ψ, and finally we get output scalars and vectors, $w'_{toc} = (\phi(S_{to})_c, \sum_{c'} \phi(S_{to})_{cc'} v_{toc'})$. Villar et al. (2021) show that this approach can approximate any equivariant map between $\mathrm{SO}(3)$ scalars and vectors under mild assumptions. In its original form, however, it can become prohibitively expensive, as the number of $\mathrm{SO}(3)$ invariants S_{to} scales quadratically with the number of channels. Thus, we first linearly map the input vectors into a smaller number of vectors, apply this transformation, and increase the number of channels again with another linear map.

Step 3: Representation unmixer. The equivariant network outputs internal representations w_{toc} that are transformed back to data representations using linear maps, in analogy to Eq. (6). Global properties, e. g. robotic degrees of freedom, are aggregated from the object-specific internal representations by taking the mean, minimum, and maximum across the objects. These three aggregates are then concatenated along the channel dimension. We find it beneficial to apply an additional geometric layer to these aggregated global features before separating them into the original representations.

We train EDGI by optimizing for a simplified variational lower bound (Ho et al., 2020) on offline trajectories without any reward information. In the section below we outline planning using EDGI.

3.3 Planning with equivariant diffusion

A diffusion model trained on offline trajectory data jointly learns a world model and a policy. Following Janner et al. (2022), we use it to solve planning problems by choosing a sequence of actions to maximize the expected task rewards.

Planning as diffusion. To do this, we use three features of diffusion models. The first is the ability to sample from them by drawing noisy trajectory data from the base distribution and iteratively denoising them with the learned network yielding trajectories similar to those in the training set. For such sampled trajectories to be useful for planning, they need to begin in the current state of the environment. We achieve this by conditioning the sampling process such that the initial state of the generated trajectories matches the current state, in analogy to inpainting. Finally, we can guide this sampling procedure toward solving concrete tasks specified at test time using classifier-based guidance where a regression model is trained offline to map trajectories to task rewards.

Symmetry breaking. By construction, our equivariant diffusion model learns a $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$-invariant density over trajectories. Unconditional samples will reflect this symmetry property—it will be equally likely to sample a trajectory and its rotated or permuted counterpart. However, concrete tasks will often break this invariance, for instance by requiring that a robot or object is brought into a particular location. Our EDGI approach allows us to elegantly break the symmetry at test time for concrete tasks. Such a soft symmetry breaking can happen through conditioning, for instance by specifying the initial or final state of the sampled trajectories, or through a non-invariant reward model used for guidance during sampling.

4 Experiments

We demonstrate the effectiveness of incorporating symmetries as a powerful inductive bias in the Diffuser algorithm with experiments in two environments. The first environment is a 3D navigation task, in which an agent needs to navigate a number of obstacles to reach a goal state. Rewards are awarded based on the distance to the goal at each step, with penalties for collisions with obstacles. The position of the obstacles and the goal state are different in each episode and part of the observation. For simplicity, the actions directly control the acceleration of the agent and we use identical spherical obstacles. Please see Fig. 1 for a schematic representation of this task and Appendix B for more details and the reward structure for this task.

	Standard setting				SO(3) generalization	
Environment	BCQ	CQL	Diffuser	EDGI (ours)	Diffuser	EDGI (ours)
Navigation	–	–	$-32.4_{\pm 5.7}$	$\mathbf{19.1}_{\pm 3.8}$	$-29.3_{\pm 6.8}$	$\mathbf{30.8}_{\pm 8.9}$
Unconditional	0.0	24.4	$\mathbf{61.3}_{\pm 2.7}$	$\mathbf{62.0}_{\pm 2.1}$	$39.3_{\pm 2.5}$	$\mathbf{59.9}_{\pm 2.4}$
Conditional	0.0	0.0	$\mathbf{52.3}_{\pm 3.5}$	$45.8_{\pm 4.3}$	$17.7_{\pm 2.3}$	$\mathbf{37.9}_{\pm 5.8}$
Rearrangement	0.0	0.0	$\mathbf{54.0}_{\pm 3.5}$	$\mathbf{53.0}_{\pm 3.5}$	$20.3_{\pm 2.7}$	$\mathbf{48.8}_{\pm 3.6}$
Average	0.0	8.1	$\mathbf{55.9}_{\pm 1.9}$	$\mathbf{53.6}_{\pm 2.0}$	$25.8_{\pm 1.4}$	$\mathbf{48.9}_{\pm 2.4}$

Table 1: Performance on navigation tasks and block stacking problems with a Kuka robot. We report normalized cumulative rewards, showing the mean and standard errors over 100 episodes. Results consistent with the best results within the errors are bold. BCQ and CQL results are taken from Janner et al. (2022); for Diffuser, we show our reproduction using their codebase. **Left**: Models trained on the standard datasets. **Right**: SO(3) generalization experiments, with training data restricted to specific spatial orientations such that the agent encounters previously unseen states at test time.

In our remaining experiments, the agent controls a simulated Kuka robotic arm interacting with four blocks on a table. Following Janner et al. (2022), we consider three different tasks: an unconditional block stacking task, a conditional block stacking task where the stacking order is specified, and a rearrangement problem, in which the stacking order has to be changed in a particular way. For both environments, we generate an offline trajectory dataset of roughly 10^4 trajectories. We outline the Kuka state representation and training method in detail in Appendix C.

Algorithms. We train our EDGI on the offline dataset and use conditional sampling to plan the next actions. For the conditional and rearrangement tasks in the Kuka environment, we also use classifier guidance following Janner et al. (2022). As our main baseline, we compare our results to the (non-equivariant) Diffuser model (Janner et al., 2022). We also compare two model-based RL baselines reported by (Janner et al., 2022), BCQ (Fujimoto et al., 2019) and CQL (Kumar et al., 2020).

Task performance. We report the results on both navigation and object tasks in Tab. 1. For each environment, we evaluate 100 episodes and report the average reward and standard error for each method. In the navigation task, the baseline diffuser fails to solve the problem, even after substantially increasing the model's capacity compared to the hyperparemeters used in Janner et al. (2022). EDGI achieves a substantially better performance. On the Kuka environment, we find that EDGI achieves rewards comparable with the original Diffuser model within the error bars and both methods clearly outperform the BCQ and CQL baselines.

Sample efficiency. Next, we study the sample efficiency by training EDGI and Diffuser models on small subsets of the Kuka training set. The results in Fig. 3 are not unambiguous, but show that our EDGI model achieves reasonable rewards even when training with only on 0.1% of the training data, while the baseline Diffuser struggles in this setting. This provides evidence for the benefits of the inductive bias of equivariant models and matches similar observations in other works for using symmetries in an RL context (van der Pol et al., 2020; Walters et al., 2020; Mondal et al., 2021; Rezaei-Shoshtari et al., 2022; Deac et al., 2023).

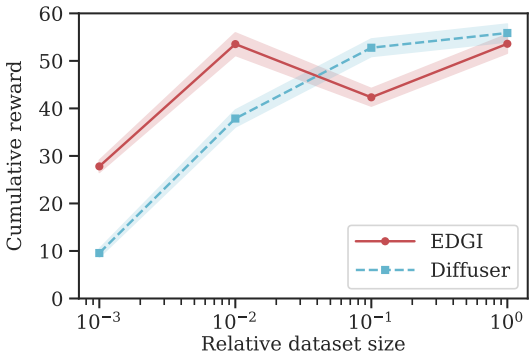

Figure 3: Dataset size versus avg. reward for EDGI and Diffuser on the three Kuka environment tasks.

Group generalization. Finally, we demonstrate that equivariance improves generalization across the SO(3) symmetry group. On both environments, we train EDGI and Diffuser models on restricted offline datasets in which all trajectories are oriented in a particular way. In particular, in the navigation environment, we only use training data that navigates towards a goal location with $x = 0$. In the robotic manipulation tasks, we only use training trajectories where the red block is in a position with $x = 0$ at the beginning of the episode. We test all agents on the original environment, where they encounter goal positions and block configurations unseen during training. We show results for these experiments in Tab. 1. The original Diffuser performs substantially worse, showing its limited

capabilities to generalize to the new setting. In contrast, the performance of EDGI is robust to this domain shift, confirming that equivariance helps in generalizing across the symmetry group.

5 Related work

Our work builds on two foundational lines of research: framing planning as a generative modeling problem and equivariant deep learning. The closest work to ours is the original Diffuser paper (Janner et al., 2022) which we used as a baseline. Concurrent to our work, Diffuser was extended by Ajay et al. (2022) who used a separate inverse dynamics model and classifier-free guidance. The key novelty of our work is that we make this approach aware of the symmetry structure of planning problems through a new $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$-equivariant denoising network yielding an invariant distribution over trajectories while allowing for soft symmetry breaking as dictated by the task.

Equivariant deep learning. Baking in symmetries into deep learning architectures was first studied in the work of (Cohen and Welling, 2016a) for geometric transformations, and the DeepSet architecture for permutations (Zaheer et al., 2017). Followup work to group convolutional networks focused on both spherical geometry (Cohen et al., 2018) and building kernels using irreducible group representations (Cohen and Welling, 2016b; Weiler and Cesa, 2019; Cesa et al., 2021). For symmetries of the 3D space—i. e. subgroups of $\mathrm{E}(3)$—a dominant paradigm is to use the message passing framework (Gilmer et al., 2017) along with geometric quantities like positions, velocities, and relative angles (Satorras et al., 2021; Schütt et al., 2021; Batatia et al., 2022).

Equivariance in RL. The role of symmetries has also been explored in reinforcement learning problems with a body of work focusing on symmetries of the joint state-action space of an MDP (van der Pol et al., 2020; Walters et al., 2020; Mondal et al., 2021; Muglich et al., 2022; Wang and Walters, 2022; Wang et al., 2022; Cetin et al., 2022; Rezaei-Shoshtari et al., 2022). More recently, model-based approaches—like our proposed EDGI—have also benefited from increased data efficiency through the use of symmetries of the environment (Deac et al., 2023).

Equivariant generative models. Early efforts in learning invariant densities using generative models utilized the continuous normalizing flow (CNF) framework. A variety of works imbued symmetries by designing equivariant vector fields (Köhler et al., 2020; Rezende and Mohamed, 2015; Bose and Kobyzev, 2021). As flow-based models enjoy exact density estimation, their application is a natural fit for applications in theoretical physics (Boyda et al., 2020; Kanwar et al., 2020) and modeling equivariant densities on manifolds (Katsman et al., 2021). Other promising approaches to CNFs include equivariant score matching (De Bortoli et al., 2022) and diffusion models (Hoogeboom et al., 2022; Xu et al., 2022; Igashov et al., 2022). Our proposed EDGI model extends the latter category to the product group $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$ and increases flexibility with respect to the data representations.

6 Discussion

Embodied agents often solve tasks that are structured through the spatial, temporal, or permutation symmetries of our 3D world. Taking this structure into account in the design of reinforcement learning (RL) and planning algorithms can improve sample efficiency and generalization—notorious weaknesses of RL algorithms. We introduced EDGI, an equivariant planning algorithm that operates as conditional sampling in a generative model. The main innovation is a new diffusion model that is equivariant with respect to the symmetry group $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$ of spatial, temporal, and object permutation symmetries. Beyond this concrete architecture, our work presents a general blueprint for the construction of networks that are equivariant with respect to a product group and support multiple representations in the inputs and outputs. Integrating this equivariant diffusion model into a planning algorithm allows us to model an invariant base density, but still solve non-invariant tasks through task-specific soft symmetry breaking. We demonstrated the performance, sample efficiency, and robustness of EDGI on object manipulation and navigation tasks.

While our work shows encouraging results, training and planning are currently expensive. Progress on these issues can come both from more efficient layers in the architecture of the denoising model as well as from switching to recent continuous-time diffusion methods with accelerated sampling.

REFERENCES

A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision-making? *International Conference on Learning Representations*, 2022. (Cited on page 8)

I. Batatia, D. P. Kovács, G. N. Simm, C. Ortner, and G. Csányi. Mace: Higher order equivariant message passing neural networks for fast and accurate force fields. *arXiv preprint arXiv:2206.07697*, 2022. (Cited on page 8)

A. J. Bose and I. Kobyzev. Equivariant finite normalizing flows. *arXiv preprint arXiv:2110.08649*, 2021. (Cited on pages 2, 4, and 8)

D. Boyda, G. Kanwar, S. Racanière, D. J. Rezende, M. S. Albergo, K. Cranmer, D. C. Hackett, and P. E. Shanahan. Sampling using $su(n)$ gauge equivariant flows. *arXiv preprint arXiv:2008.05456*, 2020. (Cited on page 8)

G. Cesa, L. Lang, and M. Weiler. A program to build e (n)-equivariant steerable cnns. In *International Conference on Learning Representations*, 2021. (Cited on page 8)

E. Cetin, B. Chamberlain, M. Bronstein, and J. J. Hunt. Hyperbolic deep reinforcement learning. *arXiv [cs.LG]*, Oct. 2022. (Cited on pages 2 and 8)

T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016a. (Cited on page 8)

T. S. Cohen and M. Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016b. (Cited on page 8)

T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018. (Cited on page 8)

E. Coumans and Y. Bai. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019. (Cited on page 13)

V. De Bortoli, E. Mathieu, M. J. Hutchinson, J. Thornton, Y. W. Teh, and A. Doucet. Riemannian Score-Based generative modelling. Oct. 2022. URL https://openreview.net/pdf?id=oDRQGo8I7P. (Cited on pages 8 and 13)

A. Deac, T. Weber, and G. Papamakarios. Equivariant MuZero. Feb. 2023. (Cited on pages 2, 7, and 8)

P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. (Cited on page 3)

S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018. (Cited on page 13)

S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019. (Cited on page 7)

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017. (Cited on page 8)

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. (Cited on pages 3 and 6)

G. Z. Holland, E. J. Talvitie, and M. Bowling. The effect of planning shape on dyna-style planning in high-dimensional state spaces. *arXiv preprint arXiv:1806.01825*, 2018. (Cited on page 1)

E. Hoogeboom, V. G. Satorras, C. Vignac, and M. Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pages 8867–8887. PMLR, 2022. (Cited on page 8)

C.-W. Huang, M. Aghajohari, A. J. Bose, P. Panangaden, and A. Courville. Riemannian diffusion models. Aug. 2022. URL http://arxiv.org/abs/2208.07949. (Cited on page 13)

I. Igashov, H. Stärk, C. Vignac, V. G. Satorras, P. Frossard, M. Welling, M. Bronstein, and B. Correia. Equivariant 3d-conditional diffusion models for molecular linker design. *arXiv preprint arXiv:2210.05274*, 2022. (Cited on page 8)

M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022. (Cited on pages 1, 2, 3, 6, 7, 8, 12, and 13)

G. Kanwar, M. S. Albergo, D. Boyda, K. Cranmer, D. C. Hackett, S. Racaniere, D. J. Rezende, and P. E. Shanahan. Equivariant flow-based sampling for lattice gauge theory. *Physical Review Letters*, 125(12):121601, 2020. (Cited on page 8)

I. Katsman, A. Lou, D. Lim, Q. Jiang, S.-N. Lim, and C. De Sa. Equivariant manifold flows. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021. (Cited on page 8)

J. Köhler, L. Klein, and F. Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International conference on machine learning*, pages 5361–5370. PMLR, 2020. (Cited on pages 2, 4, and 8)

A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020. (Cited on page 7)

A. K. Mondal, V. Jain, K. Siddiqi, and S. Ravanbakhsh. EqR: Equivariant representations for Data-Efficient reinforcement learning. Nov. 2021. (Cited on pages 2, 7, and 8)

D. Muglich, C. S. de Witt, E. van der Pol, S. Whiteson, and J. Foerster. Equivariant networks for Zero-Shot coordination. Oct. 2022. (Cited on pages 2 and 8)

G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1): 2617–2680, 2021. (Cited on pages 2 and 4)

A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. 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. (Cited on page 13)

S. Rezaei-Shoshtari, R. Zhao, P. Panangaden, D. Meger, and D. Precup. Continuous mdp homomorphisms and homomorphic policy gradient. *arXiv preprint arXiv:2209.07364*, 2022. (Cited on pages 2, 7, and 8)

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd international conference on Machine learning*. ACM, 2015. (Cited on page 8)

V. G. Satorras, E. Hoogeboom, and M. Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021. (Cited on page 8)

K. Schütt, O. Unke, and M. Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pages 9377–9388. PMLR, 2021. (Cited on page 8)

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. (Cited on page 2)

E. van der Pol, T. Kipf, F. A. Oliehoek, and others. Plannable approximations to MDP homomorphisms: Equivariance under actions. *arXiv preprint arXiv*, 2020. (Cited on pages 2, 7, and 8)

S. Villar, D. W. Hogg, K. Storey-Fisher, W. Yao, and B. Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. *Advances in Neural Information Processing Systems*, 34:28848–28863, 2021. (Cited on page 6)

R. Walters, J. Li, and R. Yu. Trajectory prediction using equivariant continuous convolution. Oct. 2020. (Cited on pages 2, 7, and 8)

D. Wang and R. Walters. So (2) equivariant reinforcement learning. In *International Conference on Learning Representations*, 2022. (Cited on pages 2 and 8)

D. Wang, M. Jia, X. Zhu, R. Walters, and R. Platt. On-robot learning with equivariant models. In *Conference on robot learning*, 2022. (Cited on pages 2 and 8)

M. Weiler and G. Cesa. General e (2)-equivariant steerable cnns. *Advances in Neural Information Processing Systems*, 32, 2019. (Cited on page 8)

M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022. (Cited on page 8)

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017. (Cited on page 8)

A ARCHITECTURE DETAILS

On a high level, EDGI follows Diffuser (Janner et al., 2022). In the following, we will describe the key difference: our $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$-equivariant architecture for the diffusion model.

Overall architecture. We illustrate the architecture in Fig. 2. After converting the input data in our internal representation (see Sec. 3.2), the data is processed with an equivariant U-net with four levels. At each level, we process the hidden state with two residual standard blocks, before downsampling (in the downward pass) or upsampling (in the upward pass).

Residual standard block. The main processing unit of our architecture processes the current hidden state with an equivariant block consisting of a temporal layer, an object layer, a normalization layer, and a geometric layer. In parallel, the context information (an embedding of diffusion time and a conditioning mask) is processed with a context block. The hidden state is added to the output of the context block and processes with another equivariant block. Finally, we process the data with a linear attention layer over time. This whole pipeline consists of an equivariant block, a context block, and another equivariant block is residual (the inputs are added to the outputs).

Temporal layers. Temporal layers consist of one-dimensional convolutions without bias along the time dimension. We use a kernel size of 5.

Object layers. Object layers consist of multi-head self-attention over the object dimension. We use four heads. Keys, queries, and values are constructed as bias-free linear transformations of the inputs to a 32-channel vector. Given inputs w_{toc}, the permutation layer computes

$$\mathbf{K}_{toc} = \sum_{c'} \mathbf{W}^K_{cc'} w_{toc}\,, \quad \mathbf{Q}_{toc} = \sum_{c'} \mathbf{W}^Q_{cc'} w_{toc}\,, \quad \mathbf{V}_{toc} = \sum_{c'} \mathbf{W}^V_{cc'} w_{toc}\,,$$
$$w'_{t\cdot} \propto \sum_{o'} \mathrm{softmax}_{o'}(\mathbf{Q}_{toc} \cdot \mathbf{K}_{to'c})\mathbf{V}_{to'c},$$

with learnable weight matrices $\mathbf{W}^{K,V,Q}$.

Normalization layers. We use a simple equivariant normalization layer that for each batch element rescales the entire tensor w_{toc} to unit variance. This is essentially an equivariant version of LayerNorm. The difference is that our normalization layer does not shift the inputs to zero means, as that would break equivariance with respect to $\mathrm{SO}(3)$.

Geometric layers. In the geometric layers, the input state is split into scalar and vector components. The vector components are linearly transformed to reduce the number of channels to 16. We then construct all $\mathrm{SO}(3)$ invariants from these 16 vectors by taking pairwise inner products and concatenating them with the scalar inputs. This set of scalars is processed with two MLPs, each consisting of two hidden layers and ReLU nonlinearities. The MLPs output the scalar outputs and coefficients for a linear map between the vector inputs and the vector outputs, respectively. Finally, there is a residual connection that adds the scalar and vector inputs to the outputs.

Linear attention over time. To match the architecture used by Janner et al. (2022) as closely as possible, we follow their choice of adding another residual linear attention over time at the end of each level in the U-net. We make the linear attention mechanism equivariant by computing the attention weights as

Context blocks. The embeddings of diffusion time and conditioning information are processed with a Mish nonlinearity and a linear layer, like in Janner et al. (2022). Finally, we embed them in our internal representation by zero-padding the resulting tensor.

Upsampling and downsampling. During the downsampling path, there is a final temporal layer that implements temporal downsampling and increases the number of channels by a factor of two. Conversely, during the upsampling path, we use a temporal layer for temporal upsampling and a reduction of the number of channels.

B NAVIGATION EXPERIMENTS

We introduce a new navigation environment. The scene consists of a spherical agent navigating a plane populated with a goal state and $n = 10$ spherical obstacles. At the beginning of every episode,

the agent position, agent velocity, obstacle positions, and goal position are initialized randomly (in a rotation-invariant way). We simulate the environment dynamics with PyBullet (Coumans and Bai, 2016–2019).

Offline dataset. To obtain expert trajectories, we train a TD3 (Fujimoto et al., 2018) agent in the implementation by Raffin et al. (2021) for 10^7 steps with default hyperparameters on this environment. We generate 10^4 trajectories for our offline dataset.

State. The state contains the agent position, agent velocity, goal position, and obstacle positions.

Actions. The action space is two-dimensional and specifies a force acting on the agent.

Rewards. At each time step, the agent receives a reward equal to the negative Euclidean distance to the goal state. In addition, a penalty of -0.1 is added to the reward if the agent touches any of the obstacles. Finally, there is an additional control cost equal to -10^3 times the force acting on the agent. We affinely normalize the rewards such that a normalized reward of 0 corresponds to that achieved by a random policy and a normalized reward of 100 corresponds to the expert policy.

C KUKA EXPERIMENTS

We use the object manipulation environments and tasks from Janner et al. (2022), please see that work for details on the environment. In our experiments, we consider three tasks: unconditional stacking, conditional stacking, and block rearrangement. For a fair comparison, we re-implement the Diffuser algorithm while making bug fixes in the codebase of Janner et al. (2022), which mainly included properly resetting the environment.

C.1 STATE

We experiment with two parameterizations of the Kuka environment state. For the Diffuser baseline, we use the original 48-dimensional parameterization from Janner et al. (2022).

For our EDGI, we need to parameterize the system in terms of $\mathrm{SE}(3) \times \mathbb{Z} \times \mathbb{S}_n$ representations. We, therefore, describe the robot and block orientations with $\mathrm{SO}(3)$ vectors as follows. Originally, the robot state is specified through a collection of joint angles. One of these encodes the rotation of the base along the vertical z-axis. We choose to represent this angle as a ρ_1 vector in the xy-plane. In addition, we add the gravity direction (the z-axis itself) as another ρ_1 vector, which is also the normal direction of the table on which the objects rest. Combined, these vectors define the pose of the base of the robot arm. Rotating gravity direction, and the robot and object pose by $\mathrm{SO}(3)$ can be interpreted as a passive coordinate transformation, or as an active rotation of the entire scene, including gravity. As the laws of physics are invariant to this transformation, this is a valid symmetry of the problem.

The n objects can be translated and rotated. Their pose is thus given by a translation $t \in \mathbb{R}^3$ and rotation in $r \in \mathrm{SO}(3)$ relative to a reference pose. The translation transforms by a global rotation $g \in \mathrm{SO}(3)$ as a vector via representation ρ_1. The rotational pose transforms by left multiplication $r \mapsto gr$. The $\mathrm{SO}(3)$ pose is not a Euclidean space, but a non-trivial manifold. Even though diffusion on manifolds is possible De Bortoli et al. (2022); Huang et al. (2022), we simplify the problem by embedding the pose in a Euclidean space. This is done by picking the first two columns of the pose rotation matrix $r \in \mathrm{SO}(3)$. These columns each transform again as a vector with representation ρ_1. This forms an equivariant embedding $\iota : \mathrm{SO}(3) \hookrightarrow \mathbb{R}^{2\times 3}$, whose image is two orthogonal 3-vectors of unit norm. Via the Gram-Schmidt procedure, we can define an equivariant map $\pi : \mathbb{R}^{2\times 3} \to \mathrm{SO}(3)$ (defined almost everywhere), that is a left inverse to the embedding: $\pi \circ \iota = \mathrm{id}_{\mathrm{SO}(3)}$. Combining with the translation, the roto-translational pose of each object is thus embedded as three ρ_1 vectors.

We also tested the performance of the baseline Diffuser method on this reparameterization of the state but found worse results.

C.2 HYPERPARAMETERS

We also follow the choices of Janner et al. (2022), except that we experiment with a linear noise schedule as an alternative to the cosine schedule they use. For each model and each dataset, we train the diffusion model with both noise schedules and report the better of the two results.